TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN HỆ THỐNG THÔNG TIN
-----*-----**

BÀI GIẢNG

NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

TÊN HỌC PHẦN : **CÔNG NGHỆ PHẦN MỀM**
MÃ HỌC PHẦN : **17404**
TRÌNH ĐỘ ĐÀO TẠO : **ĐẠI HỌC CHÍNH QUY**
DÙNG CHO SV NGÀNH : **CÔNG NGHỆ THÔNG TIN**

HẢI PHÒNG - 2011

MỤC LỤC

Tên học phần: Nhập môn Công nghệ phần mềm **Loại học phần**: 1
Bộ môn phụ trách giảng dạy: Hệ thống Thông tin **Khoa phụ trách**: CNTT.
Mã học phần: 17404 **Tổng số TC**: 2

Tổng số tiết	Lý thuyết	Thực hành/Xemina	Tự học	Bài tập lớn	Đồ án môn học
30	30	0	0	không	không

Học phần học trước: Không yêu cầu.
Học phần tiên quyết: Không yêu cầu.
Học phần song song: Không yêu cầu.
Mục tiêu của học phần:
Cung cấp cho sinh viên những kiến thức cơ bản về công nghệ phần mềm.
Nội dung chủ yếu:
Giới thiệu về công nghệ phần mềm; Các mô hình phát triển phần mềm; Lượng giá dự án phần mềm; Khảo sát và phân tích yêu cầu; Mô hình hóa hệ thống; Thiết kế hệ thống; Kiểm thử phần mềm.
Nội dung chi tiết:

TÊN CHƯƠNG MỤC	PHÂN PHỐI SỐ TIẾT				
	TS	LT	TH	BT	KT
Chương 1: Giới thiệu	**2**	**2**			
1.1. Khái niệm phần mềm					
1.2. Các đặc điểm của phần mềm					
1.3. Các ứng dụng của phần mềm					
1.4. Giới thiệu về Công nghệ phần mềm (Software engineering)					
Chương 2: Các mô hình phát triển phần mềm	**6**	**6**			
2.1. Mô hình thác nước (Waterfall model)					
2.2. Mô hình nguyên mẫu (Prototyping model)					
2.3. Mô hình phát triển nhanh (RAD model)					
2.4. Mô hình tăng trưởng (Incremental model)					
2.5. Mô hình xoắn ốc (Spiral model)					
2.6. Các mô hình hiện đại (Fourth generation techniques)					
Chương 3: Khảo sát và phân tích yêu cầu	**4**	**4**			
3.1. Thu thập yêu cầu (Requirements elicitation)					
3.2. Phân tích yêu cầu (Requirements analysis)					
3.3. Đặc tả yêu cầu (Requirements specification)					
3.4. Xét duyệt yêu cầu (Requirements validation)					
Chương 4: Mô hình hóa hệ thống	**4**	**4**			
4.1. Mô hình hóa dữ liệu (Data modeling)					
4.2. Mô hình hóa chức năng (Functional modeling)					
4.3. Mô hình hóa luồng thông tin (Information flow modeling)					
Chương 5: Thiết kế hệ thống	**4**	**4**			
5.1. Quá trình thiết kế (Design process)					
5.2. Các nguyên tắc thiết kế (Design principles)					
5.3. Các khái niệm trong thiết kế phần mềm (Design concepts)					
Chương 6: Kiểm thử phần mềm	**6**	**6**			
6.1. Mục đích (Testing objectives)					
6.2. Nguyên tắc kiểm thử (Testing principles)					
6.3. Kiểm thử theo đường cơ bản (Basic path)					
6.4. Kiểm thử theo phân vùng tương đương (Equivalence partitioning)					

TÊN CHƯƠNG MỤC	PHÂN PHỐI SỐ TIẾT				
	TS	LT	TH	BT	KT
6.5. Kiểm thử theo giá trị biên (Boundary value analysis)					
6.6. Các mức độ kiểm thử (Testing strategy)					

Nhiệm vụ của sinh viên:
Tham dự các buổi học lý thuyết và thực hành, làm các bài tập được giao, làm các bài thi giữa kỳ và bài thi kết thúc học phần theo đúng quy định.

Tài liệu học tập:

1. Roger S. Pressman, *Software Engineering- A practitioner's Approach*, 6th edition, McGraw-Hill.
2. Sommerville, *Software Engineering*, 7th edition, Pearson education.
3. Nguyễn Xuân Huy, *Giáo trình công nghệ phần mềm*, NXB Trường ĐHBK Hà Nội, 1996.

Hình thức và tiêu chuẩn đánh giá sinh viên:

- Hình thức thi: tự luận hoặc trắc nghiệm.
- Tiêu chuẩn đánh giá sinh viên: căn cứ vào sự tham gia học tập của sinh viên trong các buổi học lý thuyết và thực hành, kết quả làm các bài tập được giao, kết quả của các bài thi giữa học phần và bài thi kết thúc học phần.

Thang điểm: Thang điểm chữ A, B, C, D, F.
Điểm đánh giá học phần: $Z = 0,2X + 0,8Y$.

Bài giảng này là tài liệu **chính thức và thống nhất** của Bộ môn Hệ thống Thông tin, Khoa Công nghệ Thông tin và được dùng để giảng dạy cho sinh viên.

Ngày phê duyệt: / /

Trưởng Bộ môn

Là một tập hợp các chương trình được viết để phục vụ cho các chương trình khác. Chương trình này xử lý các thông tin phức tạp nhưng xác định cấp thấp, tạo môi trường hoạt động (trình biên dịch, trình soạn thảo, quản lý tệp tin, …).

Các chương trình này đặc trưng bởi tương tác chủ yếu với phần cứng máy tính, phục vụ nhiều người dùng, có cấu trúc dữ liệu phức tạp và nhiều giao diện ngoài.

Nhóm 2: *Phần mềm thời gian thực.*

Là phần mềm điều phối hoặc phân tích hay kiểm soát các sự kiện thế giới thực ngay khi chúng xuất hiện.

Phần mềm thời gian thực bao gồm các yếu tố:

- Một thành phần thu thập dữ liệu để thu và định dạng thông tin từ bên ngoài.
- Một thành phần phân tích để biến đổi thông tin theo yêu cầu của ứng dụng.
- Một thành phần kiểm soát hoặc đưa ra các đáp ứng cho môi trường ngoài.
- Một thành phần điều phối để điều hoà các thành phần khác sao cho có thể duy trì việc đáp ứng thời gian thực.

Hệ thống thời gian thực phải đáp ứng được những ràng buộc thời gian chặt chẽ.

Nhóm 3: *Phần mềm nghiệp vụ.*

Ngày nay, xử lý thông tin nghiệp vụ là lĩnh vự ứng dụng phần mềm lớn nhất. Phần mềm loại này phục vụ cho các hệ thống rời rạc: *hệ thông tin quản lý*. Các ứng dụng phần mềm nghiệp vụ còn bao gồm cả tính toán tương tác (như xử lý các giao tác cho các điểm bán hàng) ngoài ứng dụng xử lý dữ liệu.

Nhóm 4: *Phần mềm khoa học công nghệ.*

Phần mềm này được đặc trưng bởi các thuật toán. Phần mềm tạo ra một ứng dụng mới, thiết kế có máy tính trợ giúp (computer aided of design - CAD), có chú ý đến các đặc trưng thời gian thực và phần mềm hệ thống.

Nhóm 5: *Phần mềm nhúng.*

Nằm trong bộ nhớ chỉ đọc và được dùng để điều khiển các sản phẩm và hệ thống cho người dùng và thị trường công nghiệp. Có thể thực hiện các chức năng đơn giản nhưng mang tính chuyên biệt (huyền bí), ví dụ: điều khiển chức năng cho lò vi sóng; hay có thể đưa ra các khả năng điều khiển và vận hành (chức năng số hoá ở ô-tô, kiểm soát xăng, biểu thị bảng đồng hồ, các hệ thống phanh…).

Nhóm 6: *Phần mềm máy tính cá nhân.*

Loại phần mềm này bùng nổ trong hơn thập kỷ vừa qua (như xử lý văn bản, trang tính, đồ hoạ, quản trị cơ sở dữ liệu). Hiện nay được tiếp tục phát triển biểu thị giao diện người máy, tạo ra sự thân thiện, dễ sử dụng cho người dùng.

Nhóm 7: *Phần mềm trí tuệ nhân tạo.*

Thiết kế. Thiết kế phần mềm thực tế là một tiến trình nhiều bước tập trung vào bốn thuộc tính phân biệt của chương trình: cấu trúc dữ liệu, kiến trúc phần mềm, biểu diễn giao diện và chi tiết thủ tục (thuật toán). Tiến trình thiết kế dịch các yêu cầu thành một biểu diễn của phần mềm có thể được định giá về chất lượng trước khi giai đoạn mã hoá bắt đầu. Giống như các yêu cầu, việc thiết kế phải được lập tư liệu và trở thành một phần của cấu hình phần mềm.

Sinh mã. Thiết kế phải được dịch thành dạng máy đọc được. Bước mã hoá thực hiện nhiệm vụ này. Nếu thiết kế được thực hiện theo một cách chi tiết thì việc sinh mã có thể được thực hiện một cách máy móc.

Kiểm thử. Một khi mã đã được sinh ra thì việc kiểm thử chương trình bắt đầu. Tiến trình kiểm thử hội tụ vào nội bộ logic của phần mềm, đảm bảo rằng tất cả các câu lệnh đều được kiểm thử, và vào bên ngoài chức năng; tức là tiến hành các kiểm thử để làm lộ ra các lỗi và đảm bảo những cái vào đã định sẽ tạo ra kết quả thống nhất với kết quả muốn có.

Vận hành và bảo trì. Phần mềm chắc chắn sẽ phải trải qua những thay đổi sau khi nó được bàn giao cho khách hàng (một ngoại lệ có thể là những phần mềm nhúng). Thay đổi sẽ xuất hiện bởi vì gặp phải lỗi, bởi vì phần mềm phải thích ứng với những thay đổi trong môi trường bên ngoài (chẳng hạn như sự thay đổi do hệ điều hành mới hay thiết bị ngoại vi mới), hay bởi vì khách hàng yêu cầu nâng cao chức năng hay hiệu năng. Việc bảo trì phần mềm phải áp dụng lại các bước vòng đời nói trên cho chương trình hiện tại chứ không phải chương trình mới.

Mô hình tuần tự tuyến tính là mô hình cũ nhất và được sử dụng rộng rãi nhất cho kĩ nghệ phần mềm. Tuy nhiên, những chỉ trích về mô hình này đã làm cho những người ủng hộ nó tích cực phải đặt vấn đề về tính hiệu quả của nó. Một số các vấn đề thỉnh thoảng gặp phải khi dùng mô hình tuần tự tuyến tính này là:

Các dự án thực hiếm khi tuân theo dòng chảy tuần tự mà mô hình đề nghị. Mặc dầu mô hình tuyến tính có thể cho phép lặp, nhưng điều đó chỉ làm gián tiếp. Kết quả là những thay đổi có thể gây ra lẫn lộn khi tổ dự án tiến hành.

Khách hàng thường khó phát biểu mọi yêu cầu một cách tường minh. Mô hình tuần tự tuyến tính đòi hỏi điều này và thường khó thích hợp với sự bất trắc tự nhiên tồn tại vào lúc đầu của nhiều dự án.

Khách hàng phải kiên nhẫn. Bản làm việc được của chương trình chỉ có được vào lúc cuối của thời gian dự án. Một sai lầm ngớ ngẩn, nếu đến khi có chương trình làm việc mới phát hiện ra, có thể sẽ là một thảm hoạ.

Trong một phân tích thú vị về các dự án hiện tại, Brada thấy rằng bản chất tuyến tính của vòng đời cổ điển dẫn tới "các trạng thái nghẽn" mà trong đó một số thành viên tổ dự án phải đợi cho các thành viên khác của tổ hoàn thành các nhiệm vụ phụ thuộc. Trong thực tế, thời gian mất cho việc chờ đợi có thể vượt quá thời gian dành cho công việc sản xuất. Trạng thái nghẽn có khuynh hướng phổ biến vào lúc đầu và cuối của tiến trình tuần tự tuyến tính.

đã có hay áp dụng các công cụ (như bộ sinh báo cáo, bộ quản lí cửa sổ, v.v..) để nhanh chóng sinh ra chương trình làm việc.

Nhưng chúng ta nghĩ về bản mẫu thế nào khi nó được dùng cho mục đích được nêu trên? Brook đã nêu ra câu trả lời:

Trong hầu hết các dự án, hệ thống đầu tiên hiếm khi sử dụng được. Nó có thể là quá chậm, quá lớn, cồng kềnh trong sử dụng hay tất cả những nhược điểm này. Không có cách nào khác là bắt đầu lại, đau đớn nhưng tinh khôn hơn, và xây dựng một phiên bản được thiết kế lại trong đó những vấn đề này đã được giải quyết... Khi một khái niệm hệ thống mới hay một kĩ nghệ mới được dùng, người ta phải xây dựng một hệ thống để rồi vứt đi, cho dù việc lập kế hoạch được thực hiện chu đáo nhất thì nó cũng không thể bao quát hết để chạy đúng được ngay lần đầu. Do đó câu hỏi quản lí không phải là liệu chúng ta có nên xây dựng một hệ thống thử nghiệm và rồi vứt nó đi hay không. Bạn sẽ làm như vậy. Câu hỏi duy nhất là liệu nên lập kế hoạch trước để xây dựng một cái vứt đi hay để hứa hẹn bàn giao cái vứt đi đó cho khách hàng...

Bản mẫu có thể phục vụ như "hệ đầu tiên" - cái mà Brook lưu ý chúng ta nên vứt đi. Nhưng điều này có thể là một cách nhìn lí tưởng hoá. Giống như mô hình tuyến tính tuần tự (thác nước), việc làm bản mẫu tựa như một mô hình cho kĩ nghệ phần mềm có thể trở thành có vấn đề bởi những lí do sau:

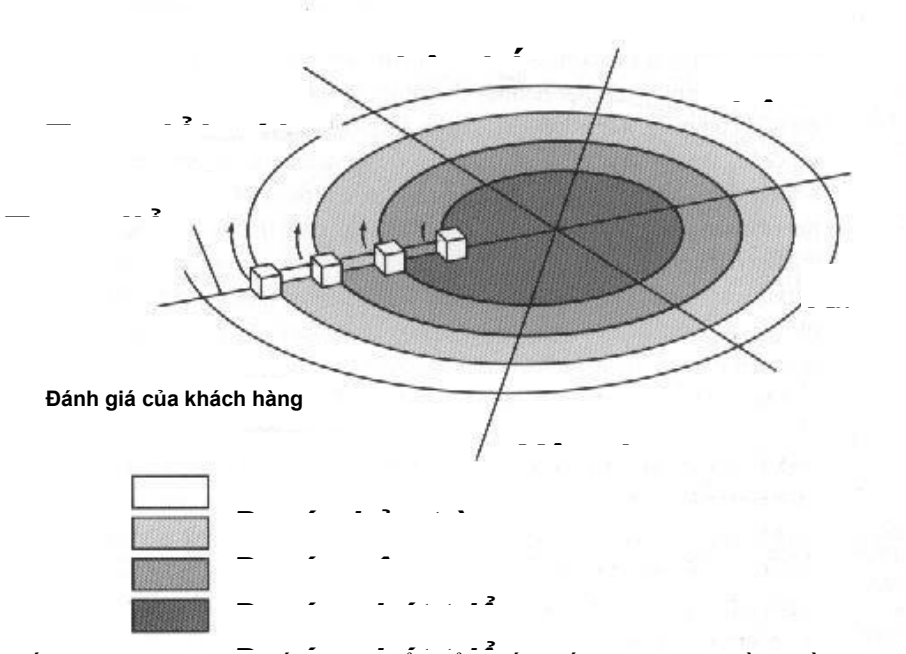
1. Khách hàng thấy được cái dường như là phiên bản làm việc của phần mềm mà không biết rằng bản mẫu được gắn lại "bằng kẹo cao su và dây gói hàng", không biết rằng trong khi xô đẩy để cho nó làm việc thì chẳng ai xem xét tới chất lượng phần mềm tổng thể hay tính bảo trì thời gian dài. Khi được thông báo rằng sản phẩm phải được xây dựng lại để cho có thể đạt tới mức độ chất lượng cao, thì khách hàng kêu trời và đòi hỏi rằng "phải ít sửa chữa" để làm bản mẫu thành sản phẩm làm việc. Rất thường là việc quản lí phát triển phần mềm bị buông lỏng.
2. Người phát triển thường hay thoả hiệp cài đặt để có được bản mẫu làm việc nhanh chóng. Hệ điều hành hay ngôn ngữ lập trình không thích hợp có thể được dùng đơn giản bởi vì nó có sẵn và đã biết; một thuật toán không hiệu quả có thể được cài đặt đơn giản để chứng tỏ khả năng. Sau một thời gian, người phát triển mới có thể trở nên quen thuộc với những chọn lựa này và quên mất mọi lí do tại sao chúng lại không thích hợp. Việc chọn lựa không được theo lí tưởng bây giờ lại trở thành một phần tích hợp của hệ thống.

Mặc dầu vấn đề có thể xuất hiện, việc làm bản mẫu có thể là một mô hình hiệu quả cho kĩ nghệ phần mềm. Chìa khoá là định nghĩa ra các qui tắc của trò chơi từ ngay lúc bắt đầu; tức là khách hàng và người phát triển phải cùng đồng ý rằng bản mẫu được xây dựng để phục vụ làm cơ chế xác định yêu cầu. Thế rồi nó phải bị bỏ đi (ít nhất cũng một phần) và phần mềm thực tại được đưa vào kĩ nghệ với con mắt hướng về chất lượng và tính bảo trì được.

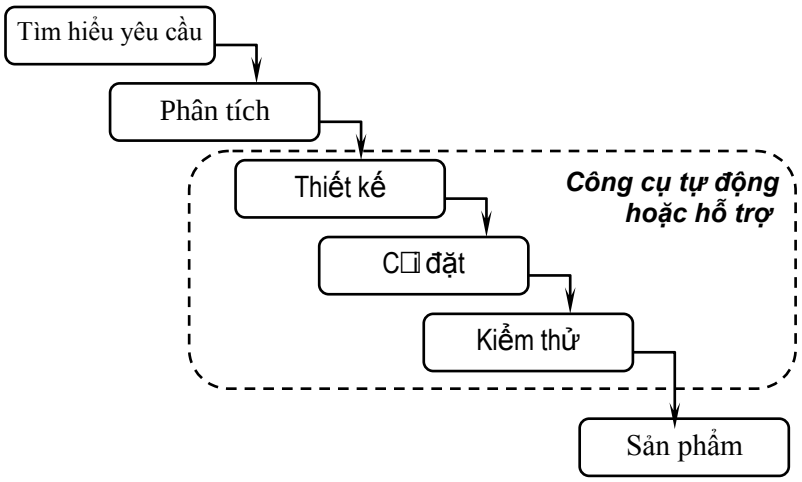
6. ***Đánh giá của khánh hàng*** - nhiệm vụ đòi hỏi thu được phản hồi của khách hàng dựa trên đánh giá về biểu diễn phần mềm được tạo ra trong giai đoạn kĩ nghệ và được cài đặt trong giai đoạn cài đặt.

Mỗi một trong các vùng đều được đặt vào một tập các nhiệm vụ, được gọi là *tập nhiệm vụ*, vốn được thích ứng với các đặc trưng của dự án được tiến hành. Với các sự án nhỏ, số các nhiệm vụ công việc và tính hình thức của chúng là thấp. Với các dự án lớn, nhiều căng thẳng hơn, thì mỗi vùng nhiệm vụ lại chứa nhiều nhiệm vụ công việc vốn được xác định để đạt tới mức độ hình thức cao hơn. Trong mọi trường hợp, hoạt động hỗ trợ (như quản lí cấu hình phần mềm và đảm bảo chất lượng phần mềm) - được nêu trong phần sau - sẽ được áp dụng.

Khi tiến trình tiến hoá này bắt đầu, tổ kĩ nghệ phần mềm đi vòng xoắn ốc theo chiều ngược kim đồng hồ, bắt đầu từ trung tâm. Mạch đầu tiên quanh xoắn ốc có thể làm phát sinh việc phát triển đặc tả sản phẩm; các bước tiếp theo quanh xoắn ốc có thể được dùng để phát triển bản mẫu và thế rồi các phiên bản phức tạp dần thêm. Mỗi bước qua vùng lập kế hoạch lại làm nảy sinh việc điều chỉnh kế hoạch dự án. Chi phí và lịch biểu được điều chỉnh dựa trên phản hồi được suy từ đánh giá của khách hàng. Bên cạnh đó, người quản lí dự án điều chỉnh số việc lặp đã lập kế hoạch cần để hoàn chỉnh phần mềm.



Không giống như mô hình tiến trình cổ điển vốn kết thúc khi phần mềm được chuyển giao, mô hình xoắn ốc có thể được thích ứng để áp dụng trong toàn bộ cuộc đời của phần mềm máy tính. Một cái nhìn khác có thể được xem xét bằng việc *kiểm tra trục điểm vào dự án*, như được vẽ trong hình trên. Mỗi hình hộp được đặt theo trục có thể được dùng để biểu diễn cho điểm bắt đầu cho các kiểu dự án khác nhau. "Dự án phát triển khái niệm" bắt đầu tại cốt lõi của xoắn ốc và sẽ tiếp tục (nhiều lần lặp xuất hiện theo con đường xoắn ốc mà vốn gắn với vùng tô đậm trung tâm) cho tới khi việc phát triển khái niệm là đầy đủ. Nếu khái niệm này được phát triển thành một sản phẩm thực tại,



Hình 3: Mô hình kỹ nghệ thứ 4 - 4GT

Hiện tại, một môi trường phát triển phần mềm hỗ trợ cho mô hình 4GT bao gồm một số hay tất cả các công cụ sau:

- Ngôn ngữ phi thủ tục để hỏi đáp cơ sở dữ liệu.
- Bộ sinh báo cáo.
- Bộ thao tác dữ liệu.
- Bộ tương tác và xác định màn hình.
- Bộ sinh chương trình.
- Khả năng đồ hoạ mức cao.
- Khả năng làm trang tính và việc sinh tự động HTML.
- Các ngôn ngữ tương tự được dùng cho việc tạo ra trang Web thông qua việc dùng các công cụ phần mềm tiên tiến.

Ban đầu nhiều trong những công cụ đã được nhắc tới đó đã có sẵn chỉ cho những lĩnh vực ứng dụng rất đặc thù, nhưng ngày nay môi trường 4GT đã được mở rộng để đề cập tới hầy hết các loại ứng dụng phần mềm.

Giống như các mô hình khác, 4GT bắt đầu từ bước thu thập yêu cầu. Một cách lý tưởng, khách hàng sẽ mô tả các yêu cầu và các yêu cầu đó sẽ được dịch trực tiếp thành một bản mẫu vận hành được. Nhưng điều này không thực hiện được. Khách hàng có thể không chắc chắn mình cần gì, có thể có sự mơ hồ trong việc xác định các sự kiện đã biết, có thể không có khả năng hay không sẵn lòng xác định thông tin theo cách thức mà công cụ 4GT có thể giải quyết được. Bởi lí do này, đối thoại khách hàng/ người phát triển được mô tả cho các mô hình tiến trình khác vẫn còn là phần bản chất của cách tiếp cận 4GT.

Với những ứng dụng nhỏ, có thể chuyển trực tiếp từ bước thu thập yêu cầu sang cài đặt bằng cách dùng *ngôn ngữ sinh thế hệ thứ tư phi thủ tục* (4GL) hay một mô hình bao gồm một mạng các biểu tượng đồ hoạ. Tuy nhiên với nỗ lực lớn hơn, cần phải phát triển một chiến lược thiết kế cho hệ

Bên cạnh việc thu thập thông tin, chúng ta cũng cần sử dụng các kỹ thuật định lượng thông tin và biên dịch và ứng dụng đề ra.

Tính chất 1: *Hướng thời gian.*

Tính hướng thời gian của dữ liệu đề cập tới *quá khứ, hiện tại* hoặc các *đòi hỏi tương lai* của ứng dụng đề ra.

Các *dữ liệu quá khứ*, ví dụ, có thể mô tả công việc đã được biến đôit thế nào qua thời gian, các quy định ảnh hưởng thế nào tới nhiệm vụ, vị trí của nó trong tổ chức và nhiệm vụ. Các thông tin quá khứ là chính xác, đầy đủ và xác đáng.

Các *thông tin hiện tại* là các thông tin và cái gì đang xảy ra. Ví dụ thông tin ứng dụng hiện tại liên quan tới quá trình hoạt động của công ty, số lượng các lệnh được thực hiện trong ngày hoặc số lượng các hang hoá được sản xuất, các chính sách, sản phẩm, đòi hỏi nghiệp vụ, yêu cầu pháp quy hiện tại hoặc các rang buộc khác cũng rất cần thiết cho việc phát triển ứng dụng. Các thông tin hiện tại nên được chuyển thành các tư liệu cho phù hợp với đội ngũ phát triển để tăng sự hiểu biết của họ về ứng dụng và phạm vi của bài toán

Các *đòi hỏi trong tương lai* liên quan đến các sự thay đổi sẽ diễn ra, chúng không chính xác và rất khó kiểm tra. Các dự đoán kinh tế, khuynh hướng tiếp thị, kinh doanh là các ví dụ.

Tính chất 2: *Tính có cấu trúc.*

Thông tin chúng ta thu thập được là những thông tin được tổ chức theo một cấu trúc (khuôn mẫu) nhất định; có như vậy mới thể hiện một ý nghĩa phản ánh một đối tượng nào đó, điều này là hiển nhiên. Tuy nhiên, trong quá trình thu thập dữ liệu, chúng ta có khi không hiểu được cấu trúc của thông tin phản ánh, mà rất có thể hiểu theo hướng khác (điều này đã được đề cập ở phần *các lỗi có thể mắc phải trong quá trình phát triển hệ thống* - Chương 2).

Cấu trúc của thông tin định hướng về phần mở rộng theo đó thông tin có thể được phân loại theo một cách nào đó. Cấu trúc có thể tham chiếu tới các hàm, môi trường hoặc dạng dữ liệu hạy hình thức xử lý. Các thông tin thay đổi từ phi cấu trúc cho tới cấu trúc mà phần cấu trúc được xác định bởi công nghệ phần mềm (SE).

Một ví dụ thực tế khi phân tích chức năng của nghiệp vụ. Các chức năng của nghiệp vụ nếu theo người quản lý hệ thống thì không thể kể ra hết vì đó là các công việc của từng bộ phận, của từng nhân viên. Do vậy ta chỉ nắm được những cái tổng quan (có tính trừu tượng cao - không rõ ràng, cụ thể). Còn các chức năng nghiệp vụ của từng bộ phận, từng nhân viên thì rất nhỏ lẻ. Và đứng giữa một danh sách các chức năng như vậy thì khó có thể thấy được tính cấu trúc của nó. Các nhà phân tích lại phải "ngồi lại" với nhau và tổ chức lại các chức năng nghiệp vụ đó. Có như vậy thì khi xây dựng chương trình, ta tránh phải làm đi làm lại các chức năng giống nhau giữa các bộ phận trong thực tế. Mà ta chỉ cần nêu ra một liên kết (link) từ bộ phận (module) này đến bộ phận khác.

Tính "không chuẩn" của dữ liệu thể hiện rõ nhất ở thông tin trong một tờ "hoá đơn". Hoá đơn thanh toán thể hiện rất nhiều thông tin, như: *Số HD, Tên HĐ, Tên khách hàng, Địa chỉ khách hàng,*

… và sau đó là một bảng liệt kê chi tiết *tên các mặt hàng, đơn giá, số lượng, thành tiền* ... nhưng trong thực tế, không một bảng dữ liệu có khuôn dạng giống như một hoá đơn nào có mặt trong kho dữ liệu của hệ thống. Điều này là do liên kết dữ liệu từ các bảng khác mà thành, tránh lưu trữ trùng lặp quá nhiêu thông tin. Do vậy, các nhà thiết kế dữ liệu đã tổ chức lại cấu trúc của dữ liệu cần lưu trữ.

Tính chất 3: *Đầy đủ.*

Hơn lúc nào hết, khi tìm hiểu về một đối tượng hay lĩnh vực nào đó, ta luôn cần thông tin phản ánh về nó một cách đầy đủ và chính xác nhất có thể có. Về mặt lý thuyết thì không bao giờ ta có được toàn bộ thông tin về đối tượng hay lĩnh vực mà ta xử lý. Trong thực tế cũng như vậy, thông tin mà ta có chỉ là tạm đủ để ta có thể xử lý mà thôi.

Các thông tin có thể xếp theo cấp độ tính đầy đủ mà cao nhất là mọi thông tin cần thiết sẽ được biểu diễn. Mỗi kiểu ứng dụng đòi hỏi một mức độ đầy đủ khác nhau. Các hệ thống xử lý giao dịch luôn tiếp cận các thông tin đầy đủ và chính xác (ví dụ hệ thống bán vé máy bay). Tuy nhiên các hệ thống xây dựng theo kiến trúc hệ chuyên gia hay trí tuệ nhân tạo (AI) là minh hoạ tốt nhất việc xử lý thông tin không đầy đủ.

Tính chất 4: *Nhập nhằng.*

Tính nhập nhằng là một thuộc tính của dữ liệu không trong sáng về nghĩa hoặc có nhiều nghĩa một cách hữu ý (có chủ định). Tính chất này liên quan đến mức độ ngữ nghĩa. Ví dụ, nhìn thấy một cửa hiệu có thể đề biển "Giặt là hấp", thì một cậu bé có thể hỏi bố một câu hỏi như sau: "Tại sao giặt lại là hấp?", vào hoàn cảnh này, ông bố sẽ phải mất rất nhiều công sức để giải thích cho con hiểu. Như vậy có hiện tượng "ông nói gà, bà hoá cuốc". Để giải quyết vấn đề này cần căn cứ vào ngữ cảnh.

Tính chất 5: *Ngữ nghĩa.*

Mọi người trong một tổ chức đều có một tập hợp các định nghĩa được chia sẻ cho biết các thuật ngữ, chính sách hoặc các hành động được biểu hiện như thế nào.

Ngữ nghĩa rất quan trọng với việc phát triển ứng dụng và với chính bản thân ứng dụng đó. Nếu mọi người dùng chung một thuật ngữ mà có cách hiểu khác nhau thì sẽ dẫn đến không thể trao đổi thông tin được. Đối với ứng dụng thì dữ liệu sẽ không bao giờ xử lý được cho đến khi người sử dụng hiểu được ngữ nghĩa của dữ liệu này. Các ứng dụng sẽ có ý nghĩa xác định với mục dữ liệu được định tính thông qua việc đào tạo và sử dụng lâu dài. Khi các cán bộ chủ chốt chuyển công tác, thì khả năng chuyển hoá ngữ nghĩa dễ mất. Việc đánh mất ngữ nghĩa của một công ty có thể gây tổn thất rất lớn cho công ty đó.

Tính chất 6: *Độ lớn (volume).*

Volume là số lượng các sự kiện nghiệp vụ hệ thống phải tiến hành trong một chu kỳ nào đó. Volume của tạo mới hay thay đổi khách hàng được tiến hành theo tháng hoặc năm, trong đó volume của giao dịch được tiến hành theo ngày giờ hoặc là theo peak volume (peak volume là số các giao

dịch hoặc các sự kiện được thực hiện trong thời kỳ bận nhất). Thời kỳ cao điểm có thể là cuối năm hoặc cuối các quý, ví dụ chuẩn bị cho báo cáo nộp thuế. Volume của dữ liệu là một nguồn thông tin phức tạp bởi vì số lượng thời gian cần thiết với một giao dịch đơn lẻ có thể trở thành rất quan trọng đối với lượng lớn dữ liệu cần xử lý sau này.

Các kỹ thuật thu thập dữ liệu.

Các kỹ thuật thu thập dữ liệu có thể kể ra là: phỏng vấn, họp nhóm, quan sát ấn định công việc tạm thời, xem xét tài liệu, xem xét phần mềm. Mỗi kỹ thuật đều có điểm mạnh và hạn chế và số lượng và kiểu dữ liệu ta thu được khi sử dụng chúng. Chúng ta hãy bàn luận về các kỹ năng này.

Phỏng vấn.

Phỏng vấn là việc tập hợp một nhóm người số lượng ít trong một khoảng thời gian cố định với một mục đích cụ thể. Phỏng vấn thường được tiến hành với 1 hoặc 2 người hỏi đối với 1 người được phỏng vấn. Trong quá trình phỏng vấn, các câu hỏi có thể được thay đổi. Bạn có thể đánh giá được cảm nhận của họ, động cơ và thói quen với các bộ phận, quá trình quản lý hoặc các thông tin về thực thể khác đáng chú ý. Kiểu của phỏng vấn là kiểu của thông tin yêu cầu. Phỏng vấn được dẫn dắt sao cho cả 2 bên tham gia đều cảm thấy thoả mãn với kết quả của nó. Cuộc phỏng vấn được chuẩn bị kỹ đồng nghĩa với việc hiểu được về người đang được phỏng vấn. Do đó bạn không là cho họ bối rối và bạn có thể hỏi vài câu ban đầu được chuẩn bị cho dù không phải là tất cả.

Một cuộc phỏng vấn bao giờ cũng có bắt đầu, đoạn giữa và kết thúc.

- Lúc bắt đầu, bạn tự giới thiệu và đặt các câu hỏi đơn giản. Nên bắt đầu với các câu hỏi tổng quát vì không đòi hỏi các trả lời mang tính quan điểm cá nhân. Hãy chú ý đến kết quả trả lời để tìm ra mối các câu hỏi tiếp theo và tính trung thực, thái độ của người được phỏng vấn.
- Vào giữa buổi, nên tập trung vào chủ đề. Hãy lấy mọi thông tin bạn cần lưu ý, sử dụng các kỹ thuật mà bạn đã chọn ban đầu. Nếu thấy một vài thông tin qua trọng, hãy hỏi xem bạn có thể được thảo luận sau này.
- Vào lúc kết thúc, hãy tóm tắt các thứ mà bạn đã nghe và nói những gì sẽ phỏng vấn tiếp. Bạn có thể ghi chép và đề nghị người được hỏi xem xét lại. Tốt nhất là trong thời gian 48 giờ và có sự chấp nhận của người dùng theo ngày xác định.

Phỏng vấn có thể sử dụng 2 loại câu hỏi:

- *Câu hỏi mở*: Là câu hỏi có nhiều cách trả lời khác nhau, câu hỏi mở thích hợp cho các chức năng ứng dụng hiện tại cũng như đang đề nghị và cho việc xác định cảm nhận ý kiến, và mong đợi về ứng dụng được đề ra. Một ví dụ là: "Ông có thể nói cho tôi về …", "Ông có thể mô tả làm thế nào …".
- *Câu hỏi đóng*: là câu hỏi mà chỉ trả lời "có" hoặc "không" hoặc một câu trả lời cụ thể. Các câu hỏi đóng tốt cho khai thác thông tin thực tế hoặc bắt người dùng tập trung vào phỏng

vấn. Ví dụ, câu hỏi có thể là: "Bạn có dùng các báo cáo hàng tháng hay không ?". Với các câu trả lời "Có" thì có thể được tiếp nối bằng câu hỏi mở: "Ông có thể giải thích …"

Các bước tiến hành phỏng vấn thành công

Tiến hành đặt cuộc hẹn phù hợp với thời gian của phỏng vấn.

Chuẩn bị tốt, tìm hiểu kỹ về người được phỏng vấn.

Đúng giờ.

Có kế hoạch mở đầu

- Giới thiệu bản thân, mục đích.
- Sử dụng câu hỏi mở để bắt đầu.
- Luôn lưu ý vào câu trả lời.
- Có kế hoạch cho nội dung chính.
- Kết hợp câu hỏi đóng và mở.
- Luôn bám sát các cách trình bày và phát triển chi tiết.
- Luôn cung cấp thông tin phản hồi, ví dụ: "Cho phép tôi trình lạ điều ông vừa nói …".
- Hạn chế ghi chép nếu thấy không tiện.
- Có kế hoạch kết thúc.
- Tóm tắt nội dung, yêu cầu hiệu chỉnh.
- Yêu cầu xác thực lại nội dung, đánh giá lại ghi chép.
- Cho biết ngày tháng họ sẽ nhận được báo cáo.
- Thống nhất ngày tháng lấy bản hiệu chỉnh.
- Xác nhận lại lịch làm việc.

Các câu hỏi có thể đưa ra theo kiểu có cấu trúc hay phi cấu trúc.

- *Phỏng vấn có cấu trúc* là phỏng vấn trong đó người được phỏng vấn đã có danh sách các mục cần duyệt qua, các câu hỏi xác định và các thông tin cần tìm hiểu đã được xác định trước.
- *Phỏng vấn không cấu trúc* là phỏng vấn được định hướng bởi câu trả lời. Các câu hỏi phần lớn là câu hỏi mở, không có một kế hoạch ban đầu. Do vậy người đi phỏng vấn biết các thông tin cần thiết sẽ dùng từ các câu hỏi mở để phát triển chi tiết hơn về chủ đề.

Phỏng vấn có cấu trúc thích hợp khi bạn biết về các thông tin cần thiết trước khi phỏng vấn. Ngược lại, phỏng vấn phi cấu trúc thích hợp khi bạn không thể đoán trước được chủ đề, hay chưa có thông tin gì về người được phỏng vấn. Các trường hợp điển hình của phỏng vấn là người khách hàng bắt đầu với phỏng vấn phi cấu trúc để cho hai bên nhận thức được về miền của bài toán (hiểu sơ lược vấn đề). Sau đó, phỏng vấn dần dần trở thành có cấu trúc và tập trung vào các thông tin bạn cần để hoàn chỉnh phần phân tích.

Các kết quả phỏng vấn người sử dụng lên được trao đổi lại với người được phỏng vấn trong một thời gian ngắn. Người được phỏng vấn phải được báo trước về thời hạn đối với việc phỏng vấn. Tuy nhiên, có thể xin bố trí bổ sung phỏng vấn trong trường hợp còn nhiều điều cần hỏi hoặc nhiều người cần gặp.

Bảng sau so sánh phỏng vấn có cấu trúc và phỏng vấn phi cấu trúc.

	Phỏng vấn có cấu trúc	**Phỏng vấn phi cấu trúc**
Ưu điểm	Dùng dạng chuẩn cho nhiều câu hỏi Dễ quản lý và đánh giá Đánh giá được nhiều mục đích. Không cần đào tạo nhiều. Có kết quả trong các phỏng vấn.	Có khả năng mềm dẻo nhất Cần chăm chú nghe và có kỹ năng mở rộng câu hỏi. Có thể bao được những thông tin chưa biết Đòi hỏi có thực hành.
Nhược điểm	Chi phí chuẩn bị lớn. Tính có cấu trúc có thể không thích hợp cho mọi tình huống. Giảm tính chủ động của người đi phỏng vấn.	Lãng phí thời gian phỏng vấn. Người được phỏng vấn có thể định kiến với các câu hỏi. Tốn thời gian lựa chọn và phân tích thông tin.

Một kỹ năng tốt là phát triển các sơ đồ như là một phần của tài liệu phỏng vấn. Khi bắt đầu một cuộc phỏng vấn mới, nên bàn bạc về các sơ đồ và đưa cho họ bản ghi chép để họ có thể kiểm tra sau này. Bạn sẽ nhận được ngay ý kiến phản hồi về tính chính xác của sơ đồ và hiểu biết của bạn về ứng dụng. Lợi ích của cách tiếp cận này thể hiện cả mặt kỹ năng và tâm lý. Từ khía cạnh kỹ thuật, bạn thường xuyên được kiểm tra lại các vấn đề mà bạn được nghe. Cho tới khi thời gian phân tích kết thúc, cả bạn và khách hàng đều tin chắc rằng quá trình xử lý ứng dụng là đầy đủ. Từ khía cạnh tâm lý, bạn làm tăng niền tin của khách hàng vào khả năng phân tích bằng cách trình bày các hiểu biết của mình. Mỗi khi bạn cải thiện sơ đồ và đi vào phân tích, bạn cũng tăng được niềm tin của người sử dụng rằng bạn có thể xây dựng được ứng dụng đáp ứng được nhu cầu của họ.

Phỏng vấn thích hợp cho việc nhận thông tin đảm bảo cả số lượng lẫn chất lượng:

Các kiểu thông tin định tính là: các ý kiến, niềm tin, thói quen, chính sách và mô tả.

Các kiểu thông tin định lượng bao gồm: tần suất, số lượng, định lượng các mục được dùng trong ứng dụng.

Phỏng vấn là một dạng khác của thu thập dữ liệu có thể làm bạn lạc lối, thiếu chính xác hoặc thông tin không thích hợp. Bạn cần học cách đọc ngôn ngữ bằng cử chỉ, thói quen để quyết đinh được các điều kiện cần thiết cho cùng một thông tin.

Trong khi phỏng vấn, chúng ta cần chú ý đến hàn động củ người được phỏng vấn để có cách ứng xử thích hợp. Bảng sau liệt kê một vài tình huống và kinh nghiệm xử lý.

Hành vi của người được phỏng vấn.	**Đáp ứng của người đi phỏng vấn.**
Đoán các câu trả lời chứ không thừa nhận là không biết	Sau phỏng vấn, kiểm tra chéo các câu trả lời.
Cố nói những điều lọt tai người đi phỏng vấn, sai sự thật.	Tránh các câu hỏi dễ đoán được câu trả lời, kiểm tra chéo các câu hỏi
Cho thông tin không đầy đủ	Kiên trì hỏi để đạt mục đích.
Dừng trình bày khi người đi phỏng vấn ghi chép	Ghi nhanh nhất có thể, chỉ hỏi các câu quan trọng
Vội vã hay trả lời rời rạc, uể oải	Nhanh chóng kết thúc, đề nghị bố trí buổi khác
Thể hiện sự không quan tâm, trả lời đứt quãng	Nói chuyện vui sau đó chuyển đề tài khác
Không muốn thay đổi môi trường hiện tại	Động viên cải thiện môi trường hiện tại và so sánh 2 khuynh hướng.
Không hợp tác, từ chối trả lời	Lấy nguồn tin khác và hỏi: "Ông có quan tâm về những điều người khác nói về ông hay không?". Nếu câu trả lời là "Không" thì thôi phỏng vấn.
Phàn nàn về vị trí công tác, lương, …	Tìm ra mấu chốt vấn đề. Cố gắng dẫn dắt về chủ đề chính, ví dụ: "Dường như cơ quan ông có rất nhiều vấn đề, có thể ứng dụng mới mà chúng tôi đề xuất sẽ giải quyết được các vấn đề trên".
Là người thích thú về công nghệ	Chọn lọc các thông tin cần thiết, không để bị lôi cuốn vào các vấn đề công nghệ.

Phỏng vấn và gặp gỡ phù hợp với mọi loại kiểu dữ liệu do đó chúng thường xuyên được sử dụng.

Ưu điểm của phỏng vấn:

- Nhận được cả thông tin chất lượng và số lượng.
- Nhận được cả thông tin đầy đủ và chi tiết.
- Là phương pháp tốt cho các yêu cầu bên ngoài.

Nhược điểm của phỏng vấn:

- Đòi hỏi có kỹ năng giao tiếp.

- Có thể có kết quả thiên vị vì mang tính chủ quan của người được phỏng vấn.
- Có thể dẫn đến các thông tin sai lạc, không liên quan, thiếu chính xác.
- Đòi hỏi phải có 3 người để kiểm tra kết quả.
- Không thích hợp với số lượng lớn người.

Quan sát.

Quan sát có thể tiến hành thủ công hoặc tự động.

- *Theo cách thủ công*, người quan sát ngồi tại chỗ và ghi chép lại các hoạt động, các bước xử lý công việc. Các băng video đôi khi có thể được dùng. Ghi chép hoặc băng ghi hình được phân tích cho các sự kiện, các mô tả động từ chính, hoặc các hoạt động chỉ rõ lý do, công việc, hoặc các thông tin về công việc.
- *Theo cách tự động*, máy tính sẽ lưu trữ chương trình thường trú, lưu lại vết của các chương trình được sử dụng, email và các hoạt động khác được xử lý bởi máy. Các file nhật ký của máy sẽ được phân tích để mô tả công việc.

Ưu điểm của quan sát:

- Bao trùm được các tiêu chuẩn quyết định, quy trình suy luận, các thủ tục khớp nối (mang tính thực hành).
- Kỹ sư phần mềm sẽ không bị định kiến (không bị ảnh hưởng bởi người khác) mà hoàn toàn tập trung vào vấn đề của mình.
- Quan sát sẽ khắc phục ngăn cách giữa kỹ sư phần mềm và người được phỏng vấn.
- Nhận được các hiểu biết tốt về môi trường công tác hiện tại, vấn đề và quá trình xử lý thông qua quan sát.

Nhược điểm của quan sát:

- Thời gian quan sát có thể không biểu diễn cho các cong việc diễn ra thông thường.
- Thói quen dễ thay đổi do biết mình bị quan sát (người bị quan sát sẽ mất tự nhiên, hành động có thể bị ghò ép).
- Mất nhiều thời gian.

Người đi quan sát nên xác định cái gì sẽ được quan sát. Nên xác định thời gian cần thiết cho việc quan sát, hãy xin sự chấp thuận của cả người quản lý và cá nhân trước khi tiến hành quan sát.

Ấn định công việc tạm thời.

Không có gì thay thế được kinh nghiệm. Với một công việc tạm thời, bạn có được nhận thức đầy đủ hơn về các nhiệm vụ. Cũng vậy, đầu tiên bạn học các thuật ngữ hoàn cảnh sử dụng nó. Thời gian kéo dài từ 2 tuần đến 1 tháng đủ dài để bạn có thể quen với phần lớn các công việc thông thường và các tình huống ngoại lệ nhưng không được quá dài để trở thành chuyên gia thực sự đối với công việc.

Công việc tạm thời cho bạn cơ sở hình thức hoá các câu hỏi về chức năng nào của phương pháp hiện thời của công việc sẽ được giữ lại và cái nào sẽ bị loại trừ hoặc thay đổi, nghiên cứu được ngữ cảnh hiện tại. Có thể bằng công việc để thay thế cho các câu hỏi không thực hiện được. Bất lợi của công việc tạm thời là tốn thời gian và sự lựa chọn về thời gian có thể làm tối thiểu hoá vấn đề, không bao hết được các hoạt động hoặc thời gian. Một nhược điểm khác nữa là kỹ sư phần mềm có thể thiên kiến hoá về quá trình xử lý công việc (do tự mình đã làm), nội dung làm ảnh hưởng đến công việc thiết kế sau này.

Họp nhóm (meeting)

Meeting là việc tập trung từ 3 người trở lên trong một khoảng thời gian để thảo luận về một chủ đề nhất định. Meeting có thể vừa bổ sung vừa thay thế phỏng vấn bằng cách cho phép các thành viên kiểm tra lại các kết quản phỏng vấn cá nhân. Nó có thể thay thế phỏng vấn bằng cách cung cấp một diễn đàn cho các thành viên cùng tìm ra các yêu cầu và các giải pháp cho ứng dụng.

Meeting có thể làm lãng phí thời gian. Nói chung nếu meeting càng lớn thì càng ít ý kiến nhất trí và thời gian để đi đến quyết định sẽ kéo dài. Do vậy lên có kế hoạch ban đầu cho meeting. Lịch trình nên cung cấp trước cho các thành viên. Số lượng chủ đề cần thảo luận chỉ nên thấp hơn 5 chủ đề. Meeting lên có thời gian cố định và có địa điểm thống nhất cụ thể với các quyết định cần thiết. Meeting không nên kéo dài quá 2 giờ để có thể đảm bảo được sự tập trung, chú ý của các thành viên.

Ưu điểm của họp nhóm :

- Có thể ra quyết định mà các thành viên đều phải tuân theo (đa số).
- Nhận được cả thông tin tổng hợp và chi tiết.
- Là phương pháp tốt cho các yêu cầu bên ngoài.
- Tập hợp được nhiều người dùng liên quan.

Nhược điểm của họp nhóm:

- Mất nhiều công sức thời gian và tiền bạc để chuẩn bị.
- Nếu số đại biểu nhiều sẽ tốn thời gian để ra được quyết định.
- Các ngắt quãng trong cuộc họp dễ làm mọi người phân tán.
- Dễ chuyển sang các chủ đề ít liên quan như : chính trị, thể thao, thời trang …
- Mời không đúng thành viên dẫn đến chậm có kết quả.

Điều tra qua bản câu hỏi

Được ứng dụng khi cần lấy ý kiến của đại đa số người dùng về một số thông tin để có thể tập hợp số liệu thống kê mà không có điều kiện gặp trực tiếp. Với cách này, người thu thập dữ liệu sẽ soạn trước một bản câu hỏi, có thể có sẵn các phương án lựa chọn để người dùng lựa chọn đánh dấu vào, sau đó thu lại và thống kê kết quả.

Ví dụ, các câu hỏi có thể như sau :

Bạn thường ứng dụng máy tính vào các lĩnh vực nào sau đây ?

A. Giải trí. B. Công việc. C. Do ý thích. D. Không dùng.

Với cách thức này, người thu thập không cần mất thời gian gặp trực tiếp (như phỏng vấn hoặc họp nhóm) mà vẫn thu được thông tin, không đòi hỏi kỹ năng giao tiếp. Các câu hỏi trong danh sách có thể là dạng phỏng vấn trên giấy hoặc máy tính. Ưu điểm chính của câu hỏi là nếu như không cần phải chỉ rõ tên của người trả lời thì thông tin các câu trả lời sẽ có tính trung thực cao hơn. Cũng vậy, các câu hỏi chuẩn xác cung cấp các dữ liệu thực mà theo đó các quyết định có thể được dựa vào. Các mục câu hỏi, như là phỏng vấn có thể là câu hỏi mở hoặc đóng.

Ưu điểm của bản câu hỏi :

- Người cho ý kiến có thể không cần biết tên do vậy cho quan điểm và cảm nhận có tính trung thực cao, có thể dựa vào đó để ra quyết định.
- Có thể tiến hành với nhiều người.
- Thích hợp với các câu hỏi đóng và hữu hạn.
- Phù hợp với công ty đa chức năng và có thể tuỳ biến theo địa phương.

Nhược điểm của bản câu hỏi :

- Khó thực hiện lại được.
- Các câu hỏi không được trả lời không có nghĩa là không có thông tin.
- Các câu hỏi có thể khó hiểu do yêu cầu cần phải ngắn gọn
- Thực hiện đánh giá có thể chậm.
- Người dùng ít có khả năng đưa ra ý kiến khác (do tính đóng của các câu hỏi).
- Không thể bổ xung thêm thông tin khi đã tiến hành công bố các bản câu hỏi.

Xem xét tài liệu

Khái niệm tài liệu ám chỉ các cẩm nang, quy định, các thao tác chuẩn mà tổ chức cung cấp như là hướng dẫn cho các nhà quản lý và nhân viên.

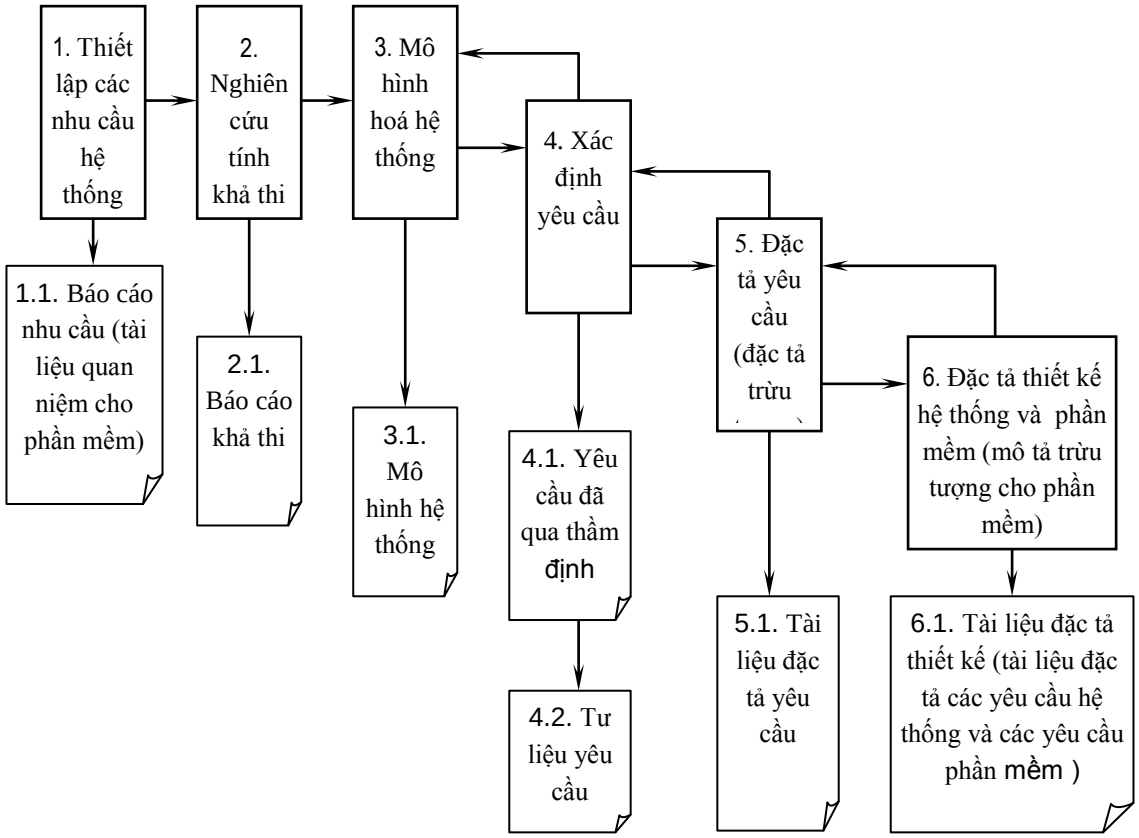
Các tài liệu không phải luôn nằm trong đơn vị đó. Tài liệu có thể là tài liệu nội bộ, có thể là các ấn phẩm kỹ thuật, các báo cáo nghiên cứu, … Các tài liệu thực sự có ý nghĩa với kỹ sư phần mềm để tìm hiểu các lĩnh vực mà họ chưa từng có kinh nghiệm. Nó hữu ích cho việc xác định các câu hỏi về quá trình thao tác và sản xuất. Tài liệu đưa ra các thông tin mang tính khách quan.

Tài liệu nội bộ mô tả được ngữ cảnh hiện thời ; phù hợp với việc nghiên cứu có tính lịch sử (quá trình hoạt động lâu dài). Tuy nhiên việc phải cung cấp tài liệu nội bộ làm cho người dùng e ngại, gây thành kiến ; khó có thể nhận biết được quan điểm, động cơ tiến hành công việc.

Tài liệu ngoài cho ta xác định được các khuynh hướng công nghiệp, ý kiến các chuyên gia, các kinh nghiệm của các công ty khác về thông tin, kỹ thuật. Tuy nhiên thông tin có thể không xác đáng, thiếu chính xác và có thể gây thành kiến.

Xem xét phần mềm

Phân tích và định rõ yêu cầu là bước kỹ thuật đầu tiên trong tiến trình kỹ nghệ phần mềm. Hoạt *động phân tích* và *định rõ yêu cầu* hướng tới đặc tả yêu cầu phần mềm đực thể hiện trong các khuôn cảnh như sau:



Các đặc tả thường mang tính trừu tượng hoá cao. Do vậy người ta phân chia thành nhiều mức đặc tả. Càng ở mức cao (những mức đầu tiên của quá trình làm mịn hoặc chính xác hoá) đặc tả càng trừu tượng. Càng xuống các mức thấp hơn, đặc tả càng tiến dần tới cụ thể - tức là một thể hiện trên một máy tính cụ thể với một ngôn ngữ lập trình cụ thể - đây chính là quá trình làm mịn dần.

Các loại hình đặc tả.

Có hai kiểu đặc tả đó là *đặc tả hình thức* và *đặc tả phi hình thức*.

Đặc tả hình thức: Là các đặc tả chính xác tức là không thể dẫn tới những cách hiểu khác nhau. Đặc tả hình thức sử dụng công cụ chủ yếu là đại số và logic.

Ví dụ: Đặc tả một ma trận:

- Cấp của ma trận n x n (n là số tự nhiên lẻ).
- Phần tử cuối của hàng 1 bằng phần tử đầu của hàng cuối.
- Phần tử trung tâm bằng trung bình cộng của các phần tử ở 4 góc.

Hoặc có thể diễn đạt như sau:

- A n x n = (a[i, j])n x n; n = 2k + 1, k ∈ Z.
- a[1, n] = a[n, 1].
- $a\left[\frac{n+1}{2}, \frac{n+1}{2}\right] = \left(a[1,1] + a[1,n] + a[n,1] + a[n,n]\right)/4$

Đặc tả phi hình thức: Diễn đạt bằng những ngôn ngữ, tuy không chặt chẽ nhưng được nhiều người biết và có thể trao đổi với nhau để chính xác hoá các điểm chưa rõ ràng, những khái niệm còn mơ hồ.

Ví dụ: Có hai con hậu trên bàn cờ. Hai con hậu sẽ đụng độ nếu chúng nằm trên cùng hàng, cùng cột hoặc trên cùng một đường chéo song song với đường chéo chính hay đường chéo phụ. => Rõ ràng ở đây có một số khái niệm mơ hồ.

Đặc tả hỗn hợp: Phối hợp cả hai kiểu đặc tả trên.

Trong thực tế, có nhiều loại hình đặc tả, ví dụ như:

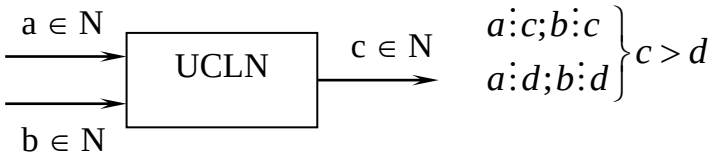
- *Đặc tả cấu trúc dữ liệu:* Nêu các thành phần của dữ liệu

  Ví dụ: Đặc tả một phân số: Phân_số = { x/y , x ∈ Z , y ∈ N }

  Số_phức = { a + b.i | a, b ∈ R }

- *Đặc tả chức năng:* Mô tả thông qua việc nêu lên các tính chất hay thuộc tính của tên vào và tên ra.

  Ví dụ:

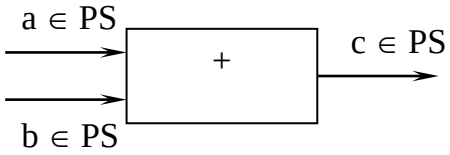


- *Đặc tả đối tượng:* Bao gồm đặc tả cấu trúc dữ liệu và mô tả các chức năng.

Ví dụ: đặc tả đối tượng phân số.

PS = { x/y , x ∈ Z , y ∈ N }

Phép cộng: +: PS x PS → PS



- *Đặc tả thao tác:* Nêu lên trình tự tiến hành công việc.

Ví dụ 1: x, y, z ∈ PS. Các bước cần thực hiện đối với phép cộng (+) 2 phân số.

```
z = x + y      {    Quy_đồng_mẫu_số(x, y);
                    z.tử_số = x.tử_số + y.tử_số;
                    z.mẫu_số = x.mẫu_số;
               };
```

Ví dụ 2: Quy trình **Bán hàng**:

1. Khách hàng yêu cầu được mua hàng.
2. Hướng dẫn khách xem và lựa chọn hàng hoá.
3. Thoả thuận hình thức thanh toán: Tiền mặt, séc, chuyển khoản, …

4. Ghi hoá đơn cho khách.
5. Nhận tiền và giao hàng hoá cho khách.

- *Đặc tả cú pháp:* Thực chất là các định nghĩa có tính truy hồi từ tổng thể đến cơ sở. Mô tả cách lắp ghép các ký hiệu, các từ với nhau lại để tạo thành chương trình. Ví dụ: Trong ngôn ngữ lập trình PASCAL, tên (định danh - identify) được khái quát như sau: Là dãy các ký tự bắt đầu bằng chữ cái hoặc dấu gạch nối dưới, sau đó có thể là chữ số, chữ cái hoặc dấu gạch nối dưới.

  <định danh> = <chữ cái> ∪ <định danh> ∪ <ký tự>

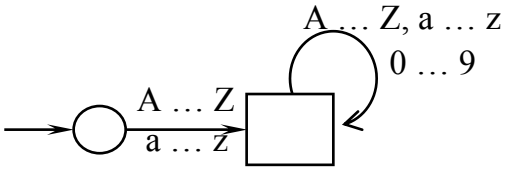
  <ký tự> = <chữ cái> ∪ <chữ số>

  <chữ cái> = { A, B, C, … , Z } ∪ { a, b, …, z }

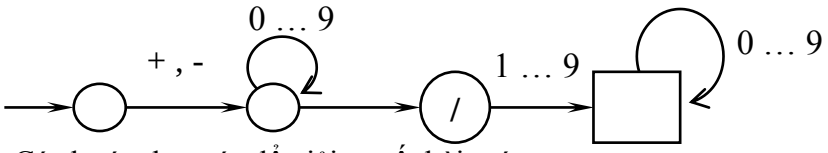
  <chữ số> = { 0, 1, 2, …, 9 }

- *Đặc tả qua sơ đồ:*

  Ví dụ: Đặc tả định danh



  Đặc tả phân số



g. Đặc tả thuật toán: Các bước thao tác để giải quyết bài toán.

Kiểu đặc tả phải phù hợp với giải pháp. Các yêu cầu của phần mềm có thể được phân tích theo một số cách khác nhau. Các ký thuật phân tích có thể dẫn tới những đặc tả trên giấy hay trên máy tính (được xây dụng nhờ CASE) có chứa các mô tả ngôn ngữ đồ hoạ và tự nhiên cho yêu cầu phần mềm. Việc làm bản mẫu giúp đặc tả có thể được triển khai, tức là bản mẫu sẽ thể hiện những công việc thực hiện các yêu cầu. Các ngôn ngữ đặc tả hình thức dẫn đến biểu diễn hình thức.

Các nguyên lý đặc tả.

Đặc tả có thể xem như một tiến trình biểu diễn. Mục đích cuối cùng của đặc tả là các yêu cầu được biểu thị sao cho dẫn tới việc cài đặt phần mềm thành công. Balzer và Goldman đề nghị 8 nguyên lý đặc tả tốt.

Nguyên lý 1*: Phân tách chức năng với cài đặt.*

Trước hết, theo định nghĩa, đặc tả là một mô tả về điều mong muốn, chứ không phải là cách thực hiện nó (cài đặt). Đặc tả có thể chấp nhận 2 dạng hoàn toàn khác nhau. *Dạng thứ nhất* là dạng của các *hàm toán học*: Với một tập *dữ liệu đầu vào* đã cho, tạo ra một *tập dữ liệu đầu ra đặc biệt*. Dạng tổng quát của đặc tả như thế là tìm ra (một hoặc tất cả những) kết quả ứng với P (đầu vào), với P biểu thị một tân từ bất kỳ. Trong đặc tả như thế, kết quả thu được phải được diến đạt một cách

đầy đủ, toàn vẹn, theo dạng đó là cái gì (không phải đó là như thế nào). Một phần điều này là vì kết quả của một hàm (toán học) của đầu vào (phép toán có điểm bắt đầu và điểm kết thúc đã xác định rõ) không bị ảnh hưởng bởi môi trường bao quanh.

Nguyên lý 2*: Cần ngôn ngữ đặc tả hệ thống hướng tiến trình.*

Xét tình huống trong đó môi trường là động và sự thay đổi của nó ảnh hưởng tới hành vi của thực thể nào đó tương tác với môi trường đó (như trong "hệ thống máy tính nhúng"). Hành vi của nó không thể biểu diễn được ở dạng hàm (toán học) của đầu vào. Thay vì thế, cần phải sử dụng cách biểu diễn khác - cách mô tả hướng tiến trình, trong đó đặc tả cái gì đã đạt được bằng cách xác định một mô hình các thao tác mong muốn đạt được của hệ thống dưới dạng các công việc đáp ứng chức năng đối với kích thích khác nhau từ môi trường.

Những đặc tả hướng tiến trình như vậy, trình bày một mô hình về hành vi hệ thống, thông thường đã bị loại ra khỏi các ngôn ngữ đặc tả hình thức, nhưng chúng lại là bản chất nếu nhiều tình huống động phức tạp hơn cần phải được đặc tả. Trong thực tế, cần phải thừa nhận rằng trong những tình huống như vậy cả tiến trình cần tự động hoá lẫn môi trường tồn tại của nó đều phải được mô tả một cách hình thức. Tức là, toàn bộ hệ thống các bộ phận tương tác phải được đặc tả chứ không chỉ một thành phần được đặc tả.

Nguyên lý 3*: Đặc tả phải bao gồm hệ thống có phần mềm là một thành phần trong đó*

Một hệ thống bao gồm các thành phần tương tác nhau. Chỉ bên trong hoàn cảnh của hệ thống toàn bộ và tương tác giữa các thành phần của nó thì hành vi của một thành phần riêng mới có thể được xác định. Nói chung, một hệ thống có thể được mô hình hoá như một tập hợp các sự vật tích cực và thụ động. Những sự vật này có liên quan lẫn nhau và qua thời gian thì mối quan hệ giữa các sự vật thay đổi. Mối quan hệ động này đưa ra sự kích thích cho các sự vật tích cực, còn gọi là các *tác nhân*, đáp ứng. Sự đáp ứng có thể gây ra những thay đổi thêm nữa, và do đó, tạo ra thêm kích thích để cho các tác nhân có thể đáp ứng lại.

Nguyên lý 4*: Đặc tả phải bao gồm cả môi trường mà hệ thống vận hành.*

Tương tự, môi trường mà trong đó hệ thống vận hành và tương tác với cũng phải được xác định.

May mắn là điều này đơn thuần chỉ cần sự thừa nhận rằng bản thân môi trường cũng là một hệ thống bao gồm các sự vật tương tác, cả tích cực lẫn thụ động, mà trong đó hệ thống chỉ là một tác nhân. Các tác nhân khác, theo định nghĩa là không thay đổi bởi vì chúng là một phần của môi trường, giới hạn phạm vi của việc thiết kế và cài đặt về sau. Trong thực tế, sự khác nhau duy nhất giữa hệ thống và môi trường của nó là ở chỗ nỗ lực thiết kế và cài đặt về sau sẽ vận hành chỉ trong đặc tả cho hệ thống. Đặc tả môi trường làm cho "giao diện" của hệ thống được xác định theo cùng cách như bản thân hệ thống chứ không đưa vào cách hình thức khác.

Cần phải chú ý rằng bức tranh đặc tả hệ thống được trình bày ở đây chính là bức tranh của tập hợp các tác nhân xoắn xuýt nhau cao độ phản ứng với những kích thích trong môi trường (thay

đổi các sự vật) do các tác nhân đó tạo ra. Chỉ có thông qua những hành động điều phối của tác nhân mà hệ thống mới đạt tới mục tiêu của nó. Sự phụ thuộc lẫn nhau vi phạm vào nguyên lí phân tách (cô lập với các phần khác của hệ thống và môi trường). Nhưng đây là một nguyên lí *thiết kế,* không phải là nguyên lí đặc tả. Thiết kế tuân theo đặc tả, và quan tâm tới việc phân rã một đặc tả thành các mẩu gần tách biệt để chuẩn bị cho cài đặt. Tuy nhiên đặc tả phải vẽ lại chính xác bức chân dung của hệ thống và môi trường của nó như cộng đồng người dùng cảm nhận theo một cách thức nhiều chi tiết như các giai đoạn cài đặt và thiết kế cần tới. Vì mức độ chi tiết cần thiết này là khó thấy trước, nếu không nói là không thể, nên đặc tả, thiết kế và cài đặt phải được thừa nhận như một hoạt động tương tác. Do đó điều mấu chốt là công nghệ cần có để bao quát thật nhiều cho hoạt động này khi bản đặc tả được soạn thảo và thay đổi (trong cả hai giai đoạn phát triển khởi đầu và bảo trì về sau).

Nguyên lý 5*: Đặc tả hệ thống phải là một mô hình nhận thức.*

Đặc tả hệ thống phải là một mô hình nhận thức chứ không phải là một mô hình thiết kế hay cài đặt. Nó phải mô tả một hệ thống như cộng đồng người sử dụng cảm nhận thấy. Các sự vật mà nó thao tác phải tương ứng với các sự vật của lĩnh vực đó; các tác nhân phải mô hình cho các cá nhân, tổ chức và trang thiết bị trong lĩnh vực đó; còn các hành động họ thực hiện thì phải mô hình cho những hoạt động thực tế xuất hiện trong lĩnh vực.

Đặc tả phải có khả năng tổ hợp vào trong nó những qui tắc hay luật bao trùm các sự vật thuộc lĩnh vực. Một số trong những trường hợp là luật *bài trừ những trạng thái nào đó của hệ thống* (như "hai sự vật không thể đồng thời ở cùng một chỗ và vào cùng một lúc"), và do đó giới hạn hành vi của các tác nhân hay chỉ ra nhu cầu soạn thảo thêm để ngăn cản những trạng thái này khỏi nảy sinh. Các luật khác *mô tả cách các sự vật đáp ứng lại khi bị kích thích* (như luật chuyển động của Newton). Những luật này, biểu thị cho "tính vật lí" của lĩnh vực, là phần cố hữu của đặc tả hệ thống.

Nguyên lý 6*: Đặc tả phải thể hiện tính vận hành.*

Đặc tả phải đủ đầy đủ và hình thức để có thể được dùng trong việc xác định liệu một cài đặt được đề nghị có thoả mãn đặc tả cho những trường hợp kiểm thử tuỳ ý không. Tức là, *với kết quả của việc cài đặt trên một tập dữ liệu được chọn một cách tuỳ ý, phải có thể dùng đặc tả để xác định tính hợp lệ cho những kết quả đó*. Điều này kéo theo rằng đặc tả, mặc dầu không phải là một đặc tả hoàn toàn về cách thức, vẫn có thể hành động như một bộ sinh các hành vi có thể trong số những hành vi phải có của cài đặt được đề nghị. Do đó, theo một nghĩa mở rộng, đặc tả này phải là vận hành ...

Nguyên lý 7*: Đặc tả chấp nhập dung sai về tính không đầy đủ.*

Không đặc tả nào có thể là đầy đủ hoàn toàn. Môi trường trong đó nó tồn tại thường quá phức tạp cho điều đó. Một đặc tả bao giờ cũng là một mô hình - một sự trừu tượng hoá - của một tình huống thực (hay được mường tượng) nào đó. Do đó, nó sẽ không đầy đủ. Hơn thế nữa, như đã được phát biểu nó sẽ tồn tại tại ở nhiều mức chi tiết. Tính vận hành được yêu cầu ở trên không nhất

thiết là cần thiết. Các công cụ phân tích được sử dụng để giúp cho người đặc tả và để kiểm thử đặc tả phải có khả năng xử lí với tính không đầy đủ. Một cách tự nhiên điều này làm cho việc phân tích bị yếu đi, khi có thể được thực hiện bằng cách mở rộng phạm vi các hành vi chấp nhận được thỏa mãn cho đặc tả, nhưng một sự suy giảm như vậy phải phản ánh các mức độ bất trắc còn lại.

***Nguyên lý 8**: Đặc tả phải được cục bộ hoá và được ghép lỏng lẻo.*

Các nguyên lí trước xử lí đặc tả như một thực thể tĩnh. Thực thể này nảy sinh từ cái động của đặc tả. Cần phải thừa nhận rằng mặc dầu mục tiêu chính của một đặc tả là để dùng làm cơ sở cho thiết kế và cài đặt một hệ thống nào đó, nó không phải là một sự vật tĩnh dựng sẵn mà là một sự vật động đang trải qua thay đổi đáng kể. Việc thay đổi như thế xuất hiện trong ba hoạt động chính: phát biểu, khi một đặc tả ban đầu đang đươc tạo ra, phát triển, khi đặc tả được soạn thảo trong quá trình thiết kế lặp để phản ánh môi trường đã thay đổi và / hoặc các yêu cầu chức năng phụ.

Với nhiều thay đổi xuất hiện cho đặc tả, điều mấu chốt là nội dung và cấu trúc của nó được chọn để làm phù hợp hoạt động này. Yêu cầu chính cho sự phù hợp đó là ở chỗ thông tin bên trong đặc tả phải được cục bộ hoá sao cho chỉ một phần nhỏ (một cách lí tưởng) cần phải sửa đổi khi thông tin thay đổi, và ở chỗ đặc tả cần được cấu trúc (ghép) một cách lỏng lẻo để cho từng phần có thể được thêm vào hay loại bỏ một cách dễ dàng, và cấu trúc được điều chỉnh một cách tự động.

Mặc dầu các nguyên lí được Balzer và Goldman tán thành tập trung vào tác động của đặc tả trên định nghĩa về ngôn ngữ hình thức, những lời bình luận của họ áp dụng được cho cả mọi dạng đặc tả. Tuy nhiên, các nguyên lí cần phải được dịch thành sự thực hiện. Trong mục sau chúng ta sẽ xem xét một tập các hướng dẫn để tạo ra một đặc tả các yêu cầu.

Các mức trừu tượng của đặc tả.

Các đặc tả được thể hiện ở một vài mức trừu tượng khác nhau cùng với mối tương liên giữa các mức ấy. Mỗi mức nhắm đến các đối tượng đọc khác nhau mà họ có quyền quyết định về việc dựa vào đó mà thực hiện đánh giá bản thiết kế của các nhà phát triển phần mềm. Các mức đó là:

***Mức 1**: Định ra yêu cầu.*

Được thể hiện bằng ngôn ngữ tự nhiên về các dịch vụ mà hệ thống sẽ phải cung cấp. Phần này phải được viết sao cho dễ hiểu đối với khách hàng và người quản lý hợp đồng, người sẽ mua sản phẩm phần mềm và người sẽ sử dụng nó. Kỹ thuật đặc tả phi hình thức là thích hợp cho mức đặc tả này.

***Mức 2**: Đặc tả yêu cầu.*

Tài liệu nêu ra các dịch vụ một cách chi tiết hơn. Tài liệu này đôi khi còn được gọi là tài liệu đặc tả chức năng. Yêu cầu đối với đặc tả ở mức này là phải chính xác đến mức có thể làm cơ sở cho hợp đồng giữa nhà phát triển phần mềm và khách hàng. Đồng thời cũng cần được viết sao cho dễ hiểu đối với nhân viên kỹ thuật của cả nơi mua phần mềm và nơi phát triển hệ thống. Kỹ thuật đặc tả hình thức hẳn là thích hợp cho mức đặc tả như vậy, tuy nhiên cũng còn tuỳ thuộc vào trình độ kiến thức cơ bản của khách hàng. Tốt hơn cả là ta có thể dùng loại hình hỗn hợp để đặc tả.

- Theo dõi những thuật ngữ mông lung (như "một số", "đôi khi", "thường", "thông thường", "bình thường", "phần lớn", "đa số"); yêu cầu làm sáng tỏ.
- Khi có nêu danh sách, nhưng không đầy đủ, thì phải đảm bảo mọi khoản mục đều được hiểu rõ. Chú ý vào các từ như "vân vân", "cứ như thế", "cứ tiếp tục như thế", "sao cho".
- Phải chắc chắn phát biểu phạm vi không chứa những giả thiết không được nói rõ (như *mã hợp lệ trong khoảng 10 tới 100*. Đó là số nguyên, số thực hay số hệ 16?
- Phải nhận biết về các động từ mơ hồ như "xử lí", "loại bỏ", "nhảy qua", "xoá bỏ" Có thể có nhiều cách hiểu về nó.
- Phải nhận biết các đại từ "vu vơ" (như "mô đun vào/ra liên lạc với mô đun kiểm tra tính hợp lệ dữ liệu và đặt cờ báo kiểm soát *của nó*." Cờ kiểm soát của ai?).
- Tìm các câu có chứa sự chắc chắn (như "bao giờ", "mọi", "tất cả", "không một", "không bao giờ") rồi yêu cầu bằng chứng.
- Khi một thuật ngữ được định nghĩa tường minh tại một chỗ thì hãy thử thay thế định nghĩa này vào chỗ xuất hiện của nó.
- Khi một cấu trúc được mô tả theo lời thì hãy vẽ ra bức tranh để giúp hiểu được nó.
- Khi một tính toán được xác định thì hãy thử với ít nhất hai thí dụ.

Một khi việc xét duyệt đã hoàn tất thì bản bản *đặc tả yêu cầu phần mềm* sẽ được cả khách hàng lẫn người phát triển "ký tắt". Bản đặc tả trở thành một "hợp đồng" cho việc phát triển phần mềm. Những thay đổi trong yêu cầu được nêu ra sau khi bản đặc tả đã hoàn thành sẽ không bị huỷ bỏ. Nhưng khách hàng phải lưu ý rằng từng thay đổi sau khi kí đều là một mở rộng của phạm vi phần mềm và do đó có thể làm tăng thêm chi phí và / hoặc kéo dài lịch biểu (thời gian thực hiện).

Ngay cả với những thủ tục xét duyệt tốt nhất tại chỗ thì một số vấn đề đặc tả thông thường vẫn còn lại. Bản đặc tả rất khó "kiểm thử" theo mọi cách có ý nghĩa, và do đó sự không nhất quán hay bỏ sót có thể bị bỏ qua không để ý tới. Trong khi xét duyệt, người ta có thể khuyến cáo những thay đổi cho bản đặc tả. Có thể sẽ cực kì khó khăn để lượng định tác động toàn cục của thay đổi; tức là, làm sao việc thay đổi trong một chức năng lại ảnh hưởng tới các yêu cầu cho chức năng khác?

Bài tập:

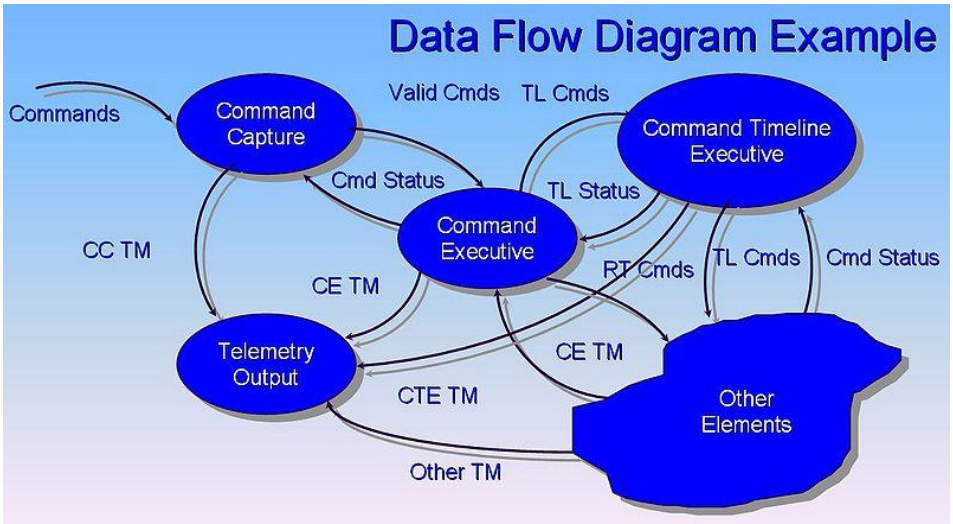
1. Trình bày các kỹ thuật thu thập yêu cầu
2. Trình bày mô hình phân tích yêu cầu
3. Trình bày các tài liệu đặc tả yêu cầu

- Dòng dữ liệu: Dòng dữ liệu là dòng chuyển dời thông tin vào hoặc ra khỏi một tiến trình,một chức năng, một kho dữ liệu hoặc một đối tượng nào đó. Các thành phần của dòng dữ liệu bao gồm đường biểu diễn dòng, mũi tên chỉ hướng dịch chuyển thông tin và tên của dòng. Cân chú ý là các dòng dữ liệu khác nhau phải mang tên khác nhau, và các thông tin trải qua thay đổi thì phải có tên mới cho phù hợp.
- Kho dữ liệu: Trong sơ đồ dòng dữ liệu, kho dữ liệu thể hiện các thông tin cần lưu trữ. Dưới dàng vật lý, kho dữ liệu này có thể là tập tài liệu, cặp hồ sơ hoặc tệp thông tin trên đĩa. Trong sơ đồ dòng dữ liệu, dưới tên kho dữ liệu chúng ta sẽ chỉ quan tâm tới các thông tin được chứa trong đó.

Trong một trang sơ đồ dòng dữ liệu ta có thể đặt một kho dữ liệu ở nhiều chỗ, nhằm giúp việc thể hiện các dòng dữ liệu trở nên dễ dàng hơn.

- Tác nhân ngoài: Tác nhân ngoài có thể là một người, một nhóm người hoặc một tổ chức bên ngoài hệ thống, nhưng có mối liên hệ với hệ thống.
- Tác nhân trong: Tác nhân trong là một chức năng hoặc một tiến trình bên trong hệ thống, được miêu tả ở trang khác của sơ đồ.
- Sơ đồ ngữ cảnh: Sơ đồ ngữ cảnh(Context Diagrams) bao gồm ba nhóm thành phần:
    + Thành phần chính là một vòng tròn nằm ở vị trí trung tâm vủa sơ đồ, biểu thị cho toàn bộ hệ thống đang được nghiên cứu.
    + Xung quanh vòng tròn trung tâm này là tất cả các phần tử bên ngoài, có quan hệ với hệ thống (tác nhân ngoài).
    + Tất cả các đường truyền thông tin vào và ra khỏi hệ thống (nghĩa là nối hệ thống với mọi tác nhân ngoài của nó).

Ví dụ về biểu đồ luồng dữ liệu:

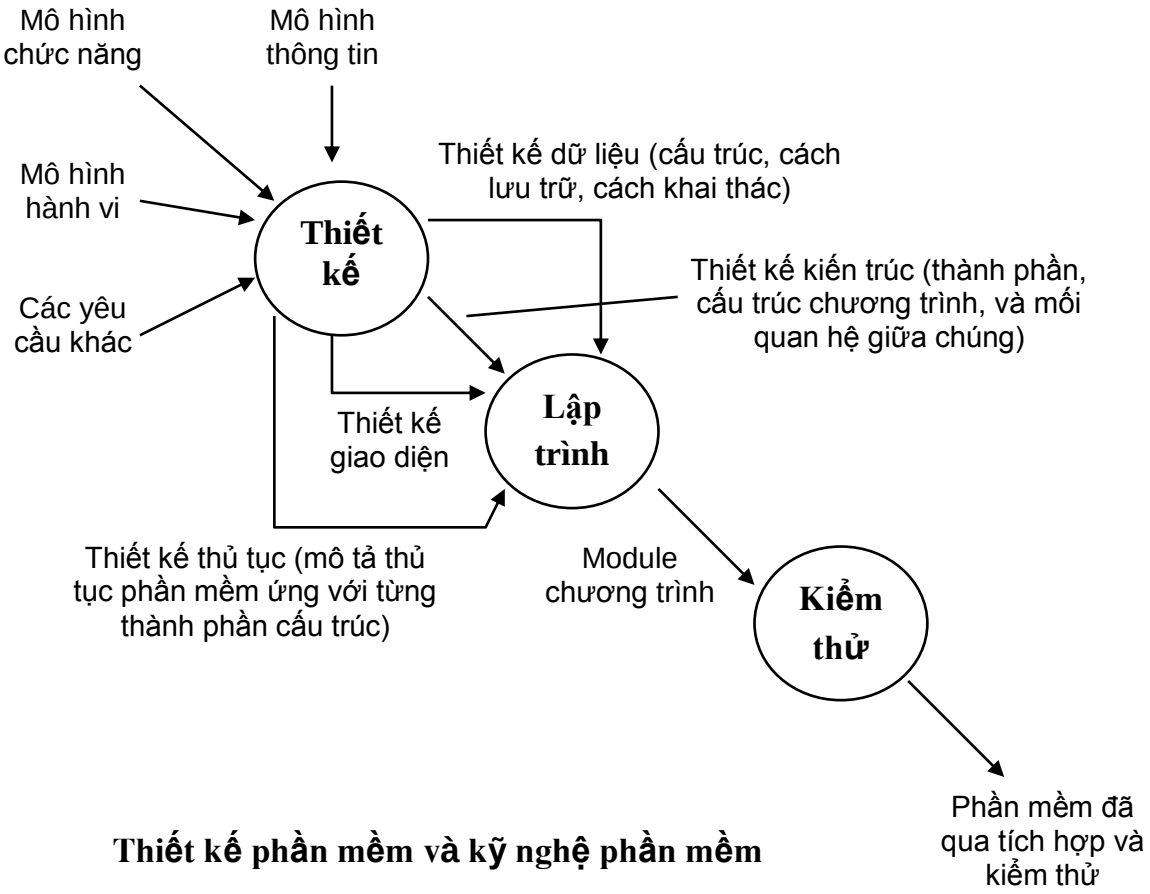


Bài tập:

1. Trình bày Biểu đồ Phân rã chức năng
2. Trình bày biểu đồ luồng dữ liệu

Một khi các yêu cầu về phần mềm đã được phân tích và đặc tả thì thiết kế phần mềm là một trong ba hoạt động kỹ thuật - *thiết kế, lập trình, kiểm thử* - những hoạt động cần để xây dựng và kiểm chứng phần mềm. Từng hoạt động này biến đổi thông tin theo cách cuối cùng tạo ra phần mềm máy tính hợp lệ.

Luồng thông tin trong giai đoạn kỹ thuật này của tiến trình kỹ nghệ phần mềm được minh hoạ trong sơ đồ sau:



Thiết kế phần mềm và kỹ nghệ phần mềm

Các yêu cầu phần mềm, được biểu thị bởi các mô hình thông tin, chức năng và hành vi là cái vào cho bước thiết kế. Bằng việc sử dụng một trong số các phương pháp thiết kế, bước thiết kế tạo ra thiết kế dữ liệu, thiết kế kiến trúc và thiết kế thủ tục.

- *Thiết kế dữ liệu*: Chuyển mô hình lĩnh vực thông tin đã tạo ra trong bước phân tích thành cấu trúc dữ liệu sẽ cần cho việc cài đặt phần mềm.
- *Thiết kế kiến trúc*: Định nghĩa ra mối quan hệ giữa các thành phần cấu trúc chính của chương trình.

"Hình mẫu thiết kế" có thể được dùng để đạt tới các yêu cầu đã được xác định cho hệ thống, và những ràng buộc ảnh hưởng tới cách mà các hình mẫu thiết kế kiến trúc này có thể được áp dụng. Biểu diễn thiết kế kiến trúc - khuôn khổ của hệ thống dựa trên máy tính - có thể được suy ra từ đặc tả hệ thống, mô hình phân tích và tương tác của các hệ con được định nghĩa bên trong mô hình phân tích.

• ***Thiết kế giao diện:*** Mô tả cho cách phần mềm trao đổi với chính nó, với hệ thống liên tác với nó, và với người dùng nó. Giao diện bao gồm một luồng thông tin (như dữ liệu và / hoặc điều khiển) và các kiểu hành vi đặc biệt. Do đó, các biểu đồ luồng dữ liệu và điều khiển cung cấp nhiều thông tin cần cho thiết kế giao diện.

• ***Thiết kế thủ tục***: Biến đổi các thành phần cấu trúc của kiến trúc phần mềm thành mô tả thủ tục cho các cấu phần phần mềm. Chương trình gốc được sinh ra rồi việc kiểm thử được tiến hành để tích hợp và làm hợp lệ.

Trong khi thiết kế chúng ta ra các quyết định mà cuối cùng sẽ ảnh hưởng tới sự thành công của việc xây dựng phần mềm và điều quan trọng là ảnh hưởng tới sự dễ dàng bảo trì nó. Nhưng tại sao thiết kế lại quan trọng?

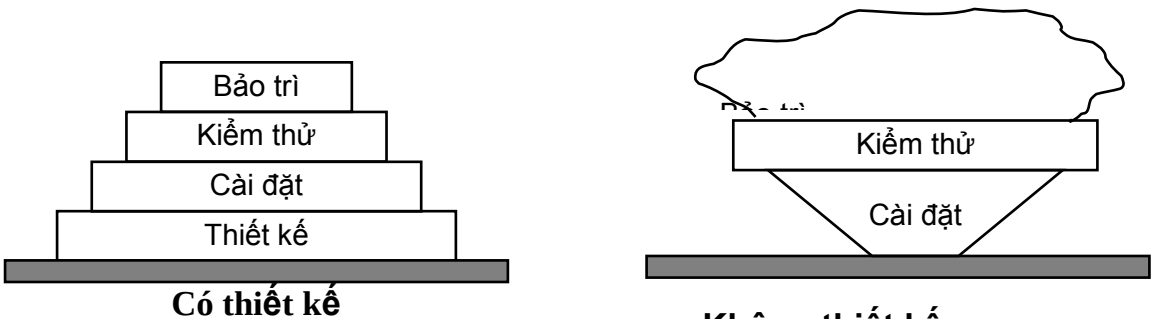
Tầm quan trọng của thiết kế phần mềm có thể được phát biểu bằng một từ - ***chất lượng***. Thiết kế là nơi chất lượng được nuôi dưỡng trong việc phát triển phần mềm: cung cấp cách biểu diễn phần mềm có thể được xác nhận về chất lượng, là cách duy nhất mà chúng ta có thể chuyển hoá một cách chính xác các yêu cầu của khách hàng thành sản phẩm hay hệ thống phần mềm cuối cùng. Thiết kế phần mềm phục vụ như một nền tảng cho mọi bước kỹ nghệ phần mềm và bảo trì:

Tầm quan trọng của thiết kế:

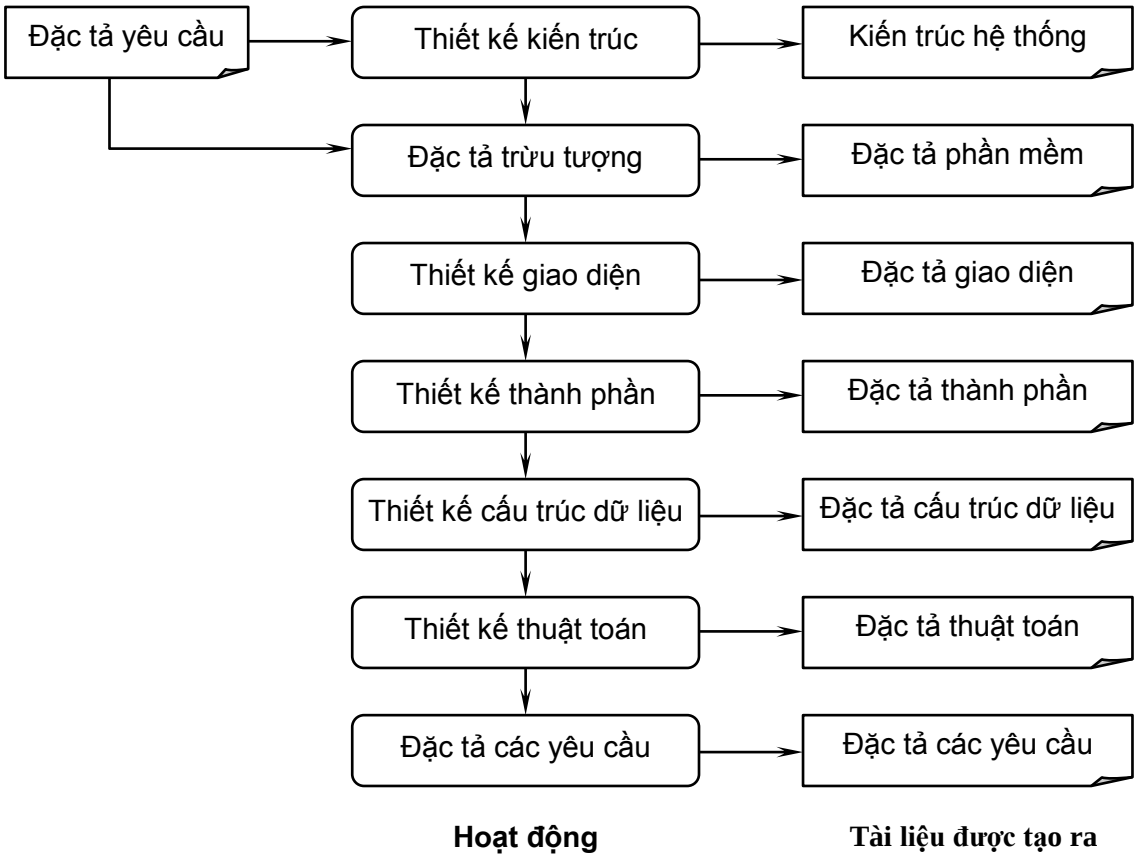
• Không có thiết kế, ta có nguy cơ dựng lên một hệ thống không ổn định - một hệ thống sẽ thất bại khi có một thay đổi nhỏ; một hệ thống khó có thể mà thử được; một hệ thống không thể nào xác định được chất lượng chừng nào chưa đến cuối tiến trình kiểm thử, khi thời gian còn rất ngắn mà không ít tiền đã phải chi ra.

• Thiết kế tốt là chìa khoá cho công trình hữu hiệu, không thể hình thức hoá quá trình thiết kế trong bất kỳ một công trình nào. Chú ý rằng RAISE chỉ là một phương pháp nghiêm ngặt để viết ra thiết kế, phát triển nó, kiểm tra nó chứ tuyệt nhiên không phải là một phương pháp hình thức để phát triển thiết kế.

Thiết kế phần mềm trải qua một số giai đoạn sau:



Giai đoạn 1: Nghiên cứu và hiểu ra vấn đề. Không hiểu rõ vấn đề thì không có thể thiết kế được phần mềm hữu hiệu.



Các hoạt động thiết kế và sản phẩm của thiết kế.

Thành quả của mỗi hoạt động thiết kế là một bản đặc tả. Đặc tả này có thể là một đặc tả trừu tượng, hình thức và được tạo ra để làm rõ các yêu cầu, nó cũng có thể là một đặc tả về một thành phần nào đó của hệ thống phải được thực hiện như thế nào. khi quá trình thiết kế tiến triển thì các yêu cầu ngày càng được bổ sung vào bản đặc tả đó. Các kết quả cuối cùng là các đặc tả về thuật toán và các cấu trúc dữ liệu được dùng làm cơ sở cho việc thực hiện hệ thống.

Thực tế, các hoạt động thiết kế diễn ra song song với các sản phẩm thiết kế khác nhau. Các sản phẩm này lại được triển khai ở các mức chi tiết khác nhau trong diễn biến của quá trình thiết kế.

Các hoạt động cốt yếu trong việc thiết kế một hệ thống phần mềm lớn

1. Thiết kế kiến trúc: Các hệ con tạo nên hệ tổng thể và các quan hệ của chúng là được phân hoạch rõ ràng và ghi thành tài liệu.

2. Đặc tả trừu tượng: Đối với mỗi hệ con, một đặc tả trừu tượng các dịch vụ mà nó cung cấp và các ràng buộc phải tuân theo cũng được hỗ trợ.

3. Thiết kế giao diện: ở đây bạn đọc không nên hiểu "giao diện" chỉ là những gì hiển thị trên màn hình, mà phải hiểu rằng đó có thể là tương tác giữa các thành phần trong hệ thống với nhau. Giao diện với từng hệ con khác cũng được thiết kế và ghi thành tài liệu. Đặc tả giao diện không được mơ hồ và cho phép sử dụng hệ con đó mà không cần biết đến những gì được diễn ra bên trong của hệ con đó (theo kiểu "hộp đen").

4. Thiết kế các thành phần: Các dịch vụ được cung cấp bởi hệ con được phần chia thành các thành phần hợp thành của hệ con đó.

5. Thiết kế cấu trúc dữ liệu: Các cấu trúc dữ liệu được dùng trong việc thực hiện hệ thống được thiết kế chi tiết và được đặc tả ở đây.

6. Thiết kế thuật toán: Các cách thức (phương pháp xử lý) được dùng để cung cấp cho các dịch vụ được thiết kế chi tiết và được đặc tả.

Quá trình này được lặp lại cho mỗi hệ con sao cho đến khi các thành phần hợp thành được xác định một cách rõ ràng và đều có thể chuyển đổi (ánh xạ) một cách trực tiếp vào các thành phần của ngôn ngữ lập trình, chẳng hạn như các gói (packets), các thủ tục (procedures) và các hàm (functions).

Phương pháp tiếp cận thường xuyên được khuyến khích sử dụng là phương pháp tiếp cận *từ trên xuống* (top down): Vấn đề lớn được phân chia một cách đệ quy thành các vấn đề con cho đến khi các vấn đề dễ giải quyết được xác định rõ ràng. Trong quá trình này người thiết kế không nhất thiết phải phân rã tất cả các thành phần trừu tượng (nghĩa là vấn đề này còn phức tạp mà cách giải quyết là chưa xác định rõ) khi mà bằng kinh nghiệm họ đã biết chắc chắn rằng có thể hoàn toàn xây dựng được. Do đó họ có thể tập trung sức lực vào các thành phần đáng xét nhất.

Chú ý rằng khi mà phương pháp hướng đối tượng được chấp nhận thì phương pháp từ trên xuống sẽ ít hiệu quả. Khi đó người thiết kế sử dụng các đối tượng sẵn có để làm khung thiết kế.

Theo quan điểm quản lý dự án, thiết kế phần mềm được tiến hành theo 2 bước:

Bước 1- Thiết kế sơ bộ: Quan tâm tới việc chuyển hoá các yêu cầu thành kiến trúc dữ liệu và các thành phần phần mềm.

Bước 2- Thiết kế chi tiết: Tập trung vào việc làm mịn biểu diễn kiến trúc để dẫn tới cấu trúc dữ liệu chi tiết và biểu diễn các quy trình tính toán và xử lý của phần mềm.

Trong phạm vi thiết kế sơ bộ và chi tiết, có xuất hiện một số hoạt động thiết kế khác nhau. Bên cạnh việc thiết kế dữ liệu, kiến trúc và thủ tục, nhiều ứng dụng hiện đại có hoạt động thiết kế giao diện phân biệt. Thiết kế giao diện lập ra cách bố trí và cơ chế tương tác người-máy (HCI – humen computer interface). Mối quan hệ giữa các khía cạnh kỹ thuật và quản lý của thiết kế được minh hoạ trong hình vẽ dưới đây.

2. Thiết kế nên theo các module, tức là phần mềm nên được phân hoạch một cách logic thành các thành phần thực hiện chức năng hay các chức năng con xác định.
3. Thiết kế nên chứa cách biểu diễn phân biệt và tách biệt giữa dữ liệu và thủ tục.
4. Thiết kế nên dẫn tới các module (như chương trình con hay thủ tục) nêu ra các đặc trưng chức năng đặc biệt.
5. Thiết kế nên dẫn đến giao diện là rút gọn độ phức tạp của việc nối ghép lại giữa các module và với môi trường bên ngoài.
6. Thiết kế nên được hướng theo cách dùng một phương pháp lặp lại được điều khiển bởi thông tin có trong phân tích các yêu cầu phần mềm.

Các đặc trưng trên của một thiết kế tốt có được khi thực hiện đúng tiến trình thiết kế kỹ nghệ phần mềm thông qua việc áp dụng các nguyên lý thiết kế cơ bản, phương pháp luận hệ thống và xét duyệt thấu đáo.

Như vậy, mỗi phương pháp thiết kế phần mềm đều đưa vào những phương pháp trực cảm và lý pháp duy nhất, cũng như một cách nhìn thiển cận thế nào đó về cái gì đặc trưng cho chất lượng thiết kế

Tuy vậy mỗi phương pháp đều có những đặc trưng sau:

1. Một cơ chế để chuyển hoá từ biểu diễn miền thông tin thành biểu diễn thiết kế
2. Một kí pháp để biểu diễn các thành phần chức năng và dao diện của chúng
3. Các trực cảm để làm mịn và phân hoạch
4. Các hướng dẫn về định giá chất lượng

Bất kể phương pháp luận thiết kế nào được dùng, công trình sư phần mềm phải áp dụng một tập các khái niệm nền tảng cho thiết kế dữ liệu, kiến trúc và thủ tục:

- Trừu tượng
- Modul
- Kiến trúc phần mềm.
- Cấp bậc điều khiển
- Cấu trúc dữ liệu
- Thủ tục phần mềm
- Che dấu thông tin

Thiết kế hướng chức năng

Hệ thống được thiết kế theo quan điểm chức năng, bắt đầu ở mức cao nhất, sau đó tinh chế dần dần để thành thiết kế chi tiết hơn. Trạng thái của hệ thống là tập trung và được chia sẻ cho các chức năng thao tác trên trạng thái đó.

Ban đầu, ta coi yêu cầu mức cao nhất của hệ thống là một chức năng duy nhất cần phải thực hiện. Sau đó, ta trả lời cho câu hỏi "Để thực hiện chức năng trên thì cần phải làm các công việc gì?"

– từ *công việc* trong câu hỏi trên được coi là chức năng con của chức năng trên. Thực hiện xong các chức năng con cũng là thực hiện xong chức năng cha. Hệ thống được phân rã dần dần, và được làm mịn. Hình ảnh của hệ thống sẽ được xây dựng theo các bước trên.

Thiết kế hướng đối tượng

Hệ thống được nhìn nhận như một bộ các đối tượng (chứ không phải là một tập hợp các chức năng). Hệ thống được phân tán, mỗi đối tượng có thông tin và trạng thái của riêng nó. Đối tượng là một bộ các thuộc tính xác định trạng thái của đối tượng đó và các phép toán thực hiện trên đó. Mỗi đối tượng là một khách thể của một lớp mà *lớp được xác định bởi thuộc tính và các phép toán* của nó. Nó được thừa kế từ một vài lớp đối tượng cấp cao hơn, sao cho định nghĩa nó chỉ cần nêu đủ các khác nhau giữa nó và các lớp cao hơn nó. Các đối tượng liên lạc với nhau chỉ bằng trao đổi các *thông báo.* Trong thực tế, hầu hết các liên lạc được thực hiện giữa các đối tượng bằng cách nối đối tượng này với một thủ tục, mà thủ tục này kết hợp với một đối tượng khác.

Thiết kế hướng đối tượng dựa trên ý tưởng che dấu thông tin. Gần đây theo cách thiết kế này, người ta đã phát triển nhiều hệ thống cấu tạo bởi nhiều thành phần độc lập và có tương tác với nhau.

Sự thật, các hệ phần mềm lớn phức tạp đến mức mà người ta đã dùng các phương pháp tiếp cận khác nhau trong việc thiết kế các thành phần khác nhau trong hệ thống. Chẳng có một chiến lược tốt nhất nào cho các dự án lớn. Các cách tiếp cận hướng chức năng và hướng đối tượng là bổ sung hỗ trợ cho nhau chứ không phải là loại bỏ nhau. Kỹ sư phần mềm sẽ chọn ra cách tiếp cận thích hợp nhất trong từng giai đoạn thiết kế. Nhìn ở mức tổng thể thì hệ thống như là một bộ các đối tượng (chứ không phải là một bộ các chức năng), cho nên ở mức trừu tượng cao thì cách tiếp cận hướng đối tượng là thích hợp hơn. Đến mức chi tiết thì một cách tự nhiên hơn nên xem chúng là các chức năng tương tác giữa các đối tượng. Sau đó mỗi đối tượng lại được phân giải thành các thành phần, tức là có thể xem nó như là một hệ con.

Rất nhiều hệ thống, đặc biệt là hệ thống thời gian thực được nhúng (vào một hệ thiết bị vật chất có thực) được cấu tạo như một hệ gồm một bộ các quá trình hoạt động song song và có liên lạc với nhau. Các hệ này thường phải tuân theo các ràng buộc nghiêm ngặt về thời gian, mà các phần cứng thường phản ứng tương đối chậm, chỉ có cách tiếp cận nhiều bộ xử lý hoạt động song song mới có thể hoàn thành được yêu cầu về thời gian.

Các chương trình tuần tự là dễ thiết kế, thực hiện và kiểm tra và thử nghiệm hơn là các hệ thống song song. Sự phụ thuộc về thời gian giữa các quá trình là khó hình thức hoá, khó khống chế và thử nghiệm.

Do đó, quá trình thiết kế nên được xem như là một hoạt động gồm 2 giai đoạn:

Giai đoạn 1: Minh định cấu trúc thiết kế logic, cụ thể là các thành phần của hệ thống và các mối quan hệ giữa chúng. Có thể dùng cách nhìn hướng chức năng hoặc cách nhìn hướng đối tượng.

Giai đoạn 2: Thực hiện cấu trúc đó trong dạng có thể thực hiện được. Giai đoạn này đôi khi được gọi là thiết kế chi tiết và đôi khi là lập trình. Chắc rằng sự quyết định về tính song song nên là ở giai đoạn này chứ không phải là các giai đoạn sớm hơn trong quá trình thiết kế.

Bài tập:

1. Trình bày các giai đoạn thiết kế phần mềm
2. Trình bày các nguyên tắc thiết kế phần mềm

```
 1.  program triangle (input,output);
 2.  VAR a, b, c          : integer;
 3.      IsATriangle      : boolean;
 4.  BEGIN
 5.  writeln('Enter three integers which are sides of a triangle:');
 6.  readln(a,b,c);
 7.  writeln('Side A is ',a,'Side B is ',b,'Side C is ',c);
 8.  IF (a < b + c) AND (b < a + c) AND (c < a + b)
 9.      THEN IsATriangle := TRUE
10.      ELSE IsATriangle := FALSE ;
11.  IF IsATriangle
12.      THEN
13.      BEGIN
14.      IF (a = b) XOR (a = c) XOR (b = c) AND NOT((a=b) AND (a=c))
15.          THEN Writeln ('Triangle is Isosceles');
16.      IF (a = b) AND (b = c)
17.          THEN Writeln ('Triangle is Equilateral');
18.      IF (a <> b) AND (a <> c) AND (b <> c)
19.          THEN Writeln ('Triangle is Scalene');
20.      END
21.      ELSE WRITELN('Not a Triangle');
22.  END.
```

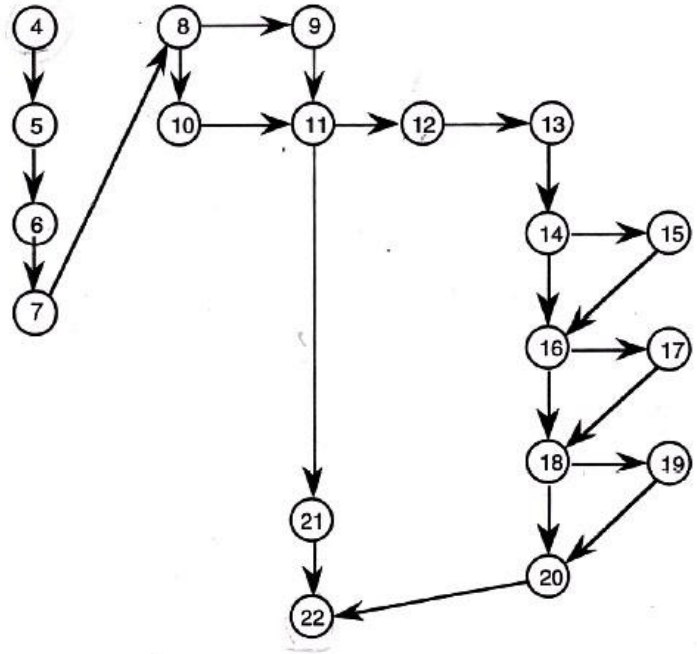


Figure 9.1 Program Graph of the Pascal Triangle Program

Một số định nghĩa

Chuỗi: là một đường dẫn mà trong đó đỉnh bắt đầu và đỉnh kết thúc là khác nhau, và các đỉnh ở bên trong có bậc vào =1 và bậc ra =1

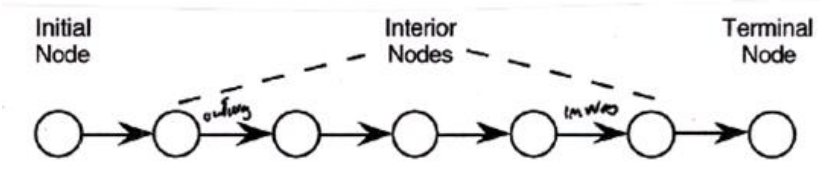


Figure 9.3 A Chain Of Nodes In A Directed Graph

Các bước thực hiện:

- Xây dựng đồ thị chương trình/đồ thị đường dẫn quyết định từ mã nguồn
- Tính độ phức tạp của đồ thị

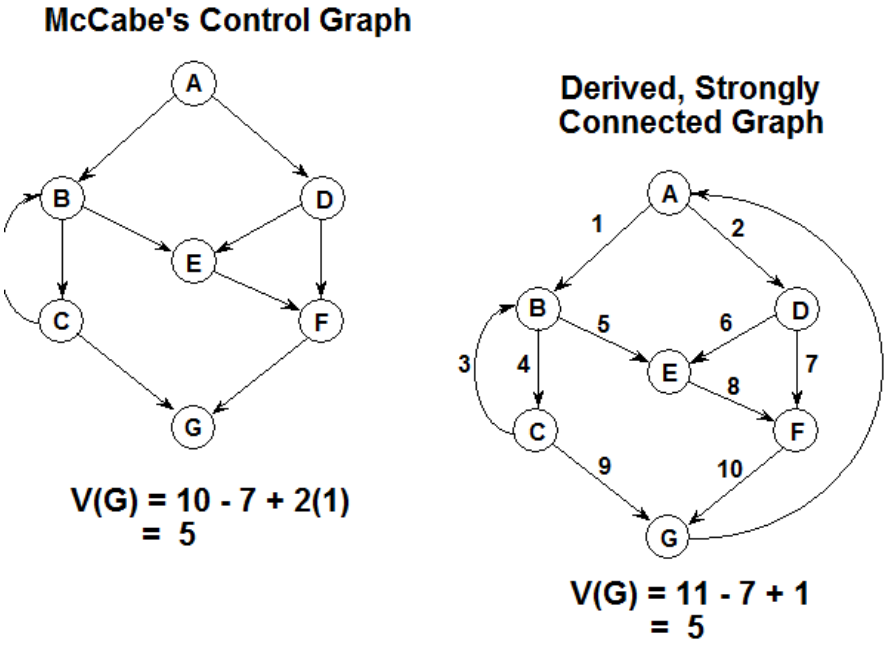
- Xác định một tập hợp các đường dẫn cơ bản
- Thiết kế một trường hợp kiểm thử tương ứng với mỗi đường dẫn cơ bản
- Thực thi các trường hợp kiểm thử

Một đường dẫn cơ bản là đường dẫn nối từ đỉnh bắt đầu đến đỉnh kết thúc.

Số lượng các đường dẫn độc lập cần được kiểm thử bằng giá trị V(G) = e-n+2*p .

Trong đó:

- G là đồ thị đường dẫn quyết định
- V(G) là độ phức tạp của đồ thị G
- e là số cạnh, n là số đỉnh, p là số thành phần



Cách xác định các đường dẫn cơ bản

Chọn một đường dẫn cơ bản ban đầu tương ứng với một sự thực thi chương trình bình thường (đường dẫn cơ bản này nên có càng nhiều đỉnh quyết định càng tốt)

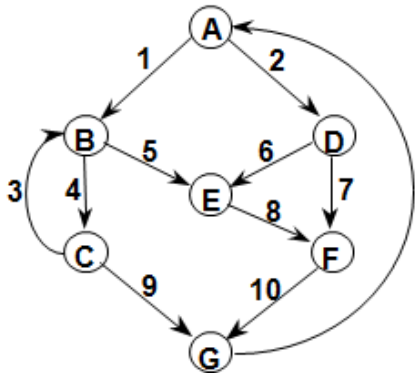
Để tìm các đường dẫn cơ bản khác, dò tìm ngược/xuôi trên đường dẫn ban đầu cho đến khi gặp một đỉnh quyết định. Thay đổi quyết định tại đỉnh này, và tiếp tục tìm đường dẫn khả thi cho đến đỉnh kết thúc

Lặp lại bước trên cho đến khi tất cả các quyết định đều đã được thay đổi với nhánh đúng và sai

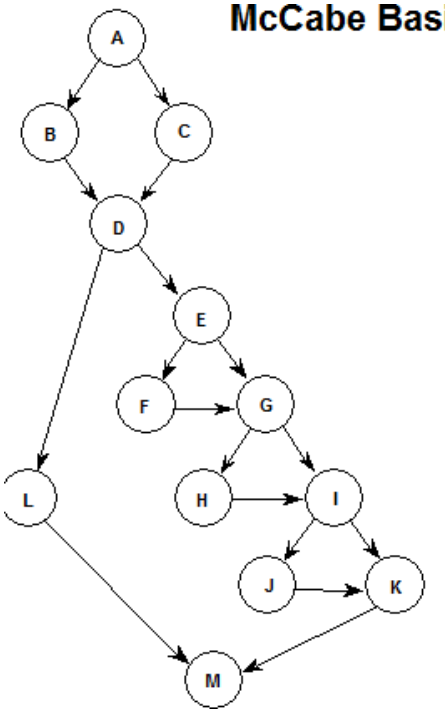
Ví dụ:

path \ edges traversed	1	2	3	4	5	6	7	8	9	10
p1: A, B, C, G	1	0	0	1	0	0	0	0	1	0
p2: A, B, C, B, C, G	1	0	1	2	0	0	0	0	1	0
p3: A, B, E, F, G	1	0	0	0	1	0	0	1	0	1
p4: A, D, E, F, G	0	1	0	0	0	1	0	1	0	1
p5: A, D, F, G	0	1	0	0	0	0	1	0	0	1
ex1: A, B, C, B, E, F, G	1	0	1	1	1	0	0	1	0	1
ex2: A, B, C, B, C, B, C, G	1	0	2	3	0	0	0	0	1	0

ex1 = p2 + p3 - p1
ex2 = 2p2 - p1



Các đường dẫn cơ bản trong bài toán tam giác



McCabe Basis Paths in the Triangle Program

V(G) = 17 - 13 + 2(1) = 6

Basis Path Set B1

A - B - D - L - M - A

A - C - D - L - M - A

A - B - D - E - G - I - K - M - A

A - B - D - E - F - G - I - K - M - A

A - B - D - E - G - H - I - K - M - A

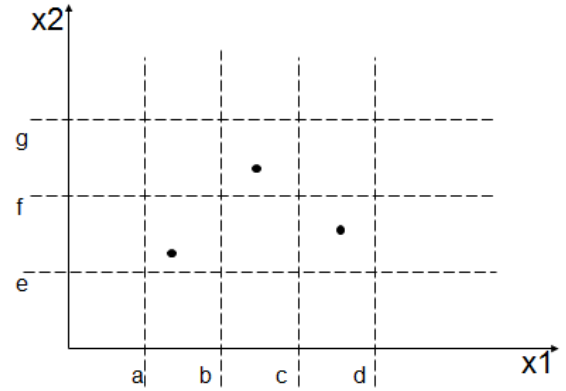
A - B - D - E - G - I - J - K - M - A

There are 18 topologically possible paths.

Các đường dẫn cơ bản khả thi

Dựa trên giả thiết lỗi đơn

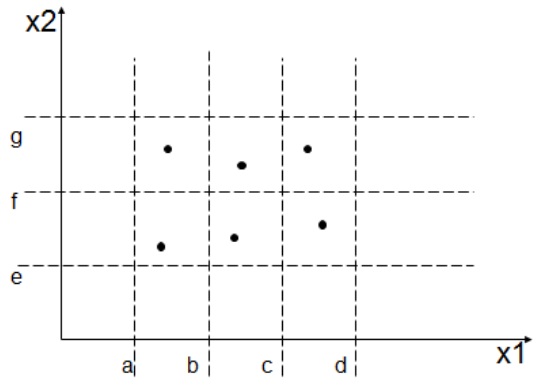
Số lượng trường hợp kiểm thử bằng số lượng nhiều nhất các khoảng giá trị đúng mà một biến có thể nhận



Kiểm thử theo lớp tương đương- lỗi kết hợp

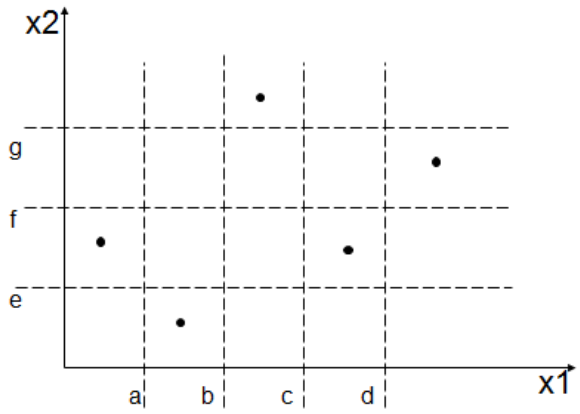
Không sử dụng giả thiết lỗi đơn

Mỗi trường hợp kiểm thử tương ứng với một phần tử của tích Đề các của các lớp tương đương



Kiểm thử theo lớp tương đương- lỗi đơn đầy đủ

Xem xét cả các giá trị không đúng với giả thiết lỗi đơn



Kiểm thử theo lớp tương đương- lỗi kết hợp đầy đủ

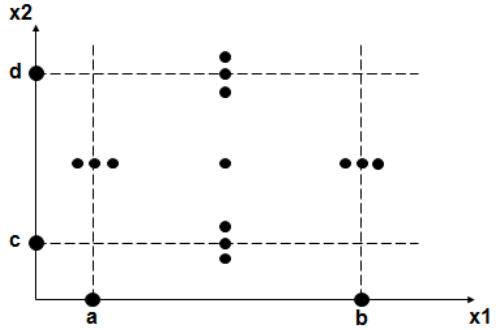
Xem xét cả các giá trị không đúng không sử dụng giả thiết lỗi đơn

- Là một hình thức kiểm thử với lỗi để biết được chương trình sẽ thực hiện như thế nào nếu dữ liệu vào có lỗi
- Có thể không được áp dụng với một số ngôn ngữ lập trình có ràng buộc kiểu

Kiểm thử theo giá trị biên đầy đủ

Test cases for a variable x, where a ≤ x ≤ b

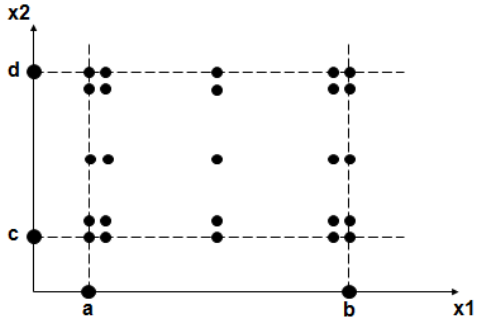
Kiểm thử theo giá trị biên đầy đủ với hai biến



Test cases for variable x1 and x2

Kiểm thử theo giá trị biên xấu nhất (hai biến)

- Loại bỏ giả thiết chỉ có một lỗi đơn
- Cho phép các giá trị đầu vào có thể cùng nhận giá trị biên



Test cases for variable x1 and x2

Số lượng trường hợp kiểm thử

- Kiểm thử theo giá trị biên: 4n+1
- Kiểm thử theo giá trị biên đầy đủ: 6n+1
- Kiểm thử theo giá trị biên xấu nhất: 5n

Với n là số lượng các biến

Nhược điểm của kiểm thử theo giá trị biên

- Các biến phải độc lập với nhau
- Không áp dụng được cho các biến thuộc kiểu logic

Unit một khi đã qua giai đoạn Unit Test với các giao tiếp giả lập thì không cần phải thực hiện Integration Test nữa. Thực tế việc tích hợp giữa các Unit dẫn đến những tình huống hoàn toàn khác. Một chiến lược cần quan tâm trong Integration Test là nên tích hợp dần từng Unit. Một Unit tại một thời điểm được tích hợp vào một nhóm các Unit khác đã tích hợp trước đó và đã hoàn tất (passed) các đợt Integration Test trước đó. Lúc này, ta chỉ cần kiểm tra giao tiếp của Unit mới thêm vào với hệ thống các Unit đã tích hợp trước đó, điều này làm cho số lượng kiểm tra sẽ giảm đi rất nhiều, sai sót sẽ giảm đáng kể.

Có 4 loại kiểm tra trong Integration Test:

- Kiểm tra cấu trúc (structure): Tương tự White Box Test (kiểm tra nhằm bảo đảm các thành phần bên trong của một chương trình chạy đúng), chú trọng đến hoạt động của các thành phần cấu trúc nội tại của chương trình chẳng hạn các lệnh và nhánh bên trong.
- Kiểm tra chức năng (functional): Tương tự Black Box Test (kiểm tra chỉ chú trọng đến chức năng của chương trình, không quan tâm đến cấu trúc bên trong), chỉ khảo sát chức năng của chương trình theo yêu cầu kỹ thuật.
- Kiểm tra hiệu năng (performance): Kiểm tra việc vận hành của hệ thống.
- Kiểm tra khả năng chịu tải (stress): Kiểm tra các giới hạn của hệ thống.

System Test - Kiểm tra mức hệ thống

Mục đích System Test là kiểm tra thiết kế và toàn bộ hệ thống (sau khi tích hợp) có thỏa mãn yêu cầu đặt ra hay không. System Test bắt đầu khi tất cả các bộ phận của phầm mềm đã được tích hợp thành công. Thông thường loại kiểm tra này tốn rất nhiều công sức và thời gian. Trong nhiều trường hợp, việc kiểm tra đòi hỏi một số thiết bị phụ trợ, phần mềm hoặc phần cứng đặc thù, đặc biệt là các ứng dụng thời gian thực, hệ thống phân bố, hoặc hệ thống nhúng. Ở mức độ hệ thống, người kiểm tra cũng tìm kiếm các lỗi, nhưng trọng tâm là đánh giá về hoạt động, thao tác, sự tin cậy và các yêu cầu khác liên quan đến chất lượng của toàn hệ thống.

Điểm khác nhau then chốt giữa Integration Test và System Test là System Test chú trọng các hành vi và lỗi trên toàn hệ thống, còn Integration Test chú trọng sự giao tiếp giữa các đơn thể hoặc đối tượng khi chúng làm việc cùng nhau. Thông thường ta phải thực hiện Unit Test và Integration Test để bảo đảm mọi Unit và sự tương tác giữa chúng hoạt động chính xác trước khi thực hiện System Test.

Sau khi hoàn thành Integration Test, một hệ thống phần mềm đã được hình thành cùng với các thành phần đã được kiểm tra đầy đủ. Tại thời điểm này, lập trình viên hoặc kiểm tra viên (tester) bắt đầu kiểm tra phần mềm như một hệ thống hoàn chỉnh. Việc lập kế hoạch cho System Test nên bắt đầu từ giai đoạn hình thành và phân tích các yêu cầu. Phần sau ta sẽ nói rõ hơn về một quy trình System Test cơ bản và điển hình.

System Test kiểm tra cả các hành vi chức năng của phần mềm lẫn các yêu cầu về chất lượng như độ tin cậy, tính tiện lợi khi sử dụng, hiệu năng và bảo mật. Mức kiểm tra này đặc biệt thích hợp cho

việc phát hiện lỗi giao tiếp với phần mềm hoặc phần cứng bên ngoài, chẳng hạn các lỗi "tắc nghẽn" (deadlock) hoặc chiếm dụng bộ nhớ. Sau giai đoạn System Test, PM thường đã sẵn sàng cho khách hàng hoặc người dùng cuối cùng kiểm tra để chấp nhận (Acceptance Test) hoặc dùng thử (Alpha/Beta Test).

Đòi hỏi nhiều công sức, thời gian và tính chính xác, khách quan, System Test thường được thực hiện bởi một nhóm kiểm tra viên hoàn toàn độc lập với nhóm phát triển dự án.

Bản thân System Test lại gồm nhiều loại kiểm tra khác nhau (xem hình 3), phổ biến nhất gồm:

- Kiểm tra chức năng (Functional Test): bảo đảm các hành vi của hệ thống thỏa mãn đúng yêu cầu thiết kế.
- Kiểm tra khả năng vận hành (Performance Test): bảo đảm tối ưu việc phân bổ tài nguyên hệ thống (ví dụ bộ nhớ) nhằm đạt các chỉ tiêu như thời gian xử lý hay đáp ứng câu truy vấn…
- Kiểm tra khả năng chịu tải (Stress Test hay Load Test): bảo đảm hệ thống vận hành đúng dưới áp lực cao (ví dụ nhiều người truy xuất cùng lúc). Stress Test tập trung vào các trạng thái tới hạn, các "điểm chết", các tình huống bất thường…
- Kiểm tra cấu hình (Configuration Test)
- Kiểm tra khả năng bảo mật (Security Test): bảo đảm tính toàn vẹn, bảo mật của dữ liệu và của hệ thống.
- Kiểm tra khả năng phục hồi (Recovery Test): bảo đảm hệ thống có khả năng khôi phục trạng thái ổn định trước đó trong tình huống mất tài nguyên hoặc dữ liệu; đặc biệt quan trọng đối với các hệ thống giao dịch như ngân hàng trực tuyến.

Acceptance Test - Kiểm tra chấp nhận sản phẩm

Thông thường, sau giai đoạn System Test là Acceptance Test, được khách hàng thực hiện (hoặc ủy quyền cho một nhóm thứ ba thực hiện). Mục đích của Acceptance Test là để chứng minh phần mềm thỏa mãn tất cả yêu cầu của khách hàng và khách hàng chấp nhận sản phẩm (và trả tiền thanh toán hợp đồng).

Regression Test - Kiểm tra hồi quy

Regression Test kiểm tra lại phần mềm sau khi có một sự thay đổi xảy ra, để bảo đảm phiên bản phần mềm mới thực hiện tốt các chức năng như phiên bản cũ và sự thay đổi không gây ra lỗi mới trên những chức năng vốn đã làm việc tốt. Regression test có thể thực hiện tại mọi mức kiểm tra.

Bài tập:

1. Trình bày phương pháp kiểm thử theo giá trị biên
2. Trình bày phương pháp kiểm thử theo phân vùng tương đương
3. Trình bày phương pháp kiểm thử theo đường cơ bản

MỘT SỐ ĐỀ THI MẪU

Trường Đại Học Hàng Hải Việt Nam

Khoa Công nghệ Thông tin

BỘ MÔN HỆ THỐNG THÔNG TIN

-----*-----**

THI KẾT THÚC HỌC PHẦN

Tên học phần: **CÔNG NGHỆ PHẦN MỀM** Năm học: **x**	Đề thi số: **x**	Ký duyệt đề: **x**
Thời gian: **60 phút**		

Câu 1: (2 điểm)

Trình bày các đặc điểm của phần mềm?

Câu 2: (2 điểm)

Trình bày các quy trình phát triển phần mềm cố điển?

Câu 3: (3 điểm)

Trình bày các kỹ thuật thu thập yêu cầu: Phỏng vấn và Quan sát?

Câu 4: (3 điểm)

Trình bày kỹ thuật Mô hình hoá luồng chức năng của hệ thống?

----------***HẾT***----------

Lưu ý: - Không sửa, xóa đề thi, nộp lại đề sau khi thi

Trường Đại Học Hàng Hải Việt Nam

Khoa Công nghệ Thông tin

BỘ MÔN HỆ THỐNG THÔNG TIN

-----*-----**

THI KẾT THÚC HỌC PHẦN

Tên học phần: **CÔNG NGHỆ PHẦN MỀM** Năm học: **x**	Đề thi số: **x**	Ký duyệt đề: **x**
Thời gian: **60 phút**		

Câu 1: (2 điểm)

Trình bày các ứng dụng của phần mềm?

Câu 2: (2 điểm)

Trình bày các quy trình phát triển phần mềm cố điển?

Câu 3: (3 điểm)

Trình bày các kỹ thuật thu thập yêu cầu: Họp nhóm và Bản câu hỏi?

Câu 4: (3 điểm)

Trình bày kỹ thuật Mô hình hoá luồng dữ liệu của hệ thống?

----------***HẾT***----------

Lưu ý: - Không sửa, xóa đề thi, nộp lại đề sau khi thi

Trường Đại Học Hàng Hải Việt Nam

Khoa Công nghệ Thông tin

BỘ MÔN HỆ THỐNG THÔNG TIN

-----*-----**

THI KẾT THÚC HỌC PHẦN

Tên học phần: **CÔNG NGHỆ PHẦN MỀM** Năm học: **x**	Đề thi số: **x**	Ký duyệt đề: **x**
Thời gian: **60 phút**		

Câu 1: (2 điểm)

Trình bày các ứng dụng của phần mềm?

Câu 2: (2 điểm)

Biểu đồ luồng dữ liệu là gì? Các quy ước vẽ biểu đồ luồng dữ liệu?

Câu 3: (3 điểm)

Trình bày quá trình thiết kế phần mềm?

Câu 4: (3 điểm)

Kiểm thử là gì? Trình bày kỹ thuật kiểm thử theo giá trị biên?

----------***HẾT***----------

Lưu ý: - Không sửa, xóa đề thi, nộp lại đề sau khi thi

Bài giảng:

CÔNG NGHỆ PHẦN MỀM

Giảng viên: Nguyễn Quang Vũ
Khoa Khoa học máy tính

Nội dung bài giảng:

Chương 1:

TỔNG QUAN VỀ CÔNG NGHỆ PHẦN MỀM

1.1 Các khái niệm cơ bản

- **Phần mềm** (software): *là một tập hợp các câu lệnh được viết bằng một hoặc nhiều ngôn ngữ lập trình (được gọi là các chương trình), nhằm tự động thực hiện một số các chức năng giải quyết một bài toán.*

1.1 Các khái niệm cơ bản (tt)

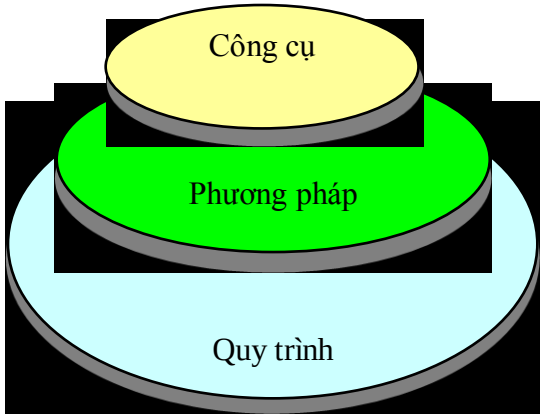
- **Công nghệ** (engineering): *là cách sử dụng các công cụ, các kỹ thuật trong cách giải quyết một vấn đề.*

1.1 Các khái niệm cơ bản (tt)

- **Công nghệ phần mềm** (software engineering): *là việc áp dụng các công nghệ một cách hệ thống trong việc phát triển các ứng dụng dựa trên máy tính.*

1.1 Các khái niệm cơ bản (tt)

- Mô hình 3 tầng của CNPM



1.1 Các khái niệm cơ bản (tt)

- Nói một cách khác, công nghệ phần mềm bao trùm kiến thức, các công cụ, và các phương pháp để:

. định nghĩa yêu cầu phần mềm

. thiết kế phần mềm

. xây dựng phần mềm

. kiểm thử phần mềm

. bảo trì phần mềm

1.1 Các khái niệm cơ bản (tt)

- Công nghệ phần mềm còn sử dụng kiến thức của các lĩnh vực khác:

. kỹ thuật máy tính

. khoa học máy tính

. quản lý

. toán học

. quản lý dự án

. quản lý chất lượng

. công nghệ hệ thống (systems engineering).

1.1 Các khái niệm cơ bản (tt)

"Khi máy tính chưa xuất hiện, thì việc lập trình chưa có khó khăn gì cả. Khi mới xuất hiện một vài chiếc máy tính chức năng kém thì việc lập trình bắt đầu gặp một vài khó khăn nho nhỏ. Giờ đây khi chúng ta có những chiếc máy tính khổng lồ thì những khó khăn ấy trở nên vô cùng lớn. Như vậy ngành công nghiệp điện tử không giải quyết khó khăn nào cả mà họ chỉ tạo thêm ra những khó khăn mới. Khó khăn mà họ tạo nên chính là việc sử dụng sản phẩm của họ."

(Edsger Dijkstra)

1.1 Các khái niệm cơ bản (tt)

- Và nhiều khái niệm khác ….

1.2 Lịch sử công nghệ phần mềm

1.3 Tiêu chuẩn của một sản phẩm phần mềm

- Tính đúng
- Tính khoa học
- Tính tin cậy
- Tính kiểm thử được
- Tính hữu hiệu
- Tính sáng tạo
- Tính an toàn
- Tính toàn vẹn

1.3 Tiêu chuẩn của một sản phẩm phần mềm (tt)

- Tính đối xứng và đầy đủ chức năng
- Tính tiêu chuẩn và tính chuẩn
- Tính độc lập
- Tính dễ phát triển, hoàn thiện
- Ngoài ra: phổ dụng, đơn giản, liên tác, súc tính, thứ lỗi, modul hóa, đầy đủ hồ sơ, theo dõi được, vận hành dễ,…

1.4 Hồ sơ của một sản phẩm phần mềm

- Đặc tả hệ thống.
- Kế hoạch dự án phần mềm.
  - . Đặc tả yêu cầu phần mềm.
  - . Bản mẫu thực hiện được hay "trên giấy".
- Tài liệu người dùng sơ bộ

1.4 Hồ sơ của một sản phẩm phần mềm (tt)

- Đặc tả thiết kế.
  - . Mô tả thiết kế dữ liệu.
  - . Mô tả thiết kế kiến trúc.
  - . Mô tả thiết kế module.
  - . Mô tả thiết kế giao diện.
  - . Mô tả sự vật (nếu kỹ thuật hướng sự vật được dùng).

1.4 Hồ sơ của một sản phẩm phần mềm (tt)

- Chương trình gốc

. Chương trình nguồn.

. Bản in chương trình nguồn (listing).

. Bản mô tả thuật toán tương ứng với chương trình nguồn.

. Kế hoạch và thủ tục kiểm thử.

. Các trường hợp kiểm thử và kết quả ghi lại.

1.4 Hồ sơ của một sản phẩm phần mềm (tt)

- Tài liệu vận hành và cài đặt.
  . Bản liệt kê các lỗi và cách xử lý.
  . Bản liệt kê các thông số đặc trưng của hệ thống.
- Mô tả cơ sở dữ liệu.
  . Diagram và tự điển dữ liệu.
  . Dữ liệu ban đầu

1.4 Hồ sơ của một sản phẩm phần mềm (tt)

- Tài liệu người sử dụng đã xây dựng.
  . Bản hướng dẫn sử dụng chi tiết.
  . Bản tóm tắt hướng dẫn sử dụng.
  . Các chương trình trợ giúp có liên quan.
- Tài liệu bảo trì.
  . Báo cáo vấn đề còn tồn tại.
  . Yêu cầu bảo trì.
  . Trình tự thay đổi công nghệ.
- Các chuẩn và thủ tục cho kỹ thuật phần mềm .
- Các tư liệu khác: hợp đồng, phiên bản, tài liệu pháp lý,...

Chương 2:

CÁC HOẠT ĐỘNG TRONG TIẾN TRÌNH PHẦN MỀM

2.1 Tiến trình phần mềm

- Là một tập hợp các hành động mà mục đích của nó là xây dựng và phát triển phần mềm
- Bao gồm các hoạt động:
  . Đặc tả
  . Phát triển: Thiết kế và cài đặt
  . Kiểm thử
  . Mở rộng: Bảo trì, cải tiến

2.2 Đặc tả

- Còn gọi là kỹ thuật xác định yêu cầu
- Là quy trình tìm hiểu và định nghĩa những dịch vụ nào được yêu cầu và các ràng buộc trong quá trình vận hành và xây dựng hệ thống.
- Gồm 4 pha chính

. Nghiên cứu khả thi

. Phân tích và rút ra các yêu cầu

. Đặc tả yêu cầu

. Đánh giá yêu cầu

2.3 Thiết kế

- Là quá trình thiết kế cấu trúc phần mềm dựa trên những tài liệu đặc tả
- Gồm các công việc chính
  . Thiết kế kiến trúc
  . Đặc tả trừu tượng
  . Thiết kế giao diện
  . Thiết kế thành phần
  . Thiết kế cấu trúc dữ liệu
  . Thiết kế thuật toán

2.4 Cài đặt

- Là quá trình chuyển đổi từ tài liệu đặc tả hệ thống thành một hệ thống thực, có thể vận hành được và phải loại bỏ các lỗi của chương trình
- Hoạt động cá nhân
- Không có quy trình chung

2.5 Kiểm thử

2.5.1 Xác minh và thẩm định

- V&V – Verification and Validation
- Là từ chung cho các quá trình kiểm thử để đảm bảo rằng phần mềm thỏa mãn các yêu cầu của chúng và các yêu cầu đó thỏa mãn các nhu cầu của người sử dụng
- Có hai mục tiêu:

  . Phát hiện các khuyết tật trong hệ thống.

  . Đánh giá xem hệ thống liệu có dùng được hay không?

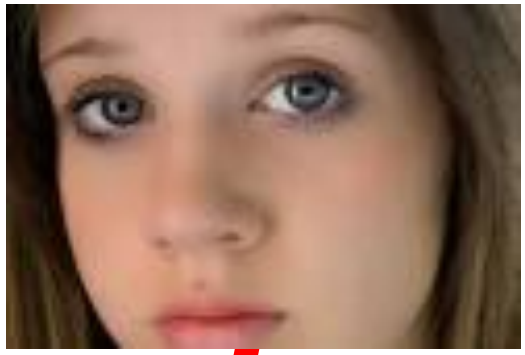
2.5 Kiểm thử

2.5.1 Xác minh và thẩm định (tt)

- Verification: Are we building the product righ?
- Validation: Are we buiding the right product ?

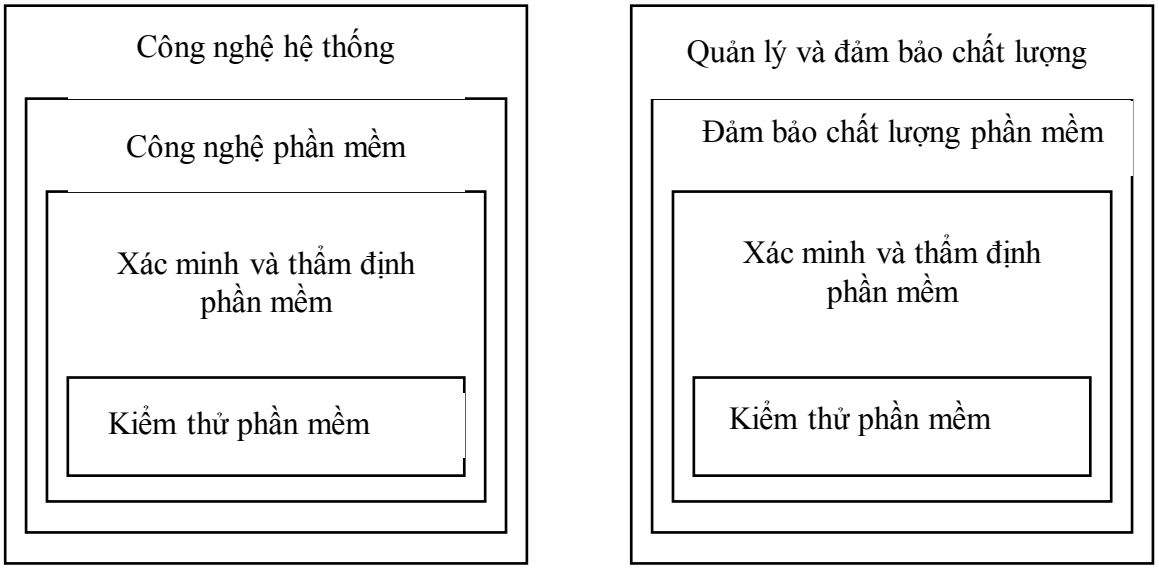
2.5.2 Kiểm thử phần mềm (KTPM)

KTPM là quá trình thực hiện một chương trình phần mềm với mục đích là tìm ra LỖI, nếu có.



KIỂM THỬ PHẦN MỀM

2.5.2 Kiểm thử phần mềm (KTPM)



(a) Ngữ cảnh quy trình (b) Ngữ cảnh chất lượng

Kiểm thử phần mềm trong một số ngữ cảnh

2.5.2 Kiểm thử phần mềm (KTPM)

- Việc kiểm thử của đội dự án được gọi là kiểm thử phát triển (Development test).
- Các kiểm thử bởi các cơ quan bên ngoài được gọi là đảm bảo chất lượng (Quality assurance-QA) và kiểm thử chấp nhận (Acceptance test).

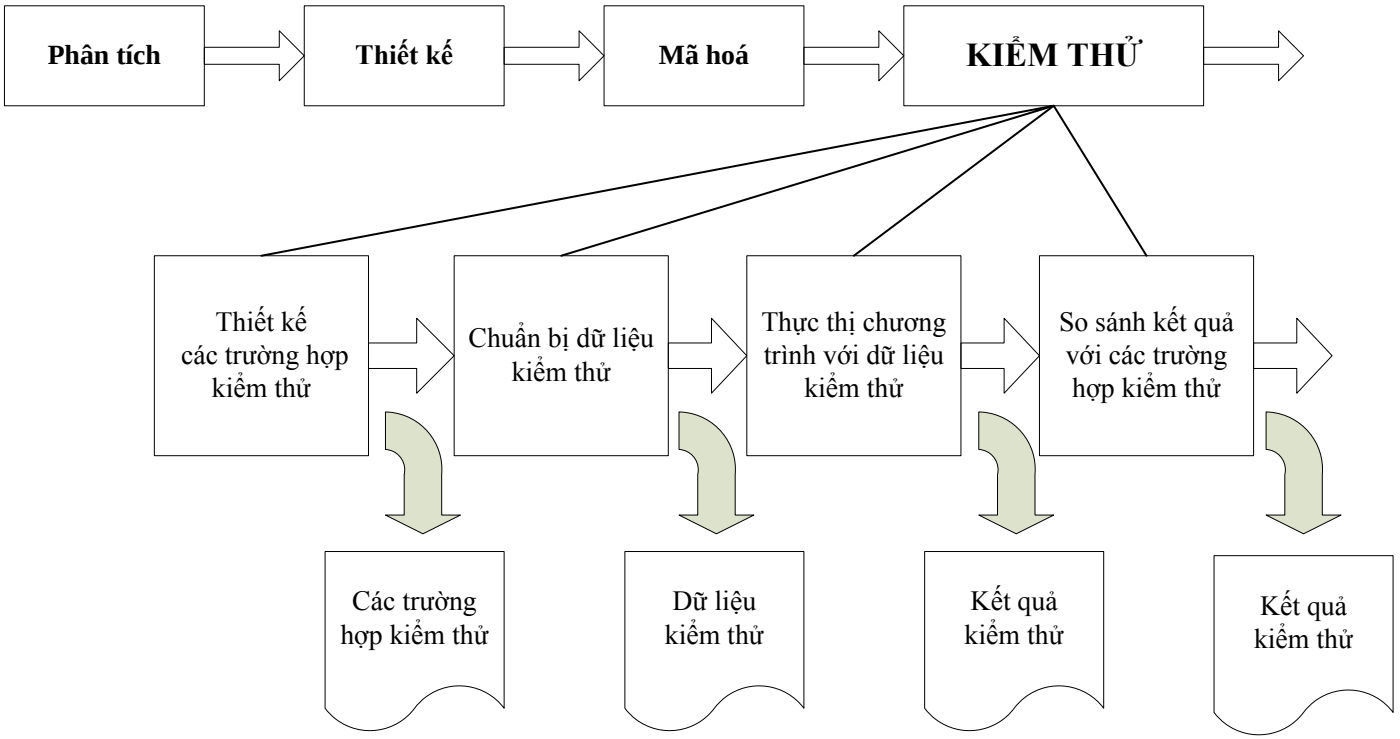
2.5.2 Kiểm thử phần mềm (KTPM)

Quy trình kiểm thử:

- Lập kế hoạch kiểm thử
- Bố trí nhân viên kiểm thử.
- Thiết kế các trường hợp kiểm thử.
- Xử lý đo lường kiểm thử bằng cách thu thập dữ liệu.
- Đánh giá sản phẩm phần mềm.

2.5.2 Kiểm thử phần mềm (KTPM)

Mô hình chung



2.5.2 KTPM (tt)


Dù LỖI nhỏ hay lớn, tớ vẫn sẽ tìm ra !
LỖI !!! 1 + 1 = 3 ???
Alo, LỖI !! Tôi có yêu cầu thế đâu !
LỖI !!! Không dùng được !

2.5.2 KTPM (tt)

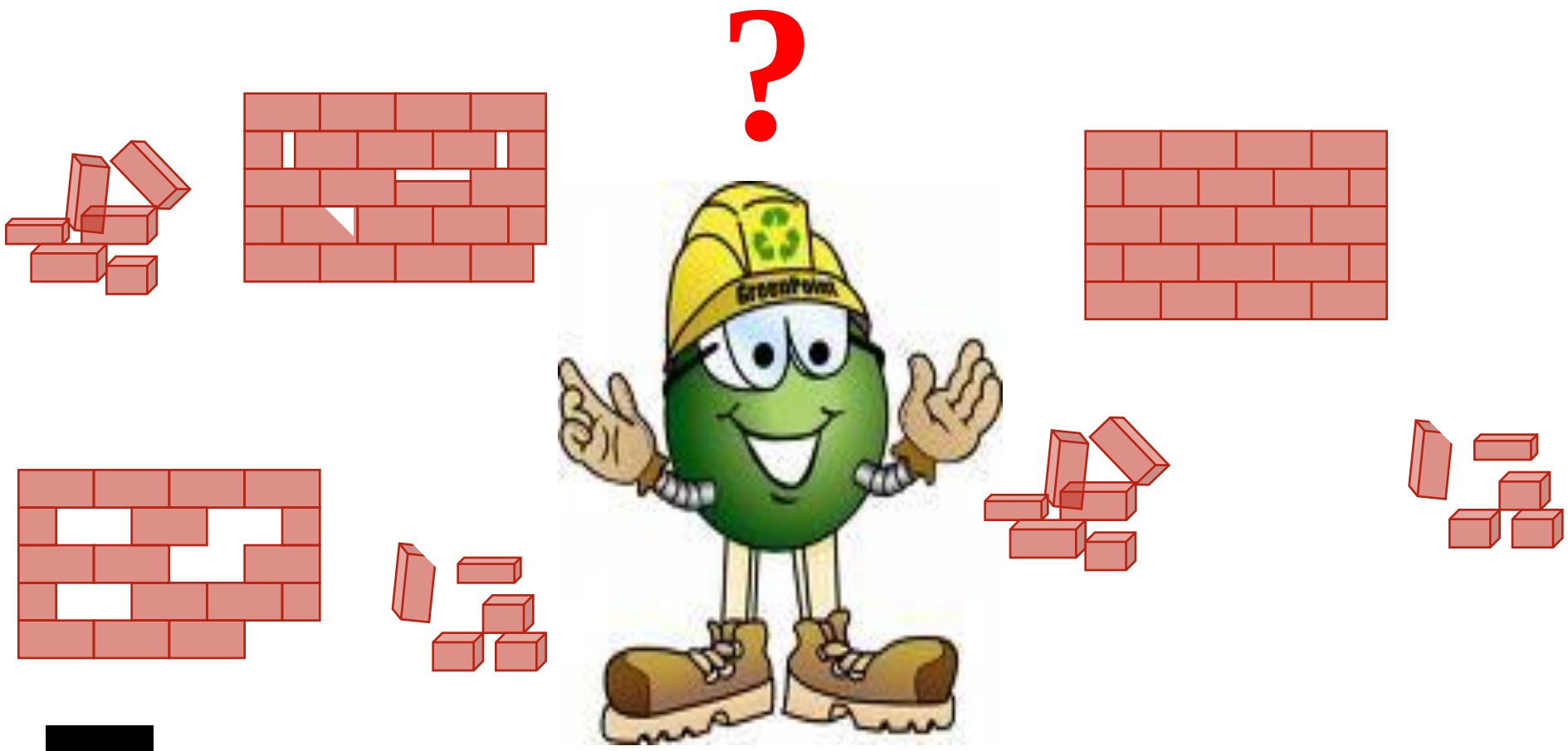
- **Theo quan điểm của người dùng:** Đảm bảo phần mềm đủ khả năng làm việc trong môi trường thực.
- **Sản phẩm của KTPM**: Bảng đánh giá về quá trình xây dựng phần mềm.
- **Vai trò của KTPM**: Công cụ tối quan trọng, quyết định đến việc đánh giá chất lượng phần mềm.

- **Kiểm thử viên:**
  - **. Giỏi chuyên môn nghiệp vụ**
  - **. Sáng tạo**
  - **. Tâm**

Các nguyên tắc cơ bản của KTPM

- Nguyên tắc **khách quan**
- Nguyên tắc **ngẫu nhiên**
- Nguyên tắc **"Người sử dụng kém"**
- Nguyên tắc **"Kẻ phá hoại"**

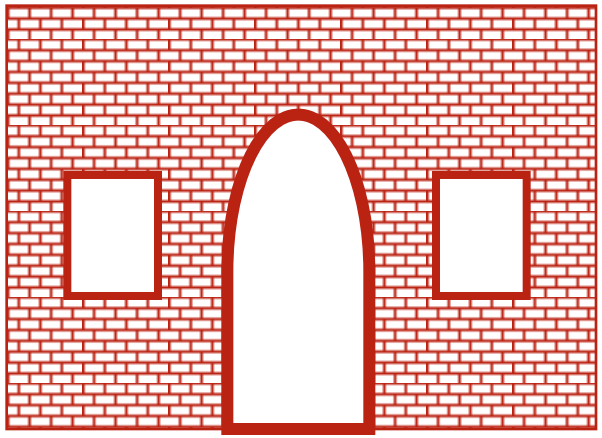
Các mức kiểm thử



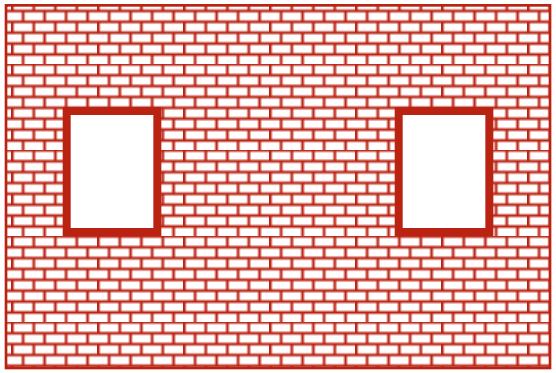
Kiểm thử đơn vị (Unit test)

- Thế nào là một đơn vị phần mềm ?
- Kiểm thử đơn vị:
  - . Riêng biệt từng đơn vị phần mềm
  - . Số lượng nhiều nhưng đơn giản
  - . Xuyên suốt thời gian lập trình và cả chu kỳ phần mềm
- Lập kế hoạch ngay khi lập trình

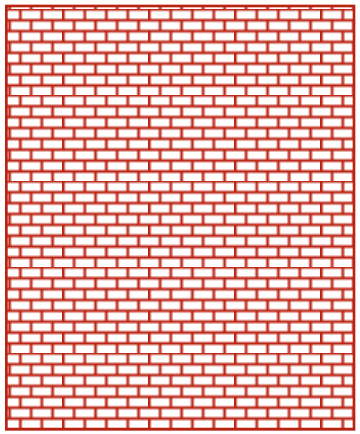
- Thợ A: Bức tường phía trước



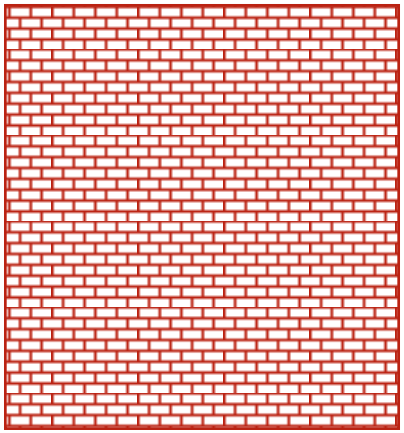
- Thợ B: Bức tường phía sau

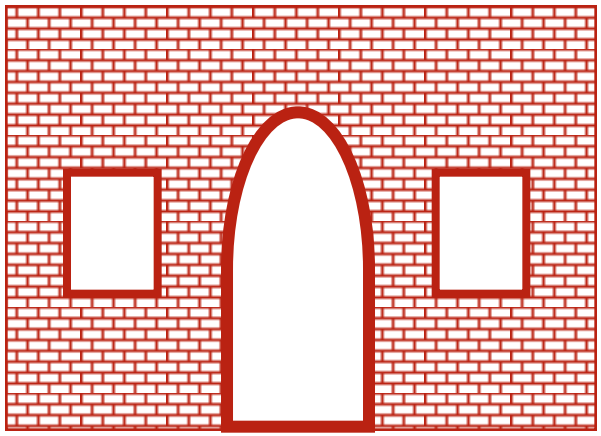


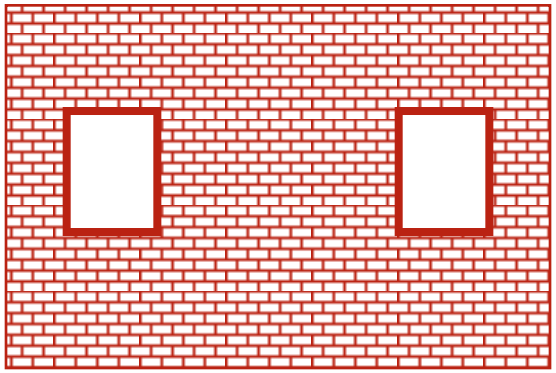
- Thợ C: Bức tường bên trái



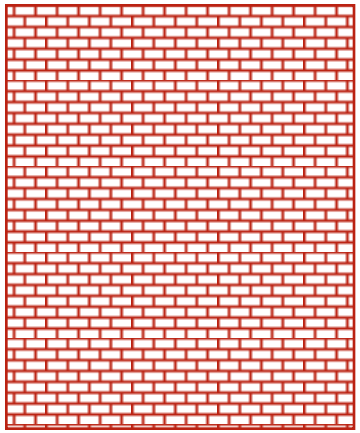
- Thợ C: Bức tường bên phải

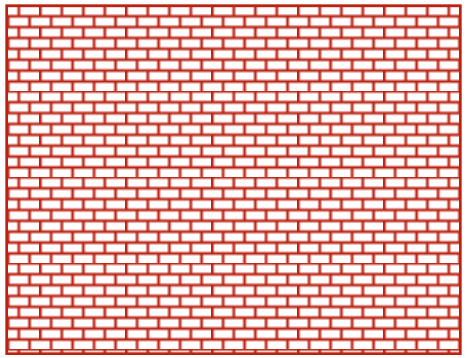






?

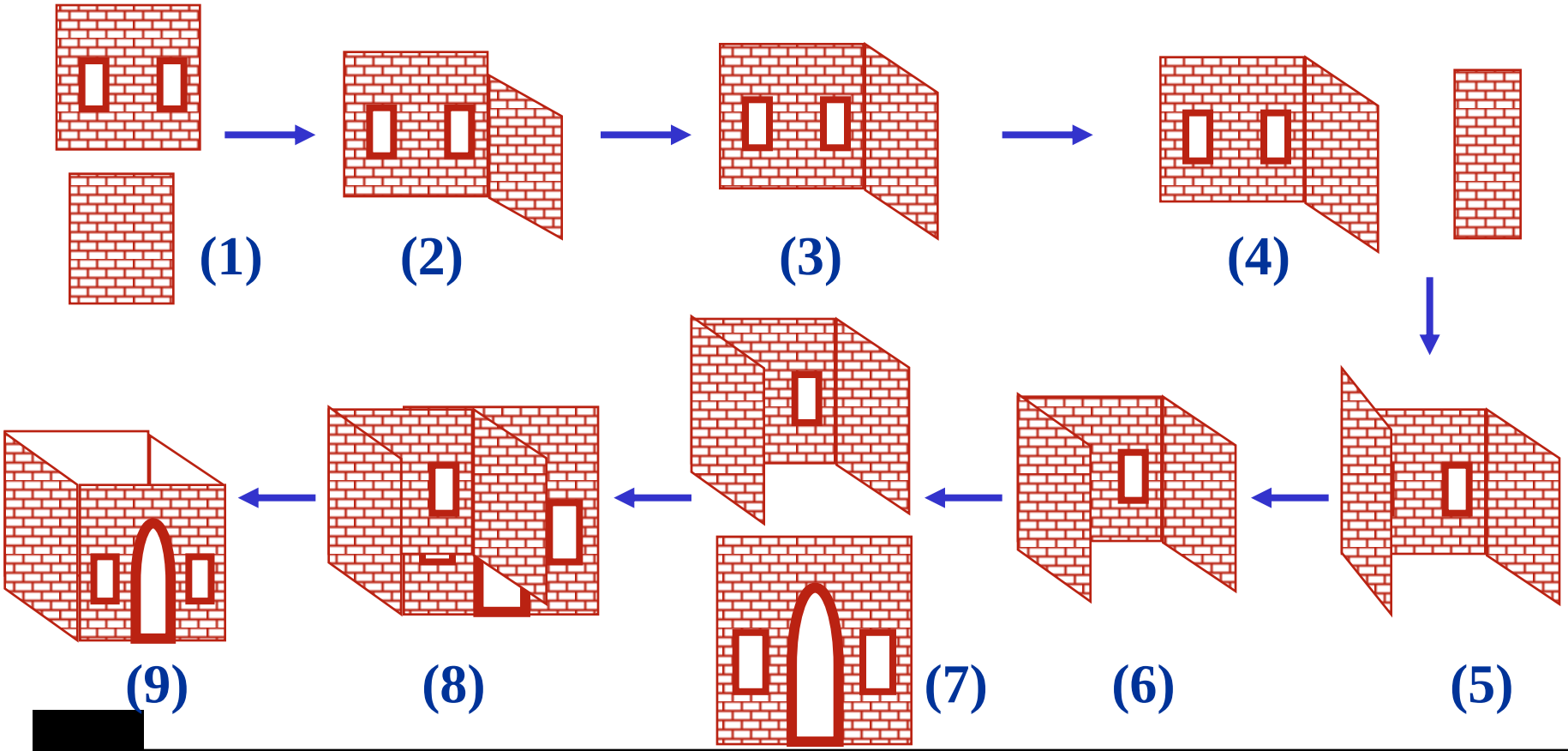




Kiểm thử tích hợp (Intergration test)

- Tích hợp dần các đơn vị phần mềm
- Phát hiện lỗi giao tiếp và sự không tương thích
- Tại mỗi thời điểm, chỉ tích hợp thêm 1 đơn vị phần mềm

Kiểm thử tích hợp (tt)



Kiểm thử tích hợp (tt)

- Có 4 loại kiểm thử cơ bản trong kiểm thử tích hợp
  - . Kiểm thử cấu trúc
  - . Kiểm thử chức năng
  - . Kiểm thử hiệu năng
  - . Kiểm thử khả năng chịu tải
- Lập kế hoạch khi thiết kế chi tiết

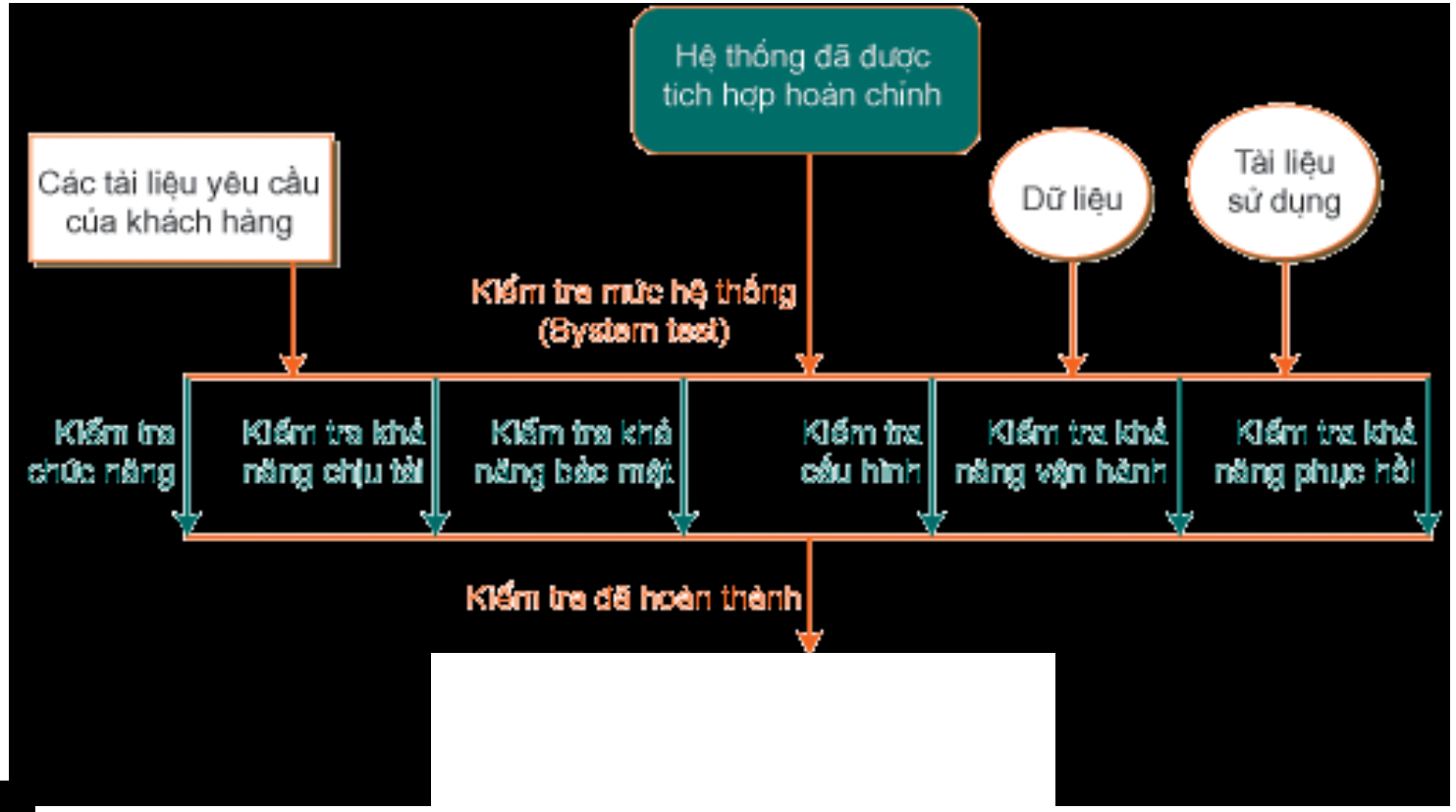
- Nắng nóng ?
- Mưa bão ?
- Thông gió ?
- Điện 220V ?
- PCCC ?

. . .

Kiểm thử hệ thống (System test)

- Toàn bộ hệ thống
  - . Chức năng phần mềm
  - . Giao tiếp với phần mềm/phần cứng bên ngoài
  - . Các yêu cầu về chất lượng
- Hệ thống phải được tích hợp thành công trước đó
- Lập kế hoạch khi thiết kế kiến trúc (thiết kế cấp cao)

Kiểm thử hệ thống (tt)




Xe ô-tô để ở đâu nhỉ ?
Nói chuyện với hàng xóm được không ?
Nhảy dây trên sàn có làm ồn không nhỉ ?
Đá bóng ?

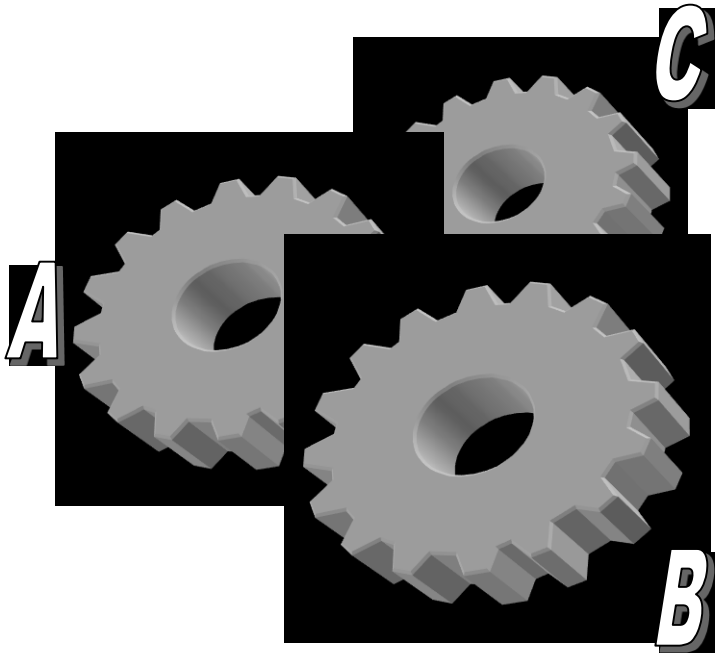
Kiểm thử chấp nhận (Acceptance test)

- Khách hàng thực hiện
- Tương tự như System Test nhưng khác nhau về bản chất
- Có hai loại kiểm thử là Alpha Test và Beta Test
- Kèm theo một nhóm những dịch vụ và tài liệu đi kèm như hướng dẫn cài đặt, sử dụng,…
- Lập kế hoạch khi nhận yêu cầu khách hàng

Tôi muốn sửa lại phòng khách lớn hơn ?

Vấn đề này …!!! ?

Kiểm thử hồi quy (Regression test)



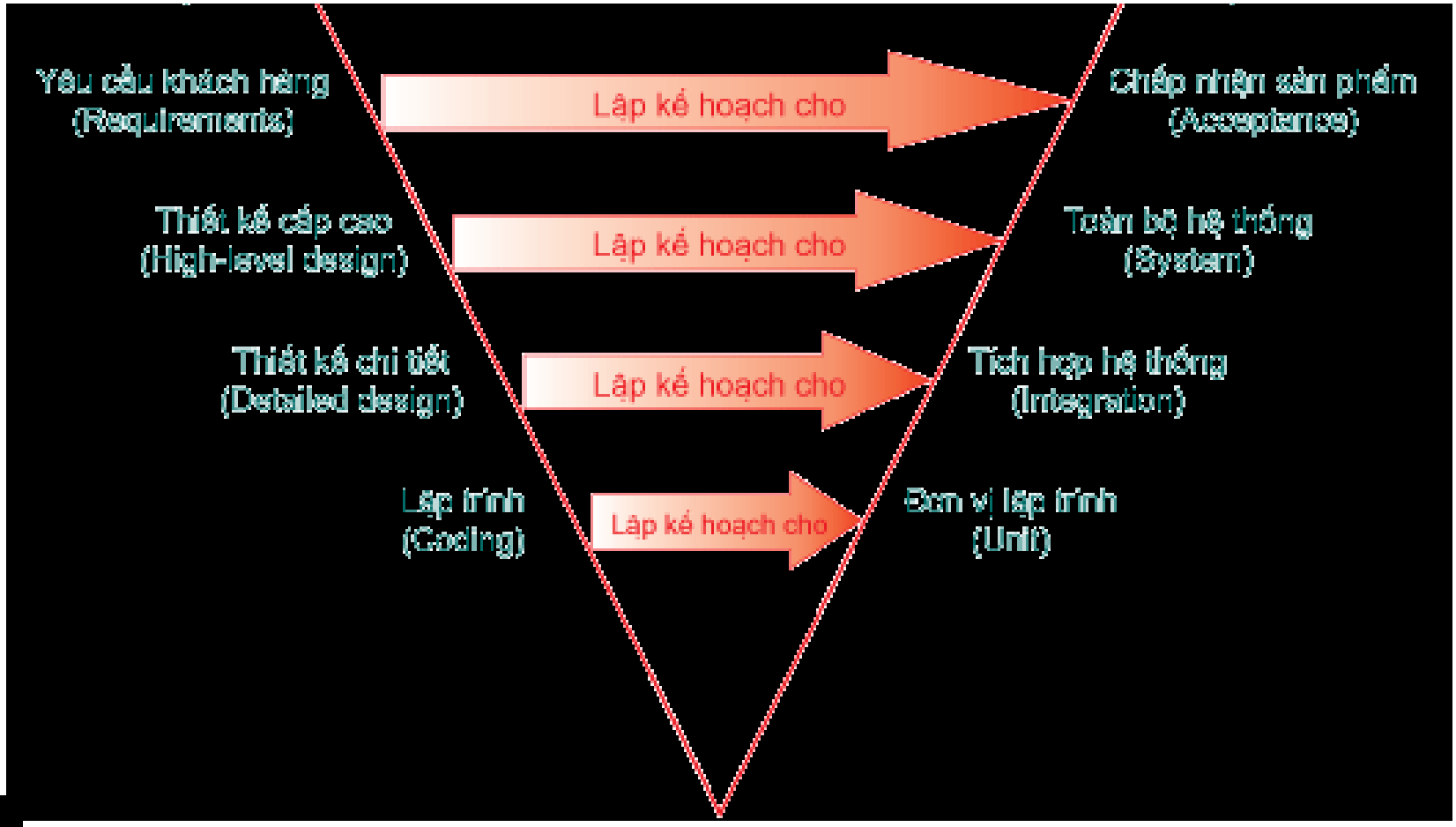
Điều gì sẽ xảy ra khi C có sự thay đổi ?

Kiểm thử hồi quy (tt)

- Không phải là một mức kiểm thử !
- Kiểm thử lại phần mềm khi có sự thay đổi xảy ra
- Có thể thực hiện tại mọi mức kiểm thử
- Không được phép bỏ qua nếu có sự thay đổi !

Phát triển phần mềm

Kiểm thử phần mềm



2.6 Bảo trì và cải tiến phần mềm

2.6.1 Bảo trì

- Hoạt động chỉnh sửa chương trình sau khi nó đã được đưa vào sử dụng
- Bảo trì là không thể tránh khỏi vì:

. Các yêu cầu hệ thống thường thay đổi

. Các hệ thống có gắn kết chặt chẽ với môi trường của nó.

. Các hệ thống phải được bảo trì nếu chúng muốn là những phần hữu ích trong môi trường nghiệp vụ.

Phân loại bảo trì:

- Bảo trì sửa lỗi
- Bảo trì tích hợp hệ thống vào một môi trường khác
- Bảo trì để bổ sung hoặc chỉnh sửa các yêu cầu chức năng của hệ thống

Các nhân tố anh hưởng đến bảo trì:

- Sự ổn định của đội dự án
- Những trách nhiệm đã cam kết
- Kỹ năng của nhân viên
- Tuổi thọ và cấu trúc chương trình

Các nhân tố anh hưởng đến bảo trì:

- Sự ổn định của đội dự án
- Những trách nhiệm đã cam kết
- Kỹ năng của nhân viên
- Tuổi thọ và cấu trúc chương trình

Các công việc bảo trì:

- Thiết lập cơ cấu bảo trì
- Báo cáo
- Lưu giữ các hồ sơ
- Xác định giá bảo trì

Các công việc bảo trì (tt)

Lưu giữ các hồ sơ gồm:

- Dấu hiệu nhận biết chương trình.
- Số lượng các câu lệnh trong chương trình nguồn.
- Số lượng các lệnh mã máy.
- Ngôn ngữ lập trình được sử dụng.
- Ngày cài đặt chương trình.
- Số các chương trình chạy từ khi cài đặt.
- Số các lỗi xử lý xảy ra.
- Mức và dấu hiệu thay đổi chương trình.
- Số các câu lệnh được thêm vào chương trình nguồn khi chương trình thay đổi.

Các công việc bảo trì (tt)

Lưu giữ các hồ sơ gồm (tt):

- Số các câu lệnh được xóa khỏi chương trình nguồn khi chương trình thay đổi.
- Số giờ mỗi người sử dụng cho mỗi lần sửa đổi.
- Ngày thay đổi chương trình.
- Dấu hiệu của kỹ sư phần mềm.
- Dấu hiệu của đơn yêu cầu bảo trì.
- Kiểu bảo trì.
- Ngày bắt đầu và kết thúc bảo trì.
- Tổng số giờ của mỗi người dùng cho việc bảo trì.

Các công việc bảo trì:

Xác định giá bảo trì bao gồm:

- Số lượng trung bình các lỗi xử lý cho một lần chạy chương trình.
- Tổng số giờ của mỗi người dùng cho mỗi loại bảo trì.
- Số lượng trung bình các thay đổi theo chương trình, theo ngôn ngữ lập trình, theo kiểu bảo trì.
- Số giờ trung bình của mỗi người cho một dòng lệnh được thêm vào và xóa đi.
- Số giờ trung bình của mỗi người cho một ngôn ngữ lập trình.
- Thời gian trung bình cho việc bảo trì một đơn yêu cầu bảo trì.
- Tỷ lệ phần trăm của mỗi kiểu bảo trì.

2.6 Bảo trì và cải tiến phần mềm

2.6.1 Cải tiến

Phải cải tiến vì thay đổi phần mềm là một điều không thể tránh khỏi vì những lí do sau:

- Những yêu cầu mới sẽ xuất hiện khi sử dụng phần mềm
- Môi trường nghiệp vụ thay đổi
- Các lỗi phần mềm cần phải sửa chữa
- Máy tính và các thiết bị mới được bổ sung vào hệ thống
- Hiệu năng hoặc độ tin cậy của hệ thống phải được cải thiện.

2.7 Quản lý thay đổi phần mềm

- Các ứng dụng thường xuyên phải thiết kế lại
- Các thay đổi có thể là các yêu cầu, thiết kế, chương trình, giao diện, phần cứng hoặc phần mềm phải mua.
- Việc quản lý thay đổi ứng dụng giúp cho nhóm triển khai bỏ qua những ý thích chợt nảy ra của người sử dụng trong khi vẫn cho phép thực hiện các yêu cầu hợp lý

Chương 3

MÔ HÌNH PHÁT TRIỂN PHẦN MỀM

* Qui trình phát triển/xây dựng phần mềm (Software Development/Engineering Process - SEP) có tính chất quyết định để tạo ra sản phẩm chất lượng tốt với chi phí thấp và năng suất cao

Thông thường một quy trình bao gồm các yếu tố cơ bản sau:

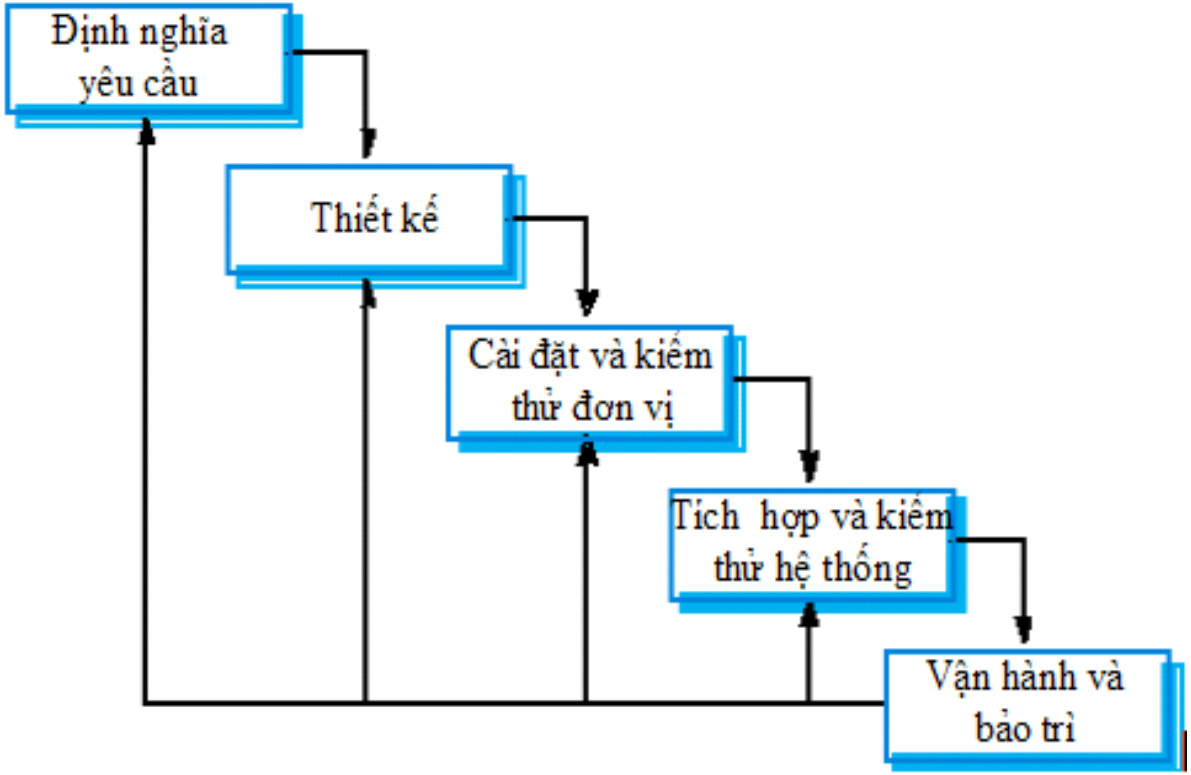
- Thủ tục (Procedures)
- Hướng dẫn công việc (Activity Guidelines)
- Biểu mẫu (Forms/templates)
- Danh sách kiểm định (Checklists)
- Công cụ hỗ trợ (Tools)

Và các nhóm công việc chính:

- Đặc tả yêu cầu (Requirements Specification)
- Phát triển phần mềm (Development)
- Kiểm thử phần mềm (Validation/Testing)
- Thay đổi phần mềm (Evolution)

3.1 Các mô hình một phiên bản

Mô hình thác nước



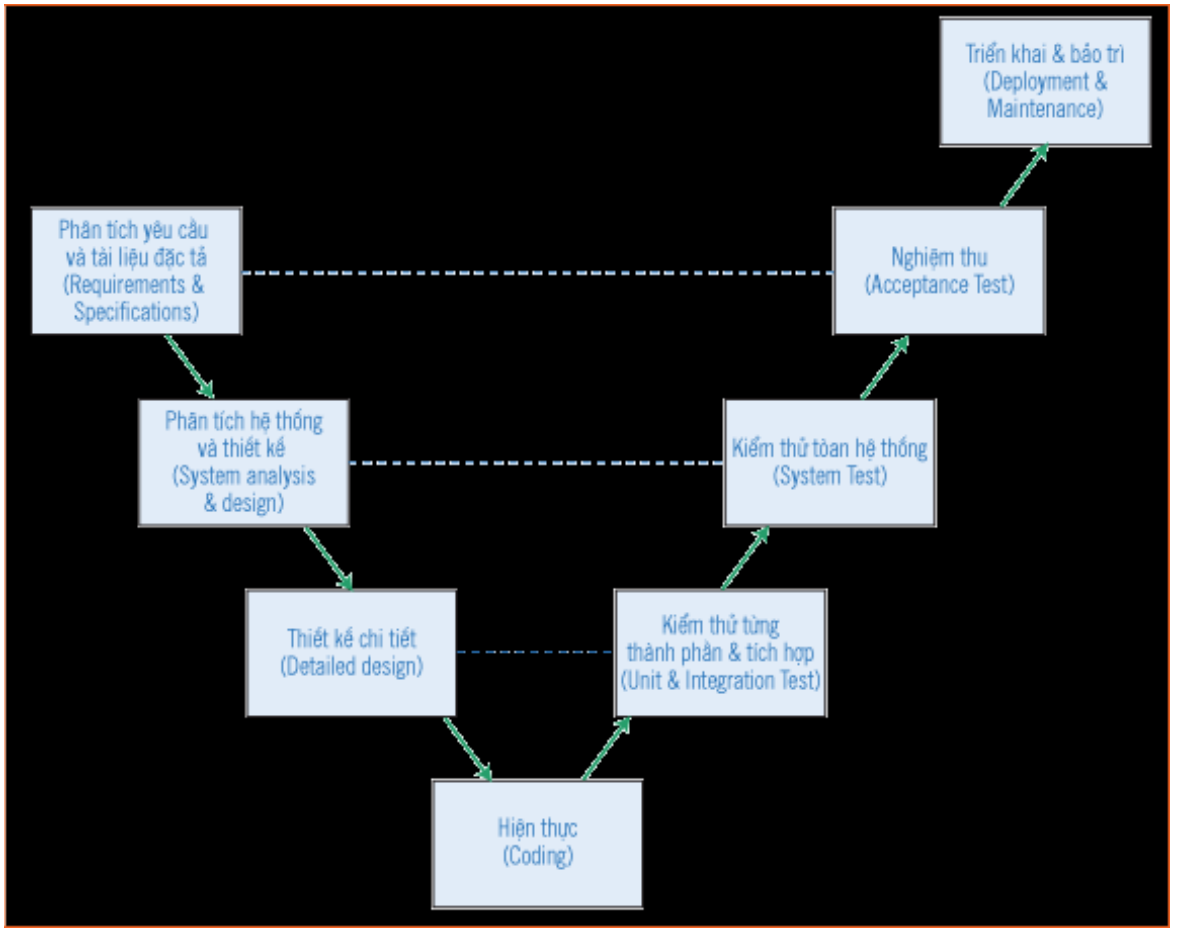
3.1 Các mô hình một phiên bản

Nhược điểm của mô hình thác nước:

- Mô hình đòi hỏi một bản yêu cầu (requirement) đầy đủ và chính xác từ phía khách hàng
- Khách hàng chỉ được tham gia vào dự án ở giai đoạn phân tích yêu cầu và kiểm thử mà thôi
- nên được sử dụng khi đội dự án đã có kinh nghiệm, yêu cầu từ khách hàng được xác định rõ ngay từ đầu và ít có khả năng thay đổi

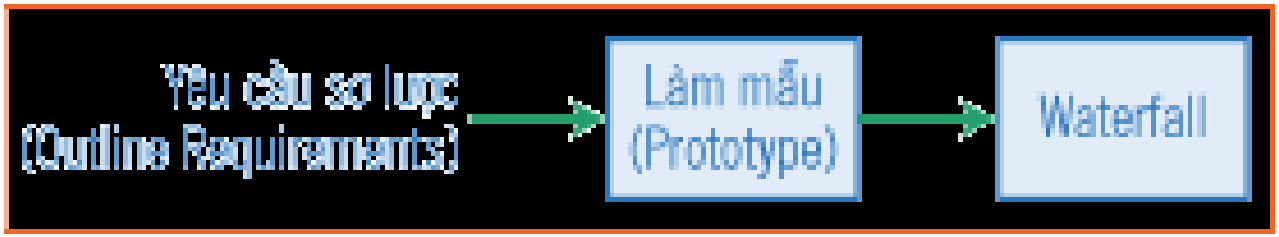
3.1 Các mô hình một phiên bản

Mô hình chữ V



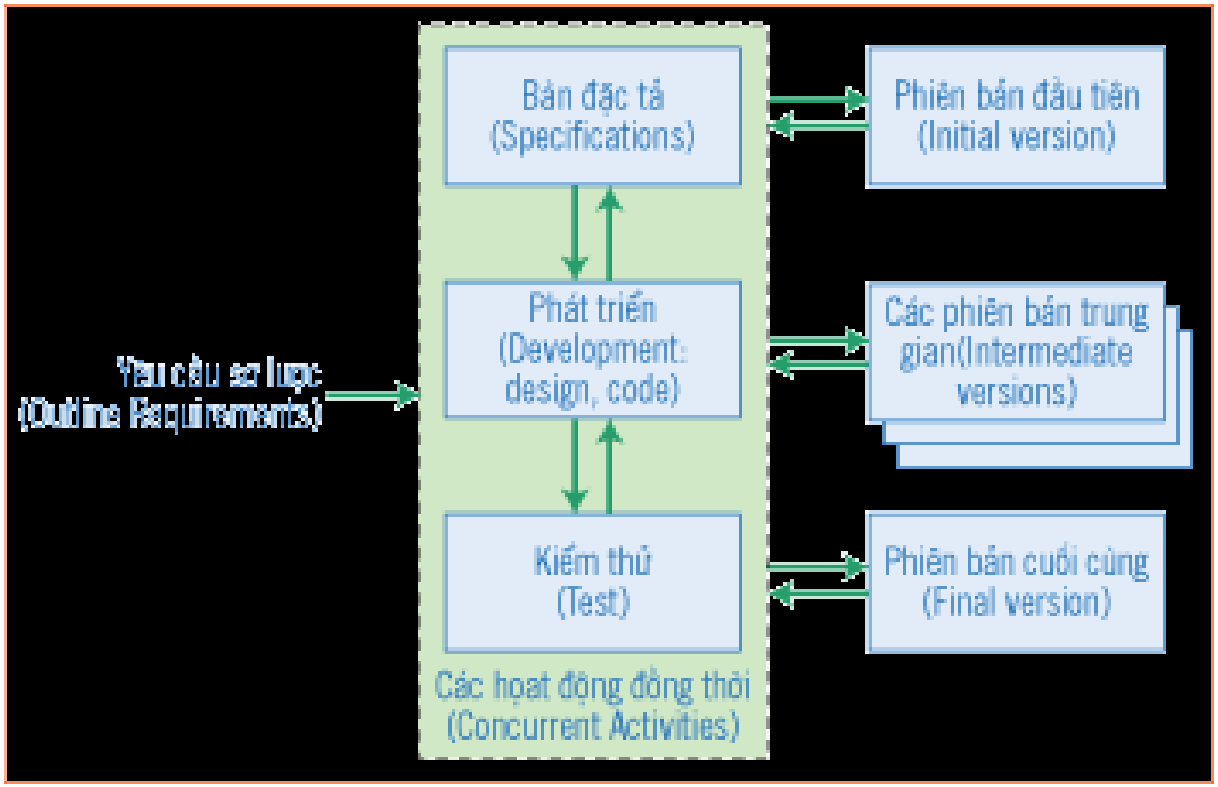
3.1 Các mô hình nhiều phiên bản

Mô hình mẫu



3.1 Các mô hình nhiều phiên bản

Mô hình tiến hóa



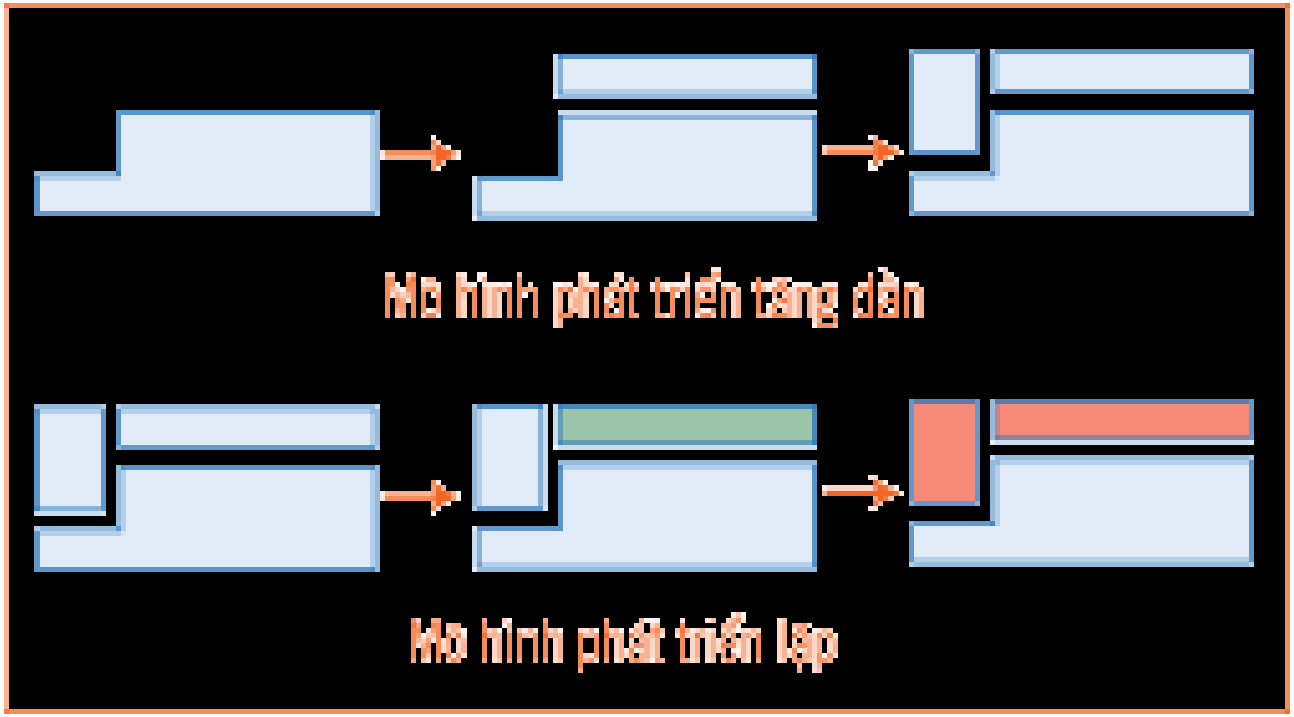
3.1 Các mô hình nhiều phiên bản

Mô hình tiến hóa thực sự cũng là một dạng dựa trên mô hình mẫu, tuy nhiên có sự khác biệt như sau:

- **Mô hình tiến hóa xây dựng nhiều phiên bản mẫu liên tiếp nhau.**
- **Những phiên bản mẫu trước sẽ được xây dựng với mục tiêu có thể tái sử dụng trong những phiên bản sau.**

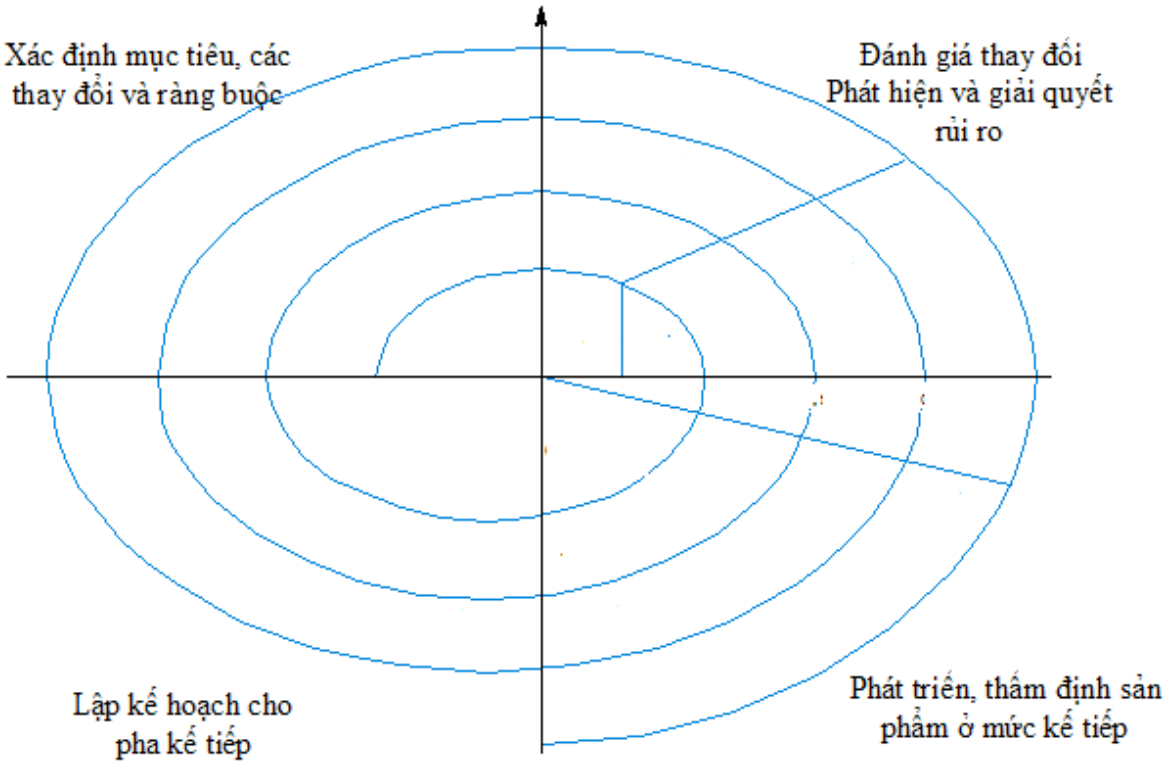
3.1 Các mô hình nhiều phiên bản

Mô hình phát triển lặp lại và tăng dần: Dựa trên mô hình tiến hóa



3.1 Các mô hình nhiều phiên bản

Mô hình xoắn ốc



3.1 Các mô hình nhiều phiên bản

Ngoài ra còn có: Mô hình phát triển ứng dụng nhanh (RAD), Mô hình hướng đối tượng, Mô hình Công nghệ phần mềm dựa thành phần, …

NHẬP MÔN
CÔNG NGHỆ PHẦN MỀM

Giảng viên: Đỗ Thị Thanh Tuyền

Email: dothithanhtuyen@gmail.com

Nội dung môn học

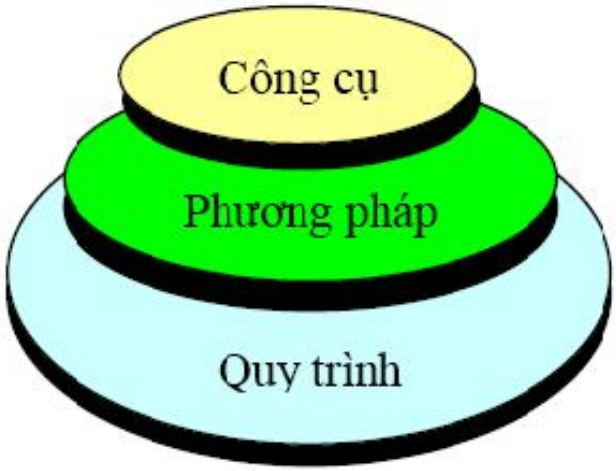
- Tổng quan về Công nghệ phần mềm
- Xác định và mô hình hóa yêu cầu phần mềm
- Thiết kế phần mềm
- Cài đặt phần mềm
- Kiểm thử và bảo trì
- Đồ án môn học

Một số khái niệm cơ bản

- **Phần mềm:** là một tập hợp những câu lệnh được viết bằng một hoặc nhiều ngôn ngữ lập trình theo một trật tự xác định nhằm tự động thực hiện một số chức năng hoặc giải quyết một bài toán nào đó. Phần mềm được thực thi trên máy, thường là máy tính.
- **Công nghệ (engineering):** là cách sử dụng các công cụ, các kỹ thuật trong cách giải quyết một vấn đề nào đó.
- **Công nghệ Phần mềm (Software Engineering):** là việc áp dụng các công cụ, các kỹ thuật một cách có hệ thống trong việc phát triển các ứng dụng dựa trên máy tính.

Một số khái niệm cơ bản (tt)

- Công nghệ Phần mềm có thể được mô hình hóa như sau:



- **Qui trình Công nghệ Phần mềm:** hệ thống các giai đoạn mà quá trình phát triển phần mềm phải trải qua.
- **Phương pháp phát triển phần mềm:** phương pháp thực hiện cho từng giai đoạn trong qui trình phát triển phần mềm.

Một số khái niệm cơ bản (tt)

- **Công cụ và môi trường phát triển phần mềm:** các phương tiện hỗ trợ tự động hay bán tự động cho một giai đoạn nào đó trong quá trình xây dựng phần mềm.

Phân loại phần mềm

- **Phân loại theo phương thức hoạt động:**
  - **Phần mềm hệ thống:** hệ điều hành, thư viện liên kết động, bộ điều vận (driver)...
  - **Phần mềm ứng dụng:** phần mềm văn phòng, phần mềm doanh nghiệp, phần mềm giáo dục, phần mềm giải trí...
  - **Phần mềm chuyển dịch mã:** bao gồm trình biên dịch và trình thông dịch.
- **Phân loại theo khả năng ứng dụng:**
  - Phần mềm được viết theo đơn đặt hàng của một khách hàng cụ thể: phần mềm hỗ trợ bán hàng, phần mềm điều khiển...

    - ***Ưu điểm:*** có tính uyển chuyển, tùy biến cao để đáp ứng được nhu cầu của một nhóm người sử dụng.

Phân loại phần mềm (tt)

- **Phân loại theo khả năng ứng dụng (tt):**

  - ***Khuyết điểm:*** ứng dụng trong chuyên ngành hẹp.

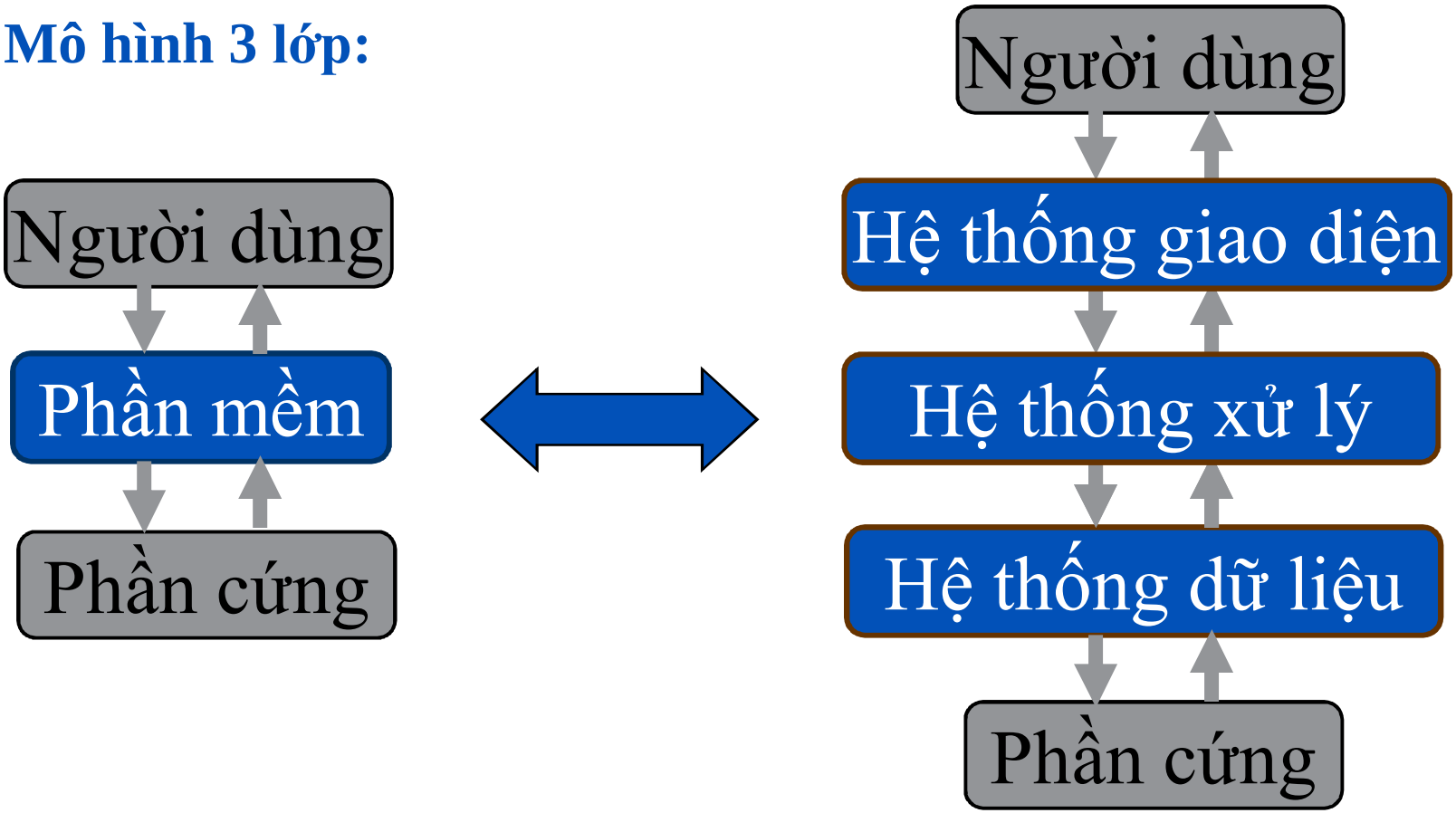
  - Phần mềm không được viết theo một đơn đặt hàng cụ thể, nó có thể được bán cho bất kỳ khách hàng nào.

  - ***Ưu điểm:*** có khả năng ứng dụng rộng rãi cho nhiều nhóm người sử dụng.

  - ***Khuyết điểm:*** thiếu tính uyển chuyển, tùy biến.

Kiến trúc phần mềm

Mô hình 3 lớp:



Chất lượng phần mềm

Chất lượng phần mềm thể hiện qua các tính chất sau:

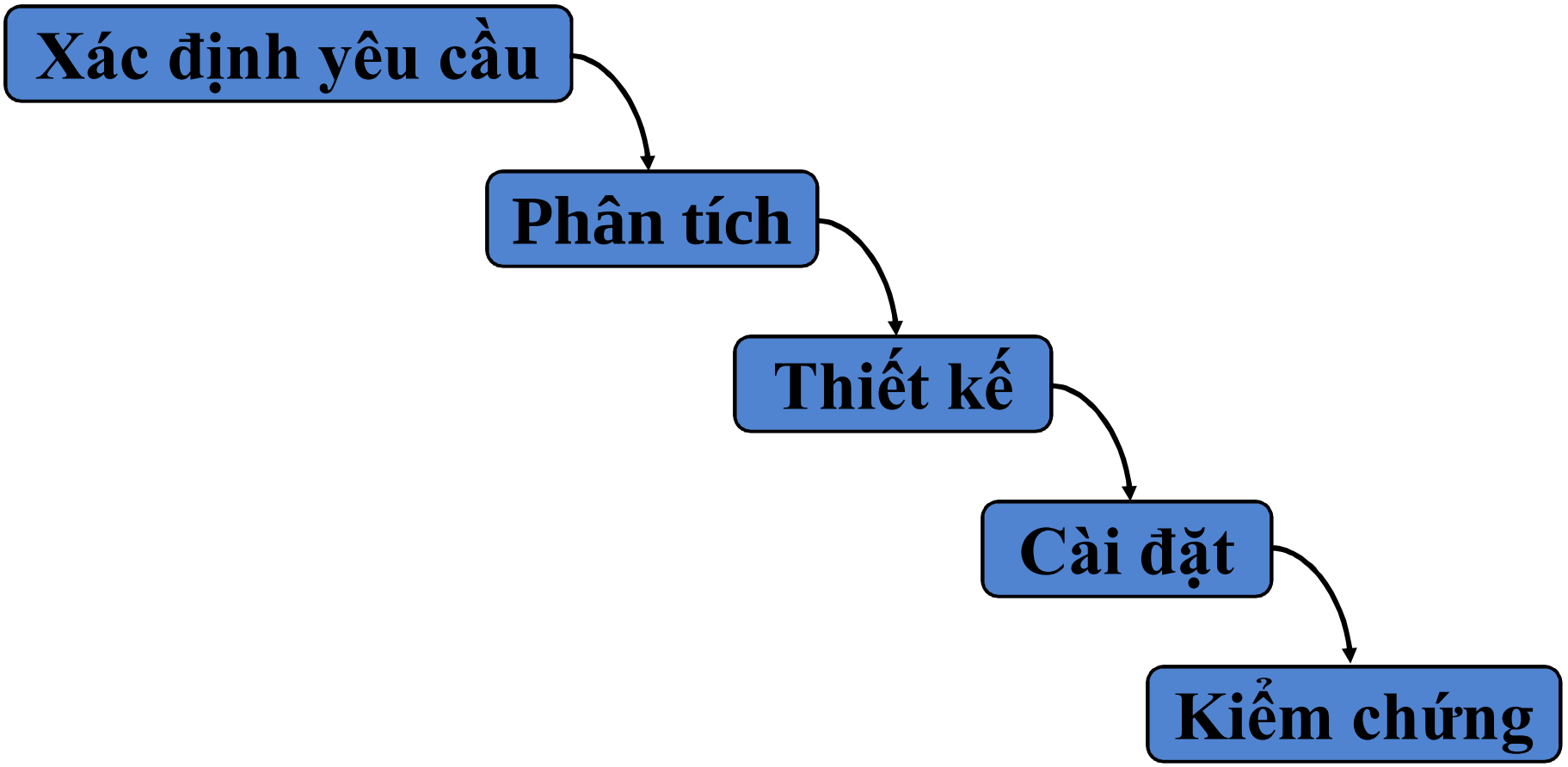
- Tính đúng đắn
- Tính tiến hóa
- Tính tiện dụng
- Tính hiệu quả
- Tính tương thích

Qui trình Công nghệ Phần mềm

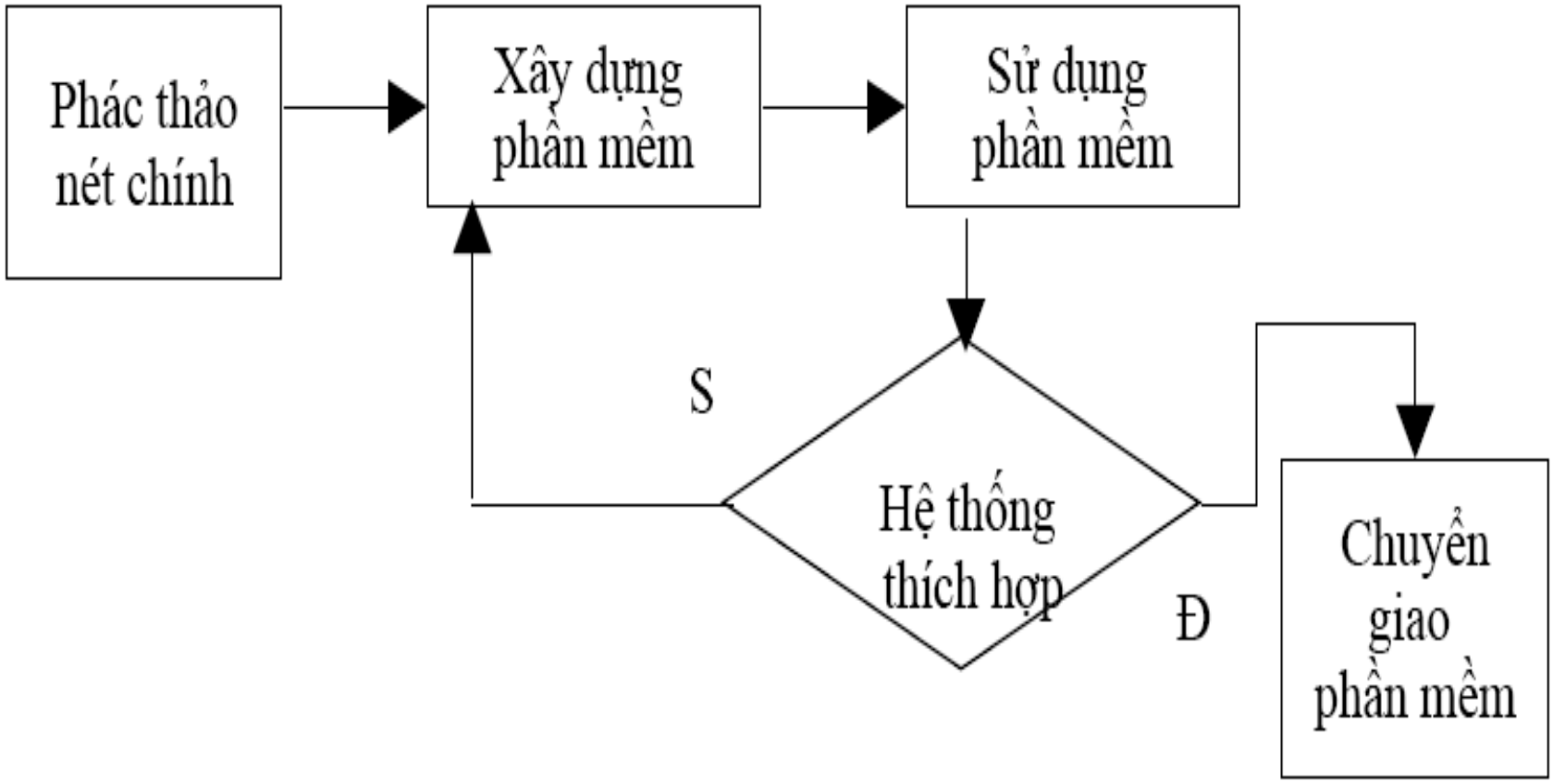
- **Mô hình thác nước**
- **Mô hình mẫu**
- **Mô hình xoắn ốc**

=> **Mô hình thác nước cải tiến**

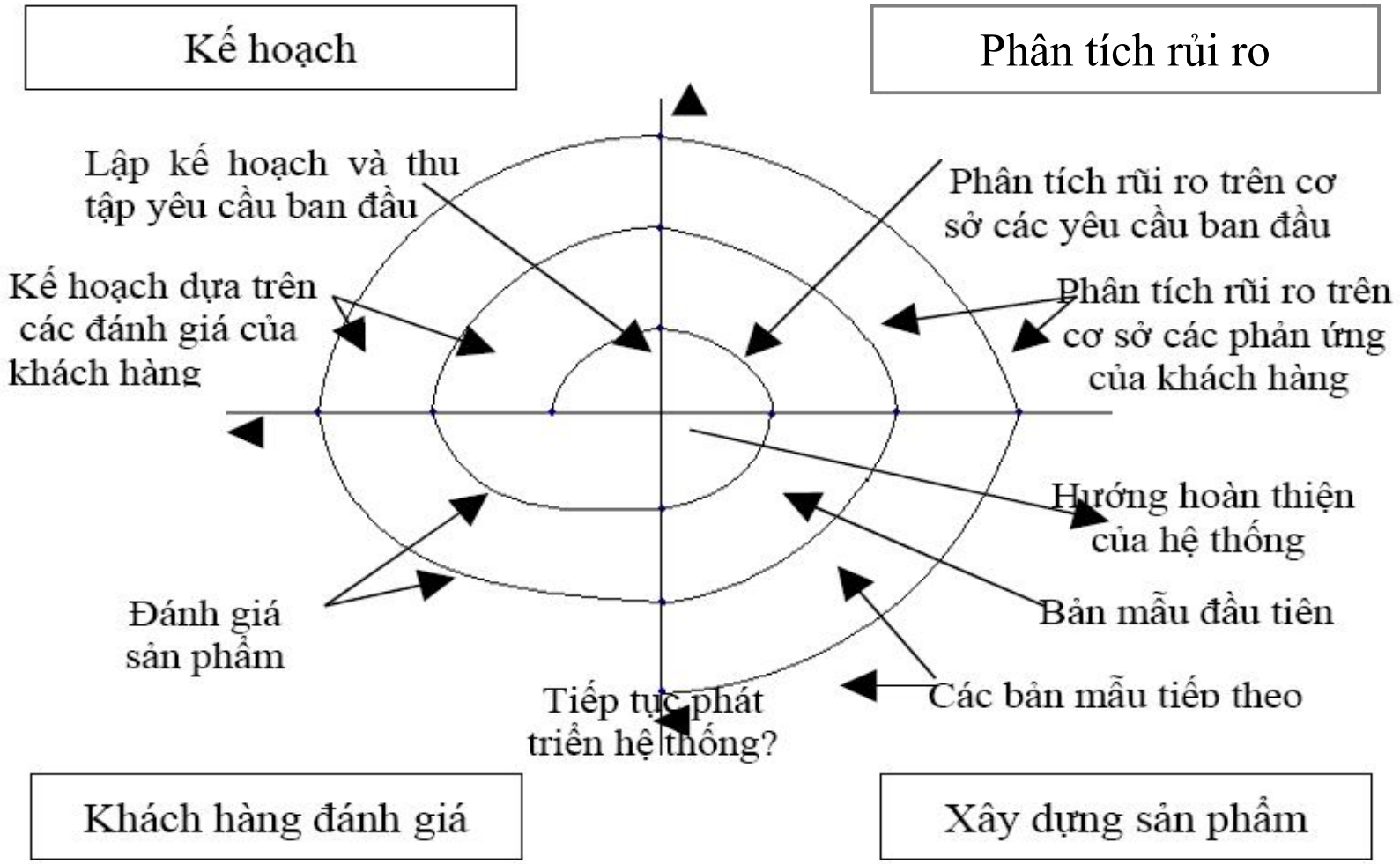
Mô hình thác nước cổ điển



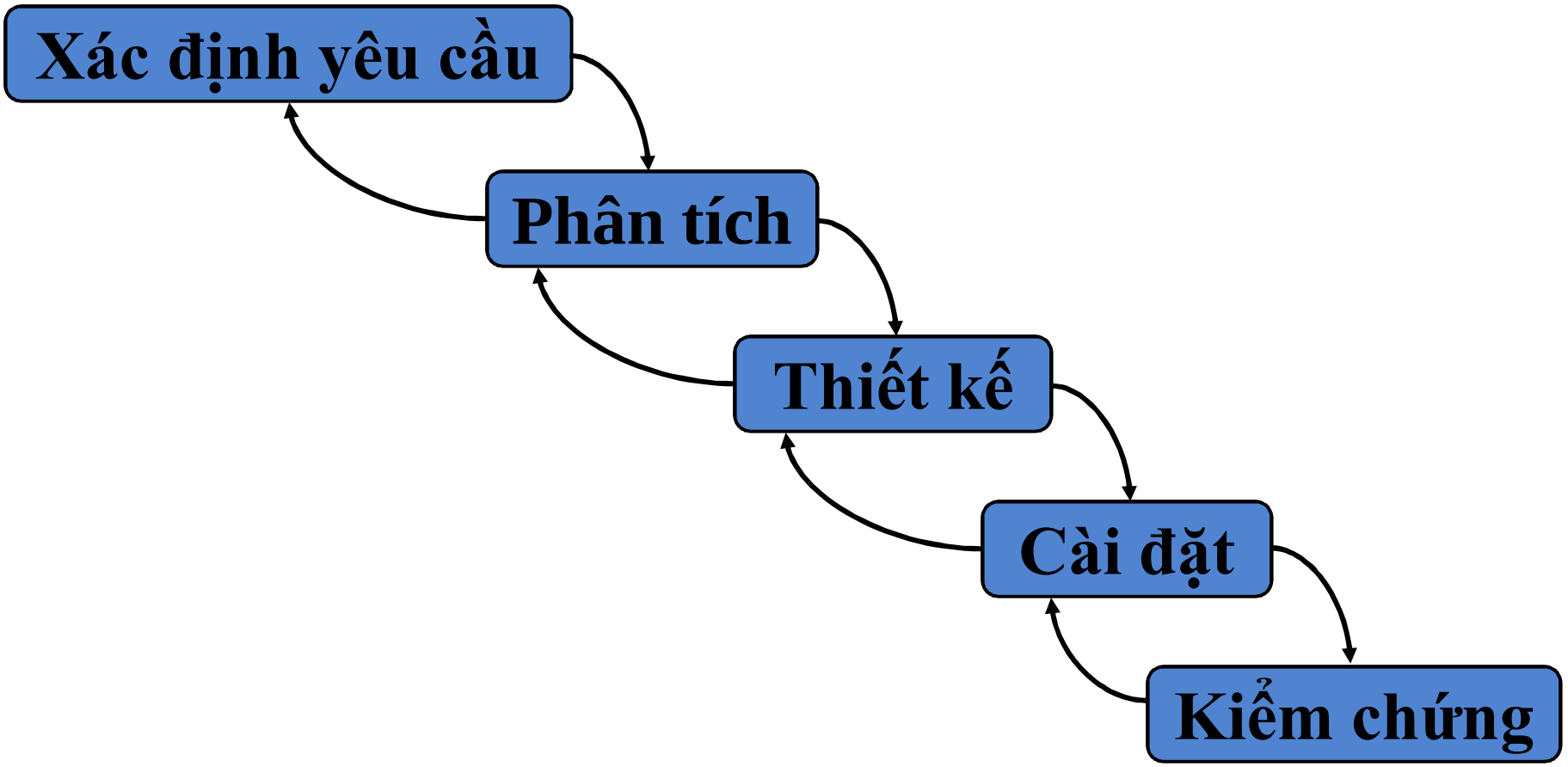
Mô hình mẫu


Phác thảo nét chính
Xây dựng phần mềm
Sử dụng phần mềm
S
Hệ thống thích hợp
Đ
Chuyển giao phần mềm

Mô hình xoắn ốc



Mô hình thác nước cải tiến



Phương pháp phát triển phần mềm

- **Phương pháp hướng chức năng:**
  - Xây dựng phần mềm dựa trên các chức năng mà hệ thống cần thực hiện.
  - Phương pháp chung để giải quyết vấn đề là áp dụng nguyên lý "chia để trị".
  - ***Hạn chế:*** có khả năng các chức năng trong hệ thống không tương thích với nhau khi thực hiện thay đổi các thông tin trong hệ thống.

Phương pháp phát triển phần mềm (tt)

- **Phương pháp hướng dữ liệu:**
  - Chú trọng đến thành phần dữ liệu của hệ thống.
  - Dùng mô hình thực thể kết hợp để biểu diễn các thực thể và mối liên hệ giữa các thực thể.
  - ***Hạn chế:*** phần mềm chỉ có chức năng chính là lưu trữ và thao tác trên các đối tượng dữ liệu, không quan tâm đến các chức năng khác của hệ thống nên hệ thống thu được sau khi thiết kế có thể thiếu một số chức năng cần thiết.

Phương pháp phát triển phần mềm (tt)

- **Phương pháp hướng đối tượng:**
  - Chú trọng đến thành phần dữ liệu và chức năng của hệ thống.
  - Hệ thống phần mềm là một tập hợp các đối tượng có khả năng tương tác với nhau.
  - Mỗi đối tượng bao gồm dữ liệu và các thao tác thực hiện trên dữ liệu của đối tượng.

Công cụ và môi trường phát triển PM

- CASE (Computer Aided Software Engineering) tools.
- CASE tools hỗ trợ phát sinh kết quả chuyển giao cho giai đoạn kế tiếp.
- CASE tools hỗ trợ việc lưu trữ, cập nhật trên kết quả chuyển giao.

Công cụ và môi trường phát triển PM (tt)

Giai đoạn phát triển	Công việc hỗ trợ	Phần mềm
Phân tích	- Soạn thảo mô hình thế giới thực. - Ánh xạ vào mô hình logic.	- Analyst Pro
Thiết kế	- Soạn thảo mô hình logic. - Ánh xạ vào mô hình vật lý.	- Power Designer
Cài đặt	- Quản lý các phiên bản. - Biên dịch.	- Visual Studio
Kiểm chứng	- Phát sinh tự động các bộ dữ liệu thử nghiệm. - Phát hiện lỗi.	- WinRuner

Công cụ và môi trường phát triển PM (tt)

Quản lý dự án	Công việc hỗ trợ	Phần mềm
Xây dựng phương án	- Tạo lập phương án. - Dự đoán rủi ro. - Tính chi phí.	- Microsoft Project
Lập kế hoạch	- Xác định các công việc. - Phân công. - Lập lịch biểu. - Theo dõi thực hiện.	- Microsoft Project

Quá trình phát triển

- **Thập niên 1940:** Các chương trình cho máy tính được viết bằng tay.
- **Thập niên 1950:** Các công cụ đầu tiên xuất hiện như phần mềm biên dịch Macro Assembler và phần mềm thông dịch đã được tạo ra và sử dụng rộng rãi để nâng cao năng suất và chất lượng. Các trình dịch được tối ưu hóa lần đầu tiên ra đời.
- **Thập niên 1960:** Các công cụ của thế hệ thứ hai như các trình dịch tối ưu hoá và công việc kiểm tra mẫu đã được dùng để nâng cao sản phẩm và chất lượng. Khái niệm công nghệ phần mềm đã được bàn thảo rộng rãi.
- **Thập niên 1970:** Các công cụ phần mềm, chẳng hạn trong UNIX các vùng chứa mã, lệnh make...được kết hợp với nhau. Số lượng doanh nghiệp nhỏ về phần mềm và số lượng máy tính cỡ nhỏ tăng nhanh.

Quá trình phát triển

- **Thập niên 1980:** Các PC và máy trạm ra đời. Cùng lúc có sự xuất hiện của mô hình dự toán khả năng. Lượng phần mềm tiêu thụ tăng mạnh.
- **Thập niên 1990:** Phương pháp lập trình hướng đối tượng ra đời. Các quá trình nhanh như là lập trình cực hạn được chấp nhận rộng rãi. Trong thập niên này, WWW và các thiết bị máy tính cầm tay phổ biến rộng rãi.
- **Hiện nay:** Các phần mềm biên dịch và ngôn ngữ lập trình cấp cao như .NET, PHP và Java làm cho việc viết phần mềm trở nên dễ dàng hơn nhiều.

Quá trình phát triển (tt)

- **Hướng tương lai của công nghệ phần mềm:**

- ***Lập trình định dạng (aspect-oriented programming)*** sẽ giúp người lập trình ứng xử với các yêu cầu không liên quan đến các chức năng thực tế của phần mềm bằng cách cung ứng các công cụ để thêm hay bớt các khối mã ít bị thay đổi trong nhiều vùng của mã nguồn.

  Lập trình định dạng mô tả các đối tượng và hàm nên ứng xử như thế nào trong một tình huống cụ thể.

  Ví dụ: Lập trình định dạng có thêm vào các cơ cấu kiểm soát hiệu chỉnh lỗi, biên bản và khoá cho tất cả các đối tượng của một số kiểu. Các nhà nghiên cứu đang tìm cách ứng dụng lập trình định dạng để thiết kế mã cho mục tiêu thông thường.

- ***Phát triển phần mềm linh hoạt:*** nhằm hướng dẫn các đề án phát triển phần mềm mà trong đó bao gồm việc thỏa mãn các nhu cầu thay đổi và sự cạnh tranh của thị trường một cách nhanh chóng.

Q & A

Giáo trình tóm tắt Công nghệ phần mềm

MỤC LỤC

MỞ ĐẦU

Sau gần nửa thế kỷ phát triển, ngành kỹ nghệ phần mềm (SE – Software Engineering) đến nay đã được thừa nhận là một bộ môn chính thống. Các phương pháp, thủ tục và công cụ kỹ nghệ phần mềm đã được chấp nhận và ứng dụng thành công trong rất nhiều lĩnh vực công nghiệp. Các nhà quản lý và chuyên gia công nghệ thông tin đều nhận ra nhu cầu về cách tiếp cận có nguyên tắc hơn tới việc phát triển phần mềm.

Mục đích của ngành kỹ nghệ phần mềm không phải là việc sản sinh ra phần mềm cụ thể mà là việc *sản sinh ra các sản phẩm một cách hiệu quả* với hạn chế về nguồn lực và thời gian.

Giáo trình Nhập môn Kỹ nghệ phần mềm trang bị cho sinh viên khoa Công nghệ thông tin những khái niệm cơ bản về phần mềm và cách chế tạo phần mềm; giúp sinh viên tiếp cận có nguyên tắc hơn tới việc phát triển phần mềm thông qua các phương pháp, thủ tục và công cụ của kỹ nghệ phần mềm và cuối cùng là x*ây dựng* phần mềm một cách hiệu quả.

CHƯƠNG 1

PHẦN MỀM VÀ KỸ NGHỆ PHẦN MỀM

0.Đối tượng nghiên cứu của môn học

Mục đích của môn học "Công nghệ phần mềm" không phải là để sản sinh ra phần mềm cụ thể mà nó liên quan đến việc *sản sinh ra sản phẩm một cách hiệu quả.*

Môn học trang bị cho học viên :

- ✓ những khái niệm cơ bản về phần mềm
- ✓ cách chế tạo phần mềm;
- ✓ các phương pháp, các thủ tục và công cụ phát triển phần mềm để x*ây dựng* phần mềm một cách hiệu quả.

I. Phần mềm- Software

1.1 Khái niệm phần mềm

Phần mềm là một sản phẩm có 3 thành phần chính :

- Các mã lệnh, khi được thực hiện trên máy tính thì thực thi các hoạt động và đưa ra các kết quả mong muốn
- Các cấu trúc dữ liệu hoặc cơ sở dữ liệu mà các mã lệnh sẽ thực thi trên chúng;
- Các tài liệu mô tả thao tác và cách dùng phần mềm.

1.2 Quá trình tiến hoá của phần mềm

1.Những năm đầu (1950-1960)

Trong những năm đầu của việc phát triển hệ thống máy tính, việc lập trình được coi là một "nghệ thuật" theo bản năng, chưa có phương pháp luận phát triển phần mềm.

Phần mềm được thiết kế theo đơn đặt hàng cho từng ứng dụng. Môi trường phần mềm có tính cá nhân, việc thiết kế là một tiến trình thường được thực hiện trong đầu người lập trình và thường là không có tài liệu.

2.**Giai đoạn thứ hai (1960 - giữa những năm 1970)**

Các hệ thống đa chương trình(multi-programming) và đa nhiệm (multi-tasking) đã đưa ra những khái niệm mới về tương tác người – máy(Interactive Man-Machine), mở ra một thế giới mới cho các ứng dụng và các mức độ mới về độ tinh vi cho cả phần cứng và phần mềm.

Các hệ thống thời gian thực có thể thu thập, phân tích và biến đổi dữ liệu từ nhiều nguồn khác nhau, do đó kiểm soát được các tiến trình và sản xuất ra "output" trong phần nghìn giây thay vì nhiều phút.

Những tiến bộ trong lưu trữ trực tuyến dẫn tới thế hệ đầu tiên của các hệ quản trị cơ sở dữ liệu.

Phần mềm đã được phát triển để phân phối theo quy mô rộng trong một thị trường nhiều bên tham dự.

Khi số lượng các hệ thống dựa trên máy tính tăng lên thì *thư viện phần mềm* cũng bắt đầu mở rộng, hàng chục ngàn câu lệnh gốc chương trình được bổ sung hàng ngày. Một cuộc khủng hoảng phần mềm đã bắt đầu "ló dạng ở chân trời": Tất cả những câu lệnh gốc này, những chương trình này đều phải sửa lại khi người ta phát hiện ra lỗi, hoặc phải được sửa lại khi yêu

cầu người dùng thay đổi, hoặc phải thích nghi với phần cứng vừa mua (gọi chung là *bảo trì phần mềm*); tuy nhiên, bản chất cá nhân của nhiều phần mềm làm cho chúng thực tế không thể bảo trì được.

3.Giai đoạn thứ ba (giữa những năm 1970 - 1990)

Xuất hiện hệ thống phân tán (là hệ thống nhiều máy tính, mỗi máy thực hiện một chức năng tương tranh và liên lạc với những máy khác) làm tăng dần độ phức tạp của hệ thống dựa trên máy tính.

Mạng toàn cục(WAN), mạng cục bộ(LAN), các liên lạc số giải thông cao, và nhu cầu thâm nhập dữ liệu "lập tức" đã đặt ra những yêu cầu rất lớn cho người lập trình.

Sự tiến bộ và sự phổ cập sử dụng các bộ vi xử lý, máy tính cá nhân và các máy trạm để bàn khá mạnh.

Phần cứng giá rẻ nhanh chóng trở thành hàng hoá tiêu dùng. Chi phí cho phần mếm có khuynh hướng tăng lên so với chi phí mua phần cứng.

4.Giai đoạn thứ tư (1990 đến nay)

Kỹ nghệ hướng đối tượng (Object oriented) đang nhanh chóng thay thế nhiều cách tiếp cận phát triển phần mềm truyền thống trong các lĩnh vực ứng dụng.

Xuất hiện các kỹ thuật mới làm thay đổi cách thức phát triển phần mềm, một trong các hướng đó là xây dựng phần mềm có khả năng tạo ra phần mềm.

Hệ chuyên gia và phần mềm trí tuệ nhân tạo cuối cùng đã đưa vào ứng dụng thực tế. Phần mềm mạng nơ ron nhân tạo đã mở ra những khả năng nhận dạng và thực hiện những khả năng xử lý thông tin kiểu con người.

1.3 Các đặc trưng của phần mềm

Phần mềm là sản phẩm của quá trình tư duy logic do đó nó có những đặc trưng khác biệt đáng kể so với phần cứng.

1. Phần mềm được phát triển hay được kỹ nghệ hoá, nó không được chế tạo theo nghĩa cổ điển.

Chi phí phần mềm tập trung vào kỹ nghệ, nghĩa là các dự án phần mềm dường như là các dự án chế tạo.

Vài thập kỷ qua, khái niệm "xưởng phần mềm" đã được đề cập nhiều, khái niệm này khuyến cáo về việc sử dụng các công cụ tự động hoá cho việc phát triển phần mềm.

2. Phần mềm không hỏng đi mà có thể bị lỗi thời

Tuy nhiên, rõ ràng rằng, phần mềm không mòn cũ đi nhưng nó lại bị lạc hậu.

Thực tế, phần mềm sẽ trải qua sự thay đổi và bảo trì. Khi thay đổi được thực hiện có thể phát sinh một số khiếm khuyết mới, có thể phần mềm bị thoái hoá do sự thay đổi.

Khi một yếu tố của phần cứng mòn đi sẽ có "vật tư thay thế". Mọi hỏng hóc trong phần mềm đều chỉ ra lỗi trong quá trình thiết kế. Do vậy, việc bảo trì phần mềm bao gồm thêm độ phức tạp phụ đáng kể so với phần cứng.

3. Phần lớn phần mềm đều được xây dựng theo đơn đặt hàng, chứ ít khi được lắp ráp từ những thành phần có sẵn

Đối với phần mềm, nhìn chung các danh mục các thành phần phần mềm là không có sẵn. Có thể đặt hàng một đơn vị phần mềm hoàn chỉnh, chứ không phải là những thành phần có thể lắp ráp thành một chương trình mới. (Tuy nhiên điều này đang thay đổi nhanh chóng)

1.4 Phân loại phần mềm

Nhóm 1: Các hệ điều hành

- Các driver (chương trình điều khiển)
- Các Monitor
- Các hệ quản lý tệp
- Các hệ thống quản lý thư viện chương trình và chương trình dịch
- Các hệ thống quản lý mạng máy tính

…

Nhóm 2: Các ngôn ngữ lập trình như PASCAL, C, C++, Visual Basic,FORTRAN, ADA v.v.

Nhóm 3: Các phần mềm hệ thống

- Các chương trình soạn thảo văn bản
- Các chương trình điều khiển các thiết bị ngoại vi
- Các chương trình mở rộng chức năng quản lý tệp: sắp xếp, sao chép, cập nhật . . .
- Các chương trình đồ hoạ
- Các giao diện thân thiện giữa người sử dụng và hệ điều hành

…

Nhóm 4: Các hệ quản trị cơ sở dữ liệu, quản lý tri thức và các chương trình mở rộng tương ứng.

Nhóm 5: Các phần mềm ứng dụng

- Các chương trình xử lý dữ liệu đa năng
- Các bộ chương trình phục vụ cho các yêu cầu tính toán cơ sở
- Các chương trình tối ưu hoá
- Các hệ chuyên gia và các hệ tương tự
- Các hệ mô phỏng
- Các lớp ngôn ngữ dữ liệu và tri thức phục vụ cho việc khai thác các cơ sở dữ liệu và tri thức.
- Các hệ tự động hoá quản lý chương trình
- Các hệ tự động hoá thiết kế
- Các hệ thống dạy học
- Các hệ thống tự học
- Các hệ thống tự động phát sinh chương trình kiểm thử và sửa chương trình
- Các hệ chương trình nhận dạng, phân tích và tổng hợp tiếng nói, hình ảnh, tín hiệu,…
- Các hệ chương trình điều khiển qui trình và các thiết bị công nghiệp

…

Nhóm 6: Các chương trình tiện ích –Utilities và trò chơi Games

- Các chương trình xử lý bảng tính điện tử
- Các chương trình chuyển đổi (tiền dịch) ngông ngữ, dịch chéo, khôi phục
- Các chương trình chống và diệt virus máy tính
- Các chương trình trò chơi giải trí

…

1.5 Các Các ngôn ngữ lập trình

Phần mềm máy tính là thông tin tồn tại dưới hai dạng cơ bản: các thành phần máy không thực hiện được và các thành phần máy thực hiện được. Ta xét thành phần phần mềm được xây dựng bằng cách dùng một ngôn ngữ nhân tạo với vốn từ vựng hạn chế, một văn phạm xác định rõ cùng các quy tắc chặt chẽ về cú pháp, ngữ nghĩa. Các thuộc tính này là ngôn ngữ cho việc dịch thành mã máy. Các dạng ngôn ngữ đã và đang dùng hiện nay là:

a) Ngôn ngữ máy :

Ngôn ngữ máy (còn được gọi **máy ngữ** hay **mã máy**; tiếng Anh là *machine language* hay *machine code*) là một loại ngôn ngữ lập trình trong đó, mọi chỉ thị đều được biểu diễn bằng các con số nhị phân 0 và 1. Đây là ngôn ngữ lập trình thế hệ đầu tiên. Tuy khó đọc và khó sử dụng, nhưng ngôn ngữ máy là ngôn ngữ duy nhất mà bộ vi xử lí có thể nhận biết và thực hiện một cách trực tiếp (tức không cần dịch sang bất kì ngôn ngữ nào khác).

Ngôn ngữ máy (mã máy) là ngôn ngữ nền tảng của bộ vi xử lý. Các chương trình được viết trong tất cả các loại ngôn ngữ khác cuối cùng đều được chuyển thành ngôn ngữ máy trước khi chương trình đó được thi hành. Lợi điểm của viết chương trình bằng ngôn ngữ máy là lập trình viên có thể điều khiển máy tính trực tiếp và đạt được chính xác điều mình muốn làm. Do đó, các chương trình ngôn ngữ máy được viết tốt là những chương trình rất hiệu quả (tốc độ thi hành nhanh, kích thước nhỏ). Bất lợi của chương trình ngôn ngữ máy là thông thường sẽ mất rất nhiều thời gian để viết, rất khó đọc, theo dõi để tìm lỗi. Thêm vào đó, bởi vì chương trình được viết bằng tập lệnh phụ thuộc vào từng bộ vi xử lý nên chương trình chỉ chạy được trên những máy tính có cùng bộ vi xử lý mà thôi. Ngôn ngữ máy cũng được gọi là ngôn ngữ cấp thấp *(low-level language)*

Nếu phần mềm được viết tốt, bảo trì được và có tư liệu tốt thì ngôn ngữ máy có thể làm cho việc sử dụng bộ nhớ và tối ưu tốc độ thực hiện chương trình rất hiệu quả.

Ví dụ tập lệnh của ngôn ngữ máy Minsk-32:

10 1000 1002

11 1000 1002

12 <nhân>

13 <chia>

b).Hợp ngữ Assembler

Hợp ngữ được phát triển nhằm giúp các lập trình viên dễ nhớ các chỉ thị của chương trình hơn. Hợp ngữ tương tự như ngôn ngữ máy nhưng lại sử dụng các ký hiệu gợi nhớ (mnemonics hay mã lệnh hình thức - symbolic operation code) để biểu diễn cho các mã lệnh của máy. Một đặc điểm khác nữa là hợp ngữ thông thường cho phép định địa chỉ hình thức (symbolic addressing), nghĩa là một vị trí bộ nhớ trong máy tính có thể được tham chiếu tới thông qua một cái tên hoặc ký hiệu, chẳng hạn như TOTAL thay vì phải sử dụng địa chỉ thực sự của nó (bằng con số nhị phân) trong ngôn ngữ máy. Các chương trình hợp ngữ còn bao gồm các chỉ thị vĩ mô *(macro instruction)* có thể tạo ra nhiều lệnh mã máy. Các chương trình hợp ngữ được chuyển sang mã máy thông qua một chương trình đặc biệt gọi là trình hợp dịch (assembler). Mặc dù hợp ngữ tương đối dễ dùng hơn mã máy nhưng hợp ngữ vẫn được xem là ngôn ngữ cấp thấp bởi vì nó vẫn còn rất gần với từng thiết kế của máy tính.

Ví dụ một đoạn lệnh assembler

```
WRITE_HEX  PROC NEAR
PUSH       CX
PUSH       DX
```

```
MOV         DH,DL
MOV         CX,04
SHR         DL,CL
CALL        WRITE_HEX_DIGIT
MOV         DL,DH
AND         DL,0Fh
CALL        WRITE_HEX_DIGIT
POP         DX
POP         CX
RET
WRITE_HEX       ENOP
PUBLIC      WRITE_HEX_DIGIT
```

c).Ngôn ngữ ký hiệu

Ngôn ngữ AvtoCode- tương tự như Assembler

d)Ngôn ngữ lập trình bậc cao: cho phép người lập trình viết chương trình theo ngôn ngữ gần giống với ngôn ngữ thông thường, không phụ thuộc vào từng máy tính cụ thể.

Ngôn ngữ cấp cao gần gũi hơn với ý niệm ngôn ngữ mà hầu hết mọi người đều biết, nó bao gồm các danh từ, động từ, ký hiệu toán học, liên hệ và các thao tác luận lý. Các yếu tố này có thể được phối hợp, liên kết với nhau tạo thành một hình thức của câu. Các "câu" này được gọi là các mệnh đề của chương trình (program statement). Chính vì những đặc điểm này, các lập trình viên dễ dàng đọc và dễ học ngôn ngữ cấp cao hơn so với ngôn ngữ máy hoặc hợp ngữ. Một lợi điểm quan trọng là ngôn ngữ cấp cao thông thường không phụ thuộc vào máy tính, nghĩa là các chương trình viết bằng ngôn ngữ cấp cao có thể chạy trên các loại máy tính khác nhau (sử dụng các bộ vi xử lý khác nhau).

cTuy đã có hàng trăm ngôn ngữ lập trình nhưng chỉ có một số trong số đó được dùng phổ biến, ví dụ: COBOL-kinh tế, FORTRAN-kỹ thuật, PASCAL,ALGOL(Algorithm Language), C, ADA, ngôn ngữ hướng đối tượng: C++, OBJECT PASCAL, EIFFEL, ngôn ngữ đặc thù: APL, LISP, OPS5, PROLOG, và các ngôn ngữ mô tả trong mạng nơ ron nhân tạo

Mã máy, hợp ngữ, các ngôn ngữ lập trình cấp cao thường được coi là "3 thế hệ đầu" của ngôn ngữ máy tính. Với những ngôn ngữ này bản thân người lập trình phải quan tâm đến cả việc đặc tả cấu trúc thông tin lẫn điều khiển chương trình. Do vậy, các ngôn ngữ trong ba thế hệ đầu này còn được gọi là ***ngôn ngữ thủ tục***.

Với ***ngôn ngữ phi thủ tục,*** thay vì phải yêu cầu người lập trình xác định chi tiết thủ tục thì các ngôn ngữ phi thủ tục đưa đến một chương trình bằng cách xác định kết quả mong muốn, thay vì xác định hành động cần để đạt được kết quả đó". Phần mềm hỗ trợ sẽ dịch đặc tả thành chương trình máy thực hiện được. Ngày nay các ngôn ngữ thế hệ thứ tư này đang được dùng trong các ứng dụng CSDL và các lĩnh vực xử lý dữ liệu nghiệp vụ khác.

e)Ngôn ngữ trực quan- Visual

Các ngôn ngữ lập trình thông dụng

Mặc dù đã có hàng trăm ngôn ngữ lập trình được sinh ra, chỉ có một số ít là được sử dụng rộng rãi và được xem là một chuẩn công nghiệp. Các ngôn ngữ này đều có thể được sử dụng trên nhiều loại máy tính khác nhau.

BASIC, viết tắt của cụm từ **B**eginner's **A**ll-Purpose **S**ymbolic **I**nstruction **C**ode, được phát triển bởi John Kermeny và Thomas Kurtz vào năm 1964 tại trường đại học Dartmouth. Ban đầu, họ thiết kế BASIC là một ngôn ngữ lập trình đơn giản, có tính tương tác để các sinh viên học tập và sử dụng. BASIC đã trở thành một trong những ngôn ngữ lập trình thông dụng nhất được sử dụng trên các máy vi tính và máy tính mini ngày nay.

ví dụ một đoạn mã BASIC :

```
10 INPUT       “DV A,B,C :”;A,B,C
20 D=B^2 – 4*A*C
30 IF D<0 THEN 80
40 X1=(-B+SQR(D))/(2*A)
50 X2=(-B-SQR(D))/(2*A)
60 PRINT “X1=”;X1;”  X2=”;X2
70 END
80 ?”KHONG CO NGHIEM THUC”
90 END
```

COBOL, viết tắt của **CO**mmon **B**usiness **O**riented **L**anguage, được giới thiệu vào năm 1960. Được hỗ trợ bởi bộ quốc phòng Hoa Kỳ, COBOL được phát triển bởi một hội đồng bao gồm các đại diện từ phía chính phủ và công nghiệp. Grace M.Hopper là người chính yếu trong hội đồng và được xem là nhà phát triển chính của ngôn ngữ COBOL. COBOL đã từng là một trong những ngôn ngữ được dùng rộng rãi nhất cho các ứng dụng thương mại. Bằng cách dùng một hình thức tựa tiếng Anh, các câu lệnh của COBOL được sắp xếp vào trong các câu và nhóm lại thành từng đoạn (paragraph). Hình thức tiếng Anh giúp COBOL dễ viết và đọc nhưng cũng làm cho chương trình nguồn dài hơn. COBOL rất tốt trong việc xử lý các tập tin lớn và thực hiện nhưng phép tính thương mại tương đối đơn giản.

C, được phát triển bởi tác giả Dennis Ritchie tại phòng thí nghiệm Bell vào năm 1972. Ban đầu, C được thiết kế như là một ngôn ngữ để viết các phần mềm hệ thống, nhưng ngày nay, nó được xem là một ngôn ngữ công dụng chung. C là một ngôn ngữ lập trình mạnh mẽ đòi hỏi kỹ năng lập trình chuyên nghiệp mới có thể sử dụng hiệu quả được. Nhu cầu dùng C để phát triển nhiều loại phần mềm kể cả các ứng thương mại đang gia tăng. Các chương trình C thường được dùng với hệ điều hành Unix (phần lớn hệ điều hành Unix được viết bằng C).

FORTRAN, viết tắt của **FOR**mula **TRAN**slator được phát triển bởi một nhóm lập trình viên của công ty IBM dưới sự lãnh đạo của John Backus. Công bố vào năm 1957, FORTRAN được thiết kế như là một ngôn ngữ lập trình dành cho các nhà khoa học, kỹ sư và toán học. FORTRAN được xem như là ngôn ngữ lập trình cấp cao đầu tiên và được chú ý bởi khả năng của nó cho phép dễ dàng diễn đạt và tính toán các phương trình toán học.

```
INTEGER A(10)
```

```
DATA  IMP/4/’thiết bị đưa ra
DO 20 I=1,10
DO 10 J=1,10
10 A(I,J)=I*J
20 WRITE(IMP,30)A   ‘đưa ra mảng
30 FORMAT1015/)  ‘ định dạng cách dòng
STOP
END
```

PASCAL, ngôn ngữ sẽ được sử dụng để giảng dạy trong giáo trình này, được phát triển vào năm 1968 bởi Niklaus Wirth, một nhà khoa học máy tính tại Zurich, Thụy Sĩ. Pascal được phát triển để giảng dạy lập trình. Tên Pascal không phải là từ viết tắt, đó là tên của một nhà toán học, Blaise Pascal (1623 - 1662) người đầu tiên tạo ra máy tính. Pascal, dùng trong cả máy tính cá nhân và máy tính lớn là một trong những ngôn ngữ lập trình đầu tiên được phát triển trong đó khuyến khích phương pháp lập trình cấu trúc.

Các loại ngôn ngữ lập trình khác

ALGOL (**ALGO**rithmetic **L**anguage). Ngôn ngữ lập trình cấu trúc dùng cho các ứng dụng khoa học và toán học.

APL(**A P**rogramming **L**anguage). Một ngôn ngữ mạnh mẽ, dễ dùng, rất tốt trong việc xử lý dữ liệu được lưu dưới dạng bảng (ma trận)

FORTH, tương tự như C, tạo ra các mã chương trình nhanh và hiệu quả. Ban đầu được phát triển để điều khiển kính viễn vọng không gian.

LISP, LISt **P**rocessing, ngôn ngữ trí tuệ nhân tạo thông dụng.

AutoLISP – dùng trong tự động hóa thiết kế

LOGO, chủ yếu được biết đến như là một công cụ để dạy khả năng giải quyết vấn đề.

MODULA-3, tương tự như PASCAL. Dùng chủ yếu để phát triển các phần mềm hệ thống.

PILOT, **P**rogrammed **I**nquiry **L**earning **O**r **T**eaching, dùng bởi các nhà giáo dục để viết các chương trình hướng dẫn CAD.

PL/I, **P**rogramming **L**anguage/ One. Ngôn ngữ thương mại và khoa học phối hợp nhiều chức năng của FORTRAN và COBOL.

PROLOG, **PRO**gramming **LO**gic. Dùng trong trí tuệ nhân tạo.

RPG, **R**eport **P**rogram **G**enerator. Dùng các mẫu đặc biệt để giúp người dùng xác định dữ liệu vào, dữ liệu ra và các yêu cầu tính toán của một chương trình.

ADA, lấy tên của Augusta Ada Bryon, người được xem là đã viết chương trình đầu tiên, được thiết kế để phục vụ cho việc viết, bảo trì các chương trình lớn trong một khoảng thời gian dài.

VISUAL BASIC

Ví dụ :

```
        For i = 1 To 15
            iArray(i) = i ^ 2
        Next

       Dim FullMsg As String = ""
        For i = 1 To 15
            FullMsg = FullMsg & "Phần tử thứ (" & CStr(i) & ")=" &
CStr(iArray(i)) & vbCrLf
        Next
        txtArray.text = FullMsg
```

Khái niệm về Trình thông dịch-Interpreter và biên dịch-compiler

Mọi chương trình được viết bằng các ngôn ngữ không phải là ngôn ngữ máy cuối cùng đều phải được chuyển đổi sang ngôn ngữ máy trước khi được thi hành. Chương trình ngôn ngữ cấp

cao được dịch sang ngôn ngữ máy bằng một trong hai cách : bằng trình biên dịch (compiler) hoặc trình thông dịch (interpreter).

***Trình biên dịch (ví dụ ngôn ngữ VISUAL BASIC)*:**

Sẽ chuyển đổi toàn bộ chương trình sang mã máy, rồi chứa kết quả vào dĩa để có thể thi hành về sau. Chương trình ngôn ngữ cấp cao được chuyển đổi được gọi là chương trình nguồn (source program) và chương trình ngôn ngữ máy được tạo ra được gọi là chương trình đối tượng (object program) hoặc mã đối tượng (object code). Khi người dùng muốn chạy chương trình, chương trình đối tượng sẽ được nạp lên bộ nhớ chính của CPU và các chỉ thị của chương trình sẽ được thi hành. Khi được hướng dẫn bởi các chỉ thị của chương trình, CPU sẽ truy xuất dữ liệu và tạo ra các kết quả. Trình biên dịch sẽ kiểm tra cú pháp chương trình, thực hiện các phép kiểm tra logic và đảm bảo các dữ liệu sắp được sử dụng trong các phép so sánh, tính toán đã được định nghĩa một cách hợp lý ở một nơi nào đó trong chương trình. Một chức năng quan trọng của trình biên dịch là nó sẽ tạo ra một danh sách lỗi của tất cả mệnh đề trong chương trình vi phạm cú pháp của ngôn ngữ. Danh sách này giúp lập trình viên dễ dàng sửa đổi chương trình.

Do ngôn ngữ máy phụ thuộc vào bộ vi xử lý nên các máy tính khác nhau sẽ cần có các trình biên dịch khác nhau đối với cùng một ngôn ngữ cấp cao. Ví dụ, một máy mainframe, máy mini và máy tính cá nhân cần có các trình biên dịch khác nhau để biên dịch cùng một chương trình nguồn sang mã máy của từng loại máy này.

***Trình thông dịch(ví dụ ngôn ngữ BASIC)* :**

Thay vì chuyển đổi toàn bộ chương trình nguồn như trình biên dịch, trình thông dịch chỉ chuyển đổi một mệnh đề của chương trình và thực hiện đoạn mã kết quả ngay, sau đó nó tiếp tục chuyển đổi mệnh đề thứ 2 rồi thi hành đoạn mã kết quả thứ 2 và cứ thế. Khi sử dụng trình thông dịch, mỗi lần chạy chương trình là mỗi lần chương trình nguồn được thông dịch sang ngôn ngữ máy. Không có chương trình đối tượng nào được tạo ra.

Các trình thông dịch thường được dùng trên các máy tính cá nhân không có đủ bộ nhớ hoặc sức mạnh tính toán cần thiết để dùng trình biên dịch. Lợi điểm của trình thông dịch là lập trình viên vẫn có thể chạy một chương trình vẫn còn lỗi cú pháp. Chỉ đến lúc thông dịch đến câu lệnh có lỗi cú pháp, quá trình thi hành chương trình mới bị ngừng lại và trình thông dịch sẽ thông báo lỗi. Điểm bất lợi là các chương trình thông dịch chạy không nhanh bằng các chương trình được biên dịch vì quá trình chuyển đổi sang ngôn ngữ máy được thực hiện cùng với quá trình thi hành chương trình. Vì lý do này, ngày nay, đa số các ngôn ngữ cấp cao đều dùng trình biên dịch.

1.6 Việc ứng dụng phần mềm

Phần mềm có thể được áp dụng khi đã có một tập các bước thủ tục (như một thuật toán) đã được xác định trước (trừ phần mềm chuyên gia và phần mềm mạng nơ ron). Nội dung thông tin và tính tất định là các nhân tố quan trọng trong việc xác định bản chất của ứng dụng phần mềm:

- nội dung thông tin nói tới ý nghĩa và hình dạng của thông tin vào ra.
- tính tất định thông tin nói tới việc tiên đoán trước trật tự và thời gian của thông tin.

Có 7 loại phần mềm ứng dụng:

1. *Phần mềm hệ thống:* là một tập hợp các chương trình được viết để phục vụ các chương trình khác . Ví dụ như: trình biên dịch, trình soạn thảo, tiện ích quản lý tệp... Phần mềm hệ thống xử lý cấu trúc thông tin phức tạp nhưng xác định. Nó được đặc trưng bởi tương tác chủ yếu tới phần cứng của máy tính, phục vụ nhiều người dùng, thao tác tương tranh, dùng chung tài nguyên, các quản lý tiến trình phức tạp, cấu trúc dữ liệu phức tạp và nhiều giao diện ngoài.
2. *Phần mềm thời gian thực- Real Time:* là phần mềm điều phối, phân tích, kiểm soát các sự kiện thế giới thực ngay khi chúng xuất hiện. Phần mềm thời gian thực bao gồm các yếu tố: một thành phần thu thập dữ liệu để thu và định dạng thông tin từ ngoài, một thành phần phân tích để biến đổi thông tin theo yêu cầu của ứng dụng, một thành phần kiểm soát hoặc đưa ra đáp ứng môi trường ngoài, một thành phần điều phối để điều hoà các thành phần khác sao cho có thể duy trì việc đáp ứng thời gian thực. Hệ thống thời gian thực phải đáp ứng trong những ràng buộc thời gian chặt chẽ.
3. *Phần mềm nghiệp vụ:* Xử lý thông tin nghiệp vụ là lĩnh vực ứng dụng phần mềm lớn nhất. Các hệ thống rời rạc như tính lương, kế toán, quản lý...đã tiến hoá thành các hệ phần mềm quản lý thông tin (*Management Information System*). Những ứng dụng trong lĩnh vực này đã cấu trúc lại dữ liệu theo cách thuận tiện cho các thao tác nghiệp vụ. Ngoài ra phần mềm này còn bao gồm cả tính toán tương tác như xử lý giao tác cho các điểm bán hàng.
4. *Phần mềm khoa học và công nghệ:* Phần mềm này được đặc trưng bởi các thuật toán "máy nghiền số". Có các ứng dụng phức tạp: thiên văn, núi lửa, biến động quỹ đạo tàu con thoi, sinh học phân tử... Thiết kế có máy tính trợ giúp (CAD – *Computer Aided Design*), mô phỏng hệ thống và những ứng dụng tương tác khác đã bắt đầu kế tục các đặc trưng thời gian thực và thậm chí cả phần mềm hệ thống.
5. *Phần mềm nhúng:* Nằm sâu trong các bộ nhớ chỉ đọc và được dùng để điều khiển các sản phẩm và hệ thống người tiêu dùng và thị trường công nghiệp. Phần mềm nhúng thực hiện các chức năng giới hạn và huyền bí như điều khiển bàn phím cho lò vi sóng, hoặc đưa ra các khả năng điều khiển vận hành có ý nghĩa như chức năng số hoá trong ô tô, kiểm soát xăng...
6. *Phần mềm máy tính cá nhân:* Thị trường này đã bùng nổ trong suốt một thập kỷ qua: xử lý văn bản, trang tính, đồ hoạ, quản trị cơ sở dữ liệu (CSDL), v.v. Phần mềm máy tính cá nhân biểu thị cho một số thiết kế giao diện người - máy được cải tiến nhiều nhất.
7. *Phần mềm trí tuệ nhân tạo (*AI - *Artificial intelligence):* Phần mềm AI dùng các thuật toán phi số để giải quyết các vấn đề phức tạp mà tính toán trực tiếp không thể giải quyết nổi. Hiện nay, mạnh nhất là các *hệ chuyên gia* (hay còn gọi là *hệ cơ sở tri thức*). Các ứng dụng khác như: nhận dạng, chứng minh định lý, trò chơi...; Hiện nay con có *mạng nơ*

ron nhân tạo đã phát triển, nó mô phỏng cấu trúc việc xử lý trong bộ óc để tạo ra một lớp phần mềm có thể nhận dạng các mẫu phức tạp.

1.7 Các thách thức đối với phần mềm máy tính

- Sự tinh vi của phần cứng luôn đi trước khả năng xây dựng phần mềm đạt tới tiềm năng của phần cứng.
- Khả năng xây dựng các chương trình mới không thể giữ cùng nhịp với các nhu cầu có chương trình mới.
- Khả năng bảo trì cho các chương trình bị đe doạ bởi những bản thiết kế nghèo nàn và tài nguyên không thích hợp

Để đáp ứng lại các vấn đề trên, việc thực hành *kỹ nghệ phần mềm - chủ đề mà giáo trình này tập trung vào* - đang được chấp nhận trong ngành công nghiệp phần mềm.

II. Kỹ nghệ phần mềm

2.0.Đối tượng môn học

Kỹ nghệ phần mềm là việc thiết lập và sử dụng các nguyên lý công nghệ đúng đắn để thu được phần mềm một cách kinh tế vừa tin cậy, vừa làm việc hiệu quả trên các máy thực.

2.1 Định nghĩa

Kỹ nghệ phần mềm gồm một tập ba yếu tố chủ chốt:

- Phương pháp - *Methods*
- Công cụ - *Tools*
- Thủ tục – *Procedures*

Các yếu tố này giúp người quản lý kiểm soát được tiến trình phát triển phần mềm, đồng thời cung cấp cho người hành nghề một nền tảng để xây dựng được một phần mềm chất lượng cao theo một cách thức hiệu quả. Chúng ta cùng xem xét tóm tắt từng yếu tố đó.

1. ***Các phương pháp*** kỹ nghệ phần mềm đưa ra các "cách làm" về mặt kỹ thuật để xây dựng phần mềm.

   Các phương pháp bao gồm các nhiệm vụ: lập kế hoạch và ước lượng dự án, phân tích yêu cầu hệ thống và phần mềm, thiết kế cấu trúc dữ liệu, kiến trúc chương trình và thủ tục thuật toán, mã hóa, kiểm thử và bảo trì;

   Các phương pháp kỹ nghệ phần mềm thường đưa ra các ký pháp đồ họa hay hướng ngôn ngữ đặc biệt và đưa ra một tập các tiêu chuẩn về chất lượng phần mềm.

2. ***Các công cụ*** kỹ nghệ phần mềm cung cấp sự hỗ trợ tự động hay bán tự động cho từng phương pháp nêu trên.

   Khi các công cụ được tích hợp đến mức có thể được dùng cho các công cụ khác de phát triển phần mềm - được gọi là **Kỹ nghệ phần mềm có máy tính hỗ trợ** (CASE - *Computer Aided Software Engineering).*

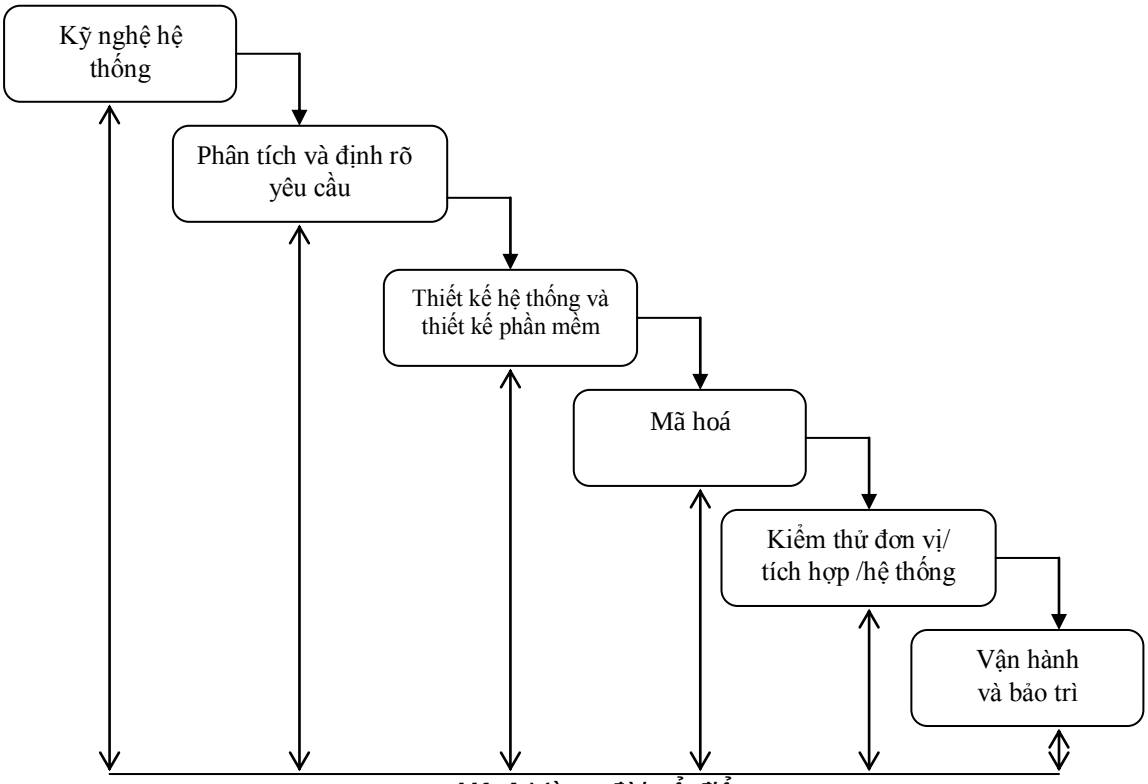
3. Các thủ tục kỹ nghệ phần mềm xác định:

- trình tự các phương pháp sẽ được sử dụng
- sản phẩm cần bàn giao (tài liệu, báo cáo, bản mẫu...)
- các mốc thời gian để người làm phần mềm nắm được tiến độ

2.2. bốn cách tiếp cận cơ bản trong tiến trình phát triển phần mềm.

Cách tiếp cận 1: Mô hình vòng đời cổ điển

Hình 1.4 minh họa cho *Mô hìnhvòng đời cổ điển (hay Khuôn cảnh vòng đời cổ điển)* đối với kỹ nghệ phần mềm. Có tên gọi là "mô hình thác nước", khuôn cảnh vòng đời yêu cầu một cách tiếp cận *tuần tự* đối với việc phát triển phần mềm. Nó bắt đầu ở mức hệ thống và tiến dần xuống phân tích, thiết kế, mã hóa, kiểm thử và bảo trì. Như vậy khuôn cảnh vòng đời bao gồm các hoạt động trong mô hình thác nước sau:



H1.4 *Vòng đời cổ điển*

1. *Kỹ nghệ và phân tích hệ thống: P*hần mềm là một phần của hệ thống ứng dụng nên công việc phải bắt đầu từ việc thiết lập yêu cầu cho tất cả các phần tử của hệ thống. Kỹ nghệ và phân tích hệ thống bao gồm việc thu thập yêu cầu ở mức hệ thống với một thiết kế sơ bộ và phân tích mức đỉnh.
2. *Phân tích yêu cầu phần mềm:* Tiến trình thu thập yêu cầu được tập trung và làm mạnh đặc biệt vào phần mềm. Các kỹ sư phần mềm cần phải diễn đạt ra được các yêu cầu đối với phần mềm dưới dạng các chức năng cần có, hiệu năng và giao diện. Cần lập tư liệu về các yêu cầu cho cả hệ thống và phần mềm, và được khách hàng duyệt lại.
3. *Thiết kế:* Thiết kế phần mềm là một tiến trình nhiều bước tập trung vào bốn thuộc tính phân biệt của chương trình:

- thiết kế cấu trúc dữ liệu
- kiến trúc phần mềm (các chức năng của chương trình)
- chi tiết các thủ tục, thuật toán
- thiết kế giao diện

Tiến trình thiết kế chuyển hóa các yêu cầu thành một biểu diễn của phần mềm có thể khẳng định về chất lượng trước khi giai đoạn mã hóa bắt đầu. Việc thiết kế phải được lập tư liệu và trở thành một phần của cấu hình phần mềm.

4. *Mã hóa:* dùng một ngôn ngữ lập trình cụ thể viết chương trình.Đây là khâu quan trọng và tốn kém nhất về thời gian và chất xám của cán bộ lập trình – chuyên gia phần mềm.
5. *Kiểm thử:* Việc kiểm thử bắt đầu sau khi đã sinh ra mã, tập trung vào phần logic bên trong chương trình, đảm bảo rằng tất cả các câu lệnh đều được kiểm thử. Về phần chức năng bên ngoài cần đảm bảo việc tiến hành kiểm thử phát hiện ra các lỗi và đảm bảo những cái vào xác định sẽ tạo ra kết quả thực tế thống nhất với kết quả muốn có.

Việc kiểm thử luôn đi liền với việc viết mã, xong một thủ tục phải thử ngay để sửa lỗi.

6. *Bảo trì:* Phần mềm chắc chắn sẽ có những thay đổi sau khi được bàn giao cho khách hàng (ngoại lệ là những phần mềm nhúng). Nguyên nhân có thể là lỗi do phần mềm, phải thích ứng với môi trường bên ngoài (hệ điều hành mới, thiết bị ngoại vi mới...), hoặc do khách hàng yêu cầu nâng cao chức năng hay hiệu năng… Việc bảo trì phần mềm phải áp dụng lại các bước vòng đời nói trên.

Vòng đời cổ điển ra đời sớm nhất và được sử dụng rộng rãi nhất cho kỹ nghệ phần mềm. Tuy nhiên có một số vấn đề hay gặp phải :

- Các dự án hiếm khi tuân theo dòng chảy *tuần tự* mà mô hình đề ra. Việc lặp lại bao giờ cũng xuất hiện và gây ra vấn đề.
- Khách hàng khó phát biểu hết mọi yêu cầu một cách tường minh khi mới triển khai dự án và thường xẩy ra mâu thuẫn giữa thực tế với khả năng đáp ứng của phần mềm.

Cách tiếp cận 2: Mô hình làm bản mẫu

Thông thường khách hàng đã xác định được mục tiêu tổng quát của phần mềm, nhưng chưa xác định được dữ liệu nào cần nhập vào, xử lý ra sao hay dữ liệu xuất ra như thế nào; người phân tích chưa hiểu rõ nhu cầu của khách hàng. Ngoài ra, người phát triển có thể không chắc về tính hiệu quả của một thuật toán hay giải pháp, việc thích nghi hệ điều hành hay dạng giao diện người máy (HCI – *Human Computer Interface*) cần có. Trong những trường hợp này và nhiều trường hợp khác cách tiếp cận làm bản mẫu cho kỹ nghệ phần mềm là tốt nhất.

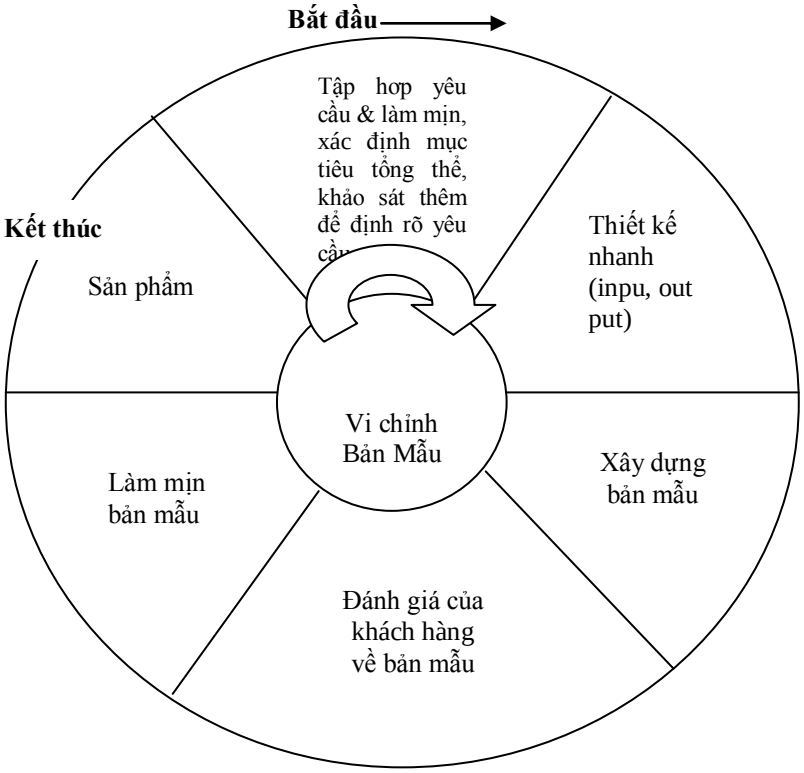
Làm bản mẫu là một tiến trình có khả năng tạo ra một mô hình cho phần mềm cần phải xây dựng. Mô hình có thể ở trong 3 dạng:

(1) bản mẫu trên giấy hay mô hình dựa trên máy PC mô tả giao diện người máy dưới dạng làm cho người dùng hiểu được cách các tương tác xuất hiện

(2) bản mẫu làm việc cài đặt một tập con các chức năng của phần mềm mong muốn

(3) một chương trình đã có thực hiện một phần hay tất cả các chức năng mong muốn nhưng cần phải cải tiến thêm các tính năng khác tùy theo khả năng phát triển.

Dãy các sự kiện của khuôn cảnh làm bản mẫu được minh họa `trong hình sau:



Giống như các mọi cách tiếp cận cho việc phát triển phần mềm, việc làm bản mẫu bắt đầu với việc thu thập yêu cầu. Người phát triển và khách hàng gặp nhau, xác định các mục tiêu tổng thể cho phần mềm, xác định các yêu cầu nào đã biết, miền nào bắt buộc phải xác định thêm. Rồi đến việc "thiết kế nhanh" để xây dựng một bản mẫu. Bản mẫu được người dùng đánh giá và được dùng để làm mịn các yêu cầu đối với phần mềm cần phát triển. Tiến trình lặp đi lặp lại xảy ra để cho bản mẫu được "vi chỉnh" thỏa mãn nhu cầu của khách hàng, đồng thời giúp người phát triển hiểu kỹ hơn cần phải thực hiện nhu cầu nào. Giống như vòng đời cổ điển, việc làm bản mẫu tựa như một khuôn cảnh cho kỹ nghệ phần mềm có thể trở thành có vấn đề.

Cách tiếp cận 3: Mô hình xoắn ốc

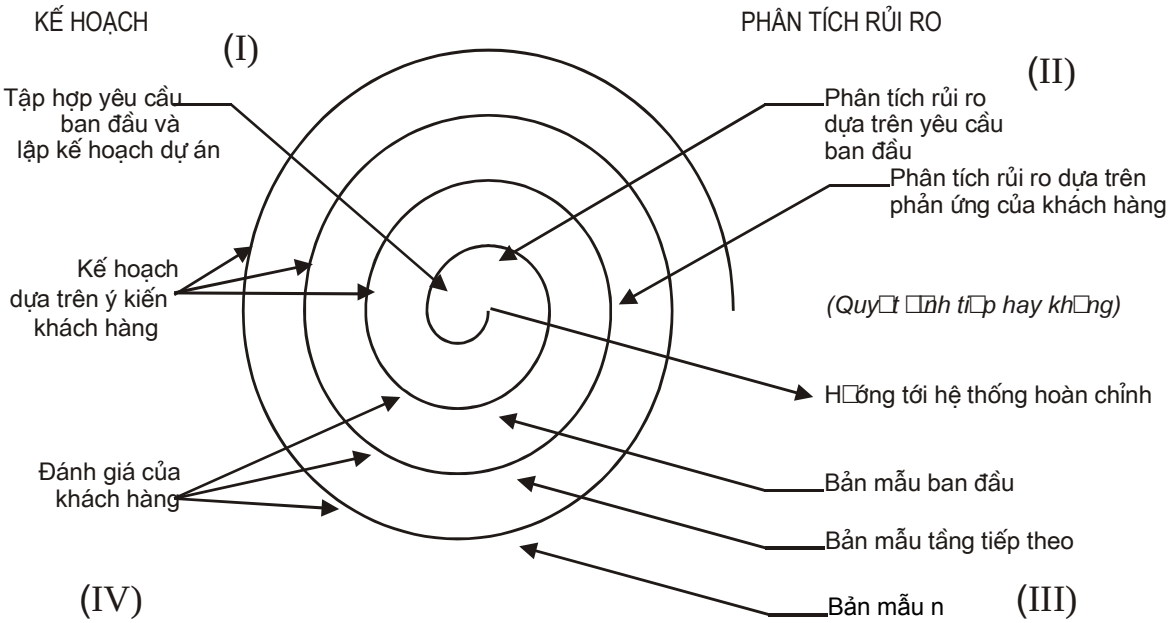
Mô hình xoắn ốc bao gồm các tính năng tốt nhất của cả vòng đời cổ điển lẫn làm bản mẫu, trong khi còn bổ xung yếu tố mới là phân tích rủi ro – cái còn thiếu trong các mô hình này. Mô hình này được biểu diễn theo đường xoắn ốc, xác định ra bốn hoạt động chính:

(1) Lập kế hoạch: xác định mục tiêu, giải pháp và ràng buộc

(2) *Phân tích rủi ro*: phân tích các phương án và xác định, giải quyết rủi ro.

(3) Kỹ nghệ: phát triển sản phẩm "mức tiếp"

(4) Đánh giá của khách hàng: khẳng định kết quả của kỹ nghệ



H1.6 *Mô hình xoắn ốc*

Cách tiếp cận thực tế nhất cho việc phát triển phần mềm của các hệ thống có quy mô lớn

Mỗi lần lặp xung quanh xoắn ốc (từ tâm đi ra ngoài) người ta lại xây dựng thêm các phiên bản được hoàn thiện dần của phần mềm. Trong mạch xoắn thứ nhất, các mục tiêu, phương pháp, ràng buộc, các rủi ro được định rõ và phân tích. Nếu phân tích rủi ro chỉ ra rằng, không chắc chắn trong các yêu cầu thì việc làm bản mẫu có thể được sử dụng trong góc phần tư kỹ nghệ. Các mô phỏng và các mô hình khác cũng có thể được dùng để xác định rõ thêm vấn để và làm mịn yêu cầu.

Khách hàng đánh giá công việc kỹ nghệ và đưa ra gợi ý về những thay đổi (góc phần tư đánh giá của khách hàng), giai đoạn tiếp của việc lập kế hoạch và phân tích rủi ro sẽ được tiến hành. Tại mỗi vòng xung quanh xoắn ốc, cao điểm của việc phân tích rủi ro là "tiến hành hay không tiến hành", nếu rủi ro quá lớn có thể đình chỉ dự án.

Mọi mạch đi xung quanh xoắn ốc đều đòi hỏi **kỹ nghệ** (góc phần tư phía dưới bên phải) có thể được thực hiện bằng cách tiếp cận vòng đời và làm bản mẫu. Tất nhiên số các hoạt động phát triển xuất hiện trong góc phần tư phía dưới bên phải tăng lên khi các hoạt động chuyển xa hơn ra khỏi trung tâm xoắn ốc.

Khuôn cảnh mô hình xoắn ốc đối với kỹ nghệ phần mềm hiện tại là cách tiếp cận thực tế nhất đến việc phát triển cho các hệ thống và phần mềm quy mô lớn. Trong đó *người ta dùng cách làm bản mẫu như một cơ chế làm giảm bới rủi ro.*

Mô hình này tương đối mới và còn chưa được sử dụng rộng rãi như vòng đời, làm bản mẫu.

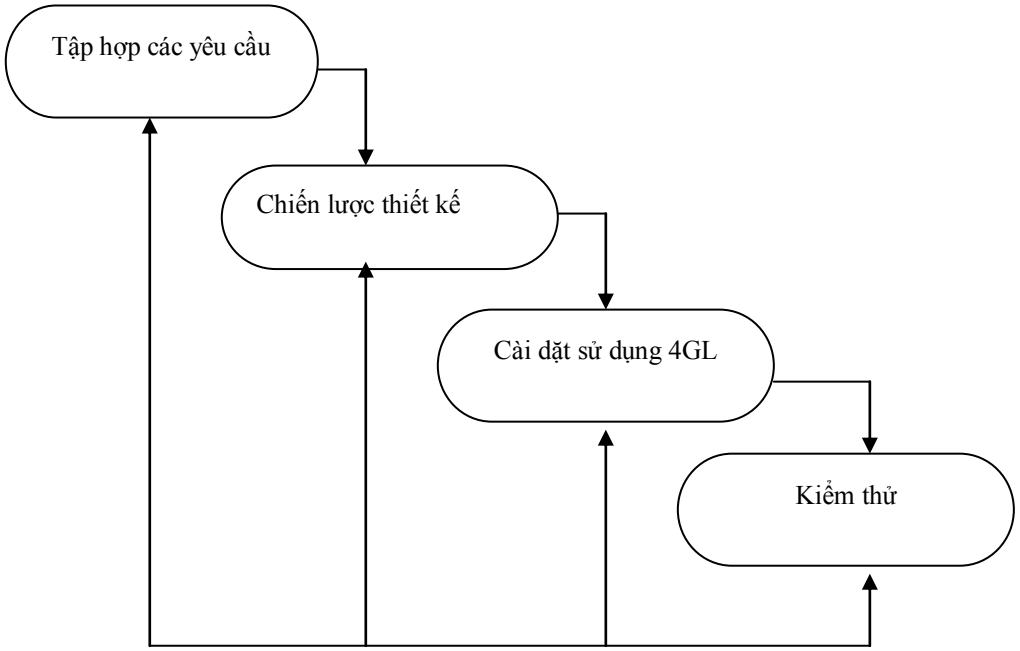
Cách tiếp cận 4: Kỹ thuật thế hệ thứ tư

Thuật ngữ "kỹ thuật thế hệ thứ tư" (4GT - *The Fourth Generation Technology*) bao gồm một phạm vi rộng các công cụ phần mềm có một điểm chung: Mỗi công cụ đều cho phép người lập trình xác định một số đặc trưng của phần mềm ở mức cao. *Công cụ đó tự động sinh ra mã chương trình gốc theo nhu cầu của người phát triển.*

Khuôn cảnh 4GT tập trung vào khả năng xác định phần mềm đối với một máy ở mức độ gần với ngôn ngữ tự nhiên hay dùng một ký pháp đem lại chức năng có ý nghĩa.

Một môi trường phát triển phần mềm hỗ trợ cho khuôn cảnh 4GT bao gồm một số hoặc tất cả các công cụ sau: ngôn ngữ phi thủ tục để truy vấn CSDL, bộ sinh báo cáo, bộ thao tác dữ liệu, bộ tương tác và xác định màn hình, bộ sinh chương trình, khả năng đồ họa mức cao, khả năng làm trang tính.

Khuôn cảnh 4GT cho kỹ nghệ phần mềm được thể hiện trên sơ đồ:



H1.7 *Các kỹ thuật thế hệ thứ tư*

Việc dùng khuôn cảnh 4GT còn có nhiều tranh cãi:

- Người ủng hộ cho rằng 4GT làm giảm đáng kể thời gian phát triển phần mềm, tăng hiệu suất của người lập trình.
- Người phản đối cho rằng các công cụ 4GT hiện tại không phải tất cả đều dễ dùng hơn các ngôn ngữ lập trình, các chương trình gốc do các công cụ này tạo ra là “không hiệu quả”, và việc bảo trì các hệ thống phần mềm lớn được phát triển bằng cách dùng 4 GT sẽ sinh ra nhiều vấn đề mới.

Ta tóm tắt trạng thái hiện tại của cách tiếp cận 4 GT như sau:

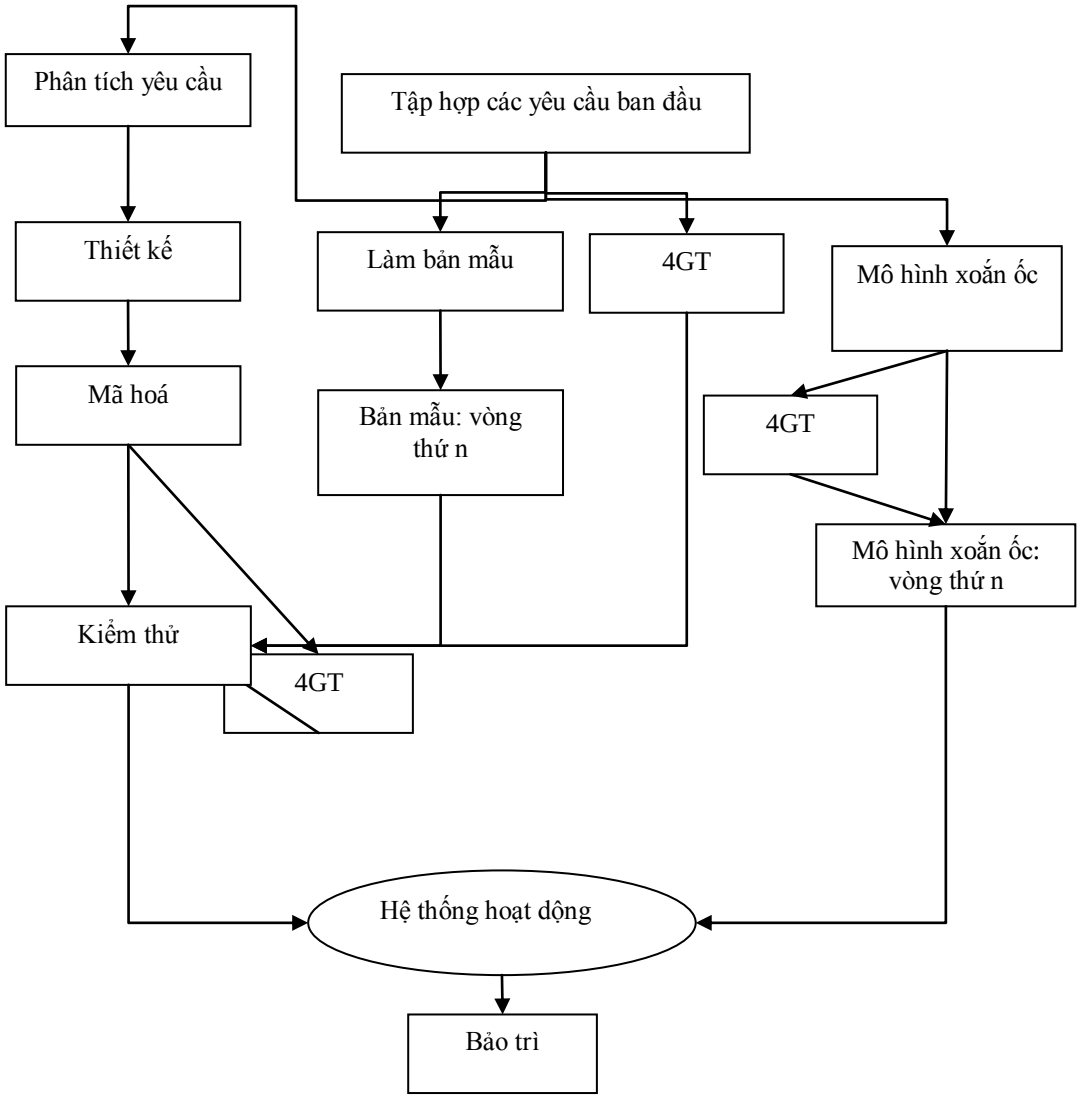
- Đối với CSDL lớn, 4 GT chỉ mới giới hạn vào các ứng dụng hệ thông tin nghiệp vụ, đặc biệt, việc phân tích thông tin và làm báo cáo (là nhân tố chủ chốt cho các CSDL lớn)
- Đối với các ứng dụng vừa và nhỏ, thời gian thu thập dữ liệu sơ bộ cần để tạo phần mềm được giảm đáng kể. Khối lượng thiết kế cho các ứng dụng nhỏ cũng được rút bớt.
- Để phát triển phần mềm lớn, đòi hỏi tập trung nhiều vào phân tích, thiết kế và kiểm thử để đạt tới việc tiết kiệm thời gian là chủ yếu.

Tóm lại, các kỹ thuật thế hệ thứ tư đã trở thành một phần quan trọng của việc phát triển phần mềm trong lĩnh vực áp dụng công nghệ thông tin. Hình dưới đây minh họa nhu cầu phần mềm sẽ tiếp tục leo thang, nhưng phần mềm được áp dụng các phương pháp và khuôn cảnh quy ước thì sẽ đóng góp ngày càng ít, kỹ thuật thế hệ thứ tư sẽ lấp lỗ hổng này.

2.6 Cách tiếp cận 5: Tổ hợp các khuôn cảnh

Các khuôn cảnh đề cập ở trên được mô tả như các cách tiếp cận khác nhau đối với kỹ nghệ phần mềm chứ không phải là cách tiếp cận bổ xung cho nhau, tuy nhiên trong nhiều trường hợp có thể và cũng nên tổ hợp các khuôn cảnh để đạt sức mạnh của từng khuôn cảnh cho một dự án riêng lẻ.

Hình 1.9 minh họa cách tổ hợp các khuôn cảnh kỹ nghệ phần mềm, trong mọi trường hợp, công việc bắt đầu với mọi khuôn cảnh là việc xác định mục tiêu, phương án và các ràng buộc (hay thu thập yêu cầu sơ bộ). Từ điểm này, bất kỳ một con đường nào trên hình 1.9 đều có thể được chọn.



H1.9 *Tổ hợp các khuôn cảnh*

Chẳng hạn, nếu có thể xác định được hoàn toàn hệ thống ngay từ đầu, có thể đi theo bước vòng đời cổ điển, nếu các yêu cầu còn chưa được chắc chắn thì có thể sử dụng bản mẫu như một bản hướng dẫn, người phát triển có thể trở lại các bước của vòng đời cổ điển (thiết kế, mã hóa, kiểm thử). Bản mẫu có thể tiến hóa thành hệ thống sản xuất, với việc quay trở về khuôn cảnh vòng đời để kiểm thử.

Các kỹ thuật thứ tư có thể dùng để cài đặt bản mẫu hay cài đặt hệ thống sản xuất trong bước mã hóa của vòng đời. 4GT có thể được dùng kèm với mô hình xoắn ốc cho các bước làm bản mẫu hay mã hóa.

Không nên cứng nhắc trong việc chọn khuôn cảnh cho kỹ nghệ phần mềm. Dựa vào bản chất của ứng dụng mà ấn định ra cách tiếp cận cần được chọn bằng cách tổ hợp các cách tiếp cận thì ích lợi một tổng thể sẽ còn lớn hơn là tổng thể của từng phần.

3. Các giai đoạn trong tiến trình kỹ nghệ phần mềm

Tiến trình kỹ nghệ phần mềm chứa ba giai đoạn chính bất kể với khuôn cảnh kỹ nghệ phần mềm nào được lựa chọn, bao gồm: giai đoạn xác định, phát triển, và bảo trì.

3.1 Giai đoạn xác định làm cái gì ?

Giai đoạn xác định làm **cái gì** bao gồm 3 bước:

Phân tích ở mức hệ thống: xác định vai trò ,chức năng của từng phần tử trong hệ thống mà phần mềm cần được tạo ra để phục vụ nó, chỉ ra vai trò của phần mềm phải làm những công việc gì để phục vụ hệ thống.

Trong quá trình phân tích,cần tập trung vào xác định thông tin nào cần được xử lý, chức năng nào là cấn có, giao diện nào cần được thiết lập, ràng buộc thiết kế nào là hiện có, và tiêu chuẩn hợp lệ nào cần có để xác định ra một hệ thống thành công.

Lập kế hoạch dự án phần mềm: sau khi đã được xác định được nhiệm vụ của phần mềm, phân tích được các rủi ro , xác định được chi phí và tài nguyên được cấp phát, thì phải xác định nhiệm vụ công việc và lập lịch, lên kế hoạch.

Phân tích yêu cầu: Nhiệm vụ của phân tích yêu cầu là tiến hành khảo sát, khám phá, mô tả lại các yêu cầu một cách rõ ràng,mạch lạc,đầy đủ, là mô hình hóa và đặc tả phần mềm. Trong quá trình phân tích,cần tập trung vào xác định thông tin nào cần được xử lý, chức năng nào là cấn có, giao diện nào cần được thiết lập, ràng buộc thiết kế nào là hiện có, và tiêu chuẩn hợp lệ nào cần có để xác định ra một hệ thống thành công.

3.2 Giai đoạn phát triển – làm như thế nào ?

Giai đoạn phát triển bao gồm 3 bước:

Thiết kế phần mềm: diễn đạt các yêu cầu về phần mềm thành các sơ đồ hoặc bảng, hoặc bằng ngôn ngữ , xây dựng các thuật toán , các thủ tục xử lý tính tóan và thủ tục xử lý dữ liệu, mô tả cấu trúc dữ liệu, mô tả các đặc trưng giao diện.

Mã hóa: sử dụng ngôn ngữ lập trình để viết chương trình cho các thuật toán, các thủ tục , tạo các giao diện, cài đặt dữ liệu như đã thiết kế ở bước trước đó.

Kiểm thử phần mềm: sau khi phần mềm đã được cài đặt dưới dạng máy thực hiện được, cần kiểm thử để phát hiện ra các lỗi, khiếm khuyết . Cần kiểm thử với dữ liệu giả định và dữ liệu thật để so sánh kết quả, đánh giá độ chính xác của phần mềm.

3.3 Giai đoạn bảo trì

Sau khi phần mềm đã kiểm tra đạt yêu cầu mới đưa vào khai thác sử dụng và chuyển sang giai đoạn bảo trì. Nhiệm vụ của giai đoạn bảo trì là nhằm bảo đảm cho phần mềm vận hành tốt, thực hiện tốt các yêu cầu của khách hàng đặt ra từ đầu.

Trong Giai đoạn bảo trì có thể phải :

Sửa đổi một số chỗ của phần mềm để khắc phục các khiếm khuyết

Sửa đổi bổ sung một số chỗ của phần mềm để thích nghi với hiện trạng nơi sử dụng hay mở rộng hoàn thiện phần mềm hơn để đáp ứng những yêu cầu mới.

Củng cố

1. Môn học kỹ nghệ phần mềm (SE) phục vụ cho ai là chính, tại sao lại cần nó?
2. Các chủ đề cần quan tâm trong SE?
3. Các thành phần phần mềm được xây dựng bằng cách nào?
4. Phần mềm ứng dụng được phân loại theo những chủ đề nào?
5. Phần mềm và kỹ nghệ phần mềm được anh (chị) hiểu như thế nào?
6. So sánh các cách tiếp cận cơ bản trong tiến trình phát triển phần mềm?
7. Ý nghĩa của việc tổ hợp các khuôn cảnh?
8. Nêu và phân tích các bước tổng quát trong tiến trình SE?

CHƯƠNG 2

PHÂN TÍCH YÊU CẦU VÀ ĐẶC TẢ PHẦN MỀM

Việc hiểu biết đầy đủ về các yêu cầu phần mềm sẽ quyết định sự thành công của phần mềm. Nhiệm vụ của phân tích yêu cầu là một tiến trình khám phá, làm mịn, mô hình hóa và đặc tả. Cả người phát triển và khách hàng đều đóng vai trò quan trọng trong việc phân tích và đặc tả yêu cầu.

1. Những kỹ năng cần có ở Người phân tích(kỹ sư hệ thống)

Trong giai đoạn phân tích, Kỹ sư phân tích cần có kỹ năng :

- Tiến hành theo phương thức "Top-Down" nghĩa là đi từ cái tổng quát tới chi tiết, các chức năng, giao diện và thông tin chủ yếu phải được hiểu hoàn toàn rõ trước khi xác định chi tiết các tầng lớp kế tiếp.
- Người phân tích cũng phải hiểu từng khuôn cảnh phần mềm và đánh giá được các bước kỹ nghệ phần mềm tổng quát, áp dụng được bất kể khuôn cảnh nào cần dùng.

2. Nhiệm vụ phân tích yêu cầu

Việc phân tích yêu cầu phần mềm có thể chia thành 5 bước:

a) Nhận thức vấn đề

b) Đánh giá vấn đề và tổng hợp các giải pháp

c) Mô hình hóa

d) Đặc tả

e) Xét duyệt,thẩm định

Ta xét khái quát 5 bước được tiến hành như sau:

Trước hết tiến hành nghiên cứu bản Phân tích hệ thống và bản Kế hoạch dự án phần mềm(nếu có), hiểu phần mềm trong toàn cảnh hệ thống và xem xét phạm vi phần mềm được dùng để ước lượng quy mô công việc và lên kế hoạch.

Bước thứ 2 là trao đổi giữa các bên để nhận thức ra vấn đề cần giải quyết.

Ta hình dung : bên A là khách hàng, đặt làm phần mềm, bên B là đơn vị làm phần mềm.Bên B có 3 loại cán bộ : người quản lý dự án,người phân tích,người phát triển phần mềm sau công đoạn phân tích.

Người quản lý dự án phải điều phối, thiết lập qua hệ để Người phân tích có thể trao đổi , tiếp xúc với các cấp quản lý và nhân viên kỹ thuật của phía khách hàng và người phát triển phần mềm. Kết quả ở đây là khách hàng và người phân tích đi đến thống nhất phải làm ra một sản phẩm phần mềm có những thành phần,chức năng như thế nào.

Bước thứ 3 là đánh giá vấn đề và tổng hợp các giải pháp, cần thực hiện các côgn việc :

+ Xác định các luồng tin và nội dung thông tin trong từng luồng

+ Xác định và soạn thảo các chức năng phần mềm

+Dự báo được phần mềm sẽ thực thi như thế nào theo tình huống của các sự kiện ảnh hưởng tới hệ thống

+Thiết lập các đặc trưng giao diện

+Vạch ra được những hạn chế

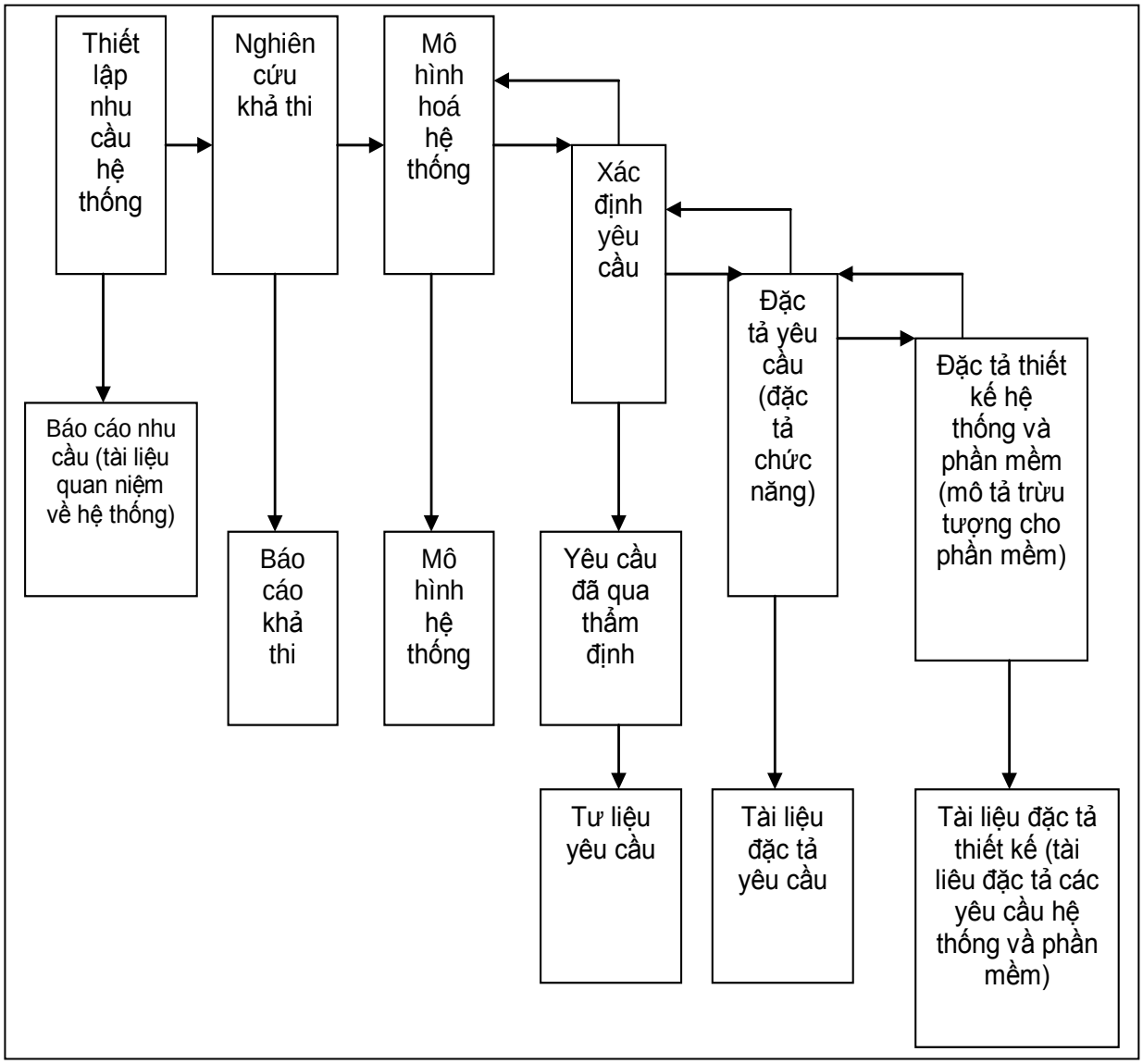
Một khi hiểu được khách hàng muốn gì thì người phân tích mới xác định được hệ thống mới cần phải tạo ra thông tin nào và dữ liệu nào cần được cung cấp cho hệ thống. Một khi đánh giá được các vấn đề hiện tại và thông tin mong muốn, người phân tích mới xác định được giải pháp và xác định những bước phát triển kế tiếp.

Bước thứ tư là mô hình hóa. Thông qua ước lượng và tổng hợp giải pháp, người phân tích tạo ra các mô hình hệ thống nhằm hiểu rõ hơn luồng dữ liệu và điều khiển xử lý chức năng, thao tác hành vi và nội dung thông tin. Mô hình này được lấy làm nền tảng cho thiết kế phần mềm

Bước cuối cùng là tạo ra một đặc tả phần mềm. Kết quả của công việc phân tích yêu cầu là bản đặc tả yêu cầu phần mềm. Đó là tư liệu chính thức đầu tiên được tạo ra trong quy trình xây dựng phần mềm.

3.Việc hình thành các yêu cầu như thế nào ?

Phân tích và định rõ yêu cầu là hướng tới đặc tả yêu cầu phần mềm được thể hiện trong các khuôn cảnh như sau:



Việc phân tích và nắm bắt yêu cầu là giai đoạn đầu của quá trình thiết lập các *dịch vụ* (mà hệ thống phải giải quyết) và các ràng buộc (mà hệ thống phải tuân theo). Các thông tin của vấn đề cần giải quyết phải được thu thập, phân tích và phải được xác định một cách rõ ràng. Khi đó thì giải pháp phần mềm mới có thể được thiết kế và thực thi. Để giải quyết vấn đề này người ta phải thực hiện các bước đầu tiên của tiến trình phân tích hệ thống như xác định nhu cầu, mô hình hóa hệ thống (giai đoạn tiền khả thi).

- Phân tích yêu cầu là một tiến trình khám phá, làm mịn, mô hình hóa và đặc tả. Phạm vi phần mềm, ban đầu do người phân tích thiết lập sơ bộ sau đó sẽ được chi tiết thêm các phần :

- Các mô hình thông tin cần tới
- Luồng thông tin
- Hành vi vận hành
- Nội dung dữ liệu được tạo ra

Người cân bộ tin tọc làm công tác phân tích yêu cầu phải có khả năng nghe và hiểu được khách hàng muốn cái gì, yêu cầu cái gì, từ những phát biểu có thể là tản mạn,không ăn khớp nhau của khách hàng phải biết xâu chuỗi, ghép nối, hình thành nên những yêu cầu rõ ràng,cụ thể mà máy tính điện tử có thể giải quyết được.

4.Việc xác định các yêu cầu

Xác định yêu cầu là mô tả trừu tượng các *dịch vụ* (mà hệ thống được mong đợi phải cung cấp) và các ràng buộc (mà hệ thống phải tuân thủ khi vận hành). Nó chỉ đặc tả tính chất bên ngoài của hệ thống mà không hề liên quan đến các đặc tính thiết kế. Nó phải được viết sao cho người ta có thể hiểu được mà không cần một kiến thức chuyên môn đặc biệt nào.

Các yêu cầu được chia làm hai loại:

1. Các yêu cầu hệ thống chức năng: các dịch vụ mà hệ thống phải cung cấp
2. Các yêu cầu phi chức năng: các ràng buộc mà hệ thống phải tuân theo

Xác định yêu cầu thường được viết bằng ngôn ngữ tự nhiên cộng thêm việc dùng các bảng, các biểu đồ để cho người dùng dễ hiểu (xem như người dùng không biết các khái niệm chuyên môn).

Chú ý:

- Vì việc xác định yêu cầu khó hoàn thiện trước khi bắt đầu phát triển hệ thống nên việc áp dụng mô hình bản mẫu sẽ thích hợp hơn là mô hình thác nước.

Các báo cáo về yêu cầu phần mềm phải viết như thế nào ?

1. Chỉ đặc tả tính chất bên ngoài của hệ thống
2. Đặc tả các ràng buộc về sự thực hiện
3. Phải là dễ thay đổi,có khả năng thích nghi với các thay đổi sẽ diễn ra
4. Phải được dùng làm công cụ tham khảo cho người bảo trì hệ thống
5. Phải báo cáo dự tính trước về vòng đời của hệ thống

Cấu trúc của một tư liệu yêu cầu được gợi ý theo kết cấu sau:

1. Phần dẫn nhập
2. Phần mô hình hệ thống
3. Phần tiến triển của hệ thống

4. Phần các yêu cầu chức năng
5. Phần từ điển thuật ngữ

5. Đặc tả phần mềm

5.1 Cách đặc tả và biểu diễn

5.1.1 Khái niệm Đặc tả - specification

Đặc tả một vấn đề là mô tả các đặc trưng của vấn đề đó. Vấn đề có thể là đối tượng, khái niệm hoặc một thủ tục nào đó, ...

Yêu cầu đầu tiên của đặc tả là tính chính xác

Các đặc tả thường mang tính trừu tượng. Càng ở mức cao (những mức đầu tiên của quá trình làm mịn hoặc chính xác hóa) đặc tả càng trừu tượng,khái quát . Càng xuống các mức thấp, đặc tả càng tiếp cận dần tới cụ thể - tức là tới một thể hiện trên một máy tính cụ thể với một ngôn ngữ lập trình cụ thể.

- Đặc tả hình thức: là những đặc tả chính xác tức là không thể dẫn tới những cách hiểu khác nhau. Đặc tả hình thức sử dụng công cụ chủ yếu là đại số và logic (formal)
- Đặc tả phi hình thức: diễn đạt bằng những ngôn ngữ, tuy không chặt chẽ nhưng được nhiều người biết và có thể trao đổi với nhau để chính xác hoá những điểm chưa rõ, những khái niệm mơ hồ
- Đặc tả hỗn hợp: phối hợp hai kiểu đặt tả trên

Trong thực tế, có nhiều loại hình đặc tả, ví dụ như: Đặc tả cấu trúc dữ liệu (mô tả các thành phần của dữ liệu ...), đặc tả chức năng (mô tả chức năng thông qua việc mô tả tính chất của input, output ...), đặc tả đối tượng (bao gồm đặc tả cấu trúc và đặc tả chức năng ...), đặc tả thao tác (mô tả các thao tác cần thực hiện ...), đặc tả cú pháp (mô tả cách lắp ghép các kí hiệu, các từ lại thành chương trình ...), đặc tả xử lý, đặc tả thuật toán...

Kiểu đặc tả cần phù hợp với giải pháp. Các yêu cầu phần mềm có thể được phân tích theo một số cách khác nhau. Các kỹ thuật phân tích có thể dẫn tới những đặc tả trên giấy hay trên máy tính (được xây dựng nhờ dùng CASE) có chứa các mô tả ngôn ngữ đồ họa và tự nhiên cho yêu cầu phần mềm. Việc làm bản mẫu đã giúp đặc tả thực hiện được, tức là bản mẫu thể hiện một biểu diễn của các yêu cầu phần mềm. Các ngôn ngữ đặc tả hình thức dẫn tới biểu diễn hình thức.

5.1.2 Biểu diễn

Các yêu cầu phần mềm có thể được biểu diễn theo nhiều cách. Các biểu diễn tốt nên tuân theo hướng dẫn sau:

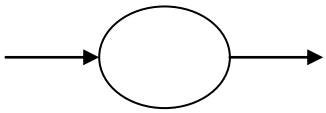
Định dạng và nội dung biểu diễn theo hướng liên quan tới vấn đề:

- Theo một dàn bài chung cho nội dung của bản đặc tả các yêu cầu phần mềm.
- Dạng biểu diễn có trong bản đặc tả có thể thay đổi theo lĩnh vực ứng dụng. (Chẳng hạn, đặc tả cho hệ thống tự động hóa chế tạo sẽ dùng cách kí hiệu khác, biểu đồ và ngôn ngữ khác với đặc tả cho trình biên dịch ngôn ngữ lập trình)

Thông tin chứa trong bản đặc tả nên được lồng nhau:

- Các biểu diễn nên làm lộ ra các tầng thông tin sao cho độc giả có thể di chuyển tới mức chi tiết mình mong muốn.
- Đoạn và các sơ đồ đánh dấu nên chỉ ra mức độ chi tiết đang được trình bày. Đôi khi cũng nên trình bày cùng một thông tin ở các mức trừu tượng khác nhau để hiểu tốt hơn.

Các biểu đồ và các dạng kí pháp khác nên giảm thiểu và nhất quán trong sử dụng: Chẳng hạn, cách kí hiệu như sau có thể hiểu theo nhiều cách:



có thể giải thích ít nhất ba (hoặc 5 hay 6) cách khác nhau. Lẫn lộn hay không nhất quán trong kí pháp, dù là đồ họa hay kí hiệu cũng đề làm suy giảm việc hiểu và làm phát sinh lỗi.

Biểu diễn nên thường được xem lại: Nội dung của đặc tả sẽ thay đổi. Vì vậy biểu diễn nên thường được xem lại để đảm bảo tính thống nhất. Một cách lý tưởng, các công cụ CASE nên có sẵn để cập nhật tất cả các biểu diễn bị ảnh hưởng bởi từng thay đổi.

Nên sử dụng các kí hiệu, sơ đồ quen thuộc nhưng có chọn lọc: Người ta đã tiến hành nhiều cuộc điều tra về nhân tố con người liên quan đến đặc tả. Dường như ít có hoài nghi rằng cách kí hiệu và thu xếp có ảnh hưởng tới việc hiểu. Tuy nhiên các kỹ sư thích các dạng ký hiệu, các sơ đồ riêng biệt. Sự quen thuộc thường thuận cho mọi người, nhưng các nhân tố chọn lọc như cách bố trí không gian, các mẫu hình dễ nhận thức và mức hợp lý của hình thức sẽ giúp cho việc đặc tả có lợi về sau.

5.2 Các nguyên lý đặc tả

Đặc tả có thể được xem như một tiến trình biểu diễn. Mục đích cuối cùng của đặc tả các yêu cầu được biểu thị sao cho dẫn tới việc cài đặt phần mềm thành công. Balzer và Goldman đề nghị 8 nguyên lý đặc tả tốt:

***Nguyên lý 1*:** Phân biệt chức năng với cài đặt- phân biệt giữa cái gì(what) – chứ kg phải làm như thế nào(how)

Trước hết, theo định nghĩa, đặt tả là một mô tả về điều mong muốn, chứ không phải là cách thực hiện nó (cài đặt). Đặc tả có thể chấp nhận hai dạng hoàn toàn khác nhau. Dạng thứ nhất là dạng của các hàm toán học: với một tập cái vào đã cho, tạo ra một tập cái ra đặc biệt. Dạng tổng quát của những đặc tả như thế là tìm ra một hoặc tất cả những kết quả ứng với P (cái vào), với P biểu thị một tân từ bất kỳ. Trong những đặc tả như thế, kết quả cần thu được phải hoàn toàn được diễn đạt *cái gì* không phải là *thế nào*.

***Nguyên lý 2*:** Cần tới ngôn ngữ đặc tả hệ thống hướng tiến trình

Xét tình huống trong đó môi trường là động và sự thay đổi của nó ảnh hưởng tới hành vi của thực thể nào đó tương tác với môi trường đó. Hành vi của nó không thể biểu diễn được ở dạng hàm toán học của cái vào. Thay vì thế, cần phải sử dụng cách biểu diễn khác: cách mô tả hướng tiến trình, trong đó đặc tả cái gì đạt được bằng cách xác định một mô hình hành vi mong muốn của hệ thống dưới dạng các đáp ứng chức năng đối với các kích thước khác nhau từ môi trường.

Những đặt tả hướng tiến trình như vậy:

1. Trình bày một mô hình về hành vi hệ thống
2. Thông thường không thuộc ngôn ngữ đặc tả hình thức
3. Lột tả được bản chất của nhiều tình huống phức tạp cần phải đặc tả
4. Trong những tình huống cần tự động hóa, cả tiến trình lẫn môi trường tồn tại của nó đều phải được mô tả một cách hình thức. Muốn vậy, toàn bộ hệ thống các bộ phận tương tác phải được đặc tả chứ không chỉ đặc tả một thành phần

***Nguyên lý 3*:** Đặc tả phải bao gồm hệ thống trong đó phần mềm là một thành phần(đảm bảo tính hệ thống, thể hiện mối liên kết giữa phần mềm cần xây dựng với các thành phần khác).

Một hệ thống bao gồm các thành phần tương tác nhau. Chỉ bên trong hoàn cảnh của toàn bộ hệ thống và tương tác giữa các thành phần của nó thì hành vi của một thành phần riêng mới có thể được xác định. Nói chung, một hệ thống được mô hình hóa như một tập hợp các mối quan hệ giữa các thụ động. Những sự vật này có liên quan lẫn nhau và qua thời gian dẫn đến mối quan hệ giữa các sự vật thay đổi. Mối quan hệ động này đưa ra sự kích thích cho các sự vật tích cực, còn gọi là các tác nhân, đáp ứng. Sự đáp ứng có thể gây ra những thay đổi thêm nữa, và do đó, tạo ra thêm kích thích để cho các tác nhân có thể đáp ứng lại.

***Nguyên lý 4*:** Đặc tả bao gồm cả môi trường mà hệ thống vận hành(có xét đến môi trường xung quanh)

Môi trường trong đó hệ thống vận hành và các tương tác phải được xác định.

Bản thân môi trường cũng là một hệ thống bao gồm các sự vật tương tác, cả tích cực lẫn thụ động, mà trong hệ thống chúng chỉ là một tác nhân. Các tác nhân khác, theo định nghĩa là không thay đổi bởi vì chúng là một phần của môi trường, giới hạn phạm vi của việc thiết kế và cài đặt về sau. Trong thực tế, sự khác nhau duy nhất giữa hệ thống và môi trường của nó là ở chỗ nỗ lực thiết kế và cài đặt về sau sẽ vận hành chỉ trong đặc tả cho hệ thống. Đặc tả môi trường làm cho "giao diện" của hệ thống được xác định theo cùng cách như bản thân hệ thống chứ không đưa vào cách hình thức hóa khác.

Đặc tả hệ thống chính là bức tranh của tập hợp các tác nhân xoắn xuýt nhau cao độ, phản ứng lại những kích thích trong môi trường (thay đổi các sự vật). Chỉ có thông qua những hành động điều phối của tác nhân mà hệ thống mới đạt được các mục tiêu của nó. Thiết kế tuân theo đặc tả và quan tâm đến việc phân rã một đặc tả thành các mẩu gần tách biệt để chuẩn bị cho cài đặt. Tuy nhiên đặc tả phải vẽ lại chính xác bức chân dung của hệ thống và môi trường của nó như cộng đồng người dùng cảm nhận tới mức chi tiết, phục vụ cho các giai đoạn thiết kế và cài đặt. Vì mức độ chi tiết cần thiết này là khó thấy trước, nếu không nói là không thể, nên đặc tả, thiết kế và cài đặt phải được thừa nhận như một hoạt động tương tác. Do đó, điều mấu chốt là công nghệ cần bao quát thật nhiều các hoạt động này khi bản đặc tả được soạn thảo và thay đổi (trong cả hai giai đoạn phát triển khởi đầu và bảo trì về sau)

***Nguyên lý 5*:** Đặc tả hệ thống phải là một mô hình nhận thức- nghĩa là ngươi đọc phải hiểu được vấn đề thông qua đặc tả

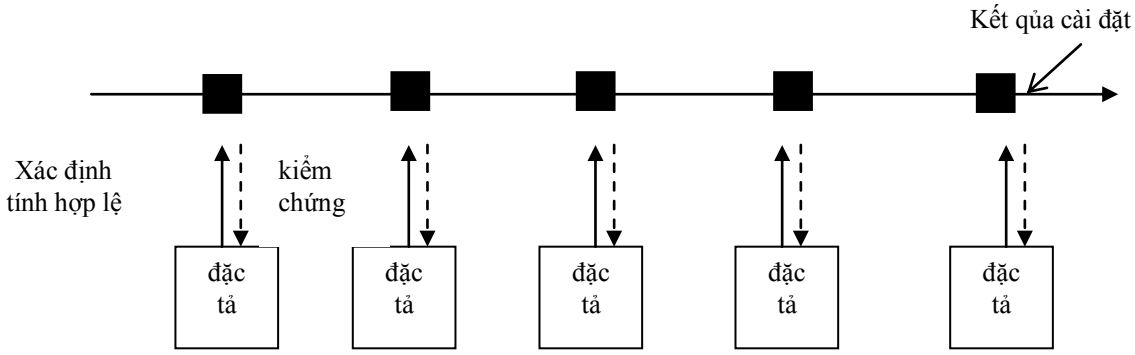
- Đặc tả hệ thống phải là một mô hình nhận thức chứ không phải là một mô hình thiết kế hay cài đặt
- Mô tả một hệ thống sao cho đạt được sự cảm nhận của cộng đồng người sử dụng. Các sự vật mà nó thao tác phải tương ứng với các sự vật của lĩnh vực đó; các tác nhân phải được mô hình hóa cho các cá nhân, tổ chức và trang thiết bị trong lĩnh vực đó; còn các hành động họ thực hiện thì phải được mô hình hóa cho những hoạt động thực tế xuất hiện trong lĩnh vực
- Phải có khả năng tổ hợp vào trong đặc tả những quy tắc hay luật bao trùm các sự vật thuộc lĩnh vực:

  Một trong những luật này bài trừ những trạng thái nào đó của hệ thống (như "hai sự vật không thể đồng thời cùng một chỗ và vào cùng một lúc") và do đó giới hạn hành vi của các tác nhân hay chỉ ra nhu cầu bổ trợ để ngăn cản những trạng thái khỏi nảy sinh.

***Nguyên lý 6*:** Đặc tả phải thể hiện tính vận hành- nghĩa là đọc tài liệu đặc tả hình dung ra được sự vận hành của phần mềm.

Đặc tả phải đầy đủ và mang tính hình thức để có thể được dùng trong việc xác định rằng liệu một cài đặt được đề nghị có thỏa mãn đặc tả cho những trường hợp kiểm thử tùy ý không. Tức là, với kết quả của việc cài đặt trên một tập dữ liệu được chọn một cách tùy ý, phải có thể dùng đặc tả để xác định tính hợp lệ cho những kết quả đó. Điều này kéo theo rằng đặc tả, mặc dầu không phải là một đặc tả hoàn toàn về cách thức, vẫn có thể hành động như một bộ sinh các hành vi. Do đó theo nghĩa mở rộng, đặc tả này phải thể hiện tính vận hành …

Quá trình cài đặt



H 2.4 *Quá trình cài đặt*

***Nguyên lý 7*:** có khả năng update

Đặc tả hệ thống chấp nhận dung sai về tính không đầy đủ và vì vậy có tính nâng cao

Không đặc tả nào có thể là đầy đủ hoàn toàn. Môi trường mà hệ thống tồn tại trong đó quá phức tạp. Một đặc tả bao giờ cũng là một mô hình - một sự trừu tượng hóa - của một tình huống thực (hay được mường tượng) nào đó. Do đó nó sẽ không đầy đủ. Hơn thế nữa, nó tồn tại ở nhiều mức chi tiết. Các công cụ phân tích được sử dụng để giúp đặc tả và kiểm thử đặc tả phải có khả năng xử lý với tính không đầy đủ.

***Nguyên lý 8*:** Đặc tả phải được cục bộ hóa và có khả năng lắp ghép- xuất phát từ lập trình hướng đối tượng.

Các nguyên lý trước xử lý đặc tả như một thực thể tĩnh. Nguyên lý này nảy sinh từ tính động của đặc tả. Cần phải thừa nhận rằng mặc dầu mục tiêu chính của một đặc tả là dùng làm cơ sở cho thiết kế và cài đặt một hệ thống nào đó, nó không phải là một sự vật tĩnh dựng sẵn mà là một vật động đang trải qua thay đổi đáng kể.

Việc thay đổi (động) của đặc tả xuất hiện trong 3 hoạt động chính:

- phát biểu: khi một đặc tả ban đầu đang được tạo ra
- phát triển: khi đặc tả được soạn thảo trong quá trình thiết kế
- lặp: để phản ánh môi trường đã thay đổi và/hoặc các yêu cầu chức năng phụ.

Với nhiều thay đổi xuất hiện đối với đặc tả, điều mấu chốt là nội dung và cấu trúc của đặc tả được chọn để làm phù hợp thay đổi này. Yêu cầu chính cho sự phù hợp đó là ở chỗ:

- thông tin bên trong đặc tả phải được cục bộ hóa sao cho chỉ một phần nhỏ (một cách lý tưởng) cần phải sửa đổi khi thông tin thay đổi.
- đặc tả cần được cấu trúc (ghép) một cách lỏng lẻo cho từng phần có thể được thêm vào hay loại bỏ một cách dễ dàng, và cấu trúc được điều chỉnh một cách tự động.

-

Các mức trừu tượng của đặc tả

Các đặc tả được thể hiện ở vài mức trừu tượng khác nhau cùng với mối tương quan giữa các mức ấy. Mỗi mức nhắm đến các đối tượng đọc khác nhau mà họ có quyền quyết định về việc mua sắm và thực hiện.

Các mức đó là:

Định ra yêu cầu:

- thể hiện bằng ngôn ngữ tự nhiên về các dịch vụ mà hệ thống sẽ phải cung cấp
- phải được viết sao cho dễ hiểu đối với khách hàng và người quản lý hợp đồng, người sẽ mua sắm sẽ sử dụng

Đặc tả yêu cầu:

- tài liệu nêu ra các dịch vụ một cách chi tiết hơn. Tài liệu này (thường được gọi là đặc tả chức năng)
- đòi hỏi chính xác tới mức nó có thể làm cơ sở cho hợp đồng giữa người mua sắm hệ thống và người phát triển phần mềm.
- được viết dễ hiểu đối với các nhân viên kĩ thuật ở cả nơi mua lẫn nơi phát triển
- kỹ thuật đặc tả hình thức hẳn là thích hợp cho các đặc tả kiểu như vậy, nhưng nó cũng tùy thuộc ở trình độ kiến thức cơ bản của người mua sắm hệ thống.

Đặc tả phần mềm/ đặc tả thiết kế (đây là một mô tả trừu tượng cho phần mềm):

- dùng làm cơ sở cho việc thiết kế và thực thi
- thể hiện một quan hệ rõ ràng giữa tư liệu này và đặc tả yêu cầu.
- đối tượng đọc ở đây chủ yếu là các kĩ sư phần mềm chứ không phải là người sử dụng hoặc người quản lý
- sử dụng kĩ thuật đặc tả hình thức là thích hợp cho tư liệu này.

5.4 Đặc tả yêu cầu

Việc định yêu cầu là việc đặc tả hướng khách hàng và được viết bởi ngôn ngữ của khách hàng. Khi đó có thể dùng các khái niệm không chính xác, ngôn ngữ tự nhiên và các biểu đồ. Đặc tả yêu cầu (được gọi là đặc tả chức năng) là cơ sở cho hợp đồng làm hệ thống nó phải không được mơ hồ, mà mơ hồ đó có thể dẫn tới sự hiểu lầm bởi khách hàng hoặc bởi người ký hợp đồng.

Rất nhiều vấn đề của kỹ nghệ phần mềm là khó khăn đối với đặc tả yêu cầu. Với một yêu cầu mơ hồ thì người phát triển sẽ thực hiện nó sao cho rẻ nhất. Trong khi đó khách hàng lại không muốn như vậy. Sau các cuộc đấu khẩu kéo dài thì các yêu cầu mới sẽ được thiết lập và có các đổi thay đối với hệ thống. Dĩ nhiên việc đó làm trì hoàn ngày phân phát hệ thống và làm tăng chi phí.

Chi phí do các sai sót trong việc phát biểu các yêu cầu có thể là rất cao, đặc biệt là nếu các sai sót này còn vẫn chưa được phát hiện khi hệ thống được thực hiện. Bochm báo cáo rằng trong một vài hệ thống lớn thì có đến 95% các mã đã phải viết lại để thỏa mãn cá yêu cầu của người dùng đã thay đổi và 12% các lỗi được phát hiện trong quá trình ba năm đầu sử dụng là các lỗi do đặc tả yêu cầu không đúng đắn.

Chi phí do thay đổi một yêu cầu là đắt hơn nhiều so với chi phí sửa chữa thiết kế hoặc do lỗi mã.

5.4.1 Những hạn chế của việc đặc tả bằng ngôn ngữ tự nhiên

1. Người đọc và người viết đặc tả bằng ngôn ngữ tự nhiên buộc phải hiểu mỗi từ theo cùng một khái niệm. Điều đó là không thể được do sự mơ hồ và bản tính cố hữu của ngôn ngữ tự nhiên và cũng vì không có một từ điển thuật ngữ máy tính chuẩn mực.
2. Một đặc tả bằng ngôn ngữ tự nhiên là quá mềm dẻo: nó cho phép các yêu cầu liên quan được biểu thị trong các cách hoàn toàn khác nhau.
3. Các yêu cầu là không phân hoạch được một cách hữu hiệu: hiệu quả của việc đổi thay chỉ có thể được xác định bởi toàn bộ các yêu cầu chứ không phải là một nhóm các yêu cầu liên quan

Có người đề nghị rằng các ngôn ngữ đặc tả hình thức toán học nên được dùng để biểu thị các yêu cầu hệ thống. Nhưng đa số các khách hàng lại không hiểu đặc tả hình thức toán học. Hall (1990) đề xuất con đường để đi vòng qua khó khăn đó là viết thêm các chú giải dài dòng ngay bên cạnh cho người dùng dễ hiểu. Khi nào người dùng trở nên quen thuộc hơn với những ưu điểm của đặc tả hình thức và các kỹ sư phần mềm không còn miễn cưỡng dùng các phương pháp hình thức thì sẽ có nhiều cách đặc tả yêu cầu sẽ dựa trên các mô hình hình thức của chức năng hệ thống

5.4.2 Các yêu cầu phi chức năng

Một yêu cầu phi chức năng của hệ thống là một hạn chế hoặc ràng buộc về các dịch vụ của hệ thống. Các yêu cầu đó có thể được đưa ra:

- vì nhu cầu của người dùng
- vì hạn chế của kinh phí
- vì chính sách của tổ chức
- vì sự cần thiết tương tác giữa các phần cứng và phần mềm hoặc
- vì các nhân tố bên ngoài như các quy tắc an toàn, luật lệ bí mật riêng tư, ...

Có 3 kiểu yêu cầu phi chức năng chính:

1. Các yêu cầu sản phẩm: đây là các yêu cầu về hệ thống được phát triển, chẳng hạn yêu cầu về tốc độ, về bộ nhớ, về độ tin cậy, về tính di chuyển được và về tính dùng lại được
2. Các yêu cầu về quá trình: đây là các yêu cầu về quá trình phát triển, chẳng hạn như các chuẩn cần phải theo, các yêu cầu về sự thức hiện như các ngôn ngữ lập trình, phương pháp thiết kế, yêu cầu về phân phát
3. Các yêu cầu ngoại lai: tức là các yêu cầu không phải về sản phẩm cũng không phải về quá trình phát triển; chẳng hạn như về giao tiếp với các hệ thống khác, về pháp lý, về chi phí, về nhóm những người phát triển và ...

Tùy theo các tổ chức cụ thể, đặc tả yêu cầu có thể được thể hiện bằng các cách khác nhau kể từ mức phát biểu bằng ngôn ngữ tự nhiên về các dịch vụ mà hệ cần cung cấp đến mức đặc tả hệ thống một cách hình thức kiểu toán học. Ranh giới giữa các yêu cầu và đặc tả hình thức là một thứ rất tinh tế và các thuật ngữ này lại còn có thể được dùng theo nghĩa như nhau.

5.4.3 Khó khăn của việc xác định đặc tả yêu cầu

1. Các hệ phần mềm lớn thường được yêu cầu cải thiện dựa trên nguyên trạng: hoặc không có hệ thống nào hoặc có một hệ thống không đầy đủ. Mặc dầu có thể biết các khó khăn của hệ thống hiện có nhưng lại rất khó dự đoán được hiệu quả của hệ thống đã được cải thiện đối với tổ chức đó.

2. Các hệ thống lớn thường có một cộng đồng người sử dụng đa dạng, ho có những yêu cầu và ưu tiên khác nhau, thậm chí là mâu thuẫn với nhau. Cuối cùng thì các yêu cầu hệ thống cũng không thể tránh được sự thỏa hiệp.
3. Những người bỏ tiền ra mua sắm hệ thống và những người sử dụng hệ thống hiếm khi lại là một. Những người mua sắm hệ thống đặt ra các yêu cầu theo những ràng buộc của tổ chức cơ quan và của ngân sách. Những cái đó lại thường mâu thuẫn với các yêu cầu thực sự của người dùng.

5.4.4 Thẩm định yêu cầu

Một khi đã thiết lập các yêu cầu thì phải thẩm định xem chúng có thỏa mãn các nhu cầu của người mua hệ thống không. Nếu việc thẩm định không phù hợp thì các sai lầm có thể lan truyền sang các giai đoạn thiết kế và thực hiện. Khi đó có thể cần đến các sự nâng cấp hệ thống rất tốn kém để làm cho nó đúng đắn, cần xác minh những vấn đề sau:

Có 4 vấn đề liên quan đến việc thẩm định yêu cầu,

1. Phải chỉ ra rằng các nhu cầu của người dùng là được thỏa mãn
2. Các yêu cầu phải không gây ra mâu thuẫn nhau
3. Các yêu cầu phải đầy đủ: chúng phải chứa mọi chức năng và mọi ràng buộc mà người dùng đã nhắm đến
4. Các yêu cầu phải là hiện thực.

5.5 Dàn bài đặc tả yêu cầu phần mềm

Bản đặc tả các yêu cầu phần mềm được tạo ra tại đỉnh cao của nhiệm vụ phân tích

Chức năng và hiệu năng được cấp phát cho phần mềm xem như một phần của kỹ nghệ hệ thống thường được làm mịn bằng cách thiết lập phần mô tả thông tin đầy đủ như mô tả chức năng chi tiết, chỉ báo về các yêu cầu hiệu năng và những ràng buộc thiết kế, các tiêu chuẩn hợp lệ thích hợp và những dữ liệu khác thường cần tới. Cơ quan tiêu chuẩn quốc gia, IEEE và Bộ quốc phòng Mỹ đề nghị các định dạng cho các đặc tả yêu cầu phần mềm. Dàn bài đơn giản hóa được trình bày trong bảng sau:

Dàn bài đặc tả yêu cầu phần mềm

I. Giới thiệu	
A. Đại cương về hệ thống	(mục tiêu, mục đích)
B. Mô tả chung	(cấu trúc)
C. Các ràng buộc dự án phần mềm	(phạm vi)
II. Mô tả thông tin	(chi tiết)
A. Biểu diễn luồng thông tin	(dựa vào cấu trúc thông tin)
1. Luồng dữ liệu	
2. Luồng điều khiển	(đặc biệt chú ý đến khi xét các hệ thời gian thực)
B. Biểu diễn nội dung thông tin	
C. Mô tả giao diện hệ thống	(cho từng chức năng)
III. Mô tả chức năng	

A. Phân hoạch chức năng	(cho từng chức năng)
B. Mô tả chức năng	(lời giải thích)
1. Tường thuật về cách xử lý	
2. Hạn chế/giới hạn	(về các mặt kinh tế, xã hội, kỹ thuật, chi phí, thời gian)
3. Yêu cầu hiệu năng	(theo từng modul)
4. Ràng buộc thiết kế	(có luận giải)
5. Biểu đồ trợ giúp	(cấu trúc tổng thể, tương quan với các phần tử hệ thống khác)
C. Mô tả điều khiển	
1. Đặc tả điều khiển	(có luận giải)
2. Ràng buộc thiết kế	(xem xét vận hành của phần mềm)
IV. Mô tả hành vi	(xem xét vận hành của phần mềm)
A. Trạng thái hệ thống	(mức toàn cảnh, mức đỉnh)
B. Sự kiện và hành động	(mức cụ thể, bộ phận)
V. Tiêu chuẩn hợp lệ	
A. Giới hạn hiệu năng	
B. Lớp các kiểm thử	
C. Đáp ứng phần mềm trông đợi	(tiêu chuẩn hướng tới (lý tưởng))
D. Các xem xét đặc biệt	
VI. Sách tham khảo	(tài liệu kỹ nghệ phần mềm khác, tham khảo kỹ thuật, ...)
VII. Phụ lục	(Danh sách các nhà cung cấp, các chuẩn, ...)

Trong nhiều trường hợp bản đặc tả yêu cầu phần mềm còn có thể kèm theo một tài liệu sơ bộ của người dùng trình bày số liệu cần đưa vào và kết quả cần đưa ra..

5.6 Xét duyệt đặc tả

Việc xét duyệt bản đặc tả các yêu cầu phần mềm (và hoặc bản mẫu) do cả người phát triển phần mềm và khách hàng cùng tiến hành. Bởi vì đặc tả tạo nên nền tảng cho giai đoạn phát triển nên cần phải rất thận trọng khi tiến hành cuộc họp xét duyệt. Có 2 mức xét duyệt.

5.6.1 Mức vĩ mô

Việc xét duyệt trước hết được tiến hành ở mức vĩ mô. Tại mức này, người xét duyệt cố gắng đảm bảo rằng bản đặc tả được đầy đủ, nhất quán và chính xác. Cần đề cập tới các câu hỏi sau:

- Các mục tiêu và mục đích được thiết lập cho phần mềm có nhất quán với mục tiêu và mục đích của hệ thống hay không?
- Những giao diện quan trọng với mọi phần tử hệ thống đã được mô tả chưa?
- Luồng và cấu trúc thông tin đã được mô tả thích hợp cho lĩnh vực vấn đề chưa?

- Các biểu đồ có rõ ràng không? Liệu mỗi biểu đồ có thể đứng riêng không cần lời giải thích không?
- Các chức năng chính có còn bên trong phạm vi và đã được mô tả thích hợp chưa?
- Liệu hành vi của phần mềm có nhất quá với thông tin nó phải xử lý và chức năng nó phải thực hiện hay không?
- Các ràng buộc thiết kế có thực tế không.
- Rủi ro công nghệ phát triển là gì?
- Các yêu cầu phần mềm khác đã được xem xét đến chưa?
- Các tiêu chuẩn hợp lệ đã được phát biểu chi tiết chưa?
- Liệu có sự không nhất quán, bỏ sót hay dư thừa nào không?
- Việc tiếp xúc với khách hàng có đầy đủ không?
- Người dùng đã xét duyệt bản Tài liệu sơ bộ của người dùng hay bản mẫu chưa?
- Các ước lượng về Kế hoạch dự án phần mềm bị ảnh hưởng thế nào?

…

5.6.2 Mức chi tiết

Để đưa ra câu trả lời cho nhiều câu hỏi trên, việc xét duyệt có thể tập trung vào mức chi tiết. Tại đây, mối quan tâm tập trung vào từ ngữ của bản đặc tả.

Việc xét duyệt chi tiết bản đặc tả có những gợi ý sau:

- Phải quan sát các từ nối có sức thuyết phục (như "chắc chắn", "do đó", "rõ ràng", "hiển nhiên", "từ đó suy ra rằng") và hỏi "tại sao chúng lại có ở đó?"
- Theo dõi những thuật ngữ mung lung (như "một số", "đôi khi", "thường", "thông thường", "bình thường", "phần lớn", "đa số"); để yêu cầu làm sáng tỏ
- Khi có nêu danh sách, nhưng không đầy đủ thì phải đảm bảo mọi khoản mục đề được hiểu rõ. Chú ý vào các từ như "vân vân", "cứ như thế", "cứ tiếp tục như thế", "sao cho", ...
- Phải chắc chắn phát biểu phạm vi không chứa những giả thiết không được nói rõ (như "mã hợp lệ trong khoảng 10 tới 100". Đó là số nguyên, số thực hay số hệ 16?)
- Phải nhận biết về các động từ mơ hồ như "xử lý", "loại bỏ". Có thể có nhiều cách hiểu về nó.
- Phải nhận biết các đại từ "vu vơ" (như "modul vào/ra liên lạc với modul kiểm tra tính hợp lệ dữ liệu và đặt cờ báo kiểm soát của nó")
- Tìm các câu có chứa sự chắc chắn (như "bao giờ", "mọi", "tất cả", "không một", "không bao giờ") rồi yêu cầu bằng chứng.
- Khi một thuật ngữ được định nghĩa tường minh tại một chỗ thì hãy thử thay thế định nghĩa này vào chỗ xuất hiện của nó
- Khi một cấu trúc được mô tả theo lời thì hãy vẽ ra bức tranh để giúp hiểu được nó
- Khi một tính toán được xác định thì hãy thử với ít nhất 2 ví dụ

…

Một khi việc xét duyệt đã hoàn tất thì bản đặc tả các yêu cầu phần mềm sẽ được cả khách hàng lẫn người phát triển "ký tắt". Bản đặc tả trở thành "hợp đồng" cho việc phát triển phần mềm. Những thay đổi trong yêu cầu được nêu ra sau khi bản đặc tả đã hoàn thành sẽ không bị loại bỏ, nhưng khách hàng phải lưu ý rằng từng thay đổi sau khi ký đều là một mở rộng của phạm vi phần mềm và do đó có thể làm tăng thêm chi phí và/hoặc kéo dài lịch biểu.

Bản đặc tả rất khó "kiểm thử" theo một cách có nghĩa, do đó sự không nhất quán hay thiếu sót có thể bị bỏ qua không chú ý tới. Trong khi xét duyệt, người ta có thể khuyến cáo những thay đổi cho bản đặc tả. Có thể sẽ khó khăn để định lượng tác động toàn cục của thay đổi, tức là làm sao việc thay đổi trong một chức năng lại ảnh hưởng tới các yêu cầu cho chức năng khác? Người ta đã phát triển các công cụ đặc tả tự động hóa để giúp giải quyết vấn đề này.

6. Kỹ nghệ hệ thống và tạo nguyên mẫu

6.1 Kỹ nghệ hệ thống – system engineering

6.1.1 Các hoạt động cơ bản trong tiến trình phân tích hệ thống

Bước 1: Xác định nhu cầu

Bước đầu tiên của tiến trình phân tích hệ thống bao gồm việc xác định nhu cầu. Để bắt đầu, người phân tích giúp cho khách hàng trong việc xác định các mục tiêu của hệ thống (sản phẩm):

- thông tin nào cần phải tạo ra?
- thông tin nào cần được cung cấp?
- cần những chức năng và hiệu suất nào?

Người phân tích phải phân biệt được giữa "nhu cầu" của khác hàng (những tính năng chủ chốt cho sự thành công của hệ thống) và "điều mong muốn" của khác hàng (những tính năng tốt nên có nhưng không bản chất)

Một khi các mục tiêu tổng thể đã được xác định thì nhà phân tích chuyển sang việc đánh giá các thông tin phụ:

- Liệu có công nghệ để xây dựng hệ thống không?
- Cần có những tài nguyên chế tạo và phát triển đặc biệt nào?
- Cần phải đặt giới hạn nào về chi phí và lịch biểu?

Nếu hệ thống mới thực tế là một sản phẩm để bán cho nhiều khách hàng thì nên có những câu hỏi sau:

- Đâu là thị trường tiềm năng cho sản phẩm này?
- Sản phẩm này so với các sản phẩm cạnh tranh khác như thế nào?
- Sản phẩm này sẽ giữ vị trí nào trong tuyến sản phẩm của cả công ty?

Thông tin thu được trong bước xác định nhu cầu được kết tinh trong *Tài liệu khái quát về hệ thống*. Tài liệu khái quát nguyên bản đôi khi được khách hàng chuẩn bị trước cuộc gặp gỡ với người phân tích. Sự trao đổi thường xuyên giữa khách hàng - người phân tích tạo ra những thay đổi cho tài liệu này.

Bước 2: Nghiên cứu khả thi

Mọi dự án đều phải làm với sự hạn hẹp về tài nguyên và bảo đảm đúng ngày bàn giao. Cho nên cần phải thận trọng trong đánh giá tính khả thi của dự án từ thời điểm sớm nhất có thể được.

Phân tích khả thi và rủi ro có liên quan với nhau theo nhiều cách. Nếu rủi ro của dự án là lớn, thì tính khả thi của việc chế tạo phần mềm chất lượng sẽ bị giảm đi.

Trong kỹ nghệ hệ thống, chúng ta tập trung vào bốn lĩnh vực quan tâm chính:

1. Khả thi về kinh tế: đánh giá về chi phí phát triển cần phải cân xứng với lợi tức cuối cùng hay lợi ích mà hệ thống được xây dựng đem lại.
2. Khả thi về kỹ thuật: khảo cứu về chức năng, hiệu suất và ràng buộc có thể ảnh hưởng tới khả năng đạt tới một hệ thống chấp nhận được.
3. Khả thi về pháp lý: quy định của pháp luật về sự xâm phạm, vi phạm hay khó khăn nào có thể gây ra từ việc xây dựng hệ thống.
4. Khả thi về phương án: đánh giá tính khả thi của phương án đã xác định để tiếp cận tới việc xây dựng hệ thống.

Luận chứng kinh tế nói chung là xem xét "nền tảng" cho hầu hết các hệ thống (các ngoại lệ là hệ thống quốc phòng, hệ thống luật, các ứng dụng công nghệ cao như chương trình không gian vũ trụ)

Luận chứng kinh tế bao gồm:

- các mối quan tâm kể cả phân tích chi phí - lợi ích,
- chiến lược lợi tức dài hạn của công ty,
- ảnh hưởng tới các trung tâm hay sản phẩm lợi nhuận khác,
- chi phí cho tài nguyên cần cho việc xây dựng và phát triển thị trường tiềm năng

Khả thi kĩ thuật thường là lĩnh vực khó thâm nhập nhất tại giai đoạn này trong tiến trình phát triển hệ thống. Điều thực chất là tiến trình phân tích và xác định nhu cầu cần được tiến hành song song với việc xác định chúng.

Các xem xét thường được gắn với tính khả thi kĩ thuật bao gồm:

- Rủi ro xây dựng: Liệu các phần tử hệ thống có thể được thiết kế sao cho chức năng và hiệu suất cần thiết làm lộ rõ những ràng buộc trong khi phân tích không?
- Có sẵn tài nguyên: có sẵn các nhân viên cho việc xây dựng phần tử hệ thống đang xét không? Các tài nguyên cần thiết khác (phần cứng và phần mềm) có sẵn cho việc xây dựng hệ thống không?
- Công nghệ: công nghệ liên quan đã đạt tới trạng thái sẵn sàng hỗ trợ cho hệ thống chưa?

Người phát triển các hệ thống về bản chất đều lạc quan. Tuy nhiên, trong đánh giá tính khả thi kỹ thuật, việc đánh giá sai trong giai đoạn này cũng có thể là một thảm họa

Tính khả thi pháp lý bao gồm một phạm vi rộng các mối quan tâm kể cả hợp đồng, nghĩa vụ pháp lý, việc đánh giá sai trong giai đoạn này cũng có thể là một thảm họa.

Mức độ các phương án được xem xét tới thường bị giới hạn bởi ràng buộc chi phí và thời gian.

Có thể soạn tư liệu về nghiên cứu khả thi thành một báo cáo riêng cho cấp quản lý trên và đính kèm như phụ lục cho đặc tả hệ thống. Mặc dầu định dạng của báo cáo khả thi có thể thay đổi nhưng bản đại cương dưới đây đã bao quát được hầu hết những điểm quan trọng:

Bản đại cương về nghiên cứu khả thi

I. Giới thiệu	A. Phát biểu vấn đề

	B. Môi trường thực hiện C. Ràng buộc
II. Tóm tắt quản lý và khuyến cáo	A. Những tóm lược quan trọng B. Bình luận C. Khuyến cáo D. Tác động
III. Các phương án	A. Các cấu hình hệ thống theo phương án B. Các tiêu chuẩn được dùng trong chọn lựa cách tiếp cận cuối cùng
IV. Mô tả hệ thống	A. Phát biểu vắn tắt về phạm vi B. Tính khả thi của các phần tử trong hệ thống
V. Phân tích chi phí - lợi ích	
VI. Đánh giá rủi rõ kỹ thuật	
VII. Các chi nhánh pháp lý	
VIII. Các chủ đề khác chuyên cho dự án	
IX. Kết luận	

Bản nghiên cứu khả thi trước tiên được cấp quản lý dự án xem xét (để đánh giá độ tin cậy nội dung) rồi đến cấp quản lý cao hơn (để đánh giá trạng thái dự án). Bản nghiên cứu phải đề xuất quyết định "tiến hành/không tiến hành". Cũng cần chú ý rằng các quyết định tiến hành hay không tiến hành sẽ được đưa ra trong các bước lập kế hoạch, đặc tả và xây dựng cho cả công nghệ phần mềm và phân cứng.

Bước 3: Mô hình hóa hệ thống -modeling

Việc mô hình hóa và mô phỏng hệ thống được sử dụng để loại bỏ những điều bất ngờ khi xây dựng hệ thống. Những công cụ CASE được áp dụng trong các tiến trình kỹ nghệ hệ thống. Vai trò của công cụ mô hình hóa và mô phỏng được tóm tắt như sau:

- Cung cấp một phương án cho tiến trình thiết kế, cài đặt và kiểm thử; thông qua việc tiếp cận các công cụ mô hình hoá (một công cụ mô hình hóa và mô phỏng)
- Cho phép xây dựng một mô hình đề cập tới tất cả các vấn đề luồng chức năng và dữ liệu thông thường đồng thời còn bao quát cả các khía cạnh hành động, hành vi của hệ thống. Có thể kiểm thử mô hình này bằng các công cụ phục vụ giám định và gỡ lỗi cho đặc tả và tìm kiếm thông tin.
- Dùng để "kiểm thử sự hoạt động" của đặc tả hệ thống: bằng cách kiểm thử mô hình (đặc tả), người kỹ sư hệ thống có thể hình dung được cách thức mà hệ thống sẽ hành xử ra sao khi được cài đặt. Người ta có thể sử dụng câu hỏi cái gì xẩy ra nếu đi theo các kịch bản; xác định, kiểm tra rằng những tình huống mong muốn nào đó có xuất hiện hay không, và các tình huống không mong muốn khác sẽ không xuất hiện hay lại xuất hiện.

6.1.2 Đặc tả hệ thống

Bản Đặc tả hệ thống:

- là một tài liệu làm nền tảng cho kỹ nghệ phần cứng, kỹ nghệ phần mềm, kỹ nghệ cơ sở dữ liệu và kỹ nghệ con người.
- mô tả về chức năng và hiệu suất của hệ thống và những ràng buộc hệ thống.
- quy định cả giới hạn cho từng phần tử hệ thống. Chẳng hạn, chỉ dẫn về vai trò của phần mềm trong hệ thống và các hệ thống con người khác nhau được mô tả trong biểu đồ luồng kiến trúc
- mô tả thông tin (dữ liệu và điều khiển) vào/ ra khỏi hệ thống

Dàn bài về bản đặc tả hệ thống sẽ được trình bày sau đây. Tuy nhiên cần lưu ý rằng đây chỉ là một trong nhiều dàn bài có thể được dùng để định nghĩa một tài liệu mô tả hệ thống. Định dạng và nội dung thực tế có thể còn tùy vào các chuẩn kỹ nghệ hệ thống hay phần mềm. Nó được điều chỉnh theo yêu cầu và tính ưa chuộng của người dùng.

Dàn bài đặc tả hệ thống

I. Giới thiệu	A. Phạm vi và mục tiêu của dự án B. Tổng quan 1. Mục tiêu 2. Ràng buộc
II. Mô tả chức năng và dữ liệu	A. Kiến trúc hệ thống 1. Biểu đồ ngữ cảnh kiến trúc 2. Mô tả về biểu đồ ngữ cảnh hệ thống
III. Mô tả các hệ thống con	A. Đặc tả biểu đồ kiến trúc cho hệ con **n** 1. Biểu đồ luồng kiến trúc 2. Chú giải modul hệ thống 3. Vấn đề về hiệu suất 4. Ràng buộc thiết kế 5. Cách tạo các thành phần hệ thống B. Từ điển kiến trúc C. Biểu đồ và mô tả liên nối hệ thống
IV. Các kết quả mô hình hóa và mô phỏng hệ thống	A. Mô hình hệ thống được dùng cho mô phỏng B. Kết quả mô phỏng C. Vấn đề hiệu suất đặc biệt
V. Vấn đề dự án	A. Chi phí xây dựng dự phòng B. Lịch biểu dự phòng
VI. Phụ lục	

Xét duyệt đặc tả hệ thống

Cuộc họp xét duyệt đặc tả hệ thống đánh giá tính đúng đắn của định nghĩa chứa trong bản đặc tả hệ thống:

Cuộc họp được cả người phát triển và khách hàng đảm bảo rằng:

1. Phạm vi của dự án đã được vạch ra đúng
2. Các chức năng, hiệu suất và giao diện đã được định nghĩa đúng

3. Phân tích rủi ro môi trường và tính khả triển của dự án
4. Người phát triển và khách hàng có cùng cảm nhận về mục tiêu hệ thống

Cuộc họp đặc tả hệ thống được tiến hành trong giai đoạn

- đưa ra những quan điểm quản lý áp dụng cho hệ thống
- tiến hành đánh giá kỹ thuật về các phần tử và chức năng hệ thống

Về khía cạnh quản lý, đặc tả hệ thống cần phải thỏa mãn những câu hỏi sau:

- Nhu cầu kinh doanh của hãng đã được thiết lập chưa? Luận chứng hệ thống có nghĩa không?
- Có cần môi trường (hay thị trường) riêng cho hệ thống đã được mô tả hay không?
- Những phương án nào đã được xem xét?
- Rủi ro phát triển cho từng phần tử hệ thống là gì?
- Các tài nguyên đã có sẵn cho việc xây dựng hệ thống chưa?
- Các giới hạn chi phí và lịch biểu có ý nghĩa gì không?

Các câu hỏi trên nên được đặt ra và trả lời một cách đều đặn trong nhiệm vụ phân tích. Một câu hỏi nên được xem xét lại tại giai đoạn đánh giá kỹ thuật.

Mức độ chi tiết trong giai đoạn đánh giá kỹ thuật (họp xét duyệt hệ thống) phụ thuộc vào mức độ chi tiết trong nhiệm vụ xác định ban đầu.

Việc xét duyệt nên bao gồm các vấn đề sau:

- Về chức năng của hệ thống: có phù hợp với những định giá về rủi ro phát triển, chi phí và lịch biểu hay không?
- Các chức năng được xác định đã đủ chi tiết chưa?
- Giao diện giữa các phần tử hệ thống và với môi trường đã được xác định đủ chi tiết chưa?
- Các vấn đề độ tin cậy, hiệu suất và bảo trì đã được đề cập tới trong bản đặc tả chưa?
- Liệu bản đặc tả hệ thống có cung cấp đủ nền tảng cho các bước kỹ nghệ phần cứng và phần mềm tiếp sau không?

Kết luận:

Sau cuộc họp xét duyệt, tiến hành các tiến trình kỹ nghệ tương ứng với các phần tử hệ thống chủ chốt như phần mềm, phần cứng, con người và CSDL. Các phần tử phần cứng, con người và CSDL của hệ thốn được đề cập tới như phần tương ứng của các tiến trình kỹ nghệ

Tại bước phân tích hệ thống người phân tích xác định ra nhu cầu của khách hàng, xác định tính khả thi kinh tế - kỹ thuật, xác định chức năng và hiệu suất cho các phần tử hệ thống chủ chốt (phần mềm, phần cứng, con người và CSDL)

Bản đặc tả hệ thống (tài liệu nền tảng cho toàn bộ công việc kỹ nghệ tiếp sau đó) được coi là đỉnh cao của nhiệm vụ kỹ nghệ hệ thống

Phải đảm bảo việc trao đổi liên lạc đều đặn giữa khách hàng và nhà phân tích vì nếu trao đổi giữa nhà phân tích và khách hàng bị gián đoạn tại giai đoạn này thì sự thành công của toàn bộ dự án sẽ bị đe dọa

Khó khăn là ở chỗ phải lập các tư liệu hệ thống sao cho:

- Dễ hiểu cho người sử dụng
- Tạo ra bản đặc tả hệ thống (cùng với đặc tả yêu cầu, được dùng làm cơ sở cho một hợp đồng giữa người mua và người cung cấp phần mềm đó). Nói chung, người dùng ưa thích một môt tả hệ thống trừu tượng chứ không thích một đặc tả chi tiết.

6.2 Tạo nguyên mẫu (prototype)

Duyệt lại là một phần của quá trình thẩm định yêu cầu. Từ đặc tả yêu cầu ta có thể tưởng tượng hệ thống sẽ được sử dụng như thế nào. Một chức năng được mô tả trong một đặc tả là hữu ích và đã được xác định tốt nhưng thực tế việc sử dụng chức năng đó kết hợp với các chức năng khác lại để lộ ra rằng cái nhìn ban đầu của người sử dụng đã không đúng và không đầy đủ.

Một cách giải quyết khó khăn đó là phát triển một nguyên mẫu hệ thống cho phép người sử dụng thí nghiệm trên nguyên mẫu đó. Các yêu cầu của hệ thống được thẩm định và người dùng sẽ phát hiện ra các sai sót.

Sự khác biệt giữa nguyên mẫu phần mềm với nguyên mẫu phần cứng:

Nguyên mẫu phần mềm hoàn toàn khác với nguyên mẫu phần cứng. Khi phát triển các hệ thống phần cứng, thì thực tế người ta phát triển một nguyên mẫu hệ thống để thẩm định thiết kế hệ thống. Một nguyên mẫu hệ thống điện tử có thể được thực hiện và được thử nghiệm bằng cách dùng các thành phần chưa được ráp vao vỏ trước khi đầu tư vào các mạch tích hợp chuyên dụng đắt tiền để thực hiện một đời sản phẩm mới của hệ thống. Một nguyên mẫu phần mềm là không phải nhằm vào việc thẩm định thiết kế (thiết kế của nó thường là hoàn toàn khác với hệ thống đã phát triển cuối cùng), mà là để thẩm định yêu cầu của người sử dụng phần mềm.

6.2.1 Lợi ích của việc phát triển nguyên mẫu

Việc phát triển nguyên mẫu ngay từ sớm trong quá trình phát triển phần mềm là:

1. Sự hiểu lầm giữa những người phát triển phần mềm và những người sử dụng phần mềm có thể được nhận thấy rõ khi các chức năng của hệ thống được trình diễn
2. Sự thiếu hụt các dịch vụ người dùng có thể được phát hiện
3. Sự khó sử dụng hoặc sự nhầm lẫn các dịch vụ người dùng có thể được thấy rõ và được sửa sang lại
4. Đội ngũ phát triển phần mềm có thể tìm ra được các yêu cầu không đầy đủ hoặc không kiên định khi họ phát triển nguyên mẫu.
5. Một hệ thống hoạt động được (mặc dầu là hạn chế) là bằng chứng thuyết minh cho tính khả thi và tính hữu ích của ứng dụng đó cho các nhà quản lý
6. Nguyên mẫu đó được dùng làm cơ sở cho việc viết đặc tả cho sản phẩm

Mặc dù mục tiêu chủ yếu của việc tạo nguyên mẫu là thẩm định các yêu cầu phần mềm cũng nên chỉ ra rằng một nguyên mẫu phần mềm cũng có các ứng dụng khác:

1. Dùng để huấn luyện người sử dụng ngay từ trước khi hệ thống được phân phối
2. Dùng trong quá trình thử nghiệm hệ thống. Điều đó nghĩa là cùng các trường hợp thử như nhau vừa dùng cho thử nguyên mẫu vừa cho thử hệ thống. Kết quả khác nhau có nghĩa là có sai sót.

Tạo nguyên mẫu là một kỹ thuật giảm bớt rủi ro. Một rủi ro lớn trong việc phát triển phần mềm là các sai sót mà đến giai đoạn cuối mới phát hiện và chỉnh sửa là rất tốn kém. Kinh

nghiệm cho thấy rằng việc tạo nguyên mẫu sẽ giảm bớt số các vấn đề của đặc tả yêu cầu và tổng chi phí của việc phát triển có thể là thấp hơn nếu ta phát triển nguyên mẫu.

6.2.2 Các giai đoạn trong việc phát triển nguyên mẫu

1. Thiết lập các mục tiêu của việc tạo nguyên mẫu, chọn input, output cho nguyên mẫu.
2. Chọn các chức năng cho việc tạo nguyên mẫu và quyết định những đặc tả phi chức năng nào cần có trong nguyên mẫu
3. Phát triển nguyên mẫu
4. Đánh giá hệ nguyên mẫu

Cần phải làm rõ ràng các mục tiêu tạo nguyên mẫu trước khi bắt đầu công việc. Mục tiêu đó có thể là phát triển một hệ thống liên quan chủ yếu đến tạo nguyên mẫu giao diện người dùng; nó cũng có thể là việc phát triển một hệ thống thẩm định các yêu cầu hệ thống chức năng, nó cũng có thể là phát triển một hệ thống để trình diễn tính khả thi của ứng dụng cho các nhà quản lý xem xét, ... CÙng một nguyên mẫu không thể đạt được tất cả các mục tiêu. Nếu mục tiêu còn chưa rõ ràng thì người quản lý hoặc người sử dụng có thể hiểu nhầm chức năng của nguyên mẫu và không đạt được lợi ích thiết thực đầy đủ của việc tạo nguyên mẫu.

Giai đoạn tiếp theo trong quá trình này là quyết định xem cái gì được đưa vào và cái gì được lấy ra từ hệ nguyên mẫu đó. Nguyên mẫu của phần mềm là đắt nến nó được thực hiện bằng cách dùng cùng các công cụ và các chuẩn y như cho chính hệ thống cuối cùng

Bởi vậy, có thể quyết định tạo nguyên mẫu cho tất cả các chức năng của hệ thống nhưng ở mức thu gọn. Cũng có thể chỉ một số chức năng hệ thống được tạo mẫu. Bình thường trong thực tế người ta hay bỏ bớt các yêu cầu phi chức năng chẳng hạn như yêu cầu về tốc độ, về không gian nhớ. Việc sử lý sai sót và việc quản trị có thể được bỏ qua và có thể là thừa khi mục tiêu của việc tạo mẫu là thiết lập một giao diện người máy. Các chuẩn của độ tin cậy và của chất lượng chương trình cũng có thể chưa tính đến.

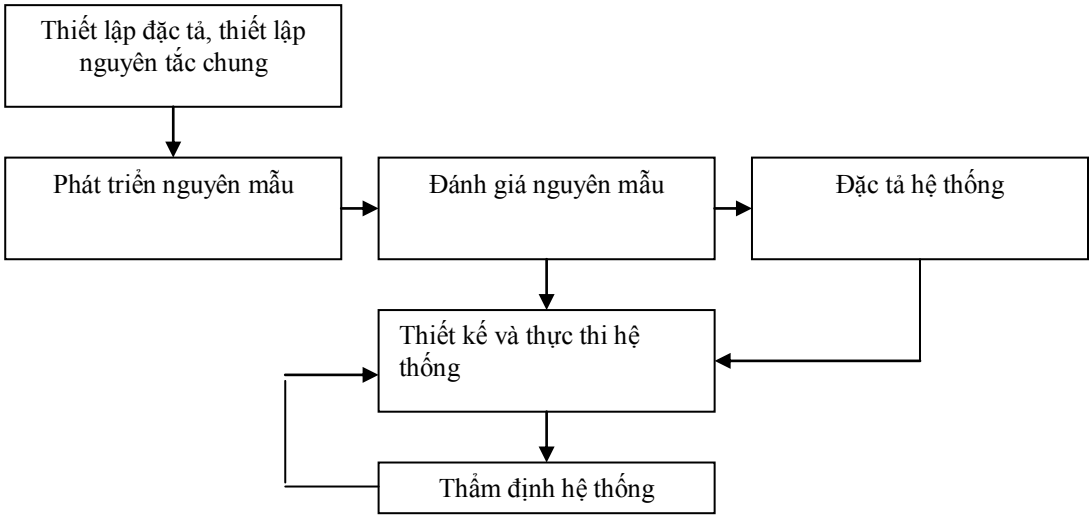
Giai đoạn cuối cùng của quá trình này là đánh giá nguyên mẫu. Có người cho rằng đây là giai đoạn quan trọng nhất của việc tạo nguyên mẫu. Phải dự tính cho việc huấn luyện người sử dụng trong giai đoạn này và các mục tiêu của việc tạo mẫu hẳn là nên dẫn ra một kế hoạch đánh giá. Phải có đủ thời gian cho người sử dụng trở nên dễ chịu với hệ thống và thiết lập mẫu sử dụng chuẩn tắc và bởi vậy phát hiện được các sai sót yêu cầu

Việc tạo nguyên mẫu là kỹ thuật chính trong mô hình quá trình xoắn ốc hỗ trợ định giá rủi ro.

6.2.3 Tạo nguyên mẫu trong tiến trình phần mềm

Vấn đề cơ bản là khó đánh giá phần mềm mới được xây dựng có ảnh hưởng thế nào tới công việc của người dùng. Đối với các hệ thống mới, đặc biệt là đối với các hệ thống lớn và phức tạp thì có lẽ là không thể đánh giá được trước khi hệ thống được xây dựng xong và đưa vào ứng dụng.

Một cách để vượt qua khó khăn là sử dụng cách tiếp cận thăm dò để phát triển hệ thống. Điều này có nghĩa là trình bày cho người dùng một hệ thống chưa đầy đủ và rồi cải biên nó tăng cường hệ thống đó khi mà các yêu cầu thực của người dùng trở nên trong suốt. Sau khi đánh giá nguyên mẫu đó sẽ bị loại và một hệ chất lượng tốt hơn sẽ được xây dựng.



H2.5 *Việc tạo nguyên mẫu tromg quá trình phần mềm*

Sự khác biệt của hai cách tiếp cận:

1. Lập trình thăm dò: Bắt đầu với một hiểu biết mơ hồ về yêu cầu hệ thống và hệ thống sẽ được tăng cường một khi các yêu cầu mới được phát hiện. Chẳng có gì giống như một đặc tả hệ thống và sự thật hệ thống được phát triển theo cách tiếp cận này thì không thể đặc tả được (ví dụ: một vài hệ trí tuệ nhân tạo là không thể đặc tả được)

2. Cách tiếp cận tạo nguyên mẫu:
   - nhằm phát hiện ra đặc tả hệ thống và kết quả của giai đoạn phát triển nguyên mẫu là bản đặc tả đó
   - mở rộng quá trình phân tích yêu cầu nhằm rút bớt tổng chi phí cho toàn bộ vòng đời phần mềm
   - làm sáng tỏ các yêu cầu
   - cung cấp các thông tin phụ cho người quản lý để họ đánh giá các rủi ro của dự án

Sau khi đánh giá, nguyên mẫu đó không dùng để phát triển hệ thống nữa.

Thay thế cho việc dẫn xuất ra đặc tả từ nguyên mẫu, đôi khi người ta đề xuất là đặc tả hệ thống chính là sự thực hiện nguyên mẫu đó. Chỉ dẫn này giúp người ký hợp đồng chỉ đơn giản là hãy viết một hệ thống kiểu như thế này.

6.2.4 Hạn chế của cách tiếp cận tạo nguyên mẫu

1. Các đặc điểm hệ thống quan trọng có thể nằm ngoài nguyên mẫu để đơn giản hóa việc thực hiện nhanh. Sự thực là không thể tạo nguyên mẫu cho một vài phần quan trong nhất của hệ thống như các đặc điểm về sự an toàn/ nguy kịch.
2. Các yêu cầu phi chức năng như các yêu cầu liên quan đến độ tin cậy, tính mâu thuẫn và độ an toàn là không thể biểu thị đầy đủ trong khi thực hiện nguyên mẫu.
3. Người dùng có thể không dùng nguyên mẫu y như cách dùng một hệ đang hoạt động

Có một vài lý do dẫn tới việc tái tạo nguyên mẫu khi một hệ thống lớn và tuổi thọ cao được phát triển

1. Các đặc trưng hệ thống quan trọng chẳng hạn như sự thực thi, an ninh, tính không mâu thuẫn, độ tin cậy có thể được lờ đi khi tạo nguyên mẫu để cho có thể thực thi nhanh

nguyên mẫu. Bản chất của nguyên mẫu có thể lại là những đặc trưng đó không thể thêm vào nguyên mẫu.

2. Trong quá trình phát triển nguyên mẫu thì nguyên mẫu sẽ bị thay đổi để phản ánh các nhu cầu của người dùng và dường như các thay đổi đó sẽ được tiến hành theo một cách không thể không chế được.
3. Các thay đổi trong quá trình phát triển nguyên mẫu sẽ có thể làm giảm sút cấu trúc hệ thống đến mức mà do yêu cầu bảo trì nên các thay đổi tiếp theo càng ngày càng trở nên khó khăn hơn

Phát triển nguyên mẫu là dễ quản lý hơn lập trình thăm do vì các chuẩn quá trình phần mềm là được tuân theo. Các kế hoạch và các tư liệu có thể viết thêm cho mỗi phần thêm vào.

6.2.5 Các bước tiến hành làm nguyên mẫu phần mềm

Bước 1: Đánh giá yêu cầu phần mềm và xác định liệu phần mềm cần xây dựng có xứng đáng để làm nguyên mẫu không

Không phải tất cả các phần mềm đều có thể đưa tới làm nguyên mẫu. Ta có thể xác định một nhân tố làm nguyên mẫu: lĩnh vực ứng dụng, độ phức tạp ứng dụng, đặc trưng khách hàng và đặc trưng dự án.

Nói chung, bất kỳ ứng dụng nào tạo ra việc hiển thị trực quan động, tương tác nhiều với người, hay yêu cầu các thuật toán hay việc xử lý tổ hợp mà cần phải được xây dựng theo một cách tiến hóa thì đều có thể làm nguyên mẫu. Tuy nhiên, những lĩnh vực ứng dụng này phải được cân nhắc dựa trên độ phức tạp của ứng dụng. Nếu một ứng dụng yêu cầu xây dựng tới 10 ngàn dòng mã lệnh thì phần chắc là nó quá phức tạp không thể làm nguyên mẫu được. Tuy nhiên nếu ta có thể phân hoạch độ phức tạp thì vẫn có thể làm nguyên mẫu cho từng phần của phần mềm.

Để đảm bảo tính tương tác giữa khách hàng với nguyên mẫu cần:

1. Khách hàng phải cam kết dùng tài nguyên để đánh giá và làm mịn nguyên mẫu
2. Khách hàng phải có khả năng đưa ra những quyết định về yêu cầu một cách kịp thời.

Bản chất của dự án quyết định tính hiệu quả của làm nguyên mẫu.

Bước 2: Với một dự án chấp thuận được người phân tích xây dựng một biểu diễn vắn tắt các yêu cầu

Trước khi có thể bắt đầu xây dựng một nguyên mẫu, người phân tích phải biểu diễn miền thông tin và các lĩnh vực hành vi và chức năng của vấn đề rồi xây dựng một cách tiếp cận hợp lý tới việc phân hoạch. Có thể ứng dụng các nguyên lý phân tích nền tảng (trên xuống) và các phương pháp phân tích yêu cầu.

Bước 3: Sau khi đã duyệt mô hình và yêu cầu, tạo ra một đặc tả thiết kế vắn tắt cho nguyên mẫu

Việc thiết kế phải xuất hiện trước khi bắt đầu làm nguyên mẫu. Tuy nhiên phải thiết kế tập trung chủ yếu vào các vấn đề thiết kế dữ liệu và kiến trúc mức đỉnh chứ không phải tập trung vào thiết kế thủ tục chi tiết.

Bước 4: Phần mềm nguyên mẫu được tạo ra, kiểm thử và làm mịn

Một cách lý tưởng, các khối xây dựng phần mềm hiện có được dùng để tạo ra nguyên mẫu một cách nhanh chóng.

Bước 5: Một khi đã kiểm thử song nguyên mẫu thì có thể trình bày nó cho khách hàng

Khách hàng sẽ kiểm thử ứng dụng và gợi ý những thay đổi. Bước này cốt lõi của cách tiếp cận làm nguyên mẫu. Chính ở đây mà khách hàng có thể xem xét cách biểu diễn được cài đặt

cho yêu cầu phần mềm, gợi ý những thay đổi làm cho phần mềm đáp ứng tốt hơn với các nhu cầu thực tế.

Bước 6: Lặp lại các bước 4 và 5 cho tới khi tất cả các yêu cầu đã được hình thức hóa hay cho tới khi nguyên mẫu đã tiến hóa thành một hệ thống sản phẩm cuối cùng.

Khuôn mẫu làm nguyên mẫu có thể được tiến hành với một trong 2 mục tiêu:

1. Mục tiêu của việc làm nguyên mẫu là thiết lập một tập hợp các yêu cầu hình thức có thể được dịch thành phần mềm (sản xuất bằng các phương pháp và kỹ thuật của kỹ nghệ phần mềm)
2. Mục tiêu của việc làm nguyên mẫu là cung cấp động lực liên tục thúc đẩy phát triển tiến hóa phần mềm.

6.2.6 Các phương pháp và công cụ làm nguyên mẫu

Để việc làm nguyên mẫu phần mềm được hiệu quả, phải xây dựng nhanh chóng nguyên mẫu sao cho khách hàng có thể định giá được kết quả và khuyến cáo những thay đổi.

Để xây dựng nguyên mẫu nhanh, hiện có 3 lớp phương pháp và công cụ sẵn có:

1. Các kỹ thuật thế hệ 4
2. Thành phần phần mềm dùng lại
3. Đặc tả hình thức và môi trường làm nguyên mẫu

Kỹ thuật thế hệ 4

Các kỹ thuật thế hệ 4 (4GT) bao gồm một phạm vi rộng các ngôn ngữ hỏi CSDL và làm báo cáo, các bộ sinh chương trình và ứng dụng các ngôn ngữ phi thủ tục cấp rất cao khác. Bởi vì 4GT sinh được mã thực hiện rất nhanh nên chúng là lý tưởng cho việc làm nguyên mẫu nhanh. Mặc dầu lĩnh vực ứng dụng 4GT hiện tại vẫn còn bị giới hạn vào các hệ thông tin kinh doanh, những công cụ cho các ứng dụng kỹ nghệ đang bắt đầu chiếm ưu thế.

Thành phần phần mềm dùng lại

Một cách tiếp cận khác với việc làm nguyên mẫu nhanh là lắp ráp, thay vì xây dựng nguyên mẫu bằng cách dùng một tập hợp các thành phần phần mềm đã có sẵn. Một thành phần phần mềm có thể là một cấu trúc dữ liệu (hay CSDL) hay một thành phần kiến trúc phần mềm (như chương trình) hay một thành phần thủ tục (như một modul). Trong mỗi trường hợp thành phần phần mềm phải được thiết kế theo cách thức làm cho nó có thể dùng lại được mà không cần biết chi tiết về cách làm việc bên trong.

Việc làm nguyên mẫu với việc dùng lại các thành phần chương trình chỉ hoạt động được nếu hệ thóng thư viện đã được phát triển sao cho các thành phần thực tồn tại có thể được đưa vào danh mục rồi tìm kiếm. Mặc dầu người ta đã tạo ra một số công cụ để đáp ứng nhu cầu này, vẫn còn nhiều việc cần phải làm thêm.

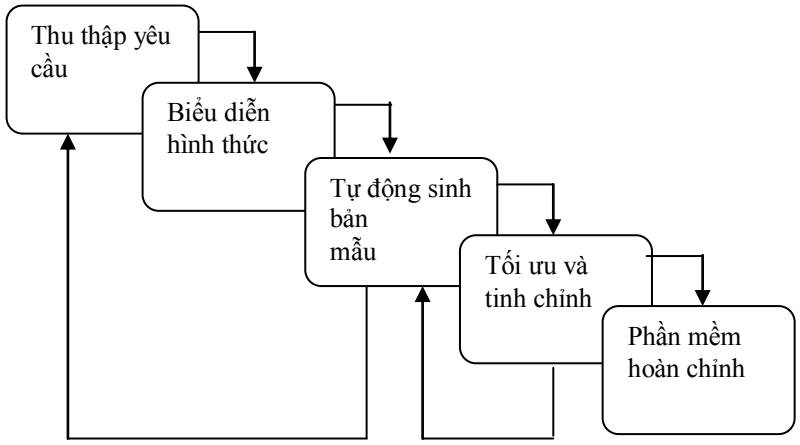
Cũng nên lưu ý rằng một sản phẩm phần mềm hiện có, có thể được dùng như nguyên mẫu cho một sản phẩm "mới", hay “được cài tiến". Mặt khác đây cũng là một hình thức của việc dùng lại cho việc làm nguyên mẫu phần mềm.

Đặc tả hình thức và môi trường làm nguyên mẫu

Một số ngôn ngữ và công cụ đặc tả hình thức cũng đã được xây dựng xem như một sự thay thế cho các kỹ thuật đặc tả theo ngôn ngữ tự nhiên. Các ngôn ngữ này đang trong tiến trình phát triển các môi trường tương tác.

1. Làm cho người phân tích tạo ra cách tương tác với một đặc tả về hệ thống hay phần mềm dựa trên ngôn ngữ

2. Gọi tự động các công cụ dịch đặc tả dựa trên ngôn ngữ này thành mã thực hiện được
3. Làm cho khách hàng có thể dùng nguyên mẫu chương trình theo mã thực hiện được để làm mịn các yêu cầu hình thức. Các ngôn ngữ đặc tả như RAISE, PSL, RSL, IORL, GYPSY, OBJ và nhiều ngôn ngữ khác đang đi kèm với các môi trường tương tác. Mặc dầu vẫn còn đang trong giai đoạn sơ khai của phát triển ứng dụng, những môi trường như vậy thường đưa ra hy vọng chủ yếu cho việc làm nguyên mẫu nâng cao và tính hiệu quả của việc phát triển phần mềm.



H2.6 *Một khuôn cảnh kỹ nghệ phần mềm tự động hoá*

Tóm tắt

Phân tích và định rõ yêu cầu là kỹ thuật đầu tiên trong tiến trình kỹ nghệ phần mềm. Chính tại điểm này mà phát triển chung về phạm vi phần mềm được làm mịn thành một bản đặc tả cụ thể để trở thành nền tảng cho mọi hoạt động kỹ nghệ phần mềm theo sau.

Việc phân tích phải tập trung vào các miền thông tin, chức năng và hành vi của vấn đề. Để hiểu rõ hơn cần cái gì, cần tạo ra mô hình, phân hoạch vấn đề và tạo ra những biểu diễn mô tả cho bản chất của yêu cầu rồi sau đó xây dựng các chi tiết cài đặt.

Đặc tả yêu cầu phần mềm được xây dựng như kết quả của việc phân tích. Việc xét duyệt là bản chất để đảm bảo rằng người phát triển và khách hàng có cùng nhận thức về hệ thống.

Trong nhiều trường hợp, không thể nào đặc tả được đầy đủ một vấn đề tại giai đoạn đầu. Việc làm nguyên mẫu thường giúp chỉ ra cách tiếp cận khác để từ đó có thể làm mịn theo yêu cầu. Để tiến hành đúng đắn việc làm nguyên mẫu, có thể cần tới các công cụ và kỹ thuật đặc biệt.

Củng cố

1. Mô tả bằng sơ đồ việc hình thành các yêu cầu trong tiến trình phân tích và định rõ yêu cầu?
2. Phân tích các nguyên lý đặc tả?
3. Khái niệm về đặc tả, biểu diễn và mức trừu tượng của đặc tả?
4. Tại sao chúng ta phải coi việc xác định nhu cầu và phân tích khả thi là những bước đầu tiên của tiến trình phân tích hệ thống?
5. Đặc tả yêu cầu bao gồm những hoạt động nào, hãy mô tả tóm tắt những hoạt động đó?

6. Tại sao nói đặc tả yêu cầu là cơ sở cho hợp đồng làm hệ thống?
7. Đặc tả hệ thống được phân biệt với đặc tả yêu cầu ở điểm nào?
8. Dàn bài đặc tả hệ thống tối thiểu cần bao quát những nội dung nào?
9. Nội dung và ý nghĩa của việc xét duyệt đặc tả hệ thống?
10. Mục tiêu và lợi ích chủ yếu của việc tạo nguyên mẫu?
11. Trình bày sơ lược nội dung của bốn giai đoạn phát triển nguyên mẫu?
12. Sơ lược về các bước/ phương pháp và công cụ làm nguyên mẫu?

CHƯƠNG 3

THIẾT KẾ PHẦN MỀM

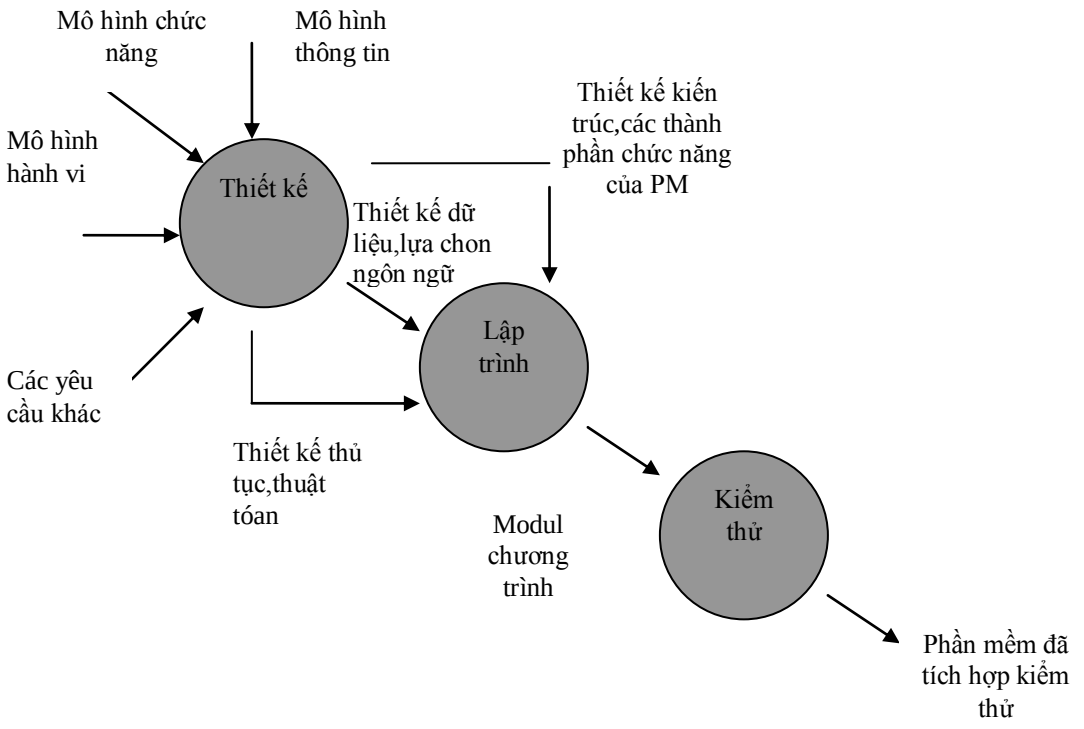
Thiết kế phần mềm là một tiến trình chuyển hóa các yêu cầu thành một biểu diễn phần mềm. Việc làm mịn tiếp sau dẫn tới một biểu diễn thiết kế rất gần với chương trình gốc.

I.Khái niệm về Thiết kế phần mềm

1.1 Thiết kế phần mềm trong kỹ nghệ phần mềm

Sau khi đã phân tích và đặc tả các yêu cầu phần mềm thì sẽ tiến hành ***thiết kế phần mềm*** **,** là giai đoạn đầu trong 3 hoạt động kỹ thuật: ***thiết kế - lập trình - kiểm thử*** phần mềm.

Có thể hình dung tiến trình kỹ nghệ phần mềm như sau:



H3.1 *Thiết kế phần mềm và kỹ nghệ phần mềm*

Các yêu cầu phần mềm được biểu thị bằng các mô hình thông tin, chức năng và hành vi, là cái vào cho bước thiết kế. Bằng việc sử dụng một trong số các phương pháp thiết kế, bước thiết kế tạo ra thiết kế dữ liệu, thiết kế kiến trúc và thiết kế thủ tục

- *Thiết kế cấu trúc dữ liệu,lựa chọn ngôn ngữ : dựa trên các* mô hình đã được tạo ra trong bước phân tích yêu cầu, sẽ mô tả,thiết kế các cấu trúc dữ liệu cho phần mềm, cấu trúc dữ liệu của từng bài tóan cụ thể còn phụ thuộc vào việc lựa chọn ngôn ngữ lập trình và cài đặt chúng như thế nào.
- *Thiết kế kiến trúc chương trình* : xác định các thành phần cấu trúc chức năng của chương trình và mối quan hệ giữa chúng.
- *Thiết kế thủ tục*: xây dựng các thuật toán tính tóan và thuật tóan xử lý số liệu

Thiết kế phần mềm rất quan trọng .Phải có Thiết kế phần mềm tốt mới có sản phẩm phần mềm có chất lượng và bảo đảm bảo trì phần mềm về sau.

1.2 Các giai đoạn trong thiết kế phần mềm

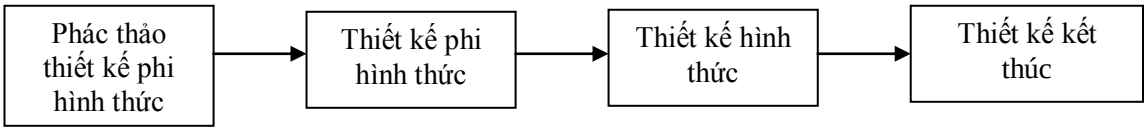
Thiết kế phần mềm trải qua 3 giai đoạn sau:

1. Nghiên cứu và tìm hiểu các kết quả phân tích và đặc tả yêu cầu phần mềm ở công đọan trước
2. Lựa chọn các giải pháp thiết kế. Chọn giải pháp phụ thuộc vào kinh nghiệm của người thiết kế
3. Triển khai công việc thiết kế . Đây là bước đòi hỏi khả năng tư duy trừu tượng,sáng tạo của người thiết kế. Các sơ đồ mô hình thường phải xây dựng một mô tả ban đầu sơ khai rồi chi tiết hóa nó, đi từ thiết kế mức đỉnh đến các mức chi tiết hơn. Chỉ sau khi đã hoàn thiện được thiết kế mới lập tư liệu thiết kế.

1.3 Quá trình thiết kế

I.3.1 Các hoạt động thiết kế

Quá trình thiết kế là quá trình tăng cường hình thức hoá trong sự tiến triển của thiết kế và phải luôn quay trở lại các thiết kế đúng đắn ít hình thức có trước đây của quá trình đó. Nhà thiết kế phải bắt đầu với một phác thảo hết sức không hình thức rồi sau đó tinh chế nó, thêm các thông tin để thiết kế trở nên hình thức hơn. Quá trình thiết kế thể hiện như sau:

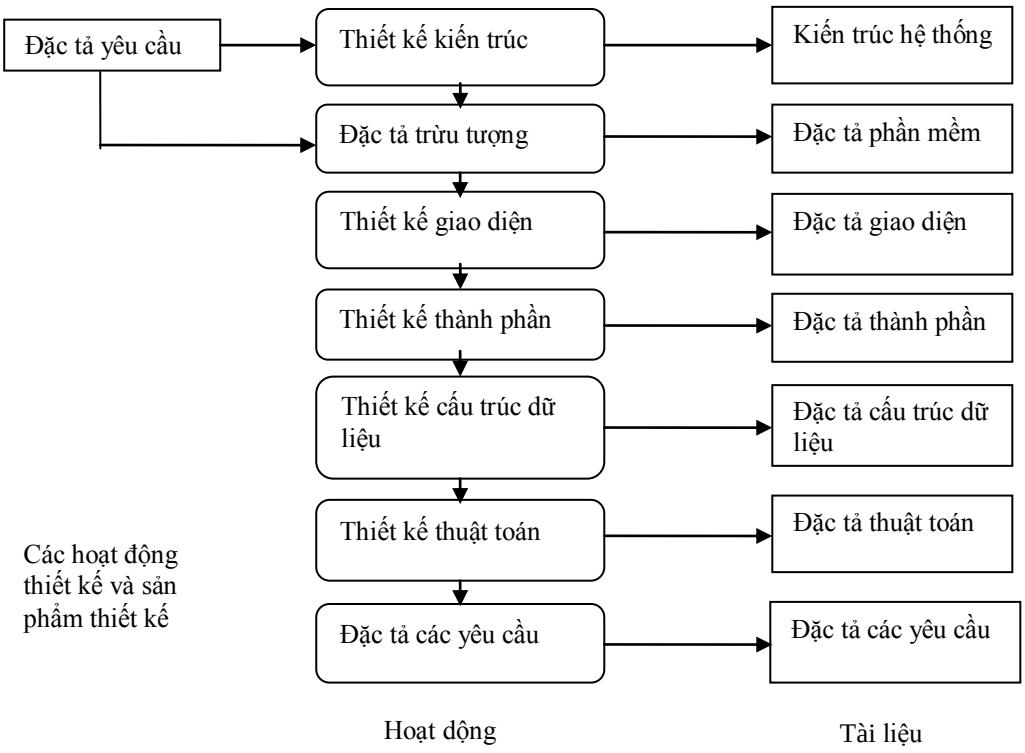


Quan hệ giữa thiết kế và đặc tả là rất chặt chẽ. Mặc dầu quá trình đưa ra một đặc tả yêu cầu được xem như một phần tử cơ bản của hợp đồng là một hoạt động riêng biệt song việc hình thức hoá đặc tả yêu cầu hẳn là một phần của quá trình thiết kế. Thực tế, người làm thiết kế sẽ lặp đi lặp lại giữa đặc tả và thiết kế.

Quá trình thiết kế liên quan mật thiết đến việc mô tả hệ thống ở một số mức trừu tượng khác nhau. Khi một thiết kế được phân chia thành nhiều thành phần thì người ta thường phát hiện được các sai sót của các giai đoạn trước, do đó phải quay trở lại các giai đoạn trước để

tinh chế lại. Thông thường là người ta bắt đầu giai đoạn này ngay trước khi giai đoạn trước kết thúc, đơn giản là để lui lại quá trình tinh chế.

Hình vẽ dưới đây nêu các hoạt động của quá trình thiết kế và các thành phẩm của nó. Các giai đoạn là khá tuỳ ý, nhưng nó làm cho quá trình thiết kể trở nên nhìn thấy được và do đó quản lý được.



Thành quả của mỗi hoạt động thiết kế là một đặc tả. Đặc tả này có thể là một đặc tả trừu tượng, hình thức và được tạo ra để làm rõ các yêu cầu, nó cũng có thể là một đặc tả về một phần nào đó của hệ thống, sẽ phải được thực hiện như thế nào. Khi quá trình thiết kế tiên triển thì các chi tiết ngày càng được bổ xung vào đặc tả đó. Các kết quả cuối cùng là các đặc tả về các thuật toán và các cấu trúc dữ liệu được dùng làm cơ sở cho việc thực hiện hệ thống.

Thực tế các hoạt động thiết kế diễn ra song song với các sản phẩm thiết kế khác nhau. Các sản phẩm này đã được triển khai ở các mức chi tiết khác nhau trong diễn biến của quá trình thiết kế,

Các hoạt động rất cốt yếu trong việc thiết kế một hệ thống phần mềm lớn:

- *Thiết kế kiến trúc*: các hệ con tạo nên hệ tổng thể và các quan hệ của chúng là được minh định và ghi thành tài liệu
- *Đặc tả trừu tượng*: đối với mỗi hệ con, một đặc tả trừu tượng các dịch vụ mà nó cung cấp và các rằng buộc tuân theo phải cung cấp

- *Thiết kế giao diện*: giao diện của từng hệ con với các hệ con khác là được thiết kế và ghi thành tài liệu, đặc tả giao diện không được mơ hồ và cho phép sử dụng hệ con đó mà không cần biết về phép toán hệ con.
- *Thiết kế các thành phần*: Các dịch vụ được cung cấp bởi một hệ con là được phân chia qua các thành phần hợp thành của hệ con đó.
- *Thiết kế cấu trúc dữ liệu*: các cấu trúc dữ liệu/CSDL được dùng trong việc thực hiện hệ thống là được thiết kế chi tiết và được đặc tả.
- *Thiết kế thuật toán*: các thuật toán được dùng để cung cấp cho các dich vụ được thiết kế chi tiết và được đặc tả.

Quá trình này được lặp lại cho mỗi hệ con sao cho đến khi các thành phần hợp thành được minh định đều có thể ánh xạ trực tiếp vào các thành phần ngôn ngữ lập trình, chẳng hạn như các gói, các thủ tục và các hàm.

Phương pháp tiếp cận thường xuyên khuyên dùng là phương pháp tiếp cận từ trên xuống – **Top Down** - vấn đề được phân chia một cách đệ quy thành các vấn đề con cho tới khi các vấn đề dễ giải quyết được xác định. Trong quá trình này, người thiết kế sẽ nhận ra các thành phần có thể dùng lại được. Chú ý rằng người thiết kế không nhất thiết phải phân chia tất cả các thành phần trừu tượng khi mà bằng kinh nghiệm đã biết chắc rằng thành phần nào là chắc chắn xây dựng được. Do đó họ có thể tập trung sức lực cho phần đáng xét nhất.

Chú ý rằng, khi mà phương pháp hướng đối tượng được chấp nhận thì phương pháp từ trên xuống ít có hiệu quả. Khi đó người thiết kế sử dụng các đối tượng sẵn có thể làm khung để thiết kế.

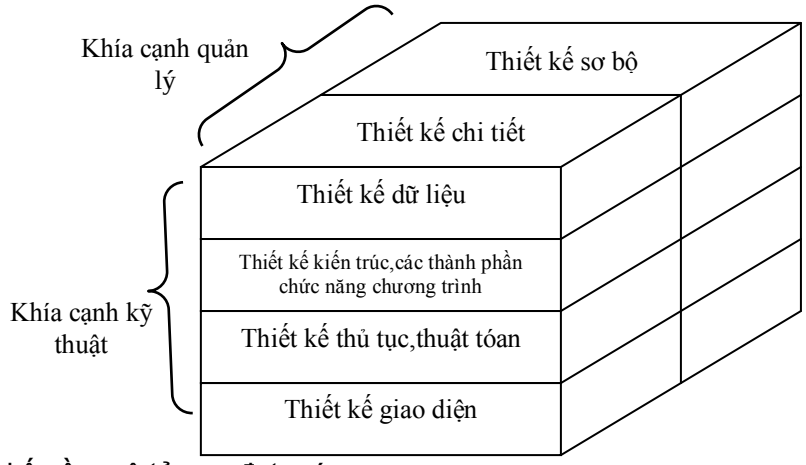
Theo quan điểm quản lý dự án, thiết kế phần mềm được tiến hành theo hai bước

1. Thiết kế sơ bộ : quan tâm tới việc chuyển hóa các yêu cầu thành sản phẩm phần mềm ở mức khái quát
2. Thiết kế chi tiết: chi tiết hóa các công đọan , các thuật toán tính tóan và thuật tóan xử lý dữ liệu và thiết kế giao diện cho phần mềm.

Mối quan hệ giữa các kỹ thuật và quản lý của thiết kế được minh hoạ trong hình vẽ sau.

1.3.2 Việc mô tả thiết kế

a. Thiết kế phần mềm tạo ra một mô hình của thế giới thực mô tả các thực thể và các mối quan hệ của chúng với nhau



b. Thiết kế cần mô tả sao đạt mức
   - Làm cơ sở cho thực hiện chi tiết

- Làm phương tiện liên lạc giữa các nhóm thiết kế các hệ con
- Cung cấp đủ thông tin cho những người bảo trì hệ thống

c. Người ta thường dùng các kỹ thuật đồ hoạ, các ngôn ngữ mô tả chương trình, văn bản không hình thức để tạo dựng các tài liệu thiết kế

1.4 Phương pháp thiết kế

1.4.1 Phương pháp thiết kế

Trong nhiều tổ chức việc thiết kế phần mềm vẫn còn là quá trình tự học, với tập hợp các yêu cầu (thường được mô tả bằng ngôn ngữ tự nhiên) người ta xây dựng một bản thiết kế không hình thức. Sau đó bắt tay ngay vào việc mã hoá và bản thiết kế được cải biên trong khi hệ thống được thực hiện. Khi giai đoạn thực hiện kết thúc thì so với đặc tả ban đầu, sản phẩm đã thay đổi hoàn toàn.

Một cách tiếp cận có phương pháp hơn là: "phương pháp cấu trúc". Đó là các phương pháp làm mịn kiến trúc phần mềm theo kiến trúc từ trên xuống. Các khía cạnh thủ tục của định nghĩa thiết kế đã tiến hoá thành một triết lý là lập trình có cấu trúc.

Phương pháp cấu trúc được dùng rộng rãi trong những năm đầu của thập kỷ 80. Nó đã được dùng thành công trong nhiều dự án lớn, nó làm giảm giá thành đáng kể, sử dụng được các khái niệm chuẩn và đảm bảo rằng các thiết kế tuân theo một chuẩn. Các công cụ CASE đã được dùng để trợ giúp cho phương pháp này.

Trong những năm sau này, phương pháp hướng đối tượng được đề cập tới, được phát triển và được dùng phổ biến hiện nay.

Các phương pháp thiết kế thường trợ giúp một vài cách nhìn nhận hệ thống như sau:

- Nhìn nhận cấu trúc: Cho cái nhìn cấu trúc thông qua Biểu đồ cấu trúc
- Nhìn nhận quan hệ thực thể: mô tả các cấu trúc dữ liệu logic được dùng, nói đến đặc tả dữ liệu, mô hình quan hệ thực thể
- Nhìn nhận dòng dữ liệu: thông qua Biểu đồ dòng dữ liệu

Người ta còn dùng lược đồ chuyển trạng thái để bổ sung cho các phương pháp trên

Để đảm bảo chất lượng cao cho một biểu diễn thiết kế, cần có các tiêu chuẩn cho thiết kế tốt. Song về mặt phương pháp, chúng ta đưa ra các hướng dẫn sau:

1. Thiết kế nêu ra cách tổ chức theo cấp bậc để dùng cách kiểm soát thông minh

2. Thiết kế nên theo các module, tức là phần mềm nên được phân hoạch một cách logic thành các thành phần thực hiện những chức năng xác định

3. Thiết kế nên chứa các biểu diễn phân biệt và tách biệt giữa dữ liệu và thủ tục

4. Thiết kế nên dẫn tới (như chương trình con hay thủ tục) nêu ra các đặc trưng chức năng đặc biệt

5. Thiết kế nên dẫn tới giao diện giúp rút gọn độ phức tạp của việc nối ghép giữa các module và với môi trường bên ngoài.

6. Thiết kế nên được hướng theo phương pháp lặp lại, được điều khiển bởi các dữ liệu có trong bản phân tích các yêu cầu phần mềm.

Các đặc trưng trên của một thiết kế tốt có được khi thực hiện đúng tiến trình thiết kế kỹ nghệ phần mềm thông qua việc áp dụng các nguyên lý thiết kế cơ bản, phương pháp luận hệ thống và việc xét duyệt thấu đáo.

Như vậy mỗi phương pháp thiết kế phần mềm đều đưa vào những cách biểu diễn trực cảm, ký pháp thống nhất cũng như một cách nhìn đặc trưng

Tuy vậy mỗi phương pháp đều có các đặc trưng sau:

1. Một cơ chế để chuyển hóa từ biểu diễn miền thông tin thành biểu diễn thiết kế
2. Một bộ ký pháp để biểu diễn các thành phần chức năng và giao diện của chúng
3. Cách trực cảm để làm mịn phân hoạch
4. Các hướng dẫn về định giá chất lượng

Bất kể phương pháp luận thiết kế nào được dùng, công trình sư phần mềm phải áp dụng một tập các khái niệm nền tảng cho thiết kế dữ liệu, kiến trúc và thủ tục bao gồm:

Trừu tượng, làm mịn, module, kiến trúc phần mềm, cấp bậc điều khiển, cấu trúc dữ liệu, thủ tục phần mềm, che dấu thông tin

Tập hợp các khái niệm thiết kế nền tảng đã thiết kế hơn 3 thập kỷ. Mỗi khái niệm đều cung cấp cho người thiết kế phần mềm một nền tảng để từ đó người ta có thể áp dụng nhiều phương pháp thiêt kế phức tạp.

M.A.JACKSON nói: ”điểm bắt đầu của một kỹ sư phần mềm khôn ngoan là thừa nhận sự khác biệt giữa việc bắt tay vào làm chương trình hay trước khi bắt tay vào làm phải hiểu vấn đề một cách đúng đắn”. Các khái niệm thiết kế phần mềm nền tảng cung cấp một khuôn khổ cần thiết để hiểu vấn đề một cách đúng đắn.

1.4.2 Các khái niệm nền tảng của thiết kế

1.4.2.1 Trừu tượng- khái quát hóa : nghĩa là khả năng tóm lược tòan bộ sản phẩm phần mềm gồm những việc chính là gì.

Có nhiều mức trừu tượng

- Mức cao nhất: một giải pháp được phát biểu theo *thuật ngữ đại thể* bằng cách dùng ngôn ngữ của môi trường vấn đề.
- Mức vừa: lấy khuynh hướng thủ tục nhiều hơn. *Thuật ngữ hướng vấn đề* thường đi đôi với thuật ngữ hướng cài đặt trong mô tả giải pháp.
- Mức thấp: giải pháp được phát biểu theo *thuật ngữ chi tiết* để có thể cài đặt trực tiếp

Mỗi bước trong tiến trình kỹ nghệ phần mềm đều là sự làm mịn cho một mức trừu tượng của phần mềm. Trong kỹ nghệ hệ thống, phần mềm được dùng như một phần tử của hệ thống dựa trên máy tính. Trong phân tích các yêu cầu phần mềm, giải pháp phần mềm được phát biểu dưới dạng ”đó là cái quan trọng trong môi trường vấn đề”. Khi chúng ta chuyển từ thiết kế sơ bộ sang thiết kế chi tiết thì mức độ trừu tượng được rút lại. Cuối cùng, ta đi tới mức trừu tượng thấp nhất khi sinh ra chương trình gốc.

Có nhiều dạng trừu tượng:

Khi chúng ta chuyển sang mức trừu tượng khác nhau, chúng ta làm việc để tạo ra các trừu tượng thủ tục và dữ liệu. *Trừu tượng thủ tục* là một dãy các lệnh có tên, có một chức năng xác định và có giới hạn. Một ví dụ về thủ tục trừu tượng là từ “đi vào” cửa. “Đi vào” kéo theo một dãy dài các bước thủ tục (như bước tới cửa, lại gần và nắm lấy quả đấm, xoay quả đấm cửa và kéo cửa ra, bước vào…). *Trừu tượng dữ liệu* là một tập hợp các dữ liệu có tên mô tả cho sự vật dữ liệu. Ví dụ về dữ liệu trừu tượng là “séc thanh toán”. Đối tượng dữ liệu này thực chất là một tập hợp nhiều mẩu thông tin khác nhau (như tên người thanh toán, số tiền thanh toán, tiền thuế…). Vậy chúng ta có thể tham khảo mọi dữ liệu bằng cách nói tên của trừu tượng dữ liệu.

Trừu tượng thủ tục

Tại mức trừu tượng này đã có việc biểu diễn thủ tục sơ bộ. Thuật ngữ này đã hướng phần mềm (như việc dùng các cấu trúc *do while*) và việc dính líu tới module bắt đầu nổi lên bề mặt.

Khái niệm về làm mịn dữ liệu từng bước và module gắn liền với việc trừu tượng. Khi thiết kế phần mềm tiến hóa, từng mức module trong cấu trúc chương trình sẽ biểu diễn cho việc làm mịn dần trong mức trừu tượng của phần mềm.

Trừu tượng dữ liệu

Giống như trừu tượng thủ tục, làm cho người thiết kế có thể biểu diễn một sự vật dữ liệu ở các mức chi tiết khác nhau, nhưng điều quan trọng hơn, là xác định một sự vật dữ liệu trong hoàn cảnh các thao tác (thủ tục) có thể được áp dụng vào nó.

Một số các ngôn ngữ lập trình (như ADA, MODULA, CLU) đưa ra cơ chế để tạo kiểu dữ liệu trừu tượng. Chẳng hạn, *package* của ADA là một cơ chế ngôn ngữ lập trình đưa ra sự hỗ trợ cho cả trừu tượng dữ liệu và thủ tục. Kiểu trừu tượng dữ liệu nguyên gốc được dùng như một tiêu bản hay cấu trúc dữ liệu sinh ra để từ đó có thể làm *thể nghiệm* cho các cấu trúc dữ liệu khác.

Trừu tượng điều khiển

Trừu tượng điều khiển là dạng thứ ba của trừu tượng hóa được dùng trong thiết kế phần mềm. Giống như trừu tượng dữ liệu và thủ tục, trừu tượng điều khiển áp dụng cho cơ chế điều khiển chương trình mà không xác định các chi tiết bên trong. Một ví dụ về điều khiển là cơ chế đồng bộ hóa được dùng để điều hòa các hoạt động trong hệ điều hành.

1.4.2.2 Làm mịn:

Làm mịn từng bước là một *chiến lược thiết kế trên - xuống* ban đầu do NIKLAUS WIRTH đề nghị. Kiến trúc của một chương trình được phát triển bằng các mức làm mịn liên tiếp các chi tiết thủ tục. Một cấp bậc được xây dựng nên bằng cách phân tách một phát biểu vĩ mô về chức năng (trừu tượng thủ tục) theo kiểu từng bước cho tới khi đạt tới phát biểu bằng ngôn ngữ lập trình.

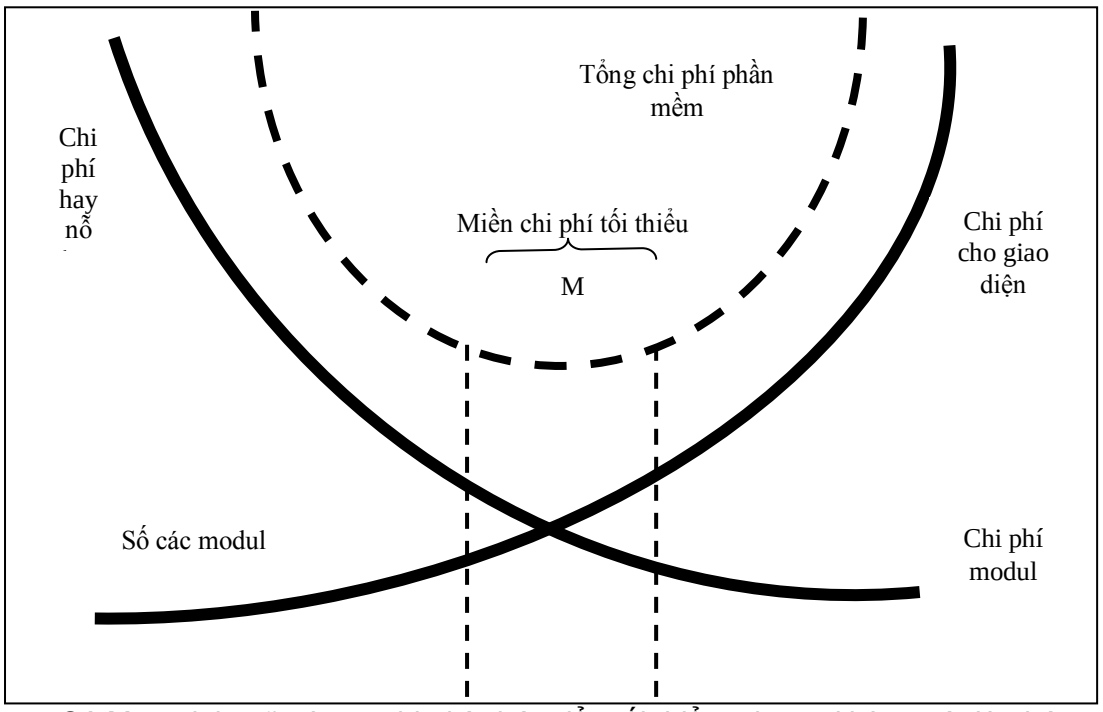
Làm mịn trước tiên là một tiến trình khởi thảo. Bắt đầu với một phát biểu về chức năng (hay mô tả thông tin) được xác định như mức trừu tượng cao. Tức là, phát biểu mô tả chức năng hay thông tin về mặt quan niệm, nhưng không đưa ra thông tin về cách làm việc nội bộ của chức năng hay cấu trúc nội bộ của thông tin đó. Việc làm mịn buộc người thiết kế phải khởi thảo phát biểu nguyên gốc, sau đó đưa ra ngày càng nhiều chi tiết khi việc làm mịn khởi thảo kế tiếp xuất hiện.

1.4.2.3 Tính module

Phần mềm được chia thành các thành phần có tên riêng biệt và định địa chỉ được, gọi là các *module,* sau đó chúng được tích hợp lại để thỏa mãn yêu cầu.

Người ta nói rằng, tính module là thuộc tính riêng của phần mềm cho phép một chương trình nên quản lý được theo cách thông minh. Người đọc không thể nào hiểu thấu phần mềm nguyên khối (như một chương trình lớn chỉ gồm một module). Điều này dẫn đến kết luận chia để trị sẽ dễ giải quyết một vấn đề phức tạp hơn khi chia nó thành những phần quản lý được.

Nỗ lực (chi phí) để phát triển một module phần mềm riêng lẻ không giảm đi khi tổng số các module tăng lên. Với cùng tập hợp các yêu cầu, nhiều module hơn tức là kích cỡ từng module sẽ nhỏ hơn. Tuy nhiên khi số các module tăng lên thì nỗ lực liên kết với việc làm giao diện cho các module cũng tăng lên. Hình vẽ mô tả đặc trưng này dẫn đến đường cong tổng phí :



Có M module sẽ gây ra chi phí phát triển tối thiểu, nhưng không có độ phức tạp cần thiết để dự kiến M với sự đảm bảo.

Đường cong được vẽ đưa ra thông tin có ích khi phải xét tính tới module. Chúng ta nên module hóa nhưng cũng phải chú ý duy trì trong vùng lân cận của M. Module hóa còn chưa đủ hay qua mức đều nên tránh.

Việc tìm vùng lân cận của M, chia module phần mềm đến đâu là câu hỏi cần trả lời vì kích cỡ của module bị khống chế bởi chức năng của nó và ứng dụng.

1.4.2.4 Kiến trúc phần mềm

Kiến trúc phần mềm gồm hai đặc trưng quan trọng của chương trình máy tính:

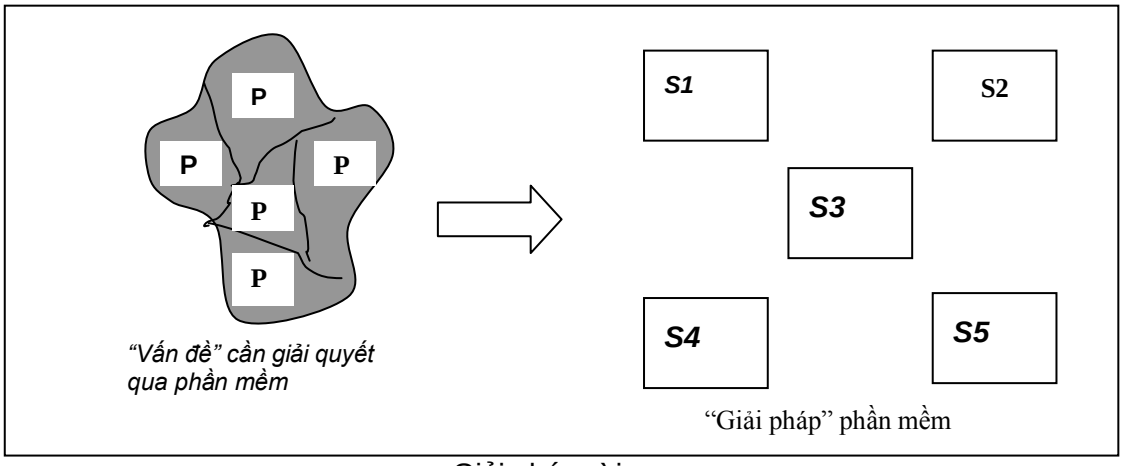
1. Cấu trúc cấp bậc điều khiển các thành phần thủ tục (module, hoặc cấu trúc chương trình)

2. Cấu trúc dữ liệu

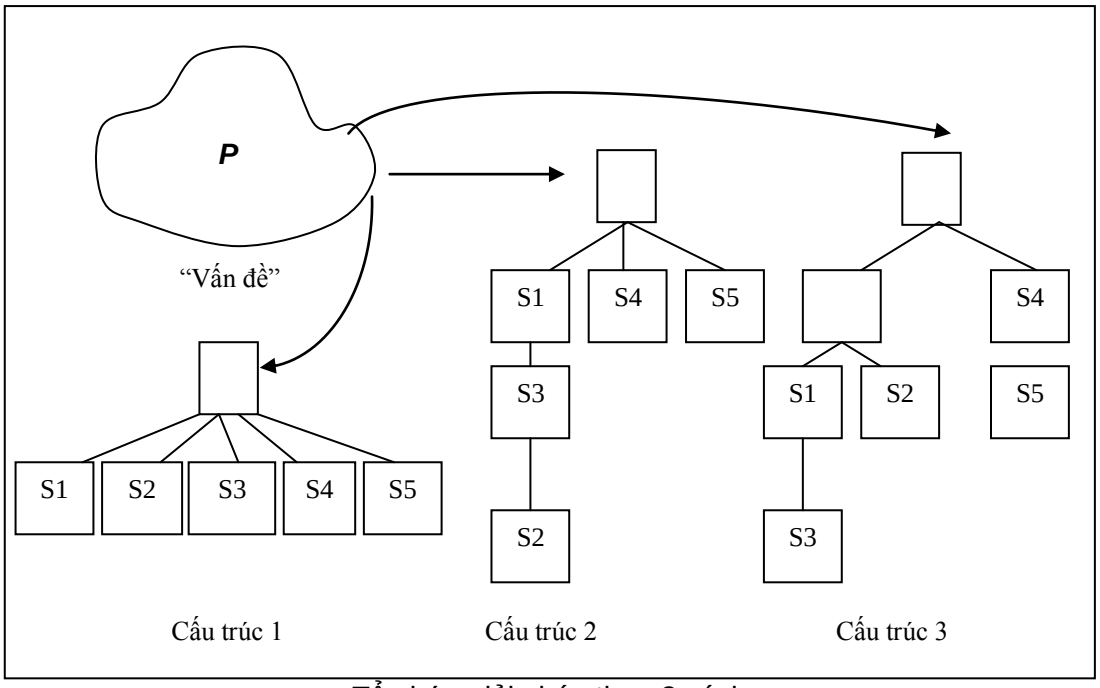
Kiến trúc phần mềm được suy dẫn ra qua tiến trình phân hoạch và đặt mối quan hệ giữa các phần tử của giải pháp phần mềm với các bộ phận của thế giới thực, được xác định không tường minh trong bản phân tích yêu cầu.

Sự tiến hóa của phần mềm và cấu trúc dữ liệu bắt đầu từ việc xác định vấn đề. Giải pháp xuất hiện khi từng phần của vấn đề được giải quyết bởi một hay nhiều phần tử phần mềm. Tiến

trình này được biểu diễn tượng trưng trong hình vẽ dưới đây, biểu thị cho một phép chuyển giữa việc phân tích yêu cầu phần mềm và thiết kế.



Giải pháp rời rạc



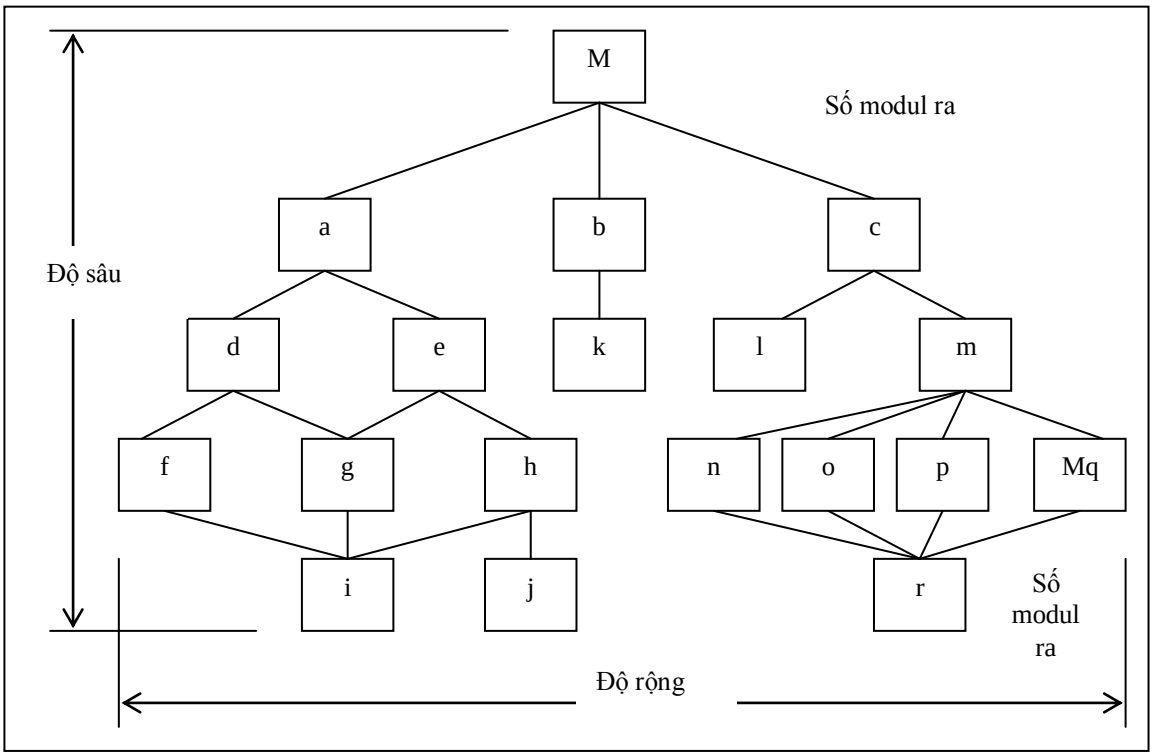
Tổ chức giải pháp theo 3 cách

Một vấn đề có thể được thỏa mãn bởi nhiều cấu trúc khác nhau. Phương pháp thiết kế phần mềm có thể được dùng để suy ra cấu trúc, nhưng vì từng cấu trúc lại dựa trên các khái niệm nền tảng khác nhau về thiết kế tốt, cho nên từng phương pháp sẽ phát sinh trong một cấu trúc khác biệt đối với cùng tập các yêu cầu phần mềm. Không có câu trả lời dễ dàng với các câu hỏi "Cái nào tốt nhất?". Tuy nhiên, có những đặc trưng về cấu trúc mà ta có thể kiểm tra để xác định chất lượng tổng thể.

1.4.2.5 Cấp bậc điều khiển

Cấp bậc điều khiển còn gọi là cấu trúc chương trình, biểu thị cho cách tổ chức của các thành phần chương trình (module). Nó không biểu thị các khía cạnh thủ tục của phần mềm như dãy các xử lý, sự xuất hiện, thứ tự các quyết định hay việc lặp lại các thao tác.

Người ta dùng các ký pháp khác nhau để biểu diễn cho cấp bậc điều khiển. Thông dụng nhất là biểu đồ kiểu cây. Trong đó độ sâu và chiều rộng đưa ra một chỉ báo về số mức điều khiển và độ trải rộng toàn bộ của điều khiển tương ứng. *Số module ra* là độ đo, đo số các module trục tiếp bị điều khiển bởi các module khác. *Số module vào* chỉ ra cách thức module trực tiếp một điều khiển một module đã cho.



Mối quan hệ điều khiển giữa các module được diễn tả theo cách sau: Một module điều khiển một module khác thì được gọi là thượng cấp của nó, và ngược lại module bị một module khác điều khiển thì được gọi là thuộc cấp của module điều khiển. Chẳng hạn: module M là thượng cấp của các module a, b và c. Module h là thuộc cấp của module e và cuối cùng là thuộc cấp của module M. Mối quan hệ theo chiều rộng (tức là mối quan hệ giữa các module d và e), mặc dầu có thể diễn tả trong thực tế nhưng không nhất thiết phải được xác định bằng thuật ngữ tường minh.

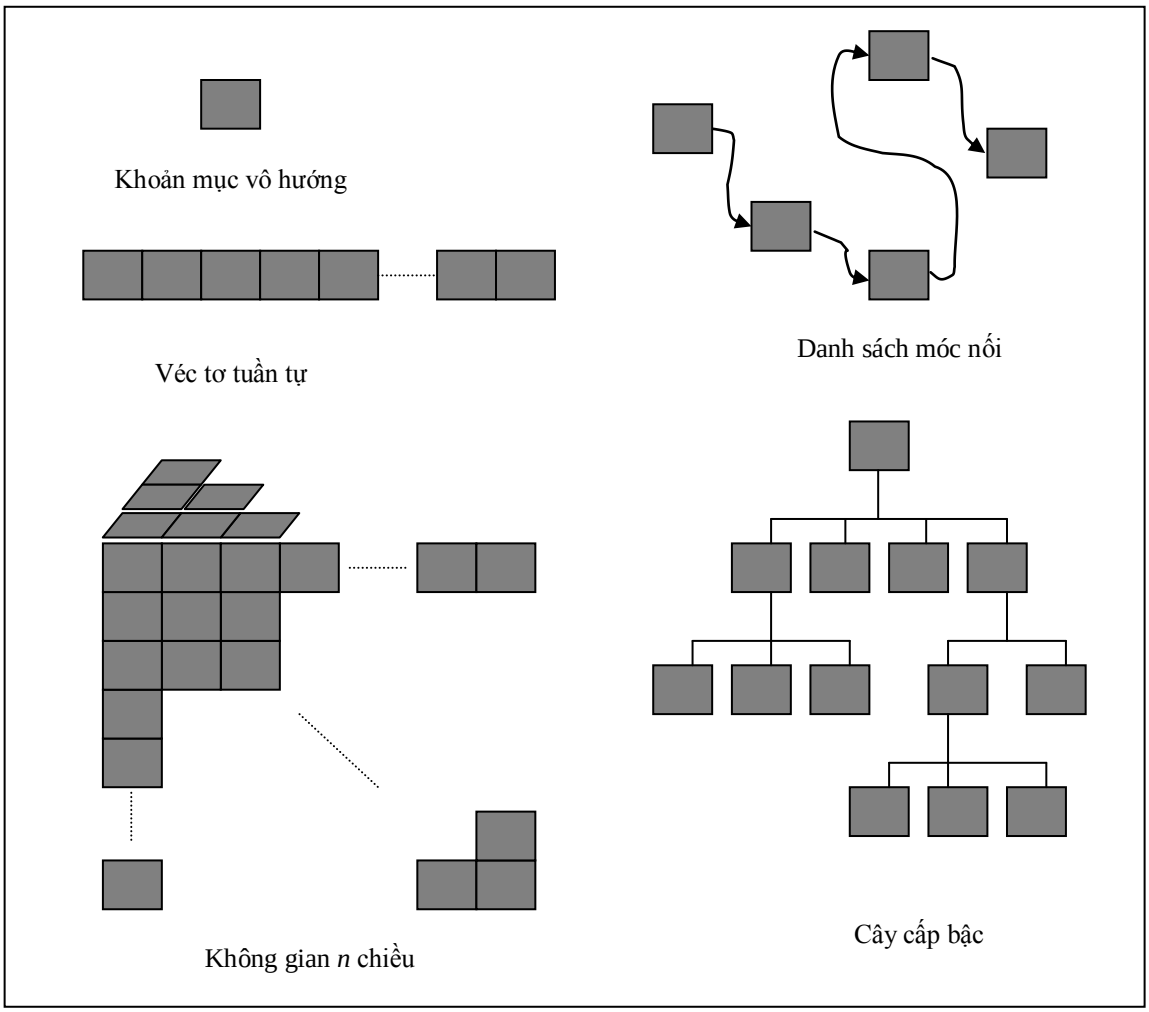
Cấp bậc điều khiển cũng biểu diễn cho hai đặc trưng khác nhau của cấu trúc phần mềm: tính *thấy được* và tính *nối được.* Tính thấy được chỉ ra tập hợp các thành phần chương trình có thể được gọi hay được dùng như dữ liệu bởi một thành phần đã cho, ngay cả khi điều này được thực hiện gián tiếp. Chẳng hạn, một module trong hệ thống hướng đối tượng có thể thâm nhập vào một mảng rộng các sự vật dữ liệu mà nó đã kế thừa, nhưng chỉ dùng một số nhỏ các các sự vật dữ liệu đó. Tính nối được chỉ tập các thành phần trực tiếp được gọi hay được sử dụng như dữ liệu bởi một thành phần đã cho. Chẳng hạn, một module trực tiếp gây ra cho một module khác bắt đầu thực hiện là được nối với nó.

1.4.2.6 Cấu trúc dữ liệu

Cấu trúc dữ liệu biểu diễn mối quan hệ logic giữa các phần tử dữ liệu riêng lẻ. Vì cấu trúc thông tin sẽ luôn luôn ảnh hưởng tới thiết kế thủ tục cuối cùng nên cấu trúc dữ liệu rất quan trọng như cấu trúc chương trình để biểu thị kiến trúc phần mềm.

Ý nghĩa: Cấu trúc dữ liệu chi phối cách tổ chức, các phương pháp thâm nhập, mức độ kết hợp và các phương án xử lý thông tin.

Tổ chức và độ phức tạp của các cấu trúc dữ liệu chỉ bị giới hạn bởi tài khéo léo của người thiết kế. Tuy nhiên cũng có một số hạn chế, các cấu trúc dữ liệu cổ điển vốn tạo nên các khối xây dựng cho nhiều cấu trúc dữ liệu phức tạp.



Khoản mục vô hướng: là cấu trúc dữ liệu đơn giản nhất trong các cấu trúc dữ liệu. Như tên của nó hàm ý, khoản mục vô hướng biểu thị cho một phần tử thông tin đơn giản có thể được đánh địa chỉ bằng một tên gọi; tức là việc thâm nhập có thể được đạt tới bằng cách xác định chỉ một địa chỉ trong bộ nhớ. Kích cỡ và định dạng của của một khoản mục vô hướng có thể thay đổi bên trong các giới hạn do ngôn ngữ lập trình khống chế. Chẳng hạn, một khoản mục vô hướng có thể là thực thể logic dài 1 bit, một số nguyên hay số phảy động dài từ 8 bit đến 64 bit, hay một xâu ký tự một trăm hay một nghìn byte.

Khi các khoản mục vô hướng được tổ chức như một danh sách hay nhóm liên tục thì một *vectơ tuần tự* sẽ được hình thành. Vectơ là phần chung nhất của tất cả các cấu trúc dữ liệu và mở cánh cửa tới việc làm chỉ số thay đổi cho thông tin. Để minh họa cho chúng ta xem xét một thí dụ Pascal đơn giản:

```
Type G = array [1..100] of integer;
…
procedure S(var T:G; n: integer; sum: integer)
```

```
var i: integer;
begin
sum:= 0;
for i:= 1 to n do
        sum:= sum + T[i];
end;
```

Vectơ tuần tự G (mảng) gồm 100 khoản mục nguyên vô hướng, được định nghĩa. Việc thâm nhập vào từng phần tử của G đều được *chỉ số hóa* trong thủ tục S sao cho các phần tử của cấu trúc dữ liệu được tham khảo tới theo một trật tự xác định.

Khi vectơ tuần tự được mở rộng thành hai, ba và cuối cùng thành số chiều bất kỳ, thì một *không gian n chiều* sẽ được tạo ra. Không gian n chiều thông dụng nhất là ma trận hai chiều. Trong hầu hết các ngôn ngữ lập trình, một không gian n chiều được gọi là một mảng.

Khoản mục véc tơ và không gian có thể được tổ chức theo nhiều định dạng. Danh sách móc nối là một cấu trúc dữ liệu tổ chức các khoản mục vô hướng, véc tơ hay không gian liên tục theo một cách (còn gọi là nút, đỉnh) làm cho chúng được xư lý như một danh sách. Mỗi đỉnh chứa cách tổ chức dữ liệu thích hợp (như véc tơ) và một hay nhiều con trỏ chỉ ra địa chỉ trong bộ nhớ của đỉnh tiếp trong danh sách. Có thể bổ xung thêm các đỉnh tại bất kỳ điểm nào trong danh sách bằng các định nghĩa lại các con trỏ để thích hợp với lối vào danh sách mới.

Các cấu trúc dữ liệu khác tổ hợp hay được xây dựng bằng cách dùng các cấu trúc dữ liệu nền tảng được mô tả ở trên. Chẳng hạn, cấu trúc dữ liệu cấp bậc được điều khiển bằng cách sử dụng danh sách đa móc nối có chứa các khoản mục vô hướng, véc tơ và có thể cả không gian n chiều. Cấu trúc cấp bậc thường hay gặp trong các ứng dụng có đòi hỏi phân loại và liên kết thông tin. Phân loại bao hàm việc gộp nhóm thông tin theo một phân lọai tổng quát nào đó.

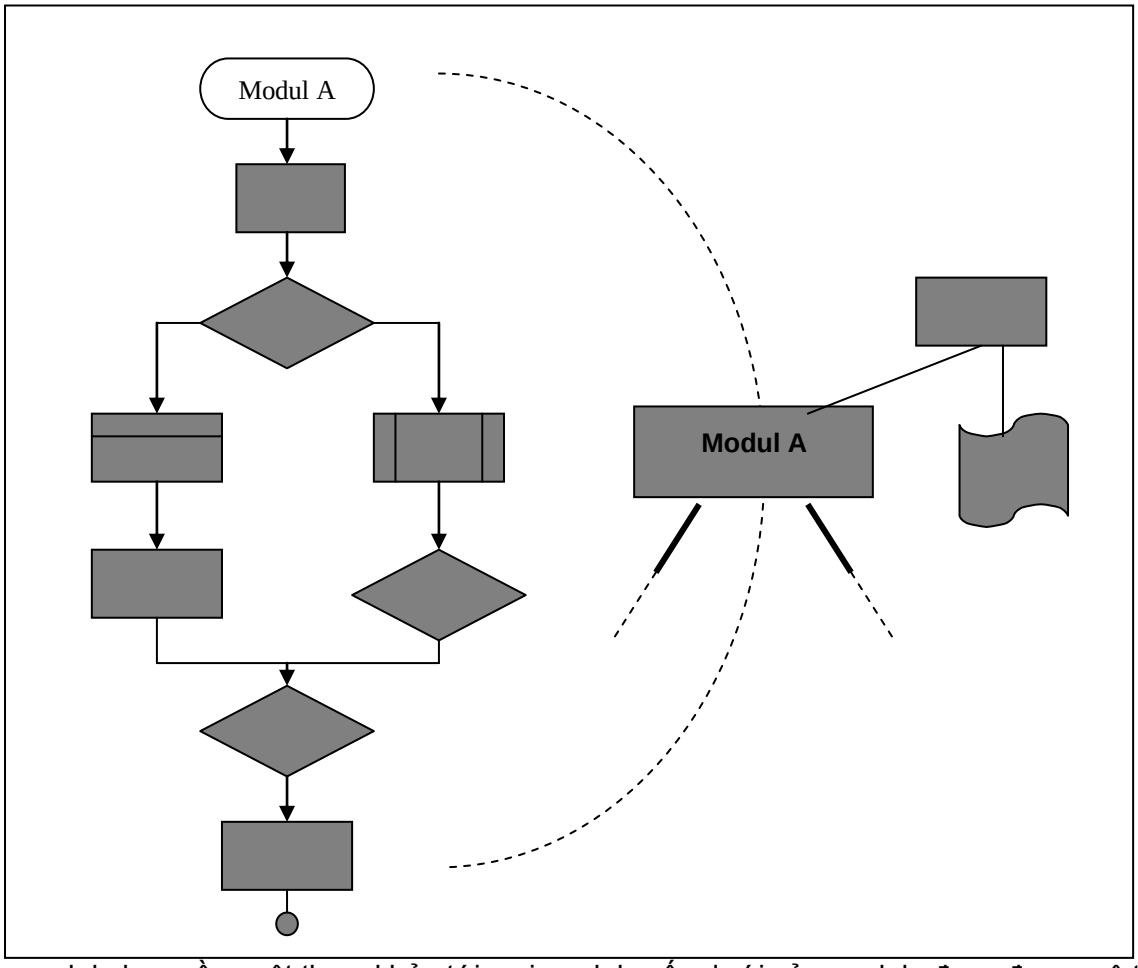
Tính kết hợp hàm ý khả năng liên kết thông tin từ các phân loại khác nhau.

Điều quan trọng cần lưu ý rằng, cấu trúc dữ liệu, giống như cấu trúc chương trình, có thể được biểu diễn ở các mức trừu tượng khác nhau. Chẳng hạn, ngăn xếp (stack) là một mô hình khái niệm về cấu trúc dữ liệu có thể được cài đặt như một véc tơ hay danh sách móc nối.

1.4.2.7 Thủ tục phần mềm

Cấu trúc chương trình xác định ra cấp bậc điều khiển không để ý đến dãy các xử lý và quyết định. Thủ tục phần mềm tập trung vào các chi tiết xử lý cho từng module riêng biệt. Thủ tục phải cung cấp một đặc tả chính xác về xử lý, kể cả trình tự các sự kiện, các điểm quyết định chính xác, các thao tác lặp lại, và ngay cả cấu trúc/ tổ chức dữ liệu.

Giữa cấu trúc và thủ tục có một mối quan hệ chặt chẽ. Việc xử lý được chỉ ra trong từng



module bao gồm một tham khảo tới mọi module cấp dưới của module đang được mô tả.

1.4.2.8 Che dấu thông tin

Khái niệm về module đưa người thiết kế phần mềm tới một câu hỏi nền tảng: làm sao ta phân rã một giải pháp phần mềm ra để thu được tập các module tốt nhất. Nguyên lý về che dấu thông tin gợi ý rằng các module nên được đặc trưng bởi những quyết định thiết kế mà ẩn kín với mọi module khác. Nói cách khác, module nên được đặc tả và thiết kế sao cho thông tin được chứa trong một module này không thể thâm nhập tới được từ các module khác vốn không cần tới những thông tin đó.

Việc che dấu kéo theo rằng, người ta có thể đạt được tính module bằng cách xác định một tập các module độc lập mà trao đổi giữa module nọ với module kia chỉ là những thông tin cần thiết cho sự vận hành của phần mềm. Việc trừu tượng hóa giúp cho việc xác định các thực thể thủ tục có chứa phần mềm. Việc che dấu xác định và áp đặt các ràng buộc thâm nhập tới cả chi tiết thủ tục bên trong module đó và bất kỳ cấu trúc dữ liệu cục bộ nào mà module đó sử dụng.

Việc che dấu thông tin được coi là một tiêu chuẩn thiết kế đối với hệ thống module, đưa ra những ích lợi lớn nhất khi cần có những thay đổi trong việc kiểm thử và sau này trong bảo trì phần mềm. Vì phần lớn dữ liệu và thủ tục đều bị che dấu khỏi các bộ phận khác của phần

mềm, nên những sai sót bất cẩn bị đưa vào trong khi thay đổi sẽ ít có khả năng lan truyền sang các vị trí khác bên trong phần mềm.

1.4.3 Các chiến lược thiết kế

Xét các chiến lược hay được nhắc đến: thiết kế chức năng, thiết kế hướng đối tượng, thiết kế hệ thống tương tranh.

1.4.3.1 Thiết kế chức năng

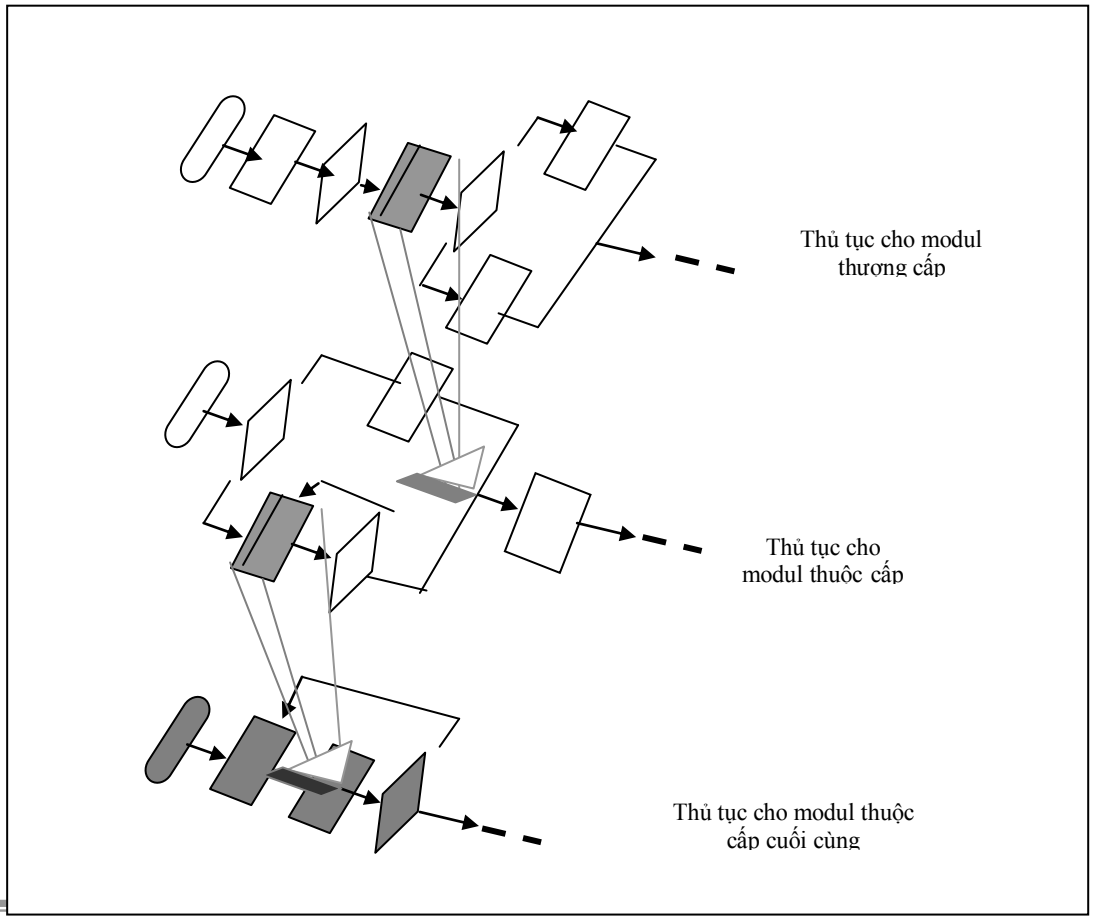
Hệ thống được thiết kế theo quan điểm chức năng, bắt đầu ở mức cao nhất, sau đó tinh chế dần dần để thành thiết kế chi tiết hơn. Trạng thái của hệ thống là tập trung và được chia sẻ cho các chức năng thao tác trên trạng thái đó.

1.4.3.2 Thiết kế hướng đối tượng

Hệ thống được nhìn nhận như một bộ các đối tượng (chứ không phải là một bộ chức năng). Hệ thống được phân tán, mỗi đối tượng có những thông tin trạng thái riêng của nó. Đối tượng là bộ các 'thuộc tính' xác định trạng thái của đối tượng đó và các phép toán thực hiện trên các thuộc tính đó. Mỗi đối tượng là một khách thể của một lớp mà *lớp được xác định bởi các thuộc tính và các phương thức của nó*. Đối tượng có thể được thừa kế từ một vài lớp đối tượng lớp cao hơn, sao cho định nghĩa nó chỉ cần nêu đủ các khác nhau giữa nó và các lớp cao hơn nó. Các đối tượng liên lạc với nhau chỉ bằng cách trao đổi các thông báo: thực tế hầu hết các liên lạc giữa các đối tượng thực hiện bằng cách một đối tượng này gọi một thủ tục, mà thủ tục này kết hợp với một đối tượng khác.

Thiết kế hướng đối tượng dựa trên ý tưởng che dấu thông tin. Thiết kế hướng đối tượng gần đây được phát triển nhiều đã tạo ra các hệ thống cấu tạo bởi nhiều thành phần độc lập và có tương tác với nhau.

Sự thật, các hệ phần mềm lớn là phức tạp, đến mức mà người ta đã dùng các phương



pháp tiếp cận khác nhau trong việc thiết kế các thành phần khác nhau của một hệ thống. Không có một chiến lược tốt nhất nào cho các dự án lớn. Các cách tiếp cận hướng chức năng và hướng đối tượng là bổ xung hỗ trợ cho nhau chứ không đối kháng nhau. Kỹ sư phần mềm sẽ chọn cách tiếp cận thích hợp nhất cho từng giai đoạn thiết kế. Nhìn ở mức tổng thể thì cách tiếp cận hướng đối tượng là thích hợp hơn. Đến mức chi tiết thì tự nhiên hơn là nên xem xét chúng là các chức năng tương tác giữa các đối tượng. Sau đó mỗi đối tượng lại được phân giải thành các thành phần, tức là lại có thể xem nó như một hệ (con).

Rất nhiều hệ thống, đặc biệt là một hệ thống thời gian thực được nhúng (vào một hệ thiết bị vật chất có thực) được cấu tạo như là một hệ gồm một bộ các quá trình song song có liên lạc với nhau. Các hệ này thường phải tuân theo các ràng buộc nghiêm ngặt về thời gian, mà các phần cứng thường hoạt động tương đối chậm, chỉ có cách tiếp cận nhiều bộ xử lý hoạt động song song mới có thể hoàn thành được yêu cầu về thời gian.

Các chương trình tuần tự là dễ thiết kế, thực hiện, kiểm tra và thử nghiệm hơn là các hệ song song. Sự phụ thuộc thời gian giữa các quá trình là khó hình thức hóa, khó khống chế và thử nghiệm

Do đó quá trình thiết kế nên được xem như một hoạt động gồm hai giai đoạn

1. Minh định cấu trúc thiết kế logic, cụ thể là các thành phần của hệ thống và các mối quan hệ giữa chúng. Có thể dùng cách nhìn chức năng hoặc cách nhìn đối tượng.
2. Thực hiện cấu trúc đó trong dạng có thể thực hiện được. Giai đoạn này đôi khi được gọi là thiết kế chi tiết và đôi khi là lập trình. Chắc rằng sự quyết định về tính song song nên làm ở giai đoạn này chứ không phải là các giai đoạn sớm hơn trong quá trình thiết kế.

1.4.4 Chất lượng thiết kế

Không có cách nào hay để xác định được thế nào là thiết kế tốt. Phụ thuộc vào ứng dụng và vào yêu cầu dự án, một thiết kế tốt hẳn là một thiết kế mà nó cho phép sinh ra mã hữu hiệu, nó có thể là một thiết kế tối thiểu mà theo đó việc thực hiện là càng chặt càng tốt, hoặc nó là thiết kế bảo dưỡng được tốt nhất.

Tiêu chuẩn để bảo dưỡng là tiêu chuẩn tốt cho người dùng. Một thiết kế bảo dưỡng được tốt có thể thích nghi với việc cải biên các chức năng và việc thêm các chức năng mới. Do đó thiết kế như thế phải là dễ hiểu và việc sửa đổi chỉ có hiệu ứng cục bộ. Các thành phần thiết kế phải là kết dính theo nghĩa là tất cả các phần trong thành phần đó phải có một quan hệ logic chặt chẽ. Các thành phần ấy phải là nối ghép lỏng lẻo. Sự nối ghép là độ đo của tính độc lập của các thành phần. Nối ghép càng lỏng lẻo càng dễ thích ghi.

Để xem một thiết kế có là tốt hay không, người ta tiến hành thiết lập một số độ đo chất lượng thiết kế:

1.4.4.1 Sự liên kết giữa các thành phần (Cohension)

Sự kết dính của một thành phần là độ đo về tính khớp lại với nhau. Một thành phần hoặc thực hiện một chức năng logic hoặc thực hiện một thực thể logic. Tất cả các phần của thành phần đó đều tham gia vào việc thực hiện. Nếu một thành phần không tham gia trực tiếp vào chức năng logic đó thì mức độ kết dính của nó là thấp.

Constantin và Yourdon định ra 7 mức kết dính theo thứ tự tăng dần sau đây:

1. Kết dính gom góp: các phần của thành phần không liên quan với nhau, song bị bó vào một thành phần

2. Hội hợp logic: các thành phần cùng thực hiện các chức năng tương tự chẳng hạn như vào, xử lý lỗi, …

3. Kết dính theo thời điểm: tất cả các thành phần cùng hoạt hoá một lúc, chẳng hạn như bắt đầu và kết thúc.

4. Kết dính thủ tục: các phần tử trong thành phần được ghép lại trong một dãy điều khiển.

5. Kết dính truyền thống: tất cả các phần tử của thành phần cùng thao tác trên một dữ liệu vào và đưa ra cùng một dữ liệu ra.

6. Kết dính tuần tự: trong một thành phần, ra của một thành phần là vào của phần tử khác.

7. Kết dính chức năng: mỗi phần tử của một thành phần đều là cần thiết để thi hành cùng một chức năng nào đó.

Một đối tượng kết dính là một đối tượng thể hiện một thực thể đơn, tất cả các phép toán trên thực thể đó đều nằm trong thực thể đó. Vậy có thể xác định một lớp kết dính nữa đó là:

h. Kết dính đối tượng: mỗi phép toán cho một chức năng, chức năng này cho phép các thuộc tính của đối tượng đó được cải biên, thanh tra và sử dụng như là cơ sở cho sự cung cấp dịch vụ.

Các lớp kết dính này không được định nghĩa chặt chẽ và cũng không phải luôn luôn quyết định được.

1.4.4.2 khả năng ghép nối (Coupling)

Ghép nối liên quan đến kết dính, nó chỉ ra độ nối ghép giữa các đơn vị của chương trình. Hệ thống có nối ghép cao sẽ có độ kết dính mạnh giữa các đơn vị, các đơn vị phụ thuộc lẫn nhau. Hệ thống nối ghép lỏng lẻo làm cho các đơn vị là độc lập hoặc là tương đối độc lập với nhau.

Các module là được ghép nối chặt chẽ nếu chúng dùng các biến chung và nếu chúng trao đổi các thông tin điều khiển (ghép nối chung nhau và ghép nối điều khiển). Ghép nối lỏng lẻo đạt được khi đảm bảo rằng các thông tin biểu diễn được giữ trong thành phần này là giao diện dữ liệu của nó với các đơn vị khác lại thông qua danh sách tham số của nó.

Có lẽ ưu việt của thiết kế hướng đối tượng là bản chất của đối tượng dẫn tới việc tạo ra các hệ ghép nối lỏng lẻo.

Việc thừa kế trong hệ thống hướng đối tượng lại dẫn tới một dạng khác của ghép nối.

1.4.4.3 Sự hiểu được (Understandability)

Sự hiểu được liên quan tới một số đặc trưng thành phần sau:

a. Tính kết dính: có thể hiểu được thành phần đó mà không cần tham khảo tới một thành phần khác hay không?
b. Đặt tên: Tên có nghĩa là những tên phản ánh tên của thực thể trong thế giới thực được mô hình bởi thành phần đó. Phải chăng là mọi tên được dùng trong thành phần đều có ý nghĩa?
c. Soạn tư liệu: Thành phần có được soạn thảo tư liệu sao cho ánh xạ giữa các thực thể trong thế giới thực và thành phần đó là rõ ràng hay không?
d. Độ phức tạp: độ phức tạp của các thuật toán được dùng để thực hiện thành phần đó như thế nào?

Từ phức tạp ở đây được dùng theo nghĩa không hình thức. Độ phức tạo cao ám chỉ nhiều quan hệ giữa các thành phần khác nhau của thành phần thiết kế đó và một cấu trúc logic phức tạp mà nó dính líu đến độ sâu lồng nhau của phát biểu if-then-else. Các thành phần phức tạp là khó hiểu, vì thế người thiết kế hẳn là nên làm cho thiết kế thành phần càng đơn giản càng tốt.

Đa số công việc về đo chất lượng thiết kế được tập trung và cố gắng đo độ phức tạp của thành phần và từ đó thu được một vài độ đo về sự dễ hiểu của thành phần. Độ phức tạp phản

ánh độ dễ hiểu, nhưng cũng có một số nhân tố khác ảnh hưởng đến độ dễ hiểu, chẳng hạn như tổ chức dư liệu và kiểu cách mô tả thiết kế. Các số đo độ phức tạp có thể chỉ cung cấp một chỉ số cho độ dễ hiểu của một thành phần.

Sự thừa kế trong thiết kế hướng đối tượng phản ánh độ dễ hiểu. Nếu sự thừa kế được dùng để gắn các chi tiết thiết kế thì thiết kế sẽ dễ hiểu hơn. Mặt khác nếu sử dụng sự thừa kế đòi hỏi người thiết kế phải nhìn nhiều lớp đối tượng khác nhau trong trật tự thừa kế thì độ dễ hiểu của thiết kế là được rút gọn.

1.4.4.4 Khả năng thích nghi(Adaptability)

Nếu một thiết kế nhằm bảo trì được thì nó phải sẵn sàng thích nghi được, nghĩa là các thành phần của chúng nên được ghép nối lỏng lẻo. Để thích nghi được, thiết kế đó phải được soạn thảo tư liệu tốt, dễ hiểu, kiên định với việc thực hiện, việc thực hiện cũng được viết ra một cách dễ đọc.

Một thiết kế dễ thích nghi nghĩa là có mức nhìn thấy được cao, có một quan hệ rõ ràng giữa các mức khác nhau của thiết kế. Khi đó người đọc thiết kế có thể tìm được các biểu diễn liên quan tới lược đồ cấu trúc biểu diễn sự vận chuyển của biểu đồ dòng dữ liệu.

Cần dễ dàng kết hợp chặt chẽ các biến đổi về thiết kế trong toàn bộ tư liệu thiết kế, nếu không, các thay đổi thiết kế có thể sẽ không thể đưa vào trong các mô tả liên quan. Tư liệu thiết kế đó có thể trở nên không kiên định.

Để có độ thích ghi cao thì một thành phần phải là tự chứa. Một thành phần có thể ghép nối lỏng lẻo theo nghĩa là chỉ hợp tác với các thành phần khác thông qua việc truyền các hộp thông báo. Điều này không giống như là tự chứa vì thành phần đó có thể dựa trên các thành phần khác. Muốn là tự chứa một cách hoàn toàn thì một thành phần không nên dùng các thành phần khác được xác định ngoại lai. Tuy nhiên điều đó lại mâu thuẫn với quan điểm là các thành phần hiện có nên được dùng lại. Vậy là cần có sự công bằng giữa tính ưu việt của sự dùng lại và sự mất mát tính thích nghi được của thành phần.

Một trong những ưu việt chính của thừa kế trong thiết kế hướng đối tượng là các thành phần này có thể sẵn sàng thích nghi được. Cơ cấu thích nghi được này không dựa trên việc cải biên thành phần đó mà trên việc tạo ra một thành phần mới có thừa kế các thuộc tính và các phép toán của thành phần gốc. Khi các thuộc tính và phép toán được cải biên, các thành phần dựa trên thành phần cơ bản đó là không bị ảnh hưởng gì. Chỉ riêng tính thích nghi này là lý do duy nhất vì sao các ngôn ngữ hướng đối tượng là hữu hiệu đến vậy trong việc tạo mẫu nhanh.

2. Thiết kế hướng đối tượng (Object Oriented Design)

2.1 Cách tiếp cận hướng đối tượng

Bản chất duy nhất của thiết kế hướng đối tượng là khả năng xây dựng ba khái niệm thiết kế phần mềm quan trọng: trừu tượng, che dấu thông tin và modul.

Che dấu thông tin là chiến lược thiết kế dấu đi càng nhiều thông tin trong các thành phần càng hay. Có nghĩa là, việc kết hợp điều khiển logic và cấu trúc dữ liệu được thực hiện trong thiết kế càng chậm càng tốt, liên lạc thông qua các thông tin trạng thái dùng chung (biến tổng thể) là ít nhất, nhờ vậy khả năng hiểu được nâng lên.

Thiết kế hướng đối tượng là dựa trên việc che dấu thông tin, nhìn hệ phần mềm như một bộ các đối tượng tương tác với nhau chứ không phải là bộ các chức năng như cách tiếp cận hướng chức năng. Các đối tượng có một trạng thái được che dấu và các phép toán trong trạng thái đó. Thiết kế biểu thị các dịch vụ được yêu cầu cùng với những hỗ trợ cung cấp bởi các đối tượng có tương tác với nó.

2.2 Đặc trưng của thiết kế hướng đối tượng

1. Không có vùng dữ liệu dùng chung. Các đối tượng liên lạc với nhau bằng cách trao đổi thông báo chứ không phải các biến dùng chung.
2. Các đối tượng là các thực thể độc lập, dễ thay đổi vì rằng tất cả các trạng thái các thông tin biểu diễn chỉ ảnh hưởng trong phạm vi chính đối tượng đó thôi. Các thay đổi về biểu diễn thông tin có thể được thực hiện không cần có sự tham khảo tới các đối tượng hệ thống nào khác.
3. Các đối tượng có thể phân tán và có thể hành động tuần tự hoặc song song.

2.3 Các ưu nhược điểm của thiết kế hướng đối tượng

Ưu điểm

1. Dễ bảo trì vì các đối tượng là độc lập. Các đối tượng có thể hiểu và cải biên như là một thực thể độc lập. Thay đổi trong khi thực hiện một đối tượng hoặc thêm các dịch vụ sẽ không làm thay đổi hay ảnh hưởng tới các đối tượng hệ thống khác.
2. Các đối tượng là các thành phần dùng lại được. Một thiết có thể dùng lại các đối tượng đã được thiết kế trong các bản thiết kế trước đó.
3. Có các lớp hệ thống thể hiện quan hệ rõ ràng giữa các thực thể có thực (như các thành phần phần cứng) với các đối tượng điều khiển nó trong hệ thống. Điều này đạt được tính dễ hiểu trong thiết kế.

Nhược điểm

Việc minh định các đối tượng trong hệ thống là khó khăn. Cách nhìn tự nhiên là cách nhìn hướng chức năng và việc thích nghi với cách nhìn hướng đối tượng đôi khi là khó khăn.

2.4 Phân biệt giữa thiết kế hướng đối tượng và lập trình hướng đối tượng

Ta dễ nhầm lẫn hai khái niệm này. Ngôn ngữ lập trình hướng đối tượng là một ngôn ngữ lập trình cho phép thực hiện trực tiếp các đối tượng và cung cấp các lớp đối tượng và sự thừa kế.

Thiết kế hướng đối tượng là một chiến lược thiết kế, không phụ thuộc vào một ngôn ngữ cụ thể nào. Các ngôn ngữ lập trình hướng đối tượng và các khả năng bao gói đối tượng làm cho thiết kế hướng đối tượng được thực hiện một cách đơn giản hơn. Tuy nhiên một thiết kế hướng đối tượng cũng có thể thực hiện trong một ngôn ngữ kiểu như PASCAL hay C (không có đặc điểm bao gói như vậy).

Việc chấp nhận thiết kế hướng đối tượng như là một chiến lược hữu hiệu dẫn đến sự phát triển phương pháp thiết kế hướng đối tượng. ADA không phải là ngôn ngữ hướng đối tượng vì nó không trợ giúp sự thừa kế của các lớp, nhưng lại có thể thực hiện các đối tượng trong ADA bằng cách sử dụng các gói hoặc các nhiệm vụ, do đó ADA được dùng để thiết kế hướng đối tượng.

Phương pháp thiết kế hướng đối tượng vẫn còn là tương đối chưa chín muồi và đang có những thay đổi nhanh chóng. Phương pháp hiện đang sử dụng rộng rãi nhất là phương pháp dựa trên sự phân rã chức năng.

3. Thiết kế hướng cấu trúc

3.1 Cách tiếp cận hướng cấu trúc

Thiết kế hướng cấu trúc (hay thiết kế hướng chức năng) là một cách tiếp cận thiết kế phần mềm trong đó bản thiết kế được phân giải thành một bộ các đơn thể tác động lẫn nhau, mà mỗi đơn thể có một chức năng được xác định rõ ràng. Các chức năng có các trạng thái cục bộ

nhưng chúng chia sẻ với nhau trạng thái hệ thống - trạng thái tập trung, mọi chức năng đều có thể truy cập được.

Có người nghĩ rằng thiết kế hướng chức năng đã lỗi thời và nên được thay thế bởi cách tiếp cận hướng đối tượng. Thế nhưng, nhiều tổ chức đã phát triển các chuẩn và các phương pháp dựa trên độ phân giải chức năng. Nhiều phương pháp thiết kế kết hợp với công cụ CASE đều là hướng chức năng. Rất nhiều các hệ thống đã được phát triển bằng cách sử dụng phương pháp tiếp cận hướng chức năng. Các hệ thống đó vẫn sẽ được bảo trì cho tới một tương lai xa, bởi vậy thiết kế hướng chức năng vẫn sẽ còn được tiếp tục sử dụng rộng rãi.

Trong thiết kế hướng chức năng, người ta dùng các biểu đồ dòng dữ liệu (mô tả việc xử lý dữ liệu logic), các lược đồ cấu trúc (cấu trúc của phần mềm) và các mô tả PDL (mô tả thiết kế chi tiết). Khái niệm dòng dữ liệu đang hướng tới việc sử dụng một hệ thống vẽ biểu đồ tự động và sử dụng một dạng lược đồ cấu trúc có kèm theo các thông tin điều khiển.

Chiến lược thiết kế hướng chức năng dựa trên việc phân giải hệ thống thành bộ các chức năng có tương tác với trạng thái hệ thống tập trung – dùng chung cho các chức năng đó. Các chức năng này có thể có các thông tin trạng thái cục bộ nhưng chỉ dùng cho quá trình thực hiện các chức năng đó mà thôi.

Thiết kế chức năng gắn với các chi tiết một thuật toán của mỗi chức năng nhưng thông tin trạng thái hệ thống là không được che dấu. Điều này có thể gây ra vấn đề vì rằng một chức năng có thể thay đổi trạng thái theo một cách mà các chức năng khác không thể ngờ tới. Việc thay đổi một chức năng và cách nó sử dụng trạng thái của hệ thống có thể gây ra các tương tác bất ngờ đối với các chức năng khác.

Cách tiếp cận chức năng để thiết kế là tốt nhất khi mà khối lượng thông tin trạng thái hệ thống được làm nhỏ nhất và thông tin dùng chung nhau là rõ ràng.

3.2 Biểu đồ luồng dữ liệu

Bước thứ nhất của thiết kế hướng chức năng là phát triển một biểu đồ dòng dữ liệu hệ thống.

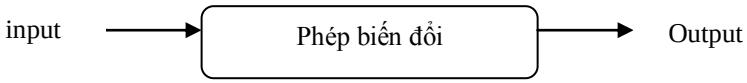
Biểu đồ dòng dữ liệu chỉ ra cách thức biến đổi dữ liệu *vào* thành dữ liệu *ra* thông qua một dãy các phép biến đổi.

Đó là một cách để mô tả hệ thống một cách dễ hiểu và không cần một sự huấn luyện đặc biệt nào. Biểu đồ này không nhất thiết bao gồm các thông tin điều khiển nhưng nên lập tư liệu các phép biến đổi dữ liệu.

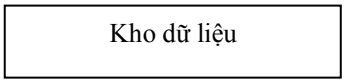
Biểu đồ dòng dữ liệu là một phần hợp nhất của một số các phương pháp thiết kế và các công cụ CASE, thường trợ giúp cho việc tạo ra biểu đồ dòng dữ liệu.

Các ký pháp trong biểu đồ dòng dữ liệu:

1. Hình chữ nhật góc tày: biểu diễn một phép biến đổi dòng dữ liệu *vào* thành dòng dữ liệu *ra,* tên của hình chữ nhật là tên của phép biến đổi đó.



2. Hình chữ nhật: biểu diễn một kho dữ liệu, tên của hình là tên của dữ liệu.



3. Hình tròn: biểu diễn giao tác người dùng với hệ thống (cung cấp thông tin vào hoặc hoặc thu nhận thông tin ra)

4. Các mũi tên: chỉ hướng dòng dữ liệu
5. Các từ khoá **and** và **or**
6. Khuyên tròn nối các dòng dữ liệu

3.3 Lược đồ cấu trúc

Lược đồ cấu trúc chỉ ra *cấu trúc* các thành phần theo thứ bậc của hệ thống. Nó chỉ ra rằng các phần tử của một biểu đồ dòng dữ liệu có thể được thực hiện như thế nào với tư cách là một thứ bậc của đơn vị chương trình.

Lược đồ cấu trúc có thể được dùng như là một mô tả chương trình nhìn thấy được và các thông tin xác định các lựa chọn và các vòng lặp, được dùng để trình bày một tổ chức tĩnh của thiết kế.

Mỗi thành phần chức năng được biểu diễn trên đồ thị có cấu trúc như là một hình chữ nhật. Thứ bậc này trình bày bằng cách nối các hình chữ nhật bởi các đường. Thông tin vào và thông tin ra cho một thành phần được chỉ bởi việc dùng các mũi tên có gán tên, gồm mũi tên vào và mũi tên ra. Các kho dữ liệu được chỉ bởi các hình chữ nhật góc tầy và các thông tin vào từ người dùng được chỉ bởi các khuyên tròn.

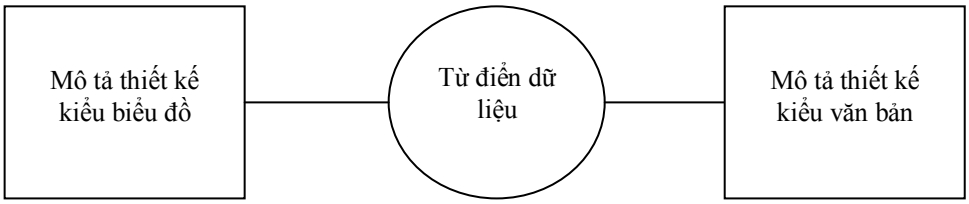
Sau này để tránh nhầm với ký pháp đã dùng trong biểu đồ dòng dữ liệu, người ta dùng các khối trụ để biểu diễn kho dữ liệu và hình bình hành biểu diễn thông tin vào.

3.4 Từ điển dữ liệu

Từ điển dữ liệu vừa có ích cho việc bảo trì hệ thống vừa có ích cho quá trình thiết kế. Với một lối vào đã được minh định trong biểu đồ phải có một từ điển dữ liệu cung cấp thông tin, thông tin về kiểu, chức năng của dữ liệu và một lý do cơ bản cho việc vào thông tin. Đôi khi người ta gọi cái này là một mô tả ngắn của chức năng thành phần.

Lối vào từ điển thành phần phải là một mô tả kiểu văn bản cho thành phần và phải được mô tả chi tiết.

Các từ điển dữ liệu dùng để nối các mô tả thiết kế kiểu biểu đồ và các mô tả thiết kế kiểu văn bản. Một vài công cụ CASE cung cấp một phép nối tự động biểu đồ dòng dữ liệu và từ điển dữ liệu.



4. Giao diện người sử dụng

4.1 Nhân tố con người và tương tác người máy

Thiết kế hệ thống máy tính bao gồm một loạt các hoạt động từ thiết kế phần cứng tới thiết kế giao diện người sử dụng. Trong đó, các kỹ sư điện tử thường chịu trách nhiệm về thiết kế phần cứng, còn các kỹ sư phần mềm phải chịu trách nhiệm thiết kế hệ thống và thiết kế giao diện người sử dụng. Các chuyên gia về nhân tố con người thường chỉ là cố vấn cho kỹ sư phần mềm chứ không trực tiếp thiết kế giao diện.

- Giao diện người dùng là cơ chế thiết lập giao tiếp giữa chương trình và người dùng. Nếu nhân tố con người được tính tới thì đối thoại sẽ trôi chảy và sẽ thiết lập được nhịp điệu giữa người dùng và chương trình. Nếu nhân tố người dùng bị bỏ qua thì hệ thống gần như bao giờ cũng bị coi như "*không thân thiện*".
- Giao diện sử dụng của một hệ thống thường được chọn làm tiêu chuẩn so sánh để đánh giá hệ thống theo quan điểm người sử dụng.

Một giao diện khó sử dụng sẽ ít nhiều gây ra sai lầm của người sử dụng. Trong trường hợp xấu nhất nó có thể làm cho hệ thống bị huỷ hoại bất chấp chức năng của nó.

Một giao diện thiết kế kém có thể làm cho người sử dụng gây ra lỗi. Nếu thông tin được biểu diễn lẫn lộn có thể làm người dùng hiểu nhầm ý nghĩa các khoản mục thông tin gây ra một chuỗi các hành vi nguy hiểm.

- Giao diện người sử dụng phải tính đến nhu cầu, kinh nghiệm và khả năng của người sử dụng.

Trong những ngày đầu của tin học (trước khi có những thiết bị hiển thị đồ hoạ như chuột, máy tính tốc độ cao…) một kiểu tương tác người-máy thực tế duy nhất là giao diện chỉ lệnh và hỏi (thế hệ 1). Trao đổi thuần tuý văn bản và được dẫn thông qua chỉ lệnh và các đáp ứng với các câu hỏi do hệ thống sinh ra. Người sử dụng có thể trao đổi với hệ thống bằng cách xác định các chỉ lệnh như:

>run progr1.exe/debug='on'/out=p1/in=1/alloc=1000k

* RUN ALLOCATION TO BE QUEUED? >> yes

*AUTOMATIC CHECKPOINTING INTERVALS?>>5

Như vậy mặc dầu chỉ lệnh và câu hỏi như thế rất chính xác nhưng cũng vẫn sinh lỗi và khó học, khó nhớ. Một cải tiến về giao diện chỉ lệnh và hỏi là *giao diện đơn* (menu) đơn giản (thế hệ 2). Tại đây, một danh sách các tùy chọn được nêu ra cho người dùng chọn thông qua một mã gõ vào nào đó.

Khi phần cúng trở nên tinh vi hơn và kỹ sư phần mềm học được nhiều hơn về nhân tố con người và tác động của chúng tới thiết kế giao diện thì giao diện trỏ và chọn hướng cửa sổ bắt đầu tiếu hóa (đôi khi còn được gọi là giao diện cửa sổ) gồm các biểu tượng, menu và thiết bị trỏ chuột (thế hệ 3)

Lợi ích của giao diện "thế hệ ba":

1. Có thể hiển thị đồng thời nhiều kiểu thông tin khác nhau, cho phép người dùng chuyển hoàn cảnh mà không mất mối nối trực quan với công việc khác. Cửa sổ cho phép người dùng thực hiện nhiều nhiệm vụ trao đổi và nhận biết mà không chán.

2. Nhiều nhiệm vụ tương tác khác có sẵn qua sơ đồ kéo xuống. Những đơn như vậy cho phép người dùng thực hiện các nhiệm vụ kiểm soát và đối thoại một cách dễ dàng.

3. Việc dùng biểu tượng đồ họa, đơn, kéo xuống, nút và kỹ thuật cuộn làm giảm khối lượng gõ. Điều này có thể làm tăng tính hiệu quả tương tác cho những người không phải là chuyên viên gõ, làm cho máy tính trở thành thân thiện với những người sợ bàn phím.

Thế hệ giao diện người máy (HCI - *Human Computer Interface*) hiện tại nối tất cả các thuộc tính của giao diện thế hệ ba với siêu văn bản (hypertext) và chế độ đa nhiệm - khả năng thực hiện một số nhiệm vụ khác nhau đồng thời (theo quan điểm của người dùng). Các giao diện "thế hệ bốn" này hiện nay đã có nhiều máy trạm làm việc (work station) và PC (Personal Computer).

4.2 Thiết kế giao diện người - máy

Thiết kế giao diện người - máy là một phần của chủ đề lớn là thiết kế phần mềm mà chúng ta đã biết. Toàn bộ tiến trình thiết kế giao diện người dùng bao gồm:

- Bắt đầu với việc tạo ra các mô hình khác nhau về chức năng hệ thống (như được cảm nhận từ bên ngoài)
- Phác họa ra các nhiêm vụ hướng con người và máy tính cần để đạt tới chức năng hệ thống
- Xem xét vấn đề thiết kế áp dụng cho mọi thiết kế giao diện
- Sử dụng các công cụ làm bản mẫu
- Cài đặt mô hình thiết kế và đánh giá kết quả và chất lượng

4.2.1. Mô hình thiết kế giao diện

Có 4 mô hình khác nhau khi thiết kế giao diện người - máy:

- Mô hình thiết kế: kỹ sư phần mềm tạo ra
- Mô hình người dùng: kỹ sư phần mềm/kỹ sư con người
- Mô hình của người dùng/cảm nhận hệ thống: người dùng cuối cùng xây dựng một hình ảnh tinh thần.
- Hình ảnh hệ thống: người cài đặt hệ thống tạo ra

Các mô hình này khác nhau. Vai trò của thiết kế giao diện là điều hòa những sự khác biệt này và đưa ra một cách biểu diễn nhất quán cho giao diện.

- Mô hình thiết kế: của toàn bộ hệ thống là tổ hợp các biểu diễn dữ liệu, kiến trúc và thủ tục của phần mềm.
- Mô hình người dùng: mô tả sơ lược hệ thống cho người dùng cuối. Để có hiệu quả, mọi thiết kế nên bắt đầu với một hiểu biết về người dùng được dự định, kể cả thông tin về tuổi tác, giới tính, khả năng thể chất, nền tảng giáo dục, văn hóa, động cơ, mục đích và nhân cách (có thể phân thành người mới học, người ít học khi dùng nhưng có hiểu biết, người dùng thường xuyên và có hiểu biết)
- Cảm nhận hệ thống là hình ảnh của hệ thống mà người dùng mang trong đầu
- Hình ảnh hệ thống tổ hợp cách biểu lộ bên ngoài của hệ thống dựa trên máy tính (nhìn và cảm thấy giao diện) với mọi thông tin hỗ trợ (sách, tài liệu sử dụng, băng video) mô tả cho cú pháp và ngữ nghĩa của hệ thống. Khi hình ảnh hệ thống cà cảm nhận hệ

thống trùng nhau thì nói chung người dùng cảm thấy thoải mái với phần mềm và dùng nó một cách hiệu quả.

Về bản chất, các mô hình này làm cho người thiết kế giao diện thỏa mãn với vấn đề mấu chốt của nguyên lý quan trọng nhất trong thiết kế giao diện người dùng: biết người dùng, biết nhiệm vụ

4.2.2. Phân tích và mô hình hóa nhiệm vụ trong thiết kế giao diện

Việc khởi thảo từng bước (cũng còn được gọi là phân tách chức năng hay làm mịn dần từng bước) như cơ chế để làm mịn các nhiệm vụ xử lý cần cho phần mềm hoàn thành một chức năng mong muốn nào đó.

Phân tích nhiệm vụ cũng có cách tiếp cận tương tự nhưng được áp dụng cho các hoạt động con người. Trước hết phải xác định và phân loại các nhiệm vụ.

Một khi từng nhiệm vụ hay hành động đã được xác định thì việc thiết kế giao diện bắt đầu. Bước đầu tiên trong tiến trình thiết kế giao diện có thể được thực hiện bằng cách tiếp cận sau:

1. Thiết lập các mục tiêu và ý đồ cho nhiệm vụ
2. Ánh xạ thành dẫy các hành động xác định
3. Xác định dẫy hành động khi nó được thực hiện ở mức giao diện
4. Chỉ ra trạng thái của hệ thống, tức là giao diện giống thế nào vào lúc hành động trong dẫy đó được thực hiện.
5. Xác định các cơ chế điều khiển, như thiết bị và hành động sẵn có cho người dùng để thay đổi trạng thái hệ thống
6. Chỉ ra cách thức cơ chế điều khiển này ảnh hưởng tới trạng thái hệ thống
7. Chỉ ra cách thức người dùng diễn giải trạng thái của hệ thống từ thông tin được cung cấp qua giao diện.

4.2.3. Các vấn đề trong thiết kế giao diện

Bốn vấn đề thiết kế thông thường gần như bao giờ cũng nổi lên:

1. Thời gian hệ thống đáp ứng (độ dài, độ biến thiên)
2. Tiện nghi giúp đỡ người dùng (tích hợp, phụ thêm)
3. Giải quyết thông tin lỗi (thông báo, cảnh bảo)
4. Gắn nhãn chỉ lệnh (tiện nghi tạo macro)

4.2.3.1 Thời gian hệ thống đáp ứng

Thời gian hệ thống đáp ứng là vấn đề nan giải đối với nhiều hệ thống tương tác. Nói chung, thời gian hệ thống đáp ứng được đo từ điểm tại đó người dùng thực hiện hành động điều khiển nào đó (như gõ phím hay nháy chuột) cho tới khi phần mềm đáp ứng cái ra hay hành động mong muốn.

Thời gian hệ thống đáp ứng có hai đặc trưng quan trọng: độ dài và độ biến thiên. Nếu độ dài thời gian đáp ứng quá lâu thì người dùng không tránh khỏi chán nản và căng thẳng. Tuy nhiên thời gian đáp ứng quá ngắn cũng có thể bất lợi nếu người dùng bị giao diện giữ nhịp. Sự đáp ứng nhanh chóng có thể buộc người dùng phải vội vã do đó phạm sai lầm.

"Độ biến thiên" nói tới độ lệch khỏi thời gian đáp ứng trung bình và theo nhiều cách thức thì nó còn quan trọng hơn đặc trưng thời gian đáp ứng. Độ biến thiên thấp làm cho người dùng thiết lập được nhịp điệu, cho dù thời gian đáp ứng tương đối lâu. Chẳng hạn đáp ứng 1 giây đối với một lệnh thì được ưa chuộng hơn cho một đáp ứng biến thiên từ 0,1 đến 0,25 giây. Trong

trường hợp sau, người dùng bao giờ cũng bị mất cân bằng, bao giờ cũng phải tự hỏi liệu có cái gì đó "khác" sẽ xuất hiện bên ngoài khung cảnh này hay không.

4.2.3.2 Tiện nghi giúp đỡ người dùng

Ta thường gặp phải hai kiểu tiện nghi trợ giúp khác nhau: tích hợp và phụ thêm. *Tiện nghi trợ giúp tích hợp* được thiết kế trong phần mềm ngay từ đầu. Nó thường cảm ngữ cảnh, làm cho người dùng lựa được từ các chủ đề có liên quan tới hành động hiện đang được thực hiện. Hiển nhiên, điều này rút gọn thời gian cần để người dùng thu được sự trợ giúp và tạo ra "sự thân thiện" của giao diện. *Tiện nghi trợ giúp phụ thêm* được thêm vào cho phần mềm sau khi hệ thống đã được xây dựng. Theo nhiều cách, nó thực sự là một tài liệu người dùng trực tuyến với một khả năng hỏi hạn chế. Người dùng có thể phải tìm kiếm trong một danh sách hàng trăm chủ đề để tìm hướng dẫn thích hợp, thường nhiều lần bắt đầu sai và nhận được những thông tin không liên quan. Chắc chắn là tiện nghi trợ giúp tích hợp được ưa thích hơn cách tiếp cận phụ thêm.

Cần phải đề cập tới một số vấn đề khi xem xét tiện nghi trợ giúp:

- Liệu trợ giúp có sẵn có với tất cả các chức năng hệ thống và vào mọi lúc trong tương tác hệ thống không? Các tùy chọn bao gồm: trợ giúp chỉ cho một tập con của mọi chức năng và hành động; trợ giúp cho tất cả các chức năng.
- Người dùng sẽ yêu cầu trợ giúp như thế nào? Các tùy chọn bao gồm: menu trợ giúp; phím trợ giúp đặc biệt; chỉ lệnh HELP.
- Trợ giúp sẽ được trình bày như thế nào? Các tùy chọn bao gồm: cửa sổ riêng biệt; tham khảo tới tài liệu in; gợi ý một hay hai dòng được tạo ra trong một vị trí màn hình cố định.
- Người dùng sẽ trở về với tương tác thông thường như thế nào? Các tùy chọn bao gồm: nút trở về được hiển thị trên màn hình; phím chức năng hay dẫy điều khiển.
- Thông tin trợ giúp sẽ được cấu trúc như thế nào? Các tùy chọn bao gồm:

- Cấu trúc "phẳng" trong đó mọi thông tin đều được thâm nhập tới qua một từ khóa;

- Cấp bậc phân tầng của thông tin cung cấp chi tiết ngày càng tăng khi người dùng tiến sâu vào cấu trúc;

- Sử dụng siêu văn bản.

4.2.3.3 Giải quyết thông tin lỗi

Thông báo lỗi và cảnh báo là tin dữ đối với người dùng khi một cái gì đó chạy không đúng. Tồi nhất thì thông báo lỗi và lời cảnh báo truyền đạt thông tin vô dụng hay hiểu sai và chỉ làm tăng thêm lỗi chán nản của người dùng. Có vài người dùng máy tính gặp phải lỗi có dạng

SERVER SYSTEM FAILURE - 14A

Ở đâu đó, nhất định phải có lời giải thích cho lỗi 14A; nếu không thì tại sao người thiết kế phải thêm vào thông tin định dạng? Quả thế, thông báo lỗi không đưa ra chỉ dẫn thực nào về cái gì sai hay phải lấy thông tin phụ ở đâu. Thông báo lỗi được trình bày theo cách được nêu ở trên chẳng có gì làm dịu bớt hay giúp sửa được vấn đề.

Nói chung, mọi thông báo lỗi hay cảnh báo do hệ thống tương tác tạo ra nên có các đặc trưng sau:

- Thông báo nên mô tả vấn đề theo lối nói mà người dùng có thể hiểu được.
- Thông báo nên đưa ra những lời khuyên có tính xây dựng để khôi phục từ lỗi.

- Thông báo nên chỉ ra bất kỳ hậu quả lỗi tiêu cực nào (như tệp dữ liệu có thể hỏng) để cho người dùng có thể kiểm tra đảm bảo rằng chúng không xuất hiện (hay sửa ngay chúng nếu có xuất hiện)
- Thông báo nên đi kèm với tín hiệu nghe được hay thấy được. Tức là một tiếng bíp có thể sinh ra đi kèm với việc hiển thị thông báo, hay thông báo có thể nhấp nháy chốc lát hay được hiển thị theo mầu dễ nhận ra như "mầu lỗi".
- Thông báo nên có tính chất "phi đánh giá". Tức là lời đưa ra đừng hàm ý trách móc người dùng.

4.2.3.4 Gắn nhãn chỉ lệnh

Ngày nay, việc dùng giao diện hướng cửa sổ, chỉ và chọn, đã làm giảm bớt việc dựa vào chỉ lệnh gõ vào, những nhiều siêu người dùng vẫn tiếp tục ưa thích mốt tương tác theo chỉ lệnh. Trong nhiều tình huống, người dùng có thể được cung cấp một tùy chọn - các chức năng phần mềm có thể được lựa ra từ một đơn tĩnh hay đơn kéo xuống thông qua một dãy chỉ lệnh bàn phím nào đó.

Một số vấn đề thiết kế nảy sinh khi chỉ lệnh được đưa ra như một mốt tương tác:

- Liệu mọi tùy chọn đơn có tương ứng với một chỉ lệnh không?
- Chỉ lệnh sẽ có dạng nào? Các tùy chọn bao gồm: dãy điều khiển (như ^P); phím chức năng; từ gõ vào.
- Việc học và nhớ chỉ lệnh sẽ khó đến mức nào? Có thể làm được gì nếu quên mất chỉ lệnh (xem thảo luận về trợ giúp được trình bày trong mục này)?
- Liệu chỉ lệnh có được người dùng làm cho phù hợp hay viết tắt không?

Trong số ứng dụng ngày càng tăng, người thiết kế giao diện đưa ra một tiện nghi tạo macro chỉ lệnh để cho phép người dùng ghi lại một dãy các chỉ lệnh hay dùng theo tên do người dùng xác định. Thay vì phải gõ từng lệnh một thì chỉ cần gõ macro chỉ lệnh và mọi chỉ lệnh trong đó sẽ được thực hiện tuần tự.

4.2.4. Công cụ cài đặt

Tiến trình thiết kế giao diện người dùng có tính lặp. Tức là một mô hình thiết kế được tạo ra, cài đặt như một bản mẫu, được người dùng xem xét (người thích hợp với mô hình người dùng đã mô tả dưới đây), và được thay đổi dựa trên lời góp ý của họ. Để thích nghi cách tiếp cận thiết kế lặp này, một lớp rộng các công cụ thiết kế giao diện và làm bản mẫu đã tiến hóa. Được gọi là bộ công cụ giao diện người dùng hay hệ thống phát triển giao diện người dùng (UIDS-*User Interface Development System*), các công cụ này đưa ra các modul hay đối tượng làm thuận tiện cho việc tạo ra cửa sổ, đơn, tương tác thiết bị, thông báo lỗi, chỉ lệnh và nhiều phần tử khác của môi trường tương tác.

Sử dụng phần mềm đóng gói sẵn có thể được người thiết kế và người cài đặt hay giao diện người dùng sử dụng trực tiếp, một UIDS cung cấp các cơ chế có sẵn cho những vấn đề sau:

- Việc quản lý thiết bị đưa vào (như chuột hay bàn phím)
- Làm hợp lệ cái vào của người dùng
- Giải quyết lỗi và hiển thị thông báo lỗi
- Cung cấp phản hồi (như tự động hiển thị cái vào)
- Cung cấp trợ giúp và lời nhắc
- Giải quyết cửa sổ và trường, cuộn bên trong cửa sổ

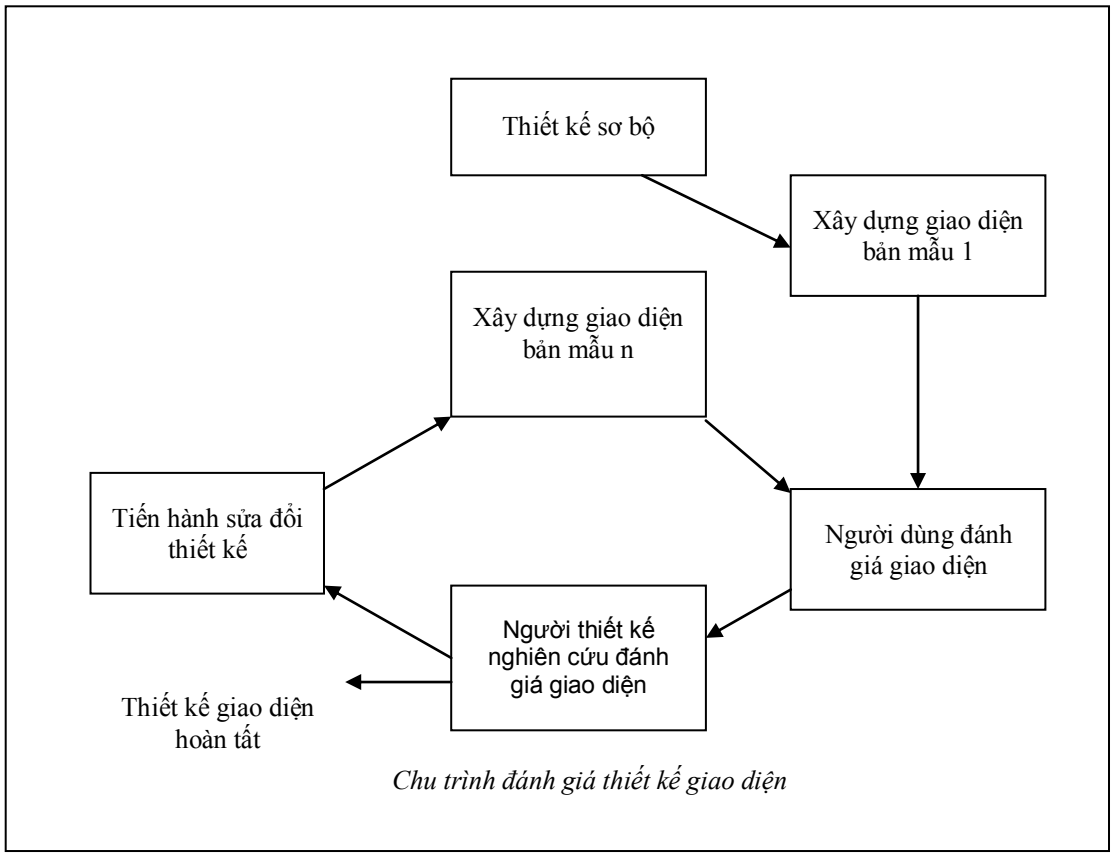
- Thiết lập mối nối giữa phần mềm ứng dụng và giao diện.
- Cô lập ứng dụng với các chức năng quản lý giao diện
- Cho phép người dùng làm phù hợp giao diện

Các chức năng được mô tả trên đây có thể được cài đặt bằng cách dùng hoặc cách tiếp cận dựa trên ngôn ngữ hoặc đồ họa.

4.2.5. Tiến hóa thiết kế

Một khi đã tạo ra bản mẫu giao diện người dùng vận hành được thì phải đánh giá đến việc xác định xem liệu nguyên mẫu có đáp ứng được nhu cầu của người dùng hay không. Việc đánh giá có thể mở rộng hay phổ biến từ "bộ kiểm thử" không hình thức trong đó người dùng cung cấp phản hồi cho tới các nghiên cứu được thiết kế hình thức dùng các phương pháp thống kê để tính toán các bảng hỏi do đa số người dùng điền vào.

Chu trình đánh giá giao diện người dùng có dạng như được vẽ trong hình dưới đây. Sau khi hoàn tất thiết kế sơ bộ, bản mẫu mức đầu sẽ được tạo ra. Người dùng sẽ đánh giá bản mẫu, cung cấp cho người thiết kế những ý kiến đóng góp trực tiếp về tính hiệu quả của giao diện. Bên cạnh đó, nếu có dùng các kỹ thuật đánh giá hình thức (như bảng hỏi), người thiết kế có thể trích thông tin từ những thông tin này. Những sửa đổi về thiết kế được thực hiện dựa trên cái vào của người dùng rồi bản mẫu mức tiếp được tạo ra. Chu trình đánh giá tiếp tục cho tới khi không còn cần tới việc sửa đổi thêm cho thiết kế giao diện nữa. Nhưng có thể đánh giá chất lượng của giao diện người dùng trước khi xây dựng bản mẫu được không? Nếu các vấn đề tiềm năng có thể được bộc lộ và sửa đổi sớm thì có thể rút gọn bớt một số chu trình và thời gian phát triển cho bản mẫu sẽ ngắn đi.



Chu trình đánh giá thiết kế giao diện

Nếu một mô hình thiết kế giao diện được tạo ra thì có thể áp dụng một số tiêu chuẩn đánh giá trong các cuộc họp xét duyệt thiết kế:

1. Độ dài và độ phức tạp của đặc tả viết về hệ thống và giao diện của nó cung cấp một chỉ dẫn về khối lượng học tập và người dùng hệ thống cần học.

2. Số các chỉ lệnh được xác định và số trung bình các đối số trên chỉ lệnh đưa ra một chỉ dẫn về thời gian tương tác và hiệu quả tổng thể của hệ thống

3. Số các hành động, chỉ lệnh và trạng thái hệ thống được mô hình thiết kế nêu ra cũng chỉ ra khối lượng cần nhớ của người dùng hệ thống

4. Phong cách tương tác, tiện nghi trợ giúp và giao thức xử lý lỗi đưa ra một chỉ dẫn chung về độ phức tạp của giao diện và mức độ người dùng sẽ chấp nhận được.

Một khi bản mẫu đầu tiên được xây dựng thì người thiết kế có thể thu thập rất nhiều dữ liệu định tính và định lượng sẽ trợ giúp cho việc đánh giá giao diện. Để thu thập dữ liệu định tính, có thể phân phát bằng bảng hỏi cho người dùng bản mẫu.

Câu hỏi có thể là:

(1) Trả lời đơn giản có/không

(2) Trả lời số

(3) Trả lời theo thang (chủ quan)

(4) Trả lời theo phần trăm (chủ quan)

Thí dụ:

1. Các chỉ lệnh có dễ nhớ không? Có/không

2. Bạn đã dùng bao nhiêu chỉ lệnh khác nhau?

3. Học cách vận hành hệ thống dễ đến mức nào? (thang 1 tới 5)

4. So sánh với các giao diện khác bạn đã dùng, giao diện này ở tỉ lệ nào? đỉnh 1%, đỉnh 10%, đỉnh 25%, đỉnh 50%, đáy 50%

Nếu đòi hỏi dữ liệu định lượng thì có thể tiến hành một dạng phân tích nghiên cứu thời gian. Quan sát người dùng trong khi tương tác và những dữ liệu như số nhiệm vụ được hoàn thành đúng trong một khảng thời gian chuẩn; tần số sử dụng chỉ lệnh; dẫy các chỉ lệnh; thời gian dành cho "việc nhìn" vào màn hình; số lỗi; kiểu lỗi và thời gian khôi phục lỗi; thời gian dùng trợ giúp; và số lần tham khảo trợ giúp trong khoảng thời gian chuẩn được thu thập và dùng như hướng dẫn cho việc sửa đổi giao diện.

4.3. Hướng dẫn thiết kế giao diện

Ba phạm trù của hướng dẫn thiết kế giao diện người máy HCI

(1) Tương tác chung

(2) Hiển thị thông tin

(3) Vào dữ liệu

4.3.1 Tương tác chung

Những hướng dẫn sau tập trung các yêu cầu tổng thể giao diện

Nhất quán: phải dùng định dạng nhất quán cho việc chọn đơn, vào chỉ lệnh, hiển thị dữ liệu và vô số các chức năng khác xuất hiện trong PC

Cho thông tin phản hồi có nghĩa: Cung cấp cho người sử dụng những thông tin phản hồi bằng hình ảnh và âm thanh nhằm thiết lập việc trao đổi thông tin hai chiều (giữa người sử dụng và giao diện)

Yêu cầu kiểm chứng mọi hành động phá hủy không tầm thường: Nếu người dùng yêu cầu xóa một tệp, ghi đè lên thông tin bản chất hay yêu cầu kết thúc chương trình thì một thông báo "Bạn có chắc ...?" nên xuất hiện ra.

Cho phép dễ dàng lần ngược hành động: Các chức năng UNDO (hoàn tác) hay REVERSE (đảo ngược) đã giúp cho hàng nghìn người dùng khỏi mất hàng nghìn giờ làm việc. Khả năng lần ngược lên có sẵn trong mọi ứng dụng tương tác.

Giảm thiểu khối lượng thông tin phải nhớ giữa các hành động: Không nên trông đợi người dùng cuối cùng nhớ được một danh sách các số hiệu hay tên gọi để cho người ấy có thể dùng lại chúng trong những chức năng kế toán sau. Cần phải tối thiểu tải trọng ghi nhớ.

Tìm kiếm tính hiệu quả trong đối thoại, vận động và ý nghĩ. Nên tối thiểu dùng các phím, cần phải xem xét khoảng cách chuột phải đi qua giữa các điểm trong thiết kế bố trí màn hình, và đừng đẩy người dùng vào tình huống phải tự hỏi "Cái này nghĩa là gì nhỉ?"

Dung thứ cho sai lầm: Hệ thống nên tự bảo vệ khỏi lỗi của người dùng để khỏi bị chết hỏng.

Phân loại các hoạt động theo chức năng và tổ chức màn hình hài hòa theo vùng. Một trong những cái lợi chính của đơn kéo xuống là khả năng tổ chức các lệnh theo kiểu. Về bản chất người thiết kế nên cố gắng đặt các chỉ lệnh và hành động "nhất quán"

Cung cấp tiện nghi trợ giúp cảm ngữ cảnh

Dùng các động từ đơn giản hay cụm động từ ngắn để đặt tên chỉ lệnh: Tên chỉ lệnh dài dòng thì khó nhận dạng và nhớ. Nó cũng có thể chiếm không gian không cần thiết trong danh sách đơn.

4.3.2 Hiển thị thông tin

Thông tin được "hiển thị" theo nhiều cách khác nhau: với văn bản, tranh ảnh và âm thanh bằng cách sắp đặt, di chuyển và kích cỡ; dùng mầu sắc, độ phân giải; và thậm chí bằng cả việc bỏ lửng. Các hướng dẫn sau đây tập trung vào hiển thị thông tin:

Chỉ hiển thị thông tin có liên quan tới hiện tại: Người dùng không phải khó nhọc lần qua dữ liệu, đơn, và hiệu ứng đồ họa để thu được thông tin liên quan tới một chức năng hệ thống riêng.

Đừng chôn vùi người dùng dưới dữ liệu - hãy dùng định dạng trình bày cho phép hấp thụ nhanh chóng thông tin: Đồ họa hoặc sơ đồ nên thay thế cho các bảng lớn.

Dùng nhãn nhất quán, cách viết tắt chuẩn và mầu sắc dự kiến trước được. Ý nghĩa của hiển thị hiển nhiên không cần tới tham khảo thêm nguồn thông tin bên ngoài.

Cho phép người dùng duy trì ngữ cảnh trực quan: Nếu việc hiển thị đồ họa máy tính được thay đổi tỷ lệ thì hình ảnh gốc nên được hiển thị thường xuyên (dưới dạng rút gọn tại góc màn hình) để cho người dùng hiểu được vị trí tương đối của phần hình ảnh hiện đang được xét.

Đưa ra những thông bảo lỗi có nghĩa

Dùng chữ hoa chữ thường, tụt lề và gộp nhóm văn bản để trợ giúp cho việc hiểu. Nhiều thông tin được HCI truyền đạt là văn bản, ngay cả cách bố trí và hình dạng của văn bản cũng có tác động đáng kể đến sự thoải mái để người dùng hấp thu thông tin.

Sử dụng cửa sổ (nếu sẵn có) để đóng khung các kiểu thông tin khác nhau: Cửa sổ cho phép người dùng "giữ" nhiều kiểu thông tin trong phạm vi truy nhập tới dễ dàng.

Dùng cách hiển thị tương tự để biểu diễn những thông tin dễ được hấp thu hơn với dạng biểu diễn này. Chẳng hạn, hiển thị áp suất của bể chứa trong xưởng lọc dầu sẽ ít tác dụng nếu dùng cách biểu diễn số. Tuy nhiên, nếu hiển thị dạng nhiệt kế dùng để chuyển động theo chiều

đứng và sự thay đổi màu sắc có thể dùng để chỉ ra những điều kiện áp suất thay đổi. Điều này sẽ cung cấp cho người dùng cả thông tin tuyệt đối và tương đối.

Xem xét vùng hiển thị có sẵn trên màn hình và dùng nó một cách có hiệu quả. Khi dùng nhiều cửa sổ, ít nhất nên có sẵn không gian để chỉ ra một phần cho từng cửa sổ này. Bên cạnh đó, kích cỡ màn hình (vấn đề công nghệ hệ thống) nên được đưa lựa chọn để hòa hợp với kiểu ứng dụng cần được cài đặt.

4.3.3 Vào dữ liệu

Những hướng dẫn sau đây tập trung vào việc đưa vào dữ liệu:

Tối thiểu số hành động đưa vào mà người dùng cần thực hiện. Việc rút gọn số liệu gõ vào là điều yêu cầu trước hết. Điều này có thể được thực hiện bằng cách dùng chuột để chọn từ một tập đã xác định sẵn các cái vào; dùng "thanh trượt" để xác định cái vào trong một miền giá trị; dùng "macro" làm cho chỉ một phím được chuyển thành một tập dữ liệu với miền cái vào phức tạp hơn.

Duy trì sự nhất quán giữa hiển thị thông tin và cái vào dữ liệu: Các kí tự hiển thị trực quan (như kích cỡ văn bản, mầu sắc, cách bố trí) nên được thực hiện đối với miền cái vào

Cho phép người dùng làm phù hợp cái vào: Người dùng chuyên gia có thể quyết định tạo ra các chỉ lệnh đã sửa đổi phù hợp mình hay bỏ qua một số kiểu cảnh báo và kiểm chứng hành động HCI nên cho phép điều này.

Tương tác nên mềm dẻo nhưng cũng nên hòa hợp với mốt đưa vào ưa thích. Mô hình người dùng sẽ trợ giúp cho việc xác định mốt đưa vào nào là ưa thích. Một thư ký có thể rất thích với cách đưa vào từ bàn phím, trong khi người quản lý lại có thể thấy thoải mái khi dùng thiết bị trỏ và nháy như chuột.

Khử kích hoạt các chỉ lệnh không thích hợp trong hoàn cảnh của hành động hiện tại. Điều này bảo vệ cho người dùng khỏi phải cố dùng một hành động nào đó có thể làm phát sinh lỗi.

Để cho người dùng kiểm soát luồng tương tác. Người dùng nên có khả năng nhảy qua các hành động không cần thiết, thay đổi trật tự các hành động yêu cầu (khi có thể được trong hoàn cảnh của ứng dụng), và khôi phục được từ các điều kiện lỗi mà không phải ra khỏi chương trình.

Cung cấp trợ giúp cho mọi hành động đưa vào, hỗ trợ tối đa. Đừng yêu cầu người dùng phải gõ nhiều. VD: phải gõ .00 cho toàn bộ số tiền đưa vào, nên đưa ra các giá trị mặc định mọi lúc có thể và không bao giờ yêu cầu người dùng đưa vào những thông tin có thể tự động thu thập hay tính toán được bên trong chương trình.

4.4 Chuẩn giao diện

Phần lớn những người xây dựng ứng dụng phần mềm hiện đại đều cài đặt các giao diện trỏ và hướng cửa sổ. Việc tạo ra những giao diện như vậy không phải là dễ dàng. Việc chuẩn hoá giao diện sẽ làm lợi cho người phát triển và người dùng HCI.

Người phát triển có thể dùng lại các module và đối tượng HCI đã có (hoặc đã đóng gói), do đó có thể tạo ra giao diện nhanh chóng hơn và có chất lượng cao hơn đáng kể.

Người dùng trở nên quen thuộc với cách bố trí và nhịp điệu của HCI và do đó có thể học bất kỳ ứng dụng mới nào có dùng chuẩn giao diện nhanh chóng hơn nhiều. Sau một thời gian việc sử dụng giao diện trở thành trực giác, do đó hiệu suất hơn nhiều đối với người dùng cuối cùng.

Mặc dầu một số chuẩn giao diện cạnh tranh đang được phát triển, chuẩn được dùng phổ biến nhất là X-Windows (nhiều cửa sổ). Hệ thống X-Windows xác định một cú pháp và ngữ

nghĩa cho thiết kế HCI và cung cấp các công cụ để tạo ra hiển thị, cửa sổ và đồ hoạ cũng như những giao thức để sử lý tài nguyên, tương tác thiết bị và giải quyết sự kiện. Một số biến thể và mở rộng của chuẩn hệ thống X-Windows đã được phát triển và dùng trên PC và máy trạm làm việc dưới UNIX và các hệ điều hành khác.

5. Tài liệu thiết kế phần mềm

Dàn bài tài liệu thiết kế được nêu trong bảng dưới đây có thể được dùng như một mô hình cho đặc tả thiết kế. Mỗi mục đều đề cập tới các khía cạnh khác nhau của biểu diễn thiết kế.

Dàn bài tài liệu trình bày một mô tả thiết kế đầy đủ về phần mềm. Các mục của đặc tả thiết kế được hoàn chỉnh khi người thiết kế làm mịn việc trình bày của mình về phần mềm

Dàn bài đặc tả thiết kế
I. Phạm vi A. Mục tiêu hệ thống B. Phần cứng, phần mềm và giao diện con người C. Các chức năng phần mềm chính D. CSDL được xác định bên ngoài E. Các ràng buộc, giới hạn thiết kế chính
II. Tài liệu tham khảo A. Tài liệu về phần mềm hiện có B. Tài liệu hệ thống C. Tài liệu người cung cấp (phần cứng, phần mềm) D. Tham khảo kỹ thuật
III. Mô tả thiết kế A. Mô tả dữ liệu 1. Tổng quan về luồng dữ liệu 2. Tổng quan về cấu trúc dữ liệu B. Cấu trúc chương trình suy dẫn C. Giao diện bên trong cấu trúc
IV. Modul (cho mỗi modul) A. Lời thuật xử lý B. Mô tả giao diện C. Mô tả ngôn ngữ thiết kế (hay những mô tả khác) D. Các modul đã dùng E. Tổ chức dữ liệu F. Bình luận

V. Cấu trúc tệp và dữ liệu toàn cục A. Cấu trúc tệp ngoài 1. Cấu trúc logic 2. Mô tả bản ghi logic 3. Phương pháp thâm nhập B. Dữ liệu toàn cục C. Tham khảo chéo tệp và dữ liệu
VI. Tham khảo chéo yêu cầu
VII. Điều khoản kiểm thử A. Hướng dẫn kiểm thử B. Chiến dịch kiểm thử C. Xem xét đặc biệt
VIII. Đóng gói A. Các điều khoản đặc biệt B. Xem xét chuyển đổi
IX. Lưu ý đặc biệt
X. Phụ lục

Mục I mô tả phạm vi toàn cục của thiết kế, nhiều thông tin được suy ra từ đặc tả hệ thống và các tài liệu khác trong pha xác định phần mềm. Các tham khảo riêng tới tài liệu hỗ trợ được thực hiện trong mục II. Mục III, mô tả thiết kế được hoàn tất như một phần của thiết kế sơ bộ. Chúng ta đã lưu ý rằng thiết kế là hướng thông tin (luồng dữ liệu và/hoặc cấu trúc dữ liệu) sẽ khống chế kiến trúc của phần mềm. Biểu đồ luồng dữ liệu hay các biểu diễn dữ liệu khác được phát triển, trong khi, các phân tích yêu cầu được làm mịn và được dùng để điều khiển cấu trúc phần mềm. Bởi vì luồng thông tin đã có sẵn nên mô tả giao diện có thể được phát triển cho các phần tử của phần mềm.

Các mục IV và V được phát triển khi thiết kế sơ bộ chuyển thành thiết kế chi tiết. Các modul hay các phần tử định địa chỉ tách biệt được của phần mềm (chương trình con, hàm thủ tục) khởi đầu được mô tả bằng lời thuật xử lý trong tiếng Anh. Lời thuật xử lý giải thích chức năng thủ tục của một modul. Sau này, một công cụ thiết kế thủ tục sẽ được dùng để dịch lời thuật thành một mô tả có cấu trúc. Mô tả cách tổ chức dữ liệu được nêu trong mục V. Các cấu trúc tệp duy trì trên bộ nhớ phụ được mô tả trong thiết kế sơ bộ; dữ liệu toàn cục được gán; tham khảo chéo gắn các moule riêng lẻ với dữ liệu toàn cục được thiết lập.

Mục VI của đặc tả thiết kế chứa các tham khảo chéo về các yêu cầu. Mục đích của ma trận tham khảo chéo này là:

1. Thiết lập mọi yêu cầu được phần mềm thoả mãn.
2. Chỉ ra module nào là chủ chốt cho việc cài đặt các yêu cầu xác định.

Mục yêu cầu	Module A	Module B	Module C	…	Module Z
Mục 3.1.1	v				v

Mục 3.1.2		v	v		
Mục 3.1.3		v			
Mục 3.1.n			v		v

Tham khảo chéo yêu cầu

Giai đoạn đầu tiên trong việc phát triển các kiểm thử được đưa vào trong mục VII của tài liệu thiết kế. Một khi cấu trúc và giao diện phần mềm đã được thiết lập thì chúng ta có thể phát triển các hướng dẫn để kiểm thử từng module riêng, rồi kiểm thử tích hợp toàn bộ hệ thống.

Trong một số trường hợp, một đặc tả chi tiết về thủ tục kỹ nghệ phần mềm sẽ xuất hiện song song với thiết kế. Trong những trường hợp như vậy, mục này có thể xoá đi khỏi bản thiết kế.

Các ràng buộc thiết kế, như những giới hạn bộ nhớ vật lý, hay sự cần thiết một giao diện ngoài chuyên dụng có thể khống chế các yêu cầu riêng để lắp ráp hay đóng gói phần mềm. Những xem xét đặc biệt cần thiết cho sự chồng chất chương trình, quản lý bộ nhớ ảo, xử lý tốc độ cao, hay các nhân tố khác, có thể gây ra sự thay đổi trong thiết kế suy ra từ luồng cấu trúc thông tin.

Các yêu cầu và xem xét cho việc đóng gói phần mềm được trình bày trong mục VIII, mô tả cho các tiếp cận sẽ được dùng để chuyển phần mềm cho khách hàng.

Mục IX và X của bản đặc tả thiết kế chứa các dữ liệu phụ trợ. Các mô tả thuật toán, thủ tục khác, dữ liệu bảng, các trích đoạn từ các văn bản khác và các thông tin liên quan được trình bày như những lưu ý đặc biệt hoặc như những phụ lục tách biệt. Chúng ta nên phát triển *Bản Hướng dẫn sơ bộ* hay *Tài liệu cài đặt* và đưa nó vào như phụ lục cho tài liệu thiết kế.

Tóm tắt

Thiết kế là cái lõi của kỹ thuật kỹ nghệ phần mềm. Trong khi thiết kế người ta sẽ phát triển, xét duyệt và làm tư liệu cho việc làm mịn dần các chi tiết thủ tục, cấu trúc chương trình, cấu trúc dữ liệu. Việc thiết kế nảy sinh trong việc biểu diễn cho phần mềm và chất lượng phần mềm có thể được xác nhận.

Trong suốt ba thập kỷ qua, người ta đã đề nghị một số khái niệm thiết kế phần mềm nền tảng. Tính module (trong cả chương trình và dữ liệu) và khái niệm trừu tượng làm cho người thiết kế có khả năng đơn giản hoá và dùng lại các thành phần phần mềm. Việc làm mịn đưa ra một cơ chế để biểu diễn các tầng kế tiếp của chi tiết chức năng. Cấu trúc chương trình và dữ liệu đóng góp cho một quan điểm tổng thể về kiến trúc phần mềm, trong khi thủ tục lại đưa ra những chi tiết cần thiết cho việc cài đặt thuật toán. Che dấu thông tin và độc lập chức năng để đạt tới tính modue hiệu quả.

Thiết kế phần mềm có thể được xem xét hoặc theo cách nhìn kỹ thuật hoặc theo cách nhìn quản lý dự án. Theo quan điểm kỹ thuật, thiết kế bao gồm 4 hoạt động: thiết kế dữ liệu, thiết kế kiến trúc, thiết kế thủ tục và thiết kế giao diện. Theo quan điểm quản lý, thiết kế tiến hoá từ thiết kế sơ bộ sang thiết kế chi tiết.

Ký pháp thiết kế, đi kèm với các khái niệm lập trình có cấu trúc làm cho người thiết kế biểu diễn được chi tiết thủ tục theo cách thức làm thuận tiện cho việc dịch sang mã chương trình. Các ký pháp đồ hoạ, bảng và văn bản đều có sẵn.

Còn nhiều phương pháp thiết kế phần mềm quan trọng như thiết kế hướng luồng dữ liệu, hướng sự vật…Những phương pháp này, được kết hợp với những nền tảng đã trình bày ở trên tạo nên một cách nhìn đầy đủ về thiết kế phần mềm.

Củng cố

1.Tầm quan trọng của thiết kế phần mềm ? Các giai đoạn phải trải qua?

2. Chuẩn bị cho lập trình, những loại thiết kế nào cần tạo ra trong giai đoạn thiết kế? Vẽ sơ đồ hoạt động thiết kế và sản phẩm thiết kế? Mô tả các hoạt động cốt yếu trong đó.

3. Vẽ và phân tích sơ đồ mô tả mối quan hệ giữa các khía cạnh quản lý và kỹ thuật?

4. Tư tưởng của phương pháp cấu trúc?

5. Các hướng dẫn đảm bảo chất lượng thiết kế?

6. Các khái niệm nền tảng cho thiết kế (trừu tượng, làm mịn, module, kiến trúc phần mềm, cấp bậc điều khiển, cấu trúc dữ liệu, thủ tục phần mềm, che dấu thông tin)

7. Tóm lược hai chiến lược thiết kế chức năng và đối tượng?

8. Độ đo chất lượng thiết kế?

9. Bàn về thiết kế hướng đối tượng (cách tiếp cận, đặc trưng ưu nhược điểm)?

10. Khái niệm biểu đồ dòng dữ liệu, lược đồ cấu trúc, từ điển dữ liệu trong thiết kế hướng chức năng?

11. Các thế hệ giao diện người - máy? Tiến trình thiết kế giao diện?

12. Các Mô hình thiết kế giao diện?

13. Các vấn đề thiết kế giao diện thường nảy sinh?

14. Chu trình đánh giá thiết kế giao diện? Tiêu chuẩn đánh giá xét duyệt thiết kế?

15. Tóm lược các hướng dẫn thiết kế giao diện?

CHƯƠNG 4

ĐẢM BẢO, KIỂM THỬ VÀ BẢO TRÌ PHẦN MỀM

Chương này nhấn mạnh vào các hoạt động được áp dụng để đảm bảo chất lượng sản phẩm phần mềm trong suốt tiến trình kỹ nghệ phần mềm.

1. Đảm bảo chất lượng phần mềm

Tất cả các phương pháp, công cụ và thủ tục được mô tả từ đầu đến giờ đều hướng tới mục tiêu duy nhất là: sản xuất ra phần mềm chất lượng cao. Vậy ***chất lượng phần mềm*** là gì?

Trong phần này chúng ta sẽ xem xét ý nghĩa của thuật ngữ: "chất lượng phần mềm" cũng như các thủ tục và biện pháp giúp đảm bảo rằng chất lượng là kết quả tự nhiên của kỹ nghệ phần mềm.

1.1 Các nhân tố chất lượng phần mềm

Các nhân tố chất lượng phần mềm được coi như những cái *mốc* để đánh giá chất lượng phần mềm. Các nhân tố chất lượng phần mềm được mô tả sau đây hội tụ vào ba khía cạnh quan trọng của sản phẩm phần mềm là: các đặc trưng vận hành, khả năng trải qua thay đổi, và tính thích nghi với môi trường mới. Một sản phẩm phần mềm cần được xét theo các tiêu chuẩn dưới đây.

Tính đúng đắn: Phần mềm thực hiện chính xác những mục tiêu và chức năng đã đề xuất ở giai đoạn thiết kế. Tính đúng đắn của phần mềm được xác minh qua những căn cứ sau:

1. Tính đúng đắn của thuật toán
2. Tính tương đương của chương trình với thuật toán. Thuật toán có thể đúng nhưng chương trình lập ra không tương đương với thuật toán nên dẫn đến kết quả sai. Trong trường hợp này người ta nói rằng việc mã hoá bị sai
3. Tính đúng đắn của chương trình có thể được chứng minh trực tiếp trong văn bản của chương trình
4. Tính đúng đắn của chương trình có thể được khẳng định dần dần qua việc kiểm thử (testing), qua việc áp dụng chương trình trong một khoảng thời gian đủ lớn trên một diện khá rộng với tần suất sử dụng cao.
5. Các tác giả của phần mềm cần cung cấp đầy đủ những luận chứng và kết quả xác nhận tính đúng đắn của sản phẩm. Những tài liệu đó bao gồm: Chương trình có kèm theo các luận đề phục vụ cho việc chứng minh, các phương pháp và kỹ thuật kiểm thử chương trình, các kết quả chạy thử, các giấy chứng nhận hay nhận xét của cá nhân đơn vị đã khảo sát và áp dụng chương trình.

Tính khoa học: Tính khoa học của sản phẩm được thể hiện qua những mặt sau

1. Khoa học về cấu trúc: Bản thân phần mềm được chia thành những đơn vị cân đối, không trùng lặp nhau về chức năng, có quan hệ hữu cơ với nhau, có thể tổ hợp thành nhiều chức năng mới. Bản thân thuật toán và chương trình được thiết kế và cài đặt một cách có cấu trúc.
2. Khoa học về nội dung: Các thuật toán dựa trên thành tựu mới của toán học và tin học phải có cơ sở chặt chẽ.

3. Khoa học về hình thức thao tác: Tên của các lệnh phải hợp lý, thể hiện tính logic và phù hợp với tư duy tự nhiên của người dùng. Ví dụ: một hệ thống biểu thị cả tiếng Anh tiếng Việt lẫn lộn sẽ được xem là vi phạm tính khoa học về hình thức. Trong trường hợp như vậy cách giải quyết tốt nhất là là thiết kế hai chế độ giao tiếp một bằng tiếng Anh một bằng tiếng Việt. Ví dụ: Các thông báo lỗi phải rõ ràng và bao gồm các chú giải sau: lỗi số mấy, nội dung lỗi, nơi xảy ra lỗi, cách giải quyết.

Tính hữu hiệu: Tính hữu hiệu của một sản phẩm được xác định qua những tiêu chuẩn sau:

1. Hiệu quả kinh tế, ý nghĩa hoặc giá trị thu được do áp dụng sản phẩm đó
2. Tốc độ xử lý của sản phẩm (v) tính bằng tỉ lệ giữa khối lượng đối tượng xử lý (m) và tổng số đơn vị thời gian cần thiết để xử lý khối lượng trên (t): v=m/t
3. Giới hạn tối đa của sản phẩm hoặc miền xác định của chương trình được xác định qua khối lượng tối đa của các đối tượng mà sản phẩm đó quản lý.
4. Dung lượng bộ nhớ mà tối đa trong RAM mà chương trình sử dụng.

Tính sáng tạo: Tính sáng tạo của sản phẩm phần mềm thể hiện qua các đặc điểm sau

1. Sản phẩm đó được thiết kế và cài đặt lần đầu tiên trên thế giới
2. Sản phẩm được sản xuất phục vụ cho những đặc thù riêng của Việt Nam. Ví dụ: bộ xử lý văn bản tiếng Việt, bộ chương trình nhận dạng các loại men gốm, hoa văn trang trí…
3. Sản phẩm phần mềm có những điểm khác về mặt nguyên lý so với các sản phẩm hiện hành.
4. Sản phẩm đó có những ưu thế nổi bật so với các sản phẩm cùng loại hiện hành.

Tính an toàn: Sản phẩm trong trường hợp cần thiết có cơ chế bảo mật và bảo vệ các đối tượng do nó phát sinh hay quản lý. Bản thân sản phẩm cũng cần được đặt trong một cơ chế bảo mật nhằm chống sự sao chép trộm hoặc làm biến dạng sản phẩm đó.

Tính toàn vẹn: Tính toàn vẹn của sản phẩm thể hiện qua những chức năng sau

1. Không gây ra nhập nhằng trong thao tác, đảm bảo tính nhất quán về cú pháp, có cơ chế ngăn ngừa việc phát sinh ra những đối tượng sai quy cách hoặc mâu thuẫn với các đối tượng có sẵn (dữ liệu, đơn thể…).
2. Có cơ chế khôi phục lại toàn bộ hoặc một phần những đối tượng thuộc diện quản lý của sản phẩm trong trường hợp có sự cố như hỏng máy, mất điện đột ngột…

Tính đầy đủ: Tính đầy đủ của một sản phẩm được đánh giá thông qua tập các chức năng mà sản phẩm đó cung cấp. Tập các chức năng cần và nên thoả mãn các tính chất sau:

1. Tính đối xứng: Nếu có thao tác phát sinh thì nên có thao tác huỷ bỏ và ngược lại. Ví dụ: Một hệ thống quản lý các bảng điện tử đã có thao tác thêm một số cột vào bảng hiện có thì cần có thêm thao tác đối xứng là xoá một số cột từ bảng hiện có.
2. Tính tiêu chuẩn: Sản phẩm phần mềm cần đạt được một số tiêu chuẩn tối thiểu được thừa nhận trên thị trường thế giới hoặc trong khoa học. Ví dụ: một hệ quản lý tệp cần có các chức năng tối thiểu sau: Tạo các tệp, Huỷ bỏ các tệp, Sao chép các tệp, Tổ chức thư mục cho các tệp, Cập nhật dữ liệu vào tệp, Quản lý được nhiều loại tệp (tệp văn bản, tệp chương trình, tệp dữ liệu…), Các lệnh thao tác tệp có thể hoạt động đơn lẻ nhưng cũng có thể nhúng vào các ngôn ngữ lập trình cao cấp…

Tính độc lập: Sản phẩm phần mềm cần và nên đảm bảo tính độc lập với các đối tượng sau:

1. Độc lập đối với thiết bị. Sản phẩm có thể cài đặt dễ dàng trên nhiều loại máy và có thể quản lý được nhiều thiết bị đi kèm với máy. Những sửa đổi để thích nghi là không đáng kể.
2. Độc lập với cấu trúc của đối tượng mà sản phẩm đó quản lý. Ví dụ: Một hệ quản trị CSDL có tính độc lập cao sẽ không đòi hỏi người dùng phải viết lại chương trình ứng dụng khi chuyển từ việc quản lý các tệp tuần tự sang các tệp tuần chỉ và ngược lại.
3. Độc lập với nội dung của đối tượng mà sản phẩm đó quản lý. Ví dụ: đơn thể COPPY có thể sao chép các tệp không phục thuộc gì vào nội dung cụ thể của các tệp đó ra sao (rỗng hay không rỗng, lưu văn bản hay chương trình, lưu mã nhị phân hay mã trung gian…).

Tính phổ dụng: Sản phẩm phần mềm có thể áp dụng cho nhiều lĩnh vực cho nhiều chế độ làm việc khác nhau.

Tính đơn giản: Sản phẩm có tính đến những yếu tố tâm lý sau đây của đông đảo người dùng

- Dễ thao tác
- Dễ học và dễ hoàn thiện
- Ngôn ngữ trong sáng dễ hiểu, dễ nhớ (tên lệnh, thực đơn, thông báo, cú pháp…)

Tính dễ phát triển, hoàn thiện: Sản phẩm có thể mở rộng, tăng cường về mặt chức năng một cách dễ dàng.

1.2 Độ đo chất lượng phần mềm

Phần trên chúng ta đã thảo luận về một tập các nhân tố chất lượng cho việc đánh giá hay "việc đo" chất lượng phần mềm. Trong mục này chúng ta xem xét một tập các độ đo phần mềm có thể được dùng cho việc thẩm định mang tính định lượng về chất lượng phần mềm. Trong mọi trường hợp các độ đo đều biểu diễn cho cách đo gián tiếp tức là chúng ta chưa bao giờ đo được chất lượng mà chỉ đo được biểu lộ nào đó của chất lượng.

1.2.1 Chỉ số chất lượng phần mềm

Bộ chỉ huy lực lượng không quân Mỹ đã phát triển một số các chỉ báo chất lượng chất lượng phần mềm dựa trên các đặc trưng thiết kế đo được về chương trình máy tính. Dùng các khái niệm tương tự như những khái niệm được đề nghị trong chuẩn 982.1(1998), lực lượng không quân dùng thông tin thu được từ thiết kế dữ liệu và kiến trúc để suy ra chỉ số chất lượng cấu trúc thiết kế (DSQI) chạy từ 0 đến 1. Các giá trị sau đây phải được tìm hiểu chắc chắn để tính DSQI:

S_1 = tổng số các modul (được định nghĩa trong kiến trúc chương trình)

S_2 = số các modul có chức năng tạo ra dữ liệu hay phụ thuộc vào nguồn dữ liệu (không tính các modul điều khiển)

S_3 = số các modul có chức năng phụ thuộc trước hết vào xử lý

S_4 = số các khoản mục cơ sở dữ liệu

S_5 = tổng số các khoản mục cơ sở dữ liệu độc lập

S_6 = số các đoạn trong cơ sở dữ liệu (các bản ghi khác nhau hay các sự vật riêng lẻ)

S_7 = số các modul với một dữ liệu vào một dữ liệu ra

Tiếp theo là tính các giá trị chung gian sau:

Cấu trúc chương trình D_1: Nếu thiết kế kiến trúc được phát triển bằng cách dùng một phương pháp phân biệt (Ví dụ: thiết kế hướng luồng dữ liệu, thiết kế hướng sự vật…) thì $D_1 = 1$; ngoài ra $D_1 = 0$

Tính độc lập modul D_2: $D_2 = 1 – (S_2 / S_1)$

Modul không phụ thuộc vào xử lý trước D_3: $D_3 = 1 – (S_3 / S_1)$

Kích cỡ cơ sở dữ liệu D_4: $D_4 = 1 – (S_5 / S_4)$

Việc phân ngăn cơ sở dữ liệu D_5: $D_5 = 1 – (S_6 / S_4)$

Đặc trưng vào / ra modul D_6: $D_6 = 1 – (S_7 / S_1)$

Khi các giá trị trên đã được xác định, DSQI được tính theo cách sau:

$$DSQI = \sum w_i D_i$$

Trong đó: *i = 1..6*

w_i là trọng số tương đối về tầm quan trọng của mỗi giá trị trung gian, $\sum w_i = 1$, nếu tất cả các D_i đều có trọng số ngang nhau thì $w_i = 0.167$.

Ý nghĩa của DSQI: Giá trị của DSQI cho các thiết kế quá khứ có thể được xác định và so sánh với thiết kế hiện đang phát triển. Nếu DSQI thấp hơn thì công việc thiết kế và xét duyệt thêm nữa sẽ cần tới. Tương tự, nếu những thay đổi được tiến hành đối với thiết kế hiện có thì sẽ tính được hiệu quả của những thay đổi này trên DSQI.

Chỉ số chín muồi phần mềm (SMI): Là chỉ số về sự ổn định của phần mềm dựa trên những thay đổi cho từng lần đưa ra sản phẩm. Các thông tin sau cần được xác định:

M_T : số các modul trong lần đưa ra hiện tại

F_c : số các modul đã được thay đổi trong lần đưa ra hiện tại

F_a : số các modul đã được thêm vào trong lần đưa ra hiện tại

F_d : số các modul của lần trước đã bị xoá trong lần đưa ra hiện tại

Chỉ số chín muồi phần mềm được tính theo công thức sau:

$$SMI = \frac{M_T - F_a - F_c - F_d}{M_T}$$

Khi SMI tiến tới 1 thì sản phẩm bắt đầu ổn định. SMI cũng có thể được dùng như độ đo cho các hoạt động bảo trì phần mềm theo kế hoạch.

1.2.2 Khoa học phần mềm của HALSTEAD

Lý thuyết của Halstead được suy ra từ một giả thuyết nền tảng “bộ óc con người tuân theo một tập hợp chặt chẽ các quy tắc hơn là nó từng biết…”. Khoa học phần mềm dùng một tập các cách đo nguyên thuỷ có thể suy ra sau khi mã được sinh ra hay được ước lượng một khi thiết kế đã hoàn thành.

Cách đo của Halstead:

n_1 : số các toán tử phân biệt xuất hiện trong chương trình

n_2 : số các toán hạng phân biệt xuất hiện trong chương trình

N_1 : tổng số lần xuất hiện toán tử

N_2 : tổng số lần xuất hiện toán hạng

Để minh hoạ cho cách đo này bạn hãy tham khảo thí dụ về chương trình SORT trong hình dưới.

Halstead đã dùng cách đo nguyên thuỷ để xây dựng các biểu thức về

- *chiều dài* toàn bộ chương trình

- *khối lượng* tối thiểu tiềm năng cho một thuật toán, khối lượng thực tế (số bit cần để xác định chương trình)
- *mức chương trình* (đo độ phức tạp của chương trình)
- *mức ngôn ngữ* (hằng số đối với một ngôn ngữ đã cho) và các tính năng khác như công sức phát triển, thời gian phát triển, thậm chí cả số lỗi trong phần mềm

Công thức tính chiều dài N:

$$N = n_1 \log_2 n_1 + n_2 \log_2 n_2$$

Công thức tính khối lượng chương trình:

$$V = N \log_2 (n_1 + n_2)$$

Lưu ý: V sẽ thay đổi theo ngôn ngữ lập trình và khối lượng thông tin biểu thị cần để xác định một chương trình. Với modul SORT, khối lượng cho bản FORTRAN là 204, khối lượng cho một bản hợp ngữ tương đương sẽ là 328 do phải mất nhiều nỗ lực hơn để xây dựng chương trình trong hợp ngữ.

Về mặt lý thuyết, một thuật toán cụ thể phải tồn tại một khối lượng tối thiểu nhất định. Halstead định nghĩa tỷ số khối lượng L là tỷ số khối lượng của dạng gọn nhất so với khối lượng thực tại của một chương trình. Trong thực tế, L bao giờ cũng bé hơn 1.

Dưới dạng cách đo nguyên thuỷ tỷ số khối lượng có thể được biểu diễn như sau:

$$L = \left[\frac{2}{n_1} \right] \left[\frac{n_2}{N_2} \right]$$

Halstead đề nghị rằng, mỗi ngôn ngữ có thể được phân loại theo mức ngôn ngữ l, là hằng số đối với một ngôn ngữ đã cho, nhưng công trình khác lại chỉ ra rằng mức ngôn ngữ lại là một hàm của cả ngôn ngữ lẫn người lập trình. Các giá trị mức ngôn ngữ sau đây đã được suy ra theo kinh nghiệm cho các ngôn ngữ chung:

Ngôn ngữ	Trung bình / l
Tiếng Anh	2,16
PL/1	1,53
ALGOL/68	2,12
FORTRAN	1,14
Hợp ngữ	0,88

Công trình của Halstead tuân theo kiểm chứng thực nghiệm, đã có một khối lượng lớn các nghiên cứu đã được tiến hành để điều tra khoa học phần mềm.

Chương trình sắp xếp tráo đổi

```
SUBROUTINE SORT (X,N)
DIMENSION X(N)
IF (N.L.T.2) RETURN
DO 20 I = 2,N
        DO 10J= 1,I
        IF(X(I).GE.X(J)) GOTO 10
                SAVE = X(I)
                X(I)  = X(J)
                X(J) = SAVE
10 CONTINUE
20 CONTINUE
        RETURN
        END
```

Toán tử của chương trình sắp xếp tráo đổi

Toán tử	Số đếm
1 Kết thúc câu lệnh	7
2 Chỉ số mảng	6
3 =	5
4 IF()	2
5 DO	2
6,	2
7 Kết thúc chương trình	1
8 .L.T	1
9 .G.E	1
n_1 = 10 GOTO 10	1
	28 = N_1

Toán hạng của chương trình sắp xếp tráo đổi

Toán hạng	Số đếm
1 X	6
2 I	5
3 J	4
4 N	2
5 2	2
6 SAVE	2
n_2 = 7 1	1
	28 = N_2

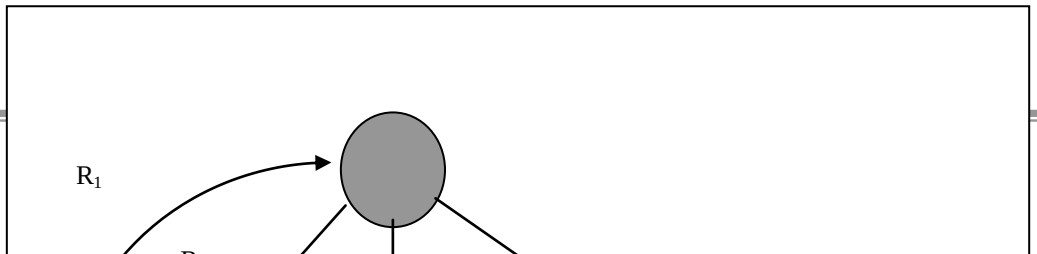
H4.1 *Toán tử và toán hạng cho một chương trình đơn giản*

1.2.3 Đo độ phức tạp của Thomas McCabe

Độ đo độ phức tạp của phần mềm do Thomas McCabe đưa ra dựa trên việc biểu diễn luồng điều khiển của chương trình. Một đồ thị chương trình G được dùng để mô tả luồng điều khiển. Mỗi khuyên tròn đều biểu thị cho một nhiệm vụ xử lý, luồng điều khiển được biểu diễn bằng mũi tên nối. Trong hình vẽ, nhiệm vụ xử lý a có thể tiếp sau là các nhiệm vụ b, c hay d tuỳ theo điều kiện kiểm thử như một phần của a. Nhiệm vụ xử lý b bao giờ cũng theo sau là e, và cả hai đều được thực hiện như một phần của chu trình lồng kép (mũi tên cong đi ngược tới b và a tương ứng)

McCabe định nghĩa cách đo độ phức tạp phần mềm dựa trên độ phức tạp xoay vòng của đồ thị chương trình cho một modul. Một kỹ thuật có thể được dùng để tính độ phức tạp xoay vòng, V(G) là xác định số các miền trong đồ thị phẳng. Một miền có thể được mô tả một cách không hình thức như một vùng được bao trên mặt phẳng của đồ thị. Số các miền được tính bằng cách đếm tất cả các miền bị giới hạn và miền không bị giới hạn bên ngoài đồ thị.

Đồ thị trong hình vẽ có 5 miền (từ R_1 đến R_5) do đó độ phức tạp xoay vòng của V(G) = 5



H4.2 Độ phức tạp đồ thị luồng điều khiển

Ý nghiã: Vì số các miền tăng lên theo số các đường quyết định và chu trình nên độ đo McCabe đưa ra một cách đo định lượng về độ khó khăn kiểm thử, và là một chỉ báo về độ tin cậy cuối cùng. McCabe cũng cho rằng V(G) có thể được dùng để đưa ra một chỉ dẫn định lượng về kích cỡ modul tối đa. Với việc thu thập dữ liệu từ một số các dự án lập trình thực tại ông đã thấy rằng V(G) = 10 dường như là giới hạn trên thực tế cho kích cỡ modul. Khi độ phức tạp xoay vòng của modul vượt quá số này thì vấn đề trở nên cực kỳ khó khăn cho việc kiểm thử tích hợp modul.

1.3 Độ tin cậy phần mềm

Độ tin cậy của chương trình máy tính là một yếu tố quan trọng của chất lượng toàn bộ của phần mềm. Nếu một chương trình thường xuyên không thực hiện được hoặc có lỗi thì hầu như các nhân tố chất lượng phần mềm có chấp nhận được cũng không thành vấn đề nữa.

Độ tin cậy phần mềm, không giống như nhiều yếu tố chất lượng khác, có thể được đo trực tiếp và ước lượng bằng cách dùng dữ liệu lịch sử và đang phát triển. Độ tin cậy phần mềm được định nghĩa dưới dạng thống kê như "xác suất vận hành không thất bại của chương trình máy tính trong một môi trường xác định với một thời gian xác định". Ví dụ: chương trình X được ước lượng có độ tin cậy 0,96 với thời gian xử lý 8 tiếng đồng hồ, tức là nếu chương trình X được thực hiện 100 lần và cần 8 giờ xử lý thì có thể vận hành đúng (không thất bại) 96 lần.

Đo độ tin cậy và tính sẵn có:

Nếu chúng ta xem xét một hệ thống dựa trên máy tính thì cách đo đơn giản về độ tin cậy là thời gian trung bình giữa những lần thất bại (MTBF):

MTBF = MTTF + MTTR

Trong đó:

MTTF: thời gian trung bình của thất bại

MTTR: thời gian trung bình để sửa chữa tương ứng.

Nhiều nhà nghiên cứu biện minh rằng MTBF là cách đo có ích hơn nhiều so với cách đo khiếm khuyết. Nói một cách đơn giản, người dùng cuối cùng quan tâm tới những thất bại, không mấy quan tâm tới số lỗi toàn bộ. Vì mỗi lỗi bao hàm trong chương trình không có cùng tỉ lệ thất bại, nên số lỗi toàn bộ ít được đưa ra chỉ báo về độ tin cậy của hệ thống. Ví dụ: hãy xét một chương trình đã vận hành 14 tháng, nhiều lỗi trong chương trình còn chưa được phát hiện, MTBF của những lỗi không rõ ràng này có thể lên đến 50 đến 100 năm. MTBF. Những lối khác chừng nào còn chưa được phát hiện ra có thể có tỷ lệ thất bại từ 18 đến 24 tháng. Ngay cả nếu mọi lỗi trong phân loại đầu tiên có bị loại bỏ ra thì tác động nên độ tin cậy phần mền vẫn không đáng kể.

Bên cạnh việc đo độ tin cậy, chúng ta phải phát triển một cách đo tính có sẵn. Tính sẵn có của phần mền là xác suất một chương trình vận hành theo yêu cầu tại một điểm nào đó trong thời gian :

$$\text{Tĩnh sẵn có} = \frac{MTTF}{MTTF + MTTR} 00\%$$

1.4 Cách tiếp cận bảo đảm chất lượng phần mềm

Lịch sử đảm bảo chất lượng trong phát triển phần mềm song song với lịch sử của chất lượng chế tạo phần cứng. Trong những ngày đầu của tin học (những năm 1950 và 1960) chất lượng là trách nhiệm duy nhất của người lập trình. Sau này, các chuẩn về đảm bảo chất lượng phần mềm đã được đưa vào các hợp đồng phát triển phần mềm và đã nhanh chóng lan rộng trong thế giới phát triển phần mềm thương mại. Đảm bảo chất lượng phần mềm SQA là một "mẫu hình hành động có hệ thống và có kế hoạch". Những người thực hiện SQA phải nhìn vào phần mềm theo quan điểm của khách hàng. Phần mềm có đáp ứng được các nhân tố chất lượng đã nêu trên không. SQA bao gồm 7 hoạt động chính: (1) Áp dụng các phương pháp kỹ thuật, (2) Tiến hành các xét duyệt kỹ thuật chính thức, (3) Kiểm thử phần mềm, (4) Buộc tôn trọng các chuẩn, (5) Kiểm soát thay đổi, (6) Đo, (7) Lưu giữ và báo cáo kết quả ghi lại.

Mặc dù nhiều nhà quản lý và người phát triển sẽ tranh luận nhiều nhu cầu phát triển phần mềm nhưng nhiều người không quan tâm tới việc thiết lập các chức năng SQA hình thức. Có nhiều nguyên nhân cho sự mâu thuẫn này .

1. Người quản lý ngần ngại gánh thêm chi phí phụ tăng thêm
2. Người phát triển cảm thấy họ đã làm xong những thứ cần làm
3. Không ai biết phải đặt chức năng này ở đâu trong tổ chức
4. Mọi người đều muốn tránh việc SQA được công nhận phải đưa vào tiến trình kỹ nghệ phầm mềm.

1.4.1 Xem xét nhu cầu cho SQA

Tất cả mọi tổ chức phát triển phần mềm đều có một cơ chế nào đó để thẩm định chất lượng. Trước khi các thủ tục đảm bảo chất lượng chính thức được thiết lập, tổ chức phát triển phần mềm nên chấp nhận các thủ tục, phương pháp, công cụ kỹ nghệ phần mềm. Phương pháp luận này khi được tổ hợp với một khuôn cảnh (đã thảo luận trong Chương II) thì có thể cải thiện rất nhiều chất lượng của tất cả các phần mền của tổ chức đó tạo ra. Bước đầu tiên cần được tiến hành là kiểm toán. Trạng thái hiện tại của việc bảo đảm chất lượng phần mềm là việc quản lý cấu hình phần mềm được thẩm định bằng cách xem xét các chủ đề sau:

Đường lối, thủ tục và *các chuẩn* hiện thời nào đang tồn tại cho mọi giai đoạn phát triển phần mềm? chúng có phải là bắt buộc không? có một đường lối riêng cho SQA không? Các đường lối có được áp dụng cho tất cả các hoạt động phát triển lẫn bảo trì không?

Tổ chức kỹ nghệ phần mềm nằm ở đâu trong sơ đồ tổ chức hiện tại? Việc bảo đảm chất lượng nằm ở đâu?

Đâu là mối quan hệ hiện thời giữa việc bảo đảm chất lượng và các thể chế khác. SQA tương tác thế nào với những người thực hiện các xét duyệt kỹ thuật quản lý cấu hình và kiểm thử?

Một khi những vấn đề trên đã được trả lời thì những điểm mạnh, điểm yếu cũng được xác định nếu nhu cầu cho SQA là rõ ràng thì việc thẩm định cẩn thận về những ý kiến ủng hộ và phản đối sẽ được xem xét .

Về phía tích cực SQA đưa ra những ích lợi sau:

1. Phần mềm sẽ có ít khiếm khuyết tiềm tàng hơn giúp giảm bớt nỗ lực và thời gian dành cho kiểm thử và bảo trì.

2. Độ tin cậy cao hơn sẽ làm nảy khách hàng yên tâm và thoả mãn hơn

3. Chi phí bảo trì có thể được rút bớt,

4. Chi phí vòng đời toàn bộ của phầm mềm giảm

Về phía tiêu cực :

SQA có thể gây ra vấn đề bởi những lý do sau:

1. Khó mà thiết lập được trong những tổ chức nhỏ nơi các tài nguyên sẵn có để thực hiện các hoạt động cần thiết là không sẵn sàng

2. Việc thay đổi không bao giờ dễ dàng .

3. Nó đòi hỏi chi phí tiền của và nếu không có thì sẽ không lập ngân sách tường minh cho kỹ nghệ phần mềm hay bảo đảm chất lượng phần mềm được.

Tại mức nền tảng SQA là chi phí hiệu quả nếu

$$C_3 > C_1 + C_2$$

C_3 Chi phí cho việc lỗi xuất hiện với chương trình không có SQA

C_2 là chi phí cho lỗi không được phát hiện ra bởi các hoạt động SQA

C_1 Là chi phí của bản thân chương trình SQA

Nói chung, SQA như một phần của cam kết quản lý toàn bộ để cải thiện chất lượng - thường được gọi là quản lý chất lượng toàn bộ.

1.4.2 Lập kế hoạch SQA và các chuẩn

Một khi tổ chức đã thành lập SQA thì cần phải phát triển một kế hoạch và cần tới các chuẩn. IEEE đã phát triển một định dạng chuẩn cho các kế hoạch SQA được liệt kê trong bảng dưới đây:

I. Mục đích của kế hoạch
II. Tham khảo
III. Quản lý
A. Tổ chức
B. Nhiệm vụ
C. Trách nhiệm
IV. Tài liệu
A. Mục đích
B. Tài liệu kỹ nghệ phần mềm cần thiết
C. Các tài liệu khác
V. Chuẩn, thực hành và quy ước
A. Mục đích
B. Quy ước
VI. Xét duyệt và kiểm toán

A. Mục đích
B. Các yêu cầu xét duyệt
1. Xét duyệt yêu cầu phần mềm
2. Xét duyệt thiết kế
3. Kiểm chứng phần mềm và xét duyệt hợp lệ
4. Kiểm toán chức năng
5. Kiểm toán vật lý
6. Kiểm toán trong tiến trình
7. Xét duyệt quản lý
VII. Quản lý cấu hình phần mềm
VIII. Báo cáo vấn đề và cách sửa chữa
IX. Công cụ, kỹ thuật và phương pháp luận
X. Kiểm soát mã
XI. Kiểm soát phương tiện
XII. Kiểm soát người cung cấp
XIII. Thu thập, bảo trì và ghi nhớ báo cáo

Bản kế hoạch SQA đưa ra bản lộ trình để thiết lập việc đảm bảo chất lượng phần mềm, trình bày một danh sách các chuẩn có liên quan tới SQA sẽ phục vụ hướng dẫn việc phát triển các thủ tục kỹ thuật để đạt tới chất lượng phần mềm.

2. Kiểm thử phần mềm

Kiểm thử phần mềm là phần tử mấu chốt của đảm bảo chất lượng phần mềm và biểu thị cho việc xét duyệt sau cùng về đặc tả thiết kế và mã hoá. Công việc này được tiến hành sau khi đã mã hoá xong và trước khi bàn giao sản phẩm cho khách hàng.

2.1 Nền tảng của kiểm thử phần mềm

Nền tảng kiểm thử phần mềm xác định ra các mục tiêu quan trọng hơn cả cho việc kiểm thử phần mềm. Trong các giai đoạn xác định và phát triển ban đầu, người kỹ sư cố gắng xây dựng phần mềm từ một quan điểm trừu tượng tới việc thực hiện hữu hình. Bây giờ đến kiểm thử. Người kỹ sư lại tạo ra một loạt các trường hợp kiểm thử với ý định "phá huỷ" phần mềm. Trong thực tế, kiểm thử là một bước trong tiến trình kỹ nghệ phần mềm.

2.1.1 Mục đích kiểm thử

- Kiểm thử là một tiến trình thực hiện chương trình với ý định tìm ra lỗi
- Làm lộ ra lỗi còn chưa phát hiện ra

Mục đích của chúng ta là thiết kế các phép kiểm thử để làm lộ ra một cách có hệ thống những lớp lỗi khác nhau trong một số lượng thời gian và công sức tối thiểu.

Nếu kiểm thử được tiến hành thành công thì nó sẽ làm lộ ra những lỗi trong phần mềm. Xem như một lợi ích phụ, kiểm thử chứng tỏ rằng các chức năng phần mềm làm việc theo đặc tả, các yêu cầu hiệu năng là được đáp ứng, đưa ra một chỉ dẫn tốt về độ tin cậy của chất lượng phần mềm.

Điều quan trọng là cần nhớ phát biểu sau trong tâm khi tiến hành kiểm thử:

"Kiểm thử không thể chứng minh được việc không có khiếm khuyết, nó chỉ có thể chứng minh rằng khiếm khuyết phần mềm hiện hữu"

2.1.2 Luồng thông tin kiểm thử

Luồng thông tin cho việc kiểm thử được tuân theo hình mẫu mô tả trong hình dưới. Hai lớp cái vào được cung cấp cho tiến trình kiểm thử:

(1) Cấu hình phần mềm gồm Bản đặc tả yêu cầu phần mềm, Bản đặc tả thiết kế, Chương trình gốc.

(2) Cấu hình kiểm thử gồm Kế hoạch và thủ tục kiểm thử, mọi công cụ kiểm thử dự đinh dùng, các trường hợp kiểm thử và kết quả dự kiến.

Kết quả kiểm thử được so sánh với kết quả dự kiến. Khi dữ liệu lỗi được phát hiện, việc gỡ lỗi bắt dầu. Tiến trình gỡ lỗi là phần không thể dự kiến nhất của tiến trình kiểm thử. Ví dụ: 1 lỗi chỉ ra sự sai biệt độ 0.01% giữa kết quả trông đợi và thực tại có thể mất 1 giờ, 1 ngày hay 1 tháng để chuẩn đoán và sửa chữa.

Khi kết quả kiểm thử được thu thập và đánh giá, thì một chỉ dẫn định lượng về chất lượng và độ tin cậy phần mềm bắt đầu nổi lên bề mặt. Nếu hay gặp phải những lỗi nghiêm trọng yêu cầu sửa đổi thiết kế thì chất lượng và độ tin cậy phần mềm là đáng ngờ, cần có các kiểm thử thêm nữa. Mặt khác nếu chức năng phần mềm dường như làm việc đúng, lỗi gặp phải dễ sửa thì có thể rút ra hai kết luận: (1)Chất lượng và độ tin cậy phần mềm là chấp nhận được, (2) Kiểm thử không tương xứng để làm lộ ra những lỗi nghiêm trọng. Nếu việc kiểm thử không làm lộ ra lỗi nào thì có thể hoài nghi rằng cấu hình kiểm thử chưa được cân nhắc đúng mức, các lỗi vẫn còn ẩn núp trong phần mềm. Những lỗi khiếm khuyết này chung cuộc sẽ bị phát hiện bởi người dùng và được người phát triển sửa chữa trong giai đoạn bảo trì.

2.2 Chiến lược kiểm thử phần mềm

Một chiến lược để kiểm thử phần mềm tích hợp cả các kỹ thuật thiết kế trường hợp kiểm thử vào trong một loạt các bước được hoạch định chu đáo để làm cho viêc xây dựng phần mềm thành công. Chiến lược phần mềm đưa ra bản lộ trìnhcho người phát triển phần mềm, cho tổ chức đảm bảo chất lượng phần mềm và cho khách hàng một bản lộ trình mô tả các bước cần được tiến hành như một phần của việc kiểm thử, thời gian tiến hành và xác định bao nhiêu nỗ lực, thời gian và tài nguyên sẽ cần tới. Do đó bất kỳ chiến lược kiểm thử nào cũng phải tổ hợp với kế hoạch kiểm thử, thiết kế trường hợp kiểm thử, thực hiện kiểm thử, và thu thập rồi đánh giá dữ liệu kết quả.

Một chiến lược kiểm thử phần mềm nên đủ mềm dẻo để thúc đẩy tính sáng tạo và việc điều chỉnh theo yêu cầu, điều cần thiết cho việc kiểm thử thích hợp với tất cả các hệ thống dựa trên phần mềm lớn. Đồng thời chiến lược này cũng phải đủ chặt chẽ để thúc đẩy việc lập kế hoạch hợp lý và theo dõi việc quản lý khi dự án tiến triển.

2.2.1 Cách tiếp cận chiến lược tới kiểm thử phần mềm

Kiểm thử là một tập hợp những hoạt động có thể được lập kế hoạch trước và tiến hành một cách có hệ thống. Do vậy một tiêu bản cho việc kiểm thử phần mềm nên được xác định cho tiến trình kỹ nghệ phần mềm. Tiêu bản gồm một tập các bước trong đó chúng ta có thể đặt vào những kỹ thuật thiết kế trường hợp kiểm thử và phương pháp kiểm thử.

Đăc trưng của tiêu bản để kiểm thử phần mềm:

- Việc kiểm thử bắt đầu tại mức modul và làm việc "hướng ra ngoài" tới việc tích hợp cho toàn bộ hệ thống dựa trên máy tính.

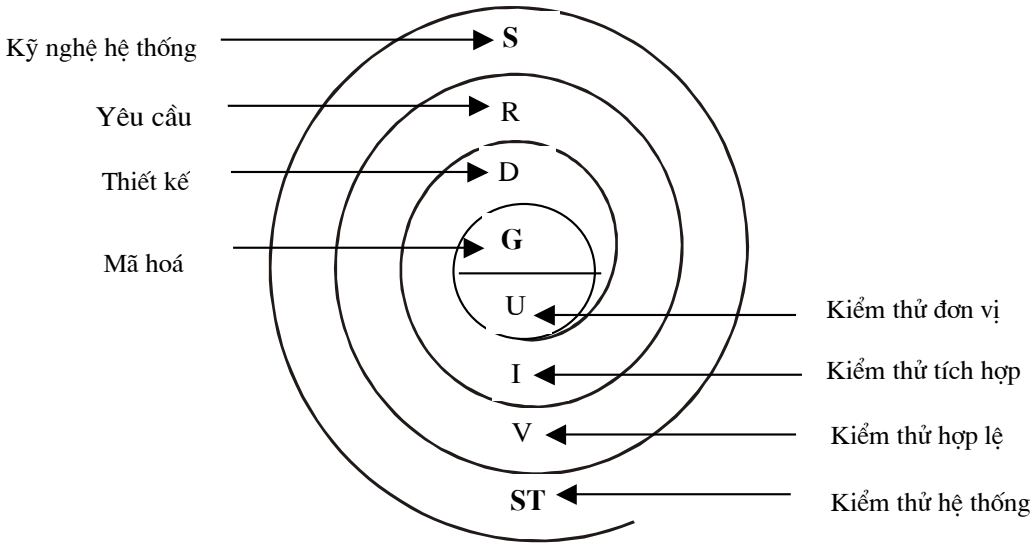
- Các kỹ thuật kiểm thử khác nhau là thích hợp tại các điểm khác nhau theo thời gian
- Việc kiểm thử được tiến hành bởi người phát triển phần mềm và (với các dự án lớn) một nhóm kiểm thử độc lập
- Việc kiểm thử và gỡ lỗi là những hoạt động khác nhau, nhưng gỡ lỗi phải phù hợp với mọi chiến lược kiểm thử.

Chiến lược kiểm thử phần mềm phải phù hợp với các kiểm thử mức thấp nhất cần thiết để kiểm chứng rằng một đoạn chương trình gốc nhỏ đã được cài đặt đúng đắn, cũng như các kiểm thử mức cao hợp lệ hoá những chức năng hệ thống chính theo yêu cầu của khách hàng.

Chiến lược phải đưa ra hướng dẫn cho người thực hành và một tập các cột mốc cho người quản lý. Bởi các bước của chiến lược kiểm thử xuất hiện vào lúc sức ép hết hạn thời gian bắt đầu tăng lên nên tiến độ phải là đo được và các vấn đề phải nổi lên càng sớm càng tốt.

2.2.2 Chiến lược kiểm thử phần mềm

Tiến trình kỹ nghệ phần mềm có thể được xét theo vòng xoắn ốc như trong hình vẽ dưới. Ban đầu, kỹ nghệ phần mềm xác định vai trò của phần mềm và đưa tới việc phân tích yêu cầu phần mềm, nơi thiết lập nên lĩnh vực thông tin, chức năng, hành vi, hiệu năng và ràng buộc và tiêu chuẩn hợp lệ cho phần mềm. Đi vào trong vòng xoắn ốc, chúng ta tới thiết kế và cuối cùng tới mã hoá. Để xây dựng phần mềm máy tính, chúng ta đi dọc theo đường xoắn ốc, mỗi lần mức độ trừu tượng lại giảm dần. Một chiến lược cho kiểm thử phần mềm cũng có thể xem xét bằng cách đi theo đường xoắn ốc ra ngoài. Việc kiểm thử đơn vị bắt đầu tại tâm xoắn ốc và tập trung vào các đơn vị của phần mềm khi được cài đặt trong chương trình gốc. Việc kiểm thử tiến triển bằng cách đi ra theo đường xoắn ốc tới kiểm thử tích hợp, nơi tập trung vào thiết kế và việc xây dựng kiến trúc phần mềm (kiểm thử đơn vị và kiểm thử tích hợp sẽ được trình bày kỹ ở mục dưới). Đi ra thêm một vòng xoáy nữa trên đường xoắn ốc chúng ta gặp kiểm thử hợp lệ, nơi các yêu cầu được thiết lập như một phần của việc phân tích yêu cầu phần mềm, được hợp lệ hoá theo phần mềm đã được xây dựng. Cuối cùng chúng ta tới kiểm thử hệ thống, nơi phần mềm và các phần tử hệ thống khác được kiểm thử tổng thể.



H4.3 Chiến lược kiểm thử

Xem xét tiến trình này theo quan điểm thủ tục thì việc kiểm thử bên trong hoàn cảnh kỹ nghệ phần mềm thực tại là một chuỗi gồm ba bước được thực hiện tuần tự nhau. Các bước

này được biểu diễn trong hình vẽ dưới. Ban đầu, việc kiểm thử tập trung vào từng modul riêng biệt, đảm bảo rằng chúng vận hành đúng đắn như một đơn vị. Do vậy mới có tên là *kiểm thử đơn vị.* Tiếp đó các modul phải được lắp ghép hay tích hợp lại để tạo nên bộ trình phần mềm hoàn chỉnh. Việc kiểm thử tích hợp để cập tới các vần đề có liên quan tới các vấn đề kiểm chứng và xây dựng chương trình. Sau khi phần mềm đã được tích hợp, một tập các kiểm thử cấp cao sẽ được tiến hành. Các tiêu chuẩn hợp lệ cũng phải được kiểm thử. Việc kiểm thử hợp lệ đưa ra sự đảm bảo cuối cùng rằng phần mềm đã đáp ứng cho tất cả các yêu cầu chức năng, hành vi và sự hoàn thiện.

Bước kiểm thử cấp cao cuối cùng rơi ra ngoài phạm vi của kỹ nghệ phần mềm và rơi vào hoàn cảnh rộng hơn của kỹ nghệ hệ thống máy tính. Phần mềm một khi đã được hợp lệ hoá phải được tổ hợp với các phần tử hệ thống khác (phần cứng, con người, cơ sở dữ liệu…). Kiểm thử hệ thống kiểm chứng lại rằng tất cả các yếu tố có khớp đúng với nhau không và rằng chức năng/độ hoàn thiện hệ thống toàn bộ đã đạt được.

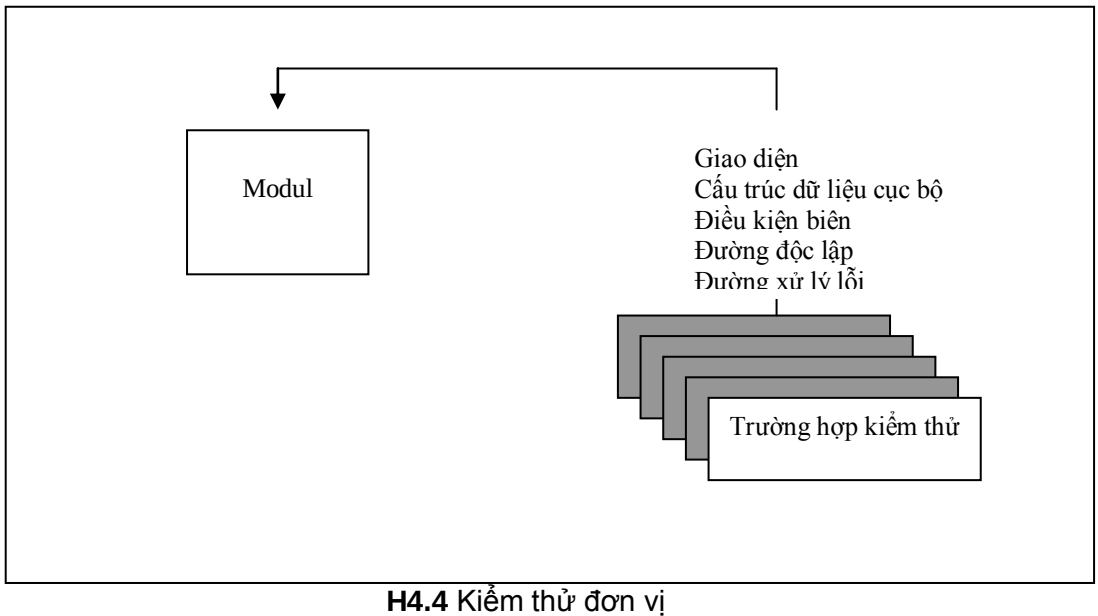
2.2.3 Tổ chức việc kiểm thử phần mềm

Người phát triển phần mềm bao giờ cũng phải có trách nhiệm với việc kiểm thử riêng các đơn vị modul chương trình, để đảm bảo rằng mỗi modul chương trình thực hiện đúng chức năng nó đã được thiết kế. Trong nhiều trường hợp người phát triển cũng nên tiến hành thêm cả kiểm thử tích hợp là bước kiểm thử dẫn tới việc xây dựng toàn bộ cấu trúc chương trình. Chỉ khi kiến trúc phần mềm đã hoàn tất thì nhóm kiểm thử độc lập (ITG – *Independent Testing Group*) mới tham gia vào.

Vai trò cuả nhóm kiểm thử độc lập là loại bỏ vấn đề cố hữu là để người xây dựng kiểm thử những cái anh ta đã xây dựng ra. Việc kiểm thử độc lập loại bỏ xung khắc lợi ích mà nếu không có nhóm này sẽ xảy ra. Cuối cùng, nhân sự (tester) trong nhóm kiểm thử độc lập được trả tiền để tìm ra lỗi.

Tuy nhiên người phát triển phần mềm không chuyển giao chương trình cho ITG rồi bỏ đi, mà làm việc chặt chẽ trong toàn bộ dự án phần mềm để đảm bảo rằng những kiểm thử kỹ lưỡng sẽ được tiến hành. Trong khi kiểm thử người phát triển phải có sẵn để sửa lỗi đã phát hiện ra.

2.2.3.1 Kiểm thử đơn vị



H4.4 Kiểm thử đơn vị

Modul giao diện được kiểm thử để bảo đảm rằng thông tin chảy đúng vào việc vào ra khỏi chương trình đang kiểm thử. *Cấu trúc dữ liệu cục bộ* được xem xét để đảm bảo rằng dữ liệu được lưu giữ tạm thời vẫn duy trì được tính toàn vẹn cuả nó trong tất cả các bước khi thực hiện thuật toán. *Các điều kiện biên* được kiểm thử để đảm bảo rằng modul vận hành đúng tại các biên được thiết lập cho việc xử lý có giới hạn. Tất cả các *đường độc lập* đi qua cấu trúc điều khiển đều được cho chạy qua để đảm bảo rằng tất cả các câu lệnh trong một modul đều được thực hiện qua ít nhất một lần. Cuối cùng, tất cả các *đường xử lý lỗi* cũng được kiểm thử.

Việc kiểm thử luồng dữ liệu đi qua giao diện modul là cần thiết trước khi bất kỳ kiểm thử nào khác được tiến hành. Nếu dữ liệu không đi vào/ra đúng thì tất cả các kiểm thử khác đều phải bàn cãi lại.

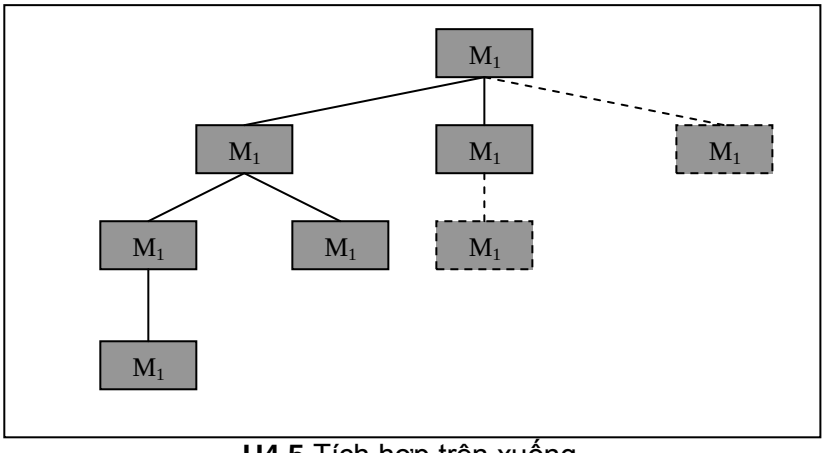
Đối với các phép kiểm thử đơn vị cần xác định:

1. Số các tham biến vào có bằng đối số không?
2. Các thuộc tính tham biến và đối số có tương ứng với nhau không?
3. Hệ thống các đơn vị tham biến và đối có đúng với nhau không?
4. Số các đối số được truyền tới modul được gọi có bằng số các tham biến không?
5. Thuộc tính của các đối được truyền cho modul được gọi có bằng với thuộc tính của tham biến không?
6. Hệ thống các đơn vị của đối được truyền tới modul được gọi có bằng hệ thống đơn vị của tham biến không?
7. Các thuộc tính số và trật tự tham biến của chức năng có sẵn có đúng không?
8. Có tham khảo nào tới các tham biến không được gắn với điểm vào hiện tại không?
9. Các đối *chỉ vào* có bị thay đổi không?
10. Các định nghĩa biến toạn cục có nhất quán trong các modul không?
11. Các ràng buộc có được truyền như đối số không?

2.2.3.2 Kiểm thử tích hợp

Kiểm thử tích hợp là một kỹ thuật hệ thống để xây dựng cấu trúc chương trình trong khi đồng thời tiến hành các kiểm thử để phát hiện lỗi liên kết trong giao tiếp. Mục đích là lấy các modul đã kiểm thử đơn vị xong và xây dựng nên một cấu trúc chương trình được quy định bởi thiết kế.

a) Tích hợp từ trên xuống



H4.5 Tích hợp trên xuống

Tích hợp từ trên xuống là cách tiếp cận tăng dần tới việc xây dựng cấu trúc chương trình. Các modul được tích hợp bằng cách đi dần xuống qua cấp bậc điều khiển, bắt đầu từ modul điều khiển chính (chương trình chính). Các modul phụ thuộc vào modul điều khiển chính sẽ được tổ hợp dần vào trong cấu trúc theo chiều sâu hoặc chiều rộng trước.

Tiến trình tích hợp được thực hiện trong một loạt năm bước:

1. Modul điều khiển chính được dùng như một khiển trình kiểm thử, các cuống (là các chương trình con "*câm*" thay thế cho các modul phụ thuộc và modul đang được kiểm thử) được thay thế cho tất cả các modul phụ thuộc trực tiếp vào modul điều khiển chính.
2. Tuỳ theo cách tích hợp được lựa chọn (theo chiều sâu hay rộng) các chương trình con phụ thuộc được thay thế từng cái một mỗi lần bằng các modul thực tại.
3. Việc kiểm thử được tiến hành khi từng modul được tích hợp vào.
4. Khi hoàn thành từng tập các phép kiểm thử, các chương trình con câm khác sẽ được thay thế bằng các modul thực.
5. Kiểm thử hồi quy (tiến hành tất cả hay một số các phép kiểm thử trước) có thể được tiến hành để đảm bảo rằng những lỗi mới không bị đưa thêm vào. Quay trở lại bước 2 cho tới khi toàn bộ cấu trúc chương trình được xây dựng.

Nhận xét: Cuống thay thế cho các modul cấp thấp khi bắt đầu kiểm thử từ trên xuống, do đó không có dữ liệu nào có nghĩa có thể chảy ngược lên trong cấu trúc chương trình. Người kiểm thử đứng trước một số lựa chọn: (1) để trễ nhiều việc kiểm thử tới khi cuống được thay thế hết, (2) xây dựng các cuống thực hiện những chức năng giới hạn mô phỏng cho modul thực tại, (3) tích hợp phần mềm từ đáy cấp bậc lên.

Cách tiếp cận thứ nhất (để trễ kiểm thử cho tới khi cuống được thay thế bởi modul thực tại) gây cho chúng ta bị mất điều khiển đối với sự tương ứng giữa kiểm thử đặc biệt và việc tổ hợp các modul đặc biệt. Điều này có thể dẫn tới những khó khăn trong việc xác định nguyên nhân lỗi và có khuynh hướng vi phạm ràng buộc của cách tiếp cận trên xuống. Cách tiếp cận thứ hai ổn hơn nhưng có thể dẫn tới kinh phí khá lớn, vì cuống ngày càng phức tạp hơn. Cách tiếp cận thứ 3 được gọi là kiểm thử từ dưới lên được thảo luận trong mục sau.

b) Tích hợp dưới lên

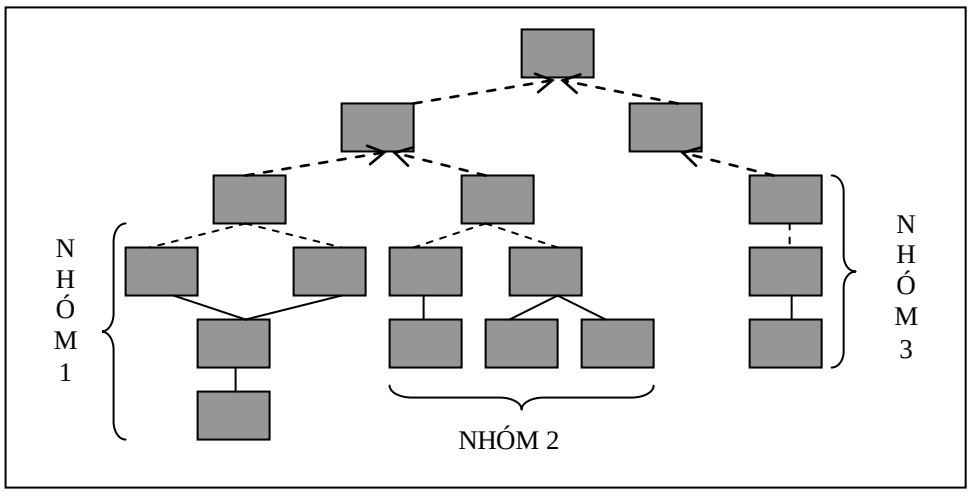
Bắt đầu xây dựng và kiểm thử với các modul nguyên tử (các modul ở mức thấp nhất trong cấu trúc chương trình). Vì các modul này được tích hợp từ dưới lên nên việc xử lý yêu cầu đối với các modul phụ thuộc vào nó ở một mức nào đó bao giờ cũng có sẵn và nhu cầu về cuống bị dẹp bỏ.

Chiến lược tích hợp từ dưới lên có thể được thực hiện qua những bước sau:

1. Các modul cấp thấp được tổ hợp vào các *chùm* (đôi khi được gọi là *kiểu kiến trúc*) thực hiện cho một chức năng con phần mềm đặc biệt.
2. Khiển trình (một chương trình điều khiển cho kiểm thử) được viết ra để phối hợp việc vào ra trường hợp kiểm thử.
3. Kiểm thử chùm
4. Loại bỏ khiển trình và chùm được tổ hợp chuyển lên trong cấu trúc chương trình

Khi việc tích hợp đi lên, nhu cầu về các khiển trình kiểm thử tách biệt ít dần. Trong thực tế, nếu hai mức đỉnh của cấu trúc chương trình được tịch hợp theo kiểu trên xuống thì số các khiển trình có thể được giảm bớt khá nhiều và việc tích hợp các chùm được đơn giản hoá đáng kể.

Việc tích hợp đi theo mẫu được minh hoạ dưới đây:



H4.6 Tích hợp dưới lên

Phạm vi kiểm thử tóm tắt các đặc trưng chức năng, sự hoàn thiện và thiết kế bên trong cần được kiểm thử riêng biệt, gắn kèm các tiêu chuẩn để hoàn tất từng giai đoạn, các ràng buộc lịch biểu.

Phần kế hoạch kiểm thử mô tả chiến lược chung cho việc tích hợp, việc kiểm thử chia thành các giai đoạn và khối, đề cập tới các chức năng riêng của phần mềm.

Dàn bài đặc tả

I. Phạm vi kiểm thử
II. Kế hoạch kiểm thử
A. Các giai đoạn và khối kiểm thử
B. Lịch biểu
C. Tổng phí phần mềm
D. Môi trường và tài nguyên
III. Thủ tục kiểm thử n. (mô tả việc kiểm thử cho khối n)
A. Thứ tự thích hợp
1. Mục đích
2. Modul cần kiểm thử
B. Kiểm thử đơn vị cho các modul khối
1. Mô tả kiểm thử cho modul m
2. Mô tả tổng phí phần mềm
3. Kết quả dự kiến
C. Môi trường kiểm thử
1. Công cụ hay kỹ thuật đặc biệt

2. Mô tả tổng phí phần mềm
D. Dữ liệu trường hợp kiểm thử
E. Kết quả dự kiến cho khối n
IV. Kết quả kiểm thử thực tế
V. Tham khảo

Ví dụ: Kiểm thử tích hợp cho một hệ thống CAD được chia thành các giai đoạn kiểm thử sau:

- Giao diện người dùng: chọn chỉ lệnh, tạo ra việc vẽ, biểu diễn hiển thị, xử lý lỗi và biểu diễn lỗi.
- Thao tác và phân tích dữ liệu: tạo ra ký hiệu, tầm hướng, quay, tính các tính chất vật lý.
- Xử lý và sinh hiển thị: hiển thị hai chiều, hiển thị ba chiều, đồ thị và sơ đồ
- Quản trị cơ sở dữ liệu: thâm nhập, cập nhật, tính toàn vẹn, hiệu năng

Trong mỗi giai đoạn và các giai đoạn con đều nêu ra các chức năng bên trong phần mềm, nói chung có thể liên quan đến một lĩnh vực riêng của cấu trúc chương trình. Do đó, các khối chương trình (nhóm các modul) được tạo ra để tương ứng với từng giai đoạn.

Các tiêu chuẩn sau đây và các phép kiểm thử tương ứng được áp dụng cho tất cả các giai đoạn kiểm thử:

Tính thống nhất giao diện: Các giao diện bên trong và bên ngoài được kiểm thử khi từng modul (hay nhóm modul) được tổ hợp vào trong cấu trúc.

Hợp lệ chức năng: Tiến hành các kiểm thử đã được thiết kế để phát hiện ra lỗi chức năng.

Nội dung thông tin: Tiến hành các kiểm thử đã được thiết kế để phát hiện ra lỗi liên kết với cấu trúc dữ liệu cục bộ hay toàn cục được sử dụng.

Sự hoàn thiện: Tiến hành các kiểm thử đã được thiết kế để kiểm chứng các cận hoàn thiện đã được thiết lập trong thiết kế phần mềm.

Những tiêu chuẩn này và các kỹ thuật kiểm thử liên kết với chúng được trình bày trong bản Đặc tả kiểm thử.

2.2.3.3 Kiểm thử hợp lệ

Vào cao điểm của việc kiểm thử tích hợp, phần mềm được lắp ráp hoàn chỉnh thành một bộ, các lỗi giao tiếp đã được phát hiện và sửa chữa, và loạt kiểm thử cuối cùng được bắt đầu – kiểm thử hợp lệ.

a) Tiêu chuẩn kiểm thử hợp lệ

Cả hai bản kế hoạch và thủ tục đều được thiết kế để đảm bảo rằng tất cả các yêu cầu chức năng đều được thoả mãn, tất cả các hiệu năng đều đạt được, tài liệu đúng và các yêu cầu khác cũng được đáp ứng (tính khả chuyển, tính tương hợp, khắc phục lỗi, bảo trì)

Sau khi tiến hành mỗi trường hợp kiểm thử, một trong hai điều kiện có thể tồn tại:

(1) Các đặc trưng chức năng và hiệu năng đúng với đặc tả và được chấp nhận

(2) Một độ lệch so với đặc tả được phát hiện ra và một danh sách các khiếm khuyết được tạo ra.

b) Xem xét cấu hình

Một phần tử quan trọng của tiến trình hợp lệ hoá là xét duyệt cấu hình. Mục đích để đảm bảo rằng tất cả các phần tử của cấu hình phần mềm đã được phát triển đúng đắn và được phân loại kèm theo mô tả chi tiết cần thiết để hỗ trợ cho giai đoạn bảo trì vòng đời phần mềm. Việc xét duyệt cấu hình đôi khi được gọi là kiểm toán

c) Kiểm thử alpha và beta

Phần lớn những người xây dựng sản phẩm phần mềm đều dùng một tiến trình gọi là: Kiểm thử alpha và beta để phát hiện ra những lỗi mà dường như chỉ người dùng cuối cùng mới có thể tìm ra.

Kiểm thử alpha được khách hàng tiến hành tại cơ quan của người phát triển. Phần mềm được dùng theo sự sắp đặt người phát triển "nhìn qua vai" người dùng và ghi lại những và các vấn đề sư dụng. Kiểm thử alpha được tiến hành trong một môi trường có kiểm soát.

Kiểm thử beta được diễn ra tại một hay nhiều cơ quan của khách hàng, được tiến hành bởi những người dùng cuối cùng, người phát triển nói chung không có mặt. Khách hàng ghi lại tất cả các vấn đề (thực hay tưởng tượng) gặp phải trong khi kiểm thử beta và báo cáo lại tất cả các vấn đề đó cho người phát triển một cách đều đặn. Kết quả là, người phát triển sửa đổi các lỗi được báo cáo và chuẩn bị đưa ra sản phẩm phần mềm cho toàn bộ cơ sở khách hàng.

2.2.3.4 Kiểm thử hệ thống (System Test)

Phần mềm là một trong những yếu tố của hệ thống dựa trên máy tính lớn hơn. Phần mềm được tổ hợp với các yếu tố hệ thống khác và một loạt các kiểm thử hợp lệ hóa và tích hợp hệ thống sẽ được tiến hành. Việc kiểm thử hệ thống là một loạt các kiểm thử khác nhau có mục đích là thử hệ thống dựa trên máy tính một cách đầy đủ.

a. Kiểm thử phục hồi (Recovery Test)

Kiểm thử phục hồi là kiểm thử hệ thống bắt buộc phần mềm phải hỏng theo nhiều cách và kiểm chứng rằng việc phục hồi được thực hiện đúng. Nếu việc phục hồi là tự động (được thực hiện bởi bản thân hệ thống) thì việc khởi đầu lại, cơ chế điểm kiểm tra, phục hồi dữ liệu và cho chạy lại sẽ được đánh giá về tính đúng đắn. Nếu việc phục hồi đòi hỏi sự can thiệp của con người thì thời gian trung bình để sửa chữa sẽ được ước lượng để định xem liệu nó có trong giới hạn chấp nhận được không.

b. Kiểm thử an toàn (Security Test)

Kiểm thử an toàn cố gắng kiểm chứng rằng các cơ chế bảo vệ được xây dựng bên trong hệ thống trong thực tế sẽ bảo vệ cho hệ thống khỏi sự thâm nhập không đúng. Người kiểm thử đóng vai trò cá nhân muốn thâm nhập vào hệ thống, với nhiều mục đích ví dụ: lấy mật khẩu, công kích vào hệ thống bằng một phần mềm làm riêng chuyên dụng để phá vỡ bất kỳ sự phòng vệ nào đã được xây dựng, gây tràn hệ thống, do đó hệ thống từ chối việc phục vụ người khác…

c. Kiểm thử gay cấn (Stress Test)

Kiểm thử gay cấn thiết kế để làm cho chương trình đương đầu được với các tình huống bất thường. Về bản chất, người kiểm thử thực hiện việc kiểm thử bất thường luôn tự hỏi: Ta có thể quay ngược lại chương trình bằng cách nào trước khi nó hỏng?

Kiểm thử gay cấn cho hệ thống thử chạy theo cách thức yêu cầu các tài nguyên theo tần số hay khối lượng các bất thường. Ví dụ:

(1) Các kiểm thử đặc biệt có thể được thiết kế để sinh ra 10 ngắt trong 1 giây, khi tỷ lệ trung bình chỉ là một hay hai.

(2) Tỷ lệ dữ liệu vào có thể tăng lên theo cấp độ nào đó để xác định cách các chức năng vào sẽ đáp ứng.

(3) Các trường hợp kiểm thử đòi hỏi bộ nhớ tối đa hay các tài nguyên khác

(4) Các trường hợp kiểm thử có thể gây ra "đập vỡ" hệ điều hành

(5) Các trường hợp kiểm thử gây ra việc tiêu tốn dữ liệu thường chú trên đĩa

Về bản chất người kiểm thử cố gắng phá vỡ chương trình

d. Kiểm thử hiệu năng (Performance Test)

Đối với các hệ thống thời gian thực và nhúng, phần mềm cung cấp chức năng yêu cầu nhưng không tuân thủ các yêu cầu hiệu năng thì không thể chấp nhận được. Kiểm thử hiệu năng được thiết kế để kiểm thử khả năng khi chạy của phần mềm bên trong hoàn cảnh của hệ thống đã tích hợp. Kiểm thử hiệu năng xuất hiện trong toàn bộ tất cả các bước của tiến trình kiểm thử. Ngay cả ở mức kiểm thử đơn vị, hiệu năng của một modul riêng cũng có thể được thẩm định. Tuy nhiên phải đến khi tất cả các phần tử hệ thống đều đã được tích hợp hết thì hiệu năng đúng của hệ thống mới có thể chắc chắn được.

Kiểm thử hiệu năng đôi khi còn đi kèm với kiểm thử gay cấn và thường đòi hỏi các thiết bị phần cứng và phần mềm. Thường phải đo việc sử dụng tài nguyên (như chu trình bộ nhớ) theo cách chinh xác. Các thiết bị ngoài có thể điều phối các khoảng thực hiện, sự kiện ghi lại (như cách ngắt) khi chúng xuất hiện. Bằng việc cung cấp thiết bị cho hệ thống, người kiểm thử có thể phát hiện ra những tình huống dẫn đến việc suy giảm và khả năng sai hỏng hệ thống.

3. Bảo trì phần mềm

Bảo trì phần mềm đã được đặc trưng là "núi băng trôi". Thực tế, ta biết rằng rất nhiều vấn đề tiềm năng và chi phí nằm ở dưới bề mặt. Việc bảo trì phần mềm hiện có, có thể chiếm đến 70% phần mềm toàn bộ nỗ lực chi tiêu của tổ chức phần mềm. Trong phạm vi này, chúng ta có thể dự kiến một tổ chức phần mềm hướng bảo trì, nó phải dành tất cả tài nguyên có sẵn cho việc bảo trì phần mềm cũ.

Bản chất thay đổi thường xuyên nằm trong mọi công việc phần mềm. Thay đổi là điều không tránh khỏi khi hệ thống dựa trên máy tính được xây dựng. Do đó, chúng ta phải xây dựng cơ chế để đánh giá, kiểm soát và thực hiện những thay đổi.

3.1 Định nghĩa về bảo trì phần mềm

Chúng ta định nghĩa bảo trì bằng cách mô tả 4 hoạt động cần thực hiện sau

(1) Trong khi dùng bất kỳ chương trình lớn nào, lỗi sẽ xuất hiện và được báo về cho người phát triển, tiến trình bao gồm việc chuẩn đoán và sửa một hay nhiều lỗi được gọi là bảo trì *sửa chữa*

(2) Hoạt động thứ 2, đóng góp thêm vào việc bảo trì, xuất hiện bởi sự thay đổi nhanh chóng thường gặp trong mọi khía cạnh của tính toán. Các thế hệ phần cứng mới dường như được công bố theo chu kỳ 24 tháng, các hệ điều hành mới xuất hiện đều đặn, thiết bị ngoại vi và các phần tử hệ thống khác thường xuyên được thay đổi và nâng cấp. Đời sống có ích của phần mềm ứng dụng lại có thể kéo dài hơn 10 năm, sống lâu hơn môi trường hệ thống ban đầu nó được phát triển. Do đó *bảo trì thích nghi* - một hoạt động làm thay đổi phần mềm để khớp với môi trường thay đổi - vừa cần thiết lại vừa phổ biến

(3) Hoạt động thứ 3 có thể được áp dụng cho định nghĩa về bảo trì xuất hiện khi bộ trình phần mềm đã thành công. Khi phần mềm được dùng người ta nhận được từ người dùng

những khuyến cáo về khả năng mới, những sửa đổi về chức năng hiện tại và những nâng cấp chung. Để thoả mãn những yêu cầu này, việc *bảo trì hoàn thiện* được tiến hành.

(4) Hoạt động bảo trì thứ 4 xuất hiện khi phần mềm được thay đổi để cải thiện tính bảo trì và hay tin cậy sau này, hay đưa ra một cơ sở tốt hơn cho khả năng nâng cấp trong tương lai, thường được gọi là bảo trì phòng ngừa.

3.2 Các đặc trưng bảo trì

Việc bảo trì phần mềm cho mãi đến rất gần đây vẫn là giai đoạn bị bỏ quên trong tiến trình kỹ nghệ phần mềm còn tương đối ít nghiên cứu hay sản phẩm được thu thập về chủ đề này và cũng chỉ có vài cách tiếp cận hay phương pháp đã được đưa ra.

Để hiểu các đặc trưng của bảo trì phầm mềm, ta xét chủ đề này theo 3 quan điểm khác nhau:

1. Các hoạt động đòi hỏi hoàn thành trong giai đoạn bảo trì và tác động của cách tiếp cận kỹ nghệ phần mềm lên tính hiệu quả của các hoạt động này.

2. Chi phí đối với giai đoạn bảo trì

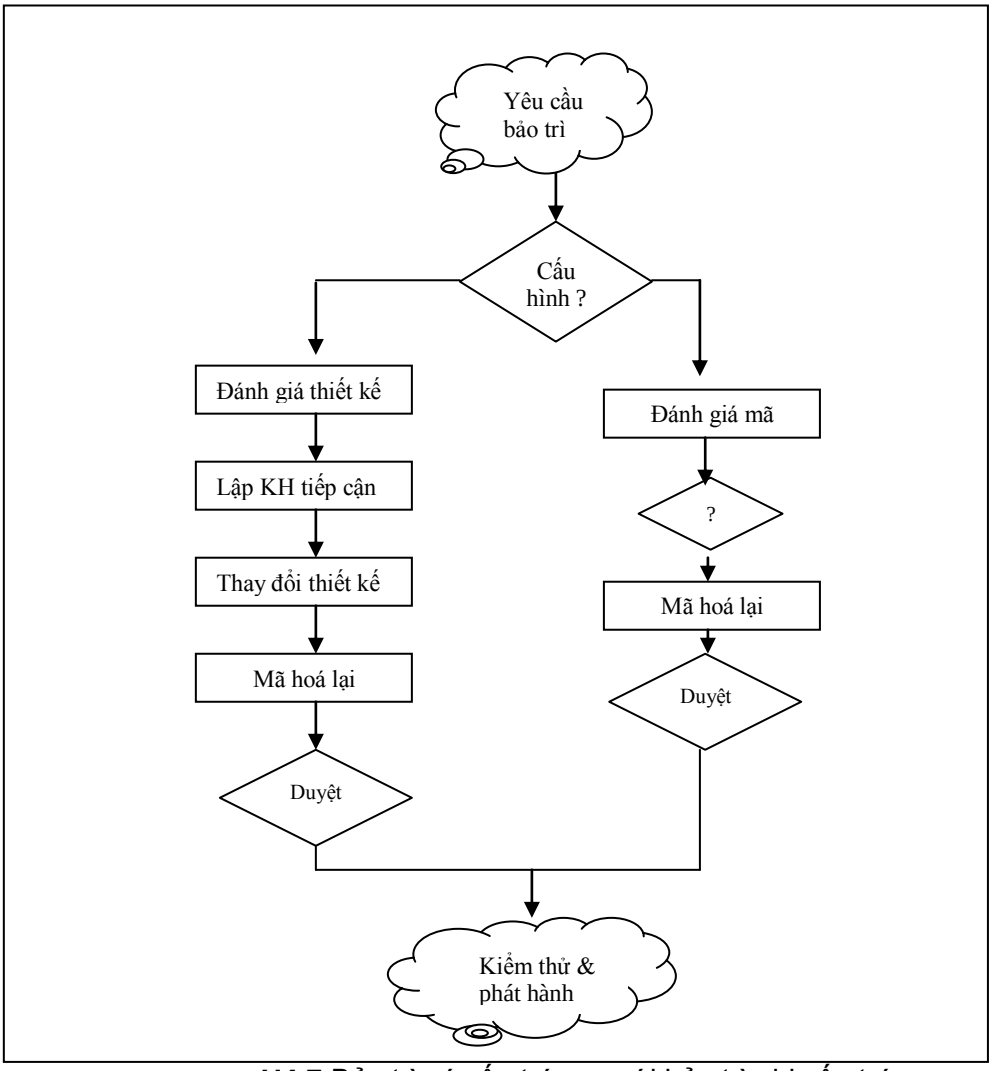
3. Các vấn đề thường gặp phải khi việc bảo trì được tiến hành

3.2.1 Bảo trì có cấu trúc so với phi cấu trúc

Luồng các sự kiện có thể xuất hiện như kết quả của nhu cầu bảo trì được minh hoạ trong hình vẽ dưới.

Nếu phần tử sẵn có của cấu hình phần mềm là chương trình gốc thì hoạt động bảo trì bắt đầu với một đánh giá cẩn thẩn về mã. Những đặc trưng tinh vi như cấu trúc chương trình, cấu trúc dữ liệu toàn cục, giao diện hệ thống, các ràng buộc hiệu năng thì khó chắc chắn được và thường bị hiểu sai. Các phép kiểm thử hồi quy (lặp lại phép thử quá khứ để đảm bảo rằng những sửa đổi không đưa lỗi vào phần mềm) không thể nào tiến hành được vì không có tải liệu ghi lại việc kiểm thử. Chúng ta đang tiến hành *việc bảo trì phi cấu trúc* và phải trả giá (lãng phí công sức, chán nản …)

Nếu tồn tại một cấu hình phần mềm đầy đủ, thì bảo trì bắt đầu với việc đánh giá về tài liệu thiết kế. Cấu trúc quan trọng, hiệu năng, các đặc trưng giao diện của phần mềm được xác định. Tác động của các sửa đổi hay sửa chữa sẽ được thẩm định. Bản thiết kế sẽ được sửa đổi và xét duyệt lại.



H4.7 Bảo trì có cấu trúc so với bảo trì phi cấu trúc

Dãy các sự kiện trong hình vẽ này lập nên *bảo trì có cấu trúc* và xuất hiện như kết quả của việc áp dụng phương pháp luận kỹ nghệ phần mềm.

3.2.2 Chi phí bảo trì

Chi phí cho việc bảo trì phần mềm đã tăng dần trong 20 năm qua, trong những năm 70, việc bảo trì chiếm khoảng 35 % đến 40 % ngân sách phần mềm, con số này nhẩy lên xấp xỉ 60 % trong những năm 1980.

Chi phí về tiền là mối quan tâm hiển nhiên của chúng ta . Tuy nhiên, những chi phí ít thấy khác cuối cùng lại có thể là nguyên nhân cho mối quan tâm lớn hơn. Các chi phí vô hình khác bao gồm : Sự không thoả mãn của khách hàng khi các yêu cầu sửa chữa hay thay đổi có vẻ hợp lý lại không được đề cập tới một cách hợp thời. Sự suy giảm chất lượng phần mềm tổng thể xem như kết quả của những thay đổi tạo thêm lỗi trong những phần mềm đã được bảo trì. Biến động đột ngột xảy ra trong nỗ lực phát triển khi nhân viên bị kéo sang làm công việc bảo trì.

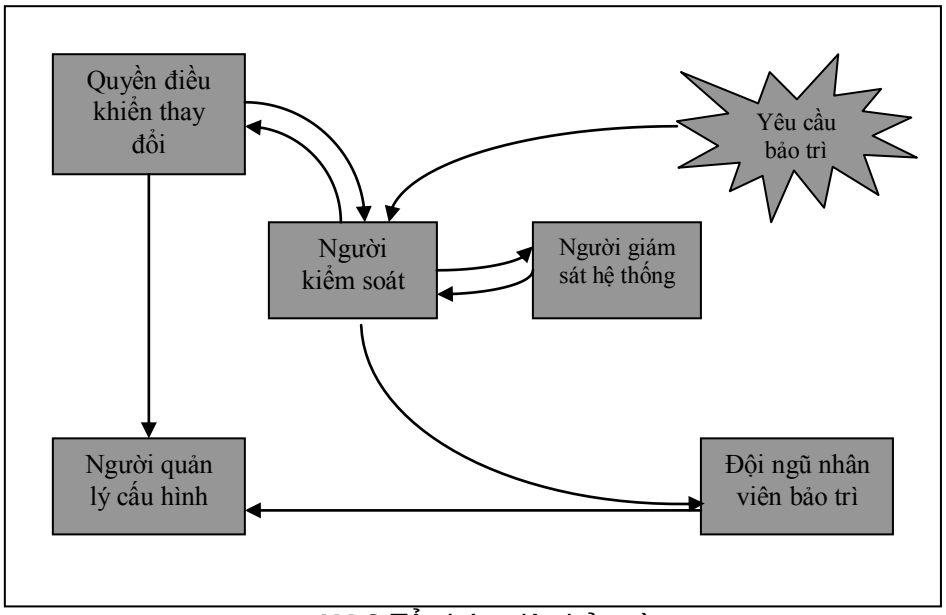
Chi phí cuối cùng cho việc bảo trì phần mềm tăng rất nhiều theo hiệu suất, điều thường gặp phải khi bảo trì chương trình cũ, đã có báo cáo về việc giảm hiệu suất 40:1 tức là nỗ lực phát triển tốn 25$/ 1 dòng mã cho việc xây dựng thì có thể tốn tới 1000$ cho mỗi dòng cần bảo trì. Chi phí có thể tăng lên theo hàm mũ nếu cách tiếp cận phát triển phần mềm nghèo nàn (tức là thiếu kỹ nghệ phần mềm) được sử dụng và người hay nhóm dùng cách tiếp cận này còn chưa sẵn có để thực hiện việc bảo trì. Việc thiếu kiểm soát và kỷ luật trong các hoạt động phát triển kỹ nghệ phần mềm gần như bao giờ cũng biến thành vấn đề trong bảo trì phần mềm.

3.3 Tổ chức bảo trì

Mặc dầu tổ chức bảo trì chính thức không nhất thiết phải thành lập, nhưng một sự uỷ quyền không chính thức thì tuyệt đối cần cho mọi nhà phát triển phần mềm nhỏ. Một sơ đồ như vậy được minh hoạ trong hình dưới. Các yêu cầu bảo trì được chuyển qua kênh ***người kiểm soát bảo trì***, người chuyển tiếp từng yêu cầu đánh giá cho ***người giám sát hệ thống***. Người giám sát hệ thống là thành viên của của bộ phận kỹ thuật, những người đã được trao trách nhiệm phải trở nên am hiểu từng tập con nhỏ nhất trong các chương trình sản phẩm. Một khi việc đánh giá đã được tiến hành thì người có thẩm quyền điều khiển thay đổi phải xác định hành động cần tiến hành.

Tổ chức được gợi ý trên nhằm giảm bớt những lẫn lộn và cải tiến luồng hoạt động bảo trì. Vì yêu cầu bảo trì đều trút về một cá nhân (hoặc nhóm) nên những thay đổi không được thừa nhận dễ gây ra lỗi sẽ khó xảy ra. Vì ít nhất một cá nhân bao giờ cũng có một sự am hiểu nào đó với chương trình, nên yêu cầu thay đổi có thể được thẩm định nhanh chóng hơn.

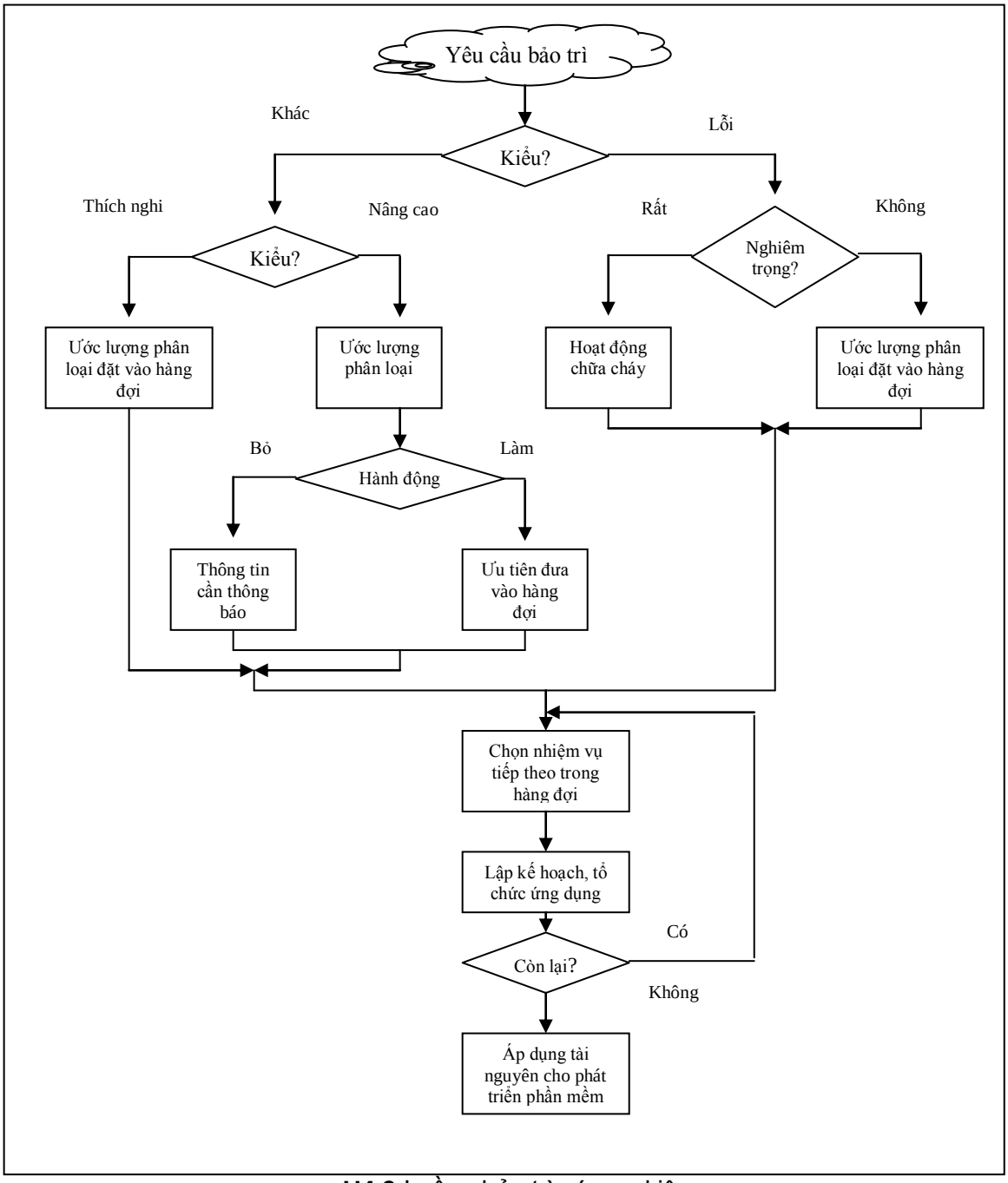
Mỗi tiêu đề công việc trên đều thiết lập nên một lĩnh vực trách nhiệm cho bảo trì. Người kiểm soát và quyền điều khiển sự thay đổi có thể là một người hay với các hệ thống lớn có thể là một nhóm các nhà quản lý và nhân viên kỹ thuật cấp cao. Khi các trách nhiệm được trao hết để bắt đầu hoạt động bảo trì thì sự lẫn lộn được giảm đi rất nhiều. Nhưng quan trọng hơn cả là việc xác định trách nhiệm sớm có thể kiềm chế bất kể cảm giác khó chịu nào vẫn thường xảy ra khi một người bị kéo ra khỏi nỗ lực phát triển để tiến hành công việc bảo trì.



H4.8 Tổ chức việc bảo trì

3.4 Luồng sự kiện

Dãy các sự kiện cuất hiện như một yêu cầu bảo trì được vẽ ở hình dưới. Yêu cầu đầu tiên là xác định kiểu bảo trì cần được tiến hành. Trong nhiều trường hợp, người dùng có thể xét yêu cầu như một chỉ dẫn về lỗi phần mềm (*bảo trì sửa chữa*) trong khi người phát triển có thể coi yêu cầu đó là *thích nghi* hoặc *nâng cấp*. Nếu tồn tại các ý kiến khác nhau thì cần thương lượng về giải pháp.



H4.9 Luồng bảo trì các sự kiện

Một yêu cầu cho bảo trì sửa lỗi bắt đầu với một với một ước lượng về mức nghiêm trọng của lỗi. Nếu lỗi nghiêm trọng xuất hiện thì nhân sự sẽ được phân bổ theo chỉ thị của người giám sát hệ thống và việc phân tích vấn đề được bắt đầu ngay lập tức. Với những lỗi ít nghiêm trọng hơn yêu cầu về bảo trì sửa chữa được ước lượng, phân loại và lập lịch cùng với các nhiệm vụ khác. Trong một số trường hợp một lỗi có thể nghiêm trọng đến mức việc kiểm soát

thông thường về bảo trì tạm thời bị ngưng lại. Chương trình phải được sửa đổi lập tức, cách chữa cháy này cho bảo trì sửa chữa chỉ được dành riêng cho các khủng hoảng và nên biểu thị cho một số phần trăm rất nhỏ trong hoạt động bảo trì. Sau khi giải quyết xong khủng hoảng thì những hoạt động này phải được tiến hành để bảo đảm rằng những lỗi hiện tại sẽ không lan truyền các vấn đề nghiêm trọng hơn.

Các yêu cầu về bảo trì thích nghi và hoàn thiện thì đi theo một con đường khác. Việc thích nghi được ước lượng và phân loại trước khi được đặt vào hàng đợi các hoạt động bảo trì. Việc nâng cấp cũng trải qua cùng ước lượng này, tuy nhiên không phải tất cả các yêu cầu nâng cao đều được tiến hành. Những sự nâng cấp cần được tiến hành cũng đặt vào hàng đợi. Độ ưu tiên cho mỗi yêu cầu đều được thiết lập và công việc cần thiết sẽ được lên lịch dường như nó là một nỗ lực phát triển khác, nếu một số ưu tiên thật cao được nêu ra thì công việc có thể bắt đầu ngay lập tức.

Trong thực tế việc bảo trì phần mềm là kỹ nghệ phần mềm được áp dụng đệ quy. Sự nhấn mạnh sẽ dịch chuyển theo từng kiểu bảo trì nhưng cách tiếp cận toàn bộ cũng không thay đổi. Sự kiện cuối cùng trong luồng bảo trì phần mềm và việc xét duyệt làm hợp lệ lại tất cả các phần tử của cấu hình phần mềm.

3.5 Bảo trì chương trình xa lạ

Gần như mọi tổ chức phần mềm chín muồi đều phải duy trì các chương trình đã phát triển từ 15 năm trước hay hơn nữa. Những chương trình như vậy đôi khi được gọi là chương trình xa lạ vì không còn nhân viên kỹ thuật nào tiếp tục làm việc phát triển chương trình đó nữa hoặc không áp dụng được phương pháp luận phát triển nào do đó gây ra kiến trúc và dữ liệu nghèo nàn, tài liệu thì không đầy đủ và việc ghi lại những thay đổi trong quá khứ thì rất sơ sài.

Ngay từ đầu phần này chúng ta đã thảo luận về sự cần thiết với người giám sát hệ thống - người đã quen thuộc đối với một tập con các chương trình cần phải bảo trì. Việc làm quen với các chương trình này đã được tiến hành bằng cách dùng cách tiếp cận kỹ nghệ phần mềm có điều kiện thuận tiện là có cấu hình phần mềm đầy đủ và thiết kế tốt. Vậy phải làm gì với các chương trình xa lạ?

1. Nghiên cứu chương trình trước khi bạn đi vào, cố gắng có đươc nhiều thông tin nền tảng nhất có thể được

2. Cố gắng quên thuộc với luồng điều khiển của toàn bộ chương trình, bỏ qua chi tiết mã hoá ban đầu. Sẽ rất ích lợi nếu bạn tự vẽ ra được biểu đồ cấu trúc và sơ đồ khối mức cao nếu chưa có.

3. Ước lượng tính hợp lý của tài liệu hiện có, đưa thêm vào lời giải thích của bạn trong bản in chương trình gốc nếu bạn nghĩ chúng có ích.

4. Dùng các bản in tham khảo chéo tốt, các bảng ký hiệu và các trợ giúp khác mà nói chung cho trình biên dịch hay hợp dịch cung cấp.

5. Thay đổi chương trình với sự thận trọng lớn nhất. Hãy tôn trọng các và định dạng của chương trình nếu có thể được. Hãy chỉ ra trên bản in những lệnh nào bạn đã thay đổi.

6. Đừng huỷ bỏ mã lệnh trừ phi bạn chắc chắn là nó không được dùng tới

7. Đừng cố dùng chung biến tạm thời và bộ nhớ làm việc đã có trong chương trình, hãy thêm vào các biến của riêng mình để tránh rắc rối.

8. Hãy giữ các bản ghi chi tiết về hoạt động và kết quả bảo trì

9. Tránh sự thôi thúc mạnh mẽ vứt chương trình đi và viết lại nó. (Tuy nhiên đôi khi thôi thúc này cũng hợp lý và thực tế).

10. Đừng xen lẫn việc kiểm tra lỗi

Những hướng dẫn trên sẽ giúp cho việc bảo trì các chương trình cũ.

Tóm tắt

Đảm bảo chất lượng phần mềm là một hoạt động bảo trợ được áp dụng lại từng bước trong tiến trình kỹ nghệ phần mềm. SQA bao quát các thủ tục cho việc áp dụng có hiệu quả các phương pháp và công cụ, chiến lược và kỹ thuật kiểm thử, thủ tục cho việc kiểm soát thay đổi, thủ tục để đảm bảo việc tuân thủ các chuẩn và các cơ chế đo, báo cáo.

Mục tiêu chính của việc thiết kế trường hợp kiểm thử là để suy dẫn ra một tập các phép kiểm thử với mục đích phát hiện ra khiếm khuyết trong phần mềm. Những người phát triển phần mềm có kinh nghiệm hay nói " kiểm thử chẳng bao giờ hết, nó chỉ được truyền từ bạn sang khách hàng của bạn. Mọi lúc khách hàng của bạn dùng chương trình thì việc kiểm thử lại được tiến hành" . Bằng cách áp dụng việc thiết kế trường hợp kiểm thử, người kỹ sư phần mềm có thể đạt tới việc kiểm thử phức tạp hơn do vậy phát hiện ra và sửa đổi lỗi các lỗi nhiều hơn khi việc " kiểm thử khách hàng " bắt đầu.

Mục đích của kiểm thử phần mềm là phát hiện ra lỗi. Để hoàn thành mục đích này, một loạt các bước kiểm thử - kiểm thử đơn vị, tích hợp, hợp lệ và hệ thống - cần được lập kế hoạch và thực hiện. Các phép kiểm thử đơn vị và tích hợp tập trung vào kiểm thử chức năng của Modul, việc tổ hợp các Modul vào trong cấu trúc chương trình. Việc kiểm chứng hợp lệ bầy tỏ tính đáp ứng được so với các yêu cầu phần mềm, còn việc kiểm thử hệ thống thì hợp lệ hoá phần mềm một khi nó đã được tổ hợp và trong hệ thống tổng thể.

Bảo trì, giai đoạn cuối cùng trong tiến trình kỹ nghệ phần mềm, tiêu tốn đại đa số chi phí cho phần mềm máy tính. Một số tổ chức phần mềm có thể trở thành khuynh hướng bảo trì khi chương trình được xây dựng càng nhiều, nỗ lực và tài nguyên chi phí cho việc bảo trì phần mềm càng tăng và không thể tiến hành được những dự án mới vì tất cả tài nguyên của họ đều phải dành cho việc bảo trì các chương trình cũ. Bốn kiểu bảo trì được thực hiện trên phần mềm máy tính : Việc *bảo trì sửa chữa* tiến hành để sửa lại những lỗi còn chưa bị phát hiện sau khi phần mềm được đưa vào sử dụng. Việc *bảo trì thích nghi* được áp dụng khi có những thay đổi trong môi trường ngoài thúc đẩy việc sửa đổi phần mềm. Việc *bảo trì hoàn thiện* tổ hợp thêm những nâng cấp do người sử dụng yêu cầu. Cuối cùng, việc *bảo trì phòng ngừa* làm tăng tính bảo trì được và tính tin cậy được trong tương lai, đưa ra một cơ sở cho việc nâng cấp phần mềm trong tương lai.

Củng cố

1. Trình bày các nhân tố chất lượng phần mềm? Theo Anh/Chị có cần bổ xung nhân tố nào nữa không?

2. Nêu ra ba loại chỉ số chất lượng phần mềm? Tác dụng của từng loại?

3. Thế nào là độ tin cậy phần mềm?

4. Những mắt tích cực và hạn chế do SQA đem lại?

5. Nội dung cần thiết trong tài liệu kế hoạch cho SQA ?

6. Nền tảng của việc kiểm thử phần mềm gồm những nội dung gì?

7. Trình bày các chiến lược trong kiểm thử phần mềm?

8. Cách thức tổ chức công việc kiểm thử?

9. Phân biệt kiểm thử đơn vị và kiểm thử tích hợp?

10. Định nghĩa bảo trì phần mềm, trình bày những nội dung liên quan: bảo trì cấu trúc, chi phí bảo trì, tổ chức việc bảo trì, luồng sự kiện của công tác bảo trì?

11. Những lưu ý cần thiết khi phải bảo trì một chương trình "xa lạ"?

CHƯƠNG 5

LẬP TRÌNH HIỆU QUẢ

1. Các đặc trưng ngôn ngữ lập trình

Ngôn ngữ lập trình là phương pháp để liên lạc giữa con người và máy tính. ***Lập trình*** là một hoạt động của con người, là sự liên lạc thông qua ngôn ngữ lập trình, là một bước cốt lõi trong tiến trình kỹ nghệ phần mềm.

1.1 Đặc trưng tâm lý của ngôn ngữ lập trình

Trong cuốn sách *Tâm lý phần mềm*, tác giả Ben Shneiderman đã viết về vai trò của nhà tâm lý phần mềm như sau: "họ tập trung vào mối quan tâm của con người như tính dễ dùng, đơn giản khi học, nâng cao độ tin cậy, giảm tần suất lỗi và tăng sự thoả mãn với người dùng, trong khi không quên tính hiệu quả của máy, khả năng phần mềm và sự ràng buộc phần cứng". Trong khi đó, người thiết kế ngôn ngữ lập trình thường bóp cách tiếp cận tới vấn đề sao cho cách tiếp cận khớp với những ràng buộc riêng do ngôn ngữ lập trình áp đặt.

Vì nhân tố con người có tầm quan trọng chủ chốt trong việc thiết kế ngôn ngữ lập trình nên các đặc trưng tâm lý của ngôn ngữ có tác động mạnh mẽ lên sự thành công của việc thiết kế trong khi dịch và cài đặt.

Một số đặc trưng của tâm lý xuất hiện như kết quả của việc thiết kế ngôn ngữ lập trình. Mặc dầu những đặc trưng này không đo được theo bất cứ cách thức định lượng nào, nhưng chúng ta thừa nhận biểu hiện của chúng trong mọi ngôn ngữ lập trình.

a. Tính đồng đều: chỉ ra mức độ theo đó ngôn ngữ ký pháp nhất quán.

b. Tính mơ hồ: ngôn ngữ lập trình được người lập trình cảm nhận. Trình biên dịch bao giờ cũng diễn giải một câu lệnh theo một cách. Nhưng độc giả có thể hiểu câu lệnh đó theo cách khác. Tại đây có sự mơ hồ tâm lý. Việc thiếu tính đồng đều và sự mơ hồ tâm lý thường đi kèm nhau. Nếu ngôn ngữ lập trình để lộ ra những khía cạnh tiêu cực của các đặc trưng này thì chương trình nguồn sẽ khó đọc và việc dịch từ thiết kế ra dễ sinh lối nhiều hơn.

c.Tính gọn gàng: chỉ dẫn về khối lượng thông tin hướng chương trình mà con người phải ghi nhớ. Trong các thuộc tính ngôn ngữ đo tính gọn gàng có:

- Mức độ ngôn ngữ hỗ trợ cho các kết cấu có cấu trúc và "giải quyết việc khó" theo logic
- Loại từ khoá và cách viết tắt có thể được dùng
- Sự phong phú của các kiểu dữ liệu và đặc trưng mặc định
- Số các phép toán logic và số học
- Số các hàm có sẵn

Ví dụ: APL là một ngôn ngữ lập trình gọn gàng ngoại lệ, thành khó đọc và khó hiểu.

Các đặc trưng ký ức con người có tác động mạnh mẽ đến cách ta dùng ngôn ngữ. Ký ức và việc nhận dạng của con người có thể chia thành hai lĩnh vực *toàn thái* và *tuần tự*. Ký ức toàn thái chỉ cho phép chúng ta nhớ và nhận lại mọi thứ như một tổng thể. (như chúng ta nhận ra khuôn mặt người ngay tức thì, nhưng chúng ta không có ý thức về từng phần riêng biệt trên khuôn mặt trước khi nhận dạng). Ký ức tuần tự, cung cấp một phương tiện để nhớ lại phần tử tiếp trong một dãy (như dòng tiếp theo trong bài hát, khi được cho những dòng trước đó). Mỗi

đặc trưng này đều có ảnh hưởng đến đặc trưng ngôn ngữ lập trình vẫn được gọi là tính cục bộ và tính tuyến tính.

d. Tính cục bộ: là đặc trưng toàn thái của ngôn ngữ lập trình. Tính cục bộ được làm nổi bật khi các câu lệnh có thể được tổ hợp thành các *khối*, khi các kết cấu có cấu trúc có thể được cài đặt trực tiếp, khi thiết kế và chương trình gốc đều mang tính module. Một đặc trưng của ngôn ngữ hỗ trợ hay khuyến khích cho xử lý biệt lệ đều vi phạm vào tính cục bộ này.

e. Tính tuyến tính: là một đặc trưng tâm lý có liên quan chặt chẽ với khái niệm bảo trì của lĩnh vực chức năng. Tức là, nhận biết con người được thuận lợi khi gặp một dãy tuyến tính các thao tác logic. Những nhánh xảy ra (các chu trình lớn) vi phạm tính tuyến tính của xử lý. Một lần nữa, việc cài đặt thông tin trực tiếp cho các kết cấu có cấu trúc trợ giúp cho tuyến tính của ngôn ngữ lập trình.

Khả năng học một ngôn ngữ mới của chúng ta bị ảnh hưởng bởi truyền thống. Các kết cấu là tương tự nhau, hình dạng thì tương thích và cảm giác về định dạng ngôn ngữ lập trình được bảo toàn.

Các đặc trưng tâm lý của ngôn ngữ lập trình có ý nghĩa quan trọng tới khả năng của chúng ta trong việc học, áp dụng và duy trì chúng. Tóm lại, ngôn ngữ lập trình tạo ra màu sắc, cho chúng ta cách nghĩ về chương trình và giới hạn cố hữu cách chúng ta liên lạc với máy tính.

1.2 Mô hình cú pháp và ngữ nghĩa

Shniderman đã phát triển một mô hình cú pháp - ngữ nghĩa cho tiến trình lập trình có liên quan đến việc xem xét các bước lập trình. Khi người lập trình các phương pháp kỹ nghệ phần mềm (như phân tích yêu cầu thiết kế) vốn độc lập với ngôn ngữ lập trình thì động tới vấn đề tri thức ngữ nghĩa. Tri thức ngữ nghĩa mặt khác lại là độc lập với ngôn ngữ, tập trung vào các đặc trưng của ngôn ngữ xác định.

Về các kiểu tri thức này, tri thức ngữ nghĩa là khó thu nhận được hơn cả và đòi hỏi dùng nhiều trí tuệ. Tất cả các bước kỹ nghệ phần mềm trước phần lập trình đều dùng rất nhiều tri thức ngữ nghĩa. Bước lập trình áp dụng tri thức cú pháp vốn là “bất kỳ và theo lệnh” được học theo kiểu vẹt. Khi học một ngôn ngữ lập trình mới thì thông tin cú pháp mới được thêm vào ký ức. Nhiều vấn đề liên quan tới phần mềm máy tính đã không là quan trọng do việc thiếu tri thức cú pháp, nhưng lại quan trọng trong phạm vi tri thức ngữ nghĩa và khả năng của chúng ta để áp dụng nó. Mục tiêu của kỹ nghệ phần mềm là mở rộng tri thức về ngữ nghĩa của việc phát triển phần mềm.

1.3 Hướng quan điểm kỹ nghệ

Cách nhìn kỹ nghệ phần mềm về các đặc trưng của ngôn ngữ lập trình tập trung vào nhu cầu xác định dự án phát triển phần mềm riêng. Mặc dầu người ta vẫn cần các yêu cầu riêng cho chương trình gốc, có thể thiết lập được một tập hợp tổng quát những đặc trưng kỹ nghệ:

(1) dễ dịch thiết kế sang chương trình

(2) có trình biên dịch hiệu quả

(3) khả chuyển chương trình gốc

(4) có sẵn công cụ phát triển

(5) dễ bảo trì

Bước lập trình bắt đầu sau khi thiết kế chi tiết đã được xác định, xét duyệt và sửa đổi nếu cần. Về lý thuyết, việc sinh chương trình gốc từ một đặc tả chi tiết nên là trực tiếp. *Dễ dịch thiết kế sang chương trình* đưa ra một chỉ dẫn về việc một ngôn ngữ lập trình phản xạ gần gũi đến mức nào cho một biểu diễn thiết kế. Một ngôn ngữ cài đặt trực tiếp cho các kết cấu có cấu trúc,

các cấu trúc dữ liệu phức tạp, vào/ra đặc biệt, khả năng thao tác bit, và kết cấu hướng sự vật sẽ làm cho việc dịch từ thiết kế sang chương trình gốc dễ hơn nhiều (nếu các thuộc tính này được xác định trong thiết kế).

Mặc dầu những tiến bộ nhanh chóng trong tốc độ xử lý và mật độ nhớ đã bắt đầu làm giảm nhẹ nhu cầu "chương trình siêu hiệu quả", nhiều ứng dụng vẫn còn đòi hỏi các chương trình nhanh, "gọn" (yêu cầu bộ nhớ thấp). Các ngôn ngữ với trình biên dịch tối ưu có thể là hấp dẫn nếu hiệu năng phần mềm là yêu cầu chủ chốt.

a. Tính khả chuyển chương trình gốc: là một đặc trưng của ngôn ngữ lập trình có thể được hiểu theo ba cách khác nhau:

(1) Chương trình gốc có thể được chuyển từ bộ xử lý này sang bộ xử lý khác và từ trình biên dịch nọ sang trình biên dịch kia với rất ít hoặc không sửa đổi gì.

(2) Chương trình gốc vẫn không thay đổi ngay cả khi môi trường của nó thay đổi (như việc cài đặt bản mới của hệ điều hành).

(3) Chương trình gốc có thể được tích hợp vào trong các bộ trình phần mềm khác nhau với ít hay không cần thay đổi gì vì các đặc trưng của ngôn ngữ lập trình.

Trong số ba cách hiểu về tính khả chuyển này thì cách thứ nhất là thông dụng nhất. Việc chuẩn hóa (do tổ chức quốc tế IFO/hoặc Viện tiêu chuẩn quốc gia Mỹ ANSI) nhằm nâng cao tính khả chuyển ngôn ngữ lập trình.

b. Tính sẵn có của công cụ phát triển: có thể làm ngắt bớt thời gian cần để sinh ra chương trình gốc và có thể cải thiện chất lượng của chương trình. Nhiều ngôn ngữ lập trình có thể cần tới một loạt công cụ kể cả trình biên dịch gỡ lỗi, trợ giúp chương trình gốc, các tiện nghi soạn thảo có sẵn, các công cụ kiểm soát chương trình gốc, thư viện chương trình con mở rộng trong nhiều lĩnh vực ứng dụng, các trình duyệt, trình biên dịch chéo cho phát triển bộ xử lý, khả năng bộ xử lý macro, công cụ kỹ nghệ ngược và những công cụ khác. Trong thực tế, khái niệm về "môi trường phát triển phần mềm" tốt (bao hàm cả công cụ) đã được thừa nhận như nhân tố đóng góp cho chính kỹ nghệ phần mềm thành công.

c. Tính dễ bảo trì của chương trình gốc có tầm quan trọng chủ chốt cho tất cả các nỗ lực phát triển phần mềm không tầm thường. Việc bảo trì không thể được tiến hành chừng nào người ta chưa hiểu được phần mềm. Các yếu tố của cấu hình phần mềm (như tài liệu thiết kế) đưa ra nền tảng cho việc hiểu biết, nhưng cuối cùng thì chương trình gốc vẫn phải được đọc và sửa đổi theo những thay đổi trong thiết kế.

Tính dễ dịch thiết kế sang chương trình là một yếu tố quan trọng để dễ bảo trì chương trình gốc. Bên cạnh đó, các đặc trưng tự làm tài liệu của ngôn ngữ (như chiều dài được phép của tên gọi, định dạng nhãn, định nghĩa kiểu, cấu trúc dữ liệu) có ảnh hưởng mạnh đến tính bảo trì.

1.4 Việc chọn ngôn ngữ

Việc chọn ngôn ngữ lập trình cho một dự án riêng phải tính tới cả đặc trưng kỹ nghệ và tâm lý. Tuy nhiên, vấn đề liên quan đến việc chọn lựa có thể bàn tới chỉ khi ngôn ngữ là có sẵn hay được quy định bởi người yêu cầu.

Meek gợi ý một quan điểm tổng quát khi phải chọn một ngôn ngữ lập trình:" nghệ thuật chọn ngôn ngữ là bắt đầu từ vấn đề, quyết định xem cái gì là yêu cầu của nó và tầm quan trọng tương đối của chúng, vì có thể sẽ không thể nào thoả mãn được chúng như nhau (với một ngôn ngữ)...các ngôn ngữ có sẵn nên được ứng với danh sách yêu cầu...".

Các tiêu chuẩn được áp dụng khi đánh giá về ngôn ngữ có sẵn. Miền ứng dụng của dự án là một tiêu chuẩn hay được áp dụng nhất trong việc áp dụng ngôn ngữ.

Xu hướng phát triển phần mềm hướng đối tượng xuyên suốt phần lớn các miền ứng dụng đã mở ra nhiều ngôn ngữ mới và các dị bản ngôn ngữ quy ước. Các ngôn ngữ lập trình hướng đối tượng được dùng rộng rãi nhất là Smalltalk, C++, Objected Pascal, và nhiều ngôn ngữ khác.

Sự phát triển gia tăng của các ngôn ngữ lập trình mới và tốt hơn vẫn đang phát triển. Mặc dù có nhiều ngôn ngữ khá hấp dẫn, tuy nhiên đôi khi tốt hơn cả vẫn là chọn ngôn ngữ yếu hơn (cũ) nhưng đã có tài liệu chắc chắn và phần mềm hỗ trợ, quen thuộc với nhiều người trong nhóm phát triển phần mềm và đã từng áp dụng thành công trong quá khứ. Tuy nhiên, nên có đánh giá kỹ càng về các ngôn ngữ mới và việc chuyển dịch từ ngôn ngữ cũ sang ngôn ngữ mới nếu có, với việc thừa nhận sự kháng cự về tâm lý đối với thay đổi thường hay gặp phải trong mọi tổ chức.

1.5 Ngôn ngữ lập trình và kỹ nghệ phần mềm

Bất kể khuôn cảnh kỹ nghệ phần mềm nào, ngôn ngữ lập trình sẽ có tác động tới việc vạch kế hoạch dự án, phân tích, thiết kế, lập trình, kiểm thử và bảo trì.

Trong khi lập kế hoạch dự án, hiếm khi người ta tiến hành xem xét các đặc trưng kỹ thuật của ngôn ngữ lập trình. Tuy nhiên, việc lập kế hoạch cho các công hỗ trợ có liên quan tới tài nguyên cần có, có thể yêu cầu rằng một trình biên dịch chuyên dụng (và phần mềm liên kết) hay môi trường lập trình được xác định.

Một khi các yêu cầu phần mềm đã được thiết lập, thì các đặc trưng ngôn ngữ lập trình ứng cử viên trở thành quan trọng hơn. Nếu cần tới cấu trúc dữ liệu phức tạp thì ngôn ngữ với sự hỗ trợ cho các cấu trúc dữ liệu phức tạp (như Pascal hay ngôn ngữ khác) nên được tính tới một cách cẩn thận. Nếu cần khả năng cao, như thời gian thực, thì ngôn ngữ được thiết kế cho ứng dụng thời gian thực như ADA hay hiệu quả về tốc độ nhớ như C có thể được xác định. Nếu cần đưa ra nhiều báo cáo hay thao tác tệp thì các ngôn ngữ như COBOL, VB hay PRG có thể thích hợp. Một cách lý tưởng, các yêu cầu phần mềm nên kết tinh việc lựa chọn ngôn ngữ thích hợp nhất cho việc xử lý cần thực hiện. Tuy nhiên trong thực hành, một ngôn ngữ thường được chọn lựa bởi vì "nó là ngôn ngữ duy nhất mà chúng ta có chạy trên máy của mình".

Chất lượng của thiết kế phần mềm được thiết lập theo cách độc lập với các đặc trưng ngôn ngữ lập trình (một ngoại lệ đáng lưu ý là thiết kế hướng sự vật). Tuy nhiên thuộc tính ngôn ngữ đóng vai trò quan trọng trong chất lượng của thiết kế được cài đặt và ảnh hưởng tới cách thiết kế được xác định. Một số độ đo định tính, định lượng của thiết kế tốt, các khái niệm về tính module và sự độc lập module cũng được nhấn mạnh. Các đặc trưng kỹ thuật của nhiều ngôn ngữ lập trình có thể ảnh hưởng tới những khái niệm này trong việc cài đặt thiết kế.

Thiết kế dữ liệu cũng có thể bị ảnh hưởng bởi đặc trưng ngôn ngữ. Các ngôn ngữ lập trình như ADA, C++, Smalltalk đều hỗ trợ cho khái niệm về kiểu dữ liệu trừu tượng - một công cụ quan trọng trong thiết kế và đặc tả dữ liệu. Các ngôn ngữ thông dụng khác như Pascal cho phép định nghĩa các kiểu dữ liệu do người dùng xác định và việc cài đặt trực tiếp danh sách móc nối và những cấu trúc dữ liệu khác. Các tính năng này cung cấp cho người thiết kế phạm vi rộng hơn trong bước thiết kế sơ bộ và chi tiết.

Ảnh hưởng của các đặc trưng ngôn ngữ tới các bước thiết kế bao gồm khó khăn trong việc đánh giá kiểm thử phần mềm. Các ngôn ngữ trực tiếp hỗ trợ cho các kết cấu có cấu trúc có khuynh hướng giảm bớt độ phức tạp của chương trình, do đó một cách nào đó làm cho nó dễ dàng kiểm thử. Các ngôn ngữ hỗ trợ cho việc đặc tả chương trình con và thủ tục ngoài (như Fotran) thường cho việc kiểm thử ít tích hợp ít sinh lỗi hơn. Mặt khác một số đặc trưng kỹ thuật của ngôn ngữ có thể gây trở ngại cho việc kiểm thử. Ví dụ: cấu trúc khối trong ALGOL có thể được xác định theo cách làm mất dữ liệu trung gian khi việc ra khỏi khối xuất hiện, do đó làm

cho trạng thái chương trình khó xác nhận hơn. Giống như kiểm thử, hiệu quả của các đặc trưng ngôn ngữ lập trình về mặt bảo trì phần mềm cũng chưa được hiểu đầy đủ.

2. Nền tảng của ngôn ngữ lập trình

Nền tảng của ngôn ngữ lập trình được thể hiện trong ngữ cảnh bốn chủ đề đại thể: định kiểu, cơ chế chương trình con, cấu trúc điều khiển và hỗ trợ cho việc tiếp cận hướng đối tượng. Mọi ngôn ngữ lập trình đều có thể được đặc trưng theo những chủ đề này và chất lượng của một ngôn ngữ lập trình có thể được đánh giá theo điểm mạnh và điểm mạnh liên quan tới từng chủ đề.

2.1 Kiểu dữ liệu và định kiểu dữ liệu

Ngày nay ích lợi của các ngôn ngữ lập trình được đánh giá không chỉ ở cú pháp và sự phóng khoáng của các kết cấu thủ tục. Định kiểu dữ liệu và các kiểu dữ liệu đặc biệt được ngôn ngữ lập trình hỗ trợ là khía cạnh quan trọng của chất lương ngôn ngữ.

Kiểu dữ liệu và định kiểu dữ liệu được mô tả là : một lớp các đối tượng dữ liệu cùng với một tập các phép toán để tạo ra và thao tác trên chúng. Một đối tượng dữ liệu kế thừa một tập các thuộc tính nền tảng của kiểu dữ liệu mà nó thuộc vào. Một đối tượng dữ liệu có thể lấy một giá trị nằm bên trong miền giá trị hợp lệ cho kiểu dữ liệu đó và có thể bị các phép toán của kiểu dữ liệu đó thao tác.

Các kiểu dữ liệu đơn trải trên một miền rộng bao gồm các kiểu số (nguyên phức, dấu phẩy động), kiểu liệt kê (như kiểu dữ liệu do người dùng định nghĩa có trong Pascal), kiểu logic (true hay false) và kiểu xâu string (như dữ liệu chữ số). Các kiểu dữ liệu phức tạp hơn bao gồm các cấu trúc dữ liệu trải qua hết từng mảng một chiều đơn giản (vectơ) cho tới cấu trúc danh sách, mảng và bản ghi đa phức tạp.

Các phép toán có thể được thực hiện trên một kiểu dữ liệu đặc biệt và theo cách thức mà trong đó các kiểu khác có thể được thao tác sẽ được điều khiển bởi việc *kiểm tra kiểu*, vốn được cài đặt bên trong trình biên dịch hay thông dịch ngôn ngữ lập trình. Fairley đưa ra năm mức kiểm tra kiểu thường gặp trong các ngôn ngữ lập trình:

Mức 0: Phi kiểu

Mức 1: Bó buộc kiểu tự động

Mức 2: Kiểu hỗn hợp

Mức 3: Kiểm tra kiểu giả mạnh

Mức 4: Kiểm tra kiểu mạnh

2.2 Chương trình con

Chương trình con là một thành phần của chương trình dịch được tách biệt có chứa dữ liệu và cấu trúc điều khiển. *Module* là cách biểu hiện tổng quát của chương trình con.

Tuỳ theo ngôn ngữ lập trình mà một chương trình con có thể được gọi là trình con, thủ tục, hàm hay bất kỳ tên gọi đặc biệt nào. Bất kể đến tên của nó, chương trình con vẫn bộc lộ ra một tập các đặc trưng tổng quát:

(1) Phần mô tả có chứa tên của nó và mô tả giao diện.

(2) Phần cài đặt có chứa dữ liệu và cấu trúc điều khiển

(3) Một cơ chế kích hoạt làm cho chương trình con được gọi tời từ một nơi nào đó khác trong chương trình

Trong các ngôn ngữ lập trình quy ước, mỗi chương trình con bản thân đều là một thực thể, vận hành trên dữ liệu theo một cách được chỉ đạo bởi cấu trúc điều khiển của chương trình lớn hơn. Trong các ngôn ngữ lập trình hướng đối tượng, cách nhìn lớp chương trình con được thay thế bởi đối tượng.

2.3 Cấu trúc điều khiển

Tại mức cơ bản, mọi ngôn ngữ lập trình hiện đại đều cho phép người lập trình biểu diễn sự tuần tự, tuyển chọn và lặp – các kết cấu logic lập trình có cấu trúc. Phần lớn các ngôn ngữ hiện đại đều đưa ra một cú pháp cho đặc tả trực tiếp về if-then-else, do-while, repeat-untill...Các ngôn ngữ khác như LISP và APL đòi hỏi người lập trình phải mô phỏng các kết cấu bên trong giới hạn cú pháp của ngôn ngữ đó.

Bên cạnh các kết cấu thủ tục cơ sở của lập trình có cấu trúc, các cấu trúc điều khiển khác cũng có thể. Đệ quy tạo ra sự kích hoạt lần thứ hai của chương trình con trong lần kích hoạt thứ nhất. Tức là, chương trình con gọi tới hay kích hoạt bản thân nó như một phần của thủ tục đã xác định. Tương tranh đưa ra sự hỗ trợ cho việc tạo ra nhiều nhiệm vụ, đồng bộ hóa các nhiệm vụ này và liên lạc nói chung giữa các nhiệm vụ ấy. Tính năng ngôn ngữ này là vô giá khi phải thực hiện các ứng dụng hệ thống hay thời gian thực. Có tính năng ngôn ngữ lập trình đặt bẫy các điều kiện lỗi hệ thống hoặc của người dùng rồi truyền điều khiển cho bộ điều khiển đặc biệt để xử lý.

2.4 Cách tiếp cận hướng đối tượng

Về lý thuyết, việc tạo ra các đối tượng và kết cấu của phần mềm hướng đối tượng có thể được thực hiện bằng cách dùng bất kỳ ngôn ngữ lập trình quy ước nào (như C hay Pascal). Nhưng trong thực tế, việc hỗ trợ cho các cách tiếp cận hướng đối tượng nên được xây dựng trực tiếp bên trong ngôn ngữ lập trình, sẽ được dùng để cài đặt thiết kế hướng đối tượng. Ngôn ngữ lập trình hướng đối tượng nên được cung cấp sự hỗ trợ trực tiếp cho định nghĩa lớp, kế thừa, bao gói và truyền thông báo.

Định nghĩa về lớp là cơ sở cho cách tiếp cận hướng đối tượng. Ngôn ngữ lập trình hướng đối tượng định nghĩa tên một lớp và xác định các thành phần chung và riêng của lớp đó. Một lớp mới có thể được suy dẫn từ định nghĩa lớp cơ sở. Các đối tượng được suy từ lớp mới có thể dùng tất cả các phương pháp được xác định cho “lớp cha”. Định nghĩa về lớp bao quát bao quát cả trừu tượng dữ liệu và thành phần chương trình vận hành trên chúng.

Các chi tiết cài đặt và thuật ngữ cho định nghĩa lớp, kế thừa, bao gói và truyền thông báo sẽ thay đổi từ ngôn ngữ nọ sang ngôn ngữ kia, nhưng khái niệm nền tảng về lớp vẫn không thay đổi. Tương tự kế thừa, bao gói và truyền thông báo được cài đặt với một cú pháp khác, nhưng cùng nền tảng. Mỗi kết cấu sẽ có sẵn trong bất kỳ ngôn ngữ nào thực sự hướng đối tượng.

2.5 Các lớp ngôn ngữ

Có hàng trăm ngôn ngữ lập trình đã được sử dụng vào lúc này lúc khác trong nhưng nỗ lực phát triển phần mềm nghiêm chỉnh.

Có 4 thế hệ ngôn ngữ lập trình đại diện:

1. **Ngôn ngữ thế hệ thứ nhất:** Thế hệ ngôn ngữ thứ nhất lập trình theo mức mã máy (một số công việc với ngôn ngữ thế hệ thứ nhất vẫn còn tiếp tục đến ngày nay). Chương trình mã máy và dạng tương đương của nó dễ học hơn cho con người. Hợp ngữ biểu thị cho thế hệ ngôn ngữ thứ nhất. Các ngôn ngữ phụ thuộc

máy này biểu hiện mức độ trừu tượng thấp nhất mà một chương trình có thể được biểu diễn.

2. **Ngôn ngữ thế hệ hai:** Ngôn ngữ thế hệ hai đã được phát triển từ cuối những năm 1950 và đầu những năm 1960 và phục vụ như nền tảng cho mọi ngôn ngữ lập trình hiện đại (thế hệ ba). Các ngôn ngữ thế hệ hai được đặc trưng bởi việc sử dụng rộng rãi một thư viện phần mềm quen thuộc: Fortran, Cobol, Algol, Basic và các ngôn ngữ nền tảng.
3. **Ngôn ngữ thế hệ ba** (Ngôn ngữ lập trình có cấu trúc): Ngôn ngữ thế hệ ba, được đặc trưng bởi khả năng cấu trúc dữ liệu và thủ tục mạnh. Các ngôn ngữ lớp này có thể được chia thành ba phạm trù lớn: ngôn ngữ cấp cao vạn năng, ngôn ngữ cấp cao hướng đối tượng và ngôn ngữ chuyên dụng. Ngôn ngữ cao cấp vạn năng: bao gồm PL/I, Pascal, C, Modula-2...
    (a) Các ngôn ngữ hướng đối tượng: Các ngôn ngữ hướng đối tượng làm cho người kỹ sư phần mềm cài đặt được các mô hình phân tích thiết kế tạo ra bằng cách dùng OOA và OOD. Mặc dầu có hàng chục ngôn ngữ hướng đối tượng đã được đưa ra hàng trong thập kỷ qua nhưng chỉ có một số ngôn ngữ có được chỗ đứng có ý nghĩa trên thị trường như dị bản của C (C++, Objective C), Smalltalk và Eiffel.
    (b) Các ngôn ngữ chuyên dụng: Các ngôn ngữ chuyên dụng được đặc trưng bởi các dạng cú pháp bất thường đã được đặc biệt thiế kế cho một ứng dụng riêng. Trong hàng trăm ngôn ngữ chuyên dụng đang được dùng, có một số các ngôn ngữ phổ biến trong kỹ nghệ phần mềm là Lisp, Prolog, Apl, Forth...
4. **Ngôn ngữ thế hệ thứ tư** (4GL – the fourth Generation Technology): Trong toàn bộ lịch sử phát triển phần mềm, chúng ta có ý định phát triển ra chương trình máy tính ở mức trừu tượng ngày càng cao. Các ngôn ngữ thế hệ thứ nhất làm việc ở mức tập lệnh máy, mức trừu tượng thấp nhất có thể. Các ngôn ngữ lập trình thế hế hai và ba đã nâng mức độ biểu diễn chương trình máy tính, nhưng vẫn còn phải xác định thủ tục thuật toán chi tiết và phân biệt. Trong thập kỷ qua, ngôn ngữ thế hệ thứ tư (4GL) đã nâng mức độ trừu tượng lên cao hơn nữa. Ngôn ngữ thế hệ thứ tư, giống như mọi ngôn ngữ nhân tạo khác, đều có chứa một cú pháp phân biệt để biểu diễn điều khiển và cấu trúc dữ liệu. Một 4GL biểu thị những cấu trúc này ở mức độ trừu tượng cao hơn bằng cách xoá bỏ yêu cầu xác định chi tiết thuật toán.

   Các ngôn ngữ thế hệ bốn tổ hợp các đặc trưng *thủ tục* và *phi thủ tục.* Tức là, ngôn ngữ có khả năng cho phép người dùng xác định các điều kiện và hành động tương ứng (thành phần thủ tục) trong khi đồng thời cổ vũ người dùng chỉ ra kết quả mong muốn (thành phần phi thủ tục) rồi áp dụng tri thức chuyên ngành để điền các cho tiết thủ tục.

2.6 Các công cụ lập trình

2.6.1 Công trình phần mềm có máy tính hỗ trợ

CASE là bốn chữ cái đầu của cụm từ tiếng Anh nghĩa là công trình phần mềm được hỗ trợ bởi máy tính. Hiện nay đã có rất nhiều công cụ CASE. Có hai cách để phân loại các công cụ CASE:

(1) Hướng hoạt động: dựa trên hoạt động của các quá trình như: đặc tả yêu cầu, thiết kế, thực hiện...

(2) Hướng chức năng: dựa trên chức năng của các công cụ đó chứ không phải là dựa trên các mục tiêu trợ giúp.

2.6.2 Môi trường phát triển phần mềm

a. Đại cương

Một môi trường phát triển phần mềm là một bộ các công cụ phần cứng và phần mềm chúng được kết lại để sản sinh ra một hệ thống phần mềm trong một miền ứng dụng chuyên biệt.

Có hai điểm quan trọng:

a. Môi trường phát triển phần mềm có thể bao gồm các công cụ phần cứng.

b. Môi trường phát triển phần mềm thường xuyên chuyên dụng hơn là khái quát

Môi trường phát triển phần mềm vận hành trên một hệ thống máy tính host và phần mềm được phát triển nhằm vào một máy tính mục tiêu, có vài lý do vì sao mô hình máy chủ - máy khách lại thích hợp nhất với môi trường phát triển phần mềm:

a. Trong một số trường hợp phần mềm ứng dụng đang được phát triển có thể là dành cho một máy không có tiện ích phần mềm.

b. Máy khách có thể là hướng ứng dụng nó không thích hợp với các môi trường phát triển phần mềm

c. Máy khách có thể được dành cho việc vận hành một ứng dụng đặc biệt và nó phải là có tính ưu tiên trên sự phát triển phần mềm (chẳng hạn như hệ xử lý giao tác)

Ưu điểm của cách thức máy chủ - máy khách là các tiện ích và các thành phần cho người phát triển dùng môi trường đó không thể hợp tác với nhau trong hệ thống ứng dụng được phân phối.

Có thể phân loại các môi trường phát triển phần mềm như sau:

1. **Môi trường lập trình**: Trợ giúp cho lập trình, thử nghiệm, gỡ lỗi. Hạn chế việc xác định yêu cầu, đặc tả, thiết kế phần mềm.

2. **Bàn thợ CASE**: Đây là các môi trường chủ yếu hướng về đặc tả phần mềm và thiết kế. Nó thường chỉ cung cấp một sự trợ giúp lập trình thô sơ (chẳng hạn các ngôn ngữ lập trình thế hệ thứ tư). Nó thường thích hợp với các máy tính cá nhân và kết hợp với các môi trường lập trình.

3. **Môi trường công trình phần mềm**: Nó trợ giúp sản sinh ra các hệ thống lớn, thọ mà chi phí cho việc bảo trì còn vượt quá chi phí phát triển và được sản sinh ra bởi một đội chứ không phải là một người lập trình riêng rẽ. Nó trợ giúp cho tất cả mọi hoạt động phát triển và bảo trì.

Thực tế biên giới giữa các loại đó không rõ ràng.

b.Các môi trường lập trình

Các môi trường lập trình có thể được nhóm lại thành các lớp sau:

i) Các môi trường mục đính khái quát

ii) Các môi truờng hướng ngôn ngữ.

Môi trường lập trình có thể có các công cụ sau:

i) Phần mềm giao tiếp máy chủ - máy khách.

ii) Phần mềm bắt chước máy khách.

iii) Các bộ biên dịch chéo.

iv) Các công cụ thử nghiệm và gỡ lỗi.

v) Các công cụ quản lý cấu hình.

vi) Các công cụ giao tiếp.

c.Bàn thợ CASE

Các thành phần điển hình của bàn thợ CASE là:

i) Tỷ lệ soạn thảo biểu đồ.

ii) Các tiện ích phân tích thiết kế và kiểm tra.

iii) Các tiện ích ngôn ngữ hỏi.

iv) Các tiên ích từ điển dữ liệu.

v) Các tiện ích sinh ra báo cáo.

vi) Các công cụ tạo dạng cho phép việc định dạng màn hình và tư liệu là được đặc tả.

vii) Các tiện ích xuất nhập khẩu.

viii) Trợ giúp các bộ sinh mã cốt tự động từ thiết kế có trong kho trung tâm.

Các hệ bàn thợ CASE thường chủ yếu dùng dể phát triển các hệ thống lý dữ liệu. Có người đã minh định một số các khiếm của bàn thợ CASE như sau:

i) Nó không thích hợp được với các công cụ chuẩn bị tư liệu khác. Tiện ích xuất nhập thường là gắn với văn bản ASCII

ii) Thiếu chuẩn hoá, trao đổi thông tin giữa các bàn thợ khác là khó khăn hoặc không thể

iii)Thiếu tiện ích cho phép một kết nối vào một ứng dụng hoặc lớp ứng dụng.

iv) Tiện ích để tạo ra tài liệu chất lượng cao thiếu tổng quát

v) Tiện ích lập biểu đồ là chậm chạp

3 Phong cách lập trình

Phong cách lập trình bao hàm một triết lí về lập trình, nhấn mạnh tới tính đơn giản và rõ ràng. Các yếu tố của phong cách lập trình bao gồm tài liệu (mức chương trình gốc) bên trong, phương pháp khai báo dữ liệu, xây dựng câu lệnh, vào/ra.

3.1 Tài liệu chương trình

Tài liệu bên trong của chương trình gốc bắt đầu với việc chọn lựa các các tên gọi định danh (biến và nhãn), vị trí và thành phần của việc chú thích, và cách tổ chức trực quan của chương trình.

Việc lựa chọn các tên gọi định danh/tên chính là điều kiện chủ chốt cho việc hiểu chương trình. Những ngôn ngữ giới hạn tên biến hay nhãn chỉ trong vài kí tự tự nó đã mang ý nghĩa mơ hồ. Cho dù một chương trình nhỏ thì một tên gọi có nghĩa cũng làm tăng tính dễ hiểu. Theo ngôn từ của mô hình cú pháp/ngữ nghiã, tên có ý nghĩa làm "đơn giản việc chuyển đổi từ cú pháp chương trình sang cấu trúc ngữ nghĩa bên trong".

Rõ ràng là: phần mềm phải chứa tài liệu bên trong. Lời chú thích cung cấp cho người phát triển một phương tiện truyền thông với các độc giả khác về chương trình gốc. Lời chú thích có thể cung cấp một hướng dẫn rõ ràng dễ hiểu trong pha cuối cùng của kỹ nghệ phần mềm là bảo trì.

Có nhiều hướng dẫn đã được đề nghị cho việc viết lời chú thích. Các chú thích mở đầu và chú thích chức năng là hai thành phần đòi hỏi cách tiếp cận khác nhau:

Lời chú thích mở đầu nên xuất hiện ở ngay đầu của mọi modul. Định dạng cho lời chú thích như sau:

1. Một phát biểu về mục đích chỉ rõ chức năng modul

2. Mô tả giao diện bao gồm:

a) Một mẫu "dãy lời gọi"

b) Mô tả về mọi đối tượng

c) Danh sách tất cả các modul thuộc cấp.

3. Thảo luận về dữ liệu tương ứng (như các biến quan trọng và những hạn chế giới hạn về cách dùng chúng) và các thông tin quan trọng khác

4. Lịch sử phát triển bao gồm:

a) Tên người thiết kế modul (tác giả)

b) Tên người xét duyệt (kiểm toán) và ngày tháng.

c) Ngày tháng sửa đổi và mô tả sửa đổi.

Các chú thích chức năng được nhúng vào bên trong thân của chương trình gốc và được dùng để mô tả cho các hàm xử lý. Bên cạnh đó lời chú thích mô tả nên:

- Mô tả các khối chương trình thay vì chú thích cho từng dòng.
- Dùng dòng trống hay tụt lề để cho lời chú thích có thể được thuận tiện với chương trình.
- Phải đúng đắn: một lời chú thích không đúng hay gây ra hiểu sai thì còn tồi tệ hơn là không có chú thích nào cả.

Với những tên gọi tượng trưng đúng đắn và việc chú thích tốt, việc làm tài liệu bên trong thích hợp sẽ được bảo đảm.

Khi một thiết kế thủ tục chi tiết được biểu diễn bằng cách dùng một ngôn ngữ thiết kế chương trình thì tài liệu thiết kế có thể được nhúng trực tiếp vào trong văn bản chương trình gốc như những câu chú thích. Kỹ thuật này đặc biệt có ích khi việc làm tài liệu được thực hiện trong hợp ngữ và giúp đảm bảo rằng cả chương trình thiết kế sẽ được bảo trì khi những thay đổi được thực hiện cho cả hai.

Tổ chức trực quan của chương trình gốc như trong bản in là một đóng góp quan trọng cho tính dễ đọc. Việc **tụt lề chương trình gốc** chỉ ra kết cấu và khối logic của chương trình sao cho những thuộc tính này là thấy được so với lề trái. Giống như việc chú thích, cách tiếp cận tốt nhất là nên để mở cho tranh luận. Việc tụt lề thủ công có thể trở nên phức tạp khi có sự sửa đổi chương trình và kinh nghiệm chỉ ra rằng khi đã tích luỹ đủ hiểu biết thì sẽ tăng cường được việc để lề cho khớp. Có lẽ cách tiếp cận tốt nhất là dùng bộ định dạng chương trình tự động (như công cụ CASE: Visual Basic, Visual Studio, Edit Plus …) sẽ đặt đúng việc tụt lề cho chương tình gốc. Bằng cách xoá bỏ đi gánh nặng của việc làm tụt lề cho người lập trình, có thể cải thiện khuôn dạng chương trình với tương đối ít công sức.

3.2 Khai báo dữ liệu

Độ phức tạp và việc tổ chức cấu trúc dữ liệu được xác định trong bước thiết kế. Phong cách khai báo dữ liệu được thiết lập khi chương trình được sinh ra. Một số hướng dẫn tương đối đơn giản có thể được lập ra để làm cho dữ liệu được dễ hiểu hơn và đơn giản hơn khi bảo trì.

Thứ tự khai báo dữ liệu nên được chuẩn hoá cho dù ngôn ngữ lập trình không có yêu cầu bắt buộc nào về diều đó. Chẳng hạn, thứ tự khai báo cho một modul FOTRAN có thể là:

1. Mọi khai báo tường minh (để có chất lượng cao, mọi biến đều nên khai báo):

INTEGER, REAL, DOUBLE, PRECISION,...

2. Mọi khối dữ liệu toàn cục: COMMON/tên khối/...

3. Mọi mảng cục bộ: DIMENSION tên mảng và chiều

4. Mọi khai báo tệp: DEFINE, OPEN, CLOSE

Thứ tự tạo ra các thuộc tính để dễ tìm, cho phép xúc tiến kiểm thử, gỡ lỗi và bảo trì.

Khi có nhiều tên biến được khai báo trong một câu lệnh thì việc sắp xếp theo trật tự chữ cái cho các tên gọi có cùng có giá trị. Tương tự, dữ liệu toàn cục có nhãn (như khối chung trong FOTRAN) cũng nên được lập thứ tự theo bảng chữ.

Nếu thiết kế có mô tả trước cấu trúc dữ liệu phức tạp thì nên chú thích những điểm đặc thù cố hữu trong việc cài đặt ngôn ngữ lập trình. Chẳng hạn, cấu trúc dữ liệu danh sách móc nối trong C hay kiểu dữ liệu người dùng xác định trong PASCAL có thể yêu cầu tài liệu bổ sung có chứa trong lời chú thích của nó.

3.3 Xây dựng câu lệnh

Việc xây dựng luồng logic phần mềm được thiết lập trong khi thiết kế việc xây dựng từng câu lệnh tuy nhiên lại là một phần của bước lập trình. Việc xây dựng câu lệnh nên tuân theo một qui tắc quan trọng: mỗi câu lệnh nên đơn giản và trực tiếp chương trình không nên bị xoắn tít để đạt tính hiệu quả.

Nhiều ngôn ngữ lập trình cho phép nhiều câu lệnh trên một dòng. Khía cạnh tiết kiệm không gian của tính năng này khó mà biện minh bởi tính khó đọc nảy sinh.

Cấu trúc chương trình và các phép toán điều kiện được chứa trong đoạn trên đều bị che lấp bởi cách xây dựng nhiều câu lệnh trên một dòng.

Cách xây dựng câu lệnh đơn và việc tụt lề minh hoạ cho các đặc trưng logic và chức năng của đoạn này. Các câu lệnh chương trình gốc riêng lẻ có thể được đơn giản hoá:

- Việc tránh dùng các phép kiểm tra dữ liệu phức tạp
- Khử bỏ các phép kiểm tra điều kiện phủ định
- Tránh lồng nhau nhiều giữa các điều kiện hay chu trình
- Dùng dấu ngoặc để làm sáng tỏ các biểu thức số học hay logic
- Dùng dấu cách và/hoặc các ký hiệu dễ đọc để làm sáng tỏ nội dung câu lệnh.
- Suy nghĩ: Liệu ta có thể hiểu được điều này nếu ta không là người lập trình cho nó không?
- Từng hướng dẫn trên đều cố gắng *"giữ cho đơn giản"*

3.4 Vào/ra

Phong cách vào và ra được thiết lập trong khi phân tích thiết kế yêu cầu phần mềm, không phải khi lập trình. Tuy cách thức vào và ra được cài đặt có thể là đặc trưng xác định việc cộng đồng người sử dụng chấp nhận hệ thống. Phong cách vào ra sẽ thay đổi theo mức độ tương tác con người. Với vào ra theo lô thì cách tổ chức cái vào logic, kiểm tra lỗi vào/ra có nghĩa, phục hồi lỗi vào/ra tốt và định dạng báo cáo ra hợp lý là đặc trưng mong muốn. Với cái vào/ra tương ứng, một sơ đồ đưa vào có hướng dẫn, đơn giản, viêc kiểm tra lỗi kỹ lưỡng và phục hồi chúng, cái ra và sự nhất quán của định dạng vào ra trở thành mối quan tâm chủ yếu.

Bất kể tới bản chất theo lô hay tương tác của phần mềm, một số hướng dẫn phong cách vào/ra nên được xét tới trong khi thiết kế và lập trình:

- Làm hợp lệ mọi cái vào

- Kiểm tra sự tin cậy của các tổ hợp khoản mục vào quan trọng.
- Giữ cho định dạng cái vào đơn giản.
- Dùng các chỉ báo cuối dữ liệu thay vì yêu cầu người dùng xác định số các khoản mục
- Đặt nhãn cho yêu cầu cái vào tương tác, xác định chọn lựa có sẵn hay gắn các giá trị
- Giữ cho định dạng cái vào thống nhất khi một ngôn ngữ lập trình có các yêu cầu định dạng nghiêm ngặt

Phong cách của vào/ra bị ảnh hưởng bởi nhiều đặc trưng khác như thiết bị vào/ra (như kiểu thiết bị cuối hay trạm làm việc, thiết bị đồ hoạ máy tính, chuột v.v...), độ phức tạp của người dùng là môi trường truyền thống.

4 Tính hiệu quả

Trong hệ thống kỹ nghệ tốt, có một khuynh hướng tự nhiên là dùng các tài nguyên chủ chốt một cách hiệu quả. Các chu trìnnh bộ xử lí và vị trí bộ nhớ thường được coi như các tài nguyên chủ chốt. Thứ nhất, tính hiệu quả là một yêu cầu hoàn thiện và do đó nên được thiết lập trong phân tích yêu cầu phần mềm. Thứ hai là tính hiệu quả được cải thiện với thiết kế tốt. Thứ ba là tính hiệu quả của chương trình và tính đơn giản của chương trình đi đôi với nhau. Nói chung, không nên gạt bỏ tính rõ ràng, dễ đọc hay tính đúng đắn chỉ để có được sự cải thiện nho nhỏ về tính hiệu quả.

4.1 Kỹ thuật lập trình hướng hiệu qủa

Lập trình là một nghề thủ công. Nó phụ thuộc vào kỹ xảo cá nhân người lập trình, sự chú ý đến các chi tiết và kiến thức về việc sử dụng các công cụ sẵn có theo cách thức tốt nhất. Trong phần này chỉ tập trung vào một vài kỹ thuật chuyên biệt được dùng nhằm đạt được một hệ thống tin cậy, khả chuyển và dùng lại được các thành phần.

Nhu cầu các hệ thống đáng tin đang tăng lên, hiển nhiên là vì các hệ thống máy tính đã lan khắp nơi. Hiện thời có hai kỹ thuật để viết các chương trình đáng tin: tránh lỗi và thứ lỗi

A. Tránh lỗi

Tất cả các kỹ sư phần mềm hẳn đều muốn làm ra các phần mềm không có lỗi. Một quá trình phát triển chỉ dựa vào việc phát hiện lỗi và khử lỗi chứ không để ý đến tránh lỗi là một quá trình chưa thật tốt.

Phần mềm không có lỗi nói ở đây là phần mềm tuân theo đúng đặc tả. Nói chung, có thể có lỗi trong đặc tả hoặc có thể không phản ánh đúng các nhu cầu của người sử dụng. Vậy là phần mềm không có lỗi không nhất thiết là các phần mềm luôn luôn hành xử như người dùng dự đoán.

Việc phát triển phần mềm không có lỗi đòi hỏi chi phí nhiều. Khi mà một số lỗi đã được tháo khỏi chương trình thì giá cả cho việc tìm và tháo các lỗi còn lại có xu hướng tăng theo hàm số mũ. Do dó một tổ chức có thể quyết định chấp nhận một vài lỗi còn lưu lại. Tính về mặt giá cả thì thà rằng chịu tiền chi trả cho các phí tổn của hệ thống do các lỗi đó gây ra còn hơn là đi điều tra và tháo gỡ các lỗi đó trước khi phân phối.

Tránh lỗi và phát triển phần mềm vô lỗi dựa trên:

i) Sản phẩm của một đặc tả hệ thống chính xác.

ii) Chấp nhận một cách tiếp cận thiết kế phần mềm lựa chọn việc che dấu thông tin và bao gói thông tin.

iii) Tăng cường duyệt lại trong quá trình phát triển và thẩm định hệ thống phần mềm.

iv) Chấp nhận triết lý chất lượng tổ chức: chất lượng là bánh lái của quy trình xây dựng phần mềm.

v) Việc lập kế hoạch cẩn thận cho việc thử nghiệm hệ thống để trưng ra các lỗi mà các lỗi này chưa được phát hiện trong quá trình duyệt lại và để định lượng độ tin cậy của hệ thống.

Có hai cách chính để hỗ trợ tránh lỗi:

Lập trình có cấu trúc

Thuật ngữ này được đặt ra từ cuối những năm 60 và có nghĩa là lập trình mà không dùng *goto*, lập trình chỉ dùng các vòng lặp *while* và các phát biểu *if* để xây dựng điều khiển và trong thiết kế thì dùng cách liếp cận từ trên xuống (top down). Việc thừa nhận lập trình có cấu trúc là quan trọng bởi vì nó là bước đầu tiên bước từ cách tiếp cận không khuôn phép tới phát triển phần mềm.

Lập trình có cấu trúc buộc người lập trình phải nghĩ cẩn thận về chương trình của họ, và vì vậy nó ít tạo ra sai lầm trong khi phát triển

Lập trình có cấu trúc làm cho chương trình có thể được đọc một cách tuần tự và do đó dễ hiểu và dễ thanh tra. Tuy nhiên nó chỉ là bước đầu tiên trong viêc lập trình nhằm đạt độ tin cậy tốt. Có một số cấu trúc có ích nhưng hay dẫn tới các lỗi trong hệ thống như: các số thực dấu phảy động, con trỏ, song song, đệ quy, các ngắt, vì vậy người lập trình nên dùng chúng một cách cẩn thận.

Phân quyền truy cập dữ liệu

Nguyên lý an ninh được thừa nhận bởi các tổ chức vũ trang là một nguyên lý nhu cầu để biết. Mỗi thành phần chương trình chỉ được phép truy cập đến dữ liệu nào cần thiết để thực hiện chức năng của nó.

Ưu điểm của việc che dấu thông tin là các thông tin bị che dấu không thể bị sụp đổ bởi các thành phần chương trình mà được xem rằng không dùng thông tin đó. Biểu diễn dữ liệu có thể được thay đổi mà không phải thay đổi các thành phần khác có sử dụng thông tin đó.

B. Thứ lỗi

Ngay với một hệ vô lỗi thì vẫn cần một tiện ích thứ lỗi: đó là vì có thể có các lỗi đặc tả. Một tiện ích thứ lỗi là cần thiết cho một hệ thống đáng tin.

Có bốn hoạt động cần phải tiến hành nếu hệ thống là thứ lỗi

i) Phát hiện lỗi.

ii) Định ra mức độ thiệt hại.

iii) Hồi phục sau khi gặp lỗi: hệ thống phải hồi phục về trạng thái mà nó biết là an toàn. Cũng có thể là chỉnh lý trạng thái bị huỷ hoại (hồi phục tiến), cũng có thể là lui về một trạng thái trước đó là an toàn (hồi phục lùi).

iv) Chữa lỗi: Cải tiến hệ thống để cho lỗi đó không xuất hiện nữa. Trong nhiều trường hợp sự thất bại của phần mềm là tàng hình và gây ra bởi một tổ hợp của thông tin vào.

C. Xử lý bất thường

Một sai lầm nào đó hoặc một sự cố bất ngờ xuất hiện được gọi là một bất thường. Các bất thường có thể do phần cứng cũng có thể do phần mềm. Khi mà một bất thường không dự đoán được thì bộ điều khiển sẽ chuyển cho cơ chế xử lý hệ thống bất thường. Nếu một bất thường đã được dự đoán thì mã phải bao gồm cả việc phát hiện và xử lý bất thường đó.

Hầu hết các ngôn ngữ lập trình là không có các tiện ích để phát hiện và xử lý bất thường. Các bất thường có thể được ghi lại bằng cách dùng một biến Boolean nhằm chỉ ra rằng có một bất thường đã xuất hiện.

D. Lập trình phòng thủ

Lập trình phòng thủ là cách phát triển chương trình mà người lập trình giả định rằng các mâu thuẫn hoặc các lỗi chưa được phát hiện có thể tồn tại trong chương trình. Phải có phần mềm kiểm tra trạng thái hệ thống sau khi biến đổi và phải đảm bảo rằng sự biến đổi trạng thái là kiên định. Nếu phát hiện một mâu thuẫn thì việc biến đổi trạng thái là phải rút lại và trạng thái phải trở về trạng thái đúng đắn trước đó.

Lập trình phòng thủ là một cách thử lỗi được tiến hành không cần bộ điều khiển thứ lỗi. Về cơ bản quá trình vẫn là: phát hiện lỗi, đánh giá lỗi và tránh lỗi.

Nói chung một lỗi gây ra một sự sụp đổ trạng thái được gán các trị không hợp luật. Ngôn ngữ lập trình như Ada cho phép phát hiện ra các lỗi đó ngay trong thời gian biên dịch. Tuy nhiên việc kiểm tra biên dịch chỉ hạn chế cho các giá trị tĩnh. Một cách để phát hiện lỗi trong chương trình Ada là dùng cơ chế xử lý bất thường kết hợp với đặc tả miền giá trị.

Hồi phục lỗi là một quá trình cải biến không gian trạng thái của hệ thống sao cho hiệu ứng của lỗi là nhỏ nhất và hệ thống có thể tiếp tục vận hành, có lẽ là trong một mức suy giảm. **Hồi phục tiến** liên quan đến việc cố gắng chỉnh lại trạng thái hệ thống. **Hồi phục lùi** liên quan đến việc lưu trạng thái của hệ thống ở một trạng đúng đã biết.

Hồi phục tiến thường là một ứng dụng chuyên biệt. Có hai tình thế khi đó hồi phục tiến có thể thành công:

1. Khi dữ liệu đã bị sụp. Việc xử dụng kỹ thuật mã hoá bằng cách thêm các dữ liệu dư thừa vào dữ liệu cho phép sửa sai khi phát hiện lỗi.

2. Khi cấu trúc nối bị suy sụp. Nếu các con trỏ tiến và lùi đã có trong cấu trúc dữ liệu thì cấu trúc đó có thể được tái tạo nếu như còn đủ các con trỏ chưa bị sụp. Kỹ thuật này thường được dùng cho việc sửa chữa hệ thống tệp và cơ sở dữ liệu.

Hồi phục lùi là một kỹ thuật đơn giản liên quan đến việc duy trì các chi tiết của trạng thái an toàn và cất giữ trạng thái đó khi mà sai lầm đã bị phát hiện. Hầu hết các hệ quản trị cơ sở dữ liệu đều có bộ phục hồi lỗi. CSDL chỉ cập nhật dữ liệu một khi giao dịch đã hoàn tất và không phát hiện đựoc vấn đề gì. Nếu giao dịch thất bại thì CSDL không được cập nhật.

Một kỹ thuật khác là thiết lập các điểm kiểm tra thường kỳ mà chúng là các bản sao của trạng thái hệ thống. Khi mà một lỗi được phát hiện thì trạng thái an toàn đó được tái lưu kho từ điểm kiểm tra gần nhất.

Trường hợp hệ thống dính líu tới nhiều quá trình hợp tác thì dãy các giao tiếp có thể là các điểm kiểm tra của các quá trình đó không đồng bộ và để hồi phục thì mỗi quá trình phải trở lại trạng thái ban đầu của nó.

E. Sử dụng lại

Một đặc trưng của công trình học là sử dụng cách tiếp cận thiết kế hệ thống mà nó sử dụng tối đa các thành phần đã tồn tại. Người kỹ sư thiết kế không đặc tả một thiết kế mà trong đó mỗi thành phần đã được chế tạo dựa vào thiết kế các thành phần đã được nghiên cứu hoặc thử nghiệm trong các hệ thống khác.

Chưa có một cơ sở chung nào về việc dùng lại các thành phần phần mềm với tư liệu được phổ biến rộng rãi được dùng để thiết kế phần mềm. Tuy vậy, chúng ta cần quan tâm đến những vấn đề sau đây:

Phân loại thành phần dùng lại được

i) Các hệ ứng dụng

ii) Các hệ con. Thí dụ hệ đo mẫu được phát triển như là một phần của hệ xử lý văn bản có thể được dùng lại trong một hệ quản trị cơ sở dữ liệu,

iii) Các modul hoặc các đối tượng,

iv) Các hàm.

Phát triển phần mềm để dùng lại được

Việc sử dụng lại một cách hệ thống đòi hỏi một cơ sở dữ liệu được xếp theo danh mục của các thành phần dùng lại được. Quan niệm sai lầm là các thành phần này đã có sẵn có trong các hệ thống đang tồn tại và một thư viện các thành phần có thể được tạo ra bằng cách trích chúng ra và viết tư liệu cho chúng.

Sự thật các thành phần được tạo ra như là một phần của một ứng dụng là không chắc sẽ dùng lại được. Các thành phần này hướng về phục vụ các yêu cầu của hệ thống mà thành phần ấy thuộc về. Để dùng lại được một cách có hiệu quả, nó phải thực sự được sinh ra như là để thoả mãn một phổ rộng các yêu cầu.

Việc phát triển các thành phần khái quát là đắt hơn việc phát triển các thành phần cho mọi mục đích chuyên biệt và do đó nó làm tăng chi phí dự án.

Để phát triển các thành phần dùng lại được cần có một quyết định chính sách có tổ chức là chi phí ngắn hạn giúp cho các mục tiêu lâu dài. Chính tổ chức chứ không phải cá nhân các nhà quản lý phải ra quyết định như vậy. Các nhà quản lý cấp trên thường miễn cưỡng ủng hộ chi phí để dùng lại vì các khó khăn trong định lượng các ưu điểm trong tương lai của một thư việc các thành phần dùng lại được.

Để đánh giá độ dùng lại được cần phải đặt ra hai câu hỏi sau đây:

i) Thành phần này biểu diễn một sự khái quát lĩnh vực ứng dụng tốt như như thế nào.

ii) Thành phần đã được viết ra có là khái quát và thích nghi được hay không.

Phát triển phần mềm có thành phần tái sử dụng là giảm được chi phí và ngoài ra còn có 5 ưu điểm nữa:

i) Độ tin cậy của hệ thống được tăng lên.

ii) Các rủi ro là giảm đi.

iii) Sử dụng hiệu quả các chuyên gia ứng dụng.

iv) Các chuẩn tổ chức có thể được bao gồm trong các thành phần dùng lại được.

v)Thời gian phát triển phần mềm có thể được rút gọn.

4.2 Một vài hướng dẫn lập trình hướng hiệu quả

1. Tính hiệu qủa của chương trình

Tính hiệu quả của chương trình gốc có liên hệ trực tiếp với tính hiệu quả của thuật toán được xác định trong thiết kế chi tiết. Tuy nhiên, phong cách lập trình có thể có một tác động đến tốc độ thực hiện và yêu cầu bộ nhớ. Tập hợp các hướng dẫn sau đây bao giờ cũng có thể áp dụng được khi thiết kế chi tiết được dịch thành chương trình:

Đơn giản hoá các biểu thức số học và logic trước khi đi vào lập trình.

Tính toán cẩn thận từng chu kì lồng nhau để xác định liệu các câu lệnh hay biểu thức có thể được chuyển ra ngoài hay không.

- Khi có thể, hãy tránh dùng mảng nhiều chiều .
- Khi có thể hãy tránh việc dùng con trỏ và danh sách phức tạp.
- Dùng các phép toán số học nhanh.
- Không trộn lẫn các kiểu dữ liệu cho dù ngôn ngữ có cho phép điều đó.

- Dùng các biểu thức số học và logic bất kì khi nào có thể được.

Nhiều trình biên dịch có tính năng tối ưu tự động sinh ra chương trình hiệu quả bằng cách dồn nén các biểu thức lặp, thực hiện tính chu trình, dùng số học nhanh và áp dụng các thuật toán liên quan khác. Với những ứng dụng trong đó tính hiệu quả có ý nghĩa quan trọng, những trình biên dịch như thế là công cụ lập trình không thể thiếu được.

2. Hiệu quả bộ nhớ

Tính hiệu quả bộ nhớ phải được tính vào đặc trưng "phân trang" (segment) của hệ điều hành. Nói chung, tính cục bộ của chương trình hay việc bảo trì lĩnh vực chức năng qua các kết cấu có cấu trúc là một phương pháp tuyệt vời làm giảm việc phân trang và do đó làm tăng tính hiệu quả.

Hạn chế bộ nhớ trong thế giới bộ vi xử lí nhúng là mối quan tâm rất thực tế, mặc dầu bộ nhớ giá thấp, mật độ cao vẫn đang tiến hoá nhanh chóng. Nếu yêu cầu hệ thống cần tới bộ nhớ tối thiểu (như sản phẩm giá thấp, khối lượng lớn) thì trình biên dịch ngôn ngữ cấp cao phải được trù tính cẩn thận với tính năng nén bộ nhớ hay như một phương kế cuối cùng, có thể phải dùng tới hợp ngữ.

Không giống như nhiều đặc trưng hệ thống khác phải trả giá lẫn nhau, các kỹ thuật cho hiệu quả về thời gian thực hiện đôi khi có thể dẫn tới hiệu quả bộ nhớ. Chẳng hạn, giới hạn việc dùng các mảng ba hay bốn chiều làm nảy sinh thuật toán thâm nhập phần tử đơn, thuật toán nhanh và ngắn nhất. Lần nữa, chìa khoá cho tính hiệu quả bộ nhớ là **"giữ cho đơn giản"**.

3. Hiệu quả vào/ra

Cái vào do người dùng cung cấp và cái ra được tạo ra cho người dùng là hiệu quả khi thông tin có thể được cung cấp hay được hiểu với một mức độ tiết kiệm nỗ lực trí tuệ.

- Một số hướng dẫn đơn giản để tăng cường hiệu quả vào/ra
- Số các yêu cầu vào / ra nên giữ mức tối thiểu
- Một sự việc vào/ra nên qua bộ đệm để làm giảm phí tổn liên lạc
- Với bộ nhớ phụ (như đĩa) nên lựa chọn và dùng phương pháp thâm nhập đơn giản nhất chấp nhận được.
- Nên xếp khối vào/ra với các thiết bị bộ nhớ phụ.
- Việc vào / ra với thiết bị cuối hay máy in nên nhận diện các tính năng của thiết bị có thể cải tiến chất lượng hay tốc độ
- Hãy nhớ rằng "siêu hiệu quả" của vào/ra là vô nghĩa nếu nó không được hiểu rõ.

Thiết kế vào ra lập nên phong cách và cuối cùng chi phối tính hiệu quả. Những hướng dẫn trình bày trên đây là áp dụng được cho cả các bước thiết kế và lập trình cho tiến trình kỹ nghệ phần mềm.

5 Thẩm định và xác minh

5.1 Đại cương về việc thẩm định và xác minh

Xác minh và thẩm định một hệ phần mềm là quá trình liên tục xuyên suốt mọi giai đoạn của quá trình phần mềm. Xác minh và thẩm định mang tính quá trình nhằm đảm bảo phần mềm thoả mãn các yêu cầu của khách hàng.

Xác minh và thẩm định là một quá trình kéo dài suốt vòng đời. Nó bắt đầu khi duyệt xét yêu cầu.

Xác minh và thẩm định có hai mục tiêu:

i)Xác định các khuyết tật trong hệ thống

ii) Đánh giá xem hệ thống có dùng được hay không?

Sự khác nhau giữa xác minh và thẩm định là

i) Thẩm định: xét xem cái được xây dựng có là sản phẩm đúng không ?

ii) Xác minh: xét xem cái được xây dựng có đúng là sản phẩm không ?

Như vậy xác minh là kiểm tra xem chương trình có phù hợp với đặc tả hay không. Còn thẩm định là kiểm tra xem chương trình có được như mong đợi của người dùng hay không.

Có hai loại phép thử

i) Phép thử thống kê: để phản ánh tần suất các input của người dùng thực và sau khi vận hành máy, có thể cho ra một đánh giá độ tin cậy thao tác của hệ thống.

ii) Phép thử khuyết tật: để bộc lộ các khuyết tật trong hệ thống.

5.2 Sơ lược về tiến trình kiểm thử phần mềm

Quá trình kiểm thử

Trừ các hệ nhỏ, nói chung không nên thử hệ thống nguyên cả khối. Quá trình thử có thể chia làm 5 giai đoạn:

1) Thử đơn vị

2) Thử modul (chức năng)

3) Thử hệ con

4) Thử hệ thống

5) Thử sau lưng (alpha) và thử điều tra (beta)

Kế hoạch kiểm thử

Thử hệ thống là rất đắt, đối với một vài hệ thống thời gian thực có các ràng buộc thời gian phức tạp thì việc thử có thể ngốn hết khoảng nửa tổng chi phí phát triển. Vì thế mà phải lập kế hoạch thử và khống chế chi phí thử

Việc thử liên quan đến nhiều việc thiết lập các mẫu cho quá trình kiểm thử nhiều hơn là mô tả các phép thử.

Chiến lược kiểm thử

Các chiến lược như đã đề cập:

1. **Thử từ trên xuống**: nên dùng các phát triển từ trên xuống
2. **Thử từ dưới lên**: việc thử từ dưới lên luôn luôn là cần thiết cho các thành phần hệ thống mức thấp.
3. **Thử luồn sợi**: dùng cho các hệ thống thời gian thực
4. **Thử gay cấn (thử áp lực)**: cho những hệ thống có giới hạn tải, và phép thử tăng dần tải cho tới khi hệ thống sập đổ để xác định độ chịu tải thực sự.

Tóm tắt

Bước lập trình của kỹ nghệ phần mềm là một tiến trình dịch chuyển hoá. Thiết kế chi tiết được dịch sang một ngôn ngữ lập trình mà cuối cùng được biến đổi thành các lệnh mã máy thực hiện được. Các đặc trưng tâm lý và kỹ thuật của ngôn ngữ lập trình có ảnh hưởng lớn tới quá trình dịch và kiểm thử cũng như bảo trì phần mềm. Các đặc trưng này có thể được áp dụng vào ngôn ngữ lập trình thuộc một trong bốn thế hệ ngôn ngữ.

Phong cách là một đặc tính quan trọng của chương trình gốc và có thể xác định ra tính dễ đọc của chương trình. Các yếu tố của phong cách bao gồm việc làm tài liệu bên trong, phương pháp khai báo dữ liệu, thủ tục xây dựng câu lệnh, và các kỹ thuật lập trình vào ra. Trong mọi

trường hợp, tính đơn giản và rõ ràng là các đặc trưng chính. Một nhân tố của phong cách lập trình là thời gian thực hiện và tính hiệu quả của bộ nhớ cần đạt tới. Mặc dầu tính hiệu quả có thể là yêu cầu cực kỳ quan trọng, chúng ta nên nhớ rằng một chương trình hiệu quả mà lại không dễ đọc thì cùng mang một giá trị đáng hoài nghi.

Lập trình là cốt lõi của tiến trình phần mềm. Các bước quan trọng chủ chốt phải hoàn tất trước lập trình như phân tích, xác định yêu cầu và đặc tả của thiết kế chi tiết.

Củng cố

1. Tóm lược các đặc trưng ngôn ngữ lập trình?

2. Nền tảng của ngôn ngữ lập trình? (định kiểu dữ liệu, cơ chế chương trình con, cấu trúc điều khiển và cách tiếp cận hướng đối tượng)

3. Sơ lược về các yếu tố trong phong cách lập trình?

4. Sơ lược về kỹ thuật lập trình hướng hiệu quả?

5. Nêu ra một vài hướng dẫn lập trình hướng hiệu quả?

6. Mục tiêu, ý nghĩa vai trò và nội dung của việc thẩm định và xác minh?

7. Sơ lược về tiến trình thử nghiệm? (quá trình, kế hoạch, chiến lược)

TÀI LIỆU THAM KHẢO

1. *Kỹ nghệ phần mềm – Cách tiếp cận của người thực hành*, Roger S.Pressman, Tập 1, Người dịch: Ngô Trung Việt, NXB Giáo dục, Năm 1997
2. *Kỹ nghệ phần mềm – Cách tiếp cận của người thực hành*, Roger S.Pressman, Tập 2, Người dịch: Ngô Trung Việt, NXB Giáo dục, Năm 1997
3. *Kỹ nghệ phần mềm – Cách tiếp cận của người thực hành*, Roger S.Pressman, Tập 3, Người dịch: Ngô Trung Việt, NXB Giáo dục, Năm 1997
4. *Giáo trình Kỹ nghệ phần mềm*, Nguyễn Văn Vỵ, Đại học Quốc gia Hà Nội, Năm 2001
5. *Giáo trình Công nghệ phần mềm*, Đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội, Năm 2000.
6. *Software Engineering – A Practitioner's Approach*, Roger S.Pressman, Third Edition, McGraw-Hill.Inc, 1992.
7. *The Unified Software Development Process,* Ivar Jacobson, Grandy Booch, James Rumbaugh.

Giáo trình

Công nghệ phần mềm

MỤC LỤC

LỜI NÓI ĐẦU

Nhập môn Công Nghệ Phần Mềm là môn học nhằm giúp cho sinh viên có kiến thức cơ bản nhất trong lĩnh vực công nghệ phần mềm. Qua môn học này sinh viên có cái nhìn khái quát về qui trình phát triển phần mềm, hiểu biết và thực hiện các giai đoạn trong qui trình trên một phần mềm cụ thể dựa trên những phương pháp, kỹ thuật trong quá trình thu thập yêu cầu, phân tích, thiết kế và cài đặt, viết sưu liệu đã được minh họa cụ thể trong giáo trình. Mục tiêu giáo trình là sinh viên có thể hiểu được những yêu cầu công việc cần phải làm ở mỗi giai đoạn của qui trình, để có thể đảm trách công việc ở một trong các giai đoạn làm phần mềm trong những nhóm dự án.

TÀI IỆU THAM KHẢO

1. Software Engineering
   By Nguyễn Xuân Huy – Institue of Information Technology

2. Nhập môn công nghệ phần mềm
   Nguyễn Tiến Huy – ĐH Khoa học Tự Nhiên

3. A Discipline for Software Engineering
   Watts S. Humphrey

4. Quá trình phát triển phần mềm thống nhất
   Nguyễn Tuấn Huy biên dịch –Nhà xuất bản thống kê

5. Analyzing Requriements and Defining Solution Architechtures
   Ian Lewis – Bruce Nielson

6. MCSD Analyzing Requirements Study Guide
   Tata McGraw-Hill Pusblishing Company Limited

7. Software Engineering
   Roger S.PressMan

8. Một số tài liệu tham khảo từ internet: Khoa CNTT ĐH KHTN, ĐH BKHN, ĐH Cần Thơ, và một số bài báo khoa học.
   - A Summary of Principles for User-Interface Design by Talin
   - The Foundation for Verifiable Software Process Improvement
   - Lecture Notes: Software Engineering I by Joey Paquet

Chương 1: TỔNG QUAN VỀ CÔNG NGHỆ PHẦN MỀM

1. CÁC KHÁI NIỆM CƠ BẢN

1.1. Phần mềm

1.1.1. Các khái niệm

Chương trình máy tính là một trình tự các chỉ thị để hướng dẫn máy tính làm việc nhằm hoàn thành một công việc nào đó do con người yêu cầu.

Phần mềm là một hệ thống các chương trình có thể thực hiện trên máy tính nhằm hỗ trợ các nhà chuyên môn trong từng lĩnh vực chuyên ngành thực hiện tốt nhất các thao tác nghiệp vụ của mình. Nhiệm vụ chính yếu của phần mềm là cho phép các nhà chuyên môn thực hiện các công việc của họ trên máy tính dễ dàng và nhanh chóng hơn so với khi thực hiện cùng công việc đó trong thế giới thực.

Hoạt động của mọi phần mềm là sự mô phỏng lại các hoạt động của thế giới thực trong một góc độ thu hẹp nào đó trên máy tính. Quá trình sử dụng một phần mềm chính là quá trình người dùng thực hiện các công việc trên máy tính để hoàn tất một công việc tương đương trong thế giới thực.

Lớp phần mềm là hệ thống các phần mềm trên cùng lĩnh vực hoạt động nào đó. Do cùng lĩnh vực hoạt động nên các phần mềm này thường có cấu trúc và chức năng (công việc mà người dùng thực hiện trên máy tính) tương tự nhau. Mục tiêu của ngành công nghệ phần mềm là hướng đến không những xây dựng được các phần mềm có chất lượng mà còn cho phép xây dựng dễ dàng một phần mềm mới từ các phần mềm đã có sẵn trong cùng kĩnh vực (thậm chí trong các lĩnh vực khác).

STT	Lớp phần mềm	Các phần mềm
1	Hỗ trợ giải bài tập	lượng giác, hình học, giải tích, số học, …
2	Trò chơi	cờ carô, cờ tướng, cờ vua, xếp hình, …
3	Xếp lịch biểu	thi đấu, thời khóa biểu, hội nghị, …
4	Xét tuyển	nhân sự, học sinh lớp 10…

5	Bình chọn	Sản phẩm, cầu thủ, …
6	Quản lý học sinh	Mầm non, trung học, trung tâm…
7	Bán hàng	thuốc tây, vật liệu xây dựng, máy tính
8	Quản lý thuê bao	điện, điện thoại, nước, …
9	Cho mượn	sách, truyện, phim, …

***Bảng 1.1**: Các phần mềm và lớp phần mềm tương ứng*

1.1.2. Phân loại

Phần mềm hệ thống là những phần mềm đảm nhận công việc tích hợp và điều khiển các thiết bị phần cứng đồng thời tạo ra môi trường thuận lợi để các phần mềm khác và người sử dụng có thể thao tác trên đó như một khối thống nhất mà không cần phải quan tâm đến những chi tiết kỹ thuật phức tạp bên dưới như cách thức trao đổi dữ liệu giữa bộ nhớ chính và đĩa, cách hiển thị văn bản lên màn hình, ...

Phần mềm ứng dụng là những phần mềm được dùng để thực hiện một công việc xác định nào đó. Phần mềm ứng dụng có thể chỉ gồm một chương trình đơn giản như chương trình xem ảnh, hoặc một nhóm các chương trình cùng tương tác với nhau để thực hiện một công việc nào đó như chương trình xử lý bản tính, chương trình xử lý văn bản, ...

1.1.3. Kiến trúc phần mềm

Sau khi đã có các khái niêm cơ bản nhất về phần mềm, tiếp sau đây chúng ta sẽ đi sâu vào tìm hiểu cấu trúc chi tiết các cấu trúc chi tiết các thành phần bên trong phần mềm. Phần mềm bao gồm 3 thành phần:

a) Thành phần giao tiếp (giao diện)

Cho phép tiếp nhận các yêu cầu về việc muốn thực hiện và cung cấp các dữ liệu nguồn liên quan đến công việc đó hoặc từ các thiết bị thu thập dữ liệu (cân, đo nhiệt độ, tế bào quang học, …)

Cho phép trình bày các kết quả của việc thực hiện các yêu cầu cho người dùng (kết quả của công việc khi thực hiện trên máy tính) hoặc điều khiển hoạt động các thiết bị điều khiển (đóng mở cửa, bật mở máy…)

Một cách tổng quát thành phần giao tiếp là hệ thống các hàm chuyên về việc nhập/xuất dữ liệu (hàm nhập/xuất) cùng với hình thức trình bày và tổ chức lưu trữ dữ liệu tương ứng, mục tiêu chính của các hàm này là đưa dữ liệu từ thế giới bên ngoài phần mềm vào bên trong hoặc ngược lại.

Trong phạm vi giáo trình này chỉ giới hạn xét đến giao tiếp với người sử dụng phần mềm và khi đó có tên gọi cụ thể hơn là thành phần giao diện.

b) Thành phần dữ liệu

Cho phép lưu trữ lại (hàm ghi) các kết quả đã xử lý (việc mượn sách đã được kiểm tra hợp lệ, bảng lương tháng đã được tính) trên bộ nhớ phụ với tổ chức lưu trữ được xác định trước (tập tin có cấu trúc, tập tin nhị phân, cơ sở dữ liệu).

Cho phép truy xuất lại (hàm đọc) các dữ liệu đã lưu trữ phục vụ cho các hàm xử lý tương ứng.

Một cách tổng quát thành phần dữ liệu là hệ thống các hàm chuyên về đọc ghi dữ liệu (hàm đọc/ghi) cùng với mô hình tổ chức dữ liệu tương ứng. Mục tiêu chính của các hàm này là chuyển đổi dữ liệu giữa bộ nhớ chính và bộ nhớ phụ.

c) Thành phần xử lý

Kiểm tra tính hợp lệ của các dữ liệu nguồn được cung cấp từ người dùng theo các quy trình ràng buộc trong thế giới thực (chỉ cho mượn tối đa 3 quyển sách, mỗi lớp học có tối đa 50 học sinh, …)

Tiến hành xử lý cho ra kết quả mong đợi theo quy định tính toán có sẵn trong thế giới thực (quy tắc tính tiền phạt khi trả sách trễ, quy tắc tính tiền điện, quy tắc trả góp khi mua nhà…) hoặc theo thuật giải tự đề xuất (xếp thời khóa biểu tự động, nén ảnh…)

Việc xử lý dựa trên dữ liệu nguồn từ người sử dụng cung cấp (tính nghiệm phương trình bậc 2 dựa trên các hệ số đã nhập) hoặc dữ liệu lưu trữ đã có sẵn (tính tồn kho tháng dựa trên các phiếu nhập xuất đã lưu trữ…) hoặc cả hai (tính tiền phạt dựa trên ngày trả sách được nhập vào và thông tin về loại sách đã được lưu trữ…) tùy vào xử lý cụ thể. Tương tự, việc xử lý cho ra kết quả có thể dùng để xuất cho người dùng xem qua thành phần giao diện (trình bày nghiệm, xuất tiền phạt), hay cùng có thể lưu trữ lại qua thành phần dữ liệu (sổ sách hiện đang được mượn của một độc giả…) hoặc cả hai (bảng lương, bảng tồn kho…)

Một cách tổng quát, thành phần xử lý là hệ thống các hàm chuyên về xử lý tính toán, biến đổi dữ liệu. Các hàm này sẽ dùng dữ liệu nguồn từ các hàm trong thành phần giao diện

(hàm nhập) hay thành phần dữ liệu (hàm đọc dữ liệu) kiểm tra tính hợp lệ (hàm kiểm tra) và sau đó tiến hành xử lý (hàm xử lý) nếu cần thiết để cho ra kết quả mà sẽ được trình bày cho người dùng xem qua các hàm trong thành phần giao diện (hàm xuất) hoặc lưu trữ lại qua các hàm trong thành phần dữ liệu (hàm ghi).

STT	Thành phần	Hàm	Ý nghĩa	Ghi chú
1	Thành phần giao diện	Hàm nhập Hàm xuất	Nhập yêu cầu, dữ liệu nguồn. Xuất kết quả đã xử lý	Cần xác định hình thức nhập/xuất và tổ chức dữ liệu tương ứng
2	Thành phần xử lý	Hàm kiểm tra Hàm xử lý	Kiểm tra tính hợp lệ của dữ liệu. Xử lý tính toán, phát sinh, biến đổi trên dữ liệu	Sử dụng hàm nhập, hàm đọc. Sử dụng hàm nhập, hàm đọc, hàm xuất, hàm ghi
3	Thành phần dữ liệu	Hàm đọc Hàm ghi	Đọc dữ liệu từ bộ nhớ phụ vào bộ nhớ chính. Ghi dữ liệu từ bộ nhớ chính vào bộ nhớ phụ	Cần xác định cáchh thức tổ chức lưu trữ dữ liệu

***Bảng 1.2**: Danh sách các hàm cùng ý nghĩa tương ứng*

1.2. Chất lượng phần mềm

1.2.1. Tính đúng đắn

Tính đúng đắn của phần mềm được thể hiện ở chỗ sản phẩm đó thực hiện đầy đủ và chính xác các yêu cầu của người dùng. Tính đúng đắn ở đây cần phải hiểu theo nghĩa rộng là

chương trình cần phải thực hiện được trong cả những trường hợp mà dữ liệu đầu vào là không hợp lệ.

Ví dụ, nếu một trong số các chức năng của phần mềm là sắp xếp một tập tin có số lượng mẫu tin tùy ý theo một cột tùy ý theo chiều tăng hoặc giảm thì những trường hợp sau là vi phạm tính đúng đắn của chương trình:

- Không thể thực hiện được (treo máy) khi tập tin rỗng (không có mẫu tin nào).
- Không thể thực hiện hoặc thực hiện nhưng cho kết quả sai khi các mẫu tin có hơn 100 cột hoặc có quá nhiều mẫu tin.
- Không thể thực hiện hoặc cho kết quả sai khi các cột có chiều dài lớn hơn 125 bytes.
- Không thể sắp xếp theo chiều tăng dần….

Tính đúng đắn của một sản phẩm phần mềm được xác minh qua các căn cứ sau đây:

- Tính đúng đắn của thuật toán.
- Tính tương đương của chương trình với thuật toán. Thuật toán có thể đúng nhưng chương trình lập ra không tương đương với thuật toán nên khi thực hiện sẽ cho kết quả sai.
- Tính đúng đắn của chương trình có thể được chứng minh trực tiếp trong văn bản của chương trình.
- Tính đúng đắn cũng có thể được khẳng định dần qua việc kiểm thử, việc áp dụng chương trình trong một khoảng thời gian dài trên diện rộng và với tần suất sử dụng cao.

1.2.2. Tính tiến hóa

Cho phép người dùng có thể khai báo các thay đổi về qui định với phần mềm tùy theo các thay đổi trong thế giới thực liên quan (thay qui định về số sách mượn tối đa, công thức tính tiền phạt, công thức tính tiền điện…)

Sản phẩm có thể mở rộng, tăng cường về mặt chức năng một cách dễ dàng.

1.2.3. Tính hiệu quả

Tính hiệu quả của một sản phẩm phần mềm được xác định qua các tiêu chuẩn sau:

- Hiệu quả kinh tế hoặc ý nghĩa, giá trị thu được do áp dụng sản phẩm đó.
- Tốc độ xử lý của phần mềm (v) tính bằng tỉ lệ giữa khối lượng đối tượng cần phải xử lý (m) và tổng thời gian (t) cần thiết để xử lý các đối tượng đó.

③ Sử dụng tối ưu tài nguyên của máy tính (CPU, bộ nhớ…)

1.2.4. Tính tiện dụng

Sản phẩm phải tính đến những yếu tố tâm lý sau đây của người dùng:

③ Dễ học, có giao diện trực quan tự nhiên.

③ Dễ thao tác,…

1.2.5. Tính tương thích

Trao đổi dữ liệu với các phần mềm khác có liên quan (nhận danh mục sách từ tập tin Excel, gửi báo cáo tổng kết năm học đến phần mềm WinFax, …)

③ Giao tiếp nội bộ

③ Giao tiếp bên ngoài

1.2.6. Tính tái sử dụng

Sản phẩm phần mềm có thể áp dụng cho nhiều lĩnh vực theo nhiều chế độ làm việc khác nhau.

③ Các phần mềm cùng lớp

③ Các phần mềm khác lớp

1.3. Công nghệ phần mềm

1.3.1. Sự ra đời

Vào những năm 1950 khi máy tính ra đời chính thức (không chỉ được dùng trong các phòng thí nghiệm mà bắt đầu ứng dụng trong hoạt động xã hội) các phần mềm đầu tiên cũng được ra đời với số lượng còn rất ít ỏi và chủ yếu phục vụ cho lĩnh vực tính toán (đặc biệt trong quốc phòng).

Đến những năm 1960, trãi qua 10 năm phát triển số lượng các phần mềm đã tăng lên rất nhiều và được ứng dụng rộng rãi trong nhiều lĩnh vực. Vào thời điểm này phát sinh một vấn đề mà các chuyên gia gọi là "cuộc khủng hoảng phần mềm". Cuộc khủng hoảng phần mềm thể hiện 2 yếu tố chính:

- Số lượng các phần mềm tăng vọt (do sự phát triển của phần cứng: tăng khả năng, giá thành hạ)
- Có quá nhiều khuyết điểm trong các phần mềm được dùng trong xã hội
    - Thực hiện không đúng yêu cầu (tính toán sai, không ổn định…)
    - Thời gian bảo trì, nâng cấp quá lâu, tốn chi phí cao, hiệu quả thấp.

- Khó sử dụng
- Thực hiện chậm
- Khó chuyển đổi dữ liệu giữa các phần mềm
- …..

Để giải quyết vấn đề trên một hội nghị đã được triệu tập đề bàn về cách giải quyết. Hội nghị đã tiến hành xem xét, phân tích và xác định nguyên nhân gây ra cuộc khủng hoảng phần mềm. Kết luận như sau:

- Việc tăng vọt của số lượng phần mềm là điều hợp lý và điều này sẽ còn tiếp diễn.
- Các khuyết điểm của phần mềm có nguồn gốc chính từ phương pháp, cách thức tiến hành xây dựng phần mềm:
    - Cảm tính: mỗi người theo một phương pháp riêng.
    - Thô sơ, đơn giản: chỉ tập trung vào việc lập trình mà ít quan tâm đến các công việc cần làm khác trước khi lập trình (khảo sát hiện trạng, phân tích yêu cầu, thiết kế…).
    - Thủ công: công cụ hỗ trợ chính khi xây dựng phần mềm chỉ là trình biên dịch.

Với các kết luận như trên, hội nghị đã đề xuất khai sinh một ngành khoa học mới: Công nghệ phần mềm với nhiệm vụ chính là nghiên cứu về các phương pháp tiến hành xây dựng phần mềm.

1.3.2. Định nghĩa

Công nghệ phần mềm là một lĩnh vực nghiên cứu của tin học nhằm đề xuất các nguyên lý, phương pháp, công cụ, cách tiếp cận phục vụ cho việc thiết kế, cài đặt các sản phẩm phần mềm đạt được đầy đủ các yêu cầu về chất lượng phần mềm.

Do quá trình tiến hóa của ngành công nghệ phần mềm nên khái niệm về nó cũng thay đổi theo thời gian. Hơn nữa do đây là một lĩnh vực mới nên các khái niệm vẫn còn phụ thuộc rất nhiều vào quan điểm chủ quan của từng người khác nhau. Cụ thể như sau:

- Bauer[1969]: việc thiết lập và sử dụng các nguyên lý công nghệ đúng đắn để thu được phần mềm một cách kinh tế vừa tin cậy vừa làm việc hiệu quả trên các máy thực.
- Ghezzi[1991]: là một lĩnh vực của khoa học máy tính liên quan đến việc xây dựng các phần mềm vừa lớn vừa phức tạp bởi một hay một số nhóm kỹ sư.
- IEEE[1993]:

1. Việc áp dụng phương pháp tiếp cận có hệ thống, bài bản và được lượng hóa trong phát triển, vận hành và bảo trì phần mềm.
2. Nghiên cứu các phương pháp tiếp cận được dùng trong (1).

- Sommervile[1995]: là lĩnh vực liên quan đến lý thuyết, phương pháp và công cụ dùng cho phát triển phần mềm.
- Kawamura[1995]: là lĩnh vực học vấn về các kỹ thuật, phương pháp luận công nghệ học (lý luận và kỹ thuật được hiện thực hóa trên các nguyên lý, nguyên tắc xác định) trong toàn bộ quy trình phát triển phần mềm nhằm nâng cao cả chất và lượng của sản xuất phần mềm.
- Pressman[1995]: là bộ môn tích hợp cả qui trình, các phương pháp, các công cụ để phát triển phần mềm máy tính.

Có thể định nghĩa tóm tắt về công nghệ phần mềm như sau: Công nghệ phần mềm là một nghành khoa học nghiên cứu về việc xây dựng các phần mềm có chất lượng trong khoảng thời gian và chi phí hợp lý.

Mục tiêu nghiên cứu được chia thành 2 phần rõ nét:

1. Xây dựng phần mềm có chất lượng.
2. Xây dựng phần mềm trong thời gian và chi phí hợp lý.

1.3.3. Đối tượng nghiên cứu

Hướng đến việc xây dựng các phần mềm có chất lượng như đã nêu, ngành công nghệ phần mềm đưa ra 3 đối tượng nghiên cứu chính: Qui trình công nghệ, Phương pháp phát triển, Công cụ và môi trường phát triển phần mềm.

- Qui trình công nghệ phần mềm: Hệ thống các giai đoạn mà quá trình phát triển phần mềm phải trải qua. Với mỗi giai đoạn cần xác định rõ mục tiêu, kết quả nhận từ giai đoạn trước đó cũng chính là kết quả chuyển giao cho giai đoạn kết tiếp.
- Phương pháp phát triển phần mềm: Hệ thống các hướng dẫn cho phép từng bước thực hiện một giai đoạn nào đó trong qui trình công nghệ phần mềm.
- Công cụ và môi trường phát triển phần mềm: Hệ thống các phần mềm trợ giúp chính trong lĩnh vực xây dựng phần mềm. Các phần mềm này sẽ hỗ trợ các chuyên viên tin học trong các bước xây dựng phần mềm theo một phương pháp nào đó với một qui trình được chọn trước.

2. QUI TRÌNH CÔNG NGHỆ PHẦN MỀM

Như đã nói để xây dựng được phần mềm có chất lượng quá trình phát triển phải trãi qua rất nhiều giai đoạn. Mỗi giai đoạn có mục tiêu và kết quả chuyển giao xác định. Trình tự thực hiện các giai đoạn này chính là chu kỳ sống của một phần mềm.

Nói cách khác, chu kỳ sống của một phần mềm là khoảng thời gian mà trong đó một sản phẩm phần mềm được phát triển, sử dụng và mở rộng cho đến khi sản phẩm phần mềm đó không còn được sử dụng nữa.

Chu kỳ sống của phần mềm được phân chia được phân chia thành các pha chính như: xác định, phát triển, kiểm thử, bảo trì (vận hành). Phạm vi và thứ tự các pha khác nhau tùy theo từng mô hình cụ thể.

2.1. Các bước cơ bản trong xây dựng phần mềm

2.1.1. Xác định

Đây là bước hình thành bài toán hoặc đề tài. Ở bước này thiết kế trưởng hoặc phân tích viên hệ thống phải biết được vai trò của phần mềm cần phát triển trong hệ thống, đồng thời phải ước lượng công việc, lập lịch biểu và phân công công việc.

Bên cạnh đó chúng ta phải biết người đặt hàng muốn gì. Các yêu cầu cần phải được thu thập đầy đủ và được phân tích theo chiều ngang (rộng) và chiều dọc (sâu). Công cụ sử dụng chủ yếu ở giai đoạn này là các lược đồ, sơ đồ phản ánh rõ các thành phần của hệ thống và mối liên quan giữa chúng với nhau.

2.1.2. Phát triển

Dựa vào các nội dung đã xác định được, nhóm phát triển phần mềm dùng ngôn ngữ đặc tả hình thức (dựa trên các kiến trúc toán học) hoặc phi hình thức (tựa ngôn ngữ tự nhiên) hoặc kết hợp cả hai để mô tả những yếu tố sau đây của chương trình:

- ③ Giá trị nhập, giá trị xuất.
- ③ Các phép biến đổi
- ③ Các yêu cầu cần đạt được ở mỗi điểm của chương trình.

Phần đặc tả chỉ quan tâm chủ yếu đến giá trị vào, ra chứ không quan tâm đến cấu trúc và nội dung các thao tác cần thực hiện.

Sau bước thiết kế là bước triển khai các đặc tả chương trình thành một sản phẩm phần mềm dựa trên một ngôn ngữ lập trình cụ thể. Trong giai đoạn này các lập trình viên sẽ tiến hành cài đặt các thao tác cần thiết để thực hiện đúng các yêu cầu đã được đặc tả.

Công việc cuối cùng của giai đoạn phát triển là chúng ta cần phải chứng minh tính đúng đắn của chương trình sau khi đã tiến hành cài đặt. Tuy nhiên thông thường ở bước này chúng ta coi các chương trình như những hộp đen. Vấn đề đặt ra là xây dựng một cách có chủ đích các tập dữ liệu nhập khác nhau để giao cho chương trình thực hiện rồi dựa vào kết quả thu được để đánh giá chương trình. Công việc như trên được gọi là kiểm thử chương trình.

Công việc kiểm thử nhằm vào các mục tiêu sau:

- ③ Kiểm tra để phát hiện lỗi của chương trình. Lưu ý rằng kiểm thử không đảm bảo tuyệt đối tính đúng đắn của chương trình do bản chất quy nạp không hoàn toàn của cách làm.
- ③ Kiểm tra tính ổn định, hiệu quả cũng như khả năng tối đa của chương trình.

Tùy theo mục đích mà người ta thiết kế các tập dữ liệu thử sao cho có thể phủ hết các trường hợp cần quan tâm.

2.1.3. Bảo trì (Vận hành)

Công việc quản lý việc triển khai và sử dụng phần mềm cũng là một vấn đề cần được quan tâm trong qui trình phát triển phần mềm. Trong quá trình xây dựng phần mềm, toàn bộ các kết quả phần tích, thiết kế, cài đặt và hồ sơ liên quan cần phải được lưu trữ và quản lý cẩn thận nhằm đảm bảo cho công việc được tiến hành một cách hiệu quả nhất và phục vụ cho công việc bảo trì phần mềm về sau.

Như vậy công việc quản lý không chỉ dừng lại trong quá trình xây dựng phần mềm mà trái lại còn phải được tiến hành liên tục trong suốt quá trình sống của nó.

2.2. Một số mô hình triển khai xây dựng phần mềm

Có nhiều mô hình cận khác nhau để triển khai các bước cơ bản trong quá trình phát triển phần mềm. Mỗi mô hình sẽ chia vòng đời của phần mềm theo một cách khác nhau nhằm đảm bảo qui trình phát triển phần mềm sẽ dẫn đến thành công. Trong phần tiếp theo của giáo trình chúng ta sẽ tìm hiểu qua các mô hình phát triển phần mềm tiêu biểu nhất đang được áp dụng.

2.2.1. Mô hình thác nước:

Mô hình thác nước là một trong những mô hình đầu tiên và phổ biến được áp dụng trong quá trình phát triển phần mềm. Mô hình này chia quá trình phát triển phần mềm thành những giai đoạn tuần tự nối tiếp nhau. Mỗi giai đoạn sẽ có một mục đích nhất định. Kết quả cuả giai đoạn trước sẽ là thông tin đầu vào cho giai đoạn tiếp theo sau. Tùy theo qui mô của phần mềm cần phát triển mà mô hình thác nước sẽ có những biến thể khác nhau như sau:

☞ Qui trình 2 giai đoạn: Là qui trình đơn giản nhất. Theo qui trình này việc phát triển phần mềm chỉ trãi qua 2 giai đoạn:

- Xác định yêu cầu: Được tiến hành ngay khi có nhu cầu về việc xây dựng phần mềm.
  - Mục tiêu: Xác định chính xác các yêu cầu đặt ra cho phần mềm sẽ xây dựng.
  - Kết quả nhận: Thông tin về hoạt động của thế giới thực.
  - Kết quả chuyển giao: Danh sách các yêu cầu (công việc sẽ thực hiện trên máy tính) cùng với các thông tin miêu tả chi tiết về các yêu cầu (cách thức thực hiện trong thế giới thực).
- Lập trình (cài đặt): Được tiến hành ngay sau khi kết thúc việc xác định yêu cầu.
  - Mục tiêu: Tạo lập phần mềm mong muốn theo yêu cầu.
  - Kết quả nhận: Danh sách các yêu cầu cùng các thông tin có liên quan.
  - Kết quả chuyển giao: Chương trình nguồn của phần mềm với cấu trúc cơ sở dữ liệu tương ứng (nếu cần thiết) và chương trình thực hiện được trên máy tính (chương trình nguồn đã được biên dịch)

☞ Qui trình 3 giai đoạn: Là qui trình cải tiến của qui trình 2 giai đoạn bằng cách bổ sung thêm một giai đoạn trung gian mới giữa xác định yêu cầu và lập trình (có sửa đổi)

- Xác định yêu cầu: được tiến hành ngay khi có nhu cầu về việc xây dựng phần mềm.
  - Mục tiêu: Xác định chính xác các yêu cầu đặt ra cho phần mềm sẽ xây dựng.
  - Kết quả nhận: Thông tin về hoạt động của thế giới thực.
  - Kết quả chuyển giao: Danh sách các yêu cầu (công việc sẽ thực hiện trên máy tính) cùng với các thông tin miêu tả chi tiết về các yêu cầu (cách thức thực hiện trong thế giới thực)
- Thiết kế: Được tiến hành ngay sau khi kết thúc việc xác định yêu cầu.

- Mục tiêu: Mô tả các thành phần của phần mềm (mô hình của phần mềm) trước khi tiến hành cài đặt.
- Kết quả nhận: Danh sách các yêu cầu và thông tin liên quan.
- Kết quả chuyển giao:
  - ③ Mô tả thành phần giao diện: các hàm nhập/xuất, cấu trúc dữ liệu nhập/xuất.
  - ③ Mô tả thành phần xử lý: các hàm kiểm tra xử lý.
  - ③ Mô tả thành phần dữ liệu: các hàm đọc/ ghi, tổ chức lưu trữ trên bộ nhớ phụ.

o Lập trình (cài đặt): Được tiến hành ngay sau khi kết thúc việc thiết kế.
- Mục tiêu: Tạo lập phần mềm theo yêu cầu.
- Kết quả nhận: Mô hình phần mềm
- Kết quả chuyển giao: Chương trình nguồn của phần mềm với cấu trúc cơ sở dữ liệu tương ứng (nếu cần thiết) và chương trình thực hiện được trên máy tính (chương trình nguồn đã được biên dịch)

☞ Qui trình 4 giai đoạn: Là qui trình cải tiến của qui trình phía trước bằng cách bổ sung thêm một giai đoạn mới giữa xác định yêu cầu và thiết kế (có sửa đổi)

o Xác định yêu cầu: Được tiến hành ngay khi có nhu cầu về việc xây dựng phần mềm.
- Mục tiêu: Xác định chính xác các yêu cầu đặt ra cho phần mềm sẽ xây dựng.
- Kết quả nhận: Thông tin về hoạt động của thế giới thực.
- Kết quả chuyển giao: Danh sách các yêu cầu (công việc sẽ thực hiện trên máy tính) cùng với các thông tin miêu tả chi tiết về các yêu cầu (cách thức thực hiện trong thế giới thực)

o Phân tích: được tiến hành ngay sau khi kết thúc việc xác định yêu cầu.
- Mục tiêu: Mô tả lại thế giới thực thông qua các mô hình (mô hình thế giới thực) trước khi thiết kế.
- Kết quả nhận: Danh sách các yêu cầu cùng các thông tin có liên quan.
- Kết quả chuyển giao:
  - ③ Mô hình xử lý (hệ thống các công việc trong thế giới thực cùng với quan hệ giữa chúng)
  - ③ Mô hình dữ liệu (hệ thống các loại thông tin được sử dụng trong thế giới thực cùng với quan hệ giữa chúng)

③ Các mô hình khác (không gian, thời gian, con người…) nếu cần thiết.

- Thiết kế: Được tiến hành ngay sau khi kết thúc việc phân tích.
  - Mục tiêu: Mô tả các thành phần của phần mềm (mô hình của phần mềm) trước khi tiến hành cài đặt.
  - Kết quả nhận: Mô hình thế giới thực.
  - Kết quả chuyển giao:
    - ③ Mô tả thành phần giao diện: các hàm nhập/xuất, cấu trúc dữ liệu nhập/xuất.
    - ③ Mô tả thành phần xử lý: các hàm kiểm tra xử lý.
    - ③ Mô tả thành phần dữ liệu: các hàm đọc/ghi, tổ chức lưu trữ trên bộ nhớ phụ.
- Lập trình (cài đặt): Được tiến hành ngay sau khi kết thúc việc thiết kế.
  - Mục tiêu: Tạo lập phần mềm theo yêu cầu
  - Kết quả nhận: Mô hình phần mềm
  - Kết quả chuyển giao: Chương trình nguồn của phần mềm với cấu trúc cơ sở dữ liệu tương ứng (nếu cần thiết) và chương trình thực hiện được trên máy tính (chương trình nguồn đã được biên dịch)

☞ Qui trình 5 giai đoạn: Là qui trình cải tiến của qui trình phía trước bằng cách bổ sung thêm một giai đoạn mới sau giai đoạn lập trình nhằm tăng cường độ tin cậy của phần mềm.

- Xác định yêu cầu: Được tiến hành ngay khi có nhu cầu về việc xây dựng phần mềm.
  - Mục tiêu: Xác định chính xác các yêu cầu đặt ra cho phần mềm sẽ xây dựng.
  - Kết quả nhận: Thông tin về hoạt động của thế giới thực.
  - Kết quả chuyển giao: Danh sách các yêu cầu (công việc sẽ thực hiện trên máy tính) cùng với các thông tin miêu tả chi tiết về các yêu cầu (cách thức thực hiện trong thế giới thực)
- Phân tích: được tiến hành ngay sau khi kết thúc việc xác định yêu cầu.
  - Mục tiêu: Mô tả lại thế giới thực thông qua các mô hình (mô hình thế giới thực) trước khi thiết kế.
  - Kết quả nhận: Danh sách các yêu cầu cùng các thông tin có liên quan.
  - Kết quả chuyển giao:

- ③ Mô hình xử lý (hệ thống các công việc trong thế giới thực cùng với quan hệ giữa chúng)
- ③ Mô hình dữ liệu (hệ thống các loại thông tin được sử dụng trong thế giới thực cùng với quan hệ giữa chúng)
- ③ Các mô hình khác (không gian, thời gian, con người…) nếu cần thiết.

o Thiết kế: Được tiến hành ngay sau khi kết thúc việc phân tích.

- Mục tiêu: Mô tả các thành phần của phần mềm (mô hình của phần mềm) trước khi tiến hành cài đặt.
- Kết quả nhận: Mô hình thế giới thực.
- Kết quả chuyển giao:
  - ③ Mô tả thành phần giao diện: các hàm nhập/xuất, cấu trúc dữ liệu nhập/xuất.
  - ③ Mô tả thành phần xử lý: các hàm kiểm tra xử lý.
  - ③ Mô tả thành phần dữ liệu: các hàm đọc/ ghi, tổ chức lưu trữ trên bộ nhớ phụ.

o Lập trình (cài đặt): Được tiến hành ngay sau khi kết thúc việc thiết kế.

- Mục tiêu: Tạo lập phần mềm theo yêu cầu.
- Kết quả nhận: Mô hình phần mềm.
- Kết quả chuyển giao: Chương trình nguồn của phần mềm với cấu trúc cơ sở dữ liệu tương ứng (nếu cần thiết) và chương trình thực hiện được trên máy tính (chương trình nguồn đã được biên dịch).

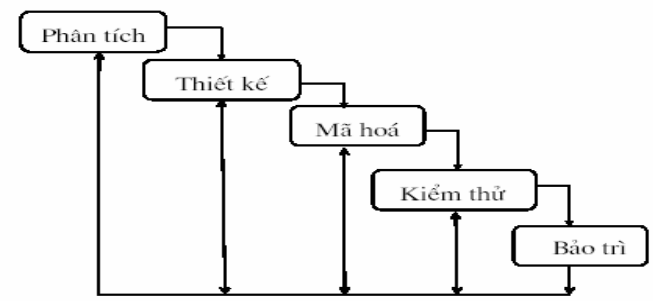
o Kiểm thử: Được tiến hành ngay sau khi đã có kết quả (từng phần) của việc lập trình.

- Mục tiêu: Tăng độ tin cậy của phần mềm.
- Kết quả nhận:
  - ③ Danh sách yêu cầu.
  - ③ Mô hình phần mềm.
  - ③ Phần mềm.
- Kết quả chuyển giao: Phần mềm với độ tin cậy cao (đã tìm và sửa lỗi).

o Bảo trì: Công việc của giai đoạn bao gồm việc cài đặt và vận hành phần mềm trong thực tế.

- Mục tiêu: đảm bảo phần mềm vận hành tốt
- Kết quả nhận: phần mềm đã hoàn thàng

- Kết quả chuyển giao: các phản ánh của khách hàng trong quá trình sử dụng phần mềm.



Mô hình vòng đời cổ điển (thác nước)

Nhận xét:

Mô hình thác nước giúp chúng ta có thể dễ dàng phân chia quá trình xây dựng phần mềm thành những giai đoạn hoàn toàn độc lập nhau. Tuy nhiên, các dự án lớn hiếm khi tuân theo dòng chảy tuần tự của mô hình vì thường phải lặp lại các bước để nâng cao chất lượng. Hơn nữa, khách hàng hiếm khi tuyên bố hết các yêu cầu trong giai đoạn phân tích.

Mô hình này cũng có một hạn chế là chúng ta rất khó thực hiện các thay đổi một khi đã thực hiện xong một giai đoạn nào đó. Điều này làm cho việc xây dựng phần mềm rất khó thay đổi các yêu cầu theo ý muốn của khách hàng. Do đó, phương pháp này chỉ thích hợp cho những trường hợp mà chúng ta đã hiểu rất rõ các yêu cầu của khách hàng.

Chú ý: Mô hình thác nước có thể được cải tiến bằng cách cho phép quay lui khi phát hiện lỗi trong giai đoạn phía trước.

2.2.2. Mô hình bản mẫu phần mềm

Tương tự như mô hình thác nước với bổ sung vào các giai đoạn thực hiện phần mềm mẫu ngay khi xác định yêu cầu nhằm mục tiêu phát hiện nhanh các sai sót về yêu cầu. Các giai đoạn trong mô hình bản mẫu phần mềm có thể tiến hành lặp đi lặp lại chứ không nhất thiết phải theo trình tự nhất định.

Ngay sau khi giai đoạn xác định yêu cầu, nhà phát triển phần mềm đưa ra ngay một bản thiết kế sơ bộ và tiến hành cài đặt bản mẫu đầu tiên và chuyển cho người sử dụng. Bản mẫu này chỉ nhằm để miêu tả cách thức phần mềm hoạt động cũng như cách người sử dụng tương tác với hệ thống.

Người sử dụng sau khi xem xét sẽ phản hồi thông tin cần thiết lại cho nhà phát triển. Nếu người sử dụng đồng ý với bản mẫu đã đưa thì người phát triển sẽ tiến hành cài đặt thực

sự. Ngược lại cả hai phải quay lại giai đoạn xác định yêu cầu. Công việc này được lặp lại liên tục cho đến khi người sử dụng đồng ý với các bản mẫu do nhà phát triển đưa ra.

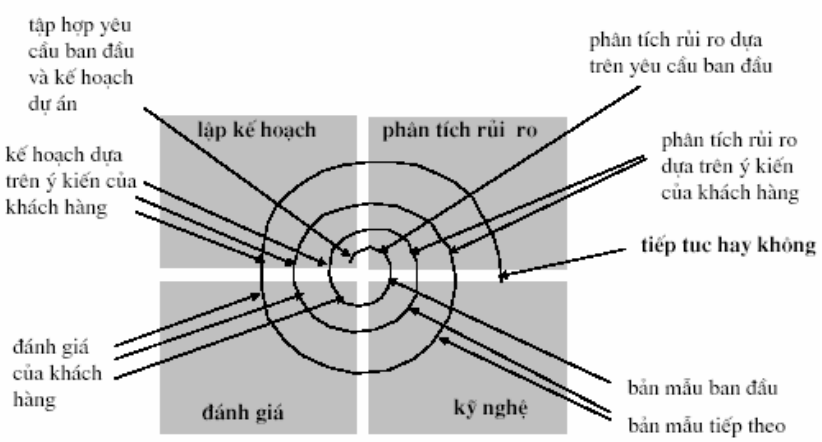
Như vậy đây là một hướng tiếp cận tốt khi các yêu cầu chưa rõ ràng và khó đánh giá được tính hiệu quả của các thuật toán. Tuy nhiên, mô hình này cũng có nhược điểm là tính cấu trúc không cao do đó khách hàng dễ mất tin tưởng.

Mô hình làm bản mẫu

2.2.3. Mô hình xoắn ốc

Mô hình này chính là sự kết hợp của mô hình bản mẫu thiết kế và mô hình thác nước được lặp lại nhiều lần. Ở lần lặp tiếp theo hệ thống sẽ được tìm hiểu và xây dựng hoàn thiện hơn ở lần lặp trước đó.

Ở mỗi lần lặp các yêu cầu của người sử dụng sẽ được hiểu ngày càng rõ ràng hơn và các bản mẫu phần mềm cũng ngày một hoàn thiện hơn. Ngoài ra ở cuối mỗi lần lặp sẽ có thêm công đoạn phân tích mức độ rủi ro để quyết định xem có nên đi tiếp theo hướng này nữa hay không.



Mô hình xoắn ốc

Mô hình này phù hợpvới các hệ thống phần mềm lớn do có khả năng kiểm soát rủi ro ở từng bước tiến hóa. Tuy nhiên vẫn chưa được sử dụng rộng rãi như mô hình thác nước hoặc bản mẫu do đòi hỏi năng lực quản lý, năng lực phân tích rủi ro cao.

3. CÁC PHƯƠNG PHÁP XÂY DỰNG PHẦN MỀM

3.1. Tổng quan

3.1.1. Khái niệm

Để tiến hành xây dựng một phần mềm, chúng ta có thể áp dụng nhiều phương pháp khác nhau. Mỗi phương pháp có những ưu và khuyết điểm riêng và phù hợp với từng loại phần mềm cụ thể.

Mỗi phương pháp sẽ có những hướng dẫn cụ thể các công việc cần phải thực hiện trong từng giai đoạn trong quy trình xây dựng phần mềm.

Bên cạnh đó mỗi phương pháp cũng sẽ quy định những cách thức khác nhau để trình bày các kết quả thu được trong quá trình xây dựng phần mềm. Những quy định này có tính chất như là ngôn ngữ thống nhất để các thành viên tham gia xây dựng phần mềm có thể trao đổi thông tin trong việc xây dựng phần mềm.

3.1.2. Phân loại

Các phương pháp xây dựng phần mềm được chia làm 02 nhóm khác nhau dựa vào tính chất của công việc cần thực hiện.

- Phương pháp xây dựng:
  - Phương pháp hướng chức năng

- Phương pháp hướng dữ liệu
- Phương pháp hướng đối tượng

☞ Phương pháp tổ chức quản lý

- Xây dựng phương án
- Tổ chức nhân sự
- Ước lượng rủi ro, chi phí
- Lập và theo dõi kế hoạch triển khai.

Trong phần tiếp theo của giáo trình này, chúng ta chỉ quan tâm đến các phương pháp xây dựng. Về phương pháp tổ chức quản lý chúng ta có thể tham khảo trong giáo trình "Quản lý dự án xây dựng các hệ thống thông tin".

3.2. Các phương pháp xây dựng phần mềm

3.2.1. Cách tiếp cận

a) Từ trên xuống

Đây là cách giải quyết vấn đề theo hướng phân tích. Khi tiến hành xây dựng phần mềm theo cách này, chúng ta bắt đầu với những thành phần chính của hệ thống. Sau đó, các thành phần này sẽ được phân tích thành các thành phần chi tiết và cụ thể hơn. Quá trình phân tích này sẽ kết thúc khi các kết quả thu được có mức độ phức đạp đúng với ý muốn của nhà xây dựng phần mềm.

b) Từ dưới lên

Ngược lại với phương pháp từ trên xuống, phương pháp từ dưới lên là cách giải quyết vấn đề theo hướng tổng hợp. Với phương pháp này, chúng ta tiến hành xây dựng những thành phần chi tiết, cụ thể mà mà chúng ta dự tính là sẽ có trong hệ thống. Sau đó, các nhà phát triển phần mềm sẽ tiến hành kết hợp các thành phần chi tiết này lại với nhau để tạo nên các thành phần chính mà hệ thống cần phải có.

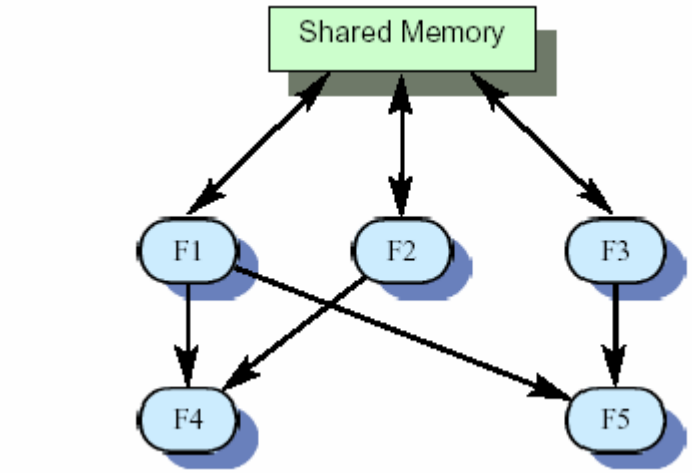
3.2.2. Cách tiến hành

a) Phương pháp hướng chức năng

Với phương pháp này công việc xây dựng phần mềm được thực hiện dựa trên các chức năng mà hệ thống cần thực hiện. Hay nói cách khác chúng ta chú trọng đến thành phần xử lý của hệ thống:

- Các thao tác tính toán
- Các thao tác phát sinh
- Các thao tác biến đổi….

Phương pháp chung để giải quyết vấn đề là áp dụng nguyên lý "chia để trị". Khi tiến hành xây dựng phần mềm theo phương pháp này, chúng ta sẽ chia các công việc lớn mà hệ thống cần thực hiện hành các công việc nhỏ hơn độc lập nhau. Việc phân chia các công việc được tiến hành cho đến khi các công việc thu được đủ nhỏ để chúng ta có thể tiến hành xây dựng hoàn chỉnh. Hình dưới: Minh họa cách tiếp cận theo hướng chức năng.



Phương pháp hướng chức năng chú trọng đến cách để giải quyết vấn đề nhưng không có khả năng che dấu các thông tin trạng thái của hệ thống. Điều này dẫn đến việc các chức năng trong hệ thống không tương thích với nhau trong việc thực hiện thay đổi các thông tin trong hệ thống. Chính vì vậy mà cách tiếp cận này chỉ thích hợp khi trong hệ thống có rất ít thông tin cần phải quản lý và chia sẻ giữa các chức năng với nhau. Để mô hình hóa cách xử lý thông tin trong hệ thống dùng lược đồ dòng dữ liệu (Data Flow Diagrams).

DFD là một công cụ đơn giản và hữu ích để miêu tả cách thức hoạt động của hệ thống. DFD sử dụng các ký hiệu sau để mô tả hệ thống:

- Ô vuông có góc tròn được dùng để biểu diễn các chức năng của hệ thống.

- Ô vuông dùng để biểu diễn thành phần dữ liệu trong hệ thống.
- Hình tròn dùng để biểu diễn các thành phần bên ngoài có giao tiếp với hệ thống.
- Dấu mũi tên dùng để biểu diễn hướng di chuyển của dữ liệu.
- Các từ khóa "and" và "or" dùng để liên kết các dòng dữ liệu khi cần thiết.

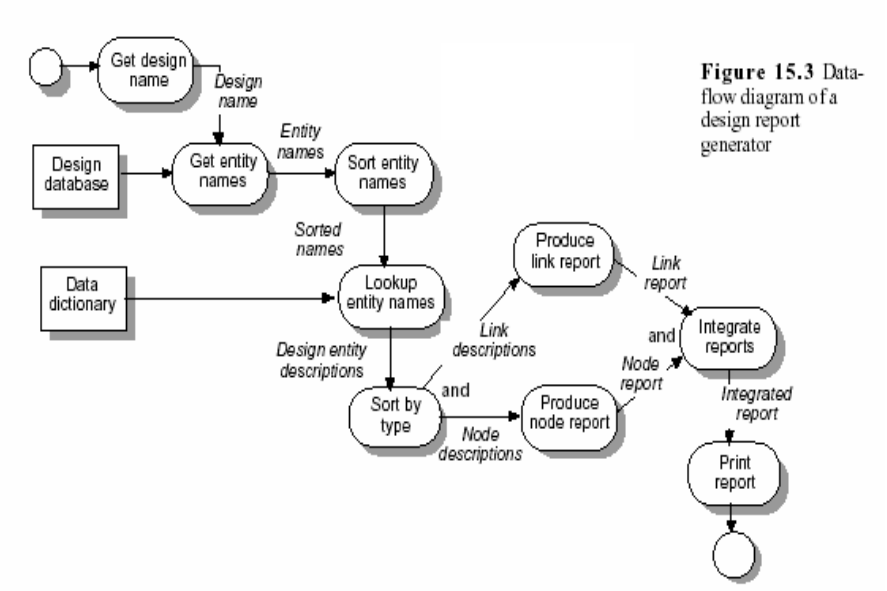


Figure 15.3 Data-flow diagram of a design report generator

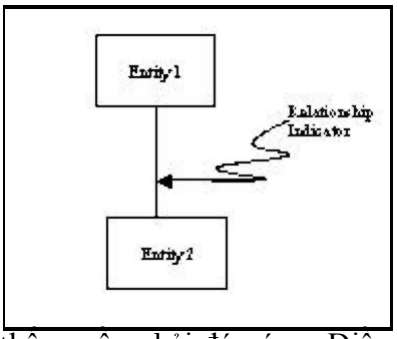
b) Phương pháp hướng dữ liệu

Ngược lại với phương pháp hướng chức năng, phương pháp hướng dữ liệu chú trọng nhiều đến thành phần dữ liệu cần phải xử lý trong hệ thống:

- Tổ chức dữ liệu
- Khối lượng lưu trữ
- Tốc độ truy xuất
- …

Khi tiến hành thiết kế theo phương pháp hướng dữ liệu chúng ta bắt đầu với việc thiết kế các cấu trúc dữ liệu cần thiết có trong bài toán, sau đó mới tiến hành thiết kết các thao tác để vận hành trên các cấu trúc dữ liệu đã thiết kế.

Phương pháp này đặc biệt chỉ thích hợp trong các loại phần mềm chỉ có chức năng chính là lưu trữ và thao tác trên các loại dữ liệu. Hạn chế của nó là không quan tâm đến các



chức năng mà hệ thống cần phải đáp ứng. Điều này dẫn đến việc có khả năng hệ thống sau khi thiết kế không có đầy đủ các chức năng cần thiết.

Kết quả thu được sau khi thiết kế theo phương pháp hướng dữ liệu là mô hình thực thể kết hợp (Entity Relationship Diagram). Một mô hình thực thể kết hợp điển hình gồm có 2 thành phần cơ bản: các thực thể và các mối kết hợp.

- Một thực thể là một đối tượng trong thế giới thực mà hệ thống có quan hệ, hoặc tương tác qua lại. Các thực thể được biểu diễn trong sơ đồ bằng các hình vuông cùng với tên và có thể có thêm các thuộc tính của thực thể.
- Mối kết hợp biểu diễn sự kết hợp giữa hai hay nhiều thực thể. Mỗi mối kết hợp gồm có ba thành phần cơ bản:
  - ③ Mối kết hợp giữa các thực thể được biểu diễn băng một đường thẳng nối giữa hai thực thể.
  - ③ Tên của môi liên hệ dùng để miêu tả ý nghĩa của mối liên hệ.
  - ③ Bản số ở hai đầu của mối kết hợp dùng để xác định con số tối đa và tối thiểu các thực thể liên quan đến mối kết hợp.

c) Phương pháp hướng đối tượng

Phương pháp thiết kế hướng đối tượng là sự kết hợp của phương pháp hướng dữ liệu và phương pháp hướng chức năng. Phương pháp này chú trọng đến cả thành phần dữ liệu và chức năng của hệ thống.

Theo phương pháp hướng đối tượng thì một hệ thống phần mềm là một tập hợp các đối tượng có khả năng tương tác với nhau. Các đối tượng chính là các sự vật và hiện tượng vật lý cũng như trừu tượng mà chúng ta có trong thế giới thực. Mỗi đối tượng có dữ liệu riêng được che dấu với thế giới bên ngoài và các thao tác mà đối tượng có thể thực hiện trên các thành phần dữ liệu của đối tượng.

Các đối tượng liên lạc, trao đổi thông tin với nhau bằng cách gửi các thông điệp cho nhau. Các thông điệp mà mỗi đối tượng có thể xử lý được gọi là giao diện của đối tượng. Khi đó mọi thao tác liên quan đến các đối tượng được phải thực hiện thông qua giao diện của đối tượng. Điều này giúp chúng ta đảm bảo rằng các thông tin bên trong các đối tượng được bảo vệ một cách chắc chắn.

Chúng ta có thể sử dụng nhiều hệ thống ký hiệu khác nhau để mô tả các đối tượng của hệ thống cũng như mối liên hệ giữa chúng. Một trong số các hệ thống ký hiệu phổ biến hiện nay là hệ thốnng ký hiệu UML.

4. CÔNG CỤ VÀ MÔI TRƯỜNG PHÁT TRIỂN PHẦN MỀM

4.1. Mở đầu

4.1.1. Khái niệm

Các công cụ và môi trường phát triển phần mềm là các phần mềm hỗ trợ chính người phát triển trong quá trình xây dựng phần mềm. Các phần mềm này có tên gọi chung là CASE (Computer Aided Software Engineering) tools.

Trong quá trình phát triển phần mềm theo các quy trình trên, các CASE tools có thể hỗ trợ cụ thể cho một giai đoạn nào đó hay cũng có thể hỗ trợ một số giai đoạn, trong trường hợp này tên gọi chung thường là môi trường phát triển phần mềm-SDE (Software Development Environment).

Việc hỗ trợ của các CASE tools trong một giai đoạn bao gồm 2 hình thức chính:

- Cho phép lưu trữ, cập nhật trên kết quả chuyển giao với một phương pháp nào đó.
- Cho phép phát sinh ra kết quả chuyển giao cho giao đoạn kế tiếp.

4.2. Phần mềm hỗ trợ thực hiện các giai đoạn

4.2.1. Phần mềm hỗ trợ phân tích

- Công việc hỗ trợ chính
    - Soạn thảo các mô hình thế giới thực
    - Ánh xạ vào mô hình logic
- Các phần mềm: WinA&D, Analyst Pro,…

4.2.2. Phần mềm hỗ trợ thiết kế

- Công việc hỗ trợ chính
    - Soạn thảo các mô hình logic

- Ánh xạ vào mô hình vật lý

- Các phần mềm: QuickUML, Power Designer, Oracle Designer…

4.2.3. Phần mềm hỗ trợ lập trình

- Công việc hỗ trợ chính
  - Quản lý các phiên bản (Dữ liệu, chương trình nguồn, giao diện)
  - Biên dịch
- Các phần mềm: Visual Studio, Visual Basic, Visual C++

4.2.4. Phần mềm hỗ trợ kiểm chứng

- Công việc hỗ trợ chính
  - Phát sinh tự động các bộ dữ liệu thử nghiệm
  - Phát hiện lỗi
- Các phần mềm: WinRuner

4.3. Phần mềm hỗ trợ tổ chức, quản lý việc triển khai

4.3.1. Xây dựng phương án

- Công việc hỗ trợ chính
  - Tạo lập phương án
  - Dự đoán rủi ro
  - Tính chi phí
- Các phần mềm: MS Project, Visio

4.3.2. Lập kế hoạch

- Công việc hỗ trợ chính
  - Xác định các công việc
  - Phân công
  - Lập lịch biểu
  - Theo dõi thực hiện
- Các phần mềm: MS Project, Visio

Chương 2: PHÂN TÍCH VÀ ĐẶC TẢ YÊU CẦU

1. Tổng quan

Phân tích yêu cầu là khâu kỹ thuật đầu tiên gồm nhiều bước nhỏ: *nghiên cứu khả thi, phân tích mô hình hóa, đặc tả thẩm định yêu cầu.* Gia đoạn này được tiến hành phối hợp giữa bên phát triển và khách hàng và nó có vai trò đặc biệt quan trọng trong tiến trình phát triển phần mềm.

Đây là bước hình thành bài toán hoặc đề tài. Ở bước này trưởng nhóm thiết kế và người phân tích hệ thống phải biết được người đặt hàng muốn gì. Các yêu cầu phải được thu thập đầy đủ và được phân tích theo chiều ngang (rộng) và dọc (sâu). Công cụ sử dụng chủ yếu ở giai đoạn này là các lược đồ, sơ đồ phản ánh rõ các đối tượng của hệ thống: lưu đồ (Flowchart), sơ đồ dòng dữ liệu (Data Flow diagram DFD), mạng thực thể-quan hệ (Entity-Relationship Network), sơ đồ cấu trúc phân cấp (Structural hierarchical schemes), mạng ngữ nghĩa (Semantic Network)

1.1 Quá trình phân tích

1.1.1 Phân tích phạm vi dự án

Người phân tích hệ thống dùng thuật ngữ phạm vi để chỉ trách nhiệm dự án phải thực thi. Ngược lại, phạm vi dự án là nhiệm vụ lớn và phức tạp được thực hiện bởi chương trình.

Để xác định phạm vi dự án, bằng xác định quá trình nghiệp vụ ứng dụng sẽ đối đầu. Đó là những phạm vi vấn đề của ứng dụng. Nói chung, có hai phần đối với bất kỳ giải pháp nghiệp vụ: phần triển khai ứng dụng và phần thực hiện bởi con người hay chương trình. Định ra ranh giới ứng dụng tức là xác định qui trình trách nhiệm.

Một khi đã định nghĩa trách nhiệm của dự án:

- Chia trách nhiệm thành những nhiệm vụ con để đưa ra ý tưởng cho chính mình về bao nhiêu mô đun chương trình khác nhau yêu cầu?
- Xác định bao nhiêu vùng địa lý liên quan (chi nhánh văn phòng).
- Ước lượng số người dùng ứng dụng và thời gian ứng dụng được duy trì.
- Tính chính xác.
- Cuối cùng, hiểu khách hàng mong đợi dự án được triển khai.

Tại thời điểm này, chúng ta có ý tưởng phạm vi dự án. Cân nhắc độ lớn dự án đối với thời gian và ràng buộc ngân sách. Nếu dự án quá lớn về thời gian và tiền bạc cho chi trả thì

bàn bạc vấn đề với khách hàng để đưa ra quyết định thương lượng cho thõa đáng. Chúng ta phải chọn lựa hoặc nhiều thời gian hơn, hoặc nhiều tiền hơn hoặc cả hai. Hoặc chúng ta phải giảm phạm vi dự án xuống. Phân tích tất cả những tình huống ở giai đoạn đầu của dự án sẽ làm cho dự án thành công nhiều hơn.

1.1.2 Phân tích mở rộng yêu cầu nghiệp vụ

a. Xác định yêu cầu nghiệp vụ

Mỗi dự án sẽ có một hay nhiều yêu cầu nghiệp vụ. Mỗi yêu cầu nghiệp vụ là một mô tả tác nhiệm cụ thể trong nghiệp vụ của khách hàng. Ví dụ. lưu vết quá trình đầu tư. Một tác vụ như kiểm soát đầu tư cần chia nhỏ thành những phần chắc chắn cho đến khi mỗi phần đủ để mô tả công việc chính xác

Khi mức độ của thành phần chia nhỏ dưới mức tối thiểu, xác định lại trình tự thành phần.

Mỗi tác vụ được gọi là yêu cầu nghiệp vụ hay quy tắc nghiệp vụ. Quy tắc doanh nghiệp được viết theo ngôn ngữ được hiểu bởi những người không chuyên máy tính sao cho người dùng có thể kiểm tra luật một cách chính xác

b. Xác định yêu cầu chất lượng khách hàng

Mỗi dự án phần mềm có thể yêu cầu nhanh, bảo mật, phụ thuộc, dễ dùng, hay bug-free. Trong thế giới thực, thời gian và ràng buộc tài chính làm cho không thể tạo ra những chương trình dự án hoàn chỉnh. Thay vào đó, điều quan trọng để quyết định dựa trên mức độ chấp nhận của chất lượng thõa mãn khách hàng.

Ví dụ: khi khách hàng quyết định ứng dụng phải sẵn sàng 23 giờ trong ngày, bỏ qua thời gian vận hành không giảm. Chất lượng khác bao gồm số người dùng truy cập hiện hành, thời gian tối đa phải chờ để hoàn thành công việc trong ứng dụng (sự phản hồi), độ bảo mật ứng dụng, hay hơn nữa.

c.Phân tích hạ tầng cơ sở hiện hành

Phần quan trọng trong thiết kế giải pháp là phân tích kỹ thuật thay thế. Điển hình, giải pháp phần mềm được đưa vào hơn là thay thế hệ thống hiện hành. Dự án cần làm việc trên phần cứng và phần mềm mà người dùng hiện có. Biết được hệ điều hành đang được cài trên máy của người dùng, loại mạng đang sử dụng, và nếu người dùng đang chạy phần mềm không tương thích với chương trình mới hơn. Nên bỏ thời gian tìm hiểu máy chủ hiện hành, hệ điều hành, phần mềm đang chạy.

Khi đưa giải pháp, nhớ rằng cơ sở hạ tầng hiện hành đảm bảo giải pháp của chúng ta có thể tương thích.

d. Phân tích ảnh hưởng kỹ thuật

Nếu cần mở rộng chức năng cho hệ thống hiện hành, chúng ta mong ước thay đổi hệ thống cũ cả việc cải thiện hệ thống cũ và tích hợp dễ dàng hơn hệ thống mới. Ví dụ, chức năng của chương trình kế toán lưu trữ dữ liệu nhỏ như CSDL hướng đến tập tin Access. Để tạo dữ liệu truy xuất hiệu quả hơn và thõa mãn yêu cầu của giải pháp mới, chúng ta mới chuyển toàn bộ dữ liệu sang hệ quản trị csdl SQL Server. Việc suy nghĩ trước sẽ tiết kiệm thời gian sau đó: trãi qua thời gian tìm hiểu sự khác biệt về giao tác, bảo mật, và những chức năng khác giữa kỹ thuật cũ và giải pháp mới.

Chúng ta nên tìm hiểu thủ tục chuyển đổi dữ liệu từ kỹ thuật cũ sang kỹ thuật mới. Đảm bảo được phép thực nghiệm những thủ tục này, và có kế hoạch bảo lưu trong trường hợp thực hiện vấn đề này bị lỗi. Đảm bảo chắc chắn những tác động chuyển đổi trên mọi thành phần của hệ thống, không chỉ phần tử gần nhất thay đổi.

1.1.3.Phân tích yêu cầu bảo mật

Khi hệ thống lưu trữ, truy xuất dữ liệu cá nhân như thông tin nhân sự, thẻ tín dụng, doanh số bán hay thông tin riêng tư, chúng ta cần có biện pháp đảm bảo an toàn những dữ liệu này.

a. Xác định vai trò

Toàn bộ ứng dụng không chỉ có 1 mức độ bảo mật. Người dùng cuối chỉ cần quyền truy xuất giới hạn vào hệ thống. Quản trị hệ thống, người thao tác viên cập nhật, và người dùng có quyền truy cập cao hơn ở mọi cấp độ. Bảo mật dựa trên vai trò là kỹ thuật dùng để cấp quyền mức độ bảo mật khác nhau tương ứng quyền hạn và độ chuyên nghiệp của mỗi người dùng trong hệ thống.

Lưu ý: Nhận biết những lớp chính của những người dùng cần truy cập đến ứng dụng của chúng ta. Gán tên vai trò cho mỗi lớp người dùng. Cuối cùng, gán mức độ tối thiểu có thể truy xuất đến mỗi vai trò. Mỗi lớp người dùng nên có đủ quyền truy xuất đến công việc của họ, và không nhiều hơn.

b. Xác định môi trường bảo mật ứng dụng

Độ bảo mật không bị giới hạn người dùng hệ thống. Chỉ người dùng đăng nhập vào ứng dụng, ứng dụnng phải “login” để kiểm soát tài nguyên chia sẻ như tập tin, dịch vụ hệ thống,

cơ sở dữ liệu. Mức độ kiểm soát của ứng dụng được gọi là ngữ cảnh bảo mật. Chúng ta cần phải làm việc với nhiều người dùng khác như quản trị mạng, cấp quyền truy xuất phù hợp ứng dụng để chia sẻ tài nguyên.

c. Xác định ảnh hưởng bảo mật

Nếu công ty có sẵn cơ chế bảo mật thay vào đó hệ thống của chúng ta nên điều chỉnh cho phù hợp với cơ chế đã có. Nếu chúng ta đang thực thi hệ thống bảo mật mới hay một hệ thống khác, cần phải phân tích tác động của hệ thống trên hệ thống hiện tại:

- Hệ thống mới có làm hỏng chức năng của phần mềm hiện tại?
- Hệ thống đòi hỏi phải hỗ trợ thêm một phần người dùng – đăng nhập mở rộng ?
- Hệ thống sẽ khóa một vài người dùng trên những tập tin hay những tài nguyên mà họ được quyền truy cập trước đây

d. Kế hoạch vận hành

Khi tổ chức phát triển và thay đổi, người dùng mới được thêm vào, người cũ được cập nhật và bỏ đi. Những thao tác này đòi hỏi thay đổi CSDL bảo mật, đó là nơi thông tin người dùng và quyền hạn truy cập của họ được lưu. Những thông tin này được lưu trữ hiện thời.

Nếu người dùng có vị trí địa lý khác nhau, ở văn phòng khác nhau, chúng ta cần lên kế hoạch tái tạo cơ sở dữ liệu bảo mật. Sự tái tạo là sự thay đổi hệ thống dữ liệu tại nơi này sao chép đến nơi khác sao cho tất cả thông tin bảo mật được lưu giữ mỗi nơi. Thuận lợi việc tạo bản sao là người dùng có thể đăng nhập dùng thông tin được lưu ở vị trí gần hơn so với vị trí địa lý. Nếu mạng WAN bị ngừng hoạt động, ví dụ người dùng vẫn có thể đăng nhập. Việc tạo bản sao cần được lên kế hoạch và vận hành.

Lưu ý: Chúng ta lên kế hoạch cho điều kiện khẩn cấp – phải làm gì nếu csdl bảo mật bị ngắt hay nếu việc tạo bản sao bị hỏng. Đối với hệ thống bảo mật bị hỏng, chúng ta cũng nên có cả hai kế hoạch khẩn cấp và thủ tục tự động chú ý đến những vấn đề chung như mạng bị hỏng.

d. Kế hoạch kiểm soát và đăng nhập

Một hệ thống bảo mật tốt không là cơ chế thụ động. Thay vào đó, chứa chức năng trợ giúp kiểm soát hoạt động của hệ thống cho vấn đề bảo mật. Vấn đề chung của chức năng này là nhật ký. Toàn bộ thao tác của hệ thống có thể được ghi nhận hầu như toàn bộ sự kiện liên quan đến bảo mật hệ thống. Có thể ghi nhận mỗi khi đăng nhập, truy xuất đến mọi tài nguyên nhưng điều này hiếm khi hiệu qủa; thường chúng ta sẽ ghi nhận một số tập thông tin này như việc cố gắng đăng nhập lỗi.

Lưu ý: Nhật ký hệ thống tự nó thì không có ý nghĩa; chúng ta phải kế hoạch kiểm soát thường xuyên bởi ta có thể phát hiện những nghi ngờ những mẫu nhật ký hoạt động. Người kiểm soát được huấn luyện nên phân tích nhật ký trên cơ sở thường xuyên, đưa ra những đề nghị nếu có bất kỳ điều nghi ngờ.

e. Xác định mức độ yêu cầu bảo mật

Bảo mật cũng giống như những phần khác trong thiết kế ứng dụng, là sự cân nhắc giữa hiệu quả và chi phí. Nếu hệ thống không lưu những dữ liệu có tính nhạy cảm cao. Cách tốt nhất để triển khai hệ thống đó là "giữ sự xác thực của người dùng" đòi hỏi lưu trữ. Nếu chúng ta lưu trữ thông tin cần cho bảo mật, chi phí cho bảo mật thông tin đặc biệt phải được kiểm chứng.

Không có hệ thống nào bảo mật 100%. Chúng ta phải xác định mức độ rủi ro bảo mật có thể chấp nhận được. Độ rủi ro bảo mật diễn tả tỉ lệ phần trăm tương xứng khả năng mà bảo mật hệ thống không bao giờ đạt đến. Điều đó có thể nhưng phí tổn để xây dựng hệ thống bảo mật 99%. Chúng ta hay khách hàng phải xác định mức độ rủi ro có thể chấp nhận được dựa trên dữ liệu nhạy cảm của hệ thống.

f. Rà soát bảo mật hiện tại

Chúng ta nên trung thành ý tưởng của yêu cầu bảo mật của ứng dụng. Ở thời điểm phân tích chính sách bảo mật hiện tại của công ty để xác định bảo mật có đạt đến những nhu cầu của hệ thống hay không. Nếu không, thảo luận vấn đề với người gách vác hệ thống bảo mật ở công ty để tìm ra giải pháp mang lại lợi ích để triển khai mở rộng bảo mật.

1.1.4.Phân tích yêu cầu tốc độ

Tốc độ của ứng dụng có thể đòi hỏi khó. Đối với người dùng, ứng dụng sẽ hầu như chạy quá chậm nhưng chạy nhanh ứng dụng thiết kế tốt có thể mang lại giá trị

Lưu ý: việc chạy nhanh một ứng dụng thiết kế kém thì dễ, nhiều ứng dụng có thể chạy chậm bởi thiết kế thiếu sót, những không bởi không tương thích giữa phần ứng và các yếu tố bên ngoài.

Chúng ta nên nhận thức yêu cầu tốc độ ứng dụng trước khi bắt đầu qui trình thiết kế. Yêu cầu tốc độ dựa theo các mục sau:

Mỗi phút giao dịch: cung cấp dịch vụ phụ thuộc vào số lượng lớn người dùng, ứng dụng phân tán dùng những giao tác. Số giao tác mỗi phút (TPM) là độ đo tốc độ hệ thống cơ sở dữ liệu.

Băng thông: Ứng dụng phân tán làm nghẽn việc sử dụng mạng. Sự phản hồi của ứng dụng xác định định băng thông mạng (độ rộng của đường truyền mạng). Băng thông thường được đo bằng megabit mỗi giây.

Khả năng chứa: Lượng lưu trữ- cả chính và phụ - sẵn sàng đối với ứng dụng là vấn đề lưu tâm quan trọng cho tốc độ chung của ứng dụng. RAM đòi hỏi của ứng dụng gây ra những khác biệt lớn cho tốc độ của ứng dụng.

Nút thắt: Trong mỗi hệ thống, có phần giới hạn tốc độ hệ thống nói chung. Ví dụ CPU tốc độ nhanh cũng không cải thiện gì mấy nếu phải chờ dữ liệu từ một ổ cứng quá chậm. Trong trường hợp này, ổ cứng sẽ là nút thắt của toàn bộ hệ thống. Không thể tăng tốc độ trừ khi nút thắc được nhận biết, bởi vì chỉ có cải thiện nút thắt làm nâng tốc độ phù hợp. Chúng ta có thể nhận biết nút thắt bằng cách sử dụng công cụ báo cáo hệ thống như Màn hình điều khiển tốc độ trên Window NT (Windows NT Performance Monitor).

Thuật ngữ tốc độ thường dùng đồng nghĩa với sự phản hồi - số lượng thời gian chiếm giữ để phản hồi lại hành động của người dùng. Có thể làm cho ứng dụng xuất hiện phản hồi mà không cần tăng tốc độ. Tuy nhiên, thời gian phản hồi trung bình của ứng dụng là đặc tính quan trọng, chúng ta phải kết hợp chặt chẽ những mục tiêu thời gian phản hồi đối vời yêu cầu chung thiết kế.

Không thể nói về tốc độ trong những ứng dụng phân tán mà không phân biệt quan trọng: giữa nhu cầu cao và trung bình. Tại một số thời điểm - tối hay cuối tuần – có lẽ ứng dụng sẽ phục vụ với số lượng nhỏ người dùng, thì tốc độ nó sẽ trên trung bình. Ở thời điểm khác, số lượng người dùng sẽ cao hơn và tốc độ ứng dụng đủ cho phép. Mục tiêu tốc độ bao gồm cả mục tiêu tốc độ trung bình và cao.

1.1.5 Phân tích yêu cầu vận hành

Chúng ta có thể giảm bớt chi phí vận hành theo nhiều cách.Cách tốt nhất để giảm chi phí vận hành là đảm bảo chương trình được kiểm thử và chạy debug trước khi đưa vào triển khai. Chi phí triển khai có thể được giảm bớt bởi phân phối trực tuyến hay những thủ tục tự động cài đặt, và qui trình vận hành có thể tự động bằng các qui trình tin học. Mức vị trí và huấn luyện đội ngũ là vấn đề xem xét quan trọng: đội ngũ nhân viện càng được huấn luyện kỹ và sâu thì vấn đề càng nhanh chóng được sửa đổi.

Trong trường hợp phần cứng, phần mềm là thành phần được mua chứ không được phát triển, chúng ta có thể nhận sự chấp thuận vận hành từ nhà xưởng hay người ủy thác của sản phẩm. Vận hành sản phẩm trung gian tiết kiệm cho chúng ta chi phí thuê mướn nhân viên

mới hay huấn luyện lại những nhân viên cũ để duy trì một hay nhiều thành phần của hệ thống.

Giảm chi phí vận hành đòi hỏi sự tự thõa mãn lợi nhuận trong thời ngắn đối với những lợi ích trong tương lai. Giảm chi phí vận hành lâu dài thường đòi hỏi đầu tư đón đầu trong tự động hóa phần cứng và phần mềm.

1.1.6 Phân tích khả năng mở rộng yêu cầu

Qua thời gian, những yêu cầu giải pháp sẽ thay đổi. Người dùng cần những chức năng mới, các quy luật đặt ra sẽ bị sửa đổi, và phần cứng phần mềm nền mới thay đổi theo. Ứng dụng thiết kế tốt là có khả năng mở rộng được – nó có thể uyển chuyển cải thiện mà không phải viết lại hoàn toàn. Khả năng mở rộng của ứng dụng bị đảo ngược so với lượng công việc cần hoàn thành để thêm những đặc trưng mới.

Khả năng mở rộng có thể đạt được thông qua những ý nghĩa khác nhau. Một cách đạt những khả năng hạn định là lưu trữ thông tin quy luật đặt ra trong cơ sở dữ liệu hơn là lập trình chúng trong đối tượng nghiệp vụ. Theo cách đó, nếu số quan trong hay thủ tục thay đổi, nó có thể thay đổi trong CSDL mà không thay đổi mã nguồn. Cách khác là đặt mã nguồn vào trong đoạn script được làm rõ hơn biên dịch chương trình; đoạn script có thể bị thay đổi một cách dễ dàng không đòi hỏi bất kỳ biên dịch hay cài đặt lại tập tin nhị phân

Lưu ý: cách tốt nhất để đạt được khả năng mở rộng là ngắt ứng dụng thành những đối tượng thành phần, mỗi thành phần hoàn thành một nhiệm vụ riêng lẻ. Nếu những yêu cầu của những nhiệm vụ đặc biệt thay đổi, đối tượng tương ứng có thể bị thay đổi và biên dịch lại mà không gây ảnh hưởng bất kỳ đối tượng khác. Những đối tượng được thêm vào dễ dàng. Đối tượng nghiệp vụ có những thuận lợi được làm hiệu quả hơn những phương pháp khác trong khi vẫn đảm bảo tốt khả năng mở rộng.

1.1.7. Phân tích những yêu cầu sẵn có

Những ứng dụng phân tán được thiết kế để chạy mỗi ngày. Nó cần thiết cho sự thành công của doanh nghiệp. Như vậy, chúng có mức độ sẵn sàng cao nên tránh thường bảo trì, sửa chữa, phát sinh không theo kế hoạch.

Rõ ràng, đối với những ứng dụng mang tính sẵn sàng, nó không được gây ra lỗi. Không có ứng dụng nào là không có lỗi, ứng dụng phải được bảo lưu để chúng có thể hoạt động thậm chí khi bug xảy ra trong một phần của chương trình. Thí dụ, nếu người dùng gây ra lỗi cho chương trình thì chỉ một phần chương trình phục vụ cho người dùng đó bị hỏng, không

ảnh hưởng người dùng còn lại đang nối kết. Bất kỳ thành phần ứng dụng nào hỏng hay không sẵn sàng thì nên khởi động lại ngay khi có thể.

Việc bảo trì có kế hoạch cũng tác động đến tính sẵn sàng. Một máy chủ chứa ứng dụng lý tưởng luôn có bản sao lưu có thể khởi động khi máy chủ bảo trì. ứng dụng có mức độ sẵn sàng cao có cách luân phiên để kết nối mạng trong trường hợp mạng WAN, LAN ngưng hoạt động

Lưu ý: Tính sẵn sàng liên quan đến nghiệp vụ. Tính sẵn sàng của ứng dụng càng cao, giá trị của ứng dụng càng cao. Chúng ta phải xác định bao nhiêu giờ trong ngày ứng dụng cần được thao tác; giờ nào là quan trọng so với các giờ trong ngày. Cân nhắc giá trị của việc tăng tính sẵn sàng đối với giá trị dự án của thời gian down ứng dụng. Những hệ thống trọng yếu, giá trị đối với công ty ở bất kỳ thời điểm down nào hoàn toàn điều chỉnh chi phí thiết kế 100 % ứng dụng sẵn sàng. Ứng dụng khác đơn giản cần trở nên sẵn sàng hầu hết mọi lúc.

1.1.8. Phân tích yêu tố con người

Thiết kế ứng dụng được giám sát bởi nhiều người lập trình là phần quan trọng của yếu tố con người. Chúng ta nên xác định kinh nghiệm gì mà chúng ta muốn người dùng có. Với bất cứ ứng dụng nào khác, kinh nghiệm người dùng càng tốt thì chi phí càng cao.

Bắt đầu định nghĩa mục tiêu của người dùng. Xác định người dùng với những nhu cầu đặc biệt như thế nào. Chúng ta cần điều tiết người dùng qua việc điều tiết nghe và nhìn, hay người dùng nói tiếng nước ngoài. Phụ thuộc vào vị trí địa lý của người sử dụng. Chúng ta cần sửa đổi ứng dụng thích ứng theo vị trí địa lý. Cần điều chỉnh nhu cầu lướt qua của người dùng, người không cần sự nối kết chắc chắn hay khả năng trả lời lại.

Xem xét mức độ chuyên nghiệp giữa người dùng. Với chuyên viên học nhanh hơn với giao diện thiết kế tốt và trợ giúp trực tuyến Help online. Người dùng với kỹ năng kém hơn để tăng tốc qua sử dụng wizard, trợ giúp online, hay chỉ dẫn. Huấn luyện khách hàng trong ứng dụng cũng nên cân nhắc chọn lựa.

1.1.9. Phân tích yêu cầu tích hợp

Nếu giải pháp giao tiếp với ứng dụng kế thừa, việc truy xuất CSDL tồn tại, hay việc chuyển đổi dữ liệu cũ sang khuôn dạng mới, bạn cần phải đưa kế hoạch tích hợp ứng dụng với phần mềm cũ. Điều này được làm thông qua kết nối của hãng trung gian như trình điều khiển thiết bị kết nối csdl (ODBC), nhưng chúng ta cũng cần viết kết nối và những tiện ích chuyển đổi

Khi phát sinh nhu cầu lớn hơn, cơ sở dữ liệu phải thiết kế lại. Kỹ thuật dữ liệu mới hay vậ lý đưa nhu cầu cải thiện CSDL bên dưới ứng dụng. Những cải tiến phải được cẩn thận bởi chúng phá vở tất cả mã nguồn CSDL hiện tại. Trước khi cải tiến khung dữ liệu, đảm bảo những phần mã nguồn hiện tại có thể truy xuất đến CSDL. Tất cả mã nguồn hiện tại phải được soát lại, có thể viết lại.

1.1.10. Phân tích thực tiễn nghiệp vụ tồn tại

Phần định nghĩa trong qui tắc nghiệp vụ liên quan đến sự hiểu biết ngữ cảnh trong những qui tắc thao tác. Hiểu được những thực tế nghiệp vụ của doanh nghiệp có thể giúp chúng ta tránh được sai sót thậm chí giúp tìm cách tốt hơn, hiệu quả hơn của tự động hóa tiến trình nghiệp vụ. Hiểu được vấn đề hợp lệ dưới mỗi tiến trình có thể ngăn bạn gây ra lỗi một cách ngây ngô dẫn đến tranh chấp.

Hiểu được cấu trúc tổ chức và sơ đồ làm việc nghiệp vụ là quyết định. Không hiểu rõ ràng sơ đồ tổ chức, không thể đem lại sự chấp thuận phù hợp cho thiết kế ứng dụng của chúng ta hay thông tin theo kíp trên thiết kế hay những vấn đề triển khai. Đồ hình tổ chức cũng giúp cho tìm kiếm thông tin người ẩn danh phản hồi lại chức năng của ứng dụng mà không dùng bất của chính họ.

Có được ứng dụng từ giai đoạn phát triển đến sản phẩm đòi hỏi sự hiểu biết mạng và chính sách hạ tầng của công ty. Biết được ai là người chịu trách nhiệm bảo trì, bảo mật, tính toàn vẹn, khả năng phản hồi tương tác trên mạng. Học những tiến trình và chính sách liên quan chạy trên ứng dụng mới. Tìm ra loại kiểm soát chất lượng và dịch vụ kiểm thử sẵn sàng trong khi chúng ta kiểm thử trên chính phần mềm, ta có thể tự động tài nguyên hay dành cho bộ phận kiểm tra chất lượng tùy ý sử dụng. Chúng ta có thể yêu cầu phương pháp thiết kế đặc biệt hay triển khai thực tế. Chúng at cũng đòi hỏi chắc chắn kế hoạch được kết chặt với ngân sách

Cuối cùng, giữ những nguyên tắc cốt lỏi: Học nhu cầu khách hàng, cố gắng thực hiện chúng. Điều này có thể trở nên khó khi khách hàng không biết nhu cầu của họ là gì, nhưng đó là cách dẫn đến ứng dụng thành công.

1.1.11.Phân tích yêu cầu khả năng quy mô

Nếu ứng dụng thành công sẽ hấp dẫn người dùng hơn. Đặc biệt, nếu ứng dụng chạy trên môi trường web như Internet thì sự thành công đồng nghĩa với tăng nhu cầu. Ứng dụng phải được thiết kế có quy mô- nó phải hỗ trợ nâng cấp cho phép phục vụ nhiều người hơn.

Một cách đơn giản để nâng cao ứng dụng là mua CPU nhanh hơn, nhiều RAM, kết nối mạng tốt hơn. Tuy nhiên việc tăng cường máy đơn chạy nhanh hơn. Thực sự những ứng dụng có thể nâng cấp phải thêm vào nhiều dịch vụ phía máy chủ. Điều này có nghĩa ứng dụng có thể chạy trên nhiều máy tính cùng một lúc, sự phân phối việc tải xuống của người dùng và xử lý thời gian qua nhiều máy chủ. Điều này sẽ gia tăng đáng kể tính phức tạp, vì vậy một lần nữa tính thuận tiện khả năng quy mô phải được cân nhắc đối với giá trị cung cấp. Tuy nhiên, ứng dụng như Miscrosoft Transaction Server giảm đáng kể chi phí phát triển ứng dụng phân tán bởi quản lý về mặt logic của phân tán tự động.

1.2 Xác định yêu cầu

Mục tiêu của việc xác định yêu cầu:

Xác định thật chính xác và đầy đủ các yêu cầu đặt ra cho phần mềm sẽ được xây dựng.

Kết quả nhận được sau giai đoạn xác định yêu cầu:

1. *Danh sách các công việc sẽ được thực hiện trên máy tính*
2. *Những mô tả chi tiết về các công việc này khi được thực hiện trong thế giới thực.*

Qua đó bước đầu hình thành thông tin khái quát về các hoạt động trong thế giới thực.

1.2.1 Yêu cầu và mô tả yêu cầu

③ Yêu cầu (hay yêu cầu phần mềm) là công việc muốn thực hiện *trên máy tính*. Những công việc này phải xuất phát từ thực tế chứ không thuần túy tin học

③ Mô tả yêu cầu là mô tả đầy đủ các thông tin liên quan đến công việc tương ứng. Các mô tả này dùng làm cơ sở để nghiệm thu và đánh giá phần mềm khi được chuyển giao.

Các yêu cầu của phần mềm cần được mô tả thật rõ ràng, cụ thể, đầy đủ và chính xác các thông tin liên quan đến công việc tương ứng. Việc mô tả sơ sài, mơ hồ yêu cầu phần mềm sẽ dẫn đến việc hiểu nhầm giữa chuyên viên tin học (người thực hiện phần mềm) và khách hàng (người đặt hàng thực hiện phần mềm). Nhiều công sức và chi phí phải hao tốn do các hiểu nhầm như thế.

Các loại thông tin chính cần được quan tâm khi xác định yêu cầu phần mềm:

③ Tên công việc ứng với từng yêu cầu
③ Người hoặc bộ phận sẽ thực hiện công việc
③ Địa điểm thực hiện công việc

③ Thời gian thực hiện công việc

③ Cách thức tiến hành công việc cùng với các quy định liên quan

Sau đây, từng loại thông tin sẽ lần lượt được xem xét chi tiết:

a. ***Tên công việc.***

Cần xác định cụ thể, tránh dùng các tên chung chung, mơ hồ

Ví dụ: xét một số tên công việc sau:

Quản lý độc giả: chung chung, mơ hồ; cụ thể như việc đăng ký mượn sách, gia hạn thẻ độc giả, trả sách

Quản lý sách: chung chung, mơ hồ; cụ thể như nhập sách vào kho, tra cứu sách, cho mượn sách, nhận trả sách, thanh lý sách.

b. ***Người thực hiện.***

Cần xác định chính xác người hoặc bộ phận sẽ thực hiện công việc trên máy tính (còn gọi là người dùng phần mềm hay người dùng).

Những người dùng có vai trò và công việc thực hiện tương tự như nhau sẽ được xếp vào cùng một loại người dùng (thông thường một loại người dùng sẽ tương ứng với một bộ phận trong thế giới thực).

Cùng một công việc có thể có nhiều loại người dùng khác nhau thực hiện và ngược lại, một loại người dùng có thể thực hiện nhiều công việc khác nhau.

c. ***Thời gian, địa điểm.***

Cần xác định chính xác địa điểm, thời điểm tiến hành công việc. Các thông tin này sẽ có ý nghĩa nhất định trong một số trường hợp đặc thù.

d. ***Cách thức tiến hành và các quy định liên quan.***

Đây là phần chính yếu khi tiến hành mô tả yêu cầu. Đối với loại thông tin này cần đặc biệt quan tâm đến một số yếu tố sau:

i. Các quy định cần kiểm tra khi thực hiện công việc ghi nhận thông tin

Ví dụ: Quy định về việc mượn sách khi cho độc giả mượn sách: chỉ cho mượn sách đối với những độc giả có thẻ độc giả còn hạn, số sách đang mượn chưa đến 2 và không có sách mượn quá hạn.

Ví dụ: Quy định tính hợp lệ của phân số trong việc ghi nhận đề bài của giáo viên và bài giải của học sinh: phân số phải có mẫu số khác 0

ii. Các quy định, công thức tính toán khi thực hiện công việc tính toán

Ví dụ: Quy định tính tiền phạt trả sách trễ khi thực hiện việc trả sách: mỗi ngày trả trễ phạt 1500 đồng/ngày. Từ ngày trả trễ thứ 10 trở đi sẽ phạt 5000 đồng/ngày và thu hồi thẻ độc giả 2 tuần.

Ví dụ: Quy định tiền lương khi thực hiện công việc tính lương nhân viên cho 1 công ty

* Lương của nhân viên thuộc bộ phận văn phòng được tính theo công thức:

Tiền_Lương = (Số_Ngày * Mức_Lương)/22 + Tiền_Thưởng

+ Tiền_Phạt

mỗi ngày làm thêm thưởng 30.000

mỗi ngày nghỉ việc phạt 50.000

* Lương của nhân viên thuộc bộ phận sản xuất được tính theo công thức:

Tiền_Lương = Số_Sản_Phẩm * Đơn_Giá

Biết rằng một sản phẩm phải trải qua 3 công đoạn sản xuất:

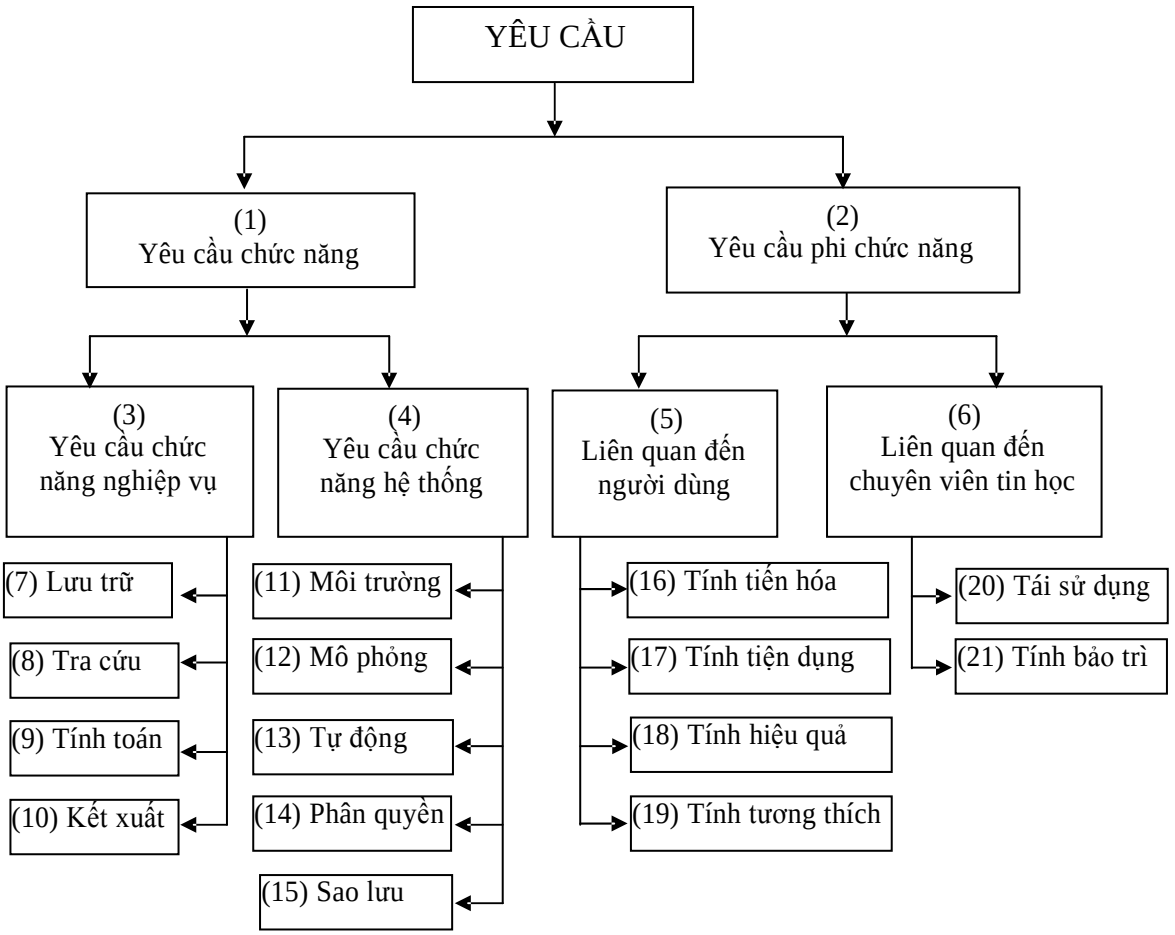
công đoạn 1: 200 đồng/sản phẩm

công đoạn 2: 400 đồng/sản phẩm

công đoạn 3: 300 đồng/sản phẩm

1.2.2 Phân loại yêu cầu

Sơ đồ cây phân loại yêu cầu


YÊU CẦU
(1)
Yêu cầu chức năng
(2)
Yêu cầu phi chức năng
(3)
Yêu cầu chức năng nghiệp vụ
(4)
Yêu cầu chức năng hệ thống
(5)
Liên quan đến người dùng
(6)
Liên quan đến chuyên viên tin học
(7) Lưu trữ
(8) Tra cứu
(9) Tính toán
(10) Kết xuất
(11) Môi trường
(12) Mô phỏng
(13) Tự động
(14) Phân quyền
(15) Sao lưu
(16) Tính tiến hóa
(17) Tính tiện dụng
(18) Tính hiệu quả
(19) Tính tương thích
(20) Tái sử dụng
(21) Tính bảo trì

Đặc tả chi tiết từng loại yêu cầu:

(1) Yêu cầu chức năng là danh sách các công việc sẽ được *thực hiện trên máy tính* cùng với các thông tin mô tả tương ứng.

(2) Yêu cầu phi chức năng là các yêu cầu liên quan đến chất lượng phần mềm, là sự ràng buộc cách thức thực hiện các yêu cầu chức năng.

(3) Yêu cầu chức năng nghiệp vụ là các chức năng của phần mềm tương ứng với công việc có thật trong thế giới thực.

(4) Yêu cầu chức năng hệ thống là các chức năng phần mềm *được phát sinh thêm* khi thực hiện công việc trên máy tính thay vì trong thế giới thực hoặc các chức năng không tương ứng với bất kỳ công việc nào trong thế giới thực.

(7) Chức năng lưu trữ: Tương ứng với công việc ghi chép thông tin trên sổ sách (kèm theo các quy định khi ghi chép).

Ví dụ:

- Ghi nhận việc cho mượn sách của một thư viện theo quy định mượn.

- Ghi nhận bài giải bài tập về phân số theo quy định về phân số,cách biến đổi phân số tương đương, các phép tính trên phân số,…

(8) Chức năng tra cứu: Tương ứng với công việc tìm kiếm, theo dõi hoạt động và xem thông tin về một đối tượng.

Ví dụ:

- Tìm tài khoản và xem tình hình gửi rút.

- Tìm sách và xem tình trạng sách

- Tìm hàng hóa và xem tình trạng của hàng hóa (số lượng tồn kho, lượng nhập, thời gian nhập,).

- Tìm bài giảng lý thuyết về phương trình, bất phương trình và xem nội dung tương ứng.

(9) Chức năng tính toán: Tương ứng với công việc tính toán (theo quy định và công thức cho trước).

Ví dụ:

- Tính điểm trung bình môn học của học sinh theo quy định hệ số cho các đợt kiểm tra.

- Xếp thứ hạng cho các đội bóng sau một lượt thi đấu theo quy định của ban tổ chức giải.

- Tính tiền phạt trả sách trễ theo quy định phạt của thư viện.

- Tìm nghiệm của phương trình bậc hai theo phương pháp giải phương trình bậc hai.

(10) Chức năng kết xuất : Tương ứng với công việc lập báo cáo (theo biểu mẫu cho trước)

Ví dụ:

- Lập bảng xếp hạng các đội bóng sau một lượt đấu.

- Lập báo cáo thống kê về số lượt mượn sách theo từng thể loại trong năm.

- Lập báo cáo thống kê về tỷ lệ xếp loại học sinh theo từng lớp, từng khối.

(11) Chức năng môi trường : Định cấu hình thiết bị, ngày giờ, số người làm việc, …

Ví dụ: Số lượng người làm việc, chọn loại máy in, khổ giấy, niên khóa hiện hành, …

(12) Chức năng mô phỏng: Mô phỏng hoạt động của thế giới thực

Ví dụ: - Mô phỏng một tai nạn máy bay, xe ô tô, trận động đất

(13) Chức năng tự động: Tự động thông báo, nhắc nhở người dùng.

Ví dụ:

- Nhắc nhở thủ thư gửi giấy báo đòi sách khi có độc giả mượn quá hạn.

- Báo động khi khách hàng thiếu nợ quá lâu hay số tiền nợ quá lớn.

(14) Chức năng phân quyền : Phân quyền sử dụng giữa các loại người dùng.

Ví dụ: Phân quyền cho 3 loại người sử dụng trong phần mềm quản lý thư viện:

+ Quản trị hệ thống: có quyền sử dụng tất cả các chức năng.

+ Thủ thư: chỉ sử dụng các chức năng liên quan đến việc cho mượn và trả sách.

+ Độc giả: chỉ sử dụng chức năng tra cứu.

Trong phần mềm quản lý bán hàng, việc phân chia khả năng truy cập dữ liệu nhập xuất cho từng nhóm người sử dụng sẽ tránh việc điều chỉnh số liệu không thuộc phạm vi quản lý của người sử dụng như nhân viên thu ngân chỉ được phép lập và điều chỉnh các hóa đơn bán hàng trong ca làm việc của mình. Ca trưởng và bộ phận quản lý quầy có thể tham khảo lượng hàng

tồn kho nhưng không được phép điều chỉnh lượng hàng nhập, không được tham khảo vốn hàng xuất, kết quả kinh doanh, …

(15) Chức năng sao lưu : Sao lưu, phục hồi dữ liệu.

Ví dụ: Sao lưu thông tin về các học sinh đã ra trường và chỉ phục hồi lại khi cần thiết

(16) Tính tiến hóa: đây là các yêu cầu liên quan đến việc cho phép người dùng thay đổi lại cách mô tả của một yêu cầu chức năng (các quy định, quy tắc tính toán), một biểu mẫu nào đó khi đang dùng phần mềm đã được chuyển giao. Điều này đòi hỏi phải có dự kiến về các thay đổi trên thành phần dữ liệu và xử lý.

Ví dụ:

- Cho phép thay đổi quy định về số sách cho mượn tối đa, hay mức phạt khi trả trễ.
- Cho phép thay đổi các biên trong quy định về xếp loại học sinh.

(17) Tính tiện dụng: là các yêu cầu liên quan đến hình thức giao diện của phần mềm, thể hiện ở sự tự nhiên, dễ sử dụng, dễ học, đầy đủ thông tin,...

Ví dụ:

- Giao diện nhập hóa đơn bán hàng dạng form, dòng nhập thể hiện bằng ô sáng và báo lỗi khi số liệu nhập làm số lượng tồn kho âm (phần mềm quản lý hàng hóa).

(18) Tính hiệu quả : đây là yêu cầu liên quan đến thời gian thực hiện các chức năng phần mềm, dung lượng lưu trữ, chi phí sử dụng tài nguyên hệ thống như sử dụng tối ưu các không gian, thao tác thực hiện nhanh ...

Ví dụ: Thời gian tra cứu sách, tra cứu độc giả không quá 10 giây.

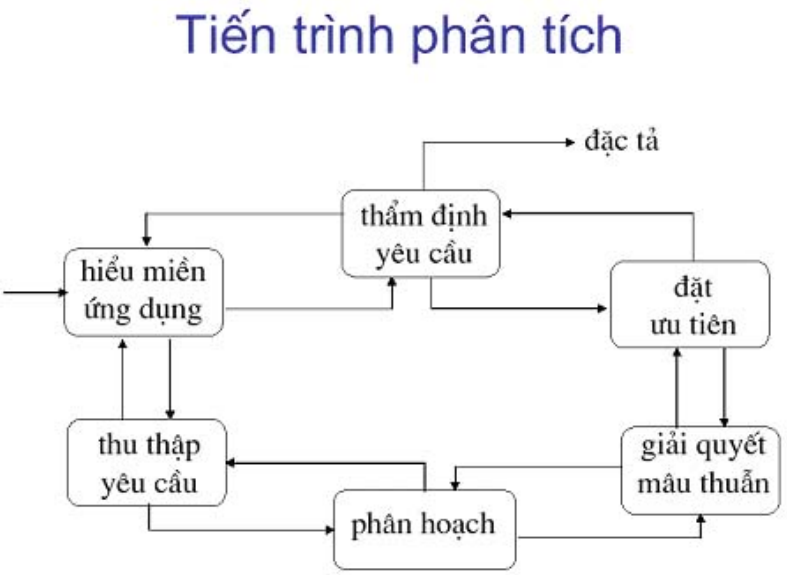
(19) Tính tương thích: là các yêu cầu liên quan đến việc chuyển đổi dữ liệu giữa phần mềm đang xét và các phần mềm khác, sự nhất quán giữa các màn hình trong hệ thống.

Ví dụ: - Cho phép chuyển tất cả các báo cáo sang định dạng file Excel

- Cho phép nhập thông tin sách mới từ tập tin Excel hay từ thiết bị đọc mã vạch.
- Cho phép thực hiện chức năng bằng giọng nói.

(20) Tính tái sử dụng: (do chuyên viên tin học đảm trách)

(21) Tính bảo trì: (do chuyên viên tin học đảm trách) là các yêu cầu cho phép thay đổi mà không làm ảnh hưởng đến phần mềm



1.2.3 Các bước xác định yêu cầu

Quá trình thực hiện xác định yêu cầu: gồm 2 bước chính như sau

Bước 1: Khảo sát hiện trạng, kết quả nhận được là các báo cáo về hiện trạng.

Bước 2: Lập danh sách các yêu cầu, kết quả nhận được là danh sách các yêu cầu sẽ được thực hiện trên máy tính.

Đối tượng tham gia xác định yêu cầu: gồm 2 nhóm người:

- *Chuyên viên tin học*: những người hiểu rõ về khả năng của máy tính. Họ phải tìm hiểu thật chi tiết về công việc của nhà chuyên môn nhằm tránh sự hiểu nhầm cho những bước phân tích sau này.
- *Nhà chuyên môn*: những người hiểu rõ về công việc của mình. Họ cần lắng nghe ý kiến của các chuyên viên tin học để đảm bảo các yêu cầu của họ là có thể thực hiện được với chi phí và thời gian hợp lý.

Hai nhóm người này cần phải phối hợp thật chặt chẽ để có thể xác định đầy đủ và chính xác các yêu cầu.

Sau đây, chúng ta sẽ phân tích chi tiết từng bước quy trình thực hiện.

1.2.3.1 Khảo sát hiện trạng

Các chuyên viên tin học sẽ tìm hiểu hiện trạng về các công việc của các nhà chuyên môn.

a. Các hình thức thực hiện phổ biến:

- Quan sát: theo dõi các hoạt động đang diễn ra ở thế giới thực có liên quan, có thể tiến hành ghi âm, ghi hình đối với những tình huống mang tính phức tạp, quan trọng, cần sự chính xác cao.

  Ví dụ:
  - Ghi hình quá trình giao dịch của một nhân viên ngân hàng với khách hàng tại một ngân hàng X.
  - Quan sát thao tác cho mượn sách của một thủ thư tại một thư viện Y

- Phỏng vấn trực tiếp: tổ chức phỏng vấn bắt đầu từ cấp lãnh đạo dần xuống các vị trí công việc. Có thể sử dụng các bảng câu hỏi có sẵn các câu trả lời cho đối tượng được phỏng vấn lựa chọn, …

- Thu thập thông tin, tài liệu: các công thức tính toán, quy định; các bảng biểu, mẫu giấy tờ có ít nhiều liên quan.

  Ví dụ:

  - Mẫu hóa đơn và các quy định lập hóa đơn bán hàng tại một cửa hàng Y.

  - Phiếu mượn sách tại thư viện của trường đại học Z.

b.Quy trình thực hiện:

- Tìm hiểu tổng quan về thế giới thực: bao gồm

- Quy mô hoạt động.
- Các hoạt động mà đơn vị có tham gia.

③ Tìm hiểu hiện trạng tổ chức (cơ cấu tổ chức)

Người tiến hành khảo sát hiện trạng cần hiểu rõ cơ cấu tổ chức các bộ phận của thế giới thực, đặc biệt là 2 yếu tố: trách nhiệm và quyền hạn. Sự hiểu rõ cơ cấu tổ chức giúp xác định bộ phận nào sẽ sử dụng phần mềm để có thể lên kế hoạch tiếp tục khảo sát chi tiết hơn bộ phận đó.

Cơ cấu tổ chức bao gồm:

- Đối nội.
- Đối ngoại.
- Các chức danh (Ví dụ: nhân viên nhập liệu, thủ thư, nhân viên bán hàng, …).

Sử dụng các đồ hình để vẽ lại cơ cấu tổ chức.

③ Tìm hiểu hiện trạng nghiệp vụ

Thường diễn ra tại các vị trí công việc. Với bộ phận được chọn khảo sát chi tiết, người thực hiện khảo sát cần lập danh sách các công việc mà bộ phận này phụ trách, sau đó tìm hiểu các thông tin chi tiết cho từng công việc (thông tin mô tả yêu cầu phần mềm).

Việc tìm hiểu dựa trên các ý sau:

- Thông tin đầu vào.
- Quá trình xử lý.
- Thông tin kết xuất.

Sau đó tiến hành xếp loại các nghiệp vụ vào 4 loại sau nhằm tránh thiếu xót khi tìm hiểu các thông tin:

- Nghiệp vụ lưu trữ.
- Nghiệp vụ tra cứu.
- Nghiệp vụ tính toán.
- Nghiệp vụ tổng hợp, thống kê

1.2.3.2 Lập danh sách các yêu cầu

Để có được danh sách đầy đủ và chính xác các, quá trình lập danh sách các yêu cầu cầu theo các bước sau:

③ Xác định yêu cầu chức năng nghiệp vụ

③ Xác định yêu cầu chức năng hệ thống

③ Xác định yêu cầu phi chức năng

a. Xác định yêu cầu chức năng nghiệp vụ.

Cách tiến hành: Nhà chuyên môn đề xuất và chuyên viên tin học sẽ xem xét lại

Bước tiến hành :

1. Xác định bộ phận (người dùng) sẽ sử dụng phần mềm
2. Xác định các công việc mà người dùng sẽ thực hiện trên phần mềm theo từng loại công việc sau:
   - Lưu trữ
   - Tra cứu
   - Tính toán
   - Kết xuất

Lần lượt lập bảng yêu cầu chức năng nghiệp vụ, bảng quy định/Công thức và các biểu mẫu – được mô tả chi tiết – như sau:

*Mẫu 1: Bảng yêu cầu chức năng nghiệp vụ

Bộ phận (người thực hiện): … Mã số: ...

stt	Công việc	Loại công việc	Quy định/ Công thức liên quan	Biểu mẫu liên quan	Ghi chú
1					
2					

* Mẫu 2: Bảng Quy định/ Công thức liên quan

stt	Mã số	Tên Quy định/ Công thức	Mô tả chi tiết	Ghi chú
1	QĐ 1			

2	QĐ 2			

Các biểu mẫu được mô tả chi tiết ngay sau bảng quy định/Công thức

Ví dụ: Xét phần mềm quản lý thư viện

Bộ phận: Thủ thư. **Mã số: TT**

stt	Công việc	Loại công việc	Quy định/Công thức liên quan	Biểu mẫu liên quan	Ghi chú
1	Cho mượn sách	Lưu trữ	TT_QĐ 1	TT_BM 1	
2	Nhận trả sách	Lưu trữ	Chỉ nhận lại những sách đã cho mượn	TT_BM 1	
3	Tính tiền phạt	Tính toán	Mỗi ngày trả trễ phạt : - 1000 đồng/ngày : từ ngày thứ nhất đến ngày thứ 5 - 3000 đồng/ngày : từ ngày thứ 6 trở đi.		
4	Tính tiền đền	Tính toán	Tiền đền cho sách bị mất dựa trên giá thị trường tại thời điểm hiện hành.		
5.	Tra cứu sách	Tra cứu	Việc tìm sách dựa trên các thông tin : tên sách, tên tác giả, nhà xuất bản, năm xuất bản		
6.	Gửi giấy báo đòi sách	Kết xuất	Sách mượn quá hạn 3 ngày sẽ tự động gửi giấy báo cho đến khi sách được trả hoặc đã tính xong tiền đền sách	TT_BM 2	

Bảng yêu cầu chức năng nghiệp vụ

stt	Mã số	Tên Quy định/	Mô tả chi tiết	Ghi chú

		Công thức		
1	QĐ 1	Quy định cho mượn sách	Chỉ cho mượn sách khi : - Thẻ độc giả còn hạn - Độc giả chưa mượn hết số sách quy định - Độc giả không có sách mượn quá hạn - Sách hiện không có người mượn	Độc giả mượn sách sẽ phải gửi lại thẻ độc giả tại bộ phận bạn đọc, nhận phiếu mượn sách (TT_BM 1, tìm kiếm mã số sách mượn và điền các sách cần mượn vào phiếu, xong gửi cho thủ thư.

Bảng Quy định/ Công thức liên quan

TT_BM 1:

PHIẾU MƯỢN SÁCH

Số thẻ: Số phiếu mượn:

Họ và tên: Ngày mượn:

[ ] Mượn về nhà [ ] Đọc tại chỗ

STT	Mã sách	Tên sách	Tác giả	Mã loại
1				
2				

Ngày ... tháng ... năm ...

TT_BM 2:

GIẤY BÁO MƯỢN SÁCH QUÁ HẠN

Thân gửi: _____

Địa chỉ: _____

Chúng tôi xin thông báo rằng, anh (chị) đã mượn của thư viện chúng tôi những quyển sách sau:

STT	Mã sách	Tên sách	Ngày mượn	Đến hôm nay đã quá hạn
1				
2				

Vậy thông báo anh(chị) vui lòng đem sách đến trả. Và mang theo số tiền _____ đồng để trả phí sách trễ.

Bộ phận: Độc giả. **Mã số: ĐG**

STT	Công việc	Loại công việc	Quy định/ Công thức liên quan	Biểu mẫu liên quan	Ghi chú
1	Tìm sách	Tra cứu	Việc tìm sách dựa trên các thông tin: tên sách, tên tác giả, nhà xuất bản, năm xuất bản		
2	Đăng ký mượn sách	Lưu trữ	Độc giả phải có thẻ độc giả.	TT_BM 1	Mọi độc giả có thẻ mượn sách đều có thể đăng ký mượn sách. Tuy nhiên, hệ thống sẽ thông báo khi thẻ mượn

					sách của độc giả đã hết hạn sử dụng.

Bộ phận: Quản lý độc giả. **Mã số : QLĐG**

STT	Công việc	Loại việc	Quy định/ Công thức liên quan	Biểu mẫu liên quan	Ghi chú
1	Làm thẻ độc giả mới	Lưu trữ	Chỉ cấp thẻ độc giả có độ tuổi từ 18 trở lên và có chứng minh thư. Lệ phí làm thẻ độc giả là 5000 đồng/thẻ. Một số chứng minh thư chỉ có thể có duy nhất một thẻ độc giả	QLDGBM1 QLDGBM2	Độc giả có yêu cầu làm thẻ mượn sách sẽ được nhận phiếu đăng ký để điền thông tin vào (QLDG_BM 1), sau đó bộ phận quản lý độc giả tiến hành cấp thẻ và thu lệ phí theo quy định (QLDG_BM 2)
2	Gia hạn thẻ độc giả	Lưu trữ	Gia hạn thẻ theo yêu cầu của độc giả và thời gian quá hạn không được quá 3 tháng. Sau thời gian 3 tháng, những thẻ hết hạn sẽ bị hủy.		
3	Huỷ thẻ độc giả	Lưu trữ	Hủy bỏ các thẻ độc giả đã quá hạn đăng ký 3 tháng.		

QLDG_BM 1:

PHIẾU ĐĂNG KÝ LÀM THẺ MƯỢN SÁCH

Họ và tên: _____ Năm sinh: _____

Địa chỉ thường trú: _____

Nghề nghiệp: _____

Ngày đăng ký: _____

QLDG_BM 2:

THẺ ĐỘC GIẢ

Họ và tên: _____

Trường: _____ Lớp: _____

Địa chỉ: _____

Ngày ___ tháng ___ năm __

Bộ phận: Quản lý sách. **Mã số: QLS**

STT	Công việc	Loại	Quy định/ Công thức liên quan	Biểu mẫu liên quan	Ghi chú
1.	Nhận sách mới vào kho sách.	Lưu trữ		QLSBM 1	Khi có sách mới nhập về, bộ phận quản lý sách có trách nhiệm rà xét xem số sách đó đã có hay chưa, nếu chưa thì lập thẻ quản lý sách và định mã số sách mới. Nếu có rồi thì gọi lại thẻ cũ để cập nhật bổ sung số lượng.
2.	Thanh lý sách cũ	Lưu trữ	Các sách hư, không đọc được		
3.	Lập báo cáo các sách cần	Kết xuất		QLS_BM 2	

	thanh lý				
4.	Lập báo cáo sách mượn	Kết xuất		QLS_BM 3	

QLS_BM 1:

THẺ QUẢN LÝ SÁCH

Tên sách: _____

Tập: _____ Số trang: _____

Số lượng: _____Năm xuất bản: _____

Mã ngôn ngữ: _____Ngôn ngữ: _____

Mã nhà xuất bản: _____ Nhà xuất bản: _____

Mã phân loại: _____ Phân loại: _____

Mã tác giả: _____Tác giả: _____

Mã vị trí: ____ Khu: ___ Kệ: ___ Ngăn: ___

QLS_BM 2:

DANH SÁCH CÁC SÁCH CẦN THANH LÝ

stt	Mã sách	Tên sách	Tác giả	Năm sản xuất	Ngày nhập kho	Tình trạng
1						
2						

Ngày lập báo cáo:

Người lập:

OLS_BM 3:

BÁO CÁO THỐNG KÊ SÁCH MƯỢN

Từ ngày _____ Đến ngày _____

stt	Mã sách	Tên sách	Tác giả	Số lượt mượn
1.				
2.				

Ngày lập báo cáo:

Người lập:

b. Xác định yêu cầu chức năng hệ thống và yêu cầu chất lượng

** Cách tiến hành:*

Chuyên viên tin học và nhà chuyên môn cùng đề xuất và cùng xem xét lại các yêu cầu.

** Bước tiến hành:*

Bước 1: Xem xét các yêu cầu chức năng hệ thống cơ bản, thông dụng (yêu cầu phát sinh thêm do thực hiện các công việc trên máy tính): phân quyền, sao lưu, phục hồi, định cấu hình hệ thống, …

Bước 2: Xem xét các yêu cầu chức năng hệ thống chuyên biệt (yêu cầu về các công việc mới, chỉ có thể tiến hành khi thực hiện trên máy tính.

Bước 3: Xem xét các yêu cầu về chất lượng theo từng loại tiêu chuẩn sau:

- Tiến hóa
- Tiện dụng
- Hiệu quả
- Tương thích

Sau đó lập bảng yêu cầu tương ứng theo mẫu sau:

STT	Nội dung	Mô tả chi tiết	Ghi chú
1.			

Mẫu 3: Bảng yêu cầu chức năng hệ thống.

STT	**Nội dung**	**Tiêu chuẩn**	**Mô tả chi tiết**	**Ghi chú**
1.				
2.				

Mẫu 4: Bảng yêu cầu về chất lượng.

Ví dụ: Xét phần mềm quản lý thư viện (giả sử phần mềm được xây dựng nhằm phục vụ cho 4 bộ phận là: độc giả, thủ thư, ban giám đốc và quản trị hệ thống).

Bảng yêu cầu chức năng hệ thống:

stt	**Nội dung**	**Mô tả chi tiết**	**Ghi chú**
1	Phân quyền sử dụng	- Người quản trị: được phép sử dụng tất cả các chức năng - Độc giả: chỉ tra cứu sách và đăng ký mượn sách - Ban giám đốc: chỉ tra cứu sách và lập các báo cáo thống kê - Thủ thư: tất cả các chức năng, ngoại trừ chức năng phân quyền, sao lưu và phục hồi dữ liệu	

Bảng yêu cầu về chất lượng hệ thống:

stt	Nội dung	Tiêu chuẩn	Mô tả chi tiết	Ghi chú
1	Cho phép thay đổi quy định tính tiền phạt	Tiến hóa	Người dùng phần mềm có thể thay đổi đơn giá phạt và biên các mức phạt.	
2	Hình thức tra cứu thật tiện dụng, tự nhiên, trực quan. Dễ sử dụng cho cả những người không chuyên tin học.	Tiện dụng	Hỗ trợ khả năng tra cứu gần đúng, tra cứu theo nội dung,...	
3	Cho phép nhập sách mới từ tập tin Excel có sẵn. Các màn hình có sự nhất quán chung	Tương thích	Có thể nhập trực tiếp danh sách các sách mới có trước trên tập tin Excel với cấu trúc hợp lý.	
4	Tốc độ thực hiện việc cho mượn và tra cứu sách nhanh	Hiệu quả	Tối đa 30 giây cho mỗi phiếu mượn sách. Hỗ trợ thiết bị đọc mã vạch. Tối đa 10 giây phải có kết quả tra cứu.	

1.2.4 Khảo sát một số phần mềm tiêu biểu minh họa cho giai đoạn xác định yêu cầu.

A. Phần mềm hỗ trợ giải bài tập phân số.

Bộ phận: Giáo viên. **Mã số: GV**

STT	Công việc	Loại công việc	Quy định/Công thức liên quan	Biểu mẫu liên quan	Ghi chú

1	Soạn tóm tắt lý thuyết và ví dụ minh họa	Lưu trữ			
2	Soạn đề bài tập	Lưu trữ	GV_QĐ 2	GV_BM 2	
3	Soạn đáp án	Lưu trữ	GV_QĐ 3	GV_BM 3	
4	Chấm điểm	Tính toán	GV_QĐ 4		

stt	**Mã số**	**Tên Quy định/ Công thức**	**Mô tả chi tiết**	**Ghi chú**
1.	GV_QĐ2	Quy định soạn đề bài tập	■ Đề bài được giới hạn chỉ là biểu thức các phép toán trên phân số với tối đa 4 phân số thành phần. ■ Có 3 mức bài tập: 1. Chỉ gồm 2 phân số và 1 phép toán. 2. Chỉ gồm 3 phân số và 2 phép toán. 3. Hỗn hợp nhiều phân số (tối đa 4 phân số) với nhiều phép toán ■ Có 4 loại phép toán : + - * /	
2.	GV_QĐ 3	Quy định soạn đáp án bài tập (cũng là quy định soạn bài giải của học sinh)	Mỗi bước giải chỉ được phép rút gọn biểu thức bằng các thực hiện phép tính trên 2 phân số. Thứ tự thực hiện phép tính theo quy tắc độ ưu tiên như sau : Ưu tiên 1 : nhân chia cao hơn công trừ. Ưu tiên 2 : bài toán ưu tiên bên phải	

			Riêng đối với bài giải của học sinh cho phép bỏ qua các bước trung gian.	
3.	GV_QĐ 4	Quy định chấm điểm	■ Có đáp án cuối cùng 🕘 Thực hiện hơn hoặc bằng 50% các bước so với đáp án : o Đã rút gọn : 10 o Chưa rút gọn : 8 🕘 Thực hiện dưới 50% các bước so với đáp án : o Đã rút gọn : 9 o Chưa rút gọn : 7 ■ Có đáp án cuối cùng sai 🕘 Thực hiện hơn hoặc bằng 70% các bước so với đáp án : 5 🕘 Thực hiện từ 50% đến dưới 70% các bước so với đáp án : 3 🕘 Thực hiện từ 50% các bước so với đáp án : 0	

GV_BM 2:

Đề bài tập của giáo viên.

Thực hiện các phép tính trên biểu thức các phân số :

<phân số> [phép toán] <phân số> [phép toán] ...

GV_BM 3:

Đáp án của giáo viên (bài giải của học sinh)

Đề bài: _____

Các bước biến đổi tương đương :

Bước 1: ...

Bước 2: ...

Bước 3: ...

Đáp số: ...

Bộ phận: Học sinh. Mã số: HS

stt	Công việc	Loại công việc	Quy định liên quan	Biểu mẫu liên quan	Ghi chú
1	Chọn bài tập	Tra cứu	GV_QĐ 2	GV_BM 2	
2	Giải bài tập	Lưu trữ	GV_QĐ 3	GV_BM 3	
3	Xem tóm tắt lý thuyết	Kết xuất			
4	Xem đánh giá và đáp án	Kết xuất	GV_QĐ 3 GV_QĐ 4	GV_BM 3	

2. Mô hình hóa yêu cầu hệ thống

Các mô tả yêu cầu trong giai đoạn xác định yêu cầu chỉ mô tả chủ yếu các thông tin liên quan đến việc thực hiện các nghiệp vụ trong thế giới thực chưa và chưa thể hiện rõ nét việc thực hiện các nghiệp vụ này trên máy tính. Mô tả thông qua các văn bản dễ gây ra nhầm lẫn và không trực quan.

Ví dụ: Xét yêu cầu lập hóa đơn bán sách, yêu cầu này chỉ mô tả biểu mẫu về hóa đơn, qui định lập hóa đơn và chưa thể hiện cách thức lập hóa đơn trên máy tính

Mục tiêu của mô hình hóa: Cho phép ta hiểu 1 cách chi tiết hơn về ngữ cảnh vấn đề cần giải quyết một cách trực quan và bản chất nhất (thông tin cốt lõi) yêu cầu.

Kết quả: cho một mô hình mô tả lại toàn bộ hoạt động của hệ thống thực. Mỗi phương pháp phân tích đưa ra một kiểu sơ đồ hay mô hình để xây dựng hệ thống.

Kỹ thuật phân tích là cách tiến hành sao cho thu thập được những yêu cầu của người sử dụng từ đó trình bày lại nhu cầu đó trên mô hình, chi tiết hóa sơ đồ hay mô hình bằng đặc tả chức năng, đặc tả dữ liệu thông qua phân tích gốc nhìn, phân tích đối tượng, phân tích dữ liệu thu thập được ở các bước trên. Trước khi đi vào tìm hiểu các phương pháp biểu diễn bằng mô hình, chúng ta hãy xem qua một số nguyên lý phân tích.

2.1 Các nguyên lý mô hình hóa

a. Mô hình hóa miền thông tin (nguyên lý phân tích 1)

Phải hiểu và biểu diễn được miền thông tin

- ③ Định danh dữ liệu (đối tượng, thực thể)
- ③ Định nghĩa các thuộc tính
- ③ Thiết lập các mối quan hệ giữa các dữ liệu

b. Mô hình hóa chức năng (nguyên lý phân tích 2)

Bản chất của phần mềm là biến đổi thông tin

- ③ Định danh các chức năng (biến đổi thông tin)
- ③ Xác định cách thức dữ liệu (thông tin) di chuyển trong hệ thống
- ③ Xác định các tác nhận tạo dữ liệu và tác nhân tiêu thụ dữ liệu

c. Mô hình hóa hành vi (nguyên lý phân tích 3)

Phần mềm (hệ thống) có trạng thái (hành vi)

③ Xác định các trạng thái hệ thống

ví dụ: giao diện đồ họa, section trong ứng dụng web

③ Xác định các dữ liệu làm thay đổi hành vi hệ thống

ví dụ: bàn phím, chuột, các cổng thông tin…

d. Phân hoạch các mô hình (Nguyên lý phân tích 4)

Làm mịn, phân hoạch và biểu diễn các mô hình ở các mức khác nhau

③ Làm mịn các mô hình dữ liệu

③ Tạo cây (mô hình) phân rã chức năng

③ Biểu diễn hành vi ở các mức chi tiết khác nhau

e. Tìm hiểu vấn đề bản chất (Nguyên lý phân tích 5)

③ Nhìn nhận bản chất của yêu cầu

③ Không quan tâm đến cách thức cài đặt

2.3 Sơ đồ phân rã chức năng

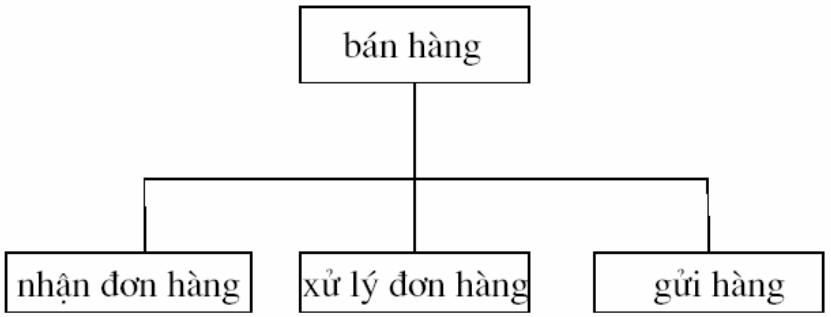
Sơ đồ phân rã chức năng - Function Decomposition Diagram - FDD: Nêu lên các chức năng thông qua việc mô tả các tính chất của đầu vào và đầu ra

③ Xác định phạm vi của hệ thống

③ Phân hoạch chức năng

③ Tạo nền tảng cho thiết kế kiến trúc hệ thống

Ví dụ: Sơ đồ phân rã chức năng



2.3 Mô hình bản mẫu (protoype)

Khi xác định yêu cầu, nhà phát triển phần mềm dựa trên các ý tưởng hay yêu cầu của khách hàng đưa ra một bản thiết kế sơ bộ một số màn hình giao diện và tiến hành mô phỏng hay giả lập sơ bộ một số chức năng, Có thể xem đây bước cài đặt bản mẫu đầu tiên và chuyển cho người sử dụng. Bản mẫu này chỉ nhằm để mô tả cách thức phần mềm hoạt động cũng như cách người sử dụng tương tác với hệ thống. Nhằn giúp cho người dùng hình dung được diện mạo ban đầu của yêu cầu mà họ đặt ra. Mô hình này cũng cần có sự hỗ trợ giữa kỹ sư phân tích và kỹ sư thiết kế phần mềm phối hợp thực hiện.

Người sử dụng khi xem xét bản mẫu sẽ đưa ra ý kiến đóng góp và phản hồi thông tin đồng ý hay không đồng ý phương án thiết kế của bản mẫu đã đưa ra. Nếu người sử dụng đồng ý với bản mẫu đã đưa thì người phát triển sẽ tiến hành cài đặt thực sự. Ngược lại cả hai phải quay lại giai đoạn xác định yêu cầu. Công việc này được lặp lại liên tục cho đến khi người sử dụng đồng ý với các bản mẫu do nhà phát triển đưa ra.

2.4 Sơ đồ luồng dữ liệu

Sơ đồ luồng dữ liệu - Data flow diagram – DFD

Đây là mô hình cho phép xem toàn bộ sơ đồ luồng dữ liệu bên trong hệ thống. Cách thức dữ liệu được xử lý bên trong hệ thống.Có nhiều mức chi tiết khác nhau. Có nhiều biến thể mở rộng khác nhau. Xem chi tiết ở chương kế tiếp thiết kế phần mềm. Ngoài ra còn có mô hình thực thể kết hợp được trình bày trong hầu hết các cuốn sách Cơ sở dữ liệu hoặc Thiết kế CSDL.

2.5 Mô hình hướng đối tượng

Phương pháp phân tích hướng đối tượng hình thành giữa thập niên 80 dựa trên ý tưởng lập trình hướng đối tượng. Phương pháp này đã phát triển, hoàn thiện và hiện nay rất phổ dụng. Nó dựa trên một số khái niệm cơ bản sau:

Đối tượng (Object): gồm dữ liệu và thủ tục tác động lên dữ liệu này.

Đóng gói (Encapsulation): Không cho phép tác động trực tiếp lên dữ liệu của đối tượng mà phải thông qua các phương pháp trung gian.

Lớp (Class): Tập hợp các đối tượng có chung một cấu trúc dữ liệu và cùng một phương pháp.

Kế thừa (Heritage): tính chất kế thừa là đặc tính cho phép định nghĩa một lớp mới từ các lớp đã có bằng cách thêm vào đó những dữ liệu mới, các phương pháp mới có thể kế thừa những đặc tính của lớp cũ.

a. Mô hình nắm bắt yều cầu hướng đối tượng bằng UML

Mục đích của hoạt động nắm bắt yêu cầu là xây dựng mô hình hệ thống mà sẽ được xây dựng bằng cách sử dụng các use-case. Các điểm bắt đầu cho hoạt động này khá đa dạng:

- Từ mô hình nghiệp vụ (business model) cho các ứng dụng nghiệp vụ.
- Từ mô hình lĩnh vực (domain model) cho các ứng dụng nhúng (embeded)
- Từ đặc tả yêu cầu của hệ thống nhúng được tạo bởi nhóm khác và hoặc dùng các phương pháp đặc tả khác (thí dụ hướng cấu trúc.

- Từ điểm nào đó nằm giữa các điểm xuất phát trên.

Mô hình use-case:

- Actor: người/ hệ thống ngoài/ thiết bị ngoài tương tác với hệ thống
- Use-case: các chức năng có nghĩa của hệ thống cung cấp cho các actor
  - luồng các sự kiện (flow of events)
  - các yêu cầu đặc biệt của use-case
- Đặc tả kiến trúc
- Các thiết kế mẫu giao diện người dùng

b. Mô hình phân tích hướng đối tượng với UML

Mục đích của hoạt động phân tích yêu cầu là xây dựng mô hình phân tích với các đặc điểm sau:

- Dùng ngôn ngữ của nhà phát triển để miêu tả mô hình
- Thể hiện gốc nhìn từ bên trong hệ thống
- Được cấu trúc từ các lớp phân tích và các package phân tích
- Được dùng chủ yếu cho các nhà phát triển để hiểu cách thức tạo hình dạng hệ thống
- Loại trừ mọi chi tiết dư thừa, không nhất quán
- Phát họa hiện thực các chất năng bên trong hệ thống
- Định nghĩa các dẫn xuất use-case, mỗi dẫn xuất use-case cấp phân tích miêu tả sự phân tích 1 use-case

Mô hình phân tích= hệ thống phân tích

- Các class phân tích: lớp biên, lớp thực thể, lớp điều khiển
- Các dẫn xuất use-case cấp phân tích: các lược đồ lớp phân tích, các lược đồ tương tác, luồng sự kiện, các yêu cầu đặc biệt của use-case
- Các package phân tích
- Đặc tả kiến trúc

Lưu ý: Các mô hình hướng đối tượng cho từng giai đoạn phát triển phần mềm được trình bày ở giáo trình khác. Xem chi tiết cụ thể ở giáo trình môn Phân tích thiết kế hướng đối tượng với UML.

2. 6 Ví dụ minh họa từ yêu cầu sang mô hình hóa

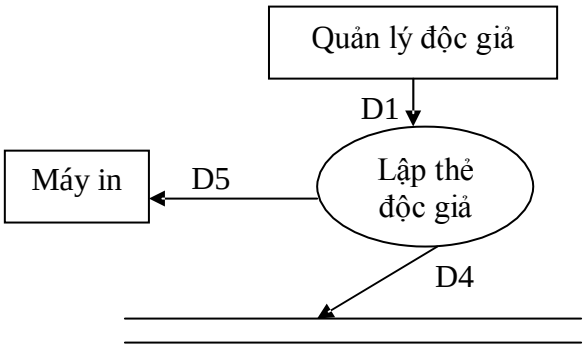
Ví dụ 1: Xét phần mềm quản lý thư viện với 4 yêu cầu

- Lập thẻ độc giả

- Nhận sách
- Cho mượn sách
- Trả sách

Giai đoạn 2 : Mô hình hóa yêu cầu

③ Sơ đồ luồng dữ liệu cho công việc lập thẻ độc giả



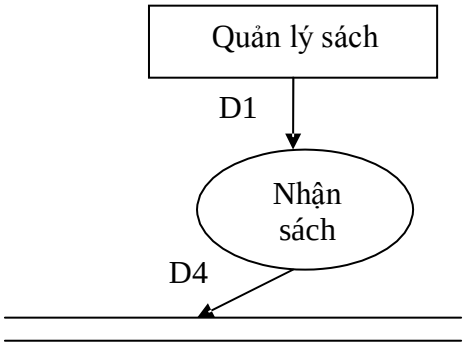
D1: Thông tin về thẻ độc giả cần nhập

D4: Thông tin về thẻ độc giả cần lưu trữ trên bộ nhớ phụ

D5: Thông tin trên thẻ độc giả (trong thế giới thực)

Xử lý thẻ độc giả: Kiểm tra tính hợp lệ của thẻ trước ghi nhận và in

③ Sơ đồ luồng dữ liệu cho công việc nhận sách

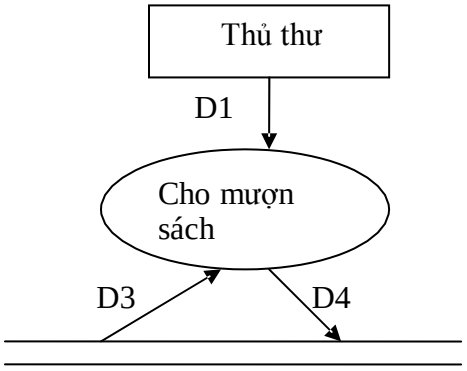


D1: Thông tin về thẻ sách cần nhập

D4: Thông tin về sách cần lưu trữ trên bộ nhớ phụ

Xử lý nhập sách: Kiểm tra tính hớp lệ của sách trước khi ghi nhận trên bộ nhớ phụ

③ Sơ đồ luồng dữ liệu cho công việc cho mượn sách



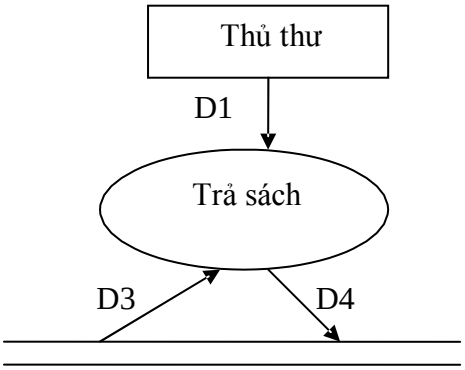
D1: Thông tin về độc giả và sách muốn mượn

D3: Thông tin được sử dụng cho việc kiểm tra qui định mượn sách

D4: Thông tin về việc mượn sách

Xử lý cho mượn sách: Kiểm tra tính hợp lệ của việc mượn sách ghi nhận trên bộ nhớ phụ

③ Sơ đồ luồng dữ liệu cho công việc trả sách



D1: Thông tin về độc giả và sách trả

D3: Thông tin sử dụng cho việc kiểm tra qui định trả sách

D4: Thông tin về việc trả sách

Xử lý trả sách: Kiểm tra tính hợp lệ của việc trả sách ghi nhận trên bộ nhớ.

Chương 3: THIẾT KẾ PHẦN MỀM

1. Tổng quan về thiết kế

Trong thiết kế, chúng ta định hình hệ thống và tìm dạng thức của nó (kể cả kiến trúc) mà đáp ứng được mọi yêu cầu, cả yêu cầu phi chức năng và các ràng buộc khác - được đặt ra cho hệ thống đó. Một đầu vào cơ bản cho thiết kế là kết quả thu được từ phần tích, đó là mô hình phân tích. Xét một cách chi tiết mục đích của thiết kế là:

- Thu được sự hiểu biết sâu về các yêu cầu phi chức năg và các ràng buộc có liên quan tới ngôn ngữ lập trình, sử dụng lại thành phần, các hệ điều hành, các công nghệ phân tán, các công nghệ cơ sở dữ liệu, các công nghệ giao diện người dùng, các công nghệ quản lý các giao dịch.
- Tạo ra một đầu vào thích hợp và xuất phát điểm cho các hoạt động cài đặt tiếp theo sau bằng cách nắm bắt các yêu cầu về mỗi hệ thống cụ thể, các giao diện, và các lớp.
- Có khả năng phân rã việc cài đặt thành các mẩu nhỏ dễ quản lý hơn được nhiều đội phát triển khác nhau xử lý và có thể tiến hành đồng thời. Điều này sẽ có ích trong các trường hợp khi mà không thể tiến hành sự phân rã giữa các kết quả thu được từ nắm bắt các yêu cầu hoặc phân tích.
- Nắm bắt sớm các giao diện chủ yếu giữa các hệ thống con trong vòng đời của phần mềm. Điều này sẽ có ích khi chúng ta suy luận về kiến trúc và khi chúng ta sử dụng các giao diện như những công cụ đồng bộ các đội phát triển khác nhau
- Trực quan hóa và suy luận thiết kế bằng cách sử dụng một hệ thống các ký pháp chung.
- Tạo ra một sự trừu tượng hóa liên tục của việc cài đặt của hệ thống, tức là cài đặt sự làm mịn dần thiết kế bằng cách đắp "thịt" vào khung xương nhưng không thay đổi cấu trúc của nó.

Mục tiêu của phần này là giới thiệu một số phương pháp và kỹ thuật chính trong thiết kế, đối với việc triển khai một hệ thống thành nhiều hệ thống con và hệ thống con thành nhiều thành phần (components), và quản lý những vấn đề liên quan đến cấu trúc nội tại của những thành phần hệ thống. Đầu tiên chúng ta xem qua vài kỹ thuật thiết kế. Kế đến chúng ta sẽ xét qua một vài kỹ thuất thiết kế và phương pháp nền tảng một cách chi tiết và một số ví dụ minh họa. Thêm vào đó, chúng ta bàn qua những khía cạnh thiết kế như thiết kế giao diện người dùng và mô đun hóa.

1.1 Kỹ thuật thiết kế

- Thiết kế đặc tả đi đến kỹ thuật cốt lõi của tiến trình của công nghệ phần mềm.
- Thiết kế đặc tả được cung cấp xem xét những mô hình của tiến trình phần mềm được sử dụng.
- Thiết kế phần mềm là bước đầu tiên trong ba hoạt động kỹ thuật - thiết kế, phát sinh mã nguồn, và thử nghiệm –đó là những yêu cầu trong xây dựng và phát triển phần mềm.

Một trong những điểm mấu chốt chính đối với độ phức tạp của hệ thống phần mềm là sự trừu tượng. Có hai phương pháp chính: thiết kế Top-down và thiết kế bottom-up

1.1.1 Thiết kế trên xuống (Top-down)

-Thiết kế bắt đầu với việc phân tích những định nghĩa yêu cầu và không nên xem xét việc thực hiện chi tiết đầu tiên.

- Một dự án được triển khai thành những dự án nhỏ, thủ tục này phải được lặp lại cho đến khi những nhiệm vụ con trở nên đơn giản sao cho một thuật toán được tính toán và giải quyết.

1.1.2 Thiết kế từ dưới lên (Bottom–up)

Ý tưởng nền tảng: Hiểu được phần cứng và tầng trên của nó như một cơ chế trừu tượng.

Kỹ thuật: Thiết kế từ dưới lên bắt đầu được cho bởi máy cụ thể và liên tiếp phát triển một máy trừu tượng sau khi những máy khác được thêm vào những thuộc tính cần thiết cho đến khi một máy đã đạt được kết quả mà cung cấp những chức năng người dùng yêu cầu.

1.1.3 Thiết kế hệ thống

Trong hệ thống lớn, tiến trình thiết kế bao gồm một yếu tố thiết kế hệ thống mà chức năng được phân chia thành những chức năng phần mềm và phần cứng.

Những thuận lợi của chức năng thực hiện trong phần cứng là thành phần phần cứng phân phối thực hiện tốt hơn đơn vị phần cứng. Nút thắt của hệ thống được xác định và thay thế bởi thành phần của phần cứng, như thế việc tối ưu phần mềm là hết sức tốn kém.

Cung cấp tốc độ phần cứng có nghĩa là thiết kế phần mềm có thể được cấu trúc cho khả năng thích ứng và khả năng xem xét thực thi cả chức năng.

1.1.4 Thiết kế bản mẫu (prototype)

Thiết kế bản mẫu nghĩa là đưa ra các màn hình giao diện sơ bộ, hay các bản thiết kế phác thảo nháp cho người dùng tham khảo trước khi đi vào thiết kế chi tiết, hay chức năng cụ thể. Các bản thiết kế này được soạn thảo dưới dạng sưu liệu hoặc một số phần mềm có khả năng thiết kế nhanh giao diện, các kỹ sư thiết kế có thể sử dùng một số phần mềm chuyên dụng để soạn thảo nhanh như MS Visual Basic, Visual C++, MS Visual Studio … với trang web thì có thể dùng Front Page, MS Visual Interdev chỉ với những đoạn chương trình đơn giản được cài đặt. Đây cũng có thể coi là bước đệm cơ bản trước khi đi vào cài đặt chi tiết cho từng chương trình con hay môđun con v.v.

1.1.5 Phân rã thiết kế

Tiến trình thiết kế không chỉ ảnh hưởng đến phương pháp thiết kế mà còn ảnh hướng đến tiêu chuẩn được sử dụng để phân rã hệ thống.

Phần lớn những yếu tố cơ bản của phân rã được đề ra.

Phương pháp phân loại phân rã

1.1.5.1 Phân rã hướng chức năng

- Khía cạnh của hệ thống hướng chức năng tạo nên cốt lõi của thiết kế

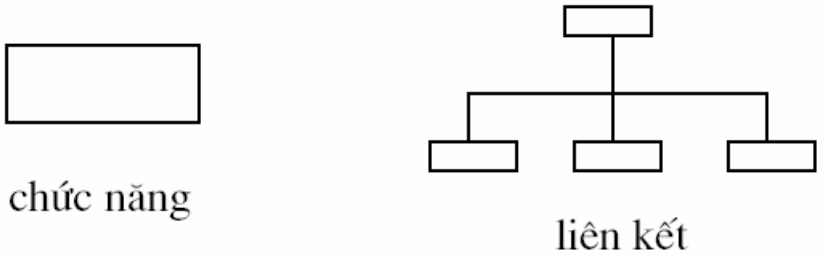
- Dựa trên những yêu cầu chức năng chứa trong những định nghĩa yêu cầu, phân rã hướng đến tác nhiệm của toàn bộ hệ thống được tổ chức

Sơ đồ phân rã chức năng - Function Decomposition Diagram - FDD: Nêu lên các chức năng thông qua việc mô tả các tính chất của đầu vào và đầu ra

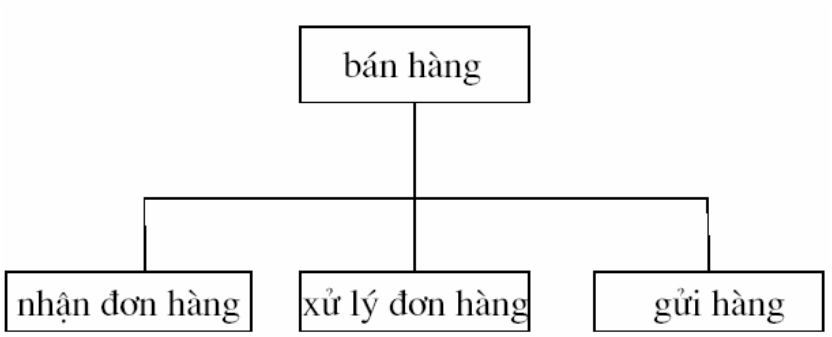
③ Xác định phạm vi của hệ thống

③ Phân hoạch chức năng

③ Tạo nền tảng cho thiết kế kiến trúc hệ thống



Ví dụ: **Sơ đồ phân rã chức năng**



1.1.5.2 Phân rã hướng dữ liệu

Tiến trình thiết kế tập trung trên khía cạnh hệ thống hướng đến dữ liệu. Chiến lược thiết kế hướng đến chính dữ liệu đực thực hiện. Phân rã những bộ phận hệ thống từ việc phân tích dữ liệu

1. Sơ đồ luồng dữ liệu

Sơ đồ luồng dữ liệu - Data flow diagram - DFD

Cho phép xem toàn bộ sơ đồ luồng dữ liệu bên trong hệ thống. Cách thức dữ liệu được xử lý bên trong hệ thống.Có nhiều mức chi tiết khác nhau. Có nhiều biến thể mở rộng khác nhau

a. Khái niệm và ký hiệu

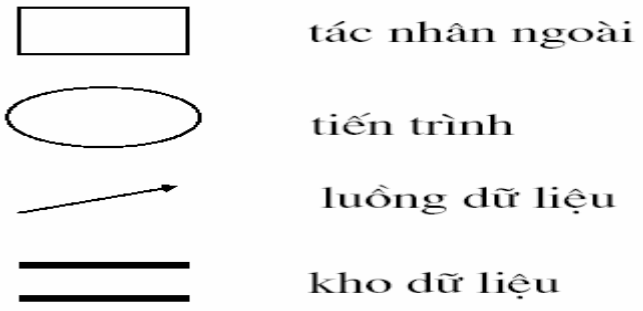
Tác nhân ngoài: đối tượng bên ngoài hệ thống, nguồn phát sinh hay thu nhận dữ liệu

Tiến trình: Thao tác đối với thông tin hay khối dữ liệu

Luồng dữ liệu: luồng thông tin di chuyển trong hệ thống

Kho dữ liệu:nơi lưu trữ dữ liệu

Các ký hiệu:



b. Các nguyên tắc và bước xây dựng mô hình DFD

Các bước xây dựng DFD:

- ③ Phân rã chức năng hệ thống
- ③ Liệt kê các tác nhân, các khoản mục dữ liệu
- ③ Vẽ DFD cho các mức

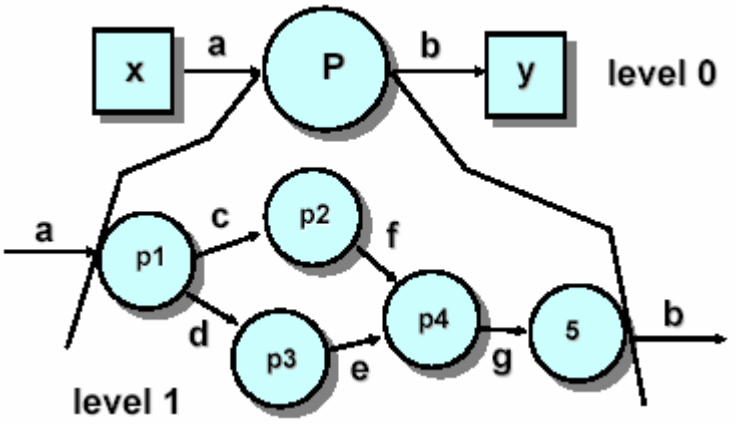
Nguyên tắc:

- ③ Các tiến trình phải có luồng vào luồng ra
- ③ Không có luồng dữ liệu trực tiếp giữa các tác nhân với tác nhân và kho dữ liệu
- ③ Luồng dữ liệu không quay lại nơi xuất phát
- ③ Bắt đầu bằng DFD mức 0, liệt kê các tác nhân ngoài ở mức 0
- ③ Các mức(cấp) sơ đồ:
    - mức 0: Toàn bộ phần mềm là khối xử lý
    - mức 1: Sơ đồ mức 0 có thể phân rã thành nhiều sơ đồ mức 1, các sơ đồ mức 1này phải đảm bảo thể hiện đầy đủ ý nghĩa sơ đồ mức 0 (tác nhân, thiết bị, luồng dữ liệu, xử lý, bộ nhớ phụ)
    - mức 2: Mỗi sơ đồ mức 1 có thể phân rã thành nhiều sơ đồ mức 2 tương ứng như việc phân rã của sơ đồ mức 0
    - …

Trình bày sơ đồ: Trong mỗi cấp có 2 hình thức trình bày sơ đồ

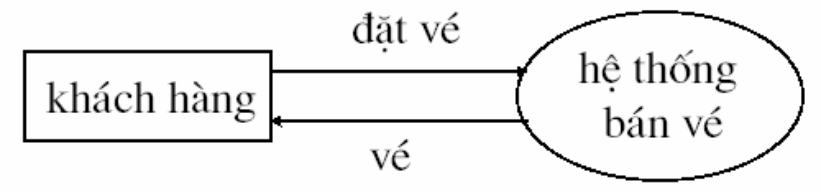
- Dạng tổng hợp : Chỉ có một khối xử lý chung, tất cả các luồng dữ liệu chỉ tập trung liên quan đến khối xử lý chung này
- Dạng chi tiết: Bao gồm nhiều khối xử lý với luồng dữ liệu riêng biệt cho từng khối xử lý

Ví dụ: biểu diễn các mức của DFD

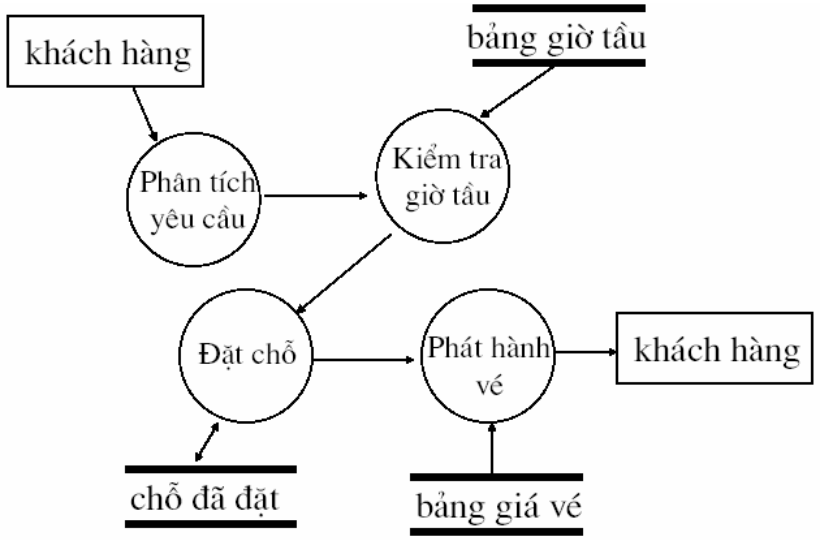


Ví dụ DFD hệ thống bán vé

mức 0:



mức 1: DFD mức 1



2. Các hướng tiếp cận lập sơ đồ luồng dữ liệu

③ Có nhiều hướng tiếp cận để tạo lập các sơ đồ luồng dữ liệu. Giáo trình này giới hạn xem xét 3 cách tiếp cận chính

+ Hướng tiếp cận từ trên xuống dưới (topdown)

+ Hướng tiếp cận từ dưới lên trên (bottomup)

+ Hướng tiếp cận phối hợp

③ **Tiếp cận từ trên xuống:**

Quá trình thực hiện theo hướng tiếp cần này như sau:

- Lập sơ đồ luồng dữ liệu cấp 0 (xem xét tất cả các luồng dữ liệu nhập xuất, tất cả các yêu cầu xử lý của phần mềm
- Phân rã sơ đồ luồng dữ liệu cấp 0 thành nhiều sơ đồ luồng dữ liệu cấp 1. Có 2 cách phân rã:

  + Phân rã các xử lý của phần mềm thành nhiều xử lý con và quyết định các luồng dữ liệu tương ứng trên các xử lý con này.

  + Phân rã các luồng dữ liệu nhập xuất thành nhiều luồng dữ liệu con và quyết định các xử lý tương ứng với các luồng dữ liệu con này.

- Quá trình kết thúc khi đạt đến các sơ đồ không thể tiếp tục phân rã được (sơ đồ lá). Thông thường đây là sơ đồ tương ứng với công việc cụ thể của một nhà chuyên môn trong thế giới thực.

Đánh giá

- Tiếp cận này thích hợp với các phần mềm có số lượng người dùng, số lượng các yêu cầu ít (nếu ngược lại sơ đồ cấp 0 sẽ rất phức tạp và khó lập chính xác).
- Tiếp cận này đặc biệt thích hợp với các loại phần mềm mà vì lý do nào đó các hệ thống yêu cầu chưa được xác định rõ ngay từ đầu (ví dụ các phần mềm hệ thống).
- Thông thường cách tiếp cận này ít đực sử dụng.

③ Hướng tiếp cận từ dưới lên (bottomup)

Quá trình thực hiện theo hướng tiếp cận này như sau

- Lập sơ đồ luồng dữ liệu ở mức cao nhất. Các sơ đồ này sẽ không được tiến hành phân rã thành các sơ đồ có cấp lớn hơn (thông thường đây là sơ đồ ứng với một công việc cụ thể của một người dùng nào đó trong thế giới thực)

  + Tích hợp các sơ đồ này để tạo lập các sơ đồ có cấp nhỏ hơn (thông thường các sơ đồ được chọn tích hợp theo một tiêu chí cụ thể: cùng một người sử dụng, cùng một loại yêu cầu, v.v). Có 2 cách tích hợp:

  + Tích hợp các xử lý của các sơ đồ cấp k vào sơ đồ cấp k-1 và giữ nguyên các luồng dữ liệu của các sơ đồ cấp k

  + Tích hợp đồng thời các xử lý và các luồng dữ liệu của các sơ đồ cấp k để tạo lập sơ đồ cấp k-1.

- Quá trình kết thúc khi đạt đến các sơ đồ cấp 0

Đánh giá

- Tiếp cận này rất thích hợp với các phần mềm có hệ thống yêu cầu chi tiết, cụ thể và có qui mô yêu cầu (số lượng người dùng, số lượng yêu cầu) thuộc mức trung bình (các đồ án môn học
- Tiếp cận này sẽ khó khăn nếu qui mô yêu cầu lớn và chưa thật rõ ràng chi tiết
- Cách tiếp cận này sẽ được sử dụng trong giáo trình với các đồ án môn học và các ví dụ minh họa

③ **Hướng tiếp cận phối hợp:**

Quá trình thực hiện theo hướng tiếp cận này như sau:

- Lập sơ đồ luồng dữ liệu cấp k theo một tiêu chí xác định (sơ đồ cho từng người dùng, sơ đồ cho một bộ phận, sơ đồ cho một loại yêu cầu, v.v)
- Phân rã sơ đồ cấp k thành nhiều sơ đồ cấp k+1 tiếp tục cho đến khi đạt được các sơ đồ lá
- Tích hợp các sơ đồ cấp k thành các sơ đồ cấp k-1 tiếp tục cho đến khi đạt được sơ đồ cấp 0

Đánh giá

- Tiếp cận này thích hợp cho các phần mềm có qui mô yêu cầu lớn, phức tạp
- Tiếp cận này được sử dụng rất thường xuyên trong thực tế.

3. Lập sơ đồ luồng dữ liệu cho từng công việc

③ Do các giới hạn đã nêu phía trên việc lập các sơ đồ luồng dữ liệu toàn bộ phần mềm chỉ qui về lập sơ đồ luồng dữ liệu cho từng công việc (sau đó chỉ thực hiện đơn giản một bước tích hợp để có sơ đồ cấp 0)

③ Quá trình lập sơ đồ luồng dữ liệu cho một công việc được tiến hành qua các bước như sau

- Bước 1: Xác định dữ liệu nhập
- Bước 2: Xác định dữ liệu xuất
- Bước 3: Mô tả xử lý

③ **Bước 1: Xác định dữ liệu nhập**

- Dữ liệu nhập từ người dùng sử dụng đươc xác định dựa vào biểu mẫu có liên quan với các lưu ý sau:

  + Không nhập vào các dữ liệu có thể tính toán được dựa trên qui định hay công thức đã có.

  + Không nhập vào các dữ liệu đã được lưu trữ trước đó (qua một công việc khác).

- Dữ liệu nhập từ thiết bị nhập (khác bàn phím) chỉ được xem xét khi có yêu cầu đặc biệt trong một số ứng dụng đặc biệt (hệ thống thời gian thực, hệ thống bản đồ, nhập thông qua sử dụng điện thoại tổng đài điện thoại trong quản lý khách sạn, v.v).

- Dữ liệu nhập (đọc) từ bộ nhớ phụ được xác định dựa trên các qui định công thức liên quan với một số lưu ý:

  + Chỉ đọc dữ liệu thật sự cần thiết cho việc thực hiện xử lý tương ứng (thông tin nhập chưa đủ để xử lý).

+ Để cải tiến chất lượng phần mềm(đặc biệt tính tiến hóa) có thể đọc thêm các tham số phục vụ cho việc xử lý từ bộ nhớ phụ (bảng qui định đơn giá phạt khi trả sách trễ hạn, bảng định mức và đơn giá tiền điện, v,v). Tuy nhiên trong giai đoạn này chỉ nên tập trung vào tính đúng đắn (các chất lượng khác sẽ được xem xét chi tiết trong giai đoạn thiết kế).

③ **Bước 2: Xác định dữ liệu xuất**

- Dữ liệu xuất cho người dùng được xác định dựa trên biểu mẫu liên quan với một số lưu ý như sau

  +Các thông báo về việc xử lý có thực hiện được hay không là luôn luôn phải có và không cần thiết thể hiện trên sơ đồ (thông báo việc mượn sách là không hợp lệ, thông báo lỗi khi tính điểm trung bình mà có môn chưa có điểm, v.v)

  + Để tăng tính tiện dụng, trong tất cả các xử lý đều phải xuất cho người dùng nhiều thông tin (kể cả xử lý lưu trữ, xử lý tính toán). **Tuy nhiên vấn đề này chỉ xem xét và thực hiện trong các giai đoạn sau, nếu chú ý quá sớm đến vấn đề này sẽ làm phức tạp sơ đồ và dễ phạm các sai lầm trong tính đúng đắn.**

- Dữ liệu xuất ra thiết bị xuất (khác màn hình) thông thường là máy in, để tăng tính tiện dụng có thể tuân theo nguyên tắc sau "Tất cả dữ liệu xuất ra màn hình đều cho phép người dùng xuất ra máy in (có thể với cách trình bày khác). Tuy nhiên vấn đề này cũng có thể dời lại xem xét chi tiết trong giai đoạn thiết kế. Các loại thiết bị xuất khác chỉ có trong các loại ứng dụng đặc biệt hoặc do yêu cầu tính tương thích.

- Dữ liệu xuất (ghi) vào bộ nhớ phụ được xác định dựa trên biểu mẫu liên quan với một số lưu ý như sau:

  + Ghi các dữ liệu kết quả mới tạo lập hoặc các dữ liệu đã có nhưng bị thay đổi trong quá trình thực hiện xử lý.

  + Để tăng tính hiệu quả có thể ghi các thông tin bổ sung có liên quan đến các yêu cầu khác. Tuy nhiên tốt nhất vấn đền này được xem xét chi tiết trong giai đoạn thiết kế.

③ **Bước 3: Mô tả xứ lý**

Mô tả quá trình sử dụng dữ liệu nhập D1, D2, D3 để tạo ra các dữ liệu xuất D4, D5, D6 với các lưu ý sau:

- Chỉ mô tả xử lý mà không cần lưu ý đến cách thực hiện nhập xuất (hình thức nhập, tổ chức lưu trữ trên bộ nhớ phụ, câu lệnh cụ thể để đọc, ghi).

- Mô tả chi tiết cách sử dụng dữ liệu nhập để tạo dữ liệu xuất (mô tả càng chi tiết thì việc thiết kế xử lý càng dễ dàng.
- Chỉ chú trọng đến tính đúng đắn mà không nên xem xét quá sớm các yêu cầu chất lượng khác.

Mô tả chính xác thứ tự nhập và xuất (trong một vài trường hợp có thể xuất trước và sau đó mới nhập).

2. Mô hình thực thể quan hệ (Entity – Relation Diagram)

a.Các khái niệm và ký hiệu

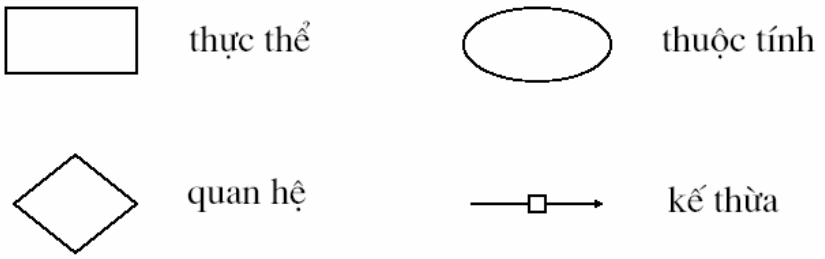
Thực thể là đối tượng thế giới thực mà chúng ta muốn xử lý, có thể là đối tượng thực hay trừu tượng

Thuộc tính: đặc điểm của thực thể

Quan hệ: là mối liên hệ giữa các thực thể, là thông tin cần lưu trừu/ xử lý

Kế thừa: là quan hệ kế thừa giữa các thực thể

b. Ký hiệu



Với giáo trình này chỉ dừng lại giới thiệu khái niệm, mô hình được học ở các giáo trình Phân tích Thiết kế hệ thống thông tin

1.1.5.3 Phân rã hướng đối tượng

Khía cạnh hệ thống hướng đối tượng cung cấp tập trung chủ yếu của thiết kế.

Hệ thống phần mềm được xem xét như tập hợp các đối tượng thông tin với nhau. Mỗi đối tượng có cấu trúc dữ liệu mà không được nhìn thấy từ bên ngoài và thao tác của chúng có thể được thực hiện trên cấu trúc này.

Những điểm cơ bản của phân rã hướng chính nó đến tính đồng nhất giữa dữ liệu và thao tác và dựa trên sự che dấu thống tin và dẫn xuất kế thừa.

1.2. Thiết kế giao diện người dùng

Thiết kế giao diện người dùng là một tác nhiệm trong giai đoạn thiết kế. Thiết kế giao diện được hỗ trợ một phần trong thiết kế dạng mô hình bản mẫu (prototype) ở giaiđoạn xác định nhằm làm sáng tỏ các yêu cầu từ người dùng, xác định đúng yêu cầu người dùng, cũng như thõa mãn các đòi hỏi về mặt thẩm mỹ, giao diện đẹp cho khách hàng. Nếu khách hàng đã đồng ý với bản mẫu đã đưa ra trong giai đoạn xác định yêu cầu thì kỹ sư thiết kế chỉ việc phát triển và hoàn chỉnh thêm giao diện để đảm bảo tính tiện dụng, đảm bảo chính xác yêu cầu người dùng. Nếu không, người thiết kế phải sáng tạo thêm theo một số tiêu chí về thẩm mỹ, tiện dụng, đầy đủ yêu cầu thông tin:

Chế độ (Modes):

Chế độ chương trình là trường hợp mà người dùng chỉ có thể thực hiện ở một số thao tác giới hạn.Kỹ thuật cửa sổ cung cấp dịch vụ có giá trị của chế độ chương trình.

Cửa sổ trợ giúp, người dùng có thể thực hiện vài thao tác con tương ứng trong những cửa sổ khác nhau thể hiển bởi những chế độ chương trình khác nhau.

Thực đơn (Menu):

Pop-up menu: thiết kế hiệu quả bởi chúng có thể xuất hiện bất cứ vị trí nào và ít đòi hỏi di chuyển chuột (mouse).

Pull-down menu: cho phép cấu trúc tốt hơn việc mở rộng tập lệnh và dễ dàng sử dụng.

Người dùng chọn vào thực đơn bằng chuột để hiển thị tất cả lệnh thao tác trên menu và có thể chọn lệnh giống như sử dụng chuột click vào menu. Chúng ta có thể phân loại menu theo tập lệnh thao tác, tập lệnh thao tác với tham số, tập lệnh chuyển đối chế độ người dùng.

1.3.Cửa sổ hội thoại (dialog window):

Đảm bảo tính đồng nhất trong giao diện người dùng, tránh những giải thích dài dòng nên ngắn gọn cô động như cách đặt nhãn Label, Checkbox, Button, List box.

Màu sắc (Color):

Màu sắc chủ yếu chỉ dùng ở những nơi cần diễn đạt những yêu cầu nào đó, hay muốn nhấn mạnh ý nghĩa nào đó, hoặc dấu hiệu cảnh báo nguy hiểm, cùng đừng hóa tô điểm qúa cho giao diện. Ví dụ màu chữ đen trên nên trắng thường dễ đọc nhất cho khả năng làm việc hàng ngày, còn màu chữ trắng trên nền xanh thì khó đọc …

Âm Thanh (Sound):

Cách tốt nhất tập trung sự chú ý của người dùng. Ứng dụng phù hợp trong các tình huống xử lý lỗi, sự kiện không chắc chắn, tạm thời. Tạo những âm thanh khác nhau với những sự kiện khác nhau, tránh dùng âm thanh gây ồn.

Tính kiên định:

Menu lệnh với những chức năng giống nhau nên vị trí giống nhau thậm chí ở những chương trình khác nhau. Phím nóng trên menu lệnh nên cố định. Nút lệnh với những chức năng tương tự giống nhãn và vị trí liên hệ như nhau trong những cửa sổ hội thoại.

1.4 Thiết kế hướng chức năng

Thiết kế hướng chức năng có nghĩa là tập trung trên thuật toán để giải quyết vấn đề.

Hãy tưởng tượng một thuật toán như một hàm tính toán mà tính kết quả từ những tham số cơ bản được cho. Tại thời điểm bắt đầu giai đoạn thiết kế, thuật toán như là hộp đen mà nội dung thì không được biết. Những tác nhiệm càng khó để giải quyết là thuật toán giải quyết của nó. Như vậy, rõ ràng thực hiện mô đun hóa để phân rã những tác nhiệm thành tác nhiệm con độc lập nhau, nhờ đó những thuật toán cho những giải quyết của tác nhiệm con được xem như là những hộp đen. Kết quả chung của những giải pháp trở thành mạng những thuật toán con gộp lại.

1.5.Thiết kế hướng đối tượng

Thiết kế hướng đối tượng là tổ chức xoay quanh những đối tượng và mối liên hệ giữa chúng

Thiết kế lớp đối tượng: mô tả các lớp đối tượng (thuộc tính, hành động

Thiết kế giao diện: Mô tả giao diện của lớp đối tượng trong từng trách nhiệm của chúng

Thiết kế dữ liệu: Mô tả cách thức tổ chức lưu trữ các đối tượng trên bộ nhớ phụ (chỉ có khi không sử dụng cơ sở dữ liệu hướng đối tượng)

Khả năng dùng lại đóng vai trò quan trong trong lập trình hướng đối tượng đối với chuên viên tin học (phải thực hiện nhiều phần mèm). Với tiếp cận mới việc tái sử dụng sẽ rất dễ dàng, nhanh chóng và tốn ít chi phí nhất có thể có (các phần mềm trong cùng lớp phần mềm bao gồm các đối tượng tương tự như nhau, cách xây dựng đối tượng tương tự như nhau cho các phần mềm khác nhau).

3. Kiến trúc phần mềm

Kiến trúc phần mềm bao gồm các thành phần cơ bản: thành phần giao diện, thành phần xử lý, phần dữ liệu. Khi thiết kế một phần mềm cụ thể, người kỹ sư tin học phải chọn lựa và ra quyết định về các "vật liệu" được dùng trong các thành phần. Sau khi đã quyết định xong, kết quả sẽ được mô tả lại hay đặc tả dưới dạng các bản vẽ phần mềm, dưới dạng sưu liệu.

Kết quả của thiết kế là các mô hình phần mềm. Mô hình cung cấp các thông tin chi tiết về 3 thành phần

- Thành phần giao diện
- Thành phần xử lý
- Thành phần dữ liệu

Thông tin về các thành phần giao diện bao gồm các thông tin sau:

- Nội dung và hình thức trình bày các màn hình giao tiếp của phần mềm. Ý niệm về màn hình giao tiếp sẽ được trình bày chi tiết trong phần thiết kế giao diện.
- Hệ thống các thao tác mà người dùng có thể thực hiện trên màn hình giao tiếp và xử lý tương ứng của phần mềm. Các ý niệm về thao tác và xử lý trên màn hình giao tiếp sẽ được trình bày chi tiết trong phần thiết kế giao diện.

Thông tin về thành phần xử lý bao gồm các thông tin sau:

- Hệ thống các kiểu dữ liệu được sử dụng trong phần mềm. Các kiểu dữ liệu này được mô tả cách tổ chức lưu trữ dữ liệu trong bộ nhớ chính của phần mềm
- Hệ thống các hàm được sử dụng trong phần mềm. Các hàm này sẽ thể hiện tương ứng việc thực hiện một công việc nào đó của thế giới thực trên máy tính (kiểm tra tính hợp lệ việc cho mượn sách, ghi vào sổ việc cho mượn sách…v.v)

Thông tin về các thành phần dữ liệu bao gồm các thông tin liên quan đến cách thức tổ chức lưu trữ các dữ liệu (nội dung của việc ghi chép vào sổ sách trong thế giới thực) trên bộ nhớ phụ.

- Dạng lưu trữ được sử dụng của phần mềm. Ý niệm về dạng lưu trữ (tập tin, cơ sở dữ liệu,..v.v) sẽ được trình bày chi tiết trong phần thiết kế dữ liệu

Hệ thống các thành phần lưu trữ cùng với quan hệ giữa chúng. Ý niệm về thành phần lưu trữ cùng với quan hệ giữa các thành phần này cũng sẽ được trình bày chi tiết trong phần thiết kế dữ liệu

4. Phương pháp thiết kế phần mềm

Tùy thuộc vào qui trình được chọn khi thực hiện phần mềm, việc thiết kế có thể được tiến hành theo 2 phương pháp chính:

- Phương pháp trực tiếp
- Phương pháp gián tiếp

Phương pháp trực tiếp được áp dụng khi thực hiện phần mềm không thông qua giai đoạn phân tích. Với phương pháp này việc thiết kế sẽ nhận kết quả chuyển giao trực tiếp từ giai đoạn xác định yêu cầu. Mô hình phần mềm sẽ được xây dựng trực tiếp từ các yêu cầu. Cách tiếp cận này sẽ rất khó khăn cho người thực hiện với các phần mềm có qui mô lớn (nhiều yêu cầu, yêu cầu phức tạp. v.v).

Với phương pháp trực tiếp, thiết kế phần mềm là quá trình cho phép chuyển đổi từ các yêu cầu (kết quả giai đoạn xác định yêu cầu) đến mô hình phần mềm tương ứng. Mục tiêu chính của việc thiết kế là mô tả các thành phần của phần mềm (thành phần giao diện, thành phần xử lý, thành phần dữ liệu) tương ứng với các yêu cầu của phần mềm (yêu cầu chức năng nghiệp vụ, yêu cầu chức năng hệ thống, yêu cầu phi chức năng.

Phương pháp gián tiếp được áp dụng với các qui trình có giai đoạn phân tích. Với phương pháp này việc thiết kế sẽ chỉ nhận một phần các kết quả chuyển giao trực tiếp từ giai đoạn xác định yêu cầu, phần chính yếu sẽ được nhận gián tiếp qua giai đoạn phân tích.

Mô hình phần mềm sẽ được xây dưng tương ứng theo các mô hình trong giai đoạn phân tích. Cách tiếp cận này sẽ rất thuận lợi trong đa số trường hợp với các phần mềm qui mô lớn.

Với phương pháp gián tiếp, thiết kế phần mềm là quá trình cho phép chuyển từ mô hình thế giới thực (kết quả giai đoạn phân tích) đến mô hình phần mềm tương ứng. Mục tiêu chính của việc thiết kế là mô tả các thành phần của phần mềm (thành phần giao diện, thành phần xử lý, thành phần dữ liệu) tương ứng với các mô hình của thế giới thực (mô hình xử lý, mô hình dữ liệu).

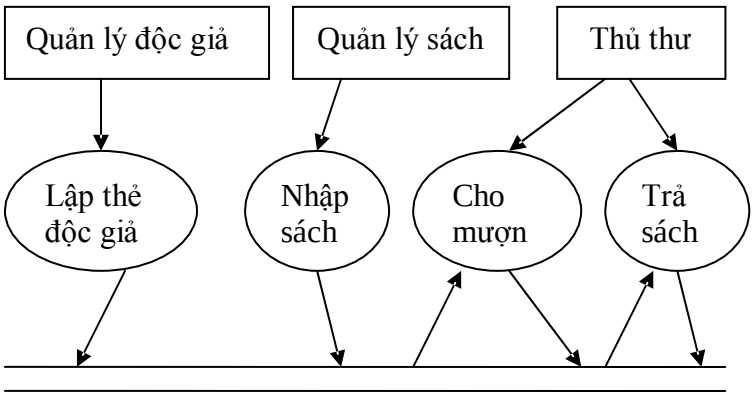
5. Ví dụ minh họa

Các ví dụ sau đây chỉ nhằm **minh họa** quá trình thiết kế phần mềm sau khi thực hiện giai đoạn mô hình hóa yêu cầu, các kết quả chỉ chú trọng chủ yếu **tính đúng đắn** và bỏ qua các yêu cầu chất lượng khác (tiến hóa, hiệu quả, tiện dụng). Kết quả thực tế khi xem xét đầy đủ các yêu cầu chất lượng là quá phức tạp và không thích hợp cho việc min họa

Ví dụ 1: Xét phần mềm quản lý thư viện với 4 yêu cầu

- 1. Lập thẻ đọc giả
- 2. Nhận sách
- 3. Cho mượn sách
- 4. Trả sách

a. Mô hình hóa các yêu cầu



b.Thiết kế phần mềm

Hệ thống các màn hình giao diện

③ Màn hình chính:

- Nộidung:
  + Thông tin về thư viện
  + Thông tin về các độc giả trong thư viện
  + Thông tin về các sách trong thư viện
- Thao tác người dùng
  + Tra cứu và chọn độc giả
  + Tra cứu và chọn sách

③ Màn hình Lập thẻ

- Nộidung:
  + Thông tin về thẻ độc giả
- Thao tác người dùng
  + Nhập thông tin về thẻ
  + Yêu cầu lập thẻ

③ Màn hình Cho mượn sách:

- Nộidung:
  + Thông tin về thẻ độc giả

+ Ngày mượn sách

+ Danh sách các sách muốn mượn

- Thao tác người dùng

+ Nhập thông tin về việc cho mượn sách

+ Yêu cầu cho mượn sách

③ Màn hình Nhận sách:

- Nộidung:

+ Ngày nhận sách

+ Danh sách các sách nhận cùng thông tin liên quan

- Thao tác người dùng

+ Nhập thông tin về việc cho nhận sách

+ Yêu cầu cho nhận sách

③ Màn hình Trả sách:

- Nộidung:

+ Ngày trả sách

+ Thông tin về việc trả sách

- Thao tác người dùng

+ Nhập thông tin về việc trả sách

+ Yêu cầu trả sách

c. Hệ thống các hàm xử lý

③ Hàm Lập thẻ: Kiểm tra tính hợp lệ và ghi nhận thẻ trên bộ nhớ phụ

③ Hàm Tra cứu độc giả: Tìm thẻ độc giả theo các tiêu chuẩn khác nhau để cho phép cập nhật hay xóa thẻ

③ Hàm Xóa thẻ: Xóa thẻ trên bộ nhớ phụ

③ Hàm Nhập sách: Kiểm tra tính hợp lệ của sách và ghi nhận sách trên bộ nhớ phụ

③ Hàm Xóa sách: Xóa sách trên bộ nhớ phụ

③ Hàm Cho mượn sách: Kiểm tra tính hợp lệ của việc cho mượn sách và ghi nhận các thông tin cho mượn sách trên bộ nhớ phụ

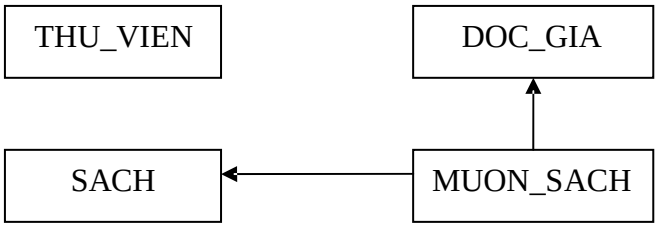
③ Hàm Tra cứu sách: Tìm sách theo các tiêu chuẩn khác nhau để cho phép cập nhật hay xóa sách

③ Hàm Tính số sách độc giả đang mượng: Tính số lượng sách độc giả đang mượn và chưa trả

- Hàm Kiểm tra độc giả có sách mượn quá hạn: Kiểm tra độc giả có sách mượn quá hạn và trả về 1 nếu đúng, 0 nếu sai
- Hàm Kiểm tra tình trạng sách: Kiểm tra sách đang được mượn trả về 1 nếu đúng , 0 nếu sai
- Hàm Tra cứu phiếu cho mượn sách: Tra cứu các phiếu mượn sách theo nhiều tiêu chuẩn để cập nhật hay xó phiếu cho mượn
- Hàm Xóa phiếu cho mượn sách: Xóa thông tin về việc cho mượn sách trên bộ nhớ phụ
- Hàm Trả sách: Ghi nhận việc trả sách trên bộ nhớ phụ
- Hàm Tính tiền phạt: Tính tiền phạt khi độc giả trả sách trễ hạn

d. Hệ thống các bảng dữ liệu:

- **Sơ đồ logic**

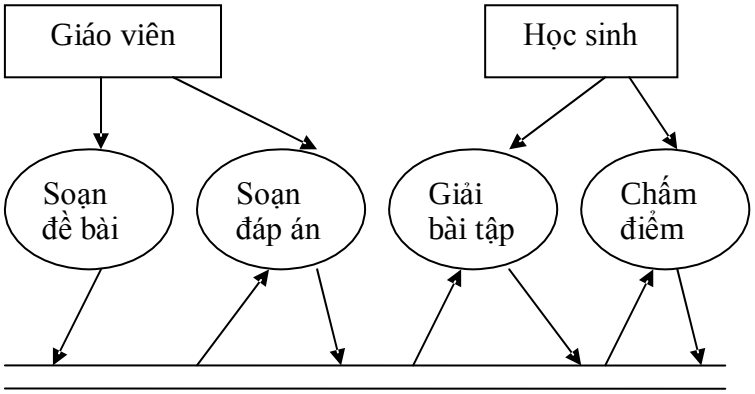


- Bảng THU_VIEN: các thông tin về thư viện
- Bảng DOC_GIA: Các thông tin về độc giả
- Bảng SACH: Các thông tin về sách
- Bảng MUON_SACH: Các thông tin về mượn trả sách

Ví dụ 2: Xét phần mềm hỗ trợ giải bài tập phương trình đại số với 4 yêu cầu

- 1. Soạn đề bài
- 2. Soạn đáp án
- 3. Giải bài tập
- 4. Chấm điểm

a. Mô hình hóa yêu cầu



b. Thiết kế phần mềm

③ Màn hình chính:

- Nộidung:
  + Thông tin về sách bài tập
  + Thông tin về các bài tập của sách
- Thao tác người dùng: Tra cứu và chọn bài tập

③ Màn hình Soạn đề bài:

- Nộidung:
  + Thông tin về đề bài
- Thao tác người dùng
  + Nhập thông tin về đề bài
  + Yêu cầu phát sinh đề
  + Yêu cầu ghi nhận đề

③ Màn hình Soạn đáp án:

- Nộidung:
  + Thông tin về đáp án
- Thao tác người dùng
  + Nhập thông tin về đáp án
  + Yêu cầu ghi nhận đáp án

③ Màn hình Nhận bài giải:

- Nộidung:
  + Thông tin về bài giải
- Thao tác người dùng
  + Nhập thông tin về bài giải
  + Yêu cầu ghi nhận bài giải

+ Yêu cầu chấm điểm

③ Màn hình Chấm điểm:

- Nộidung:
  + Thông tin về bài giải
  + Thông tin về việc chấm điểm
  + Thông tin về đáp án
- Thao tác người dùng
  + Xem thông tin điểm
  + Yêu cầu xem đáp án

c. Hệ thống các hàm xử lý:

③ Hàm Soạn thảo đề bài: Ghi nhận đề bài của giáo viên trên bộ nhớ phụ (giới hạn không kiểm tra tính hợp lệ của đề bài)

③ Hàm Tra cứu bài tập: Tìm kiến bài tập theo nhiều tiêu chuẩn khác nhau để có thể cập nhật, xóa hay soạn đáp án

③ Hàm Xóa bài tập : Xóa bài tập trên bộ nhớ phụ

③ Hàm Soạn đáp án: Kiểm tra tính hợp lệ của đáp án của giáo viên và ghi nhận đáp án trên bộ nhớ phụ

③ Hàm Xóa đáp án: Xóa bài tập trên bộ nhớ phụ

③ Hàm Ghi nhận bài giải: Kiểm tra tính hợp lệ bài giải của học sinh và ghi nhận bài giải trên bộ nhớ phụ

③ Hàm Biến đổi: Biến đổi một biểu thức thành một đa thức

③ Hàm Khai triển: Nhân đa thức

③ Hàm Rút gọn: Cộng 2 đa thức

③ Hàm so sánh: So sành đa thức

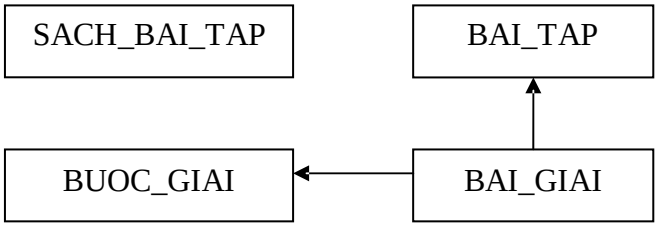
③ Hàm Xóa bài giải: Xóa bài giải của học sinh trên bộ nhớ phụ

③ Hàm Chấm điểm: Tính điểm số bài giải của học sinh

③ Hàm Xem đáp án: Trình bày các bước giải của đáp án cho học sinh xem

d. Hệ thống lưu trữ

③ Sơ đồ logic



- ③ Bảng SACH_BAI_TAP: các thông tin về sách bài tập
- ③ Bảng BAI_TAP: Các thông tin về các bài tập của sách
- ③ Bảng BUOC_GIAI: Các thông tin về các bước giải trong một bài giải
- ③ Bảng BAI_GIAI: Các thông tin về đáp án và các bài giải của một bài tập

Chương 4: THIẾT KẾ DỮ LIỆU

1. Tổng quan

Mục tiêu chính của thiết kế dữ liệu là mô tả cách thức tổ chức lưu trữ các dữ liệu của phần mềm. Có hai dạng lưu trữ chính mà người thiết kế cần phải cân nhắc và lựa chọn.

- Lưu trữ dưới dạng tập tin
- Lưu trữ dưới dạng cơ sở dữ liệu

Lưu trữ dưới dạng tập tin thường chỉ thích hợp với một số phần mềm đặc thù (cờ tướng, trò chơi, v.v.) đặc điểm chung của các phần mềm này là chú trọng rất nhiều vào xử lý, hình thức giao diện và không chú trọng nhiều đến việc lưu trữ lại các thông tin được tiếp nhận trong quá trình sử dụng phần mềm (thông thường các thông tin này được tiếp nhận và xử lý ngay).

Cách tiếp cận dùng cơ sở dữ liệu rất thông dụng và giáo trình này sẽ giới hạn trình bày chi tiết các phương pháp kỹ thuật liên quan đến việc tổ chức lưu trữ dữ liệu dùng cơ sở dữ liệu quan hệ. Giáo trình này sẽ không nhắc lại các khái niệm cơ bản về cơ sở dữ liệu và giả sử rằng người xem đã biết qua các khái niệm này. Tuy nhiên chúng ta cũng nên xem lại các bước để hình thành nên mô hình dữ liệu quan hệ trong quá trình thiết kế dữ liệu

2. Kết quả của thiết kế

Cách thức tổ chức lưu trữ dữ liệu của phần mềm được mô tả thông qua 2 loại thông tin sau:

- Thông tin tổng quát

  Cung cấp góc nhìn tổng quan về các thành phần lưu trữ

  - Danh sách các bảng dữ liệu: Việc lưu trữ cần sử dụng bao nhiêu bảng dữ liệu và đó là các bảng nào ?
  - Danh sách các liên kết: Các bảng dữ liệu có quan hệ, có mối liên kết giữa chúng ra sao?

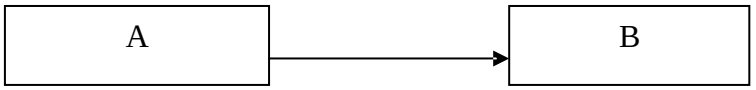
- Thông tin chi tiết:
  - Danh sách các thuộc tính của từng thành phần: Các thông tin cần lưu trữ của thành phần ?
  - Danh sách các Miền giá trị toàn vẹn: Các qui định về tính hợp lệ của các thông tin được lưu trữ

Có nhiều phương pháp, nhiều đề nghị khác nhau về việc mô tả các thông tin trên. Giáo trình này chọn sơ đồ logic để biểu diễn các thông tin tổng quát và bảng thuộc tính. Miền giá trị để mô tả chi tiết các thành phần trong sơ đồ logic.

Sơ đồ logic là sơ đồ cho phép thể hiện hệ thống các bảng dữ liệu cùng với quan hệ mối nối liên kết giữa chúng. Các ký hiệu được dùng trong sơ đồ rất đơn giản như sau:

Bảng: hình chủ nhật

Liên kết: (xác định duy nhất một): Mũi tên



Mũi tên hình trên có ngữ nghĩa: 1 phần tử A sẽ xác định duy nhất 1 phần tử B, ngược lại 1 phần tử B có thể tương ứng với nhiều phần tử A.

Ví dụ: Với phần mềm quản lý thư viện có sơ đồ logic sau:

Theo sơ đồ này việc lưu trữ các dữ liệu của phần mềm quản lý thư viện được tổ chức 3 bảng (DOCGIA, MUONSACH, SACH) vùng với 2 liên kết giữa chúng

Tất nhiên sơ đồ trên chỉ là một trong các cách thức tổ chức lưu trữ dữ liệu còn nhiều cách khác có thể có. Chi tiết các cách này sẽ được trình bày trong phương pháp thiết kế cơ sở dữ liệu.

Bảng thuộc tính cho phép mô tả chi tiết thành phần trong sơ đồ logic theo dạng như sau:

- ③ Thành phần
- ③ Ý nghĩa

STT	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1					
2					
…					

Bảng miền giá trị cho phép mô tả các Miền giá trị giữa các thuộc tính cùng một thành phần hay nhiều thành phần khác nhau.

MSố	Mô tả miền giá trị	Thành phần liên quan	Ghi chú
RB1			
RB2			
…			

Ví dụ:

Ghi chú:

- Bảng thuộc tính cho phép mô tả chi tiết thanh phần cần lưu trữ và sẽ được dùng trong báo cáo về thiết kế dữ liệu của phần mềm. Tuy nhiên cách mô tả trên khá dài dòng, trong giáo trình này sẽ sử dụng một dạng trình bày cô đọng hơn theo dạng lược đồ quan hệ. Với dạng trình bày này gồm tên bảng và thuộc tính đi kèm, các thuộc tính khóa được gạch chân.

Ví dụ:

DOC_GIA(MDG,Hoten,Loaidg,Ngsinh, Nglapthe, Diachi)

SACH(MSACH,Tensach,Theloai, NgNhap, Tacgia, Nhaxb, Namxb)

MUON(MDG,MSACH,NgMuon,Ngtra)

3. Quá trình thiết kế

Có 2 cách tiếp cận chính để thiết kế dữ liệu:

- **Phương pháp trực tiếp:**

Từ các yêu cầu đã xác định, tạo lập trực tiếp sơ đồ logic cùng với bảng thuộc tính, bảng miền giá trị. Các tiếp cận này rất khó thực hiện đối với sơ đồ logic phức tạp.

- **Phương pháp gián tiếp:**

Từ các yêu cầu đã xác định, tạo lập mô hình quan niệm dữ liệu, và sau đó đưa vào mô hình này sẽ tạo lập sơ đồ logic, bảng thuộc tính, bảng miền giá trị. Các tiếp cận này dễ thực hiện hơn vì mô hình quan niệm dữ liệu thường đơn giản (chứa các thành phần dữ liệu bản chất nhất của phần mềm). Khái niệm chi tiết về mô hình quan niệm dữ liệu cùng với các bước cụ thể sẽ được trình bày chi tiếp trong phần sau.

Tương ứng với 3 yêu cầu của phần mềm, quá trình thiết kế dữ liệu bao gồm 3 bước lớn:

- Thiết kế với tính đúng đắn
- Thiết kế với yêu cầu chất lượng
- Thiết kế với yêu cầu hệ thống

③ Thiết kế với tính đúng đắn

- Đảm bảo đầy đủ và chính xác về mặt ngữ nghĩa các thông tin liên quan đến các công việc trong yêu cầu.
- Các thông tin phục vụ cho các yêu cầu chất lượng sẽ không được xét đến trong bước thiết kế này.

③ Thiết kế với yêu cầu chất lượng

- Vẫn đảm bảo tínhh đúng đắn nhưng thỏa mãn thêm các yêu cầu chất lượng khác (tiến hóa, tốc độ nhanh, lưu trữ tối ưu).
- Cần chú ý bảo đảm tính đúng đắn khi cải tiến sơ đồ logic.

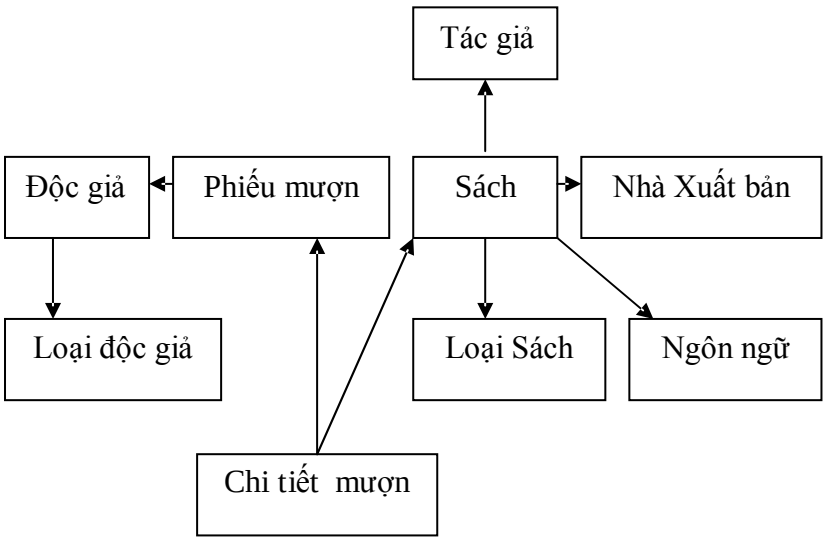
③ Thiết kế với yêu cầu hệ thống

- Vẫn đảm bảo tính đúng đắn và các yêu cầu chất lượng khác nhưng thỏa mãn thêm các yêu cầu hệ thống (phân quyền, cấu hình phần cứng, môi trường phần mềm, v.v)

Ví dụ: phần mềm quản lý thư viện:

Với phương pháp trực tiếp sẽ cho kết qủa như sau:

Sơ đồ logic:



Các bảng thuộc tính:

DOC_GIA(MDG,MLDG,HoTen,NgaySinh,DiaChi,DienThoai)

SACH(MSACH,MTG,MNXB,MLSACH,MNN,TenSach, Ngàymua, SoTrang)

PHIEU_MUON(MPHM, NgayMuon)

CHITIETMUON(MPHM, MSACH, NgayTra)

LOAISACH(MLSACH,TenLS,GhiChu)

LOAIDOCGIA(MLDG,TenLDG,GhiChu)

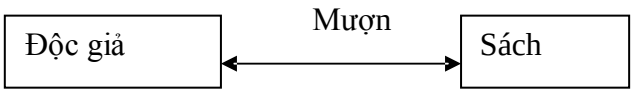
NHAXB(MNXB,TenNXB, GhiChu)

TACGIA(MTG,Ten, Ghichu)

NGONNGU(MNN,Ten,Ghichu)

Với phương pháp gián tiếp, ngoài kết qủa cuối cùng tương tự như phương pháp trực tiếp, còn có kết quả trung gian là mô hình quan niệm dữ liệu như sau:

+ Sơ đồ lớp đối tượng với 2 đối tượng chính Sách, Độc giả và 1 quan hệ Mượn giữa 2 lớp đối tượng trên

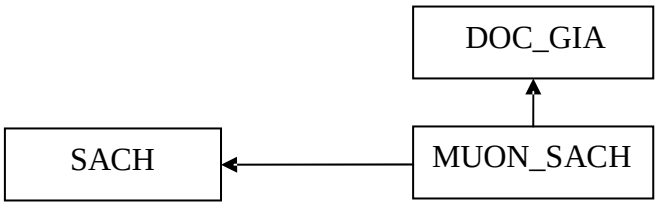


+ Mô hình chi tiết các thành phần trong sơ đồ lớp: **Xem chi tiết ở phụ lục B**

Ví dụ : Xét phần mềm với 4 yêu cầu: Lập thẻ độc giả, Nhận sách, Cho mượn sách, Trả sách

Thiết kế dữ liệu với tính đúng đắn

Sơ đồ logic



Chi tiết các bảng

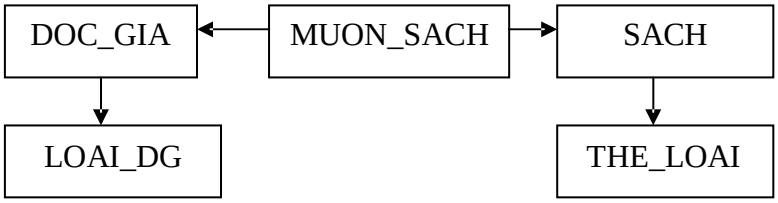
DOCGIA**(**MDG,MLDG,HoTen,NgaySinh,DiaChi,DienThoai)

SACH(MSACH,MTG,MNXB,MLSACH,MNN,TenSach, Ngàymua, SoTrang)

MUON_SACH(MDG,MSACH,NgMuon,NgTra,Tienphat)

Thiết kế dữ liệu với tính tiến hóa

Sơ đồ logic



Chi tiết các bảng:

DOC_GIA**(**MDG,MLDG,HoTen,NgaySinh,DiaChi,DienThoaiNg_lapthe,Ng_hethan)

SACH(MSACH,Tensach,MTL,ng_Nhap, Tacgia,NamXB, NhaXB)

MUON_SACH(MDG,MSACH,NgMuon,NgTra,Tienphat

THE_LOAI(MTL,Tentheloai,GhiChu)

LOAI_DG(MLDG,TenLDG,GhiChu)

Thiết kế với tính hiệu quả (truy xuất nhanh)

Sơ đồ logic

Cũng với sơ đồ logic như trên nhưng ta có các bảng thuộc tính:

DOC_GIA**(**MDG,MLDG,HoTen,NgaySinh,DiaChi,DienThoaiNg_lapthe,Ng_hethan, SosachMuon, TinhTrangtra)

SACH(MSACH,Tensach,MTL,ng_Nhap, Tacgia,NamXB, NhaXB, TinhTrangMuon)

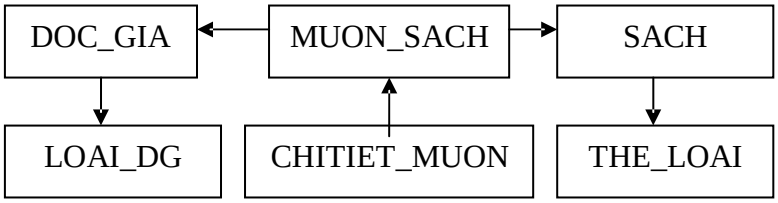
MUON_SACH(MDG,MSACH,NgMuon,NgTra,Tienphat)

THE_LOAI(MTL,Tentheloai,GhiChu)

LOAI_DG(MLDG,TenLDG,GhiChu)

Thiết kế dữ liệu với tính hiệu quả (lưu trữ tối ưu)

Sơ đồ logic



Chi tiết các bảng thuộc tính

DOC_GIA**(**MDG,MLDG,HoTen,NgaySinh,DiaChi,DienThoaiNg_lapthe,Ng_hethan, SosachMuon, TinhTrangtra)

SACH(MSACH,Tensach,MTL,ng_Nhap, Tacgia,NamXB, NhaXB, TinhTrangMuon)

MUON_SACH(MDG,MSACH,NgMuon,NgTra,Tienphat)

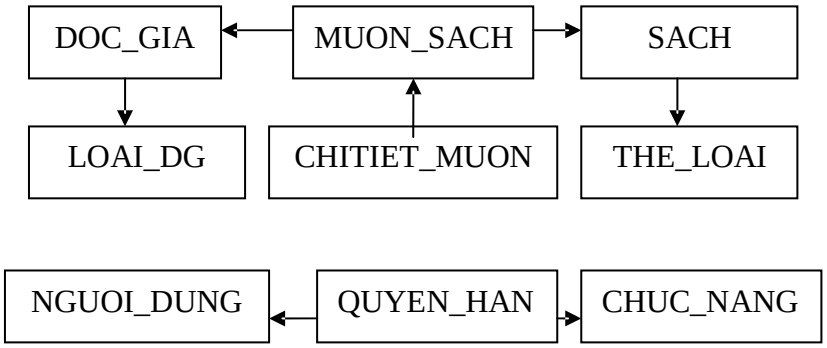
CHITIET_MUON(MMUON,MSACH,NgTra,Tienphat)

THE_LOAI(MTL,Tentheloai,GhiChu)

LOAI_DG(MLDG,TenLDG,GhiChu)

Thiết kế dữ liệu với yêu cầu phân quyền hệ thống (phân quyền)

Sơ đồ logic



Chi tiết các bảng

DOC_GIA(MDG,MLDG,HoTen,NgaySinh,DiaChi,DienThoaiNg_lapthe,Ng_hethan, SosachMuon, TinhTrangtra)

SACH(MSACH,Tensach,MTL,ng_Nhap, Tacgia,NamXB, NhaXB, TinhTrangMuon)

MUON_SACH(MDG,MSACH,NgMuon,NgTra,Tienphat)

CHITIET_MUON(MMUON,MSACH,NgTra,Tienphat)

THE_LOAI(MTL,Tentheloai,GhiChu)

LOAI_DG(MLDG,TenLDG,GhiChu)

NGUOI_DUNG(MND,HoTen, Ghichu)

CHUC_NANG(MCN,Ten_Chucnang, Ghichu)

QUYEN_HAN(MND,MCN)

4. Phương pháp thiết kế dữ liệu

4.1 Phương pháp trực tiếp

Bước 1:

- Lập sơ đồ với 1 thành phần duy nhất
- Đánh giá tính đúng đắn so với các yêu cầu và chuyển sang bước 2 nếu cần thiết

Bước 2:

- Tách 1 số thuộc tính để tạo ra các thành phần mới

- Xác định liên kết giữa các thành phần
- Đánh giá tính đúng đắn so với các yêu cầu và lặp lại bước 2 nếu cần thiết

Ví dụ: phần mềm quản lý thư viện

Cách 1: Chỉ dùng 1 thành phần SÁCH

Masach, Ten, Theloai, Ngaymua, Tacgia, NhaXB, NamXB

HotenDG, LoaiDG, Ngaylamthe, Ngaymuon, Ngaytra

Cách 2: Dùng 2 thành phần SACH,DOCGIA

Cách 2.1 : Chỉ lưu trừ lần mượn sách cuối cùng

SACH

MSACH, MADG, Ten, Theloai, NgayMua, TacGia, NhaXB, NamXB, Ngaymuon, NgayTra.

DOCGIA

MDG, HoTen, LoaiDG,Ngaylamthe,

Cách 2.2: Chỉ cho phép độc giả mượn tối đa 1 quyển sách

SACH

MSACH, Ten, Theloai, NgayMua, TacGia, NhaXB, NamXB, Ngaymuon, NgayTra.

DOCGIA

MDG, MSACH,HoTen, LoaiDG,Ngaylamthe, Ngaymuon

Cách 3: Dùng 3 thành phần SACH,DOCGIA, MUONSACH

SACH

MSACH, Ten, Theloai, NgayMua, TacGia, NhaXB, NamXB, Ngaymuon, NgayTra.

DOCGIA

MDG, HoTen, LoaiDG,Ngaylamthe,

MUONSACH

Mmuon,MDG,MSACH, Ngaymuon, Ngaytra

Ví dụ: Phần mềm quản lý học sinh

Cách 1: Dùng 1 thành phần HOCSINH

HOCSINH

MAHS, HoTen, Ngaysinh, GioiTinh, Lop, Monhoc, LoaiKT, HocKy,Diem, Ngayvang, Lydo

Cách 2: Dùng 3 bảng HOCSINH, KIEMTRA, DIEMDANH

HOCSINH

MAHS, Hoten, Ngaysinh, GioiTinh, Lop

KIEMTRA

MAKT,MAHS, Monhoc,LoaiKT,Hocky, Diem

DIEMDANH

MADD,MAHS,Ngayvang, Lydo

4.2 Phương pháp gián tiếp

Bước 1:

- Lập sơ đồ lớp
- Xác định các lớp đối tượng
- Xác định quan hệ giữa các lớp đối tượng và lập sơ đồ

Bước 2:

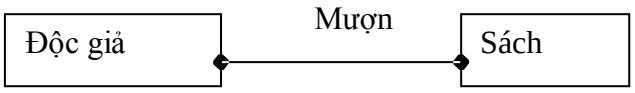
- Ánh xạ từ sơ đồ lớp vào sơ đồ logic
- Ánh xạ các lớp đối tượng
- Ánh xạ các quan hệ giữa các lớp đối tượng

Bước 3:

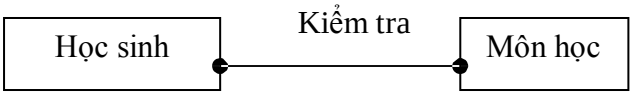
- Hoàn chỉnh sơ đồ logic
- Bổ sung các thành phần theo yêu cầu
- Mô tả chi tiết các thuộc tính của các thành phần

4.2.1 Lập sơ đồ lớp

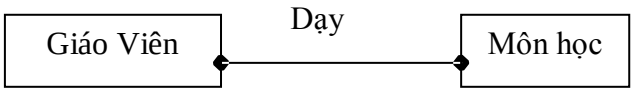
Ví dụ: Với phần mềm quản lý thư viện 2 đối tượng chính là Độc giả, Sách và quan hệ giữa chúng là quan hệ mượn sách



Với phần mềm quản lý học sinh trường phổ thông trung học 2 đối tượng chính là Học sinh, Môn học và quan hệ giữa chúng là quan hệ kiểm tra



Với phần mềm xếp thời khóa biểu trường trung học phổ thông 2 đối tượng chính là Giáo Viên, Môn học và quan hệ giữa chúng là quan hệ dạy.



4.2.2 Ánh xạ sơ đồ lớp

Ánh xạ lớp đối tượng. Mỗi đối tượng trong sơ đồ lớp tương ứng với 1 thành phần trong sơ đồ logic

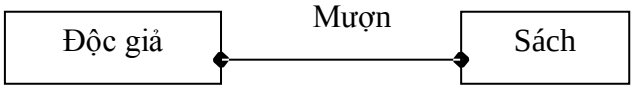
Sơ đồ lớp: Độc giả Sách

Sơ đồ logic: Độc giả Sách

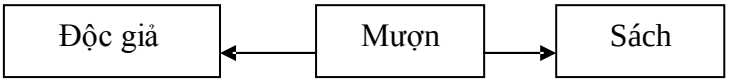
4.2.3 Ánh xạ quan hệ

- Quan hệ 1-n: Quan hệ 1-n trong sơ đồ lớp giữa 2 lớp đối tượng A,B (1 A nhiều B) tương ứng với liên kết xácđịnh duy nhất vừ A sang B trong sơ đồ logic.
- Quan hệ m-n: Quan hệ m-n C trong sơ đồ lớp giữa 2 lớp đối tượng A,B tương ứng với 1 thành phần C trong sơ đồ logic. Thành phần này có liên hệ xác định duy nhất A,B.

Sơ đồ lớp:



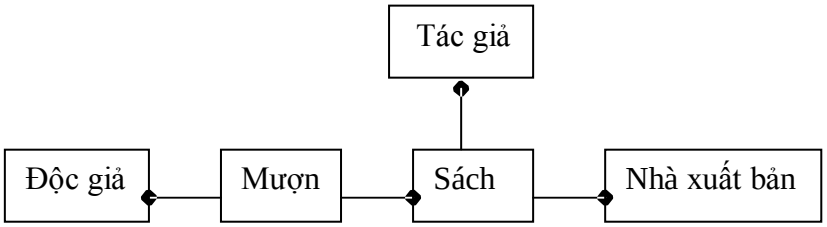
Sơ đồ logic:



4.2.4 Hoàn chỉnh sơ đồ logic

1. Bổ sung các thành phần

+ Đối tượng phụ: Mỗi đối tượng phụ tương ứng với 1 thành phần trong sơ đồ logic
+ Các thành phần khác: Xem xét lại tính đúng đắn và bổ sung thêm nếu cần thiết



2. Mô tả chi tiết thuộc tính các thành phần

+ Thuộc tính khóa chính:

- Mỗi thành phần ứng với đối tượng (chính, phụ) cần 1 thuộc tính khóa riêng)

- Các thành phần còn lại, tùy theo ý nghĩa sử dụng sẽ có thuộc tính khóa riêng hay dùng tổ hợp thuộc tính khóa của các thành phần khác

Ví dụ: Các thành phần Độcgiả, Sách, Nhà xuất bản, Tác giả sẽ có thuộc tính khóa chính tương ứng là MDG, MSACH, MNXB, MTG.

Thành phần mượn cũng sẽ có khóa chính là MMUON (không dùng tổ hợp các thuộc tính khóa ngoại được ?)

+ Thuộc tính khóa ngoại:

- Thể hiện đúng liên kết giữa các thành phần trong sơ đồ logic: nếu A xác định duy nhất B thì A có thuộc tính là khoá chính của B (đó là khóa ngoại của A)

Ví dụ:

Thành phần Mượn có 2 khóa ngoại: MDG,MSACH

Thành phần Sách có 2 khoá ngoại: MNXB, MTG, MDG

+ Các thuộc tính khác:

Dựa vào yêu cầu lưu trữ, chú ý các loại thuộc tính sau:

- Định danh: Tên
- Loại: Sự phân loại
- Thời gian: Ngày tháng
- Không gian: vị trí
- Định lượng: độ đo, tính chất, v.v.v

Ví dụ: Độc giả sẽ có thuộc tính khác như:

HoTen (định danh)

LoaiDG (loại)

Ngaysinh (thời gian)

Ngayhethan (thời gian)

Diachi (không gian)

Sách sẽ có thuộc tính khác như:

TenSach (định danh)

LoaiSach (loại)

NgayMua (thời gian)

GiaTien (định lượng)

5. Thiết kế dữ liệu với tính đúng đắn

③ Các bước thực hiện:

Bước 1: Chọn một yêu cầu và xác định sơ đồ logic cho yêu cầu đó

Bước 2: Bổ sung thêm một yêu cầu và xem xét lại sơ đồ logic

+ Nếu sơ đồ logic vẫn đáp ứng được thì tiếp tục bước 3

+ Nếu sơ đồ logic không đáp ứng được thì bổ sung vào sơ đồ thuộc tính mới (ưu tiên 1) hoặc thành phần mới (ưu tiên 2) cùng với các thuộc tính và liên kết tương ứng

Bước 3: Quay lại bước 2 cho đến khi đã xem xét đầy đủ các yêu cầu

Ghi chú:

- Với mỗi yêu cầu cần xác định rõ cần lưu trữ các thông tin gì? dựa vào luồng dữ liệu đọc/ghi trong sơ đồ luồng dữ liệu tương ứng) và tìm cách bổ sung các thuộc tính để lưu trữ các thông tin này
- Chỉ xem xét tính đúng đắn
- Cần chọn các yêu cầu theo thứ tự từ đơn giản đến phức tạp (thông thường yêu cầu tra cứu là đơn giản nhất)
- Với yêu cầu phức tạp có thể phải bổ sung vào sơ đồ logic nhiều thành phần mới

Khóa của các thành phần phải dựa trên ngữ nghĩa tương ứng trong thế giới thực

6. Thiết kế dữ liệu và yêu cầu chất lượng

⑨ **Mục tiêu**

Xem xét đánh giá sơ đồ logic theo các yêu cầu về chất lượng và tiến hành cập nhật lại sơ đồ để bảo đảm các tiêu chuẩn về chất lượng. Ngoài tính đúng đắn cần ưu tiên hàng đầu xem xét sự hơn kém nhau giữa các phần mềm chính là mức độ thỏa mãn các tiêu chuẩn chất lượng còn lại (đặc biệt là tính tiến hóa).

6.1 Xem xét tính tiến hóa

Để bảo đảm tính tiến hóa, sơ đồ logic sẽ còn bổ sung cập nhật lại nhiều thành phần qua các bước thiết kế chi tiết. Trong các bước đầu tiên là thiết kế dữ liệu, chúng sẽ giới hạn xem xét đến các thuộc tính có giá trị rời rạc.

Thuộc tính có giá trị rời rạc là các thuộc tính mà miền giá trị chỉ bao gồm một số giá trị nhất định. Các giá trị này thông thường thuộc về tập hợp có độ biến động rất ít trong quá trình sử dụng phần mềm.

Ví dụ:

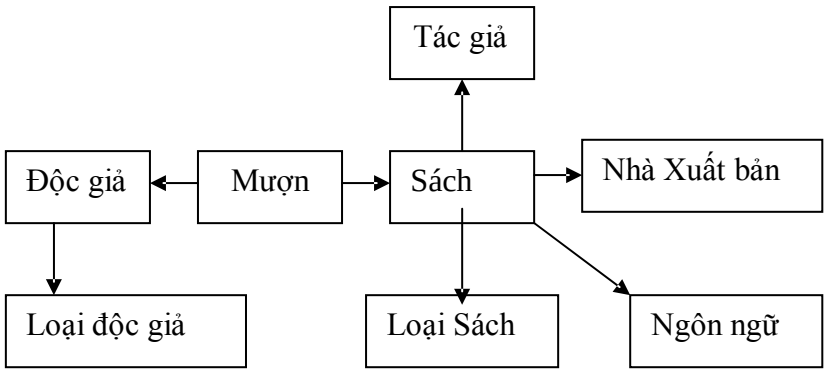
LOAIDG (thành phần độc giả): Thư viện hiện tại chỉ có 3 loại độc giả là ‘A’, ‘B’,’C’ và khả năng có thêm loại độc giả mới rất thấp.

Ngôn ngữ (thành phần Sách): Các sách trong thư viện hiện tại chỉ có 3 loại ngôn ngữ ‘Việt’, ‘Anh’, ‘Pháp’ và khả năng thêm sách thuộc ngôn ngữ mới rất thấp.

Tuy nhiên cần lưu ý rằng khả năng biến động trên tập hợp giá trị của thuộc tính rời rạc là thấp nhưng không phải là không có. Và khi xảy ra biến động (thêm loại độc giả, thêm sách thuộc ngôn ngữ mới) nếu không chuẩn bị trước trong thiết kế thì người dùng sẽ không thể khai báo được các biến động này với phần mềm, và do đó có thể một số chức năng sẽ không thực hiện được (ví dụ khong thể thêm sách mới với ngôn ngữ tiếng Hoa).

Để chuẩn bị tốt cho biến động về sau (nếu có) trong tập hợp các giá trị của thuộc tính rời rạc. Chúng ta sẽ tách các thuộc tính này thành một thành phần trong sơ đồ logic. Khi đó người dùng trong quá trình sử dụng hoàn toàn có thể cập nhật lại tập hợp các giá trị này tương ứng với các biến động thực tế trong thế giới thực.

Sơ đồ logic khi tách các thuộc tính rời rạc như sau:



6.2 Xem xét tính hiệu quả (tốc độ)

③ Phạm vi xem xét:

- Chỉ giới hạn xem xét việc tăng tốc độ thực hiện của phần mềm bằng cách bổ sung thêm các thuộc tính vào các bảng dùng lưu trữ các thông tin đã tính toán trước (theo qui tắc nào đó từ các thông tin gốc đã được lưu trữ)

  Ví dụ: số sách đang mượn của độc giả

- Các thông tin này phải được tự động cập nhật khi có bất kỳ thay đổi thông tin gốc liên quan

  Ví dụ độc giả mượn thêm hoặc trả sách

  Học sinh có thêm cột điểm

③ Các bước tiến hành:

- Bước 1: Chọn một yêu cầu và xem xét cần bổ sung thông tin gì trên bộ nhớ phụ để tăng tốc độ thực hiện của xử lý liên quan (các thông tin xử lý phải đọc mà không cần thực hiện việc tính toán)

- Bước 2: Quay lại bước 1 cho đến khi đã xem xét đầy đủ các yêu cầu

Ghi chú:

- Sau mỗi bước nhất thiết phải lập bảng danh sách các thuộc tính tính toán cùng với thông tin liên quan

  + Thông tin gốc

  + Xử lý tự động cập nhật thông tin gốc (chi tiết về các xử lý này sẽ được mô tả trong phần thiết kế xử lý)

- Nếu thông tin gốc thường xuyên bị thay đổi, việc bổ sung thuộc tính tính toán để tăng tốc độ thực hiện sẽ mất ý nghĩa (thậm chí theo chiều ngược lại)

- Việc tăng tốc độ truy xuất có thể sẽ dẫn đến việc lưu trữ không tối ưu
- Thứ tự xem xét các yêu cầu theo thứ tự từ đầu đến cuối (không cần chọn như các bước trong thiết kế dữ liệu)

  Ví dụ: Phần mềm quản lý giải vô địhc bóng đá quốc gia với bảng thuộc tính tính toán

  Thuộc tính: Tong_ban_thang, Tong_the_phat, Diem_so là những thuộc tính có thể xử lý tự động cập nhật

6.3 Xem xét tính hiệu quả (lưu trữ)

Tính hiệu quả trong thiết kế dữ liệu sẽ được xem xét dưới góc độ lưu trữ tối ưu. Vấn đề đặt ra là xây dựng sơ đồ logic sao cho vẫn bảo đảm lưu trữ đầy đủ thông tin theo yêu cầu nhưng với dung lượng lưu trữ nhỏ nhất có thể có. Vấn đề này đặc biệt quan trọng với các phần mềm với hệ thống lưu trữ lớn và nhiều phát sinh thông tin cần lưu trữ theo thời gian.

Khi đó cần đặc biệt quan tâm đến các thành phần mà dữ liệu tương ứng được phát sinh nhiều theo thời gian. Chúng ta sẽ tìm cách bố trí lại sơ đồ logic sao cho vẫn đảm bảo thông tin mà dung lượng lưu trữ lại ít hơn.

③ Các bước tiến hành:

Bước 1:Lập danh sách các bảng cần được xem xét để tối ưu hóa việc lưu trữ

- Xem xét và xác định các công việc có tần suất thực hiện thường xuyên và bổ sung vào danh sách chọn các bảng được sử dụng tương ứng của công việc này
- Xem xét các bảng mà khóa của bảng bao gồm nhiều thuộc tính và bổ sung bảng này vào danh sách được chọn

Bước 2: Tối ưu hóa việc lưu trữ các bảng có khối lượng dữ liệu lưu trữ lớn thông qua việc tối ưu hóa lưu trữ từng thuộc tính trong bảng

- Xác định các thuộc tính mà việc lưu trữ chưa tối ưu. Ưu tiên xem xét các thuộc tính có kiểu chuỗi
- Tối ưu hóa việc lưu trữ tùy theo từng trường hợp cụ thể
- Một trong các trường hợp thông dụng nhất là chuỗi có kích thước lớn và giá trị được sử dụng nhiều lần trong các mẫu tin khác nhau (ví dụ: thuộc tính tác giả, Nha_xb trong bảng SACH của phần mềm quản lý sách)

- Với trường hợp trên việc tối ưu hoá có thể thực hiện thông qua việc bổ sung các bảng mới (bảng TAC_GIA, NHA_XB) và tổ chức cấu trúc bảng SACH (thay thuộc tính TAC_GIA bằng MTG, thay thuộc tính NHA_XB bằng MNXB)

Bước 3: Tối ưu hóa các bảng mà khóa của bảng bao gồm nhiều thuộc tính.

Phân rã bảng đang xét thành hai bảng. Trong đó, một bảng chứa các thuộc tính mà giá trị được lặp lại nhiều lần trong cùng một lần thực hiện công việc tương ứng trong thế giới thực. Bảng này cần có khóa riêng (sẽ được bảng còn lại sử dụng để tham chiếu đến)

Ghi chú:

- Việc phân rã giúp cho lưu trữ tối ưu tuy nhiên:
  - Tốc độ truy xuất có thể sẽ chậm hơn
  - Việc thực hiện xử lý khó khăn hơn (thuật giải phức tạp hơn)
- Cần cân nhắc trước khi thực hiện phân rã
- Việc đánh giá khóa riêng cho bảng đã phân ra có thể cần kiểm tra thêm số phụ thuộc hàm

Ví dụ minh họa: Phần mềm quản lý bán sách

Bước 1: Các bảng cần xem xét

NHAP_SACH(MSACH, Ng_Nhap, So_luong, Don_gia, Thanh_tien)

HOA_DON(MHD,MSACH, Khach_hang, Ng_lap_hd, So_Luong, Don_gia, Thanh_tien)

Bước 2:

- Bổ sung bảng KHACH_HANG

KHACH_HANG(MKH,Ho_ten,Ghi_chu)

- Tổ chức lại bảng HOA_DON

HOA_DON (MHD,MSACH,MKH,Ng_lap_hd,So_luong, Don_gia, Thanh_tien)

Bước 3:

- Phân rã bảng NHAP_SACH thành 2 bảng NHAP_SACH, CT_NHAP

NHAP_SACH(MNHAP,Ng_Nhap)

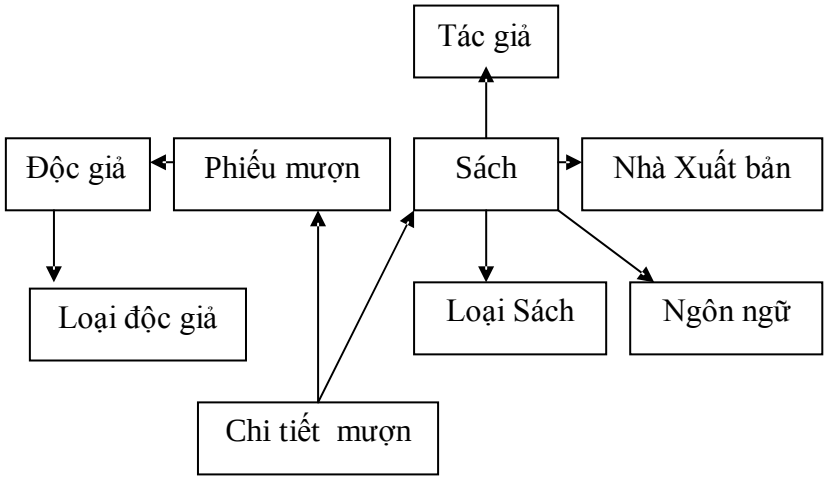
CT_NHAP(MNHAP,MSACH,So_luong, Don_gia, Thanh_tien)

- Phân rã bảng HOA_DON thành 2 bảng HOA_DON, CT_HOA_DON

HOA_DON(MHD,MKH, Ng_lap_hd)

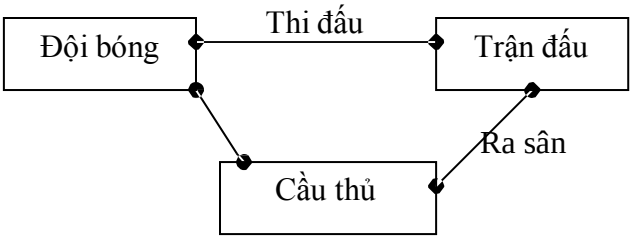
CT_HD(MHD, MSACH,So_luong, Don_gia, Thanh_tien)

Ví dụ: Xét phần mềm quản lý thư viện.



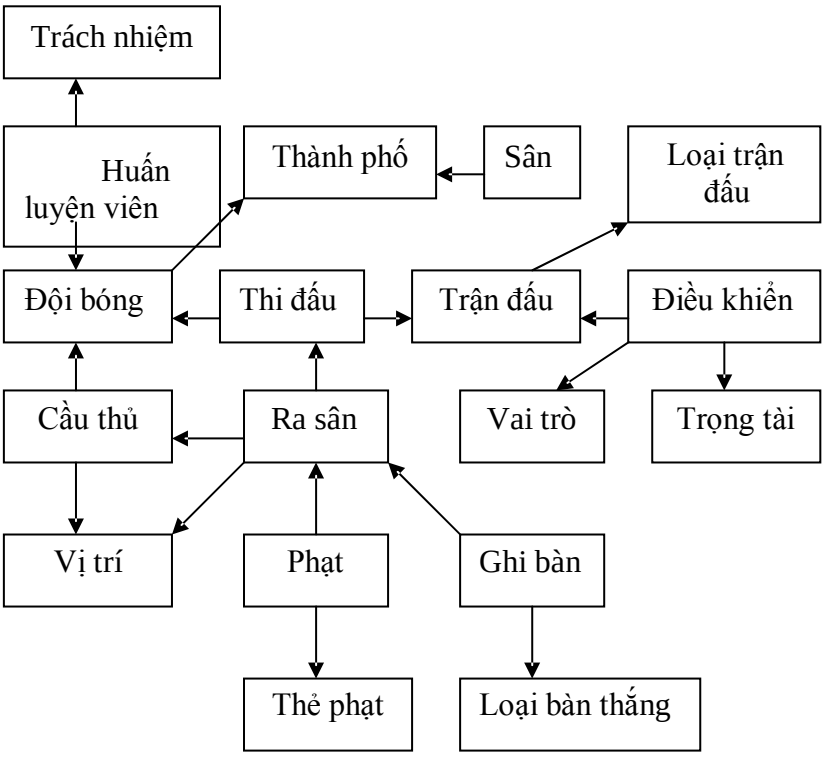
Ví dụ: Xét phần mềm quản lý giải bóng đá

Sơ đồ lớp



Mô tả chi tiết các thuộc tính: **Xem chi tiết phụ lục B**

- **Sơ đồ logic**



- **Mô tả chi tiết thuộc tính:** Xem chi tiết phụ lục B

Chương 5 : THIẾT KẾ GIAO DIỆN

1. Tổng quan

Chương này đi sâu tìm hiểu cách thức thiết kế giao diện là công đoạn không kém phần quan trọng trong quá trình làm phần mềm, đây cũng có thể xem là công đoạn phác thảo đồ hình hay prototype cho phần mềm và để sau đó nhận phản hồi yêu cầu của khách hàng đối với chương trình và để người thiết kế có thể điều chỉnh theo yêu cầu đề ra. Tùy theo mục đích yêu cầu, theo độ phức tạp của chương trình, người thiết kế giao diện có thể làm theo các những thiết kế sau và kết qủa thiết kế tương ứng. Thiết kế giao diện phải nắm bắt các điều chính yếu sau:

1. Hồ sơ cá nhân người dùng: Biết họ là ai, mục đích của người dùng là gì, Kỹ năng và kinh nghiệm của người dùng, nhu cầu của họ
2. Mượn những ứng xử từ những hệ thống quen thuộc đối với người dùng.
3. Cho người dùng thấy rõ các chức năng một cách sẵn sàng
4. ứng xử của chương trình từ trong ra ngoài phải kết dính, gắn kết
5. Thay đổi trong ứng xử nên tương phản với diện mạo của chương trình
6. Shortcut: Cung cấp cả cách thức cụ thể và tóm tắt tác vụ được làm
7. Tính tập trung: Một số giao diện GUI được tập trung chú ý nhiều hơn
8. Ngữ pháp: thông qua giao diện biết số luật thao tác
9. Trợ giúp, độ an toàn, hạn chế ngữ cảnh người dùng, giao diện đẹp,…

1.1 Kết quả thiết kế

③ Màn hình giao diện:

Màn hình giao diện là một trong các hình thức giao tiếp giữa người sử dụng và phần mềm khi họ thực hiện các công việc của mình trên máy tính. Mục tiêu chính của thiết kế giao diện là mô tả hệ thống các màn hình giao diện này

③ Kết quả thiết kế giao diện: bao gồm 2 phần

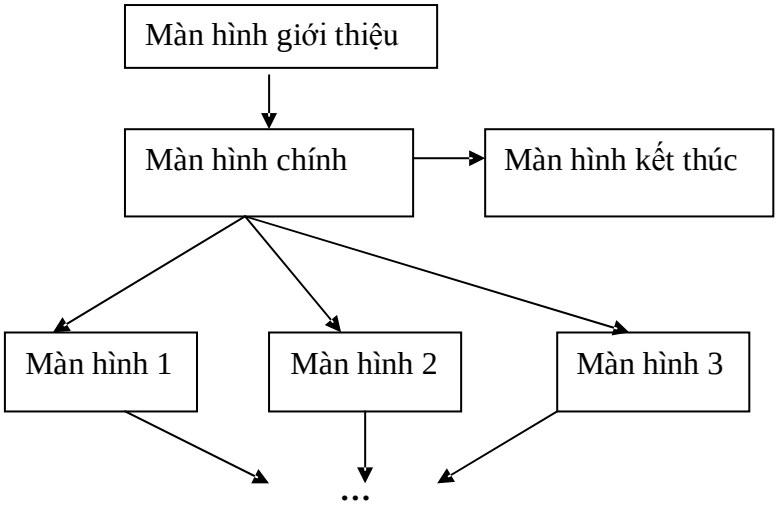
- Sơ đồ màn hình: Mô tả các thông tin tổng quát về hệ thống các màn hình cùng với quan hệ về việc chuyển điều khiển giữa chúng
- Mô tả chi tiết từng màn hình: Mô tả chi tiết nội dung, hình thức trình bày và các thao tác mà người dùng có thể thực hiện trên từng màn hình

Ví dụ: Liệt kê các phần sau: màn hình, ý nghĩa sử dụng

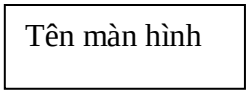
Danh sách các thao tác có thể thực hiện

STT	Thao tác	Ý nghĩa	Xử lý liên quan	Ghi chú
1				
2				

③ Sơ đồ màn hình



Ký hiệu:



Màn hình với tên tương ứng

Chuyển điều khiển đến màn hình khác

③ Mô tả màn hình giao diện:

Các thông tin cần mô tả môt màn hình giao diện bao gồm

- Tên màn hình:

Tên của công việc tương ứng muốn thực hiện trên máy tính.

- Nôi dung:

Cấu trúc các thành phần bên trong màn hình. Các thành phần này có thể chia làm 2 loại: Thanh phần dữ liệu, thành phần xử lý.

+ Thành phần dữ liệu

Các thông tin liên quan đến công việc đang xét như sau:

a. Thông tin nhập liệu:loại thông tin người dùng chịu trách nhiệm cung cấp giá trị (ngày lập, hóa đơn, hàng bán,..) có thể là nhập liệu trực tiếp, nhập liệu với giá trị định sẵn(có thể sửa nếu muốn) hoặc chọn trong danh sách có trước.

b. Thông tin kết xuất: loại thông tin này phần mềm chịu trách nhiệm cung cấp giá trị (ví dụ lượng hàn tồn hiện nay, tổng tiền trả,…).

+Thành phần xử lý:

Các nút điều khiển cho phép người dùng yêu cầu phần mềm thực hiện một xử lý nào đó.

+ Hình thức trình bày:

Cách thức bố trí sắp xếp các thành phần trong màn hình (ví trí, màu sắc, kích thước,…)

Với màn hình có biểu mẫu liên quan, tốt nhất là trình bày đúng với biểu mẫu tương ứng hoặc trình bày đúng như yêu cầu của khách hàng. Tuy nhiên cần lưu ý trong trường hợp biểu mẫu liên quan chỉ là kết quả cuối cùng cần ghi nhận trước khi đạt đến kết quả đó cần phải thực hiện một số công việc trung gian không có biểu mẫu rõ ràng. Với những trường hợp này cần bổ sung, sáng tạo hình thức trình bày các màn hình trung gian thể hiện các công việc trung gian.

Với màn hình không có biểu mẫu liên quan hình thức trình bày màn hình hoàn toàn là sự sáng tạo khi thiết kế.

+ Các thao tác có thể thực hiện

Mô tả hệ thống các thao tác mà người dùng có thể thực hiện trên màn hình cùng với ý nghĩa của chúng. Có rất nhiều loại thao tác khác nhau có thể cung cấp cho người dùng trên một màn hình giao diện, tuy nhiên giáo trình này chỉ giới hạn xem xét việc mô tả thao tác khi người dùng nhấn vào nút điều khiển hay nút lệnh hoặc kết thúc việc nhập liệu tại một thành phần nhập liệu nào đó.

1.2 Phân loại màn hình giao diện

Quá trình sử dụng phần mềm bao gồm các bước sau:

- Chọn công việc muốn thực hiện trên máy tính.
- Cung cấp các thông tin cần thiết tương ứng với công việc đã chọn.
- Yêu cầu phần mềm thực hiện.
- Xem xét kết quả thực hiện.

Dựa trên quá trình trên các màn hình giao diện có thể được chia thành nhiều loại tùy theo ý nghĩa sử dụng.

- **Màn hình chính**: Cho phép người dùng sử dụng chọn lựa công việc mong muốn thực hiện trên máy tính từ danh sách các công việc

- **Màn hình nhập liệu** lưu trữ: Cho phép người dùng thực hiện lưu trữ các thông tin được phát sinh trong thế giới thực.
- **Màn hình nhập liệu** xử lý: Cho phép người sử dụng cung cấp các thông tin cần thiết cho việc thực hiện một công việc nào đó
- **Màn hình kết quả**: Trình bày cho người sử dụng các kết quả việc thực hiện của một công việc nào đó
- **Màn hình thông báo**: Thông báo, nhắc nhở người sử dụng trong quá trình thực hiện một công việc nào đó
- **Màn hình tra cứu**: Cho phép tìm kiếm thông tin đã được lưu trữ với các tiêu chuẩn tìm kiếm

Một màn hình giao diện có thể thuộc một trong các loại trên hay cũng có thể tích hợp từ nhiều màn hình cơ sở thuộc vào các loại trên tùy theo bản chất công việc liên quan.

Trong thực tế còn có rất nhiều màn hình khác, tuy nhiên giáo trình chỉ giới hạn xem xét chủ yếu đến các loại màn hình đã trình bày phía trên, giáo trình sẽ chú trọng trình bày chi tiết 3 loại màn hình quan trọng và thông dụng nhất: màn hình chính, màn hình tra cứu, màn hình nhập liệu lưu trữ.

1.3 Quá trình thiết kế

Qui trình chung: Dựa trên yêu cầu chức năng, đầu tiên người thiết kế sẽ xem xét thiết kế các giao diện từ tính đúng đắn, đến tính tiện dụng thõa yêu cầu về tiện dụng, và xét đến tính hiệu quả nếu yêu cầu về hiệu quả được đưa ra và một số thiết kế theo yêu cầu khác, v.v

☙ Thiết kế giao diện với tính đúng đắn

③ Sơ đồ màn hình

Giả sử cần thực hiện n công việc trên máy tính, sơ đồ màn hình trong trường hợp này chỉ bao gồm n+1 màn hình sau:

+ Một màn hình chính cho phép chọn công việc

+ n màn hình liên quan trực tiếp đến n công việc muốn thực hiện

③ Mô tả chi tiết từng màn hình

1. Màn hình chính:

Xác định chính xác nội dung dựa trên danh sách các công việc được yêu cầu và chọn hình thức trình bày đơn giản nhất.

Ví dụ 1: Phần mềm thự viện

Màn hình chính	
1. Cho mượn sách	8. Lập báo cáo về độc giả
2. Nhận trả sách	9.Nhận sách mới
3.Tìm sách	10.Thanh lý sách
4.Lập báo cáo mượn trả	11.Lập báo cáo sách
5.Lập thẻ độc giả	12.Thay đổi qui đinh tổ chức
6.Gian hạn thẻ độc giả	13.Thay đổi quy định mượn trả
7.Tìm độc giả	14. Thoát

Đây là thiết kế cho ứng dụng chạy độc lập có thể hiển thị tất cả danh sách các màn hình, còn đối với ứng dụng lập trình mạng web có thể tùy theo quyền hạn sử dụng màn hình chính hạn chế bởi các màn hình tương tác cho người sử dụng đó.

Ví dụ 2: Phần mềm quản lý học sinh cấp 3

Màn hình chính	
1. Tiếp nhận hồ sơ	8. Thay đổi qui đinh tổ chức
2. Xếp lớp	9. Thay đổi qui đinh xếp loại
3.Tìm học sinh	10.Thoát
4. Nhận bảng điểm danh	
5.Nhận bảng điểm kiểm tra	
6.Xếp loại học sinh	
7.Lập báo cáo tổng kết	

Ví dụ 3: Phần mềm quản lý giải bóng đá

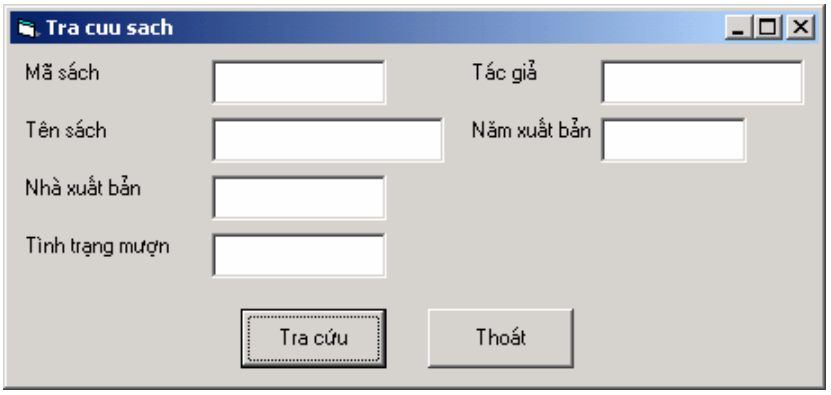
Màn hình chính

1. Tiếp nhận hồ sơ đội bóng
2. Xếp lịch thi đấu
3. Phân công trọng tài
4. Ghi nhận kết quả thi đấu
5. Lập bảng xếp hạng tạm thời
6. Tra cứu cầu thủ
7. Lập báo cáo tổng kết giải
8. Thay đổi qui đinh tổ chức
9. Thay đổi qui đinh xếp hạng
10. Thoát

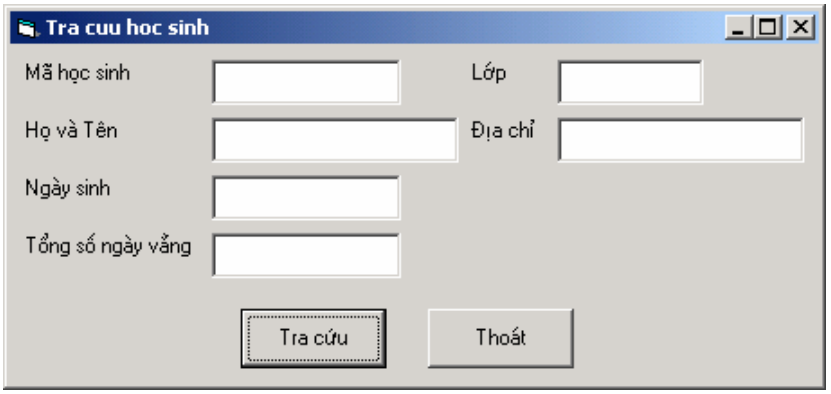
2. Màn hình tra cứu

Chọn tiêu chuẩn tra cứu đơn giản nhất (chỉ có mã số) và kết quả tìm kiếm đơn giản (cho biết có hay không có mã số trên).

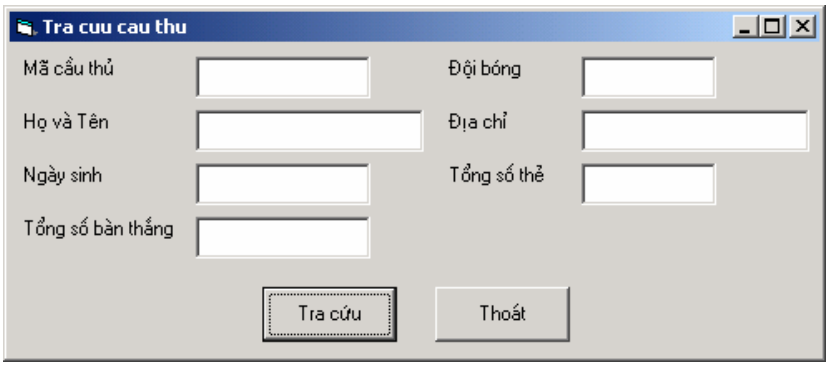
Ví dụ 1: Tra cứu sách với phần mềm quản lý thư viện



Ví dụ 2: Tra cứu học sinh với phần mềm quản lý học sinh



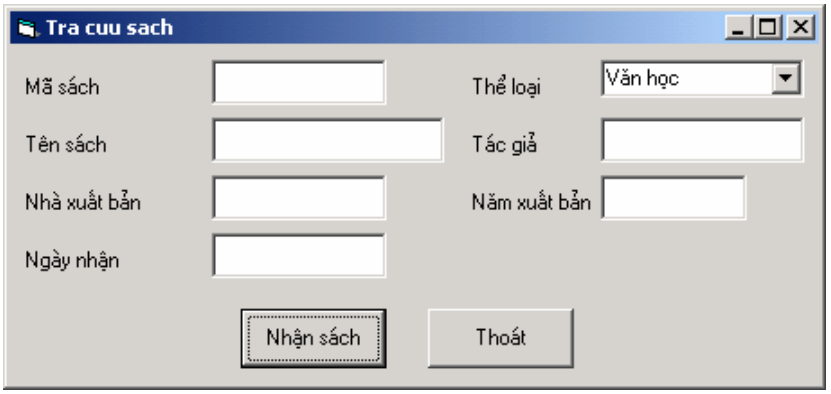
Ví dụ 3: Tra cứu cầu thủ với phần mềm quản lý giải bóng đá

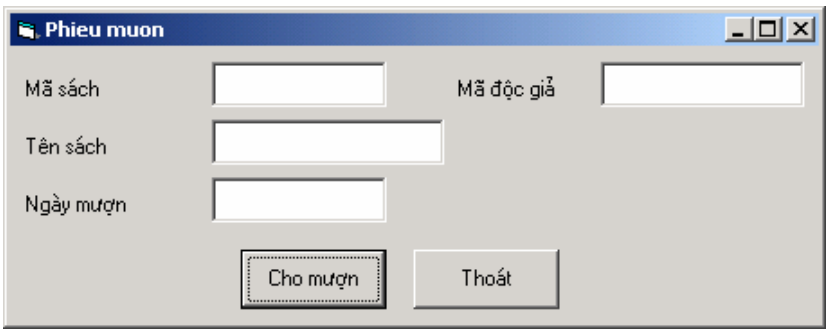


3. Màn hình nhập liệu

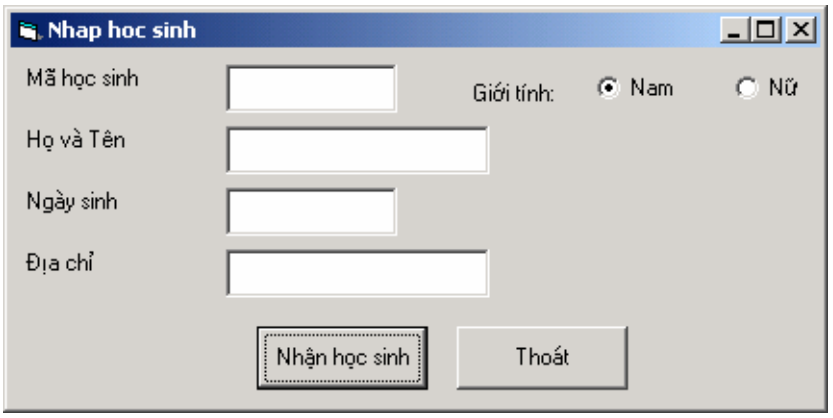
Xác định chính xác nội dung dựa trên biểu mẫu hoặc thông tin liên quan đến công việc tương ứng và chọn hình thức trình bày đơn giản nhất có thể có (liệt kê tuần tự các nội dung)

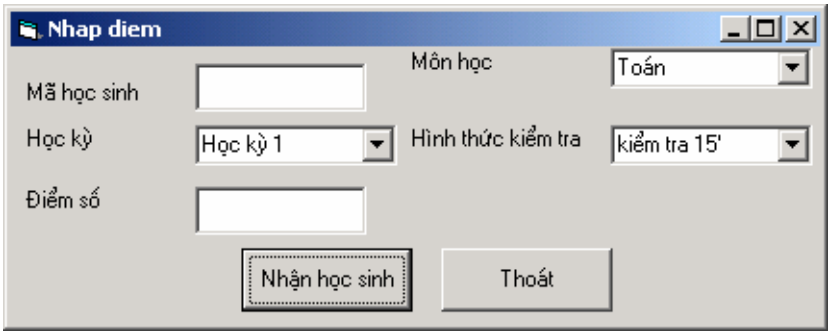
Ví dụ: Nhập sách, mượn sách với quản lý thư viện



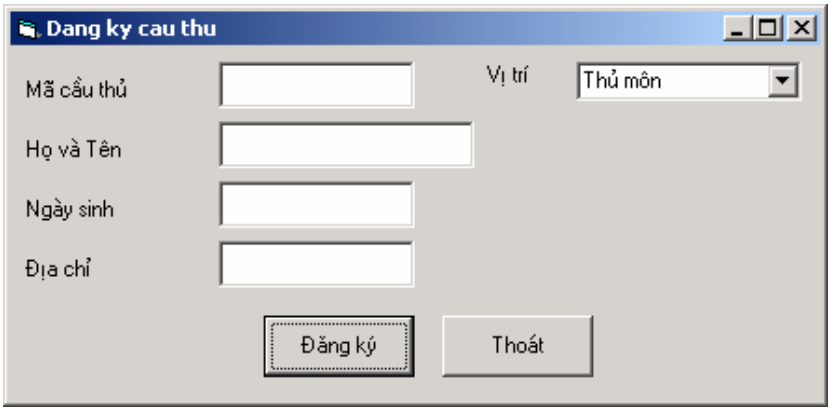


Ví dụ: Nhập học sinh, điểm số với phần mềm quản lý học sinh





Ví dụ: Đăng ký cầu thủ với phần mềm quản lý giải bóng đá



☙ **Thiết kế giao diện với tính tiện dụng**

③ **Sơ đồ màn hình**

Bổ sung vào sơ đồ các màn hình công việc trung gian giúp cho việc sử dụng các màn hình công việc chính dễ dàng hơn, tự nhiên hơn.

③ **Mô tả chi tiết từng màn hình**

1. Màn hình chính

Phân chia các công việc theo từng nhóm tùy theo ý nghĩa và chọn hình thức trình bày tự nhiên nhất có thể có (menu, sơ đồ,…)

Ví dụ 1: Màn hình chính phần mềm quản lý giải bóng đá. Có các nút điều khiển sau:

Tổ chức: Sân, Trọng tài, Loại thẻ phạt, Loại bàn thắng, Qui chế tổ chức.

Kế hoạch: Đăng ký đội bóng, Xếp lịch thi đấu, Phân trọng tài

Thi đấu: Ghi nhận kết quả, tra cứu cầu thủ, xếp loại tạm thời

Tổng kết: Xếp hạng chính thức, Lập báo cáo tổng kết

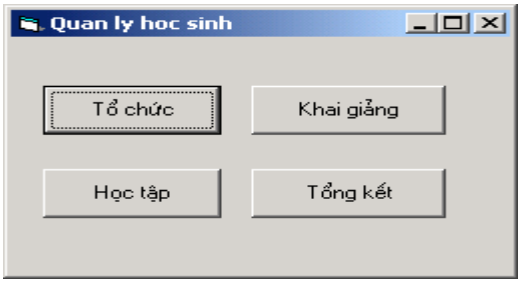
Ví dụ 2: Màn hình quản lý học sinh. Có các nút điều khiển sau

Tổ chức: Học kỳ, Lớp, Môn học, hình thức kiểm tra, danh hiệu, qui định chung, Thoát

Khai giảng: Tiếp nhận hồ sơ, Xếp lớp

Học tập: Điểm danh, Bảng điểm, tra cứu học sinh

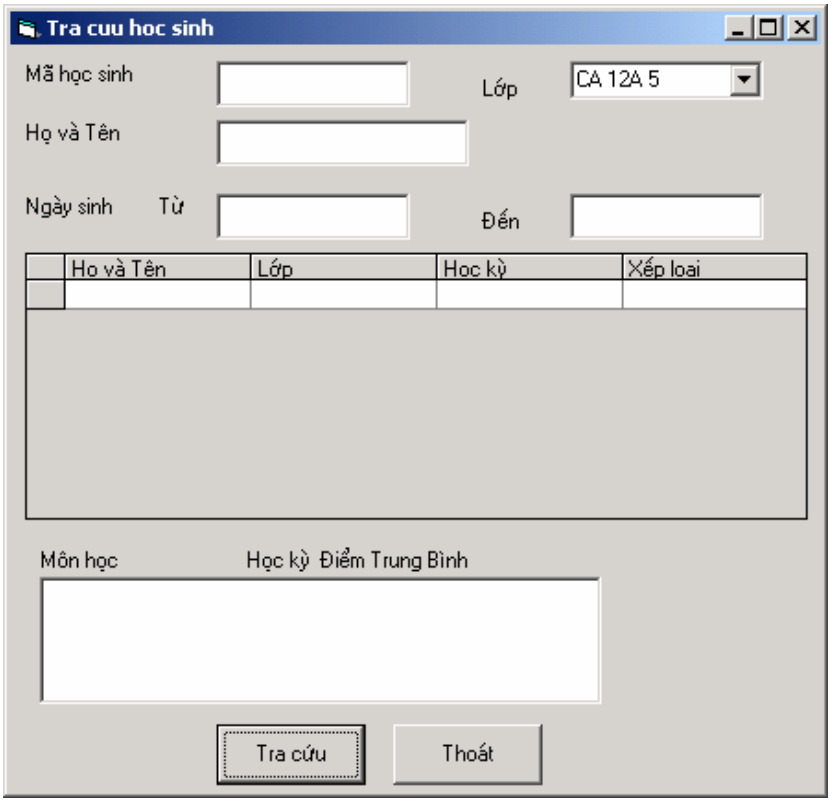
Tổng kết: Xếp loại học sinh, Báo cáo tổng hợp



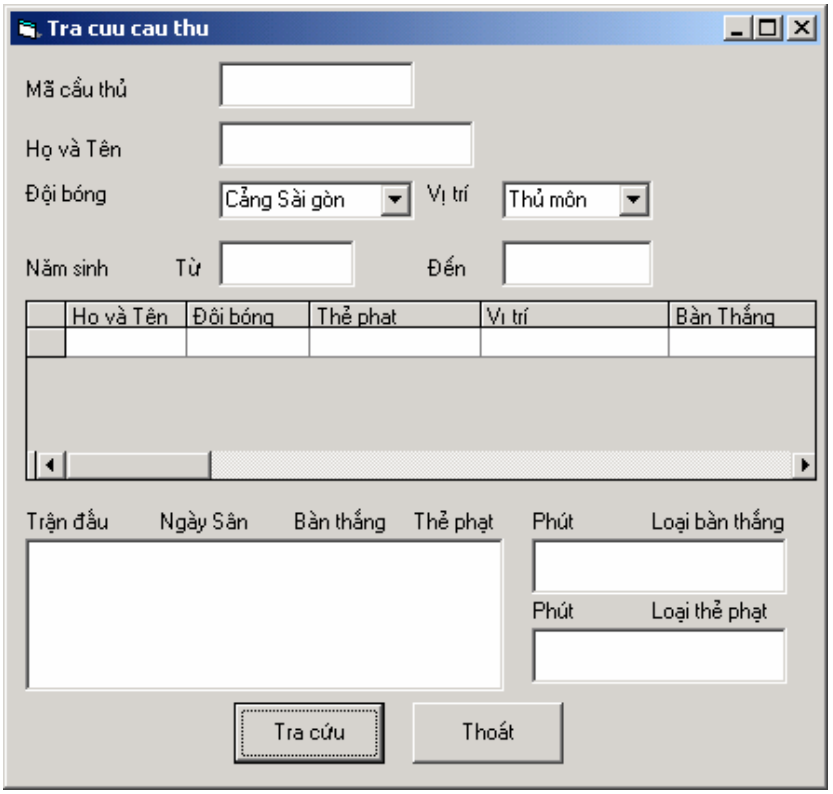
2. Màn hình tra cứu

Mở rộng các tiêu chuẩn tra cứu (các thông tin khác về đối tượng cần tìm). Mở rộng kết quả tìm kiếm (các thông tin liên quan đến đối tượng khi tìm thấy). Cho phép người dung xem các kết quả tra cứu dưới nhiều hình thức trình bày khác nhau (các thứ tự khác nhau với một danh sách, các dạng thể hiện biểu đồ, hình ảnh, v.v.)

Ví dụ 1: Tra cứu học sinh



Ví dụ 2: Tra cứu cầu thủ



3. Màn hình nhập liệu

Chọn dạng trình bày là biểu mẫu liên quan (nếu có) và bổ sung vào đó các thông tin giúp việc sử dụng thuận tiện hơn. Nếu không có biểu mẫu liên quan, cố gắng thiết kế hình thức trình bày tự nhiên nhất có thể có.

2. Thiết kế màn hình

2.1 Mô tả màn hình chính

③ **Ý nghĩa sử dụng:**

Màn hình chính là màn hình cho phép ngươi dùng chọn được công việc mà họ muốn thực hiện với phần mềm. Thông thường mỗi phần mềm chỉ có một màn hình chính duy nhất.

③ **Nội dung:** Danh sách các công việc có thể thực hiện với phần mềm

③ **Hình thức trình bày**

Phím nóng: Hình thức này cho phép chọn nhanh một công việc cần thực hiện đối với người sử dụng chuyên nghiệp. Thông thường không được sử dụng riêng rẻ mà phải kết hợp với các hình thức khác.

Thực đơn: nhóm từng công việc theo chức năng (ví dụ lưu trữ, kết xuất). Đây là dạng sử dụng thông dụng nhất.

Biểu tượng: công việc thể trực quan qua biểu tượng (ký hiệu hay hình ảnh tượng trưng cho công việc. Tương tự như phím nóng nhưng thông dụng hơn và thường kết hợp với các hình thức khác.

Sơ đồ: Dùng sơ đồ để hiển thị trực quan các đối tượng chính, được thể hiện qua các thao tác trực tiếp trên sơ đồ.

Tích hợp: Sử dụng đồng thời nhiều hình thức, thông thường hình thức thực đơn sẽ được ưu tiên trước và kết hợp với nhiều hình thức khác.

③ **Thao tác người dùng**

Trên màn hình này thao tác chính của ngừoi dùng là chọn công việc trong danh sách các công việc được đưa ra bởi phần mềm.

2.2 Thiết kế màn hình chính dùng thực đơn (menu)

③ **Tổ chức của thực đơn**

Thực đơn bao gồm nhiều nhóm chức năng (tương ứng nhóm các công việc) mỗi nhóm chức năng bao gồm nhiều chức năng, mỗi chức năng tương ứng với một công việc.

③ **Phân loại thực đơn: có 3 loại**

- Thực đơn hướng chức năng: Các nhóm chức năng tương ứng với các loại yêu cầu

Ví dụ:

Tổ chức: các công việc liên quan đến tổ chức

Lưu trữ: Các công việc liên quan đến lưu trữ

Tra cứu: Các công việc liên quan đến tra cứu tìm kiếm

- Thực đơn hướng đối tượng: Các nhóm chức năng tương ứng với các lớp đối tượng. Với sơ đồ lớp gồm n lớp đối tượng, thực đơn sẽ bao gồm (n+1) nhóm chức năng. Trong đó:

+ Một nhóm chức năng tương ứng với đối tượng thế giới thực.

+ n nhóm chức năng tương ứng n lớp đối tượng.

- Thực đơn hướng qui trình: Các nhóm chức năng tương ứng với các giai đoạn trng hoạt động của thế giới thực. Thông thường thế giới thực bao gồm các giai đoạn sau như Tổ chức, Kế hoạch, Tiếp nhận, Hoạt động, Tổng kết.

3. Thiết kế màn hình tra cứu

3.1 Mô tả màn hình tra cứu

③ **Ý nghĩa sử dụng**

Màn hình tra cứu là màn hình cho phép người dùng tìm kiếm và xem các thông tin về các đối tượng.

③ **Nội dung**

+ Tiêu chuẩn tra cứu: Các thông tin được sử dụng cho việc tìm kiếm (thông thường là các thuộc tính).

+ Kết quả tra cứu: Cho biết có tìm thấy hay không. Các thông tin cơ bản về đối tượng tìm kiếm (các thuộc tính). Các thông tin về quá trình hoạt động của đối tượng (quan hệ với các đối tượng khác).

③ **Hình thức trình bày**

+Tiêu chuẩn tra cứu: Biểu thức logic, Cây, tích hợp

+ Kết quả tra cứu: Thông báo, danh sách đơn, xâu các danh sách, cây danh sách.

③ **Thao tác người dùng:** Nhập các giá trị cho các tiêu chuẩn tra cứu, yêu cầu bắt đầu tra cứu, xem chi tiết các kết quả tra cứu.

3.2 Thể hiện tiêu chuẩn tra cứu

3.2.1 Tra cứu với biểu thức logic

Tiêu chuẩn tra cứu được thể hiện dưới dạng một biểu thức logic có dạng như sau:

<Biểu thức logic>=<biểu thức cơ sở>

Phép toán logic …

Phép toán AND, OR, NOT, phép so sánh

3.2.2 Tra cứu với hình thức cây

Tiêu chuẩn tra cứu được thể hiện qua cây mà các nút chính là các bộ phận trong tổ chức của thế giới thực. Hình thức này rất thích hợp với các thế giới thực có cấu trúc tổ chức phân cấp.

3.2.3 Tích hợp

Sử dụng đồng thời cả hai hình thức trên

3.3 Thể hiện kết quả tra cứu

3.3.1 Kết quả tra cứu dùng thông báo

Với hình thức này kết quả tra cứu chỉ đơn giản là câu thông báo cho biết có hay không đối tượng cần tìm. Đây là hình thức đơn giản nhất và có tính tiện dụng thấp nhất. Với hình thức này người sử dụng không biết thêm thông tin gì của đối tượng tìm thấy.

3.3.2 Kết quả tra cứu dùng danh sách đơn

Với hình thức này kết quả tra cứu là danh sách các đối tượng tìm thấy cùng với một số thông in cơ bản về đối tượng. Hình thức này cho phép người dùng biết thêm thông tin cơ bản về đối tượng tìm thấy nhưng không biết chi tiết về các hoạt động của đối tượng qua các quan hệ với đối tượng khác.

3.3.3 Kết quả tra cứu dùng xâu các danh sách

Với hình thức này kết quả tra cứu bao gồm nhiều danh sách mà trong đó danh sách thứ k dsk chứa các mô tả cho một phẩn tử trong danh sách thứ k-1 dsk-1. Danh sách đầu tiên chính là danh sách đơn trong hình thức trên.

Hình thức này không những cho phép xem các thông tin cơ bản về đối tượng tìm thấy mà còn cho biết chi tiết về hoạt động của đối tượng qua các quan hệ với các đối tượng khác.

3.3.4 Cây các danh sách

Với hình thức này kết quả tra cứu là cây mà các nút chính là các danh sách. Danh sách tương ứng trong một nút con sẽ là các thông tin mô tả chi tiết về một phần tử được chọn trong danh sách của nút cha. Danh sách đầu tiên chính là danh sách đơn trong hình thức phía trên.

Hình thức trình bày này cho phép xem được quá trình hoạt động của đối tượng với nhiều quan hệ, nhiều loại hoạt động khác.

3.4 Thao tác người dùng và xử lý của phần mềm

③ Nhập giá trị cho các tiêu chuẩn tra cứu:

- Có thể nhập một số hoặc tất cả tiêu chuẩn tra cứu

-Với các tiêu chuẩn thường dùng có thể dùng giá trị định sẵn (loại sách thường tìm, loại hàng thường mua, …) để tiện dụng hơn cho người dùng.

- Trong quá trình nhập liệu thông thường phần mềm sẽ không có xử lý tính toán nào ngoài việc chờ nhập giá trị cho các tiêu chuẩn tra cứu.

③ Yêu cầu bắt đầu tra cứu:

- Nhấn vào nút tra cứu.

- Dựa vào giá trị các tiêu chuẩn tra cứu phần mềm sẽ tiến hành đọc và xuất các kết quả tra cứu tương ứng (xử lý tra cứu).

③ Xem xét chi tiết các kết quả tra cứu

- Chọn đối tượng cần xem chi tiết trong danh sách của kết quả tra cứu..

- Nhập phạm vi thời gian cần quan sát (thông thường là thời gian từ ngày … đến này … hoặc đơn vị thời gian cụ thể tháng … năm …).

- Dựa vào đối tượng được chọn và dựa vào phạm vị thời gian, phần mềm sẽ đọc và xuất các kết quả tra cứu cấp chi tiết hơn theo từng loại hoạt động.

③ Yêu cầu kết xuất

- Có thể bổ sung các nút điều khiển tương ứng với việc in ấn hoặc ghi lên tập tin các kết quả tra cứu. Thông thường mỗi kết quả tra cứu sẽ có một nút riêng, nhưng cũng có thể dùng chung một nút cho mọi kết quả tra cứu (dựa vào kết quả hiện hành).

- Việc kết xuất thông thường là qua máy in, tuy nhiên cũng có thể cho phép người dùng xác định lại đích của kết xuất (tập tin Excel, trang web,…) tùy theo mục đích sử dụng.

4. Thiết kế màn hình nhập liệu

4.1 Mô tả màn hình nhập liệu

③ **Ý nghĩa sử dụng:**

Màn hình nhập liệu là màn hình cho phép người dùng thực hiện các công việc có liên quan đến ghi chép trong thế giới thực.

③ **Nội dung:**

- Các thông tin nhập liệu: Với loại thông tin này, người dùng chịu trách nhiệm nhập trực tiếp các giá trị, phần mềm sẽ tiến hành kiểm tra tính hợp lệ các giá trị nhập dựa vào qui định liên quan.

- Các thông tin tính toán: Với thông tin này, phần mềm chịu trách nhiệm tính toán và xuất trên màn hình. Thông thường loại thông tin này giúp việc nhập liệu thuận tiện hơn (nhập số lượng hàng bán khi biết số lượng đang tồn tương ứng, nhập sách mượn khi biết số sách độc giả đang mượn …).

③ **Hình thức trình bày:** Một số hình thức thông dụng

- Danh sách: Màn hình nhập liệu có dạng một danh sách trong thế giới thực (danh sách các thể loại sách, danh sách các lớp học).

- Hồ sơ: Màn hình nhập liệu có dạng một hồ sơ với nhiều thông tin chi tiết (hồ sơ học sinh, hồ sơ cầu thủ).

- Phiếu: Màn hình nhập liệu có dạng phiếu với nhiều dòng chi tiết (hóa đơn bán hàng, phiếu nhập hàng, …).

- Tích hợp: Sử dụng đồng thời các hình thức trên.

③ **Thao tác người dùng:** Có 3 thao tác cơ bản trên màn hình nhập liệu.

- Nút Ghi: Lưu trữ thông tin.
- Nút Xóa: Xóa các thông tin đã lưu trữ.
- Nút Tìm: Tìm và cập nhật lại thông tin đã lưu trữ.

Ngoài ra để tăng tính tiện dụng có bổ sung các thao tác khác.

- Tạo phím nóng: Định nghĩa các phím nóng tương ứng với các giá trị nhập liệu thường dùng, điều này cho phép tăng tốc độ nhập liệu.
- Tạo các nút chuyển điều khiển: Chuyển điều khiển trực tiếp đến màn hình khác có liên quan đến việc nhập liệu hiện hành (bổ sung thể loại sách mới, nhà xuất bản mới, …).

4.2 Các hình thức trình bày màn hình nhập liệu

4.2.1 Thiết kế màn hình nhập liệu dạng danh sách

③ **Sử dụng:** Dạng danh sách thích hợp khi cần nhập liệu các bảng danh sách với kích thước nhỏ (danh sách các thể loại sách, các môn học,…).

③ **Thành phần nhập liệu:**

- Thông tin nhập liệu: Các thuôc tính các bảng liên quan

- Thông tin tính toán: Thông thường các mã số được tự động phát sinh.

③ **Thành phần xử lý:**

- Ghi: Ghi nhận các thay đổi trên danh sách (thêm mới, sửa đổi.

- Xóa: Xóa 1 dòng trong danh sách.

- Thoát: Quy về màn hình trước đó.

③ **Các thao tác:**

Người dùng có thể tùy ý sửa đổi các thông tin trên các dòng hoặc thêm dòng mới (nhập vào cuối danh sách), xóa dòng sau khi đã chọn dòng cần xóa và cuối cùng yêu cầu ghi các thay đổi trên bộ nhớ phụ.

Tuy nhiên trong một số trường hợp đặc biệt một số thao tác có thể bị cấm (không cho xóa, không cho thay đổi một số thuộc tính …) tùy vào ý nghĩa cụ thể của danh sách.

4.2.2 Thiết kế màn hình nhập liệu dạng hồ sơ

③ **Sử dụng**: Dạng hồ sơ thích hợp khi cần nhập liệu các hồ sơ các đối tượng trong thế giới thực (hồ sơ học sinh, đội bóng).

③ **Thành phần dữ liệu**:

- Thông tin nhập liệu: Các thuộc tính các bảng liên quan

- Thông tin tính toán: Thông thường các mã số được tự động phát sinh.

③ **Thành phần xử lý**: Thêm, Ghi, Xóa, Tìm, Thoát

③ **Các thao tác**: Người dùng có thể thêm hồ sơ mới, tìm lại hồ sơ đã lưu trữ và sau đó tuỳ ý sửa đổi, các thông tin trên hồ sơ tìm thấy, xóa hồ sơ tìm thấy, và cuối cùng yêu cầu lưu trữ hồ sơ. Tuy nhiên để tăng tính tiện dụng một số thao tác chuyển điều khiển có thể được bổ sung cho phép di chuyển nhanh đến các màn hình nhập liệu liên quan khi cần thiết.

4.2.3 Thiết kế màn hình nhập liệu dạng phiếu

③ **Sử dụng**: Dạng phiếu thích hợp khi cần nhập liệu các phiếu ghi nhận thông tin về hoạt động các đối tượng trong thế giới thực.

③ **Thành phần dữ liệu**:

- Thông tin nhập liệu: Các thông tin liên quan đến bảng.

- Thông tin tính toán: Thông thường các mã số được tự động phát sinh.

Thành phần xử lý: Thêm, Thêm chi tiết, Ghi, Xóa, Xóa chi tiết, Tìm, Sửa chi tiết, Thoát.

Chương 6: CÀI ĐẶT

1. Tổng quan

Trong cài đặt, chúng ta xuất phát từ kết quả của thiết kế và cài đặt hệ thống dưới dạng các thành phần, tức là các mã nguồn, các kịch bản, các tập tin nhị phân, các tập tin thực thi, thư viện, bảng, dữ liệu

May thay, phần lớn kiến trúc của hệ thống đã được nắm bắt trong quá trình thiết kế. Mục tiêu chủ yếu của cài đặt là bổ sung thêm cho kiến trúc và hệ thống để trở thành một khối hoàn chỉnh. Cụ thể hơn, các mục tiêu cài đặt là:

- Lên kế hoạch tích hợp hệ thống (system integration) trong mỗi bước lặp một các tăng cường. Điều này có nghĩa là một hệ thống được cài đặt bởi một dãy các bước nhỏ liên tiếp và có thể quản lý được.
- Phân phối hệ thống bằng cách ánh xạ các thành phần thi hành được vào các nút trong mô hình triển khai. Công việc này chủ yếu dựa vào các lớp động được tìm thấy trong quá trình thiết kế.
- Cài đặt các lớp thiết kế và các hệ thống con đã tìm được trong quá trình thiết kế. Đặc biệt, các lớp thiết kế được cài đặt thành các thành phần file chứa mã nguồn.
- Kiểm thử đơn vị các thành phần, rồi sau đó tích hợp chúng bằng các biên dịch chúng và liên kết chúng lại với nhau thành một hoặc nhiều thành phần thi hành được trước khi kiểm thử tích hợp và kiểm thử hệ thống

Để đạt được những mục tiêu trên, chương này sẽ đưa ra những bàn luận chung quá trình cài đặt phần mềm. Trước tiên chúng ta đưa ra khái quát chung của tiêu chí chất lượng và yếu tố then chốt của ngôn ngữ lập trình và chúng ta sẽ thảo luận trên những phần chính của kiểu lập trình như cấu trúc, những diễn giải, hình thức, và hiệu qủa. Cuối cùng chúng ta sẽ bàn đến thuộc tính chính của phần mềm như khả năng uyển chuyển và dùng lại.

Một cài đặt tốt phản ánh những quyết định của thiết kế

Cài đặt nên đảm bảo theo các mục sau:

- Cấu trúc, cấu trúc dữ liệu và những định nghĩa được chọn lựa và thiết lập trong suốt thủ tục thiết kế cần được tổ chức dễ dàng nhận biết trong quá trình cài đặt.

- Mức trừu tượng của thiết kế (các lớp (class), mô đun (module), thuật toán (algorithm), cấu trúc dữ liệu (data structure), và kiểu dữ liệu (data type)) cũng phải linh động trong thực hiện.
- Giao diện giữa các thành phần (components) của hệ thống phần mềm được mô tả rõ ràng trong thực hiện.
- Quá trình thực hiện cũng có thể được kiểm tra độ tin cậy của đối tượng và thao tác với trình biên dịch (trước khi qua giai đoạn kiểm tra chương trình thực sự).

Đảm bảo những đặc trưng ở trên phụ thuộc vào việc chọn lựa ngôn ngữ thực hiện và kiểu lập trình.

2. Môi trường lập trình

Câu hỏi cho việc chọn lựa "đúng" ngôn ngữ lập trình luôn là chủ đề được đưa ra trong qui trình lập trình.Việc chọn lựa ngôn ngữ lập trình trong công đoạn thực thi của một dự án luôn đóng vai trò quan trọng.

Trong trường hợp lý tưởng, thiết kế nên đảm trách mà không có bất cứ kiến thức liên quan đến ngôn ngữ thực hiện sau đó sao cho thiết kế có thể thực hiện được trên bất kỳ ngôn ngữ lập trình nào.

2.1 Chất lượng đòi hỏi cho một ngôn ngữ lập trình:

- Tính mô đun hóa
- Giá trị của tài liệu
- Cấu trúc dữ liệu
- Control flow (luồng điều khiển)
- Tính hiệu quả
- Khả năng tích hợp (Integrity)
- Tính khả chuyển (Portability)
- Hỗ trợ hộp thoại
- Yếu tố ngôn ngữ chuyên biệt

2.2 Khả năng Mô đun hóa của ngôn ngữ lập trình

Khả năng mô đun hóa là mức độ hỗ trợ những khả năng mô đun hóa chương trình. Phác thảo một chương trình lớn thành nhiều mô đun là điều kiện tiên quyết để thực thi trong dự án phần mềm.

Không có khả năng mô đun hóa thì phân chia công việc trong giai đoạn thực hiện trở nên không thể được. Những chương trình đơn nhất trở nên không thể quản lý: chúng khó có thể bảo trì và sưu liệu và chúng thực hiện với thời gian biên dịch dài.

Ngôn ngữ như Pascal chuẩn (không hỗ trợ mô đun, nhưng so sánh với Turbo Pascal và Mođun 2) để chứng minh tính không thích hợp cho những dự án lớn.

Nếu một ngôn ngữ hỗ trợ phát thảo một chương trình thành những phần nhỏ, chúng phải đảm bảo những thành phần phải hoạt động với nhau. Nếu một thủ tục được thực thi ở mô đun khác, cũng được kiểm tra thủ tục có thực sự tồn tại và nó có được sử dụng chính xác hay không (nghĩa là số tham số và kiểu dữ liệu là chính xác).

Những ngôn ngữ với việc biên dịch độc lập (ví dụ như C) nơi việc kiểm tra của ngôn ngữ chỉ thay thế ở quá trình run-time.

Ngôn ngữ với việc biên dịch tách biệt (ví dụ Ada và Modula-2) mỗi mô đun có một mô tả giao diện cung cấp những phương thức cơ bản cho việc kiểm tra những thành phần của mô đun dùng tại thời điểm chạy chương trình (run time).

2.3 Giá trị sưu liệu của ngôn ngữ lập trình

Ảnh hưởng của khả năng có thể đọc và bảo trì của chương trình. Điều quan trọng của giá trị sưu liệu được nâng lên đối với những chương trình lớn và cho những phần mềm mà khách hàng vẫn tiếp tục phát triển.

Giá trị của sưu liệu cao mang lại kết quả hơn. Vì chương trình nói chung chỉ được 1 lần nhưng việc đọc nó có thể lặp lại, hiệu quả tối thiểu thêm vào trong cách viết sẽ chịu ảnh hưởng không đâu nhiều hơn là trong quá trình bảo trì. Giống như phạm vi ngôn ngữ ảnh hưởng đến khả năng đọc chương trình

Nhiều ngôn ngữ mở rộng với quá nhiều chức năng chuyên biệt sẽ khó để hiểu thấu tất cả chi tiết, vì vậy dẫn đến giải thích sai.

2.4 Cấu trúc dữ liệu trong ngôn ngữ lập trình

Dữ liệu phức tạp phải được xử lý, sự sẵn sàng trong cấu trúc dữ liệu trong ngôn ngữ lập trình đóng vai trò quan trọng.

Ngôn ngữ như C cho phép khai báo con trỏ đối với cấu trúc dữ liệu. Điều này cho phép cấu trúc dữ liệu phức tạp, và phạm vi và cấu trúc của chúng có thể thay đổi ở thời điểm run-time. Tuy nhiên, việc drawback những cấu trúc dữ liệu chúng được mở và được phép truy xuất không nghiêm ngặt (nhưng so sánh với Java).

Trọng tâm trong dự án lớn với nhiều nhóm dự án, dữ liệu trừu tượng mang nghĩa cụ thể. Mặc dù dữ liệu trừu tượng có thể phân biệt với bất kỳ mô đun ngôn ngữ, bởi khả năng đọc tốt hơn.

Ngôn ngữ lập trình hướng đối tượng có những đặc trưng mở rộng loại kiểu dữ liệu trừu tượng cho phép hiện thực hoá những hệ thống phần mềm phức tạp. Đối với những giải pháp mở rộng và uyển chuyển, ngôn ngữ lập trình hướng đối tượng cung cấp tuỳ chọn đặc biệt tốt.

2.5 Ví dụ minh họa

Ví dụ: Giai đoạn thực hiện phần mềm quản lý thư viện, các giai đoạn trước đã được minh họa ở các chương trước

Giai đoạn 5: Thực hiện phần mềm

- ③ Hệ thống lớp đối tượng: Tạo lập các lớp đối tượng (THU_VIEN, DOC_GIA, SACH) theo mô tả của phần thiết kế trong một môi trường cụ thể nào đó (Visual Basic, Visual C++, Java,…)
- ③ Hệ thống giao diện: Tạo lập (vẽ) các màn hình giao diện (màn hình chính, màn hình lập thẻ, màn hình cho mượn sách, màn hình nhận sách, màn hình trả sách) theo mô tả của phần thiết kế trong một môi trường cụ thể nào đó (Visual Basic, Viusal C++, Java)
- ③ Hệ thống lưu trữ: Tạo lập cấu trúc cơ sở dữ liệu (các bảng THU_VIEN, DOC_GIA, SACH, MUON_SACH) theo mô tả của phần thiết kế trong môi trường cụ thể nào đó(Access, SQL Server, Oracle,…)

3. Phong cách lập trình

Sau khi thực hiện và kiểm tra, hệ thống phần mềm hiếm khi được sử dụng một thời gian dài mà không có sửa đổi điều chỉnh. Thực vậy, điều này luôn là đúng: khi yêu cầu được cập nhật hoặc mở rộng sau khi hoàn chỉnh sản phẩm và trongsuốt quá trình thực hiện thao tác, không phát hiện ra lỗi hay những thiếu sót phát sinh. Giai đoạn thực hiện chắc chắn phải được sửa đổi và mở rộng, đòi hỏi lặp lại việc đọc và hiểu chương trình nguồn. Trong trường hợp lý tưởng, chức năng của một thành phần chương trình được hiểu mà không có kiến thức từ tài liệu thiết kế mà chỉ từ chương trình nguồn. Chương trình nguồn chỉ là tài liệu luôn phản ánh hiện trạng của thực thị.

Khả năng đọc được một chương trình phụ thuộc vào ngôn ngữ lập trình được dùng và vào phong cách lập trình của người thực hiện. Việc viết một chương trình có thể đọc được là tiến trình sáng tạo. Phong cách lập trình của người thực hiện ánh hưởng đến khả năng đọc được chương trình hơn là ngôn ngữ lập trình được sử dụng.

Yếu tố quan trọng nhất của phong cách lập trình tốt là:

- ③ Tính cấu trúc
- ③ Sự trình bày diễn đạt
- ③ Cách thức trình bày bên ngoài
- ③ Hiệu suất

3.1 Tính cấu trúc

Việc phân rã một hệ thống phần mềm dựa trên mục đích chính là độ phức tạp thông qua mức trừu tượng phần thành phần cô đọng rõ nét (cấu trúc chương trình lớn).

Chọn lựa những thành phần chương trình phù hợp trong việc định ra những thuật toán của thủ tục con.(cấu trúc chương trình nhỏ).

3.2 Thế mạnh của diễn đạt

Qui trình thực hiện một hệ thống phần mềm chứa đựng việc đặt tên đối tượng và mô tả các công việc thực thi của đối tượng này.

Chọn lựa tên đặc biệt quan trọng trong việc viết thuật toán

Một số đề nghị:

- ③ Nếu dùng chữ viết tắt, thì sử dụng tên đặt này người đọc chương trình có thể hiểu mà không cần bất cứ sự giải thích nào. Việc sử dụng những từ viết tắt chỉ bao gồm ngữ cảnh.
- ③ Với một hệ thống gán tên chỉ nên một ngôn ngữ (ví dụ đừng dùng lẫn lộn tiếng Anh và tiết Việt).
- ③ Dùng chữ hoa chữ thường để phân biệt những loại định nghĩa khác nhau (ví dụ chữ hoa đầu tiên cho kiểu dữ liệu, lớp, mô đun, chữ thường đầu tiên cho biến) và đặt tên dài hơn có thể đọc (ví dụ CheckInputValue).
- ③ Dùng danh từ cho giá trị, động từ cho hoạt động, và thuộc tính cho điều kiện để làm rõ ý nghĩa nhận diện (ví dụ width, ReadKey, valid).
- ③ Thiết lập những qui luật cho chính bạn sử dụng theo chúng một cách thích hợp.

Phong cách lập trình tốt được tìm thấy trong diễn giải sử dụng ghi chú: đóng góp cho khả năng đọc được chương trình và như vậy nó là thành phần quan trọng của chương trình. Hiệu chỉnh việc ghi chú chương trình không dễ dàng và đòi hỏi kinh nghiệm, sáng tạo và khả năng diễn đạt thông điệp gọn gàng và chính xác.

Một số luật cho việc viết những ghi chú:

③ Mỗi thành phần hệ thống (mỗi mô đun và lớp) nên bắt đầu với ghi chú chi tiết cho người đọc những thông tin với một vài vấn đề liên quan đến thành phần của hệ thống:

- Thành phần này làm gì?
- Thành phần này được sử dụng như thế nào trong những ngữ cảnh gì?
- Những phương thức đặc biệt được sử dụng.
- Ai là Tác giả của thành phần này?
- Thành phần này được viết khi nào?
- Những sửa đổi cập nhật nó được thực hiện.

③ Mỗi thủ tục và phương thức cung cấp ghi chú mô tả công việc (có thể có). Điều này ứng dụng đặt biệt cho đặc tả giao diện.

③ Giải thích ý nghĩa của biến với ghi chú.

③ Những thành phần của chương chịu trách nhiệm cho những tác nhiệm riêng nên được đánh nhãn với những ghi chú.

③ Những khối lệnh khó để hiểu (ví dụ thủ tục rắc rối hay những thành phần mà đặc trưng cho một máy tính cụ thể) nên được mô tả ghi chú sao cho người đọc dễ dàng hiểu chúng.

③ Hệ thống phần mềm nên chứa mà một vài ghi chú gãy gọn súc tích như nếu có thể nhưng nhiều ghi chú chi tiết tương xứng nếu cần thiết.

③ Đảm bảo những thay đổi chương trình không chỉ có tác động phần khai báo và khối lệnh mà còn phản ánh những cập nhật trong phần ghi chú. Những ghi chú không chính xác thì sẽ tệ hơn.

Lưu ý: những luật trên tuân thủ cân nhắc bởi vì không có luật áp dụng đồng nhất cho tất cả các hệ thống phần mềm và mỗi phạm vi ứng dụng. Việc ghi chú hệ thống phần mềm là một nghệ thuật cũng giống như phần thiết kế cài đặt hệ thống phần mềm.

3.3 Cách thức trình bày bên ngoài

Ngoài sự chọn tên và ghi chú, khả năng đọc được của hệ thống phần mềm cũng phụ thuộc vào cách thức trình bày bên ngoài.

Một số luật đề nghị cho hình thức trình bày chương trình:

③ Mỗi thành phần của chương trình (components), những khai báo (của kiểu dữ liệu, hằng biến, …) nên được tách biệt mỗi phần của khối lệnh.

- ③ Phần khai báo nên có một cấu trúc đồng nhất khi có thể như thứ tự sau: hằng, kiểu dữ liệu, lớp, mô đun, phương thức và thủ tục.
- ③ Mô tả giao diện (danh sách tham số cho phương thức và thủ tục) nên tách tham số nhập liệu, kết xuất và nhập/xuất.
- ③ Phần ghi chú và chương trình nguồn nên tách bạch.
- ③ Cấu trúc của chương trình nên được nhấn mạnh ở phần canh chỉnh lề (sử dụng phím tab cho từ mỗi đầu khối lệnh đến khối lệnh theo sau).

4. Đánh giá chất lượng công việc

4.1 Hiện thực tăng cường

Ý tưởng cơ bản của việc hiện thực tăng cường gần với việc trộn giai đoạn thiết kế và cài đặt hơn là tách biệt hai giai đoạn này mà mô hình qui trình phát triển phần mềm tuần tự cổ điển đề ra.

Điểm nhấn mạnh của phương pháp này được tìm thấy dựa trên thực nghiệm rằng những quyết định trong thiết kế và cài đặt có tác động lẫn nhau và vì vậy nếu tách bạch thiết kế khắt khe sẽ không đạt được mục tiêu. Trong nhiều trường hợp, chỉ có cài đặt mới quyết định việc phân rã cấu trúc của thiết kế chứng minh sự thõa mãn đầy đủ vấn đề.

Hiện thực tăng cường nghĩa là sau mỗi bước thiết kế có liên quan đến kiến trúc, kiến trúc phần mềm hiện hành được thẩm định dựa trên những trường hợp thực. Sự tác động qua lại giữa các thành phần hệ thống cụ thể trong thiết kế (trong hình thức đặc tả giao diện) được thẩm định. Để có thể làm được điều này, những thành phần hệ thống (hành vi xuất/ nhập của chúng) được mô phỏng hay thực tế hóa như khuôn mẫu. Nếu có những nghi ngờ liên quan đến tính khả thi của thành phần thì tiến trình thiết kế được ngắt và những thành phần được thực hiện. Chỉ khi hiện thực và nhúng chúng vào trong kiến trúc hệ thống trước đó được kiểm tra thì tiến trình thiết kế tiếp tục hay kiến trúc được chấp nhận tương ứng kiến thức thu được trong hiện thực thành phần.

Hiệu quả của phương pháp này phụ thuộc vào việc mở rộng vào khả năng tích hợp thành phần hệ thống mà được viết trong chuẩn mực khác nhau và được hoàn chỉnh ở nhữn cấp độ khác nhau, đối với toàn bộ hệ thống để thực hiện gần với thực tế. Một vài thành phần hệ thống, ví dụ giao diện người dùng và mô hình dữ liệu được thể hiện dưới dạng khuôn mẫu, những thành phần khác từ thư viện thành phần có sẵn hay tồn tại như hiện thực hoàn chỉnh được thể hiện dưới dạng mã nguồn thực thi còn các thành phần hệ thống khác có sẵn như đặc

tả giao diện. Đối với sự hợp lệ của thiết kế hệ thống hiện hành, bất kỳ lúc nào giao diện người dùng triển khai thì tương ứng khuôn mẫu cần được kích hoạt.

4.2 Đánh giá lại thiết kế và chương trình (Design and Code Review)

Với việc xem lại thiết kế và chương trình, sẽ giúp hoàn chỉnh chất lượng hiệu quả của công việc hơn là chúng ta chỉ điều chỉnh những thay đổi đơn lẻ trong quá trình phát triển phần mềm.Trong những chương trình lớn, đòi hỏi xem xét lại những yêu cầu, đặc tả, thiết kế, và cả chương trình của chúng ta. Giúp điều chỉnh thiếu sót, logic, cấu trúc, tính sáng tỏ. Khi chương trình không rõ hay mơ hồ xáo trộn, thêm những ghi chú thì tốt hơn hay viết lại nó một cách đơn giản hơn sẽ làm cho chương trình dễ đọc và dễ hiểu. Việc làm này sẽ tạo cho chúng ta sự tự tin xuất bản hay trình bày cho bạn bè hay tập thể.

Mục đích của review để đảm bảo chương trình tạo ra đạt chất lượng cao nhất. Một trong việc review là kiểm duyệt, duyệt qua, xem xét mục riêng từ thiết kế đến từng dòng lệnh. Review có thể được dùng trên yêu cầu, thiết kế, hồ sơ tài liệu, hay bất kỳ yếu tố của sản phẩm.

Nhiều dự án phần mềm trãi qua nữa qúa trình phát triển ở giai đoạn kiểm thử. Điều này không hiệu quả. Review thiết kế và chương trình là những cách thức hiệu quả tìm và sửa chữa thiếu sót. Với review, chúng ta có thể tìm ra những thiếu sót trực tiếp, trong khi chúng ta chỉ có thể tìm ra những dấu hiệu. Khi chúng ta xem lại chương trình, chúng ta biết nơi và những logic gì giả sử phải được làm.Những sửa lỗi của chúng ta sẽ hoàn chỉnh và chính xác.

Công việc review cho phép quay trở lại bất cứ việc gì chúng ta đã làm. Nhóm phát triển nên ngồi lại với nhau để đọc lại mọi thiết kế và chương trình, nghiên cứu, hiểu nó. Sửa những sai sót: logic, cấu trúc, tính rõ ràng. Sau khi đã được xem xét và đánh giá, viết lại chương trình. Chỗ nào không rõ ràng hay lộn xộn, thêm ghi chú và viết lại hoàn chỉnh làm cho dễ đọc và dễ hiểu.

5. Ví dụ minh họa

Ví dụ: Xét phần mềm hỗ trợ giải bài tập phương trình đại só với 4 yêu cầu: Soạn đề bài, Soạn đáp án, Giải bài tập, Chấm điểm.

Nhằm thể hiện các giai đoạn thực hiện trong qui trình

Giai đoạn 1: Xác định yêu cầu

③ Yêu cầu 1: Soạn đề bài với qui định về Soạn đề bài

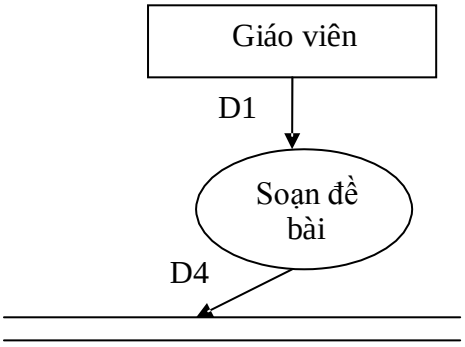
③ Yêu cầu 2: Soạn đáp án với qui định về Soạn đáp án và biểu mẫu Soạn đáp án

③ Yêu cầu 3: Giải bài tập với qui định về Giải bài tập và biểu mẫu Giải bài tập

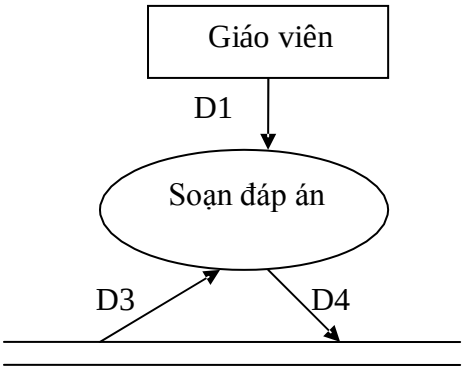
③ Yêu cầu 4: Chấm điểm với qui định về Chấm điểm

Giai đoạn 2:

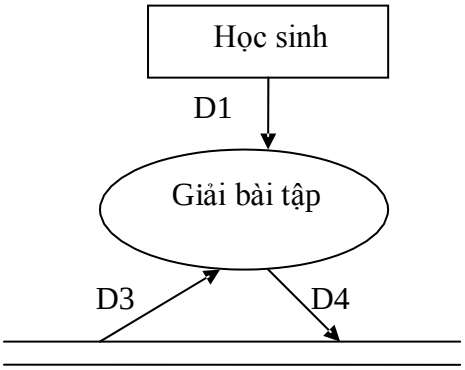
③ Sơ đồ luồng dữ liệu cho công việc Soạn đề bài



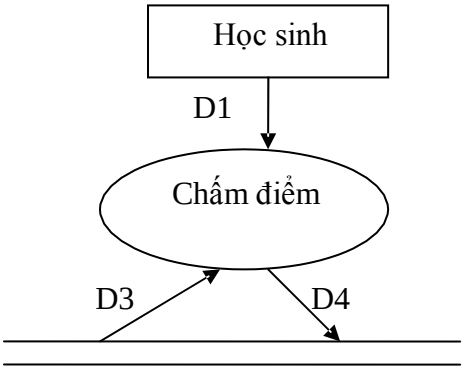
③ Sơ đồ luồng dữ liệu cho công việc Soạn đáp án



③ Sơ đồ luồng dữ liệu cho công việc Giải bài tập



③ Sơ đồ luồng dữ liệu cho công việc Chấm điểm



Giai đoạn 3: Phân tích yêu cầu chức năng

Giai đoạn 4: Thiết kế phần mềm (đã trình bày trong phần kiến trúc phần mềm của chương Thiết kế phần mềm)

Giai đoạn 5: Thực hiện phần mềm

- ③ Hệ thống Lớp đối tượng: Tạo lập các lớp đối tượng SACH_BAI_TAP, BAI_TAP theo mô tả phần thiết kế trong môi trường cụ thể nào đó (VIsual Basic, Visual C++, Java
- ③ Hệ thống giao diện: Tạo lập (vẽ) các màn hình giao diện (màn hình chính, màn hình soạn đề bài, màn hình soạn đáp án, màn hình giải bài tập, màn hình chấm điểm) theo mô tả của phần thiết kế trong một môi trường cụ thể nào đó (Visual Basic, Visual C++, Java, v.v.)

Hệ thông lưu trữ: Tạo cấu trúc cơ sở dữ liệu (các bảng SACH_BAI_TAP, BAI_TAP, BAI_GIAI, BUOC_GIAI) theo mô tả của phần thiết kế trong một môi trường cụ thể nào đó (Access, SQL Server, Oracle,. v.v)

Giai đoạn 6: Kiểm chứng phần mềm xem ở chương Kiểm thử

Chương 7: KIỂM THỬ PHẦN MỀM

1. Tổng quan

Kiểm thử phần mềm là tiến hành thí nghiệm để so sánh kết quả thực tế với lý thuyết nhằm mục đích phát hiện lỗi.

Bộ thử nghiệm (test cases) là dữ liệu dùng để kiểm tra hoạt động của chương trình. Một bộ kiểm thử tốt là bộ có khả năng phát hiện ra lỗi của chương trình. Khi tiến hành kiểm thử, chúng ta chỉ có thể chứng minh được sự tồn tại của lỗi nhưng không chứng minh được rằng trong chương trình không có lỗi.

Nội dung của bộ thử nghiệm:

- Tên môđun/chức năng muốn kiểm thử
- Dữ liệu vào
  - Dữ liệu của chương trình: số, xâu ký tự, tập tin,
  - Môi trường thử nghiệm: phần cứng, hệ điều hành,
  - Thứ tự thao tác (kiểm thử giao diện)
- Kết quả mong muốn
  - Thông thường: số, xâu ký tự, tập tin, …
  - Màn hình, thời gian phản hồi
- Kết quả thực tế

Không gian thử nghiệm là tập các bộ số thử nghiệm. Không gian này nói chung là rất lớn. Nếu có thể vét cạn được không gian thử nghiệm thì chắc chắn qua phép kiểm tra đơn vị sẽ không còn lỗi. Tuy nhiên điều này không khả thi trong thực tế. Do đó khi đề cập đến tính đúng đắn của phần mềm chúng ta dùng khái niệm độ tin cậy.

Phương pháp kiểm thử là cách chọn bộ số thử nghiệm để tăng cường độ tin cậy của đơn vị cần kiểm tra. Hay nói cách khác phương pháp kiểm thử là cách phân hoạch không gian thử nghiệm thành nhiều miền rồi chọn bộ số liệu thử nghiệm đại diện cho miền đó. Như vậy cần tránh trường hợp mọi bộ thử nghiệm đều rơi vào một miền kiểm tra.

2. Yêu cầu đối với kiểm thử

- Tính lặp lại:
  - Kiểm thử phải lặp lại được (kiểm tra xem lỗi đã được sửa hay chưa)

- Dữ liệu/trạng thái phải mô tả được

- Tính hệ thống: phải đảm bảo đã kiểm tra hết các trường hợp.
- Được lập tài liệu: phải kiểm soát được tiến trình/kết quả.

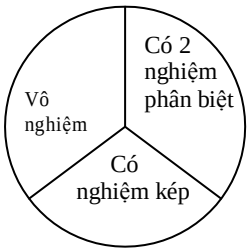
3. Các kỹ thuật kiểm thử

3.1 Phương pháp hộp đen (Kiểm thử chức năng)

Phương pháp kiểm thử này chỉ dựa trên bản đặc tả các chức năng. Do đó, chúng ta chỉ chú tâm đến phát hiện các sai sót về chức năng mà không quan tâm đến cách cài đặt cụ thể. Với phương pháp này chúng ta có khả năng phát hiện các sai sót, thiếu sót về mặt chức năng; sai sót về giao diện của môđun, kiểm tra tính hiệu quả; phát hiện lỗi khởi tạo, lỗi kết thúc.

Do không thể kiểm thử mọi trường hợp trên thực tế, chúng ta sẽ chia không gian thử nghiệm dựa vào giá trị nhập xuất của đơn vị cần kiểm tra. Ứng với mỗi vùng dữ liệu chúng ta sẽ thiết kế những bộ thử nghiệm tương ứng và đặc biệt là các bộ thử nghiệm tại các gía trị biên của vùng dữ liệu.

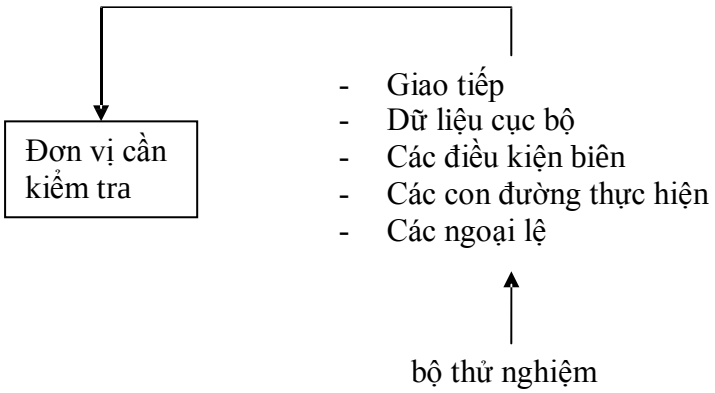
Để kiểm chứng chương trình giải phương trình bậc 2 theo phương pháp hộp đen, chúng ta sẽ phân chia không gian thử nghiệm thành 3 vùng như sau:



Sau khi đã thử kiểm tra với các bộ thử nghiệm đã thiết kế, chúng ta cần mở rộng bộ thử nghiệm cho các trường hợp đặc biệt như: biên của số trong máy tính (32767,-32768), số không, số âm, số thập phân, dữ liệu sai kiểu, dữ liệu ngẫu nhiên.

a. Phương pháp hộp trắng (Kiểm thử cấu trúc)

Theo phương pháp này, chúng ta sẽ chia không gian thử nghiệm dựa vào cấu trúc của



đơn vị cần kiểm tra.

Kiểm tra giao tiếp của đơn vị là để đảm bảo dòng thông tin vào ra đơn vị luôn đúng (đúng giá trị, khớp kiểu...)

Kiểm tra dữ liệu cục bộ để đảm bảo dữ liệu được lưu trữ trong đơn vị toàn vẹn trong suốt quá trình thuật giải được thực hiện.

Ví dụ: nhập dữ liệu sai, tên biến không đúng, kiểu dữ liệu không nhất quán, các ràng buộc hoặc ngoại lệ.

Kiểm tra các điều kiện biên của các câu lệnh if, vòng lặp để đảm bảo đơn vị luôn chạy đúng tại các biên này.

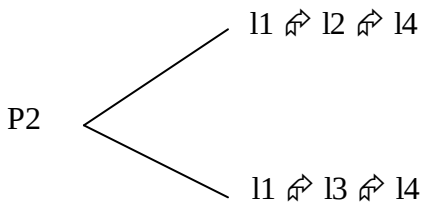
Kiểm tra để đảm bảo mọi con đường thực hiện phải được đi qua ít nhất một lần. Con đường thực hiện của một đơn vị chương trình là một dãy có thứ tự các câu lệnh bên trong đơn vị đó sẽ được thực hiện khi kích hoạt đơn vị.

Ví dụ:

P1	P2
l1	l1
l2	if (đk) l2
l3	else l3
l4	l4

Con đường thực hiện của p1 và p2 như sau:

P1: l1 ⇨ l2 ⇨ l3 ⇨ l4



4. Các giai đoạn và chiến lược kiểm thử

Đối với những dự án phần mềm lớn, những người tham gia được chia thành 2 nhóm:

- Nhóm thứ nhất: gồm những người tham gia trong dự án phát triển phần mềm. Nhóm này chịu trách nhiệm kiểm tra các đơn vị của chương trình để chắc chắn chúng thực hiện đúng theo thiết kế.
- Nhóm thứ hai: độc lập gồm các chuyên gia tin học nhưng không thuộc nhóm thứ nhất. Nhóm này có nhiệm vụ phát hiện các lỗi do nhóm thứ nhất chủ quan còn để lại.

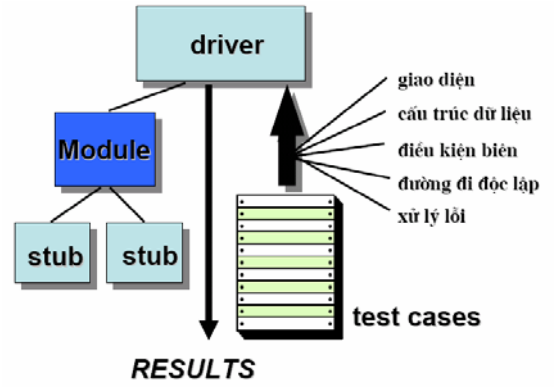
4.1 Kiểm thử đơn vị

Sử dụng kỹ thuật hộp trắng và dựa vào hồ sơ thiết kế để xây dựng các bộ thử nghiệm sao cho khả năng phát hiện lỗi là lớn nhất.

Vì đơn vị được kiểm tra không là 1 chương trình đầy đủ, hơn nữa đơn vị này có thể được gọi bởi những đơn vị khác hoặc gọi đến những đơn vị khác nên dù chương trình đã được hoàn tất đầy đủ các đơn vị, chúng ta cũng không nên giả thuyết sự tồn tại hoặc tính đúng đắn của các đơn vị khác mà phải xây dựng các module giả lập đơn vị gọi tên là driver và đơn vị bị gọi là stub.

Driver đóng vai trò như một chương trình chính nhập các bộ số thử nghiệm và gởi chúng đến đơn vị cần kiểm tra đồng thời nhận kết quả trả về của đơn vị cần kiểm tra.

Stub là chương trình giả lập thay thế các đơn vị được gọi bởi đơn vị cần kiểm tra. Stub thực hiện các thao tác xử lý dữ liệu đơn giản như in ấn, kiểm tra dữ liệu nhập và trả kết quả ra.



4.2 Kiểm thử tích hợp

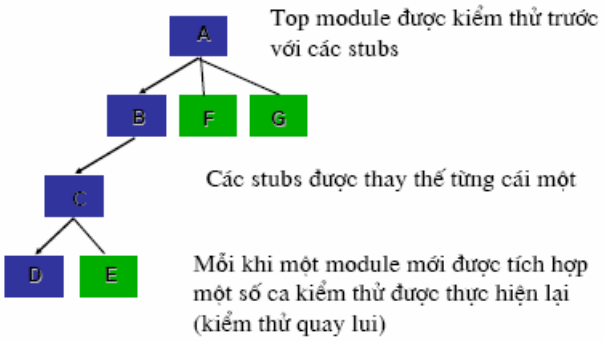
Giai đoạn này được tiến hành sau khi đã hoàn tất công việc kiểm thử từng môđun riêng lẻ bằng cách tích hợp các môđun này lại với nhau. Mục đích của giai đoạn này là kiểm tra giao diện của các đơn vị, kiểm tra tính đúng đắn so với đặc tả, kiểm tra tính hiệu quả.

Phương pháp thực hiện chủ yếu sử dụng kiểm tra chức năng. Các đơn vị có thể được tích hợp theo một trong hai chiến lược: từ trên xuống (top-down) hoặc từ dưới lên (bottom-up).

4.2.1 Trên xuống

Thuật giải của hướng tiếp cận này gồm những bước sau:

- Sử dụng Module chính như 1 driver và các stub được thay cho tất cả các module là con trực tiếp của module chính.
- Lần lượt thay thế các stub mỗi lần 1 cái bởi các module thực sự.
- Tiến hành kiểm tra tính đúng đắn.
- Một tập hợp bộ thử nghiệm được hoàn tất khi hết stub.
- Kiểm tra lùi có thể được tiến hành để đảm bảo rằng không phát sinh lỗi mới.



a) Ưu điểm

Kiểm thử trên xuống kết hợp với phát triển trên xuống sẽ giúp phát hiện sớm các lỗi thiết kế và làm giảm giá thành sửa đổi.

Nhanh chóng có phiên bản thực hiện với các chức năng chính.

Có thể thẩm định tính dùng được của sản phẩm sớm.

b) Nhược điểm

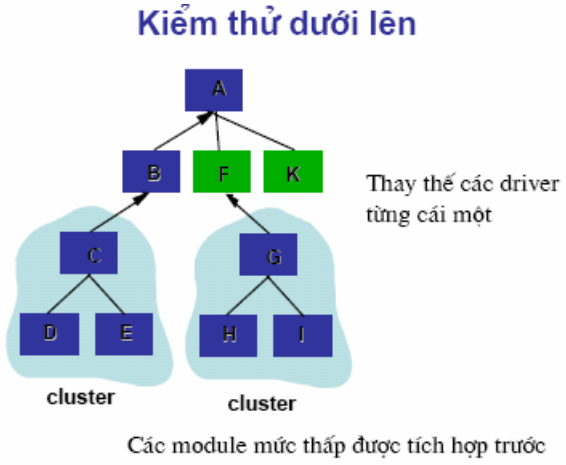
Nhiều môđun cấp thấp rất khó mô phỏng: thao tác với cấu trúc dữ liệu phức tạp, kết quả trả về phức tạp…

4.2.2 Dưới lên

Kiểm ta module lá trước do đó không cần phải viết stub.

Thuật giả của hướng này là:

- Các module cấp thấp được nhóm thành từng nhóm (thực hiện cùng chức năng)
- Viết driver điều khiển tham số nhập xuất.
- Bỏ driver và gắn chùm vào module cao hơn.



a)Ưu điểm

Tránh xây dựng các môđun tạm thời phức tạp.

Tránh sinh các kết quả nhân tạo (nhập từ bàn phím)

Thuận tiện cho phát triển các môđun để dùng lại

b) Nhược điểm

Chậm phát hiện các lỗi kiến trúc

Chậm có phiên bản thực hiện

4.3 Kiểm thử chấp nhận

Kiểm thử chấp nhận được tiến hành bởi khách hàng, còn được gọi là alpha testing. Mục đích là nhằm thẩm định lại xem phần mềm có những sai sót, thiếu sót so với yêu cầu người sử dụng không.

Trong giai đoạn này dữ liệu dùng để kiểm thử do người sử dụng cung cấp.

4.4 Kiểm thử beta

Đây là giai đoạn mở rộng của alpha testing. Công việc kiểm thử được thực hiện bởi một số lượng lớn người sử dụng.

Công việc kiểm thử được tiến hành một cách ngẫu nhiên mà không có sự hướng dẫn của các nhà phát triển. Các lỗi nếu được phát hiện sẽ được thông báo lại cho nhà phát triển.

4.5 Kiểm thử hệ thống

Đến giai đoạn này, công việc kiểm thử được tiến hành với nhìn nhận phần mềm như là một yếu tố trong một hệ thống thông tin phức tạp hoàn chỉnh.

Công việc kiểm thử nhằm kiểm tra khả năng phục hồi sau lỗi, độ an toàn, hiệu năng và giới hạn của phần mềm.

5. Ví dụ minh họa

Ví dụ 1: Phần mềm quản lý thư viện trong giai đoạn kiểm thử, các giai đoạn trước đã được trình bày ở các chương trước

Giai đoạn 6: Kiểm chứng phần mềm hướng đối tượng

③ Kiểm tra tính đúng đắn của cá lớp đối tượng

- Chuẩn bị dữ liệu thử nghiệm: Nhập dữ liệu thử nghiệm cho các bảng THU_VIEN, SACH, DOC_GIA, MUON_SACH
- Kiểm tra:

  + Kiểm tra từng lớp đối tượng:

    - Kiểm tra lớp THU_VIEN (Tra cứu độc giả, Tra cứu sách)
    - Kiểm tra lớp DOC_GIA (Lập thẻ, cho mượn sách)

- Kiểm trả lớp SACH (Nhận sách, Trả sách)

+ Kiểm tra phối hợp các lớp đối tượng

- Kiểm tra phối hợp giữa lớp THU_VIEN và lớp DOC_GIA (Lập thẻ và sau đó Tra cứu độc giả)
- Kiểm tra phối hợp giữa lớp THU_VIEN và lớp SACH (Nhận sách và sau đó Tra cứu sách)
- Kiểm tra phối hợp giữa lớp DOC_GIA và lớp SACH (Lập thẻ, Nhận sách, Cho mượn sách và Tra sách)
- Kiểm tra phối hợp giữa 3 lớp THU_VIEN, DOC_GIA và lớp SACH

Xác nhận của khách hàng: Khách hàng sử dụng phần mềm để thực hiện các công việc của mình và so sánh kết quả khi sử dụng phần mềm với kết quả khi thực hiện trong thế giới thực

Ví dụ 2: Minh họa giai đoạn kiểm chứng của phần mềm hỗ trợ giải bài tập phương trình đại số

Giai đoạn 6: Kiểm chứng phần mềm

③ Kiểm tra tính đúng đắn của các lớp đối tượng

- Chuẩn bị dữ liệu thử nghiệm: Chuẩn bị các đề tài, đáp án, bài giải, điểm số đã có trong thế giới thực và nhập điểm cho các bảng SACH_BAI_TAP, BAI_TAP, BAI_GIAI, BUOC_GIAI
- Kiểm tra:

+ Kiểm tra từng lớp đối tượng:

- Kiểm tra lớp SACH_BAI_TAP (Tra cứu bài tập)
- Kiểm tra lớp BAI_TAP (Soạn đề, Phát sinh đề, Soạn đáp án, Giải bài tập, Xem đáp án, Chấm điểm

- Ghi chú: Cần phải kiểm tra từng công việc và sau đó kiểm tra phối hợp giữa các công việc

+ Kiểm tra phối hợp các lớp đối tượng: Kiểm tra phối hợp giữa các lớp SACH_BAI_TAP và lớp BAI_TAP (Soạn đề thi và sau đó tra cứu bài tập

③ Xác nhận của khách hàng: Khách hàng sử dụng phần mềm để thực hiện các công việc của mình và so sánh kết quả khi sử dụng phần mềm với kết quả khi thực hiện trong thế giới thực.

Chương 8: SƯU LIỆU

6. Tổng quan

Chương này xem xét sưu liệu là một phần của hệ thống phần mềm. Cấu trúc của sưu liệu người dùng và hệ thống được mô tả và điều quan trọng của việc tạo ra những sưu liệu đạt chất lượng phải được nhấn mạnh. Phần cuối của chương này chú ý đến khả năng bảo trì, tính khả chuyển của sưu liệu.

Có hai lớp sưu liệu kết hợp với hệ thống máy tính. Lớp sưu liệu người dùng được mô tả làm thế nào để sử dụng hệ thống và sưu liệu hệ thống được mô tả thiết kế và thực hiện hệ thống

Sưu liệu được cung cấp cùng với hệ thống có thể hữu dụng trong bất kỳ giai đoạn sống của hệ thống

Tất cả sưu liệu cần có chỉ mục hiệu quả. Một chỉ mục tốt, cho phép người dùng tìm kiếm thông tin họ cần, và đó là đặc tính hữu dụng nhất được cung cấp nhưng cũng thường là phần rối nhất trong khi tạo sưu liệu. Một chỉ mục cặn kẻ có thể làm cho sưu liệu được viết tệ có thể sử dụng được, nhưng không có chỉ mục, cho dù sưu liệu viết tốt thì không chắc người đọc sưu liệu có hiệu quả không.

7. Sưu liệu người dùng

Sưu liệu người dùng là sưu liệu mô tả chức năng của hệ thống, mà không tham chiếu đến chức năng được thực hiện như thế nào

Sưu liệu người dùng nên được cấu trúc sao cho không nhất thiết phải đọc hết tất cả sưu liệu trước khi bắt đầu dùng hệ thống. Nó phải được hợp nhất với on-line help nhưng nó quá đơn giản để in văn bản trong help như sưu liệu người dùng

Có 5 loại sưu liệu cho người dùng

- Mô tả chức năng, giải thích hệ thống có thể làm gì
- Sưu liệu cài đặt, giải thích làm thế nào để install hệ thống và chi tiết cho từng cấu hình phần cứng cụ thể
- Giới thiệu, giải thích thuật ngữ đơn giản, và làm thế nào để bắt đầu hệ thống
- Tham chiếu, mô tả chi tiết tất cả tiện ích của hệ thống, chúng được sử dụng như thế nào

- Hướng dẫn người quản trị hệ thống (nếu cần), giải thích làm thể nào để ứng xử với những trường phát sinh khi hệ thống đang sử dụng và làm thể nào để thực hiện bảo quản hệ thống như backup hệ thống

7.1. Mô tả chức năng

- Phác thảo yêu cầu của hệ thống.
- Phác thảo mục đích của người thiết kế hệ thống.
- Mô tả hệ thống có thể làm được gì?
- Mô tả hệ thống không thể làm được gì?
- Giới thiệu ví dụ minh họa nhỏ bất kỳ chỗ nào có thể.
- Vẽ sơ đồ thì tốt nhất.

7.2. Bảng Giới thiệu

- Cung cấp cái nhìn tổng quan của hệ thống.
- Cho phép người dùng quyết định nếu hệ thống phù hợp với nhu cầu của họ.
- Trình bày giới thiệu thông tin đối với hệ thống.
- Mô tả làm thế nào để bắt đầu với hệ thống và làm thế nào người thực hiện sử dụng những tiện ích chung của hệ thống.
- Bảo người dùng hệ thống làm thể nào để tránh những rắc rối khi họ làm sai.

7.3. Bảng tham khảo

- Bảng tham khảo hệ thống là Sưu liệu được định nghĩa cho cách sử dụng hệ thống.
- Bảng tham khảo nên hoàn chỉnh.
- Kỹ thuật mô tả chuẩn nên được dùng để đảm bảo độ hoàn chỉnh đạt được.
- Người viết bảng này có thể giả sử:
    - Người đọc quen với cả mô tả hệ thống và phần giới thiệu.
    - Người đọc dùng vài hệ thống và hiểu được khái niệm và thuật ngữ.
- Phần tham khảo hệ thống cũng nên mô tả:
    - Những báo cáo lỗi phát sinh trong hệ thống.
    - Những tình huống lỗi phát sinh, nếu phù hợp, hướng người dùng đến những mô tả tiện ích đã gây ra lỗi.
- Chỉ mục cặn kẽ đặc biệt quan trọng trong phần Sưu liệu.

7.4. Sưu liệu cài đặt

Sưu liệu cài đặt nên được cung cấp đầy đủ chi tiết làm thế nào để install hệ thống trong mô trường cụ thể.

Phải bao gồm mô tả thiết bị có thể đọc của máy mà hệ thống cung cấp như định dạng, mã ký tự dùng, làm thế nào thông tin được viết, và những tập tin được tạo của hệ thống.

Sưu liệu này gồm các mô tả:

- Cấu hình tối thiểu đòi hỏi để có thể chạy của hệ thống.

- Tập tin cố định phải được thiết lập.

- Làm thế nào để bắt đầu của hệ thống.

- Những tập tin phụ thuộc cấu hình phải thay đổi để thích ứng với hệ thống đối với hệ thống máy chủ cụ thể.

Hướng dẫn cho quản trị hệ thống (cho hệ thống đòi hỏi người theo dõi tương tác)

- Mô tả những thông điệp phát sinh ở màn hình hệ thống, và làm thể nào để ứng phó với những thông điệp này
- Giải thích những nhiệm vụ của người theo dõi trong duy trì phần cứng đó

Những tài liều dễ đế đọc khác

- Danh sách các tham khảo nhanh sẵn có của tiện ích của hệ thống và làm thể nào để sử dụng chúng
- Hệ thống on-line help

8. Sưu liệu hệ thống

Sưu liệu hệ thống chứa tất cả những sưu liệu mô tả quá trình thực hiện của hệ thông từ những sưu liệu đặc tả đến bản kế hoạch test cuối cùng

-Tài liệu mô tả thiết kế

- Sưu liệu mô tả thực hiện

- Sưu liệu mô tả việc kiểm thử

Sưu liệu hệ thống thì cần thiết để hiểu và bảo trì hệ thống phần mềm.

Sưu liệu nên được cấu trúc và được mô tả tổng quan hướng người đọc đến mô tả hình thức và chi tiết với mỗi khía cạnh của hệ thống.

Một trong những khó khăn của sưu liệu hệ thống là duy trì tính kiên định qua những sưu liệu khác mô tả hệ thống. Lưu vết những thay đổi, cân nhắc những sưu liệu nên được thay thế dưới những kiểm soát của hệ thống quản lý cấu hình.

Những thành phần của sưu liệu hệ thống:

1.Định nghĩa và đặc tả yêu cầu và kết hợp.

2.Trình bày đặc tả tất cả hệ thống làm thế nào những yêu cầu được phân rã thành những nhóm các chương trình tương tác với nhau. Sưu liệu này không được yêu cầu khi hệ thống được thực hiện chỉ với chương trình đơn lẻ.

3.Mỗi chương trình của hệ thống, một mô tả làm thế nào chương trình được phân rã thành những thành phần và khẳng định của những đặc tả của thành phần.

4.Mỗi đơn vị, một mô tả của những thao tác. Điều này không cần mở rộng để mô tả hoạt động của chương trình.

5.Mô tả kế hoạch kiểm thử (test plan) chi tiết làm thế nào mỗi đơn vị chương trình được kiểm thử.

6.Một kế hoạch kiểm thử chỉ ra những kiểm thử hợp nhất như là kiểm tra tất cả đơn vị chương trình kết hợp với nhau được thực hiện.

7.Một kế hoạch kiểm thử được chấp nhận, vạch ra sự nối kết những người dùng hệ thống. Tài liệu này nên mô tả những kiểm thử phải được thõa mãn trước khi hệ thống được chấp nhận.

9. Chất lượng của sưu liệu

Chất lượng của sưu liệu quan trọng như chất lượng của chương trình.

Khi không có thông tin làm thế nào sử dụng hệ thống và làm thế nào để hiểu nó thì những tiện ích của hệ thống sẽ bị giảm chất lượng.

Tạo những sưu liệu tốt không dễ dàng không rẻ và tiến trình cũng khó như tạo một chương trình tốt.

Tiêu chuẩn sưu liệu nên được mô tả chính xác như sưu liệu bao gồm những gì và nên mô tả hệ thống các ký hiệu dùng trong sưu liệu.

Với một tổ chức, cần thiết lập một chuẩn cho sưu liệu và yêu cầu tất cả các sưu liệu phải tuân thủ theo những định dạng đó.

Những tiêu chuẩn của sưu liệu có thể bao gồm:

+ Mô tả định dạng trước được chấp bởi tất cả tài liệu
+ Đánh số trang và cách thức ghi chú trang.
+ Phương thức tham khảo tài liệu khác

+ Số đề mục và đề mục con

Phong cách viết là yếu tố nền tảng ảnh hưởng đến chất lượng của sưu liệu và đó là khả năng của người viết để xây dưng một cách rõ ràng kỹ thuật soạn thảo chính xác. Một số cách viết nên tránh như dùng câu quá dài, mô tả phức tạp, lặp lại, thông tin tham chiếu chỉ toàn là số không những chi tiết gơi nhớ cho người đọc v.v

10. Bảotrì sưu liệu

Bởi vì hệ thống phần mềm được cập nhật, sưu liệu kết hợp với hệ thống cũng phải được cập nhật tướng ứng với những thay đổi của hệ thống.

Tất cả những sưu liệu kết hợp nên được cập nhật khi một thay đổi được làm bởi chương trình. Giả sử những thay đổi này được nhìn thấy bởi người dùng, chỉ mô tả thực hiện hệ thống cần phải thay đổi. Nếu hệ thống thay đổi nhiều hơn sự chính xác của lỗi chương trình thì điều này có nghĩa là xem xét lại sưu liệu thiết kế và kiểm thử và có thể sưu liệu ở thiết kế mức cao mô tả đặc tả và yêu cầu.

Một trong những vấn đề chính của bảo trì sưu liệu là lưu những thể hiện khác nhau của hệ thống từng bước với nhau. Giải pháp tốt nhất cho vấn đề này là hỗ trợ bảo trì sưu liệu với công cụ phần mềm mà ghi nhận mối liên hệ sưu liệu, nhắc nhỡ những kỹ sư phần mềm khi thay đổi một sưu liệu có tác động đến sưu liệu khác, và ghi nhận những thay đổi trong sưu liệu

Nếu sự thay đổi của hệ thống tác động giao diện người dùng một cách trực tiếp hoặc thêm mới một tiện ích hoặc mở rộng tiện ích tồn tại.

11. Các mẫu sưu liệu cho qui trình làm phần mềm

11.1. Xác định yêu cầu (SRS)

Software Requirements Specifications (w/o Use Cases)

Chuẩn IEE 830-1984

1. Giới thiệu

    1.1 Mục đích

    1.2 Phạm vi

    1.3 Định nghĩa (định nghĩa, từ viết tắt)

    1.4 Tài liệu tham khảo

1.5 Mô tả cấu trúc tài liệu

2. Mô tả chung

2.1 Tổng quan về sản phẩm

2.2 Chức năng sản phẩm

2.3 Đối tượng người dùng

2.4 Ràng buộc tổng thể

2.5 Giả thiết và sự lệ thuộc

3. Yêu cầu chi tiết

3.1 Yêu cầu chức năng

3.1.1 Yêu cầu chức năng 1

3.1.1.1 Giới thiệu

3.1.1.2 Dữ liệu vào

3.1.1.3 Xử lý

3.1.1.4 Kết quả

3.1.2 Yêu cầu chức năng 2

3.1.n Yêu cầu chức năng n

…

b. Thiết kế

Sưu liệu cho giai đoạn thiết kế có các mẫu thiết kế sau:

- Thiết kế cơ sở dữ liệu (Database Design)
- Thiết kế ràng buộc (Design Criteria)
- Sưu liệu kiến trúc phần mềm (Software Architecture Document)
- Thiết kế Thành phần (Components Design)

11.2. Mô tả thiết kế phần mềm (SDD)

Software Design Descriptions Chuẩn IEEE 1016-1998

1. Introduction (Giới thiệu)

- Purpose (mục đích)

- Scope (Phạm vi)
- Definitions, acronyms, and abbreviations (Định nghĩa, viết tắt)

2 References (Tham khảo)

3 Decomposition description (Mô tả phân rã)

4 Dependency description (Mô tả phụ thuộc)

5 Interface description (mô tả giao diện)

6 Detailed design (thiết kế chi tiết)

11.3. System Design Rationale Document (SDRD)

1. Introduction (Giới thiệu)

    1.1 Purpose of the document (Mục đích của sưu liệu)

    1.2 Design goals (Mục tiêu của thiết kế đạt được)

    1.3 Definitions, acronyms, and abbreviations

    1.4 References (Tham khảo)

    1.5 Overview (Tổng quan)

2. Rationale for Current Software Architecture

3. Rationale for Proposed Software Architecture

    3.1 Overview (Tổng quan)

    3.2 Rationale for Subsystem decomposition

    3.3 Rationale for Hardware/software mapping

    3.4 Rationale for Persistent data management

    3.5 Rationale for Access control and security

    3.6 Rationale for Global software control

    3.7 Rationale for Boundary conditions

4. Subsystem Services

Glossary

Phụ Lục A

1. Câu hỏi lý thuyết

Chương (1->4)

1. Trình bày sự khác biệt của giai đoạn thiết kế trong các qui trình khác nhau
2. Trình bày sự khác biệt của giai đoạn lập trình trong các qui trình khác nhau
3. Khi tiến hành thực hiện phần mềm qua các giai đoạn (trong qui trình 5 giai đoạn) có thể phát sinh lỗi trong một giai đoạn nào đó (kết quả chuyển giao không chính xác, thiếu sót, v.v…). Theo các anh chị lỗi (nếu phát sinh) của giai đoạn nào là nghiêm trọng nhất
4. Theo các anh chị trong các giai đoạn của qui trình công nghệ phần mềm

- Giai đoạn nào là quan trọng nhất (tại sao)
- Giai đoạn nào dễ thực hiện nhất (tại sao)
- Giai đoạn nào là tốn nhiều thời gian và chi phí nhất (tại sao)
- Giai đoạn nào là có thể bỏ qua (trong trường hợp nào và tại sao)

Chương 2 (5-10)

5. Cho biết sự khác biệt cơ bản giữa yêu cầu chức năng (yêu cầu nghiệp vụ, yêu cầu hệ thống) và phi chức năng (yêu cầu chất lượng). Theo anh chị thì loại yêu cầu nào là quan trọng hơn
6. Xác định tất cả các yêu cầu chức năng hệ thống có thể có trong các phần mềm sau (chi tiết về qui định, biểu mẫu liên quan có trong mô tả của đề tài)
    1) Phần mềm quản lý bán sách
    2) Phần mềm quản lý học sinh trường cấp 3
    3) Phần mềm đánh cờ gánh
    4) Phần mềm hỗ trợ giải bài tập phương trình đại số
    5) Phần mềm quản lý giải vô địch bóng đá quốc giá
7. Nhận xét về phát biểu sau “Mọi phần mềm đều có yêu cầu về tính tiện dụng”.

- Nếu đúng: giải thích
- Nếu sai: giải thích và ví dụ minh họa

8. Nhận xét về phát biểu sau: “Mọi phần mềm đề có yêu cầu về tính hiệu quả

- Nếu đúng: giải thích

- Nếu sai: giải thích và cho ví dụ minh họa

9. Nhận xét về phát biểu sau: "Mọi phần mềm đề có yêu cầu chức năng hệ thống

- Nếu đúng: giải thích
- Nếu sai: giải thích và cho ví dụ minh họa

Chương (3-14)

10. Nhận xét về phát biểu sau "Việc mô hình hóa yêu cầu không cung cấp thêm thông tin mới về yêu cầu của phần mềm mà chỉ giúp trình bày lại yêu cầu của phần mềm dưới dạng trực quan hơn

- Nếu đúng: giải thích
- Nếu sai: giải thích và cho ví dụ minh họa

11. Nếu không thực hiện qua bước mô hình hóa yêu cầu thì việc lập mô hình đối tượng sẽ có các khó khăn gì? tại sao?

12. Cho biết các kết quả của việc mô hình hóa yêu cầu có được sử dụng trong bước thiết kế giao diện của đối tượng hay không ?

- Nếu đúng: giải thích
- Nếu sai: giải thích và cho ví dụ minh họa

13. Cho biết các kết quả của việc mô hình hóa yêu cầu có được sử dụng trong bước xác định thuộc tính đối tượng (giai đoạn thiết kế) hay không

- Nếu đúng: giải thích
- Nếu sai: giải thích và cho ví dụ minh họa

14. Cho biết các kết quả của việc mô hình hóa yêu cầu có được sử dụng trong bước xác định hàm xử lý của đối tượng (giai đoạn thiết kế) hay không

- Nếu đúng: giải thích
- Nếu sai: giải thích và cho ví dụ minh họa

2. Nội dung và yêu cầu bài tập

2.1 Quản lý thuê bao điện thoại

Lưu trữ: Các thông tin về

- Các hợp đồng thuê bao điện thoại (Khách hàng, loại thuê bao, máy điện thoại)
- Các cuộc gọi (Máy điện thoại, Ngày, Giờ, Thời gian, Nơi gọi đến).

Tính toán:

- Số tiền phải trả của từng máy điện thoại trong từng tháng:
  - Tiền thuê bao hàng tháng (phụ thuộc vào từng loại thuê bao với các định mức riêng).
  - Tiền cước phí trả thêm (hụ thuộc vào thời gian gọi, số phút gọi, nơi gọi đến)
- Tính công nợ khách hàng đối với các khách hàng chưa thanh toán tiền điện thoại.

Kết xuất:

- Hóa đơn tính tiền điện thoại cho từng khách hàng trong từng tháng.
- Danh sách khách hàng chưa thanh toán tiền điện thoại.
- Thống kê về nơi gọi đến, thời điểm gọi theo từng khu vực trong từng tháng.

2.2 Quản lý học sinh trường phổ thông trung học

Lưu trữ: Các thông tin về

- Học sinh: Họ, tên, lớp, ngày sinh, giới tính, địa chỉ, thành phần, kết quả học tập, điểm danh.

Tra cứu: Thông tin về học sinh

Tính toán:

- Điểm trung bình từng môn học theo từng học kỳ: Tính theo các điểm của từng hình thức kiểm tra (15 phút: hệ số 1, 1 tiết hệ số 2, thi học kỳ: hệ số 3)
- Điểm trung bình học kỳ 1, học kỳ 2, cả năm (học kỳ 1 hệ số 1, học kỳ 2 hệ số 2) .
- Xếp loại: Xuất sắc nếu điểm trung bình niên khóa [redacted].0 và không có môn nào có điểm trung bình dưới 7.5. Tiên tiến nếu điểm trung bình niên khóa [redacted].5 và không có môn nào có điểm trung bình dưới 6.0. Đạt yêu cầu nếu điểm trung bình niên khóa [redacted] 5 và không có môn nào có điểm trung bình dưới 5. Không đạt yêu cầu nếu có ít nhất 1 môn dưới 5.

Ghi chú: Nếu tổng số ngày vắng vượt quá 20 sẽ bị xếp vào loại không đạt yêu cầu. Nếu số ngày vắng vượt quá 10 hay số ngày vắng không phép vượt quá 5 thì sẽ bị hạ xuống một bậc (chỉ áp dụng với loại xuất sắc và tiên tiến).

Kết xuất:

- Danh sách học sinh theo từng lớp.

- Phiếu điểm cho mỗi học sinh.
- Bảng điểm các môn và bảng điểm tổng kết cho từng lớp.
- Thống kê về xếp loại học sinh của tòan trường trong 1 niên khóa.

2.2 Quản lý các tài khoản trong ngân hàng

Lưu trữ:

- Các tài khoản: Khách hàng, loại tài khỏan, số tiền, loại tiền, ngày gởi, tình trạng
- Quá trình gửi và rút tài khỏan: Khách hàng, ngày số tiền, hình thức.
- Các qui định về lãi suất và tỷ giá.

Tra cứu: Tài khoản theo các tiêu chuẩn

- Mã số
- Khách hàng
- Loại tài khoản
- Ngày mở, ngày đóng.

Tính toán:

- Lãi suất cho từng tài khoản khi đến kỳ hạn hay khi khách hàng rút trước kỳ hạn (chỉ được không kỳ hạn).

Kết suất:

- Danh sách các biến động trên 1 tài khoản
- Danh sách tài khoản cùng số dư hiện tại theo từng loại tài khoản.
- Tình hình gửi, rút tiền theo từng loại tài khoản.
- Số dư của ngân hàng theo từng ngày của tháng.

2.3 Theo dõi kế hoạch sản lượng cao su

Lưu trữ: Các thông tin về

- Nông trường: Tên, diện tích các lô cao su theo từng năm.
- Sản lượng kế hoạch theo tháng, năm của từng loại mủ.
- Sản lượng thực tế theo ngày của từng loại mủ.

Tính toán:

- Tỷ lệ đạt của từng loại mủ theo từng nông trường theo kế hoạch.
- Kế hoạch dự kiến cho năm tới.

Kết xuất:

- Báo cáo nhanh hàng ngày.
- Báo cáo tháng.
- Kế hoạch năm cho từng nông trường cho từng loại mủ.

2.4 Quản lý giải vô địch bóng đá

Lưu trữ: Các thông tin về

- Các đội bóng tham gia giải: Tên đội bóng, tên huấn luyện viên, các cầu thủ, sân nhà.
- Lịch thi đấu: đội tham dự, sân, thời gian
- Kết quả các trận đấu: Trọng tài, tỷ số, khán giả, các cầu thủ ra sân của 2 đội cùng vị trí tương ứng, việc ghi bàn, phạt thẻ.

Tra cứu: Cầu thủ, đội bóng

Tính toán:

- Tính điểm cho từng đội: mỗi trận thắng được 3 điểm, mỗi trận hòa được 1 điểm, mỗi trận thua được 0 điểm.
- Xếp hạng cho từng đội: Dựa vào các tiêu chuẩn: tổng số điểm, tổng số bàn thắng, hiệu số, đối kháng trực tiếp, bốc thăm.

Kết xuất:

- Danh sách các cầu thủ theo từng đội, vị trí.
- Lịch thi đấu.
- Bảng xếp hạng các đội bóng.
- Tổng kết việc ghi bàn của giải.
- Tình hình phạt thẻ các đội bóng.

2.4 Thi trắc nghiệm trên máy tính

Lưu trữ: Các thông tin về

- Thí sinh dự thi: Họ và tên, môn thi, ngày thi, địa chỉ, đề thi, bài làm, phòng thi.
- Câu hỏi trắc nghiệm: Nội dung câu hỏi, các cầu trả lời có thể có, đáp án, mức độ khó, thang điểm, môn tương ứng.

Tính toán:

- Phát sinh các đề thi tương đương cho một đề thi đã chọn cho một môn thi nào đó (đề thi tương có cùng các câu hỏi trắc nghiệm nhưng có số thứ tự khác nhau và trật tự các câu trả lời cũng khác nhau).
- Tính điểm thi cho từng thí sinh: Tổng điểm các câu hỏi với thang điểm tương ứng.

Kết xuất:

- Danh sách các thí sinh theo từng phòng thi.
- Đề thi.
- Bài làm của từng thí sinh cùng với điểm số.
- Danh sách kết quả thi của mỗi môn thi.
- Thống kê kết quả thi theo từng mức theo từng môn thi.
- Thống kê kết quả thi theo từng câu hỏi.

2.5 Quản lý trung tâm giới thiệu việc làm sinh viên

Lưu trữ: Các thông tin về

- Sinh viên đăng ký tìm việc: Họ và tên, ngày sinh, địa chỉ, tình hình sức khỏe, quá trình học tập và bằng cấp, các công việc có thể đảm nhận, các yêu cầu khi tìm việc.
- Đơn vị đăng ký tìm người: Tên, địa chỉ, người đại diện, các công việc cùng yêu cầu tuyển dụng.
- Giới thiệu việc làm: Sinh viên, đơn vị, công việc, tình trạng.

Tra cứu:

Sinh viên tra cứu công việc

- Loại công việc.
- Mức lương.
- Hình thức làm việc.
- Nơi làm việc

Đơn vị tuyển dụng tra cứu các sinh viên

- Bằng cấp chuyên môn.
- Sức khỏe.
- Phương tiện làm việc.

Tính toán:

- Các công việc thích hợp cho sinh viên đăng ký làm việc.

- Các sinh viên thích hợp cho công việc cần tuyển dụng của 1 đơn vị.

Kết xuất:

- Danh sách sinh viên đăng ký theo từng công việc.
- Danh sách số lượng sinh viên đăng ký theo từng loại công việc.
- Danh sách các đơn vị tuyển dụng theo từng công việc.
- Danh sách số lượng đơn vị tuyển dụng theo từng công việc.
- Thống kê tình hình giới thiệu việc làm thực hiện trong năm.

c. Phần mềm quản lý bán sách

Khảo sát thực tế và rút ra yêu cầu cần phải làm cho đề tài

d. Phần mềm quản lý bán vé chuyến bay

Khảo sát thực tế và rút ra yêu cầu cần phải làm cho đề tài

e. Phần mềm quản lý phòng mạch

Khảo sát thực tế và rút ra yêu cầu cấn phải làm cho đề tài

3. Bài tập nâng cao

3.1 Đăng ký môn học và học phí

Một trường đại học có nhu cầu tin học hóa khâu quản lý việc đăng ký môn học và học phí của sinh viên. Một sinh viên sau khi hoàn thành thủ tục nhập học phải cho biết họ và tên, ngày sinh, giới tính, quê quán gồm tên huyện và tên tỉnh. Nếu sinh viên thuộc đối tượng (con liệt sỹ, con thương binh, con gia đình có công với nước, vùng sâu, vùng xa,) thì phải có xác nhận của địa phương. Mỗi đối tượng có một tỷ lệ tương ứng về việc giảm học phí. Để thuận tiện trong việc quản lý người ta gán cho mỗi sinh viên một mã số gọi là mã số sinh viên, mã số này là duy nhất, không thay đổi trong suốt quá trình sinh viên học tại trường. Căn cứ ngành học mà sinh viên thi đậu vào mà sinh viên đó sẽ thuộc sự quản lý của một khoa nào đó: nghĩa là mỗi sinh viên thuộc một ngành, và một khoa có thể gồm nhiều ngành học khác nhau; dĩ nhiên không tồn tại một ngành thuộc sự quản lý của hai khoa khác nhau.

Vào đầu học kỳ mới sinh viên đến phòng Giáo vụ đăng ký các môn học. Việc đăng ký môn học được thể hiện qua một phiếu đăng ký. Trên phiếu đăng ký có một số phiếu, thông tin về sinh viên (mã số, họ và tên), ngày đăng ký, học kỳ và niên khóa đăng ký. Một phiếu đăng ký có thể có nhiều môn học (mã môn, tên môn và số đơn vị học trình tương ứng của môn đó).

Tất nhiên là các môn học đó sẽ được dạy trong học kỳ cho sinh viên đăng ký mà phòng Giáo vụ đã có kế hoạch trong thời khóa biểu đã thông báo cho sinh viên biết trước khi đăng ký.

Mỗi môn học ngoài việc định danh bằng tên còn kèm theo số tín chỉ học trình và được gán cho một mã số môn học. Số tín chỉ của mỗi môn học tùy thuộc vào thời gian giảng dạy (thường 15 tiết lý thuyết hoặc bài tập hay 30 tiết thực hành tương đương 1 tín chỉ). Để đơn giản người ta phân thành hai loại môn: môn lý thuyết (hoặc bài tập) và môn thực hành. Nếu đăng ký môn lý thuyết sinh viên sẽ phải trả 27000 đồng/ tín chỉ, còn với môn thực hành là 37000 đồng/tín chỉ. Có một số môn, muốn đăng ký học, sinh viên phải học và đạt trên điểm trung bình một số môn trước để làm cơ sở cho việc học môn đó (gọi là các môn tiên quyết của môn học đó). Mỗi ngành học bao gồm một hệ thống nhiều môn mà sinh viên thuộc ngành đó phải theo học nằm trong nội dung chương trình giảng dạy của ngành đó; có thể có nhiều môn thuộc chương trình giảng dạy của nhiều ngành học khác nhau. Mỗi học kỳ, căn cứ vào việc đăng ký các môn học và đối tượng của từng sinh viên mà người ta xác định được số tiền học phí mà mỗi sinh viên sẽ phải đóng.

Sau khi đăng ký xong các môn học, sinh viên phải đến Phòng Tài vụ của trường để đóng học phí. Mỗi lần khi một sinh viên đến nộp học phí, một phiếu thu được lập, trên đó ghi nhận mã số sinh viên, ngày lập, số tiền mà sinh viên đóng và được đánh số thứ tự để tiện việc theo dõi. Mỗi phiếu thu chỉ thu tiền học phí của một sinh viên tại một học kỳ. Một phiếu thu được in thành hai liên, một liên gửi cho sinh viên như một biên lai, liên còn lại để lưu. Nhân viên của Phòng Tài vụ lập phiếu phải nhận tiền học phí của sinh viên để cuối buổi nộp cho thủ quỹ. Mỗi học kỳ, nhà trường khống chế thời điểm cuối cùng (một ngày nào đó) mà sinh viên phải hoàn thành thủ tục trên, nếu quá hạn đó phòng Tài vụ khóa sổ không thu nữa, và như vậy những sinh viên không đóng, không kịp đóng hoặc đóng không đủ học phí sẽ không được tham dự kỳ thi cuối học kỳ đó. Mỗi học kỳ, sau khi cho sinh viên đăng ký môn học, để khuyến khích sinh viên đóng học phí sớm nhà trường cũng qui định một ngày mà nếu sinh viên đóng học phí trước ngày đó sẽ được giảm một tỷ lệ nào đó (thường là 5% số tiền học phí mà sinh viên phải đóng cho học kỳ đó). Mỗi học kỳ sinh viên có thể đóng học phí làm nhiều lần tùy theo tình hình tài chính của mình và phải đóng trước ngày hết hạn đóng học phí của học kỳ đó.

Khi hết hạn đóng học phí Phòng Tài vụ sẽ tổng kết số tiền học phí mà mỗi sinh viên đã đóng, kết hợp với số tiền học phí mà sinh viên phải đóng xác định danh sách những sinh viên đang còn nợ học phí của học kỳ đó để gửi cho bộ phận quản lý của Phòng Giáo vụ loại những sinh viên đó ra khỏi danh sách dự thi.

3.2 Quản lý đồ án – Niên luận

Bộ môn Hệ thống thông tin và toán ứng dụng khoa Công Nghệ Thông Tin muốn quản lý tất cả các đồ án - niên luận của sinh viên tin học chính quy cũng như tại chức. Để dễ dàng trong việc quản lý, ngay sau khi vào trường mỗi sinh viên ngoài họ tên, ngày sinh, giới tính đều được gán một mã số gọi là mã số sinh viên. Sinh viên chính quy thuộc sự quản lý của trường còn đối với sinh viên tại chức sẽ thuộc sự quản lý của một đơn vị đào tạo (thường là trung tâm giáo dục thường xuyên) của một tỉnh nào đó.

Trong chương trình đào tạo sinh viên phải thực hiện một số loại đồ án (niên luận 1 - lập trình chuyên ngành, niên luận 2 - lập trình quản lý, niên luận 3 – lập trình ứng dụng, tiểu luận tốt nghiệp, và luận văn tốt nghiệp cho một số sinh viên xuất sắc khi ra trường). Mỗi loại đồ án - niên luận có một số đơn vị học trình tương ứng gọi là số tín chỉ.

Theo chương trình học, đến kỳ triển khai đồ án - niên luận bộ môn yêu cầu các giáo viên ra đề tài cho sinh viên chọn. Mỗi một đề tài giáo viên yêu cầu những điều mà sinh viên sẽ phải làm, cung cấp các tài liệu để sinh viên tham khảo. Sau khi giáo viên nộp đề tài bộ môn sẽ gán cho mỗi đề tài một mã số. Việc định danh (đặt tên) do các giáo viên ra đề tài quyết định. Mỗi đề tài chỉ thuộc một loại đồ án - niên luận duy nhất, và được ra bởi ít nhất một giáo viên trong bộ môn.

Mỗi một giáo viên được nhận biết qua mã số giáo viên, họ tên, ngày sinh, phái và một chức danh. Mỗi chức danh có một hệ số chức danh, và căn cứ vào chức danh này để sau này tính tiền cho giáo viên ra đề tài hay giáo viên hướng dẫn đồ án - niên luận.

Đến học kỳ mà sinh viên phải thực hiện loại đồ án nào đó, bộ môn sẽ triển khai việc thực hiện đồ án - niên luận cho sinh viên. Trước hết bộ môn cung cấp danh sách các đề tài mà các giáo viên đã ra thuộc loại đó đểø sinh viên lựa chọn thực hiện. Đối với các loại niên luận, tiểu luận, các sinh viên tự lập nhóm, tối đa hai sinh viên một nhóm, nhóm này chọn làm chung một quyển đồ án và một quyển đồ án như vậy làm về một đề tài duy nhất trong danh sách các đề tài được bộ môn cung cấp. Riêng trường hợp đối với luận văn tốt nghiệp, chỉ có một số sinh viên xuất sắc được chọn và mỗi sinh viên làm một đồ án tốt nghiệp riêng rẽ.

Sau khi sinh viên lựa chọn đề tài, bộ môn sẽ phân công giáo viên hướng dẫn cho từng nhóm sinh viên làm chung một đề tài và viết chung một quyển đồ án - niên luận. Nói chung giáo viên ra đề tài là người hướng dẫn những sinh viên thực hiện đề tài đó, tuy nhiên có khi giáo viên ra đề tài bận đi công tác, bộ môn có thể cử người khác hướng dẫn. Đến hạn sinh viên phải hoàn thành và nộp các quyển đồ án. Quyển đồ án phải được soạn theo mẫu mà bộ môn đã quy định để dễ dàng trong việc quản lý và đánh giá. Cán bộ trực bộ môn phải chịu

trách nhiệm thu nhận các quyển đồ án mà sinh viên nộp. Để đơn giản trong quản lý, mỗi quyển đồ án - niên luận được cán bộ trực bộ môn gán cho một số thứ tự, ghi nhận lại ngày mà sinh viên nộp.

Ngay sau ngày hết hạn nộp trưởng hoặc phó bộ môn sẽ phân công giáo viên đánh giá và chấm điểm cho từng quyển đồ án. Bộ môn cũng yêu cầu các giáo viên nộp kết quả đúng kỳ hạn để tổng kết điểm. Các sinh viên thực hiện chung một đề tài sẽ được chung một điểm kết quả qua sự cho điểm đó. Khi đến hạn, bộ môn sẽ tổng kết điểm, lập danh sách báo cáo cho phòng Giáo vụ.

Cuối học kỳ bộ môn tổng kết số đề tài mà mỗi giáo viên đã ra (mà được sinh viên chọn làm đồ án - niên luận), số đồ án - niên luận mà mỗi giáo viên đã hướng dẫn, đã chấm để làm cơ sở cho việc tính tiền giảng dạy.

3.3 Quản lý cơ sở sản xuất và chất lượng sản phẩm

Chi cục tiêu chuẩn đo lường chất lượng sản phẩm một tỉnh cần quản lý chất lượng các sản phẩm của những cơ sở sản xuất trong tỉnh.

Trên địa bàn tỉnh quản lý có nhiều cơ sở sản xuất. Để thuận tiện trong quản lý người ta gán mỗi cơ sở một mã số cơ sở duy nhất. Mỗi một cơ sở có một địa chỉ, một người chịu trách nhiệm gọi là chủ cơ sở, được biết bằng họ và tên, có thể không có, có một hoặc có một vài số điện thoại để tiện liên hệ.

Cơ sở muốn sản xuất một mặt sản phẩm nào phải đăng ký thông qua một phiếu đăng ký chất lượng cho nó. Một phiếu đăng ký có một số đăng ký hay số thứ tự và chỉ cấp cho một sản phẩm duy nhất, tuy nhiên một cơ sở sản xuất có thể đăng ký nhiều sản phẩm khác nhau. Mỗi phiếu đăng ký có một thời hạn (từ một ngày đến một ngày nào đó) và số lượng đăng ký sẽ sản xuất trong thời hạn đó. Mỗi sản phẩm được gán cho một mã số sản phẩm, một định danh rõ ràng và một đơn vị tính tương ứng. Một sản phẩm thường phải đăng ký nhiều chỉ tiêu, mỗi chỉ tiêu có một đơn vị tính cho chỉ tiêu đó, và khi đăng ký thì chỉ số đăng ký cho chỉ tiêu tương ứng là bao nhiêu.

Trong thời hạn đăng ký, về nguyên tắc sản phẩm đã đăng ký sản xuất được bán trên thị trường phải bảo đảm các chỉ tiêu đã đăng ký. Theo định kỳ hoặc có gì nghi vấn chi cục tiêu chuẩn đo lường chất lượng sản phẩm sẽ bốc mẫu sản phẩm của cơ sở về để kiểm nghiệm, đánh giá. Khi đánh giá xong một phiếu kiểm nghiệm được lập. Một phiếu kiểm nghiệm chỉ kiểm một sản phẩm theo một số chỉ tiêu với chỉ số kiểm nghiệm tương ứng. Hơn nữa một phiếu kiểm nghiệm có một số thứ tự, ngày đánh giá và chỉ dùng cho một cơ sở duy nhất đã

sản xuất sản phẩm đã đăng ký đó. Dựa vào kết quả kiểm nghiệm mà người có trách nhiệm cho đánh giá là đạt hay không đạt chất lượng theo mức đăng ký. Sản phẩm của cơ sở nào không đạt chất lượng sẽ không được phép tiếp tục sản xuất và lưu hành trên thị trường, và bị rút giấy phép kinh doanh. Nếu sản phẩm gây nguy hại cho người dùng thì chủ cơ sở có thể bị truy tố trước pháp luật.

Đến lúc nào đó cục tiêu chuẩn đo lường chất lượng sản phẩm muốn biết các sản phẩm của cơ sở nào hết thời hạn đăng ký, những sản phẩm nào không đạt chất lượng, vv...

3.4 Quản lý lương sản phẩm

Một công ty sản xuất muốn quản lý tiền lương của tất cả các nhân viên. Các nhân viên thuộc hai loại: nhân viên hành chánh và công nhân. Mỗi một nhân viên có một mã số, họ tên, phái, ngày sinh, và ngày bắt đầu tham gia công tác. Mỗi nhân viên sẽ thuộc một đơn vị quản lý nào đó.

Đối với công nhân hưởng lương sản phẩm. Các sản phẩm này thường được các công ty khác đặt hàng thông qua một hợp đồng với một số lượng tương ứng cùng những yêu cầu về kỹ thuật và thẩm mỹ kèm theo. Một sản phẩm có một mã số và mang một tên để gọi và đơn vị tính của nó.

Các hợp đồng được đánh số thứ tự, tên hợp đồng, ngày bắt đầu và ngày kết thúc. Một hợp đồng ít nhất về một sản phẩm, nếu liên qua đến nhiều sản phẩm thì tất cả các sản phẩm này đều cùng kết thúc cùng một thời điểm ghi trên hợp đồng để giao hàng và thanh lý hợp đồng.

Quá trình sản xuất một sản phẩm gồm nhiều công đoạn tùy theo sản phẩm. Do đặc tính kỷ thuật, thẩm mỹ và môi trường làm việc mà mỗi công đoạn được trả một đơn giá tương ứng. Các công đoạn sản xuất một sản phẩm được gọi bằng tên công đoạn và thường được đánh số thứ tự.

Hàng ngày, bộ phận quản lý sẽ ghi nhận kết quả làm việc của công nhân ngày hôm trướùc do đơn vị sản xuất báo lên. Kết quả làm việc của mỗi công nhân trong ngày thể hiện việc công nhân đó thực hiện được những công đoạn nào của sản phẩm được hợp đồng với số lượng tương ứng của công đoạn đó là bao nhiêu trong ca làm việc nào. Làm việc ở ca 3 hoặc các ca của ngày chủ nhật được hưởng thêm một hệ số cao hơn làm việc các ca khác trong ngày làm việc bình thường. Kết quả này sẽ xác định thu nhập của công nhân trong ngày hôm đó.

Đối với việc tính lương cho nhân viên hành chánh căn cứ vào hệ số lương và số ngày làm việc trong tháng của người đó. Nếu nghỉ có lý do (bệnh đột xuất, thai sản, ...) sẽ được hưởng tiền bảo hiểm xã hội tùy theo số ngày nghỉ có lý do trong tháng. Nếu nghỉ không lý do thì không được tính lương. Hệ số lương thường căn cứ vào trình độ chuyên môn, trình độ ngoại ngữ, và thâm niên công tác và do lãnh đạo công ty xem xét và quyết định. Đối với những người có đảm trách chức vụ thì được hưởng phụ cấp chức vụ tùy theo đặc thù của chức vụ. Do nhu cầu của công tác, có thể các nhân viên hành chánh có thể làm việc ngoài giờ. Bộ phân theo dõi lương sẽ tổng kết số buổi làm thêm ngoài giờ của từng nhân viên trong tháng để tính lương ngoài giờ cho nhân viên.

3.5 Quản lý công tác thực hành tin học

Khoa Công nghệ thông tin muốn quản lý công tác thực hành tin học của các phòng thực hành. Khoa có nhiều phòng máy tính phục vụ các môn học thực hành và làm niên luận, luận văn cho sinh viên. Mỗi phòng có số phòng, cùng hệ thống các máy tính trong đó. Các máy tính được đánh số và có thể có cấu hình (các phụ tùng: Mainboard, Ram, Harddisk, ... với đặc tính kỷ thuật liên quan) khác nhau. Mỗi phòng thực hành do một cán bộ phụ trách. Người ta quan tâm đến họ tên, phái, ngày sinh, địa chỉ của cán bộ và để cho đơn giản người ta cho mỗi cán bộ một mã số để phân biệt.

Dựa vào việc đăng ký các môn thực hành của sinh viên vào đầu học kỳ mà phòng Giáo vụ chuyển danh sách cho, trợ lý giáo vụ của khoa sẽ phân thành các nhóm thực hành. Các sinh viên cùng một nhóm sẽ có cùng một lịch thực hành. Lịch thực hành của mỗi môn học tại một học kỳ được bố trí thành các buổi tại các phòng thực hành. Mỗi buổi thực hành chỉ dành cho một môn thực hành của một nhóm nào đó. Phòng Giáo vụ dựa vào việc đăng ký môn học đầu học kỳ của sinh viên mà cung cấp danh sách các nhóm thực hành cho từng môn, căn cứ vào đó cán bộ coi thi thực hành điểm danh và kiểm tra.

Khi tiến hành mỗi buổi thực hành, cán bộ phụ trách sẽ bố trí vị trí của sinh viên (ngồi vào máy nào của phòng máy). Nói chung sinh viên tham dự các buổi thực hành theo lịch thực hành mà trợ lý giáo vụ hay trưởng phòng thí nghiệm đã sắp xếp. Cũng như đối với cán bộ, người ta quan tâm đến họ tên, phái, ngày sinh, địa chỉ của sinh viên và để cho đơn giản người ta cho mỗi sinh viên một mã số gọi lã mã sinh viên để phân biệt. Những thông tin về sinh viên được ghi nhận tại Phòng Giáo vụ khi sinh viên nhập học sau khi trúng tuyển qua kỳ tuyển sinh.

Một buổi thực hành tại một phòng máy chỉ thực hành một môn học nào đó. Chú ý là một ngày làm việc có thể có 3 buổi thực hành (sáng, chiều, và tối). Sau khi trợ lý giáo vụ công

bố lịch thực hành, bộ môn sẽ phân công cán bộ giảng dạy hướng dẫn sinh viên thực hiện các bài tập cho thực hành này. Cùng một môn nhưng có thể có nhiều cán bộ coi thực hành tại một buổi thi.

Xong mỗi đợt thực hành cán bộ phụ trách phòng thực hành kiểm tra sinh viên nào đủ tiêu chuẩn thi, sinh viên nào không tham dự đầy đủ số buổi thực hành sẽ bị cấm thi.

Cuối học kỳ bộ môn sẽ tổng kết số giờ coi thực hành của từng cán bộ để giáo vụ khoa tổng hợp công tác giảng dạy.

3.6 Công tác tổ chức thi học kỳ

Trợ lý giáo vụ khoa công nghệ thông tin muốn tin học hóa việc tổ chức thi cử ở các đơn vị đào tạo mà khoa phụ trách. Hàng năm khoa phải tổ chức nhiều đợt thi cho sinh viên các đơn vị đào tạo: thường là thi cuối học kỳ của mỗi năm học, mỗi học kỳ có hai lần thi, mỗi lần thi lại tổ chức nhiều đợt khác nhau do có thể có nhiều môn thi trong một lần thi. Tùy từng lần thi có thể có 1, 2 hoặc thậm chí 3 đợt cho một lần thi.

Trước hết trợ lý giáo vụ phải dự kiến thời gian tổ chức cho mỗi đợt trong mỗi lần của kỳ thi, danh sách các môn thi, số sinh viên tham dự cho các lớp tương ứng tại mỗi đơn vị đào tạo. Mỗi đợt thi tại các đơn vị đào tạo khác nhau có hệ thống các môn thi khác nhau. Có một số đơn vị đào tạo do điều kiện khách quan có thể không có một số khóa học nào đó. Nếu tất cả sinh viên của lớp tại một đơn vị đào tạo đã đạt lần thi thứ nhất thì không cần tổ chức thi môn đó cho lớp đó trong lần 2; nhưng cũng tại đợt thi của lần thi đó ở đơn vị đào tạo khác lại phải bố trí do có sinh viên chưa đạt kết quả trong lần thi trước. Số sinh viên trong lần thi thứ nhất thường là tất cả các sinh viên của lớp đủ điều kiện dự thi.

Nói chung một đợt thi có lịch thi thống nhất áp dụng cho một số đơn vị đào tạo nào đó. Tuy nhiên do hoàn cảnh và nhiều nguyên nhân chủ quan cũng như khách quan, nên có thể thời gian thi áp dụng cho đơn vị đào tạo này khác với thời gian thi áp dụng cho đơn vị đào tạo kia là điều có thể xảy ra. Việc dự kiến trước thời gian tổ chức thi nhằm có kế hoạch trước để các bộ môn phân công cán bộ ra đề thi. Cùng một môn thi trong một đợt thi, nhưng ở những đơn vị đào tạo khác nhau có thể do những giáo viên khác nhau ra đề. Giáo viên ra đề tự quyết định thời gian làm bài của thí sinh cho đề mình ra. Giáo viên ra đề phải hoàn thành trước thời gian bắt đầu tổ chức đợt thi một tuần.

Sau đó trợ lý giáo vụ khoa làm lịch thi và cử các cán bộ làm giám sát đợt thi tại các đơn vị đào tạo có tổ chức thi. Có thể có nhiều cán bộ tham gia làm giám sát đợt thi tại cùng một đơn vị đào tạo (thường đầu tuần một người, cuối tuần lại người khác và sang tuần sau lại có thể là cán bộ khác nữa) do một đợt thi có thể kéo dài đến vài tuần. Khi làm lịch thi trợ lý giáo

vụ khoa dự kiến ngày, giờ bắt đầu cho mỗi môn thi tại từng đơn vị đào tạo. Người có trách nhiệm tại các đơn vị đào tạo chịu trách nhiệm phân công cán bộ coi thi cho mỗi môn. Theo qui định thường có 2 cán bộ coi thi cho mỗi môn. Tùy theo thời gian làm bài của sinh viên đối với mỗi môn thi đã được qui định bởi giáo viên ra đề mà tính tiền coi thi cho cán bộ coi thi môn đó. Thời gian thi càng dài thì tiền coi thi càng nhiều.

Mỗi đợt thi, sau khi phòng đào tạo xử lý bài thi (cắt phách) xong các trưởng bộ môn phân công cán bộ đến phòng đào tạo nhận bài thi về chấm. Chú ý là trong cùng một đợt thi, cùng một môn thi, bài thi các lớp tại các đơn vị đào tạo khác nhau, có thể do các cán bộ khác nhau chấm. Sau khi chấm xong các cán bộ chấm thi nộp kết quả cho phòng đào tạo và báo số sinh viên còn thiếu điểm để trợ lý giáo vụ khoa ghi nhận, làm cơ sở xác định số lượng đề cần photocopy cho việc ra đề lần sau.

Mỗi đợt thi tại mỗi đơn vị đào tạo, trợ lý giáo vụ khoa cần biết các thông tin cụ thể như: ngày, giờ, môn, lớp. thời gian thi, cán bộ ra đề, cán bộ chấm, số sinh viên còn nợ sau khi chấm, các cán bộ coi thi tương ứng đối với từng môn thi.

4. Biểu mẫu thực hiện đồ án môn học

I. Yêu cầu chung

Mỗi sinh viên đăng ký thực hiện phần mềm. Kết quả gồm báo cáo viết, đĩa/CD (chương trình nguồn, EXE, báo cáo viết.

II. Cấu trúc báo cáo viết

1. Hiện trạng và yêu cầu

Hiện trạng

- Giới thiệu về thới giới thực liên quan
- Mô tả qui trình các công việc liên quan đến đề tài
- Mô tả các mẫu biểu có liên quan
- Mô tả các qui định ràng buộc có liên quan
- Mô tả các qui định công thức tính có liên quan

Yêu cầu

Danh sách các công việc sẽ được hỗ trợ thực hiện trên máy tính (dựa theo tóm tắt yêu cầu đã cho)

2. Mô hình hóa yêu cầu

Mô hình luồng dữ liệu theo yêu cầu

- Sơ đồ luồng dữ liệu cho từng yêu cầu
- Mô tả chi tiết cho từng sơ đồ

Mô hình luồng dữ liệu chung cho toàn bộ hệ thốn

Sơ đồ luồng dữ liệu chung cho toàn bọê hệ thống

3. Thiết kế phần mềm

Thiết kế dữ liệu

+ Sơ đồ logic

+ Danh sách các thành phần của sơ đồ

Stt	Tên	Loại	Ý nghĩa	Ghi chú

+ Danh sách các thuộc tính của từng thành phần

Tên thành phần:

Stt	Tên	Loại	Kiểu	Miền giá trị	Ý nghĩa

Thiết kế giao diện

Stt	Mã số	Loại	Ý nghĩa	Ghi chú

+ Mô tả chi tiết từng màn hình

- Nội dung
- Danh sách biến cố và xử lý tương ứng trên màn hình

Stt	Biến cố	Ý nghĩa	Xứ lý tương ứng	Mã số xử lý

Thiết kế xử lý

+ Danh sách các xử lý (Các xử lý quan trọng)

Stt	Mã số	Loại	Ý nghĩa	Ghi chú

+ Mô tả chi tiết từng xử lý

- Sơ đồ luồng dữ liệu
- Mô tả chi tiết sơ đồ

4. Cài đặt thử nghiệm

Cài đặt

+ Danh sách tình trạng cài đặt các chức năng (mức độ hoàn thành)

Stt	Chức năng	Mức độ hoàn thành	Ý nghĩa

Thử nghiệm

+ Nội dung các bảng dữ liệu
+ Một số test-case chạy thử nghiệm
+ Các báo biểu màn hình cùng các số liệu tương ứng

5. Tổng kết

+ Các kết quả đã thực hiện
+ Đánh giá ưu khuyết điểm
+ Hướng mở rộng tương lai

PHỤC LỤC B

1. Phần mềm quản lý thư viên

③ Mô tả chi tiết các thuộc tính

1. **Độc giả:**

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MDG	Chuỗi	Khóa chính		
2	MLDG	Chuỗi	Khóa ngoại		
3	HoTen	Chuỗi			
4	NgaySinh	Ngày			
5	DiaChi	Chuỗi			
6	DienThoai	Chuỗi			

2. **Sách:**

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MSACH	Chuỗi	Khóa chính		
2	MTG	Chuỗi	Khóa ngoại		
3	MNXB	Chuỗi	Khóa ngoại		
4	MLSACH	Chuỗi	Khóa ngoại		
5	MNN	Chuỗi	Khóa ngoại		
6	TenSach	Chuỗi			
7	Ngàymua	Ngày			
8	SoTrang	Số	>0		

3. **Phiếu mượn**:

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MPHM	Chuỗi	Khóa chính		
2	NgayMuon	Ngày			

4. **Chi tiết mượn**:

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú

1	MPHM	Chuỗi	Khóa ngoại		
2	MSACH	Chuỗi	Khóa ngoại		
3	NgayTra	Ngày			

5.**Loại sách**:

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MLSACH	Chuỗi	Khóa ngoại		
2	TenLoai	Chuỗi			
3	Ghi Chu	Chuỗi			

6. **Loại độc giả:**

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MLDG	Chuỗi	Khóa chính		
2	Tenloại	Chuỗi			
3	GhiChu	Chuỗi			

7. **Nhà xuất bản**:

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MNXB	Chuỗi	Khóa chính		
2	Tenloai	Chuỗi			
3	GhiChu	Chuỗi			

8. **Tác giả**:

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MTG	Chuỗi	Khóa chính		
2	Ten	Chuỗi			
3	GhiChu	Chuỗi			

8. **Ngôn ngữ**

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MNN	Chuỗi	Khóa chính		
2	Ten	Chuỗi			

3	GhiChu	Chuỗi			

③ **Mô hình chi tiết các thành phần trong sơ đồ lớp**

1. Đối tượng Độc Giả

Stt	**Thuộc tính**	**Kiểu**	**Miền giá trị**	**Ý nghĩa**	**Ghi chú**
1	MDG	Chuỗi			
2	Loại độc giả	Số	giá trị rời rạc		
3	HoTen	Chuỗi			
4	NgaySinh	Ngày	từ 19 đến 90		
5	DiaChi	Chuỗi			
6	DienThoai	Chuỗi			

2. Đối tượng Sách

Stt	**Thuộc tính**	**Kiểu**	**Miền giá trị**	**Ý nghĩa**	**Ghi chú**
1	MSACH	Chuỗi			
2	Loại sách	Số			
3	Tác giả	Chuỗi			
4	Nhà xuất bản	Chuỗi			
5	Ngày nhập	Chuỗi			
6	TenSach	Chuỗi			
7	Ngôn ngữ	Ngày			
8	SoTrang	Số	>0		

3. Quan hệ mượn

Stt	**Thuộc tính**	**Kiểu**	**Miền giá trị**	**Ý nghĩa**	**Ghi chú**
1	Ngày mượn	Ngay	>=Ngày nhập		
2	Ngày trả	Ngay	>=Ngày mượn		

2. Phần mềm quản lý giải vô địch bóng đá

③ Mô tả chi tiết các thuộc tính

1. Đối tượng Đội bóng

Stt	Thuộc tính	Kiểu	Miền giá trị	Ghi chú
1	Tên	Chuỗi	Giá trị rời rạc	
2	Thành Phố	ĐT phụ		
3	Sân nhà	ĐT phụ		
4	Địa chỉ	Chuỗi		
5	Trạng thái	Số	Giá trị rời rạc	
6	Huấn luyện viên	ĐT phụ	Nhiều	

2. Đối tượng Cầu thủ

Stt	Thuộc tính	Kiểu	Miền giá trị	Ghi chú
1	Họ Tên	Chuỗi		
2	Ngày sinh	Ngày		
3	Vị trí	ĐT phụ		
4	Số Áo	Số	>=0	
5	Chiều cao	Số	>1.5	
6	Trạng thái	Số	Rời rạc	

3. Đối tượng Trận đấu

Stt	Thuộc tính	Kiểu	Miền giá trị	Ghi chú
1	Loại trận đấu	Số	Giá trị rời rạc	
2	Ngày	NGAY	>=0	
3	Giờ			
4	Thời gian	Số	>=0	
5	Sân	ĐT phụ		
6	Trọng tài	ĐT phụ	Nhiều	
7	Số khán giả	Số	Ít hơn sức chứa của sân	

4. Quan hệ Thi đấu

Stt	Thuộc tính	Kiểu	Miền giá trị	Ghi chú
1	Số bàn thắng	Số	Giá trị rời rạc	Tính toán

2	Số bàn thua	NGAY	>=0	Tính toán
3	Thẻ phạt	ĐT phụ	Nhiều	Tính toán

5. Quan hệ Ra sân

Stt	Thuộc tính	Kiểu	Miền giá trị	Ghi chú
1	Thời điểm	Số	>=0	
2	Vị trí	ĐT phụ		
3	Bàn thắng	ĐT phụ	Nhiều	
4	Thẻ phạt	ĐT phụ	Nhiều	

⊙ Mô tả chi tiết thuộc tính

Đội bóng

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MDBTên	Chuỗi	Khóa chính	
2	MTP	Chuỗi	Khóa ngoại	
3	HoTen	Chuỗi		
4	Diachi	Chuỗi		
5	DienThoai	Chuỗi		

Cầu thủ:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MCT	Chuỗi	Khóa chính	
2	MDB	Chuỗi	Khóa ngoại	
3	MVT	Chuỗi	Khóa ngoại	
4	HoTen	Chuỗi		
5	Ngaysinh	NGAY		
6	SoAo	Số	>0	
7	TrangThai	Logic		

Trận đấu:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MTRD	Chuỗi	Khóa chính	
2	MLTRD	Chuỗi	Khóa ngoại	
3	MSAN	Chuỗi	Khóa ngoại	
4	Ngay	NGAY		

5	GIO	GIO		
6	Thoigian	Số	>0	
7	Sokhangia	Số	>0	

Thi đấu:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MTD	Chuỗi	Khóa chính	
2	MTRD	Chuỗi	Khóa ngoại	
3	MDB	Chuỗi	Khóa ngoại	
4	Ketqua	Số		

Ra Sân

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MRS	Chuỗi	Khóa chính	
2	MTD	Chuỗi	Khóa ngoại	
3	MCT	Chuỗi	Khóa ngoại	
4	MVT	Chuỗi	Khóa ngoại	
5	Thoidiem	Số		

Ghi bàn:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MRS	Chuỗi	Khóa chính, khóa ngoại	
2	MTD	Chuỗi	Khóa chính, Khóa ngoại	
3	Thoidiem	Số		

Phạt

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MRS	Chuỗi	Khóa chính, khóa ngoại	
2	MTHE	Chuỗi	Khóa chính, Khóa ngoại	
3	Thoidiem	Số		

Điều khiển:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú

1	MTRD	Chuỗi	Khóa chính, khóa ngoại	
2	MTTAI	Chuỗi	Khóa chính, Khóa ngoại	
3	MVTRO	Chuỗi	Khóa ngoại	

Loại trận đấu:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MLTRD	Chuỗi	Khóa chính	
2	Tên	Chuỗi		
3	Ghichu	Chuỗi		
4	Sobanthang	Số	>0	tính tóan
5	SoThe	Số	>0	tính toán

Vị trí:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MVT	Chuỗi	Khóa chính	
2	Tên	Chuỗi		
3	Ghichu	Chuỗi		

Trách nhiệm:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MTN	Chuỗi	Khóa chính	
2	Tên	Chuỗi		
3	Ghichu	Chuỗi		

Vai trò:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MVTRO	Chuỗi	Khóa chính	
2	Tên	Chuỗi		
3	Ghichu	Chuỗi		
4	Soluong	Số		tính toán

Loại bàn thắng:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MLBT	Chuỗi	Khóa chính	
2	Tên	Chuỗi		

3	Ghichu	Chuỗi		
4	Soluong	Số		tính toán

Thẻ phạt:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MTHE	Chuỗi	Khóa chính	
2	Tên	Chuỗi		
3	Ghichu	Chuỗi		
4	Soluong	Số		tính toán

Trọng tài:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MTTAI	Chuỗi	Khóa chính	
2	Tên	Chuỗi		
3	Ghichu	Chuỗi		

Huấn luyện viên:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MHLV	Chuỗi	Khóa chính	
2	MDB	Chuỗi	Khóa ngoại	
3	MTN	Chuỗi	Khóa ngoại	
4	Ten	Chuỗi		
5	Ghichu	Chuỗi		

Thành phố:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MTP	Chuỗi	Khóa chính	
2	Tên	Chuỗi		
3	Ghichu	Chuỗi		

Sân:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MSAN	Chuỗi	Khóa chính	
2	MTP	Chuỗi	Khóa ngoại	
2	Tên	Chuỗi		
3	Succhua	Số	>0	

LỜI NÓI ĐẦU

Nhập môn Công Nghệ Phần Mềm là môn học nhằm giúp cho sinh viên có kiến thức cơ bản nhất trong lĩnh vực công nghệ phần mềm. Qua môn học này sinh viên có cái nhìn khái quát về qui trình phát triển phần mềm, hiểu biết và thực hiện các giai đoạn trong qui trình trên một phần mềm cụ thể dựa trên những phương pháp, kỹ thuật trong quá trình thu thập yêu cầu, phân tích, thiết kế và cài đặt, viết sưu liệu đã được minh họa cụ thể trong giáo trình. Mục tiêu giáo trình là sinh viên có thể hiểu được những yêu cầu công việc cần phải làm ở mỗi giai đoạn của qui trình, để có thể đảm trách công việc ở một trong các giai đoạn làm phần mềm trong những nhóm dự án.

TÀI IỆU THAM KHẢO

1. Software Engineering
   By Nguyễn Xuân Huy – Institue of Information Technology

2. Nhập môn công nghệ phần mềm
   Nguyễn Tiến Huy – ĐH Khoa học Tự Nhiên

3. A Discipline for Software Engineering
   Watts S. Humphrey

4. Quá trình phát triển phần mềm thống nhất
   Nguyễn Tuấn Huy biên dịch –Nhà xuất bản thống kê

5. Analyzing Requriements and Defining Solution Architechtures
   Ian Lewis – Bruce Nielson

6. MCSD Analyzing Requirements Study Guide
   Tata McGraw-Hill Pusblishing Company Limited

7. Software Engineering
   Roger S.PressMan

8. Một số tài liệu tham khảo từ internet: Khoa CNTT ĐH KHTN, ĐH BKHN, ĐH Cần Thơ, và một số bài báo khoa học.
   - A Summary of Principles for User-Interface Design by Talin
   - The Foundation for Verifiable Software Process Improvement
   - Lecture Notes: Software Engineering I by Joey Paquet

Chương 1: TỔNG QUAN VỀ CÔNG NGHỆ PHẦN MỀM

1. CÁC KHÁI NIỆM CƠ BẢN

1.1. Phần mềm

1.1.1. Các khái niệm

Chương trình máy tính là một trình tự các chỉ thị để hướng dẫn máy tính làm việc nhằm hoàn thành một công việc nào đó do con người yêu cầu.

Phần mềm là một hệ thống các chương trình có thể thực hiện trên máy tính nhằm hỗ trợ các nhà chuyên môn trong từng lĩnh vực chuyên ngành thực hiện tốt nhất các thao tác nghiệp vụ của mình. Nhiệm vụ chính yếu của phần mềm là cho phép các nhà chuyên môn thực hiện các công việc của họ trên máy tính dễ dàng và nhanh chóng hơn so với khi thực hiện cùng công việc đó trong thế giới thực.

Hoạt động của mọi phần mềm là sự mô phỏng lại các hoạt động của thế giới thực trong một góc độ thu hẹp nào đó trên máy tính. Quá trình sử dụng một phần mềm chính là quá trình người dùng thực hiện các công việc trên máy tính để hoàn tất một công việc tương đương trong thế giới thực.

Lớp phần mềm là hệ thống các phần mềm trên cùng lĩnh vực hoạt động nào đó. Do cùng lĩnh vực hoạt động nên các phần mềm này thường có cấu trúc và chức năng (công việc mà người dùng thực hiện trên máy tính) tương tự nhau. Mục tiêu của ngành công nghệ phần mềm là hướng đến không những xây dựng được các phần mềm có chất lượng mà còn cho phép xây dựng dễ dàng một phần mềm mới từ các phần mềm đã có sẵn trong cùng kĩnh vực (thậm chí trong các lĩnh vực khác).

STT	Lớp phần mềm	Các phần mềm
1	Hỗ trợ giải bài tập	lượng giác, hình học, giải tích, số học, …
2	Trò chơi	cờ carô, cờ tướng, cờ vua, xếp hình, …
3	Xếp lịch biểu	thi đấu, thời khóa biểu, hội nghị, …
4	Xét tuyển	nhân sự, học sinh lớp 10…

5	Bình chọn	Sản phẩm, cầu thủ, …
6	Quản lý học sinh	Mầm non, trung học, trung tâm…
7	Bán hàng	thuốc tây, vật liệu xây dựng, máy tính
8	Quản lý thuê bao	điện, điện thoại, nước, …
9	Cho mượn	sách, truyện, phim, …

***Bảng 1.1**: Các phần mềm và lớp phần mềm tương ứng*

1.1.2. Phân loại

Phần mềm hệ thống là những phần mềm đảm nhận công việc tích hợp và điều khiển các thiết bị phần cứng đồng thời tạo ra môi trường thuận lợi để các phần mềm khác và người sử dụng có thể thao tác trên đó như một khối thống nhất mà không cần phải quan tâm đến những chi tiết kỹ thuật phức tạp bên dưới như cách thức trao đổi dữ liệu giữa bộ nhớ chính và đĩa, cách hiển thị văn bản lên màn hình, ...

Phần mềm ứng dụng là những phần mềm được dùng để thực hiện một công việc xác định nào đó. Phần mềm ứng dụng có thể chỉ gồm một chương trình đơn giản như chương trình xem ảnh, hoặc một nhóm các chương trình cùng tương tác với nhau để thực hiện một công việc nào đó như chương trình xử lý bản tính, chương trình xử lý văn bản, ...

1.1.3. Kiến trúc phần mềm

Sau khi đã có các khái niêm cơ bản nhất về phần mềm, tiếp sau đây chúng ta sẽ đi sâu vào tìm hiểu cấu trúc chi tiết các cấu trúc chi tiết các thành phần bên trong phần mềm. Phần mềm bao gồm 3 thành phần:

a) Thành phần giao tiếp (giao diện)

Cho phép tiếp nhận các yêu cầu về việc muốn thực hiện và cung cấp các dữ liệu nguồn liên quan đến công việc đó hoặc từ các thiết bị thu thập dữ liệu (cân, đo nhiệt độ, tế bào quang học, …)

Cho phép trình bày các kết quả của việc thực hiện các yêu cầu cho người dùng (kết quả của công việc khi thực hiện trên máy tính) hoặc điều khiển hoạt động các thiết bị điều khiển (đóng mở cửa, bật mở máy…)

Một cách tổng quát thành phần giao tiếp là hệ thống các hàm chuyên về việc nhập/xuất dữ liệu (hàm nhập/xuất) cùng với hình thức trình bày và tổ chức lưu trữ dữ liệu tương ứng, mục tiêu chính của các hàm này là đưa dữ liệu từ thế giới bên ngoài phần mềm vào bên trong hoặc ngược lại.

Trong phạm vi giáo trình này chỉ giới hạn xét đến giao tiếp với người sử dụng phần mềm và khi đó có tên gọi cụ thể hơn là thành phần giao diện.

b) Thành phần dữ liệu

Cho phép lưu trữ lại (hàm ghi) các kết quả đã xử lý (việc mượn sách đã được kiểm tra hợp lệ, bảng lương tháng đã được tính) trên bộ nhớ phụ với tổ chức lưu trữ được xác định trước (tập tin có cấu trúc, tập tin nhị phân, cơ sở dữ liệu).

Cho phép truy xuất lại (hàm đọc) các dữ liệu đã lưu trữ phục vụ cho các hàm xử lý tương ứng.

Một cách tổng quát thành phần dữ liệu là hệ thống các hàm chuyên về đọc ghi dữ liệu (hàm đọc/ghi) cùng với mô hình tổ chức dữ liệu tương ứng. Mục tiêu chính của các hàm này là chuyển đổi dữ liệu giữa bộ nhớ chính và bộ nhớ phụ.

c) Thành phần xử lý

Kiểm tra tính hợp lệ của các dữ liệu nguồn được cung cấp từ người dùng theo các quy trình ràng buộc trong thế giới thực (chỉ cho mượn tối đa 3 quyển sách, mỗi lớp học có tối đa 50 học sinh, …)

Tiến hành xử lý cho ra kết quả mong đợi theo quy định tính toán có sẵn trong thế giới thực (quy tắc tính tiền phạt khi trả sách trễ, quy tắc tính tiền điện, quy tắc trả góp khi mua nhà…) hoặc theo thuật giải tự đề xuất (xếp thời khóa biểu tự động, nén ảnh…)

Việc xử lý dựa trên dữ liệu nguồn từ người sử dụng cung cấp (tính nghiệm phương trình bậc 2 dựa trên các hệ số đã nhập) hoặc dữ liệu lưu trữ đã có sẵn (tính tồn kho tháng dựa trên các phiếu nhập xuất đã lưu trữ…) hoặc cả hai (tính tiền phạt dựa trên ngày trả sách được nhập vào và thông tin về loại sách đã được lưu trữ…) tùy vào xử lý cụ thể. Tương tự, việc xử lý cho ra kết quả có thể dùng để xuất cho người dùng xem qua thành phần giao diện (trình bày nghiệm, xuất tiền phạt), hay cùng có thể lưu trữ lại qua thành phần dữ liệu (sổ sách hiện đang được mượn của một độc giả…) hoặc cả hai (bảng lương, bảng tồn kho…)

Một cách tổng quát, thành phần xử lý là hệ thống các hàm chuyên về xử lý tính toán, biến đổi dữ liệu. Các hàm này sẽ dùng dữ liệu nguồn từ các hàm trong thành phần giao diện

(hàm nhập) hay thành phần dữ liệu (hàm đọc dữ liệu) kiểm tra tính hợp lệ (hàm kiểm tra) và sau đó tiến hành xử lý (hàm xử lý) nếu cần thiết để cho ra kết quả mà sẽ được trình bày cho người dùng xem qua các hàm trong thành phần giao diện (hàm xuất) hoặc lưu trữ lại qua các hàm trong thành phần dữ liệu (hàm ghi).

STT	Thành phần	Hàm	Ý nghĩa	Ghi chú
1	Thành phần giao diện	Hàm nhập Hàm xuất	Nhập yêu cầu, dữ liệu nguồn. Xuất kết quả đã xử lý	Cần xác định hình thức nhập/xuất và tổ chức dữ liệu tương ứng
2	Thành phần xử lý	Hàm kiểm tra Hàm xử lý	Kiểm tra tính hợp lệ của dữ liệu. Xử lý tính toán, phát sinh, biến đổi trên dữ liệu	Sử dụng hàm nhập, hàm đọc. Sử dụng hàm nhập, hàm đọc, hàm xuất, hàm ghi
3	Thành phần dữ liệu	Hàm đọc Hàm ghi	Đọc dữ liệu từ bộ nhớ phụ vào bộ nhớ chính. Ghi dữ liệu từ bộ nhớ chính vào bộ nhớ phụ	Cần xác định cáchh thức tổ chức lưu trữ dữ liệu

***Bảng 1.2**: Danh sách các hàm cùng ý nghĩa tương ứng*

1.2. Chất lượng phần mềm

1.2.1. Tính đúng đắn

Tính đúng đắn của phần mềm được thể hiện ở chỗ sản phẩm đó thực hiện đầy đủ và chính xác các yêu cầu của người dùng. Tính đúng đắn ở đây cần phải hiểu theo nghĩa rộng là

chương trình cần phải thực hiện được trong cả những trường hợp mà dữ liệu đầu vào là không hợp lệ.

Ví dụ, nếu một trong số các chức năng của phần mềm là sắp xếp một tập tin có số lượng mẫu tin tùy ý theo một cột tùy ý theo chiều tăng hoặc giảm thì những trường hợp sau là vi phạm tính đúng đắn của chương trình:

- Không thể thực hiện được (treo máy) khi tập tin rỗng (không có mẫu tin nào).
- Không thể thực hiện hoặc thực hiện nhưng cho kết quả sai khi các mẫu tin có hơn 100 cột hoặc có quá nhiều mẫu tin.
- Không thể thực hiện hoặc cho kết quả sai khi các cột có chiều dài lớn hơn 125 bytes.
- Không thể sắp xếp theo chiều tăng dần….

Tính đúng đắn của một sản phẩm phần mềm được xác minh qua các căn cứ sau đây:

- Tính đúng đắn của thuật toán.
- Tính tương đương của chương trình với thuật toán. Thuật toán có thể đúng nhưng chương trình lập ra không tương đương với thuật toán nên khi thực hiện sẽ cho kết quả sai.
- Tính đúng đắn của chương trình có thể được chứng minh trực tiếp trong văn bản của chương trình.
- Tính đúng đắn cũng có thể được khẳng định dần qua việc kiểm thử, việc áp dụng chương trình trong một khoảng thời gian dài trên diện rộng và với tần suất sử dụng cao.

1.2.2. Tính tiến hóa

Cho phép người dùng có thể khai báo các thay đổi về qui định với phần mềm tùy theo các thay đổi trong thế giới thực liên quan (thay qui định về số sách mượn tối đa, công thức tính tiền phạt, công thức tính tiền điện…)

Sản phẩm có thể mở rộng, tăng cường về mặt chức năng một cách dễ dàng.

1.2.3. Tính hiệu quả

Tính hiệu quả của một sản phẩm phần mềm được xác định qua các tiêu chuẩn sau:

- Hiệu quả kinh tế hoặc ý nghĩa, giá trị thu được do áp dụng sản phẩm đó.
- Tốc độ xử lý của phần mềm (v) tính bằng tỉ lệ giữa khối lượng đối tượng cần phải xử lý (m) và tổng thời gian (t) cần thiết để xử lý các đối tượng đó.

- Sử dụng tối ưu tài nguyên của máy tính (CPU, bộ nhớ…)

1.2.4. Tính tiện dụng

Sản phẩm phải tính đến những yếu tố tâm lý sau đây của người dùng:

- Dễ học, có giao diện trực quan tự nhiên.
- Dễ thao tác,…

1.2.5. Tính tương thích

Trao đổi dữ liệu với các phần mềm khác có liên quan (nhận danh mục sách từ tập tin Excel, gửi báo cáo tổng kết năm học đến phần mềm WinFax, …)

- Giao tiếp nội bộ
- Giao tiếp bên ngoài

1.2.6. Tính tái sử dụng

Sản phẩm phần mềm có thể áp dụng cho nhiều lĩnh vực theo nhiều chế độ làm việc khác nhau.

- Các phần mềm cùng lớp
- Các phần mềm khác lớp

1.3. Công nghệ phần mềm

1.3.1. Sự ra đời

Vào những năm 1950 khi máy tính ra đời chính thức (không chỉ được dùng trong các phòng thí nghiệm mà bắt đầu ứng dụng trong hoạt động xã hội) các phần mềm đầu tiên cũng được ra đời với số lượng còn rất ít ỏi và chủ yếu phục vụ cho lĩnh vực tính toán (đặc biệt trong quốc phòng).

Đến những năm 1960, trãi qua 10 năm phát triển số lượng các phần mềm đã tăng lên rất nhiều và được ứng dụng rộng rãi trong nhiều lĩnh vực. Vào thời điểm này phát sinh một vấn đề mà các chuyên gia gọi là “cuộc khủng hoảng phần mềm”. Cuộc khủng hoảng phần mềm thể hiện 2 yếu tố chính:

- Số lượng các phần mềm tăng vọt (do sự phát triển của phần cứng: tăng khả năng, giá thành hạ)
- Có quá nhiều khuyết điểm trong các phần mềm được dùng trong xã hội
    - Thực hiện không đúng yêu cầu (tính toán sai, không ổn định…)
    - Thời gian bảo trì, nâng cấp quá lâu, tốn chi phí cao, hiệu quả thấp.

- Khó sử dụng
- Thực hiện chậm
- Khó chuyển đổi dữ liệu giữa các phần mềm
- …..

Để giải quyết vấn đề trên một hội nghị đã được triệu tập đề bàn về cách giải quyết. Hội nghị đã tiến hành xem xét, phân tích và xác định nguyên nhân gây ra cuộc khủng hoảng phần mềm. Kết luận như sau:

- Việc tăng vọt của số lượng phần mềm là điều hợp lý và điều này sẽ còn tiếp diễn.
- Các khuyết điểm của phần mềm có nguồn gốc chính từ phương pháp, cách thức tiến hành xây dựng phần mềm:
    - Cảm tính: mỗi người theo một phương pháp riêng.
    - Thô sơ, đơn giản: chỉ tập trung vào việc lập trình mà ít quan tâm đến các công việc cần làm khác trước khi lập trình (khảo sát hiện trạng, phân tích yêu cầu, thiết kế…).
    - Thủ công: công cụ hỗ trợ chính khi xây dựng phần mềm chỉ là trình biên dịch.

Với các kết luận như trên, hội nghị đã đề xuất khai sinh một ngành khoa học mới: Công nghệ phần mềm với nhiệm vụ chính là nghiên cứu về các phương pháp tiến hành xây dựng phần mềm.

1.3.2. Định nghĩa

Công nghệ phần mềm là một lĩnh vực nghiên cứu của tin học nhằm đề xuất các nguyên lý, phương pháp, công cụ, cách tiếp cận phục vụ cho việc thiết kế, cài đặt các sản phẩm phần mềm đạt được đầy đủ các yêu cầu về chất lượng phần mềm.

Do quá trình tiến hóa của ngành công nghệ phần mềm nên khái niệm về nó cũng thay đổi theo thời gian. Hơn nữa do đây là một lĩnh vực mới nên các khái niệm vẫn còn phụ thuộc rất nhiều vào quan điểm chủ quan của từng người khác nhau. Cụ thể như sau:

- Bauer[1969]: việc thiết lập và sử dụng các nguyên lý công nghệ đúng đắn để thu được phần mềm một cách kinh tế vừa tin cậy vừa làm việc hiệu quả trên các máy thực.
- Ghezzi[1991]: là một lĩnh vực của khoa học máy tính liên quan đến việc xây dựng các phần mềm vừa lớn vừa phức tạp bởi một hay một số nhóm kỹ sư.
- IEEE[1993]:

1. Việc áp dụng phương pháp tiếp cận có hệ thống, bài bản và được lượng hóa trong phát triển, vận hành và bảo trì phần mềm.
2. Nghiên cứu các phương pháp tiếp cận được dùng trong (1).

- Sommervile[1995]: là lĩnh vực liên quan đến lý thuyết, phương pháp và công cụ dùng cho phát triển phần mềm.
- Kawamura[1995]: là lĩnh vực học vấn về các kỹ thuật, phương pháp luận công nghệ học (lý luận và kỹ thuật được hiện thực hóa trên các nguyên lý, nguyên tắc xác định) trong toàn bộ quy trình phát triển phần mềm nhằm nâng cao cả chất và lượng của sản xuất phần mềm.
- Pressman[1995]: là bộ môn tích hợp cả qui trình, các phương pháp, các công cụ để phát triển phần mềm máy tính.

Có thể định nghĩa tóm tắt về công nghệ phần mềm như sau: Công nghệ phần mềm là một nghành khoa học nghiên cứu về việc xây dựng các phần mềm có chất lượng trong khoảng thời gian và chi phí hợp lý.

Mục tiêu nghiên cứu được chia thành 2 phần rõ nét:

1. Xây dựng phần mềm có chất lượng.
2. Xây dựng phần mềm trong thời gian và chi phí hợp lý.

1.3.3. Đối tượng nghiên cứu

Hướng đến việc xây dựng các phần mềm có chất lượng như đã nêu, ngành công nghệ phần mềm đưa ra 3 đối tượng nghiên cứu chính: Qui trình công nghệ, Phương pháp phát triển, Công cụ và môi trường phát triển phần mềm.

- Qui trình công nghệ phần mềm: Hệ thống các giai đoạn mà quá trình phát triển phần mềm phải trải qua. Với mỗi giai đoạn cần xác định rõ mục tiêu, kết quả nhận từ giai đoạn trước đó cũng chính là kết quả chuyển giao cho giai đoạn kết tiếp.
- Phương pháp phát triển phần mềm: Hệ thống các hướng dẫn cho phép từng bước thực hiện một giai đoạn nào đó trong qui trình công nghệ phần mềm.
- Công cụ và môi trường phát triển phần mềm: Hệ thống các phần mềm trợ giúp chính trong lĩnh vực xây dựng phần mềm. Các phần mềm này sẽ hỗ trợ các chuyên viên tin học trong các bước xây dựng phần mềm theo một phương pháp nào đó với một qui trình được chọn trước.

2. QUI TRÌNH CÔNG NGHỆ PHẦN MỀM

Như đã nói để xây dựng được phần mềm có chất lượng quá trình phát triển phải trãi qua rất nhiều giai đoạn. Mỗi giai đoạn có mục tiêu và kết quả chuyển giao xác định. Trình tự thực hiện các giai đoạn này chính là chu kỳ sống của một phần mềm.

Nói cách khác, chu kỳ sống của một phần mềm là khoảng thời gian mà trong đó một sản phẩm phần mềm được phát triển, sử dụng và mở rộng cho đến khi sản phẩm phần mềm đó không còn được sử dụng nữa.

Chu kỳ sống của phần mềm được phân chia được phân chia thành các pha chính như: xác định, phát triển, kiểm thử, bảo trì (vận hành). Phạm vi và thứ tự các pha khác nhau tùy theo từng mô hình cụ thể.

2.1. Các bước cơ bản trong xây dựng phần mềm

2.1.1. Xác định

Đây là bước hình thành bài toán hoặc đề tài. Ở bước này thiết kế trưởng hoặc phân tích viên hệ thống phải biết được vai trò của phần mềm cần phát triển trong hệ thống, đồng thời phải ước lượng công việc, lập lịch biểu và phân công công việc.

Bên cạnh đó chúng ta phải biết người đặt hàng muốn gì. Các yêu cầu cần phải được thu thập đầy đủ và được phân tích theo chiều ngang (rộng) và chiều dọc (sâu). Công cụ sử dụng chủ yếu ở giai đoạn này là các lược đồ, sơ đồ phản ánh rõ các thành phần của hệ thống và mối liên quan giữa chúng với nhau.

2.1.2. Phát triển

Dựa vào các nội dung đã xác định được, nhóm phát triển phần mềm dùng ngôn ngữ đặc tả hình thức (dựa trên các kiến trúc toán học) hoặc phi hình thức (tựa ngôn ngữ tự nhiên) hoặc kết hợp cả hai để mô tả những yếu tố sau đây của chương trình:

- Giá trị nhập, giá trị xuất.
- Các phép biến đổi
- Các yêu cầu cần đạt được ở mỗi điểm của chương trình.

Phần đặc tả chỉ quan tâm chủ yếu đến giá trị vào, ra chứ không quan tâm đến cấu trúc và nội dung các thao tác cần thực hiện.

Sau bước thiết kế là bước triển khai các đặc tả chương trình thành một sản phẩm phần mềm dựa trên một ngôn ngữ lập trình cụ thể. Trong giai đoạn này các lập trình viên sẽ tiến hành cài đặt các thao tác cần thiết để thực hiện đúng các yêu cầu đã được đặc tả.

Công việc cuối cùng của giai đoạn phát triển là chúng ta cần phải chứng minh tính đúng đắn của chương trình sau khi đã tiến hành cài đặt. Tuy nhiên thông thường ở bước này chúng ta coi các chương trình như những hộp đen. Vấn đề đặt ra là xây dựng một cách có chủ đích các tập dữ liệu nhập khác nhau để giao cho chương trình thực hiện rồi dựa vào kết quả thu được để đánh giá chương trình. Công việc như trên được gọi là kiểm thử chương trình.

Công việc kiểm thử nhằm vào các mục tiêu sau:

- Kiểm tra để phát hiện lỗi của chương trình. Lưu ý rằng kiểm thử không đảm bảo tuyệt đối tính đúng đắn của chương trình do bản chất quy nạp không hoàn toàn của cách làm.
- Kiểm tra tính ổn định, hiệu quả cũng như khả năng tối đa của chương trình.

Tùy theo mục đích mà người ta thiết kế các tập dữ liệu thử sao cho có thể phủ hết các trường hợp cần quan tâm.

2.1.3. Bảo trì (Vận hành)

Công việc quản lý việc triển khai và sử dụng phần mềm cũng là một vấn đề cần được quan tâm trong qui trình phát triển phần mềm. Trong quá trình xây dựng phần mềm, toàn bộ các kết quả phần tích, thiết kế, cài đặt và hồ sơ liên quan cần phải được lưu trữ và quản lý cẩn thận nhằm đảm bảo cho công việc được tiến hành một cách hiệu quả nhất và phục vụ cho công việc bảo trì phần mềm về sau.

Như vậy công việc quản lý không chỉ dừng lại trong quá trình xây dựng phần mềm mà trái lại còn phải được tiến hành liên tục trong suốt quá trình sống của nó.

2.2. Một số mô hình triển khai xây dựng phần mềm

Có nhiều mô hình cận khác nhau để triển khai các bước cơ bản trong quá trình phát triển phần mềm. Mỗi mô hình sẽ chia vòng đời của phần mềm theo một cách khác nhau nhằm đảm bảo qui trình phát triển phần mềm sẽ dẫn đến thành công. Trong phần tiếp theo của giáo trình chúng ta sẽ tìm hiểu qua các mô hình phát triển phần mềm tiêu biểu nhất đang được áp dụng.

2.2.1. Mô hình thác nước:

Mô hình thác nước là một trong những mô hình đầu tiên và phổ biến được áp dụng trong quá trình phát triển phần mềm. Mô hình này chia quá trình phát triển phần mềm thành những giai đoạn tuần tự nối tiếp nhau. Mỗi giai đoạn sẽ có một mục đích nhất định. Kết quả cuả giai đoạn trước sẽ là thông tin đầu vào cho giai đoạn tiếp theo sau. Tùy theo qui mô của phần mềm cần phát triển mà mô hình thác nước sẽ có những biến thể khác nhau như sau:

- Qui trình 2 giai đoạn: Là qui trình đơn giản nhất. Theo qui trình này việc phát triển phần mềm chỉ trãi qua 2 giai đoạn:
  - Xác định yêu cầu: Được tiến hành ngay khi có nhu cầu về việc xây dựng phần mềm.
    - Mục tiêu: Xác định chính xác các yêu cầu đặt ra cho phần mềm sẽ xây dựng.
    - Kết quả nhận: Thông tin về hoạt động của thế giới thực.
    - Kết quả chuyển giao: Danh sách các yêu cầu (công việc sẽ thực hiện trên máy tính) cùng với các thông tin miêu tả chi tiết về các yêu cầu (cách thức thực hiện trong thế giới thực).
  - Lập trình (cài đặt): Được tiến hành ngay sau khi kết thúc việc xác định yêu cầu.
    - Mục tiêu: Tạo lập phần mềm mong muốn theo yêu cầu.
    - Kết quả nhận: Danh sách các yêu cầu cùng các thông tin có liên quan.
    - Kết quả chuyển giao: Chương trình nguồn của phần mềm với cấu trúc cơ sở dữ liệu tương ứng (nếu cần thiết) và chương trình thực hiện được trên máy tính (chương trình nguồn đã được biên dịch)
- Qui trình 3 giai đoạn: Là qui trình cải tiến của qui trình 2 giai đoạn bằng cách bổ sung thêm một giai đoạn trung gian mới giữa xác định yêu cầu và lập trình (có sửa đổi)
  - Xác định yêu cầu: được tiến hành ngay khi có nhu cầu về việc xây dựng phần mềm.
    - Mục tiêu: Xác định chính xác các yêu cầu đặt ra cho phần mềm sẽ xây dựng.
    - Kết quả nhận: Thông tin về hoạt động của thế giới thực.
    - Kết quả chuyển giao: Danh sách các yêu cầu (công việc sẽ thực hiện trên máy tính) cùng với các thông tin miêu tả chi tiết về các yêu cầu (cách thức thực hiện trong thế giới thực)
  - Thiết kế: Được tiến hành ngay sau khi kết thúc việc xác định yêu cầu.

- Mục tiêu: Mô tả các thành phần của phần mềm (mô hình của phần mềm) trước khi tiến hành cài đặt.
- Kết quả nhận: Danh sách các yêu cầu và thông tin liên quan.
- Kết quả chuyển giao:
  - Mô tả thành phần giao diện: các hàm nhập/xuất, cấu trúc dữ liệu nhập/xuất.
  - Mô tả thành phần xử lý: các hàm kiểm tra xử lý.
  - Mô tả thành phần dữ liệu: các hàm đọc/ ghi, tổ chức lưu trữ trên bộ nhớ phụ.

o Lập trình (cài đặt): Được tiến hành ngay sau khi kết thúc việc thiết kế.

- Mục tiêu: Tạo lập phần mềm theo yêu cầu.
- Kết quả nhận: Mô hình phần mềm
- Kết quả chuyển giao: Chương trình nguồn của phần mềm với cấu trúc cơ sở dữ liệu tương ứng (nếu cần thiết) và chương trình thực hiện được trên máy tính (chương trình nguồn đã được biên dịch)

❖ Qui trình 4 giai đoạn: Là qui trình cải tiến của qui trình phía trước bằng cách bổ sung thêm một giai đoạn mới giữa xác định yêu cầu và thiết kế (có sửa đổi)

o Xác định yêu cầu: Được tiến hành ngay khi có nhu cầu về việc xây dựng phần mềm.

- Mục tiêu: Xác định chính xác các yêu cầu đặt ra cho phần mềm sẽ xây dựng.
- Kết quả nhận: Thông tin về hoạt động của thế giới thực.
- Kết quả chuyển giao: Danh sách các yêu cầu (công việc sẽ thực hiện trên máy tính) cùng với các thông tin miêu tả chi tiết về các yêu cầu (cách thức thực hiện trong thế giới thực)

o Phân tích: được tiến hành ngay sau khi kết thúc việc xác định yêu cầu.

- Mục tiêu: Mô tả lại thế giới thực thông qua các mô hình (mô hình thế giới thực) trước khi thiết kế.
- Kết quả nhận: Danh sách các yêu cầu cùng các thông tin có liên quan.
- Kết quả chuyển giao:
  - Mô hình xử lý (hệ thống các công việc trong thế giới thực cùng với quan hệ giữa chúng)
  - Mô hình dữ liệu (hệ thống các loại thông tin được sử dụng trong thế giới thực cùng với quan hệ giữa chúng)

        - Các mô hình khác (không gian, thời gian, con người…) nếu cần thiết.

- Thiết kế: Được tiến hành ngay sau khi kết thúc việc phân tích.
    - Mục tiêu: Mô tả các thành phần của phần mềm (mô hình của phần mềm) trước khi tiến hành cài đặt.
    - Kết quả nhận: Mô hình thế giới thực.
    - Kết quả chuyển giao:
        - Mô tả thành phần giao diện: các hàm nhập/xuất, cấu trúc dữ liệu nhập/xuất.
        - Mô tả thành phần xử lý: các hàm kiểm tra xử lý.
        - Mô tả thành phần dữ liệu: các hàm đọc/ghi, tổ chức lưu trữ trên bộ nhớ phụ.
- Lập trình (cài đặt): Được tiến hành ngay sau khi kết thúc việc thiết kế.
    - Mục tiêu: Tạo lập phần mềm theo yêu cầu
    - Kết quả nhận: Mô hình phần mềm
    - Kết quả chuyển giao: Chương trình nguồn của phần mềm với cấu trúc cơ sở dữ liệu tương ứng (nếu cần thiết) và chương trình thực hiện được trên máy tính (chương trình nguồn đã được biên dịch)

❖ Qui trình 5 giai đoạn: Là qui trình cải tiến của qui trình phía trước bằng cách bổ sung thêm một giai đoạn mới sau giai đoạn lập trình nhằm tăng cường độ tin cậy của phần mềm.

- Xác định yêu cầu: Được tiến hành ngay khi có nhu cầu về việc xây dựng phần mềm.
    - Mục tiêu: Xác định chính xác các yêu cầu đặt ra cho phần mềm sẽ xây dựng.
    - Kết quả nhận: Thông tin về hoạt động của thế giới thực.
    - Kết quả chuyển giao: Danh sách các yêu cầu (công việc sẽ thực hiện trên máy tính) cùng với các thông tin miêu tả chi tiết về các yêu cầu (cách thức thực hiện trong thế giới thực)
- Phân tích: được tiến hành ngay sau khi kết thúc việc xác định yêu cầu.
    - Mục tiêu: Mô tả lại thế giới thực thông qua các mô hình (mô hình thế giới thực) trước khi thiết kế.
    - Kết quả nhận: Danh sách các yêu cầu cùng các thông tin có liên quan.
    - Kết quả chuyển giao:

- Mô hình xử lý (hệ thống các công việc trong thế giới thực cùng với quan hệ giữa chúng)
- Mô hình dữ liệu (hệ thống các loại thông tin được sử dụng trong thế giới thực cùng với quan hệ giữa chúng)
- Các mô hình khác (không gian, thời gian, con người…) nếu cần thiết.

o Thiết kế: Được tiến hành ngay sau khi kết thúc việc phân tích.

- Mục tiêu: Mô tả các thành phần của phần mềm (mô hình của phần mềm) trước khi tiến hành cài đặt.
- Kết quả nhận: Mô hình thế giới thực.
- Kết quả chuyển giao:
  - Mô tả thành phần giao diện: các hàm nhập/xuất, cấu trúc dữ liệu nhập/xuất.
  - Mô tả thành phần xử lý: các hàm kiểm tra xử lý.
  - Mô tả thành phần dữ liệu: các hàm đọc/ ghi, tổ chức lưu trữ trên bộ nhớ phụ.

o Lập trình (cài đặt): Được tiến hành ngay sau khi kết thúc việc thiết kế.

- Mục tiêu: Tạo lập phần mềm theo yêu cầu.
- Kết quả nhận: Mô hình phần mềm.
- Kết quả chuyển giao: Chương trình nguồn của phần mềm với cấu trúc cơ sở dữ liệu tương ứng (nếu cần thiết) và chương trình thực hiện được trên máy tính (chương trình nguồn đã được biên dịch).

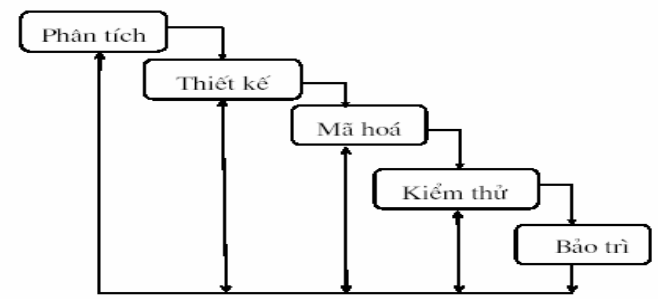
o Kiểm thử: Được tiến hành ngay sau khi đã có kết quả (từng phần) của việc lập trình.

- Mục tiêu: Tăng độ tin cậy của phần mềm.
- Kết quả nhận:
  - Danh sách yêu cầu.
  - Mô hình phần mềm.
  - Phần mềm.
- Kết quả chuyển giao: Phần mềm với độ tin cậy cao (đã tìm và sửa lỗi).

o Bảo trì: Công việc của giai đoạn bao gồm việc cài đặt và vận hành phần mềm trong thực tế.

- Mục tiêu: đảm bảo phần mềm vận hành tốt
- Kết quả nhận: phần mềm đã hoàn thàng

- Kết quả chuyển giao: các phản ánh của khách hàng trong quá trình sử dụng phần mềm.



Mô hình vòng đời cổ điển (thác nước)

Nhận xét:

Mô hình thác nước giúp chúng ta có thể dễ dàng phân chia quá trình xây dựng phần mềm thành những giai đoạn hoàn toàn độc lập nhau. Tuy nhiên, các dự án lớn hiếm khi tuân theo dòng chảy tuần tự của mô hình vì thường phải lặp lại các bước để nâng cao chất lượng. Hơn nữa, khách hàng hiếm khi tuyên bố hết các yêu cầu trong giai đoạn phân tích.

Mô hình này cũng có một hạn chế là chúng ta rất khó thực hiện các thay đổi một khi đã thực hiện xong một giai đoạn nào đó. Điều này làm cho việc xây dựng phần mềm rất khó thay đổi các yêu cầu theo ý muốn của khách hàng. Do đó, phương pháp này chỉ thích hợp cho những trường hợp mà chúng ta đã hiểu rất rõ các yêu cầu của khách hàng.

Chú ý: Mô hình thác nước có thể được cải tiến bằng cách cho phép quay lui khi phát hiện lỗi trong giai đoạn phía trước.

2.2.2. Mô hình bản mẫu phần mềm

Tương tự như mô hình thác nước với bổ sung vào các giai đoạn thực hiện phần mềm mẫu ngay khi xác định yêu cầu nhằm mục tiêu phát hiện nhanh các sai sót về yêu cầu. Các giai đoạn trong mô hình bản mẫu phần mềm có thể tiến hành lặp đi lặp lại chứ không nhất thiết phải theo trình tự nhất định.

Ngay sau khi giai đoạn xác định yêu cầu, nhà phát triển phần mềm đưa ra ngay một bản thiết kế sơ bộ và tiến hành cài đặt bản mẫu đầu tiên và chuyển cho người sử dụng. Bản mẫu này chỉ nhằm để miêu tả cách thức phần mềm hoạt động cũng như cách người sử dụng tương tác với hệ thống.

Người sử dụng sau khi xem xét sẽ phản hồi thông tin cần thiết lại cho nhà phát triển. Nếu người sử dụng đồng ý với bản mẫu đã đưa thì người phát triển sẽ tiến hành cài đặt thực

sự. Ngược lại cả hai phải quay lại giai đoạn xác định yêu cầu. Công việc này được lặp lại liên tục cho đến khi người sử dụng đồng ý với các bản mẫu do nhà phát triển đưa ra.

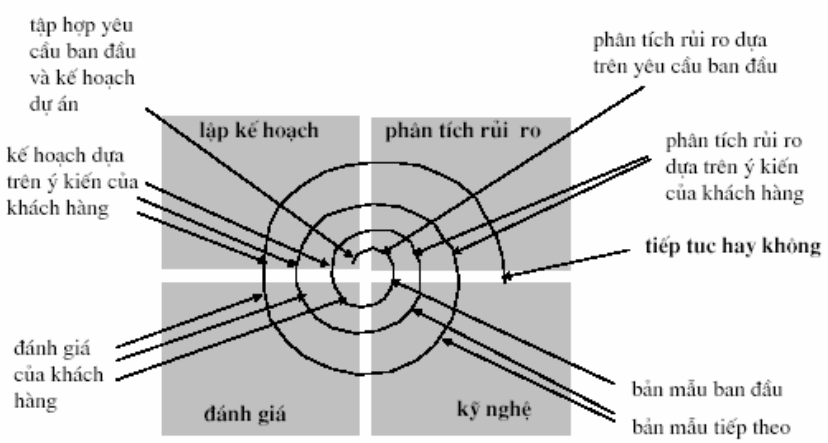
Như vậy đây là một hướng tiếp cận tốt khi các yêu cầu chưa rõ ràng và khó đánh giá được tính hiệu quả của các thuật toán. Tuy nhiên, mô hình này cũng có nhược điểm là tính cấu trúc không cao do đó khách hàng dễ mất tin tưởng.

Mô hình làm bản mẫu

2.2.3. Mô hình xoắn ốc

Mô hình này chính là sự kết hợp của mô hình bản mẫu thiết kế và mô hình thác nước được lặp lại nhiều lần. Ở lần lặp tiếp theo hệ thống sẽ được tìm hiểu và xây dựng hoàn thiện hơn ở lần lặp trước đó.

Ở mỗi lần lặp các yêu cầu của người sử dụng sẽ được hiểu ngày càng rõ ràng hơn và các bản mẫu phần mềm cũng ngày một hoàn thiện hơn. Ngoài ra ở cuối mỗi lần lặp sẽ có thêm công đoạn phân tích mức độ rủi ro để quyết định xem có nên đi tiếp theo hướng này nữa hay không.



Mô hình xoắn ốc

Mô hình này phù hợpvới các hệ thống phần mềm lớn do có khả năng kiểm soát rủi ro ở từng bước tiến hóa. Tuy nhiên vẫn chưa được sử dụng rộng rãi như mô hình thác nước hoặc bản mẫu do đòi hỏi năng lực quản lý, năng lực phân tích rủi ro cao.

3. CÁC PHƯƠNG PHÁP XÂY DỰNG PHẦN MỀM

3.1. Tổng quan

3.1.1. Khái niệm

Để tiến hành xây dựng một phần mềm, chúng ta có thể áp dụng nhiều phương pháp khác nhau. Mỗi phương pháp có những ưu và khuyết điểm riêng và phù hợp với từng loại phần mềm cụ thể.

Mỗi phương pháp sẽ có những hướng dẫn cụ thể các công việc cần phải thực hiện trong từng giai đoạn trong quy trình xây dựng phần mềm.

Bên cạnh đó mỗi phương pháp cũng sẽ quy định những cách thức khác nhau để trình bày các kết quả thu được trong quá trình xây dựng phần mềm. Những quy định này có tính chất như là ngôn ngữ thống nhất để các thành viên tham gia xây dựng phần mềm có thể trao đổi thông tin trong việc xây dựng phần mềm.

3.1.2. Phân loại

Các phương pháp xây dựng phần mềm được chia làm 02 nhóm khác nhau dựa vào tính chất của công việc cần thực hiện.

- ❖ Phương pháp xây dựng:
  - Phương pháp hướng chức năng

- Phương pháp hướng dữ liệu
- Phương pháp hướng đối tượng

❖ Phương pháp tổ chức quản lý

- Xây dựng phương án
- Tổ chức nhân sự
- Ước lượng rủi ro, chi phí
- Lập và theo dõi kế hoạch triển khai.

Trong phần tiếp theo của giáo trình này, chúng ta chỉ quan tâm đến các phương pháp xây dựng. Về phương pháp tổ chức quản lý chúng ta có thể tham khảo trong giáo trình "Quản lý dự án xây dựng các hệ thống thông tin".

3.2. Các phương pháp xây dựng phần mềm

3.2.1. Cách tiếp cận

a) Từ trên xuống

Đây là cách giải quyết vấn đề theo hướng phân tích. Khi tiến hành xây dựng phần mềm theo cách này, chúng ta bắt đầu với những thành phần chính của hệ thống. Sau đó, các thành phần này sẽ được phân tích thành các thành phần chi tiết và cụ thể hơn. Quá trình phân tích này sẽ kết thúc khi các kết quả thu được có mức độ phức đạp đúng với ý muốn của nhà xây dựng phần mềm.

b) Từ dưới lên

Ngược lại với phương pháp từ trên xuống, phương pháp từ dưới lên là cách giải quyết vấn đề theo hướng tổng hợp. Với phương pháp này, chúng ta tiến hành xây dựng những thành phần chi tiết, cụ thể mà mà chúng ta dự tính là sẽ có trong hệ thống. Sau đó, các nhà phát triển phần mềm sẽ tiến hành kết hợp các thành phần chi tiết này lại với nhau để tạo nên các thành phần chính mà hệ thống cần phải có.

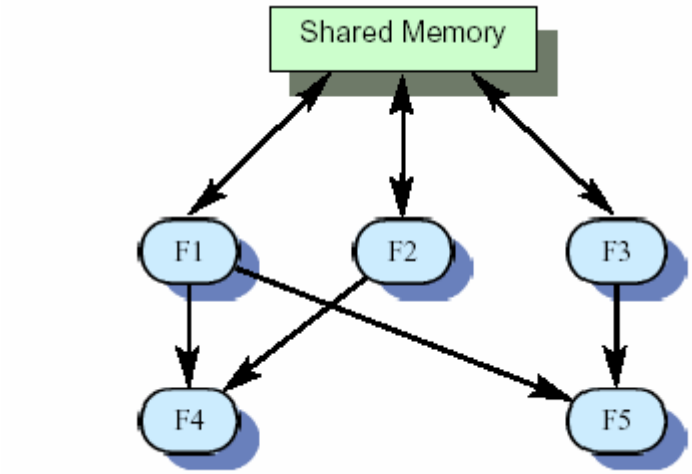
3.2.2. Cách tiến hành

a) Phương pháp hướng chức năng

Với phương pháp này công việc xây dựng phần mềm được thực hiện dựa trên các chức năng mà hệ thống cần thực hiện. Hay nói cách khác chúng ta chú trọng đến thành phần xử lý của hệ thống:

- Các thao tác tính toán
- Các thao tác phát sinh
- Các thao tác biến đổi….

Phương pháp chung để giải quyết vấn đề là áp dụng nguyên lý "chia để trị". Khi tiến hành xây dựng phần mềm theo phương pháp này, chúng ta sẽ chia các công việc lớn mà hệ thống cần thực hiện hành các công việc nhỏ hơn độc lập nhau. Việc phân chia các công việc được tiến hành cho đến khi các công việc thu được đủ nhỏ để chúng ta có thể tiến hành xây dựng hoàn chỉnh. Hình dưới: Minh họa cách tiếp cận theo hướng chức năng.



Phương pháp hướng chức năng chú trọng đến cách để giải quyết vấn đề nhưng không có khả năng che dấu các thông tin trạng thái của hệ thống. Điều này dẫn đến việc các chức năng trong hệ thống không tương thích với nhau trong việc thực hiện thay đổi các thông tin trong hệ thống. Chính vì vậy mà cách tiếp cận này chỉ thích hợp khi trong hệ thống có rất ít thông tin cần phải quản lý và chia sẻ giữa các chức năng với nhau. Để mô hình hóa cách xử lý thông tin trong hệ thống dùng lược đồ dòng dữ liệu (Data Flow Diagrams).

DFD là một công cụ đơn giản và hữu ích để miêu tả cách thức hoạt động của hệ thống. DFD sử dụng các ký hiệu sau để mô tả hệ thống:

- Ô vuông có góc tròn được dùng để biểu diễn các chức năng của hệ thống.

- Ô vuông dùng để biểu diễn thành phần dữ liệu trong hệ thống.
- Hình tròn dùng để biểu diễn các thành phần bên ngoài có giao tiếp với hệ thống.
- Dấu mũi tên dùng để biểu diễn hướng di chuyển của dữ liệu.
- Các từ khóa "and" và "or" dùng để liên kết các dòng dữ liệu khi cần thiết.

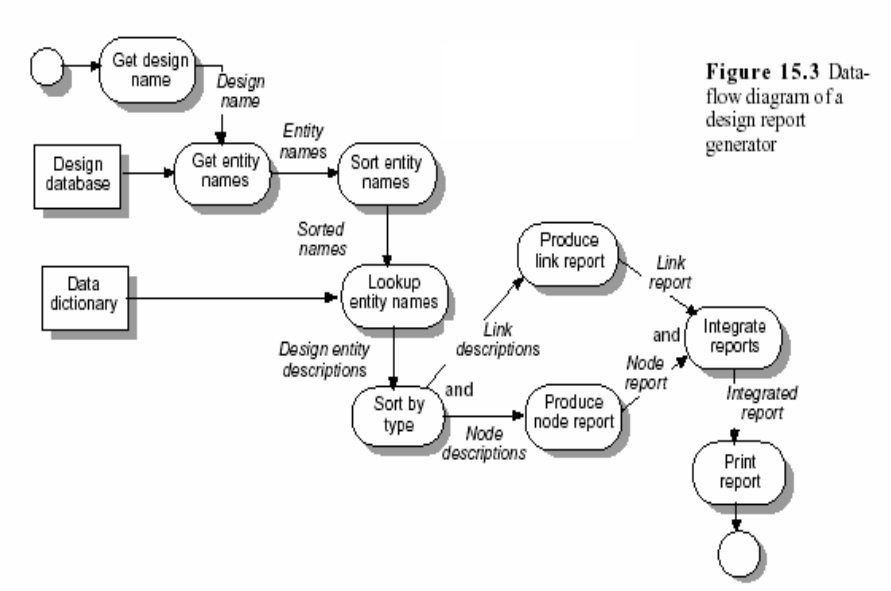


Figure 15.3 Data-flow diagram of a design report generator

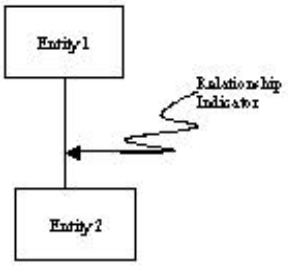
b) Phương pháp hướng dữ liệu

Ngược lại với phương pháp hướng chức năng, phương pháp hướng dữ liệu chú trọng nhiều đến thành phần dữ liệu cần phải xử lý trong hệ thống:

- Tổ chức dữ liệu
- Khối lượng lưu trữ
- Tốc độ truy xuất
- …

Khi tiến hành thiết kế theo phương pháp hướng dữ liệu chúng ta bắt đầu với việc thiết kế các cấu trúc dữ liệu cần thiết có trong bài toán, sau đó mới tiến hành thiết kết các thao tác để vận hành trên các cấu trúc dữ liệu đã thiết kế.

Phương pháp này đặc biệt chỉ thích hợp trong các loại phần mềm chỉ có chức năng chính là lưu trữ và thao tác trên các loại dữ liệu. Hạn chế của nó là không quan tâm đến các



chức năng mà hệ thống cần phải đáp ứng. Điều này dẫn đến việc có khả năng hệ thống sau khi thiết kế không có đầy đủ các chức năng cần thiết.

Kết quả thu được sau khi thiết kế theo phương pháp hướng dữ liệu là mô hình thực thể kết hợp (Entity Relationship Diagram). Một mô hình thực thể kết hợp điển hình gồm có 2 thành phần cơ bản: các thực thể và các mối kết hợp.

- Một thực thể là một đối tượng trong thế giới thực mà hệ thống có quan hệ, hoặc tương tác qua lại. Các thực thể được biểu diễn trong sơ đồ bằng các hình vuông cùng với tên và có thể có thêm các thuộc tính của thực thể.
- Mối kết hợp biểu diễn sự kết hợp giữa hai hay nhiều thực thể. Mỗi mối kết hợp gồm có ba thành phần cơ bản:
    - Mối kết hợp giữa các thực thể được biểu diễn băng một đường thẳng nối giữa hai thực thể.
    - Tên của môi liên hệ dùng để miêu tả ý nghĩa của mối liên hệ.
    - Bản số ở hai đầu của mối kết hợp dùng để xác định con số tối đa và tối thiểu các thực thể liên quan đến mối kết hợp.

c) Phương pháp hướng đối tượng

Phương pháp thiết kế hướng đối tượng là sự kết hợp của phương pháp hướng dữ liệu và phương pháp hướng chức năng. Phương pháp này chú trọng đến cả thành phần dữ liệu và chức năng của hệ thống.

Theo phương pháp hướng đối tượng thì một hệ thống phần mềm là một tập hợp các đối tượng có khả năng tương tác với nhau. Các đối tượng chính là các sự vật và hiện tượng vật lý cũng như trừu tượng mà chúng ta có trong thế giới thực. Mỗi đối tượng có dữ liệu riêng được che dấu với thế giới bên ngoài và các thao tác mà đối tượng có thể thực hiện trên các thành phần dữ liệu của đối tượng.

Các đối tượng liên lạc, trao đổi thông tin với nhau bằng cách gửi các thông điệp cho nhau. Các thông điệp mà mỗi đối tượng có thể xử lý được gọi là giao diện của đối tượng. Khi đó mọi thao tác liên quan đến các đối tượng được phải thực hiện thông qua giao diện của đối tượng. Điều này giúp chúng ta đảm bảo rằng các thông tin bên trong các đối tượng được bảo vệ một cách chắc chắn.

Chúng ta có thể sử dụng nhiều hệ thống ký hiệu khác nhau để mô tả các đối tượng của hệ thống cũng như mối liên hệ giữa chúng. Một trong số các hệ thống ký hiệu phổ biến hiện nay là hệ thốnng ký hiệu UML.

4. CÔNG CỤ VÀ MÔI TRƯỜNG PHÁT TRIỂN PHẦN MỀM

4.1. Mở đầu

4.1.1. Khái niệm

Các công cụ và môi trường phát triển phần mềm là các phần mềm hỗ trợ chính người phát triển trong quá trình xây dựng phần mềm. Các phần mềm này có tên gọi chung là CASE (Computer Aided Software Engineering) tools.

Trong quá trình phát triển phần mềm theo các quy trình trên, các CASE tools có thể hỗ trợ cụ thể cho một giai đoạn nào đó hay cũng có thể hỗ trợ một số giai đoạn, trong trường hợp này tên gọi chung thường là môi trường phát triển phần mềm-SDE (Software Development Environment).

Việc hỗ trợ của các CASE tools trong một giai đoạn bao gồm 2 hình thức chính:

- Cho phép lưu trữ, cập nhật trên kết quả chuyển giao với một phương pháp nào đó.
- Cho phép phát sinh ra kết quả chuyển giao cho giao đoạn kế tiếp.

4.2. Phần mềm hỗ trợ thực hiện các giai đoạn

4.2.1. Phần mềm hỗ trợ phân tích

- Công việc hỗ trợ chính
    - Soạn thảo các mô hình thế giới thực
    - Ánh xạ vào mô hình logic
- Các phần mềm: WinA&D, Analyst Pro,…

4.2.2. Phần mềm hỗ trợ thiết kế

- Công việc hỗ trợ chính
    - Soạn thảo các mô hình logic

- Ánh xạ vào mô hình vật lý

- Các phần mềm: QuickUML, Power Designer, Oracle Designer…

4.2.3. Phần mềm hỗ trợ lập trình

- Công việc hỗ trợ chính
  - Quản lý các phiên bản (Dữ liệu, chương trình nguồn, giao diện)
  - Biên dịch
- Các phần mềm: Visual Studio, Visual Basic, Visual C++

4.2.4. Phần mềm hỗ trợ kiểm chứng

- Công việc hỗ trợ chính
  - Phát sinh tự động các bộ dữ liệu thử nghiệm
  - Phát hiện lỗi
- Các phần mềm: WinRuner

4.3. Phần mềm hỗ trợ tổ chức, quản lý việc triển khai

4.3.1. Xây dựng phương án

- Công việc hỗ trợ chính
  - Tạo lập phương án
  - Dự đoán rủi ro
  - Tính chi phí
- Các phần mềm: MS Project, Visio

4.3.2. Lập kế hoạch

- Công việc hỗ trợ chính
  - Xác định các công việc
  - Phân công
  - Lập lịch biểu
  - Theo dõi thực hiện
- Các phần mềm: MS Project, Visio

Chương 2: PHÂN TÍCH VÀ ĐẶC TẢ YÊU CẦU

1. Tổng quan

Phân tích yêu cầu là khâu kỹ thuật đầu tiên gồm nhiều bước nhỏ: *nghiên cứu khả thi, phân tích mô hình hóa, đặc tả thẩm định yêu cầu.* Gia đoạn này được tiến hành phối hợp giữa bên phát triển và khách hàng và nó có vai trò đặc biệt quan trọng trong tiến trình phát triển phần mềm.

Đây là bước hình thành bài toán hoặc đề tài. Ở bước này trưởng nhóm thiết kế và người phân tích hệ thống phải biết được người đặt hàng muốn gì. Các yêu cầu phải được thu thập đầy đủ và được phân tích theo chiều ngang (rộng) và dọc (sâu). Công cụ sử dụng chủ yếu ở giai đoạn này là các lược đồ, sơ đồ phản ánh rõ các đối tượng của hệ thống: lưu đồ (Flowchart), sơ đồ dòng dữ liệu (Data Flow diagram DFD), mạng thực thể-quan hệ (Entity-Relationship Network), sơ đồ cấu trúc phân cấp (Structural hierarchical schemes), mạng ngữ nghĩa (Semantic Network)

1.1 Quá trình phân tích

1.1.1 Phân tích phạm vi dự án

Người phân tích hệ thống dùng thuật ngữ phạm vi để chỉ trách nhiệm dự án phải thực thi. Ngược lại, phạm vi dự án là nhiệm vụ lớn và phức tạp được thực hiện bởi chương trình.

Để xác định phạm vi dự án, bằng xác định quá trình nghiệp vụ ứng dụng sẽ đối đầu. Đó là những phạm vi vấn đề của ứng dụng. Nói chung, có hai phần đối với bất kỳ giải pháp nghiệp vụ: phần triển khai ứng dụng và phần thực hiện bởi con người hay chương trình. Định ra ranh giới ứng dụng tức là xác định qui trình trách nhiệm.

Một khi đã định nghĩa trách nhiệm của dự án:

- Chia trách nhiệm thành những nhiệm vụ con để đưa ra ý tưởng cho chính mình về bao nhiêu mô đun chương trình khác nhau yêu cầu?
- Xác định bao nhiêu vùng địa lý liên quan (chi nhánh văn phòng).
- Ước lượng số người dùng ứng dụng và thời gian ứng dụng được duy trì.
- Tính chính xác.
- Cuối cùng, hiểu khách hàng mong đợi dự án được triển khai.

Tại thời điểm này, chúng ta có ý tưởng phạm vi dự án. Cân nhắc độ lớn dự án đối với thời gian và ràng buộc ngân sách. Nếu dự án quá lớn về thời gian và tiền bạc cho chi trả thì

bàn bạc vấn đề với khách hàng để đưa ra quyết định thương lượng cho thõa đáng. Chúng ta phải chọn lựa hoặc nhiều thời gian hơn, hoặc nhiều tiền hơn hoặc cả hai. Hoặc chúng ta phải giảm phạm vi dự án xuống. Phân tích tất cả những tình huống ở giai đoạn đầu của dự án sẽ làm cho dự án thành công nhiều hơn.

1.1.2 Phân tích mở rộng yêu cầu nghiệp vụ

a. Xác định yêu cầu nghiệp vụ

Mỗi dự án sẽ có một hay nhiều yêu cầu nghiệp vụ. Mỗi yêu cầu nghiệp vụ là một mô tả tác nhiệm cụ thể trong nghiệp vụ của khách hàng. Ví dụ. lưu vết quá trình đầu tư. Một tác vụ như kiểm soát đầu tư cần chia nhỏ thành những phần chắc chắn cho đến khi mỗi phần đủ để mô tả công việc chính xác

Khi mức độ của thành phần chia nhỏ dưới mức tối thiểu, xác định lại trình tự thành phần.

Mỗi tác vụ được gọi là yêu cầu nghiệp vụ hay quy tắc nghiệp vụ. Quy tắc doanh nghiệp được viết theo ngôn ngữ được hiểu bởi những người không chuyên máy tính sao cho người dùng có thể kiểm tra luật một cách chính xác

b. Xác định yêu cầu chất lượng khách hàng

Mỗi dự án phần mềm có thể yêu cầu nhanh, bảo mật, phụ thuộc, dễ dùng, hay bug-free. Trong thế giới thực, thời gian và ràng buộc tài chính làm cho không thể tạo ra những chương trình dự án hoàn chỉnh. Thay vào đó, điều quan trọng để quyết định dựa trên mức độ chấp nhận của chất lượng thõa mãn khách hàng.

Ví dụ: khi khách hàng quyết định ứng dụng phải sẵn sàng 23 giờ trong ngày, bỏ qua thời gian vận hành không giảm. Chất lượng khác bao gồm số người dùng truy cập hiện hành, thời gian tối đa phải chờ để hoàn thành công việc trong ứng dụng (sự phản hồi), độ bảo mật ứng dụng, hay hơn nữa.

c.Phân tích hạ tầng cơ sở hiện hành

Phần quan trọng trong thiết kế giải pháp là phân tích kỹ thuật thay thế. Điển hình, giải pháp phần mềm được đưa vào hơn là thay thế hệ thống hiện hành. Dự án cần làm việc trên phần cứng và phần mềm mà người dùng hiện có. Biết được hệ điều hành đang được cài trên máy của người dùng, loại mạng đang sử dụng, và nếu người dùng đang chạy phần mềm không tương thích với chương trình mới hơn. Nên bỏ thời gian tìm hiểu máy chủ hiện hành, hệ điều hành, phần mềm đang chạy.

Khi đưa giải pháp, nhớ rằng cơ sở hạ tầng hiện hành đảm bảo giải pháp của chúng ta có thể tương thích.

d. Phân tích ảnh hưởng kỹ thuật

Nếu cần mở rộng chức năng cho hệ thống hiện hành, chúng ta mong ước thay đổi hệ thống cũ cả việc cải thiện hệ thống cũ và tích hợp dễ dàng hơn hệ thống mới. Ví dụ, chức năng của chương trình kế toán lưu trữ dữ liệu nhỏ như CSDL hướng đến tập tin Access. Để tạo dữ liệu truy xuất hiệu quả hơn và thõa mãn yêu cầu của giải pháp mới, chúng ta mới chuyển toàn bộ dữ liệu sang hệ quản trị csdl SQL Server. Việc suy nghĩ trước sẽ tiết kiệm thời gian sau đó: trãi qua thời gian tìm hiểu sự khác biệt về giao tác, bảo mật, và những chức năng khác giữa kỹ thuật cũ và giải pháp mới.

Chúng ta nên tìm hiểu thủ tục chuyển đổi dữ liệu từ kỹ thuật cũ sang kỹ thuật mới. Đảm bảo được phép thực nghiệm những thủ tục này, và có kế hoạch bảo lưu trong trường hợp thực hiện vấn đề này bị lỗi. Đảm bảo chắc chắn những tác động chuyển đổi trên mọi thành phần của hệ thống, không chỉ phần tử gần nhất thay đổi.

1.1.3.Phân tích yêu cầu bảo mật

Khi hệ thống lưu trữ, truy xuất dữ liệu cá nhân như thông tin nhân sự, thẻ tín dụng, doanh số bán hay thông tin riêng tư, chúng ta cần có biện pháp đảm bảo an toàn những dữ liệu này.

a. Xác định vai trò

Toàn bộ ứng dụng không chỉ có 1 mức độ bảo mật. Người dùng cuối chỉ cần quyền truy xuất giới hạn vào hệ thống. Quản trị hệ thống, người thao tác viên cập nhật, và người dùng có quyền truy cập cao hơn ở mọi cấp độ. Bảo mật dựa trên vai trò là kỹ thuật dùng để cấp quyền mức độ bảo mật khác nhau tương ứng quyền hạn và độ chuyên nghiệp của mỗi người dùng trong hệ thống.

Lưu ý: Nhận biết những lớp chính của những người dùng cần truy cập đến ứng dụng của chúng ta. Gán tên vai trò cho mỗi lớp người dùng. Cuối cùng, gán mức độ tối thiểu có thể truy xuất đến mỗi vai trò. Mỗi lớp người dùng nên có đủ quyền truy xuất đến công việc của họ, và không nhiều hơn.

b. Xác định môi trường bảo mật ứng dụng

Độ bảo mật không bị giới hạn người dùng hệ thống. Chỉ người dùng đăng nhập vào ứng dụng, ứng dụnng phải “login” để kiểm soát tài nguyên chia sẻ như tập tin, dịch vụ hệ thống,

cơ sở dữ liệu. Mức độ kiểm soát của ứng dụng được gọi là ngữ cảnh bảo mật. Chúng ta cần phải làm việc với nhiều người dùng khác như quản trị mạng, cấp quyền truy xuất phù hợp ứng dụng để chia sẻ tài nguyên.

c. Xác định ảnh hưởng bảo mật

Nếu công ty có sẵn cơ chế bảo mật thay vào đó hệ thống của chúng ta nên điều chỉnh cho phù hợp với cơ chế đã có. Nếu chúng ta đang thực thi hệ thống bảo mật mới hay một hệ thống khác, cần phải phân tích tác động của hệ thống trên hệ thống hiện tại:

- Hệ thống mới có làm hỏng chức năng của phần mềm hiện tại?
- Hệ thống đòi hỏi phải hỗ trợ thêm một phần người dùng – đăng nhập mở rộng ?
- Hệ thống sẽ khóa một vài người dùng trên những tập tin hay những tài nguyên mà họ được quyền truy cập trước đây

d. Kế hoạch vận hành

Khi tổ chức phát triển và thay đổi, người dùng mới được thêm vào, người cũ được cập nhật và bỏ đi. Những thao tác này đòi hỏi thay đổi CSDL bảo mật, đó là nơi thông tin người dùng và quyền hạn truy cập của họ được lưu. Những thông tin này được lưu trữ hiện thời.

Nếu người dùng có vị trí địa lý khác nhau, ở văn phòng khác nhau, chúng ta cần lên kế hoạch tái tạo cơ sở dữ liệu bảo mật. Sự tái tạo là sự thay đổi hệ thống dữ liệu tại nơi này sao chép đến nơi khác sao cho tất cả thông tin bảo mật được lưu giữ mỗi nơi. Thuận lợi việc tạo bản sao là người dùng có thể đăng nhập dùng thông tin được lưu ở vị trí gần hơn so với vị trí địa lý. Nếu mạng WAN bị ngừng hoạt động, ví dụ người dùng vẫn có thể đăng nhập. Việc tạo bản sao cần được lên kế hoạch và vận hành.

Lưu ý: Chúng ta lên kế hoạch cho điều kiện khẩn cấp – phải làm gì nếu csdl bảo mật bị ngắt hay nếu việc tạo bản sao bị hỏng. Đối với hệ thống bảo mật bị hỏng, chúng ta cũng nên có cả hai kế hoạch khẩn cấp và thủ tục tự động chú ý đến những vấn đề chung như mạng bị hỏng.

d. Kế hoạch kiểm soát và đăng nhập

Một hệ thống bảo mật tốt không là cơ chế thụ động. Thay vào đó, chứa chức năng trợ giúp kiểm soát hoạt động của hệ thống cho vấn đề bảo mật. Vấn đề chung của chức năng này là nhật ký. Toàn bộ thao tác của hệ thống có thể được ghi nhận hầu như toàn bộ sự kiện liên quan đến bảo mật hệ thống. Có thể ghi nhận mỗi khi đăng nhập, truy xuất đến mọi tài nguyên nhưng điều này hiếm khi hiệu qủa; thường chúng ta sẽ ghi nhận một số tập thông tin này như việc cố gắng đăng nhập lỗi.

Lưu ý: Nhật ký hệ thống tự nó thì không có ý nghĩa; chúng ta phải kế hoạch kiểm soát thường xuyên bởi ta có thể phát hiện những nghi ngờ những mẫu nhật ký hoạt động. Người kiểm soát được huấn luyện nên phân tích nhật ký trên cơ sở thường xuyên, đưa ra những đề nghị nếu có bất kỳ điều nghi ngờ.

e. Xác định mức độ yêu cầu bảo mật

Bảo mật cũng giống như những phần khác trong thiết kế ứng dụng, là sự cân nhắc giữa hiệu quả và chi phí. Nếu hệ thống không lưu những dữ liệu có tính nhạy cảm cao. Cách tốt nhất để triển khai hệ thống đó là "giữ sự xác thực của người dùng" đòi hỏi lưu trữ. Nếu chúng ta lưu trữ thông tin cần cho bảo mật, chi phí cho bảo mật thông tin đặc biệt phải được kiểm chứng.

Không có hệ thống nào bảo mật 100%. Chúng ta phải xác định mức độ rủi ro bảo mật có thể chấp nhận được. Độ rủi ro bảo mật diễn tả tỉ lệ phần trăm tương xứng khả năng mà bảo mật hệ thống không bao giờ đạt đến. Điều đó có thể nhưng phí tổn để xây dựng hệ thống bảo mật 99%. Chúng ta hay khách hàng phải xác định mức độ rủi ro có thể chấp nhận được dựa trên dữ liệu nhạy cảm của hệ thống.

f. Rà soát bảo mật hiện tại

Chúng ta nên trung thành ý tưởng của yêu cầu bảo mật của ứng dụng. Ở thời điểm phân tích chính sách bảo mật hiện tại của công ty để xác định bảo mật có đạt đến những nhu cầu của hệ thống hay không. Nếu không, thảo luận vấn đề với người gách vác hệ thống bảo mật ở công ty để tìm ra giải pháp mang lại lợi ích để triển khai mở rộng bảo mật.

1.1.4.Phân tích yêu cầu tốc độ

Tốc độ của ứng dụng có thể đòi hỏi khó. Đối với người dùng, ứng dụng sẽ hầu như chạy quá chậm nhưng chạy nhanh ứng dụng thiết kế tốt có thể mang lại giá trị

Lưu ý: việc chạy nhanh một ứng dụng thiết kế kém thì dễ, nhiều ứng dụng có thể chạy chậm bởi thiết kế thiếu sót, những không bởi không tương thích giữa phần ứng và các yếu tố bên ngoài.

Chúng ta nên nhận thức yêu cầu tốc độ ứng dụng trước khi bắt đầu qui trình thiết kế. Yêu cầu tốc độ dựa theo các mục sau:

Mỗi phút giao dịch: cung cấp dịch vụ phụ thuộc vào số lượng lớn người dùng, ứng dụng phân tán dùng những giao tác. Số giao tác mỗi phút (TPM) là độ đo tốc độ hệ thống cơ sở dữ liệu.

Băng thông: Ứng dụng phân tán làm nghẽn việc sử dụng mạng. Sự phản hồi của ứng dụng xác định định băng thông mạng (độ rộng của đường truyền mạng). Băng thông thường được đo bằng megabit mỗi giây.

Khả năng chứa: Lượng lưu trữ- cả chính và phụ - sẵn sàng đối với ứng dụng là vấn đề lưu tâm quan trọng cho tốc độ chung của ứng dụng. RAM đòi hỏi của ứng dụng gây ra những khác biệt lớn cho tốc độ của ứng dụng.

Nút thắt: Trong mỗi hệ thống, có phần giới hạn tốc độ hệ thống nói chung. Ví dụ CPU tốc độ nhanh cũng không cải thiện gì mấy nếu phải chờ dữ liệu từ một ổ cứng quá chậm. Trong trường hợp này, ổ cứng sẽ là nút thắt của toàn bộ hệ thống. Không thể tăng tốc độ trừ khi nút thắc được nhận biết, bởi vì chỉ có cải thiện nút thắt làm nâng tốc độ phù hợp. Chúng ta có thể nhận biết nút thắt bằng cách sử dụng công cụ báo cáo hệ thống như Màn hình điều khiển tốc độ trên Window NT (Windows NT Performance Monitor).

Thuật ngữ tốc độ thường dùng đồng nghĩa với sự phản hồi - số lượng thời gian chiếm giữ để phản hồi lại hành động của người dùng. Có thể làm cho ứng dụng xuất hiện phản hồi mà không cần tăng tốc độ. Tuy nhiên, thời gian phản hồi trung bình của ứng dụng là đặc tính quan trọng, chúng ta phải kết hợp chặt chẽ những mục tiêu thời gian phản hồi đối vời yêu cầu chung thiết kế.

Không thể nói về tốc độ trong những ứng dụng phân tán mà không phân biệt quan trọng: giữa nhu cầu cao và trung bình. Tại một số thời điểm - tối hay cuối tuần – có lẽ ứng dụng sẽ phục vụ với số lượng nhỏ người dùng, thì tốc độ nó sẽ trên trung bình. Ở thời điểm khác, số lượng người dùng sẽ cao hơn và tốc độ ứng dụng đủ cho phép. Mục tiêu tốc độ bao gồm cả mục tiêu tốc độ trung bình và cao.

1.1.5 Phân tích yêu cầu vận hành

Chúng ta có thể giảm bớt chi phí vận hành theo nhiều cách.Cách tốt nhất để giảm chi phí vận hành là đảm bảo chương trình được kiểm thử và chạy debug trước khi đưa vào triển khai. Chi phí triển khai có thể được giảm bớt bởi phân phối trực tuyến hay những thủ tục tự động cài đặt, và qui trình vận hành có thể tự động bằng các qui trình tin học. Mức vị trí và huấn luyện đội ngũ là vấn đề xem xét quan trọng: đội ngũ nhân viện càng được huấn luyện kỹ và sâu thì vấn đề càng nhanh chóng được sửa đổi.

Trong trường hợp phần cứng, phần mềm là thành phần được mua chứ không được phát triển, chúng ta có thể nhận sự chấp thuận vận hành từ nhà xưởng hay người ủy thác của sản phẩm. Vận hành sản phẩm trung gian tiết kiệm cho chúng ta chi phí thuê mướn nhân viên

mới hay huấn luyện lại những nhân viên cũ để duy trì một hay nhiều thành phần của hệ thống.

Giảm chi phí vận hành đòi hỏi sự tự thõa mãn lợi nhuận trong thời ngắn đối với những lợi ích trong tương lai. Giảm chi phí vận hành lâu dài thường đòi hỏi đầu tư đón đầu trong tự động hóa phần cứng và phần mềm.

1.1.6 Phân tích khả năng mở rộng yêu cầu

Qua thời gian, những yêu cầu giải pháp sẽ thay đổi. Người dùng cần những chức năng mới, các quy luật đặt ra sẽ bị sửa đổi, và phần cứng phần mềm nền mới thay đổi theo. Ứng dụng thiết kế tốt là có khả năng mở rộng được – nó có thể uyển chuyển cải thiện mà không phải viết lại hoàn toàn. Khả năng mở rộng của ứng dụng bị đảo ngược so với lượng công việc cần hoàn thành để thêm những đặc trưng mới.

Khả năng mở rộng có thể đạt được thông qua những ý nghĩa khác nhau. Một cách đạt những khả năng hạn định là lưu trữ thông tin quy luật đặt ra trong cơ sở dữ liệu hơn là lập trình chúng trong đối tượng nghiệp vụ. Theo cách đó, nếu số quan trong hay thủ tục thay đổi, nó có thể thay đổi trong CSDL mà không thay đổi mã nguồn. Cách khác là đặt mã nguồn vào trong đoạn script được làm rõ hơn biên dịch chương trình; đoạn script có thể bị thay đổi một cách dễ dàng không đòi hỏi bất kỳ biên dịch hay cài đặt lại tập tin nhị phân

Lưu ý: cách tốt nhất để đạt được khả năng mở rộng là ngắt ứng dụng thành những đối tượng thành phần, mỗi thành phần hoàn thành một nhiệm vụ riêng lẻ. Nếu những yêu cầu của những nhiệm vụ đặc biệt thay đổi, đối tượng tương ứng có thể bị thay đổi và biên dịch lại mà không gây ảnh hưởng bất kỳ đối tượng khác. Những đối tượng được thêm vào dễ dàng. Đối tượng nghiệp vụ có những thuận lợi được làm hiệu quả hơn những phương pháp khác trong khi vẫn đảm bảo tốt khả năng mở rộng.

1.1.7. Phân tích những yêu cầu sẵn có

Những ứng dụng phân tán được thiết kế để chạy mỗi ngày. Nó cần thiết cho sự thành công của doanh nghiệp. Như vậy, chúng có mức độ sẵn sàng cao nên tránh thường bảo trì, sửa chữa, phát sinh không theo kế hoạch.

Rõ ràng, đối với những ứng dụng mang tính sẵn sàng, nó không được gây ra lỗi. Không có ứng dụng nào là không có lỗi, ứng dụng phải được bảo lưu để chúng có thể hoạt động thậm chí khi bug xảy ra trong một phần của chương trình. Thí dụ, nếu người dùng gây ra lỗi cho chương trình thì chỉ một phần chương trình phục vụ cho người dùng đó bị hỏng, không

ảnh hưởng người dùng còn lại đang nối kết. Bất kỳ thành phần ứng dụng nào hỏng hay không sẵn sàng thì nên khởi động lại ngay khi có thể.

Việc bảo trì có kế hoạch cũng tác động đến tính sẵn sàng. Một máy chủ chứa ứng dụng lý tưởng luôn có bản sao lưu có thể khởi động khi máy chủ bảo trì. ứng dụng có mức độ sẵn sàng cao có cách luân phiên để kết nối mạng trong trường hợp mạng WAN, LAN ngưng hoạt động

Lưu ý: Tính sẵn sàng liên quan đến nghiệp vụ. Tính sẵn sàng của ứng dụng càng cao, giá trị của ứng dụng càng cao. Chúng ta phải xác định bao nhiêu giờ trong ngày ứng dụng cần được thao tác; giờ nào là quan trọng so với các giờ trong ngày. Cân nhắc giá trị của việc tăng tính sẵn sàng đối với giá trị dự án của thời gian down ứng dụng. Những hệ thống trọng yếu, giá trị đối với công ty ở bất kỳ thời điểm down nào hoàn toàn điều chỉnh chi phí thiết kế 100 % ứng dụng sẵn sàng. Ứng dụng khác đơn giản cần trở nên sẵn sàng hầu hết mọi lúc.

1.1.8. Phân tích yêu tố con người

Thiết kế ứng dụng được giám sát bởi nhiều người lập trình là phần quan trọng của yếu tố con người. Chúng ta nên xác định kinh nghiệm gì mà chúng ta muốn người dùng có. Với bất cứ ứng dụng nào khác, kinh nghiệm người dùng càng tốt thì chi phí càng cao.

Bắt đầu định nghĩa mục tiêu của người dùng. Xác định người dùng với những nhu cầu đặc biệt như thế nào. Chúng ta cần điều tiết người dùng qua việc điều tiết nghe và nhìn, hay người dùng nói tiếng nước ngoài. Phụ thuộc vào vị trí địa lý của người sử dụng. Chúng ta cần sửa đổi ứng dụng thích ứng theo vị trí địa lý. Cần điều chỉnh nhu cầu lướt qua của người dùng, người không cần sự nối kết chắc chắn hay khả năng trả lời lại.

Xem xét mức độ chuyên nghiệp giữa người dùng. Với chuyên viên học nhanh hơn với giao diện thiết kế tốt và trợ giúp trực tuyến Help online. Người dùng với kỹ năng kém hơn để tăng tốc qua sử dụng wizard, trợ giúp online, hay chỉ dẫn. Huấn luyện khách hàng trong ứng dụng cũng nên cân nhắc chọn lựa.

1.1.9. Phân tích yêu cầu tích hợp

Nếu giải pháp giao tiếp với ứng dụng kế thừa, việc truy xuất CSDL tồn tại, hay việc chuyển đổi dữ liệu cũ sang khuôn dạng mới, bạn cần phải đưa kế hoạch tích hợp ứng dụng với phần mềm cũ. Điều này được làm thông qua kết nối của hãng trung gian như trình điều khiển thiết bị kết nối csdl (ODBC), nhưng chúng ta cũng cần viết kết nối và những tiện ích chuyển đổi

Khi phát sinh nhu cầu lớn hơn, cơ sở dữ liệu phải thiết kế lại. Kỹ thuật dữ liệu mới hay vậ lý đưa nhu cầu cải thiện CSDL bên dưới ứng dụng. Những cải tiến phải được cẩn thận bởi chúng phá vở tất cả mã nguồn CSDL hiện tại. Trước khi cải tiến khung dữ liệu, đảm bảo những phần mã nguồn hiện tại có thể truy xuất đến CSDL. Tất cả mã nguồn hiện tại phải được soát lại, có thể viết lại.

1.1.10. Phân tích thực tiễn nghiệp vụ tồn tại

Phần định nghĩa trong qui tắc nghiệp vụ liên quan đến sự hiểu biết ngữ cảnh trong những qui tắc thao tác. Hiểu được những thực tế nghiệp vụ của doanh nghiệp có thể giúp chúng ta tránh được sai sót thậm chí giúp tìm cách tốt hơn, hiệu quả hơn của tự động hóa tiến trình nghiệp vụ. Hiểu được vấn đề hợp lệ dưới mỗi tiến trình có thể ngăn bạn gây ra lỗi một cách ngây ngô dẫn đến tranh chấp.

Hiểu được cấu trúc tổ chức và sơ đồ làm việc nghiệp vụ là quyết định. Không hiểu rõ ràng sơ đồ tổ chức, không thể đem lại sự chấp thuận phù hợp cho thiết kế ứng dụng của chúng ta hay thông tin theo kíp trên thiết kế hay những vấn đề triển khai. Đồ hình tổ chức cũng giúp cho tìm kiếm thông tin người ẩn danh phản hồi lại chức năng của ứng dụng mà không dùng bất của chính họ.

Có được ứng dụng từ giai đoạn phát triển đến sản phẩm đòi hỏi sự hiểu biết mạng và chính sách hạ tầng của công ty. Biết được ai là người chịu trách nhiệm bảo trì, bảo mật, tính toàn vẹn, khả năng phản hồi tương tác trên mạng. Học những tiến trình và chính sách liên quan chạy trên ứng dụng mới. Tìm ra loại kiểm soát chất lượng và dịch vụ kiểm thử sẵn sàng trong khi chúng ta kiểm thử trên chính phần mềm, ta có thể tự động tài nguyên hay dành cho bộ phận kiểm tra chất lượng tùy ý sử dụng. Chúng ta có thể yêu cầu phương pháp thiết kế đặc biệt hay triển khai thực tế. Chúng at cũng đòi hỏi chắc chắn kế hoạch được kết chặt với ngân sách

Cuối cùng, giữ những nguyên tắc cốt lỏi: Học nhu cầu khách hàng, cố gắng thực hiện chúng. Điều này có thể trở nên khó khi khách hàng không biết nhu cầu của họ là gì, nhưng đó là cách dẫn đến ứng dụng thành công.

1.1.11.Phân tích yêu cầu khả năng quy mô

Nếu ứng dụng thành công sẽ hấp dẫn người dùng hơn. Đặc biệt, nếu ứng dụng chạy trên môi trường web như Internet thì sự thành công đồng nghĩa với tăng nhu cầu. Ứng dụng phải được thiết kế có quy mô- nó phải hỗ trợ nâng cấp cho phép phục vụ nhiều người hơn.

Một cách đơn giản để nâng cao ứng dụng là mua CPU nhanh hơn, nhiều RAM, kết nối mạng tốt hơn. Tuy nhiên việc tăng cường máy đơn chạy nhanh hơn. Thực sự những ứng dụng có thể nâng cấp phải thêm vào nhiều dịch vụ phía máy chủ. Điều này có nghĩa ứng dụng có thể chạy trên nhiều máy tính cùng một lúc, sự phân phối việc tải xuống của người dùng và xử lý thời gian qua nhiều máy chủ. Điều này sẽ gia tăng đáng kể tính phức tạp, vì vậy một lần nữa tính thuận tiện khả năng quy mô phải được cân nhắc đối với giá trị cung cấp. Tuy nhiên, ứng dụng như Miscrosoft Transaction Server giảm đáng kể chi phí phát triển ứng dụng phân tán bởi quản lý về mặt logic của phân tán tự động.

1.2 Xác định yêu cầu

Mục tiêu của việc xác định yêu cầu:

Xác định thật chính xác và đầy đủ các yêu cầu đặt ra cho phần mềm sẽ được xây dựng.

Kết quả nhận được sau giai đoạn xác định yêu cầu:

1. *Danh sách các công việc sẽ được thực hiện trên máy tính*
2. *Những mô tả chi tiết về các công việc này khi được thực hiện trong thế giới thực.*

Qua đó bước đầu hình thành thông tin khái quát về các hoạt động trong thế giới thực.

1.2.1 Yêu cầu và mô tả yêu cầu

- Yêu cầu (hay yêu cầu phần mềm) là công việc muốn thực hiện *trên máy tính*. Những công việc này phải xuất phát từ thực tế chứ không thuần túy tin học
- Mô tả yêu cầu là mô tả đầy đủ các thông tin liên quan đến công việc tương ứng. Các mô tả này dùng làm cơ sở để nghiệm thu và đánh giá phần mềm khi được chuyển giao.

Các yêu cầu của phần mềm cần được mô tả thật rõ ràng, cụ thể, đầy đủ và chính xác các thông tin liên quan đến công việc tương ứng. Việc mô tả sơ sài, mơ hồ yêu cầu phần mềm sẽ dẫn đến việc hiểu nhầm giữa chuyên viên tin học (người thực hiện phần mềm) và khách hàng (người đặt hàng thực hiện phần mềm). Nhiều công sức và chi phí phải hao tốn do các hiểu nhầm như thế.

Các loại thông tin chính cần được quan tâm khi xác định yêu cầu phần mềm:

- Tên công việc ứng với từng yêu cầu
- Người hoặc bộ phận sẽ thực hiện công việc
- Địa điểm thực hiện công việc

- Thời gian thực hiện công việc
- Cách thức tiến hành công việc cùng với các quy định liên quan

Sau đây, từng loại thông tin sẽ lần lượt được xem xét chi tiết:

a. ***Tên công việc.***

Cần xác định cụ thể, tránh dùng các tên chung chung, mơ hồ

Ví dụ: xét một số tên công việc sau:

Quản lý độc giả: chung chung, mơ hồ; cụ thể như việc đăng ký mượn sách, gia hạn thẻ độc giả, trả sách

Quản lý sách: chung chung, mơ hồ; cụ thể như nhập sách vào kho, tra cứu sách, cho mượn sách, nhận trả sách, thanh lý sách.

b. ***Người thực hiện.***

Cần xác định chính xác người hoặc bộ phận sẽ thực hiện công việc trên máy tính (còn gọi là người dùng phần mềm hay người dùng).

Những người dùng có vai trò và công việc thực hiện tương tự như nhau sẽ được xếp vào cùng một loại người dùng (thông thường một loại người dùng sẽ tương ứng với một bộ phận trong thế giới thực).

Cùng một công việc có thể có nhiều loại người dùng khác nhau thực hiện và ngược lại, một loại người dùng có thể thực hiện nhiều công việc khác nhau.

c. ***Thời gian, địa điểm.***

Cần xác định chính xác địa điểm, thời điểm tiến hành công việc. Các thông tin này sẽ có ý nghĩa nhất định trong một số trường hợp đặc thù.

d. ***Cách thức tiến hành và các quy định liên quan.***

Đây là phần chính yếu khi tiến hành mô tả yêu cầu. Đối với loại thông tin này cần đặc biệt quan tâm đến một số yếu tố sau:

i. Các quy định cần kiểm tra khi thực hiện công việc ghi nhận thông tin

Ví dụ: Quy định về việc mượn sách khi cho độc giả mượn sách: chỉ cho mượn sách đối với những độc giả có thẻ độc giả còn hạn, số sách đang mượn chưa đến 2 và không có sách mượn quá hạn.

Ví dụ: Quy định tính hợp lệ của phân số trong việc ghi nhận đề bài của giáo viên và bài giải của học sinh: phân số phải có mẫu số khác 0

ii. Các quy định, công thức tính toán khi thực hiện công việc tính toán

Ví dụ: Quy định tính tiền phạt trả sách trễ khi thực hiện việc trả sách: mỗi ngày trả trễ phạt 1500 đồng/ngày. Từ ngày trả trễ thứ 10 trở đi sẽ phạt 5000 đồng/ngày và thu hồi thẻ độc giả 2 tuần.

Ví dụ: Quy định tiền lương khi thực hiện công việc tính lương nhân viên cho 1 công ty

* Lương của nhân viên thuộc bộ phận văn phòng được tính theo công thức:

Tiền_Lương = (Số_Ngày * Mức_Lương)/22 + Tiền_Thưởng

+ Tiền_Phạt

mỗi ngày làm thêm thưởng 30.000

mỗi ngày nghỉ việc phạt 50.000

* Lương của nhân viên thuộc bộ phận sản xuất được tính theo công thức:

Tiền_Lương = Số_Sản_Phẩm * Đơn_Giá

Biết rằng một sản phẩm phải trải qua 3 công đoạn sản xuất:

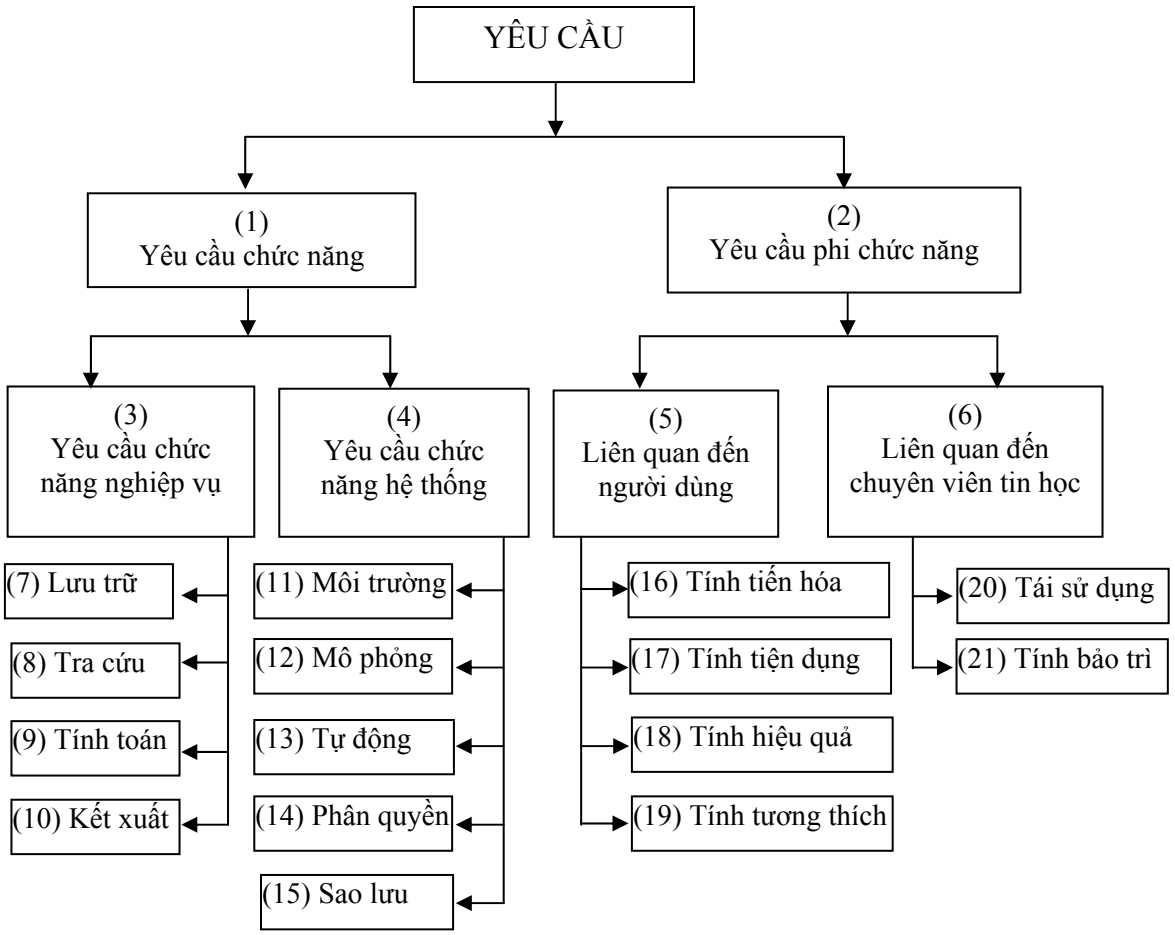
công đoạn 1: 200 đồng/sản phẩm

công đoạn 2: 400 đồng/sản phẩm

công đoạn 3: 300 đồng/sản phẩm

1.2.2 Phân loại yêu cầu

Sơ đồ cây phân loại yêu cầu


YÊU CẦU
(1)
Yêu cầu chức năng
(2)
Yêu cầu phi chức năng
(3)
Yêu cầu chức năng nghiệp vụ
(4)
Yêu cầu chức năng hệ thống
(5)
Liên quan đến người dùng
(6)
Liên quan đến chuyên viên tin học
(7) Lưu trữ
(8) Tra cứu
(9) Tính toán
(10) Kết xuất
(11) Môi trường
(12) Mô phỏng
(13) Tự động
(14) Phân quyền
(15) Sao lưu
(16) Tính tiến hóa
(17) Tính tiện dụng
(18) Tính hiệu quả
(19) Tính tương thích
(20) Tái sử dụng
(21) Tính bảo trì

Đặc tả chi tiết từng loại yêu cầu:

(1) Yêu cầu chức năng là danh sách các công việc sẽ được *thực hiện trên máy tính* cùng với các thông tin mô tả tương ứng.

(2) Yêu cầu phi chức năng là các yêu cầu liên quan đến chất lượng phần mềm, là sự ràng buộc cách thức thực hiện các yêu cầu chức năng.

(3) Yêu cầu chức năng nghiệp vụ là các chức năng của phần mềm tương ứng với công việc có thật trong thế giới thực.

(4) Yêu cầu chức năng hệ thống là các chức năng phần mềm *được phát sinh thêm* khi thực hiện công việc trên máy tính thay vì trong thế giới thực hoặc các chức năng không tương ứng với bất kỳ công việc nào trong thế giới thực.

(7) Chức năng lưu trữ: Tương ứng với công việc ghi chép thông tin trên sổ sách (kèm theo các quy định khi ghi chép).

Ví dụ:

- Ghi nhận việc cho mượn sách của một thư viện theo quy định mượn.

- Ghi nhận bài giải bài tập về phân số theo quy định về phân số,cách biến đổi phân số tương đương, các phép tính trên phân số,…

(8) Chức năng tra cứu: Tương ứng với công việc tìm kiếm, theo dõi hoạt động và xem thông tin về một đối tượng.

Ví dụ:

- Tìm tài khoản và xem tình hình gửi rút.

- Tìm sách và xem tình trạng sách

- Tìm hàng hóa và xem tình trạng của hàng hóa (số lượng tồn kho, lượng nhập, thời gian nhập,).

- Tìm bài giảng lý thuyết về phương trình, bất phương trình và xem nội dung tương ứng.

(9) Chức năng tính toán: Tương ứng với công việc tính toán (theo quy định và công thức cho trước).

Ví dụ:

- Tính điểm trung bình môn học của học sinh theo quy định hệ số cho các đợt kiểm tra.

- Xếp thứ hạng cho các đội bóng sau một lượt thi đấu theo quy định của ban tổ chức giải.

- Tính tiền phạt trả sách trễ theo quy định phạt của thư viện.

- Tìm nghiệm của phương trình bậc hai theo phương pháp giải phương trình bậc hai.

(10) Chức năng kết xuất : Tương ứng với công việc lập báo cáo (theo biểu mẫu cho trước)

Ví dụ:

- Lập bảng xếp hạng các đội bóng sau một lượt đấu.

- Lập báo cáo thống kê về số lượt mượn sách theo từng thể loại trong năm.

- Lập báo cáo thống kê về tỷ lệ xếp loại học sinh theo từng lớp, từng khối.

(11) Chức năng môi trường : Định cấu hình thiết bị, ngày giờ, số người làm việc, …

Ví dụ: Số lượng người làm việc, chọn loại máy in, khổ giấy, niên khóa hiện hành, …

(12) Chức năng mô phỏng: Mô phỏng hoạt động của thế giới thực

Ví dụ: - Mô phỏng một tai nạn máy bay, xe ô tô, trận động đất

(13) Chức năng tự động: Tự động thông báo, nhắc nhở người dùng.

Ví dụ:

- Nhắc nhở thủ thư gửi giấy báo đòi sách khi có độc giả mượn quá hạn.

- Báo động khi khách hàng thiếu nợ quá lâu hay số tiền nợ quá lớn.

(14) Chức năng phân quyền : Phân quyền sử dụng giữa các loại người dùng.

Ví dụ: Phân quyền cho 3 loại người sử dụng trong phần mềm quản lý thư viện:

+ Quản trị hệ thống: có quyền sử dụng tất cả các chức năng.

+ Thủ thư: chỉ sử dụng các chức năng liên quan đến việc cho mượn và trả sách.

+ Độc giả: chỉ sử dụng chức năng tra cứu.

Trong phần mềm quản lý bán hàng, việc phân chia khả năng truy cập dữ liệu nhập xuất cho từng nhóm người sử dụng sẽ tránh việc điều chỉnh số liệu không thuộc phạm vi quản lý của người sử dụng như nhân viên thu ngân chỉ được phép lập và điều chỉnh các hóa đơn bán hàng trong ca làm việc của mình. Ca trưởng và bộ phận quản lý quầy có thể tham khảo lượng hàng

tồn kho nhưng không được phép điều chỉnh lượng hàng nhập, không được tham khảo vốn hàng xuất, kết quả kinh doanh, …

(15) Chức năng sao lưu : Sao lưu, phục hồi dữ liệu.

Ví dụ: Sao lưu thông tin về các học sinh đã ra trường và chỉ phục hồi lại khi cần thiết

(16) Tính tiến hóa: đây là các yêu cầu liên quan đến việc cho phép người dùng thay đổi lại cách mô tả của một yêu cầu chức năng (các quy định, quy tắc tính toán), một biểu mẫu nào đó khi đang dùng phần mềm đã được chuyển giao. Điều này đòi hỏi phải có dự kiến về các thay đổi trên thành phần dữ liệu và xử lý.

Ví dụ:

- Cho phép thay đổi quy định về số sách cho mượn tối đa, hay mức phạt khi trả trễ.

- Cho phép thay đổi các biên trong quy định về xếp loại học sinh.

(17) Tính tiện dụng: là các yêu cầu liên quan đến hình thức giao diện của phần mềm, thể hiện ở sự tự nhiên, dễ sử dụng, dễ học, đầy đủ thông tin,...

Ví dụ:

- Giao diện nhập hóa đơn bán hàng dạng form, dòng nhập thể hiện bằng ô sáng và báo lỗi khi số liệu nhập làm số lượng tồn kho âm (phần mềm quản lý hàng hóa).

(18) Tính hiệu quả : đây là yêu cầu liên quan đến thời gian thực hiện các chức năng phần mềm, dung lượng lưu trữ, chi phí sử dụng tài nguyên hệ thống như sử dụng tối ưu các không gian, thao tác thực hiện nhanh ...

Ví dụ: Thời gian tra cứu sách, tra cứu độc giả không quá 10 giây.

(19) Tính tương thích: là các yêu cầu liên quan đến việc chuyển đổi dữ liệu giữa phần mềm đang xét và các phần mềm khác, sự nhất quán giữa các màn hình trong hệ thống.

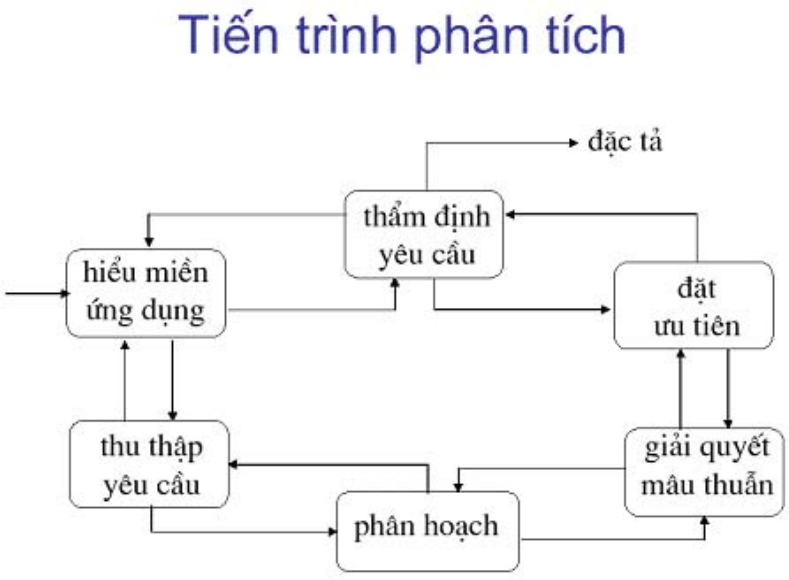
Ví dụ: - Cho phép chuyển tất cả các báo cáo sang định dạng file Excel

- Cho phép nhập thông tin sách mới từ tập tin Excel hay từ thiết bị đọc mã vạch.

- Cho phép thực hiện chức năng bằng giọng nói.

(20) Tính tái sử dụng: (do chuyên viên tin học đảm trách)

(21) Tính bảo trì: (do chuyên viên tin học đảm trách) là các yêu cầu cho phép thay đổi mà không làm ảnh hưởng đến phần mềm



1.2.3 Các bước xác định yêu cầu

Quá trình thực hiện xác định yêu cầu: gồm 2 bước chính như sau

Bước 1: Khảo sát hiện trạng, kết quả nhận được là các báo cáo về hiện trạng.

Bước 2: Lập danh sách các yêu cầu, kết quả nhận được là danh sách các yêu cầu sẽ được thực hiện trên máy tính.

Đối tượng tham gia xác định yêu cầu: gồm 2 nhóm người:

- *Chuyên viên tin học*: những người hiểu rõ về khả năng của máy tính. Họ phải tìm hiểu thật chi tiết về công việc của nhà chuyên môn nhằm tránh sự hiểu nhầm cho những bước phân tích sau này.
- *Nhà chuyên môn*: những người hiểu rõ về công việc của mình. Họ cần lắng nghe ý kiến của các chuyên viên tin học để đảm bảo các yêu cầu của họ là có thể thực hiện được với chi phí và thời gian hợp lý.

Hai nhóm người này cần phải phối hợp thật chặt chẽ để có thể xác định đầy đủ và chính xác các yêu cầu.

Sau đây, chúng ta sẽ phân tích chi tiết từng bước quy trình thực hiện.

1.2.3.1 Khảo sát hiện trạng

Các chuyên viên tin học sẽ tìm hiểu hiện trạng về các công việc của các nhà chuyên môn.

a. Các hình thức thực hiện phổ biến:

- Quan sát: theo dõi các hoạt động đang diễn ra ở thế giới thực có liên quan, có thể tiến hành ghi âm, ghi hình đối với những tình huống mang tính phức tạp, quan trọng, cần sự chính xác cao.

  Ví dụ:
  - Ghi hình quá trình giao dịch của một nhân viên ngân hàng với khách hàng tại một ngân hàng X.
  - Quan sát thao tác cho mượn sách của một thủ thư tại một thư viện Y

- Phỏng vấn trực tiếp: tổ chức phỏng vấn bắt đầu từ cấp lãnh đạo dần xuống các vị trí công việc. Có thể sử dụng các bảng câu hỏi có sẵn các câu trả lời cho đối tượng được phỏng vấn lựa chọn, …

- Thu thập thông tin, tài liệu: các công thức tính toán, quy định; các bảng biểu, mẫu giấy tờ có ít nhiều liên quan.

  Ví dụ:

  - Mẫu hóa đơn và các quy định lập hóa đơn bán hàng tại một cửa hàng Y.

  - Phiếu mượn sách tại thư viện của trường đại học Z.

b.Quy trình thực hiện:

- Tìm hiểu tổng quan về thế giới thực: bao gồm

- Quy mô hoạt động.
- Các hoạt động mà đơn vị có tham gia.

- Tìm hiểu hiện trạng tổ chức (cơ cấu tổ chức)

  Người tiến hành khảo sát hiện trạng cần hiểu rõ cơ cấu tổ chức các bộ phận của thế giới thực, đặc biệt là 2 yếu tố: trách nhiệm và quyền hạn. Sự hiểu rõ cơ cấu tổ chức giúp xác định bộ phận nào sẽ sử dụng phần mềm để có thể lên kế hoạch tiếp tục khảo sát chi tiết hơn bộ phận đó.

  Cơ cấu tổ chức bao gồm:

  - Đối nội.
  - Đối ngoại.
  - Các chức danh (Ví dụ: nhân viên nhập liệu, thủ thư, nhân viên bán hàng, …).

  Sử dụng các đồ hình để vẽ lại cơ cấu tổ chức.

- Tìm hiểu hiện trạng nghiệp vụ

  Thường diễn ra tại các vị trí công việc. Với bộ phận được chọn khảo sát chi tiết, người thực hiện khảo sát cần lập danh sách các công việc mà bộ phận này phụ trách, sau đó tìm hiểu các thông tin chi tiết cho từng công việc (thông tin mô tả yêu cầu phần mềm).

  Việc tìm hiểu dựa trên các ý sau:

  - Thông tin đầu vào.
  - Quá trình xử lý.
  - Thông tin kết xuất.

  Sau đó tiến hành xếp loại các nghiệp vụ vào 4 loại sau nhằm tránh thiếu xót khi tìm hiểu các thông tin:

  - Nghiệp vụ lưu trữ.
  - Nghiệp vụ tra cứu.
  - Nghiệp vụ tính toán.
  - Nghiệp vụ tổng hợp, thống kê

1.2.3.2 Lập danh sách các yêu cầu

Để có được danh sách đầy đủ và chính xác các, quá trình lập danh sách các yêu cầu cầu theo các bước sau:

- Xác định yêu cầu chức năng nghiệp vụ

- Xác định yêu cầu chức năng hệ thống
- Xác định yêu cầu phi chức năng

a. Xác định yêu cầu chức năng nghiệp vụ.

Cách tiến hành: Nhà chuyên môn đề xuất và chuyên viên tin học sẽ xem xét lại

Bước tiến hành :

1. Xác định bộ phận (người dùng) sẽ sử dụng phần mềm
2. Xác định các công việc mà người dùng sẽ thực hiện trên phần mềm theo từng loại công việc sau:
    - Lưu trữ
    - Tra cứu
    - Tính toán
    - Kết xuất

Lần lượt lập bảng yêu cầu chức năng nghiệp vụ, bảng quy định/Công thức và các biểu mẫu – được mô tả chi tiết – như sau:

*Mẫu 1: Bảng yêu cầu chức năng nghiệp vụ

Bộ phận (người thực hiện): … Mã số: ...

stt	Công việc	Loại công việc	Quy định/ Công thức liên quan	Biểu mẫu liên quan	Ghi chú
1					
2					

* Mẫu 2: Bảng Quy định/ Công thức liên quan

stt	Mã số	Tên Quy định/ Công thức	Mô tả chi tiết	Ghi chú
1	QĐ 1			

2	QĐ 2			

Các biểu mẫu được mô tả chi tiết ngay sau bảng quy định/Công thức

Ví dụ: Xét phần mềm quản lý thư viện

Bộ phận: Thủ thư. **Mã số: TT**

stt	Công việc	Loại công việc	Quy định/Công thức liên quan	Biểu mẫu liên quan	Ghi chú
1	Cho mượn sách	Lưu trữ	TT_QĐ 1	TT_BM 1	
2	Nhận trả sách	Lưu trữ	Chỉ nhận lại những sách đã cho mượn	TT_BM 1	
3	Tính tiền phạt	Tính toán	Mỗi ngày trả trễ phạt : - 1000 đồng/ngày : từ ngày thứ nhất đến ngày thứ 5 - 3000 đồng/ngày : từ ngày thứ 6 trở đi.		
4	Tính tiền đền	Tính toán	Tiền đền cho sách bị mất dựa trên giá thị trường tại thời điểm hiện hành.		
5.	Tra cứu sách	Tra cứu	Việc tìm sách dựa trên các thông tin : tên sách, tên tác giả, nhà xuất bản, năm xuất bản		
6.	Gửi giấy báo đòi sách	Kết xuất	Sách mượn quá hạn 3 ngày sẽ tự động gửi giấy báo cho đến khi sách được trả hoặc đã tính xong tiền đền sách	TT_BM 2	

Bảng yêu cầu chức năng nghiệp vụ

stt	Mã số	Tên Quy định/	Mô tả chi tiết	Ghi chú

		Công thức		
1	QĐ 1	Quy định cho mượn sách	Chỉ cho mượn sách khi : - Thẻ độc giả còn hạn - Độc giả chưa mượn hết số sách quy định - Độc giả không có sách mượn quá hạn - Sách hiện không có người mượn	Độc giả mượn sách sẽ phải gửi lại thẻ độc giả tại bộ phận bạn đọc, nhận phiếu mượn sách (TT_BM 1, tìm kiếm mã số sách mượn và điền các sách cần mượn vào phiếu, xong gửi cho thủ thư.

Bảng Quy định/ Công thức liên quan

TT_BM 1:

PHIẾU MƯỢN SÁCH

Số thẻ: Số phiếu mượn:

Họ và tên: Ngày mượn:

[ ] Mượn về nhà [ ] Đọc tại chỗ

STT	Mã sách	Tên sách	Tác giả	Mã loại
1				
2				

Ngày ... tháng ... năm ...

TT_BM 2:

GIẤY BÁO MƯỢN SÁCH QUÁ HẠN

Thân gửi: _____

Địa chỉ: _____

Chúng tôi xin thông báo rằng, anh (chị) đã mượn của thư viện chúng tôi những quyển sách sau:

STT	Mã sách	Tên sách	Ngày mượn	Đến hôm nay đã quá hạn
1				
2				

Vậy thông báo anh(chị) vui lòng đem sách đến trả. Và mang theo số tiền _____ đồng để trả phí sách trễ.

Bộ phận: Độc giả. **Mã số: ĐG**

STT	Công việc	Loại công việc	Quy định/ Công thức liên quan	Biểu mẫu liên quan	Ghi chú
1	Tìm sách	Tra cứu	Việc tìm sách dựa trên các thông tin: tên sách, tên tác giả, nhà xuất bản, năm xuất bản		
2	Đăng ký mượn sách	Lưu trữ	Độc giả phải có thẻ độc giả.	TT_BM 1	Mọi độc giả có thẻ mượn sách đều có thể đăng ký mượn sách. Tuy nhiên, hệ thống sẽ thông báo khi thẻ mượn

					sách của độc giả đã hết hạn sử dụng.

Bộ phận: Quản lý độc giả. **Mã số : QLĐG**

STT	Công việc	Loại việc	Quy định/ Công thức liên quan	Biểu mẫu liên quan	Ghi chú
1	Làm thẻ độc giả mới	Lưu trữ	Chỉ cấp thẻ độc giả có độ tuổi từ 18 trở lên và có chứng minh thư. Lệ phí làm thẻ độc giả là 5000 đồng/thẻ. Một số chứng minh thư chỉ có thể có duy nhất một thẻ độc giả	QLDGBM1 QLDGBM2	Độc giả có yêu cầu làm thẻ mượn sách sẽ được nhận phiếu đăng ký để điền thông tin vào (QLDG_BM 1), sau đó bộ phận quản lý độc giả tiến hành cấp thẻ và thu lệ phí theo quy định (QLDG_BM 2)
2	Gia hạn thẻ độc giả	Lưu trữ	Gia hạn thẻ theo yêu cầu của độc giả và thời gian quá hạn không được quá 3 tháng. Sau thời gian 3 tháng, những thẻ hết hạn sẽ bị hủy.		
3	Huỷ thẻ độc giả	Lưu trữ	Hủy bỏ các thẻ độc giả đã quá hạn đăng ký 3 tháng.		

QLDG_BM 1:

PHIẾU ĐĂNG KÝ LÀM THẺ MƯỢN SÁCH

Họ và tên: _____ Năm sinh: _____

Địa chỉ thường trú: _____

Nghề nghiệp: _____

Ngày đăng ký: _____

QLDG_BM 2:

THẺ ĐỘC GIẢ

Họ và tên: _____

Trường: _____ Lớp: _____

Địa chỉ: _____

Ngày ___ tháng ___ năm __

Bộ phận: Quản lý sách. **Mã số: QLS**

STT	Công việc	Loại	Quy định/ Công thức liên quan	Biểu mẫu liên quan	Ghi chú
1.	Nhận sách mới vào kho sách.	Lưu trữ		QLSBM 1	Khi có sách mới nhập về, bộ phận quản lý sách có trách nhiệm rà xét xem số sách đó đã có hay chưa, nếu chưa thì lập thẻ quản lý sách và định mã số sách mới. Nếu có rồi thì gọi lại thẻ cũ để cập nhật bổ sung số lượng.
2.	Thanh lý sách cũ	Lưu trữ	Các sách hư, không đọc được		
3.	Lập báo cáo các sách cần	Kết xuất		QLS_BM 2	

	thanh lý				
4.	Lập báo cáo sách mượn	Kết xuất		QLS_BM 3	

QLS_BM 1:

THẺ QUẢN LÝ SÁCH

Tên sách: _____

Tập: _____ Số trang: _____

Số lượng: _____Năm xuất bản: _____

Mã ngôn ngữ: _____ Ngôn ngữ: _____

Mã nhà xuất bản: _____ Nhà xuất bản: _____

Mã phân loại: _____ Phân loại: _____

Mã tác giả: _____Tác giả: _____

Mã vị trí: ____ Khu: ___ Kệ: ___ Ngăn: ___

QLS_BM 2:

DANH SÁCH CÁC SÁCH CẦN THANH LÝ

stt	Mã sách	Tên sách	Tác giả	Năm sản xuất	Ngày nhập kho	Tình trạng
1						
2						

Ngày lập báo cáo:

Người lập:

QLS BM 3:

BÁO CÁO THỐNG KÊ SÁCH MƯỢN

Từ ngày _____ Đến ngày _____

stt	Mã sách	Tên sách	Tác giả	Số lượt mượn
1.				
2.				

Ngày lập báo cáo:

Người lập:

b. Xác định yêu cầu chức năng hệ thống và yêu cầu chất lượng

** Cách tiến hành:*

Chuyên viên tin học và nhà chuyên môn cùng đề xuất và cùng xem xét lại các yêu cầu.

** Bước tiến hành:*

Bước 1: Xem xét các yêu cầu chức năng hệ thống cơ bản, thông dụng (yêu cầu phát sinh thêm do thực hiện các công việc trên máy tính): phân quyền, sao lưu, phục hồi, định cấu hình hệ thống, …

Bước 2: Xem xét các yêu cầu chức năng hệ thống chuyên biệt (yêu cầu về các công việc mới, chỉ có thể tiến hành khi thực hiện trên máy tính.

Bước 3: Xem xét các yêu cầu về chất lượng theo từng loại tiêu chuẩn sau:

- Tiến hóa
- Tiện dụng
- Hiệu quả
- Tương thích

Sau đó lập bảng yêu cầu tương ứng theo mẫu sau:

STT	Nội dung	Mô tả chi tiết	Ghi chú
1.			

2.			

Mẫu 3: Bảng yêu cầu chức năng hệ thống.

STT	Nội dung	Tiêu chuẩn	Mô tả chi tiết	Ghi chú
1.				
2.				

Mẫu 4: Bảng yêu cầu về chất lượng.

Ví dụ: Xét phần mềm quản lý thư viện (giả sử phần mềm được xây dựng nhằm phục vụ cho 4 bộ phận là: độc giả, thủ thư, ban giám đốc và quản trị hệ thống).

Bảng yêu cầu chức năng hệ thống:

stt	Nội dung	Mô tả chi tiết	Ghi chú
1	Phân quyền sử dụng	- Người quản trị: được phép sử dụng tất cả các chức năng - Độc giả: chỉ tra cứu sách và đăng ký mượn sách - Ban giám đốc: chỉ tra cứu sách và lập các báo cáo thống kê - Thủ thư: tất cả các chức năng, ngoại trừ chức năng phân quyền, sao lưu và phục hồi dữ liệu	

Bảng yêu cầu về chất lượng hệ thống:

stt	Nội dung	Tiêu chuẩn	Mô tả chi tiết	Ghi chú
1	Cho phép thay đổi quy định tính tiền phạt	Tiến hóa	Người dùng phần mềm có thể thay đổi đơn giá phạt và biên các mức phạt.	
2	Hình thức tra cứu thật tiện dụng, tự nhiên, trực quan. Dễ sử dụng cho cả những người không chuyên tin học.	Tiện dụng	Hỗ trợ khả năng tra cứu gần đúng, tra cứu theo nội dung,...	
3	Cho phép nhập sách mới từ tập tin Excel có sẵn. Các màn hình có sự nhất quán chung	Tương thích	Có thể nhập trực tiếp danh sách các sách mới có trước trên tập tin Excel với cấu trúc hợp lý.	
4	Tốc độ thực hiện việc cho mượn và tra cứu sách nhanh	Hiệu quả	Tối đa 30 giây cho mỗi phiếu mượn sách. Hỗ trợ thiết bị đọc mã vạch. Tối đa 10 giây phải có kết quả tra cứu.	

1.2.4 Khảo sát một số phần mềm tiêu biểu minh họa cho giai đoạn xác định yêu cầu.

A. Phần mềm hỗ trợ giải bài tập phân số.

Bộ phận: Giáo viên. **Mã số: GV**

STT	Công việc	Loại công việc	Quy định/Công thức liên quan	Biểu mẫu liên quan	Ghi chú

1	Soạn tóm tắt lý thuyết và ví dụ minh họa	Lưu trữ			
2	Soạn đề bài tập	Lưu trữ	GV_QĐ 2	GV_BM 2	
3	Soạn đáp án	Lưu trữ	GV_QĐ 3	GV_BM 3	
4	Chấm điểm	Tính toán	GV_QĐ 4		

stt	**Mã số**	**Tên Quy định/ Công thức**	**Mô tả chi tiết**	**Ghi chú**
1.	GV_QĐ2	Quy định soạn đề bài tập	• Đề bài được giới hạn chỉ là biểu thức các phép toán trên phân số với tối đa 4 phân số thành phần. • Có 3 mức bài tập: 1. Chỉ gồm 2 phân số và 1 phép toán. 2. Chỉ gồm 3 phân số và 2 phép toán. 3. Hỗn hợp nhiều phân số (tối đa 4 phân số) với nhiều phép toán • Có 4 loại phép toán : + - * /	
2.	GV_QĐ 3	Quy định soạn đáp án bài tập (cũng là quy định soạn bài giải của học sinh)	Mỗi bước giải chỉ được phép rút gọn biểu thức bằng các thực hiện phép tính trên 2 phân số. Thứ tự thực hiện phép tính theo quy tắc độ ưu tiên như sau : Ưu tiên 1 : nhân chia cao hơn công trừ. Ưu tiên 2 : bài toán ưu tiên bên phải	

			Riêng đối với bài giải của học sinh cho phép bỏ qua các bước trung gian.	
3.	GV_QĐ 4	Quy định chấm điểm	• Có đáp án cuối cùng đúng ➢ Thực hiện hơn hoặc bằng 50% các bước so với đáp án : o Đã rút gọn : 10 o Chưa rút gọn : 8 ➢ Thực hiện dưới 50% các bước so với đáp án : o Đã rút gọn : 9 o Chưa rút gọn : 7 • Có đáp án cuối cùng sai ➢ Thực hiện hơn hoặc bằng 70% các bước so với đáp án : 5 ➢ Thực hiện từ 50% đến dưới 70% các bước so với đáp án : 3 ➢ Thực hiện từ 50% các bước so với đáp án : 0	

GV_BM 2:

Đề bài tập của giáo viên.

Thực hiện các phép tính trên biểu thức các phân số :

<phân số> [phép toán] <phân số> [phép toán] ...

GV_BM 3:

Đáp án của giáo viên (bài giải của học sinh)

Đề bài: _____

Các bước biến đổi tương đương :

Bước 1: ...

Bước 2: ...

Bước 3: ...

Đáp số: ...

Bộ phận: Học sinh. Mã số: HS

stt	Công việc	Loại công việc	Quy định liên quan	Biểu mẫu liên quan	Ghi chú
1	Chọn bài tập	Tra cứu	GV_QĐ 2	GV_BM 2	
2	Giải bài tập	Lưu trữ	GV_QĐ 3	GV_BM 3	
3	Xem tóm tắt lý thuyết	Kết xuất			
4	Xem đánh giá và đáp án	Kết xuất	GV_QĐ 3 GV_QĐ 4	GV_BM 3	

2. Mô hình hóa yêu cầu hệ thống

Các mô tả yêu cầu trong giai đoạn xác định yêu cầu chỉ mô tả chủ yếu các thông tin liên quan đến việc thực hiện các nghiệp vụ trong thế giới thực chưa và chưa thể hiện rõ nét việc thực hiện các nghiệp vụ này trên máy tính. Mô tả thông qua các văn bản dễ gây ra nhầm lẫn và không trực quan.

Ví dụ: Xét yêu cầu lập hóa đơn bán sách, yêu cầu này chỉ mô tả biểu mẫu về hóa đơn, qui định lập hóa đơn và chưa thể hiện cách thức lập hóa đơn trên máy tính

Mục tiêu của mô hình hóa: Cho phép ta hiểu 1 cách chi tiết hơn về ngữ cảnh vấn đề cần giải quyết một cách trực quan và bản chất nhất (thông tin cốt lõi) yêu cầu.

Kết quả: cho một mô hình mô tả lại toàn bộ hoạt động của hệ thống thực. Mỗi phương pháp phân tích đưa ra một kiểu sơ đồ hay mô hình để xây dựng hệ thống.

Kỹ thuật phân tích là cách tiến hành sao cho thu thập được những yêu cầu của người sử dụng từ đó trình bày lại nhu cầu đó trên mô hình, chi tiết hóa sơ đồ hay mô hình bằng đặc tả chức năng, đặc tả dữ liệu thông qua phân tích gốc nhìn, phân tích đối tượng, phân tích dữ liệu thu thập được ở các bước trên. Trước khi đi vào tìm hiểu các phương pháp biểu diễn bằng mô hình, chúng ta hãy xem qua một số nguyên lý phân tích.

2.1 Các nguyên lý mô hình hóa

a. Mô hình hóa miền thông tin (nguyên lý phân tích 1)

Phải hiểu và biểu diễn được miền thông tin

- Định danh dữ liệu (đối tượng, thực thể)
- Định nghĩa các thuộc tính
- Thiết lập các mối quan hệ giữa các dữ liệu

b. Mô hình hóa chức năng (nguyên lý phân tích 2)

Bản chất của phần mềm là biến đổi thông tin

- Định danh các chức năng (biến đối thông tin)
- Xác định cách thức dữ liệu (thông tin) di chuyển trong hệ thống
- Xác định các tác nhận tạo dữ liệu và tác nhân tiêu thụ dữ liệu

c. Mô hình hóa hành vi (nguyên lý phân tích 3)

Phần mềm (hệ thống) có trạng thái (hành vi)

- Xác định các trạng thái hệ thống

ví dụ: giao diện đồ họa, section trong ứng dụng web

- Xác định các dữ liệu làm thay đổi hành vi hệ thống

ví dụ: bàn phím, chuột, các cổng thông tin…

d. Phân hoạch các mô hình (Nguyên lý phân tích 4)

Làm mịn, phân hoạch và biểu diễn các mô hình ở các mức khác nhau

- Làm mịn các mô hình dữ liệu
- Tạo cây (mô hình) phân rã chức năng
- Biểu diễn hành vi ở các mức chi tiết khác nhau

e. Tìm hiểu vấn đề bản chất (Nguyên lý phân tích 5)

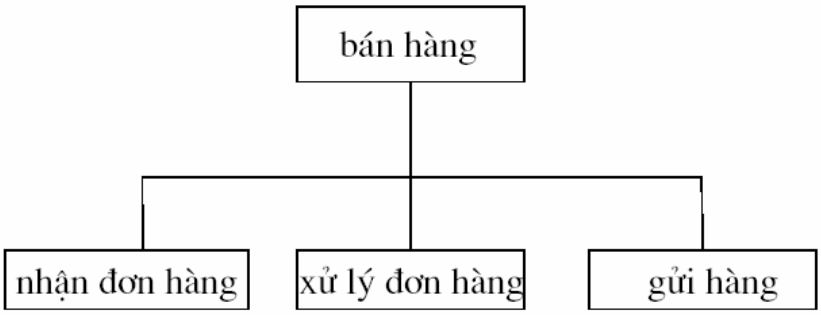
- Nhìn nhận bản chất của yêu cầu
- Không quan tâm đến cách thức cài đặt

2.3 Sơ đồ phân rã chức năng

Sơ đồ phân rã chức năng - Function Decomposition Diagram - FDD: Nêu lên các chức năng thông qua việc mô tả các tính chất của đầu vào và đầu ra

- Xác định phạm vi của hệ thống
- Phân hoạch chức năng
- Tạo nền tảng cho thiết kế kiến trúc hệ thống

Ví dụ: Sơ đồ phân rã chức năng



2.3 Mô hình bản mẫu (protoype)

Khi xác định yêu cầu, nhà phát triển phần mềm dựa trên các ý tưởng hay yêu cầu của khách hàng đưa ra một bản thiết kế sơ bộ một số màn hình giao diện và tiến hành mô phỏng hay giả lập sơ bộ một số chức năng, Có thể xem đây bước cài đặt bản mẫu đầu tiên và chuyển cho người sử dụng. Bản mẫu này chỉ nhằm để mô tả cách thức phần mềm hoạt động cũng như cách người sử dụng tương tác với hệ thống. Nhằn giúp cho người dùng hình dung được diện mạo ban đầu của yêu cầu mà họ đặt ra. Mô hình này cũng cần có sự hỗ trợ giữa kỹ sư phân tích và kỹ sư thiết kế phần mềm phối hợp thực hiện.

Người sử dụng khi xem xét bản mẫu sẽ đưa ra ý kiến đóng góp và phản hồi thông tin đồng ý hay không đồng ý phương án thiết kế của bản mẫu đã đưa ra. Nếu người sử dụng đồng ý với bản mẫu đã đưa thì người phát triển sẽ tiến hành cài đặt thực sự. Ngược lại cả hai phải quay lại giai đoạn xác định yêu cầu. Công việc này được lặp lại liên tục cho đến khi người sử dụng đồng ý với các bản mẫu do nhà phát triển đưa ra.

2.4 Sơ đồ luồng dữ liệu

Sơ đồ luồng dữ liệu - Data flow diagram – DFD

Đây là mô hình cho phép xem toàn bộ sơ đồ luồng dữ liệu bên trong hệ thống. Cách thức dữ liệu được xử lý bên trong hệ thống.Có nhiều mức chi tiết khác nhau. Có nhiều biến thể mở rộng khác nhau. Xem chi tiết ở chương kế tiếp thiết kế phần mềm. Ngoài ra còn có mô hình thực thể kết hợp được trình bày trong hầu hết các cuốn sách Cơ sở dữ liệu hoặc Thiết kế CSDL.

2.5 Mô hình hướng đối tượng

Phương pháp phân tích hướng đối tượng hình thành giữa thập niên 80 dựa trên ý tưởng lập trình hướng đối tượng. Phương pháp này đã phát triển, hoàn thiện và hiện nay rất phổ dụng. Nó dựa trên một số khái niệm cơ bản sau:

Đối tượng (Object): gồm dữ liệu và thủ tục tác động lên dữ liệu này.

Đóng gói (Encapsulation): Không cho phép tác động trực tiếp lên dữ liệu của đối tượng mà phải thông qua các phương pháp trung gian.

Lớp (Class): Tập hợp các đối tượng có chung một cấu trúc dữ liệu và cùng một phương pháp.

Kế thừa (Heritage): tính chất kế thừa là đặc tính cho phép định nghĩa một lớp mới từ các lớp đã có bằng cách thêm vào đó những dữ liệu mới, các phương pháp mới có thể kế thừa những đặc tính của lớp cũ.

a. Mô hình nắm bắt yêu cầu hướng đối tượng bằng UML

Mục đích của hoạt động nắm bắt yêu cầu là xây dựng mô hình hệ thống mà sẽ được xây dựng bằng cách sử dụng các use-case. Các điểm bắt đầu cho hoạt động này khá đa dạng:

- Từ mô hình nghiệp vụ (business model) cho các ứng dụng nghiệp vụ.
- Từ mô hình lĩnh vực (domain model) cho các ứng dụng nhúng (embeded)
- Từ đặc tả yêu cầu của hệ thống nhúng được tạo bởi nhóm khác và hoặc dùng các phương pháp đặc tả khác (thí dụ hướng cấu trúc.

- Từ điểm nào đó nằm giữa các điểm xuất phát trên.

Mô hình use-case:

- Actor: người/ hệ thống ngoài/ thiết bị ngoài tương tác với hệ thống
- Use-case: các chức năng có nghĩa của hệ thống cung cấp cho các actor
  - luồng các sự kiện (flow of events)
  - các yêu cầu đặc biệt của use-case
- Đặc tả kiến trúc
- Các thiết kế mẫu giao diện người dùng

b. Mô hình phân tích hướng đối tượng với UML

Mục đích của hoạt động phân tích yêu cầu là xây dựng mô hình phân tích với các đặc điểm sau:

- Dùng ngôn ngữ của nhà phát triển để miêu tả mô hình
- Thể hiện gốc nhìn từ bên trong hệ thống
- Được cấu trúc từ các lớp phân tích và các package phân tích
- Được dùng chủ yếu cho các nhà phát triển để hiểu cách thức tạo hình dạng hệ thống
- Loại trừ mọi chi tiết dư thừa, không nhất quán
- Phát họa hiện thực các chất năng bên trong hệ thống
- Định nghĩa các dẫn xuất use-case, mỗi dẫn xuất use-case cấp phân tích miêu tả sự phân tích 1 use-case

Mô hình phân tích= hệ thống phân tích

- Các class phân tích: lớp biên, lớp thực thể, lớp điều khiển
- Các dẫn xuất use-case cấp phân tích: các lược đồ lớp phân tích, các lược đồ tương tác, luồng sự kiện, các yêu cầu đặc biệt của use-case
- Các package phân tích
- Đặc tả kiến trúc

Lưu ý: Các mô hình hướng đối tượng cho từng giai đoạn phát triển phần mềm được trình bày ở giáo trình khác. Xem chi tiết cụ thể ở giáo trình môn Phân tích thiết kế hướng đối tượng với UML.

2. 6 Ví dụ minh họa từ yêu cầu sang mô hình hóa

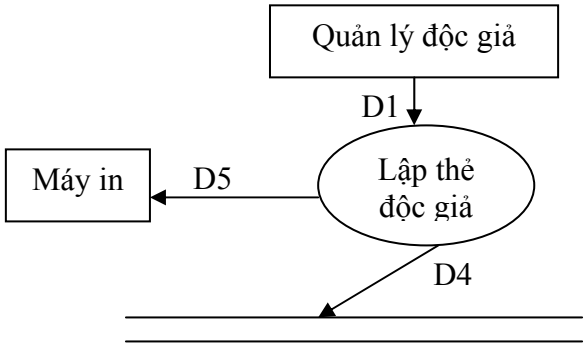
Ví dụ 1: Xét phần mềm quản lý thư viện với 4 yêu cầu

- Lập thẻ độc giả

- Nhận sách
- Cho mượn sách
- Trả sách

Giai đoạn 2 : Mô hình hóa yêu cầu

- Sơ đồ luồng dữ liệu cho công việc lập thẻ độc giả



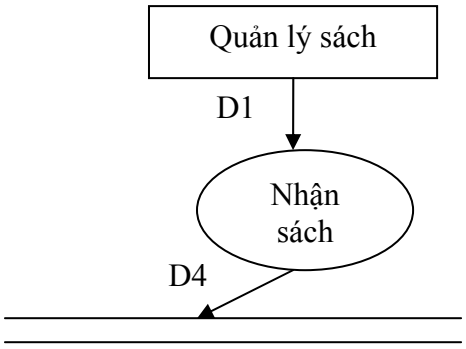
D1: Thông tin về thẻ độc giả cần nhập

D4: Thông tin về thẻ độc giả cần lưu trữ trên bộ nhớ phụ

D5: Thông tin trên thẻ độc giả (trong thế giới thực)

Xử lý thẻ độc giả: Kiểm tra tính hợp lệ của thẻ trước ghi nhận và in

- Sơ đồ luồng dữ liệu cho công việc nhận sách

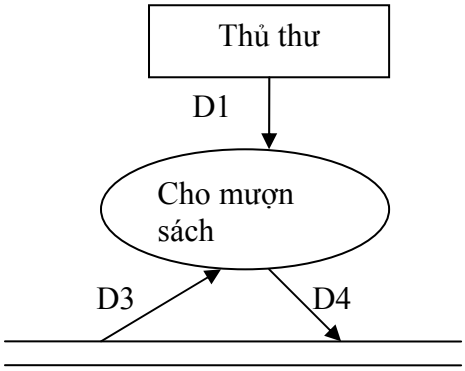


D1: Thông tin về thẻ sách cần nhập

D4: Thông tin về sách cần lưu trữ trên bộ nhớ phụ

Xử lý nhập sách: Kiểm tra tính hớp lệ của sách trước khi ghi nhận trên bộ nhớ phụ

- Sơ đồ luồng dữ liệu cho công việc cho mượn sách



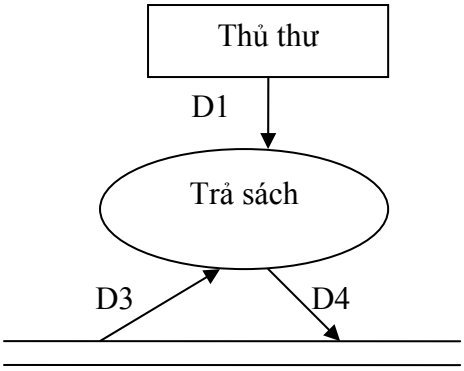
D1: Thông tin về độc giả và sách muốn mượn

D3: Thông tin được sử dụng cho việc kiểm tra qui định mượn sách

D4: Thông tin về việc mượn sách

Xử lý cho mượn sách: Kiểm tra tính hợp lệ của việc mượn sách ghi nhận trên bộ nhớ phụ

- Sơ đồ luồng dữ liệu cho công việc trả sách



D1: Thông tin về độc giả và sách trả

D3: Thông tin sử dụng cho việc kiểm tra qui định trả sách

D4: Thông tin về việc trả sách

Xử lý trả sách: Kiểm tra tính hợp lệ của việc trả sách ghi nhận trên bộ nhớ.

Chương 3: THIẾT KẾ PHẦN MỀM

1. Tổng quan về thiết kế

Trong thiết kế, chúng ta định hình hệ thống và tìm dạng thức của nó (kể cả kiến trúc) mà đáp ứng được mọi yêu cầu, cả yêu cầu phi chức năng và các ràng buộc khác - được đặt ra cho hệ thống đó. Một đầu vào cơ bản cho thiết kế là kết quả thu được từ phần tích, đó là mô hình phân tích. Xét một cách chi tiết mục đích của thiết kế là:

- Thu được sự hiểu biết sâu về các yêu cầu phi chức năg và các ràng buộc có liên quan tới ngôn ngữ lập trình, sử dụng lại thành phần, các hệ điều hành, các công nghệ phân tán, các công nghệ cơ sở dữ liệu, các công nghệ giao diện người dùng, các công nghệ quản lý các giao dịch.
- Tạo ra một đầu vào thích hợp và xuất phát điểm cho các hoạt động cài đặt tiếp theo sau bằng cách nắm bắt các yêu cầu về mỗi hệ thống cụ thể, các giao diện, và các lớp.
- Có khả năng phân rã việc cài đặt thành các mẩu nhỏ dễ quản lý hơn được nhiều đội phát triển khác nhau xử lý và có thể tiến hành đồng thời. Điều này sẽ có ích trong các trường hợp khi mà không thể tiến hành sự phân rã giữa các kết quả thu được từ nắm bắt các yêu cầu hoặc phân tích.
- Nắm bắt sớm các giao diện chủ yếu giữa các hệ thống con trong vòng đời của phần mềm. Điều này sẽ có ích khi chúng ta suy luận về kiến trúc và khi chúng ta sử dụng các giao diện như những công cụ đồng bộ các đội phát triển khác nhau
- Trực quan hóa và suy luận thiết kế bằng cách sử dụng một hệ thống các ký pháp chung.
- Tạo ra một sự trừu tượng hóa liên tục của việc cài đặt của hệ thống, tức là cài đặt sự làm mịn dần thiết kế bằng cách đắp "thịt" vào khung xương nhưng không thay đổi cấu trúc của nó.

Mục tiêu của phần này là giới thiệu một số phương pháp và kỹ thuật chính trong thiết kế, đối với việc triển khai một hệ thống thành nhiều hệ thống con và hệ thống con thành nhiều thành phần (components), và quản lý những vấn đề liên quan đến cấu trúc nội tại của những thành phần hệ thống. Đầu tiên chúng ta xem qua vài kỹ thuật thiết kế. Kế đến chúng ta sẽ xét qua một vài kỹ thuất thiết kế và phương pháp nền tảng một cách chi tiết và một số ví dụ minh họa. Thêm vào đó, chúng ta bàn qua những khía cạnh thiết kế như thiết kế giao diện người dùng và mô đun hóa.

1.1 Kỹ thuật thiết kế

- Thiết kế đặc tả đi đến kỹ thuật cốt lõi của tiến trình của công nghệ phần mềm.
- Thiết kế đặc tả được cung cấp xem xét những mô hình của tiến trình phần mềm được sử dụng.
- Thiết kế phần mềm là bước đầu tiên trong ba hoạt động kỹ thuật - thiết kế, phát sinh mã nguồn, và thử nghiệm –đó là những yêu cầu trong xây dựng và phát triển phần mềm.

Một trong những điểm mấu chốt chính đối với độ phức tạp của hệ thống phần mềm là sự trừu tượng. Có hai phương pháp chính: thiết kế Top-down và thiết kế bottom-up

1.1.1 Thiết kế trên xuống (Top-down)

-Thiết kế bắt đầu với việc phân tích những định nghĩa yêu cầu và không nên xem xét việc thực hiện chi tiết đầu tiên.

- Một dự án được triển khai thành những dự án nhỏ, thủ tục này phải được lặp lại cho đến khi những nhiệm vụ con trở nên đơn giản sao cho một thuật toán được tính toán và giải quyết.

1.1.2 Thiết kế từ dưới lên (Bottom–up)

Ý tưởng nền tảng: Hiểu được phần cứng và tầng trên của nó như một cơ chế trừu tượng.

Kỹ thuật: Thiết kế từ dưới lên bắt đầu được cho bởi máy cụ thể và liên tiếp phát triển một máy trừu tượng sau khi những máy khác được thêm vào những thuộc tính cần thiết cho đến khi một máy đã đạt được kết quả mà cung cấp những chức năng người dùng yêu cầu.

1.1.3 Thiết kế hệ thống

Trong hệ thống lớn, tiến trình thiết kế bao gồm một yếu tố thiết kế hệ thống mà chức năng được phân chia thành những chức năng phần mềm và phần cứng.

Những thuận lợi của chức năng thực hiện trong phần cứng là thành phần phần cứng phân phối thực hiện tốt hơn đơn vị phần cứng. Nút thắt của hệ thống được xác định và thay thế bởi thành phần của phần cứng, như thế việc tối ưu phần mềm là hết sức tốn kém.

Cung cấp tốc độ phần cứng có nghĩa là thiết kế phần mềm có thể được cấu trúc cho khả năng thích ứng và khả năng xem xét thực thi cả chức năng.

1.1.4 Thiết kế bản mẫu (prototype)

Thiết kế bản mẫu nghĩa là đưa ra các màn hình giao diện sơ bộ, hay các bản thiết kế phác thảo nháp cho người dùng tham khảo trước khi đi vào thiết kế chi tiết, hay chức năng cụ thể. Các bản thiết kế này được soạn thảo dưới dạng sưu liệu hoặc một số phần mềm có khả năng thiết kế nhanh giao diện, các kỹ sư thiết kế có thể sử dùng một số phần mềm chuyên dụng để soạn thảo nhanh như MS Visual Basic, Visual C++, MS Visual Studio … với trang web thì có thể dùng Front Page, MS Visual Interdev chỉ với những đoạn chương trình đơn giản được cài đặt. Đây cũng có thể coi là bước đệm cơ bản trước khi đi vào cài đặt chi tiết cho từng chương trình con hay môđun con v.v.

1.1.5 Phân rã thiết kế

Tiến trình thiết kế không chỉ ảnh hưởng đến phương pháp thiết kế mà còn ảnh hướng đến tiêu chuẩn được sử dụng để phân rã hệ thống.

Phần lớn những yếu tố cơ bản của phân rã được đề ra.

Phương pháp phân loại phân rã

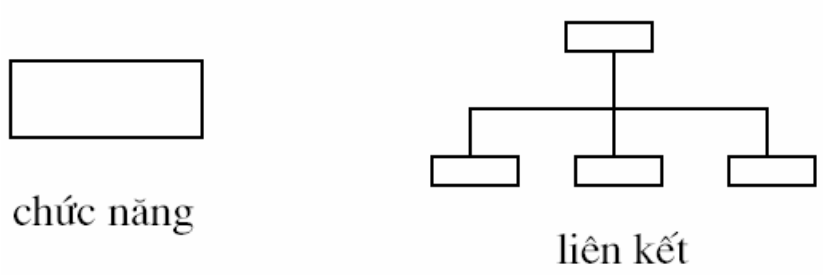
1.1.5.1 Phân rã hướng chức năng

- Khía cạnh của hệ thống hướng chức năng tạo nên cốt lõi của thiết kế

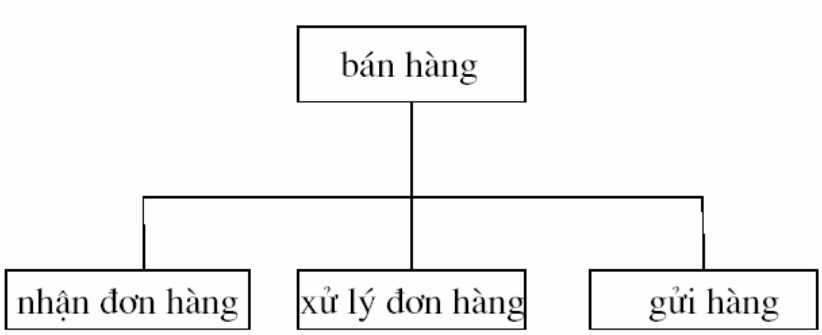
- Dựa trên những yêu cầu chức năng chứa trong những định nghĩa yêu cầu, phân rã hướng đến tác nhiệm của toàn bộ hệ thống được tổ chức

Sơ đồ phân rã chức năng - Function Decomposition Diagram - FDD: Nêu lên các chức năng thông qua việc mô tả các tính chất của đầu vào và đầu ra

- Xác định phạm vi của hệ thống
- Phân hoạch chức năng
- Tạo nền tảng cho thiết kế kiến trúc hệ thống



Ví dụ: **Sơ đồ phân rã chức năng**



1.1.5.2 Phân rã hướng dữ liệu

Tiến trình thiết kế tập trung trên khía cạnh hệ thống hướng đến dữ liệu. Chiến lược thiết kế hướng đến chính dữ liệu được thực hiện. Phân rã những bộ phận hệ thống từ việc phân tích dữ liệu

1. Sơ đồ luồng dữ liệu

Sơ đồ luồng dữ liệu - Data flow diagram - DFD

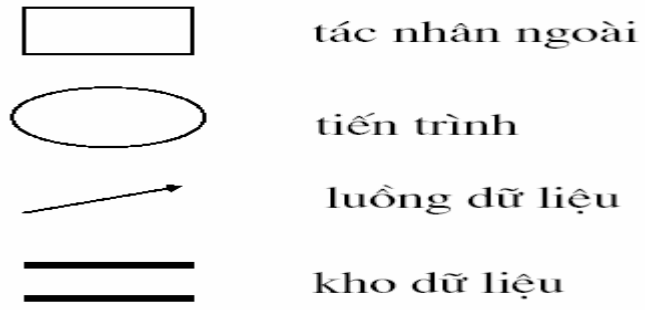
Cho phép xem toàn bộ sơ đồ luồng dữ liệu bên trong hệ thống. Cách thức dữ liệu được xử lý bên trong hệ thống.Có nhiều mức chi tiết khác nhau. Có nhiều biến thể mở rộng khác nhau

a. Khái niệm và ký hiệu

Tác nhân ngoài: đối tượng bên ngoài hệ thống, nguồn phát sinh hay thu nhận dữ liệu

Tiến trình: Thao tác đối với thông tin hay khối dữ liệu
Luồng dữ liệu: luồng thông tin di chuyển trong hệ thống
Kho dữ liệu:nơi lưu trữ dữ liệu

Các ký hiệu:



b. Các nguyên tắc và bước xây dựng mô hình DFD

Các bước xây dựng DFD:

- Phân rã chức năng hệ thống
- Liệt kê các tác nhân, các khoản mục dữ liệu
- Vẽ DFD cho các mức

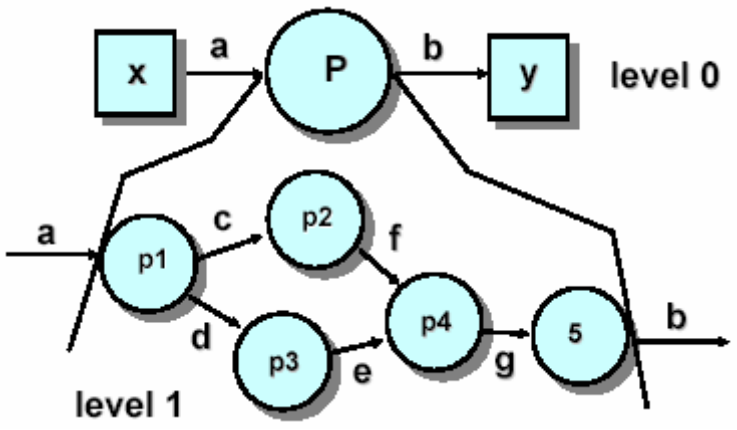
Nguyên tắc:

- Các tiến trình phải có luồng vào luồng ra
- Không có luồng dữ liệu trực tiếp giữa các tác nhân với tác nhân và kho dữ liệu
- Luồng dữ liệu không quay lại nơi xuất phát
- Bắt đầu bằng DFD mức 0, liệt kê các tác nhân ngoài ở mức 0
- Các mức(cấp) sơ đồ:
  - mức 0: Toàn bộ phần mềm là khối xử lý
  - mức 1: Sơ đồ mức 0 có thể phân rã thành nhiều sơ đồ mức 1, các sơ đồ mức 1này phải đảm bảo thể hiện đầy đủ ý nghĩa sơ đồ mức 0 (tác nhân, thiết bị, luồng dữ liệu, xử lý, bộ nhớ phụ)
  - mức 2: Mỗi sơ đồ mức 1 có thể phân rã thành nhiều sơ đồ mức 2 tương ứng như việc phân rã của sơ đồ mức 0
  - …

Trình bày sơ đồ: Trong mỗi cấp có 2 hình thức trình bày sơ đồ

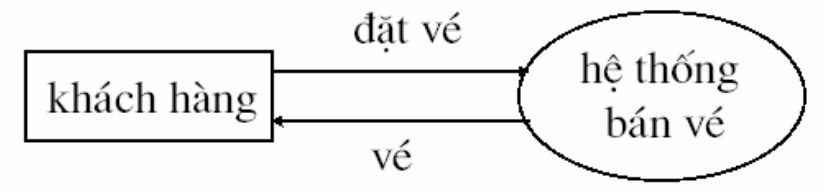
- Dạng tổng hợp : Chỉ có một khối xử lý chung, tất cả các luồng dữ liệu chỉ tập trung liên quan đến khối xử lý chung này
- Dạng chi tiết: Bao gồm nhiều khối xử lý với luồng dữ liệu riêng biệt cho từng khối xử lý

Ví dụ: biểu diễn các mức của DFD

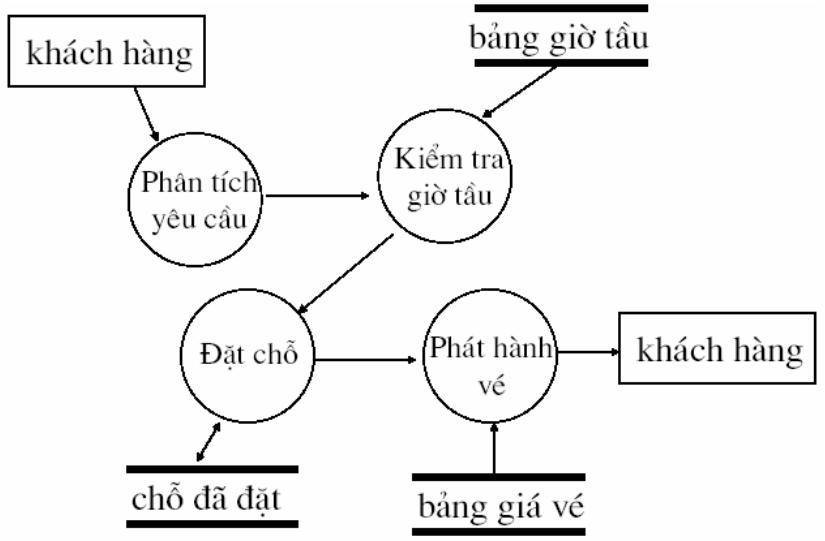


Ví dụ DFD hệ thống bán vé

mức 0:



mức 1: DFD mức 1



2. Các hướng tiếp cận lập sơ đồ luồng dữ liệu

- Có nhiều hướng tiếp cận để tạo lập các sơ đồ luồng dữ liệu. Giáo trình này giới hạn xem xét 3 cách tiếp cận chính

  + Hướng tiếp cận từ trên xuống dưới (topdown)

  + Hướng tiếp cận từ dưới lên trên (bottomup)

  + Hướng tiếp cận phối hợp

- **Tiếp cận từ trên xuống:**

Quá trình thực hiện theo hướng tiếp cần này như sau:

- Lập sơ đồ luồng dữ liệu cấp 0 (xem xét tất cả các luồng dữ liệu nhập xuất, tất cả các yêu cầu xử lý của phần mềm
- Phân rã sơ đồ luồng dữ liệu cấp 0 thành nhiều sơ đồ luồng dữ liệu cấp 1. Có 2 cách phân rã:

  + Phân rã các xử lý của phần mềm thành nhiều xử lý con và quyết định các luồng dữ liệu tương ứng trên các xử lý con này.

  + Phân rã các luồng dữ liệu nhập xuất thành nhiều luồng dữ liệu con và quyết định các xử lý tương ứng với các luồng dữ liệu con này.

- Quá trình kết thúc khi đạt đến các sơ đồ không thể tiếp tục phân rã được (sơ đồ lá). Thông thường đây là sơ đồ tương ứng với công việc cụ thể của một nhà chuyên môn trong thế giới thực.

Đánh giá

- Tiếp cận này thích hợp với các phần mềm có số lượng người dùng, số lượng các yêu cầu ít (nếu ngược lại sơ đồ cấp 0 sẽ rất phức tạp và khó lập chính xác).
- Tiếp cận này đặc biệt thích hợp với các loại phần mềm mà vì lý do nào đó các hệ thống yêu cầu chưa được xác định rõ ngay từ đầu (ví dụ các phần mềm hệ thống).
- Thông thường cách tiếp cận này ít đươc sử dụng.

▪ Hướng tiếp cận từ dưới lên (bottomup)

Quá trình thực hiện theo hướng tiếp cận này như sau

- Lập sơ đồ luồng dữ liệu ở mức cao nhất. Các sơ đồ này sẽ không được tiến hành phân rã thành các sơ đồ có cấp lớn hơn (thông thường đây là sơ đồ ứng với một công việc cụ thể của một người dùng nào đó trong thế giới thực)

  + Tích hợp các sơ đồ này để tạo lập các sơ đồ có cấp nhỏ hơn (thông thường các sơ đồ được chọn tích hợp theo một tiêu chí cụ thể: cùng một người sử dụng, cùng một loại yêu cầu, v.v). Có 2 cách tích hợp:

  + Tích hợp các xử lý của các sơ đồ cấp k vào sơ đồ cấp k-1 và giữ nguyên các luồng dữ liệu của các sơ đồ cấp k

  + Tích hợp đồng thời các xử lý và các luồng dữ liệu của các sơ đồ cấp k để tạo lập sơ đồ cấp k-1.

- Quá trình kết thúc khi đạt đến các sơ đồ cấp 0

Đánh giá

- Tiếp cận này rất thích hợp với các phần mềm có hệ thống yêu cầu chi tiết, cụ thể và có qui mô yêu cầu (số lượng người dùng, số lượng yêu cầu) thuộc mức trung bình (các đồ án môn học
- Tiếp cận này sẽ khó khăn nếu qui mô yêu cầu lớn và chưa thật rõ ràng chi tiết
- Cách tiếp cận này sẽ được sử dụng trong giáo trình với các đồ án môn học và các ví dụ minh họa

▪ **Hướng tiếp cận phối hợp:**

Quá trình thực hiện theo hướng tiếp cận này như sau:

- Lập sơ đồ luồng dữ liệu cấp k theo một tiêu chí xác định (sơ đồ cho từng người dùng, sơ đồ cho một bộ phận, sơ đồ cho một loại yêu cầu, v.v)
- Phân rã sơ đồ cấp k thành nhiều sơ đồ cấp k+1 tiếp tục cho đến khi đạt được các sơ đồ lá
- Tích hợp các sơ đồ cấp k thành các sơ đồ cấp k-1 tiếp tục cho đến khi đạt được sơ đồ cấp 0

Đánh giá

- Tiếp cận này thích hợp cho các phần mềm có qui mô yêu cầu lớn, phức tạp
- Tiếp cận này được sử dụng rất thường xuyên trong thực tế.

3. Lập sơ đồ luồng dữ liệu cho từng công việc

- Do các giới hạn đã nêu phía trên việc lập các sơ đồ luồng dữ liệu toàn bộ phần mềm chỉ qui về lập sơ đồ luồng dữ liệu cho từng công việc (sau đó chỉ thực hiện đơn giản một bước tích hợp để có sơ đồ cấp 0)
- Quá trình lập sơ đồ luồng dữ liệu cho một công việc được tiến hành qua các bước như sau
    - Bước 1: Xác định dữ liệu nhập
    - Bước 2: Xác định dữ liệu xuất
    - Bước 3: Mô tả xử lý
- **Bước 1: Xác định dữ liệu nhập**

- Dữ liệu nhập từ người dùng sử dụng được xác định dựa vào biểu mẫu có liên quan với các lưu ý sau:

  + Không nhập vào các dữ liệu có thể tính toán được dựa trên qui định hay công thức đã có.

  + Không nhập vào các dữ liệu đã được lưu trữ trước đó (qua một công việc khác).

- Dữ liệu nhập từ thiết bị nhập (khác bàn phím) chỉ được xem xét khi có yêu cầu đặc biệt trong một số ứng dụng đặc biệt (hệ thống thời gian thực, hệ thống bản đồ, nhập thông qua sử dụng điện thoại tổng đài điện thoại trong quản lý khách sạn, v.v).

- Dữ liệu nhập (đọc) từ bộ nhớ phụ được xác định dựa trên các qui định công thức liên quan với một số lưu ý:

  + Chỉ đọc dữ liệu thật sự cần thiết cho việc thực hiện xử lý tương ứng (thông tin nhập chưa đủ để xử lý).

+ Để cải tiến chất lượng phần mềm(đặc biệt tính tiến hóa) có thể đọc thêm các tham số phục vụ cho việc xử lý từ bộ nhớ phụ (bảng qui định đơn giá phạt khi trả sách trễ hạn, bảng định mức và đơn giá tiền điện, v,v). Tuy nhiên trong giai đoạn này chỉ nên tập trung vào tính đúng đắn (các chất lượng khác sẽ được xem xét chi tiết trong giai đoạn thiết kế).

- **Bước 2: Xác định dữ liệu xuất**

- Dữ liệu xuất cho người dùng được xác định dựa trên biểu mẫu liên quan với một số lưu ý như sau

  +Các thông báo về việc xử lý có thực hiện được hay không là luôn luôn phải có và không cần thiết thể hiện trên sơ đồ (thông báo việc mượn sách là không hợp lệ, thông báo lỗi khi tính điểm trung bình mà có môn chưa có điểm, v.v)

  + Để tăng tính tiện dụng, trong tất cả các xử lý đều phải xuất cho người dùng nhiều thông tin (kể cả xử lý lưu trữ, xử lý tính toán). **Tuy nhiên vấn đề này chỉ xem xét và thực hiện trong các giai đoạn sau, nếu chú ý quá sớm đến vấn đề này sẽ làm phức tạp sơ đồ và dễ phạm các sai lầm trong tính đúng đắn.**

- Dữ liệu xuất ra thiết bị xuất (khác màn hình) thông thường là máy in, để tăng tính tiện dụng có thể tuân theo nguyên tắc sau "Tất cả dữ liệu xuất ra màn hình đều cho phép người dùng xuất ra máy in (có thể với cách trình bày khác). Tuy nhiên vấn đề này cũng có thể dời lại xem xét chi tiết trong giai đoạn thiết kế. Các loại thiết bị xuất khác chỉ có trong các loại ứng dụng đặc biệt hoặc do yêu cầu tính tương thích.

- Dữ liệu xuất (ghi) vào bộ nhớ phụ được xác định dựa trên biểu mẫu liên quan với một số lưu ý như sau:

  + Ghi các dữ liệu kết quả mới tạo lập hoặc các dữ liệu đã có nhưng bị thay đổi trong quá trình thực hiện xử lý.

  + Để tăng tính hiệu quả có thể ghi các thông tin bổ sung có liên quan đến các yêu cầu khác. Tuy nhiên tốt nhất vấn đền này được xem xét chi tiết trong giai đoạn thiết kế.

- **Bước 3: Mô tả xứ lý**

Mô tả quá trình sử dụng dữ liệu nhập D1, D2, D3 để tạo ra các dữ liệu xuất D4, D5, D6 với các lưu ý sau:

- Chỉ mô tả xử lý mà không cần lưu ý đến cách thực hiện nhập xuất (hình thức nhập, tổ chức lưu trữ trên bộ nhớ phụ, câu lệnh cụ thể để đọc, ghi).

- Mô tả chi tiết cách sử dụng dữ liệu nhập để tạo dữ liệu xuất (mô tả càng chi tiết thì việc thiết kế xử lý càng dễ dàng.
- Chỉ chú trọng đến tính đúng đắn mà không nên xem xét quá sớm các yêu cầu chất lượng khác.

Mô tả chính xác thứ tự nhập và xuất (trong một vài trường hợp có thể xuất trước và sau đó mới nhập).

2. Mô hình thực thể quan hệ (Entity – Relation Diagram)

a.Các khái niệm và ký hiệu

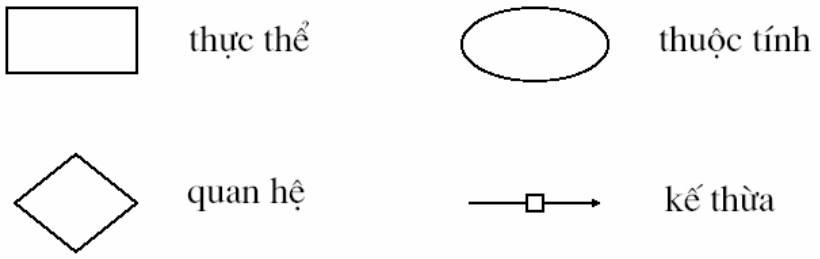
Thực thể là đối tượng thế giới thực mà chúng ta muốn xử lý, có thể là đối tượng thực hay trừu tượng

Thuộc tính: đặc điểm của thực thể

Quan hệ: là mối liên hệ giữa các thực thể, là thông tin cần lưu trừu/ xử lý

Kế thừa: là quan hệ kế thừa giữa các thực thể

b. Ký hiệu



Với giáo trình này chỉ dừng lại giới thiệu khái niệm, mô hình được học ở các giáo trình Phân tích Thiết kế hệ thống thông tin

1.1.5.3 Phân rã hướng đối tượng

Khía cạnh hệ thống hướng đối tượng cung cấp tập trung chủ yếu của thiết kế.

Hệ thống phần mềm được xem xét như tập hợp các đối tượng thông tin với nhau. Mỗi đối tượng có cấu trúc dữ liệu mà không được nhìn thấy từ bên ngoài và thao tác của chúng có thể được thực hiện trên cấu trúc này.

Những điểm cơ bản của phân rã hướng chính nó đến tính đồng nhất giữa dữ liệu và thao tác và dựa trên sự che dấu thống tin và dẫn xuất kế thừa.

1.2. Thiết kế giao diện người dùng

Thiết kế giao diện người dùng là một tác nhiệm trong giai đoạn thiết kế. Thiết kế giao diện được hỗ trợ một phần trong thiết kế dạng mô hình bản mẫu (prototype) ở giaiđoạn xác định nhằm làm sáng tỏ các yêu cầu từ người dùng, xác định đúng yêu cầu người dùng, cũng như thõa mãn các đòi hỏi về mặt thẩm mỹ, giao diện đẹp cho khách hàng. Nếu khách hàng đã đồng ý với bản mẫu đã đưa ra trong giai đoạn xác định yêu cầu thì kỹ sư thiết kế chỉ việc phát triển và hoàn chỉnh thêm giao diện để đảm bảo tính tiện dụng, đảm bảo chính xác yêu cầu người dùng. Nếu không, người thiết kế phải sáng tạo thêm theo một số tiêu chí về thẩm mỹ, tiện dụng, đầy đủ yêu cầu thông tin:

Chế độ (Modes):

Chế độ chương trình là trường hợp mà người dùng chỉ có thể thực hiện ở một số thao tác giới hạn.Kỹ thuật cửa sổ cung cấp dịch vụ có giá trị của chế độ chương trình.

Cửa sổ trợ giúp, người dùng có thể thực hiện vài thao tác con tương ứng trong những cửa sổ khác nhau thể hiển bởi những chế độ chương trình khác nhau.

Thực đơn (Menu):

Pop-up menu: thiết kế hiệu quả bởi chúng có thể xuất hiện bất cứ vị trí nào và ít đòi hỏi di chuyển chuột (mouse).

Pull-down menu: cho phép cấu trúc tốt hơn việc mở rộng tập lệnh và dễ dàng sử dụng.

Người dùng chọn vào thực đơn bằng chuột để hiển thị tất cả lệnh thao tác trên menu và có thể chọn lệnh giống như sử dụng chuột click vào menu. Chúng ta có thể phân loại menu theo tập lệnh thao tác, tập lệnh thao tác với tham số, tập lệnh chuyển đối chế độ người dùng.

1.3.Cửa sổ hội thoại (dialog window):

Đảm bảo tính đồng nhất trong giao diện người dùng, tránh những giải thích dài dòng nên ngắn gọn cô động như cách đặt nhãn Label, Checkbox, Button, List box.

Màu sắc (Color):

Màu sắc chủ yếu chỉ dùng ở những nơi cần diễn đạt những yêu cầu nào đó, hay muốn nhấn mạnh ý nghĩa nào đó, hoặc dấu hiệu cảnh báo nguy hiểm, cùng đừng hóa tô điểm qúa cho giao diện. Ví dụ màu chữ đen trên nên trắng thường dễ đọc nhất cho khả năng làm việc hàng ngày, còn màu chữ trắng trên nền xanh thì khó đọc …

Âm Thanh (Sound):

Cách tốt nhất tập trung sự chú ý của người dùng. Ứng dụng phù hợp trong các tình huống xử lý lỗi, sự kiện không chắc chắn, tạm thời. Tạo những âm thanh khác nhau với những sự kiện khác nhau, tránh dùng âm thanh gây ồn.

Tính kiên định:

Menu lệnh với những chức năng giống nhau nên vị trí giống nhau thậm chí ở những chương trình khác nhau. Phím nóng trên menu lệnh nên cố định. Nút lệnh với những chức năng tương tự giống nhãn và vị trí liên hệ như nhau trong những cửa sổ hội thoại.

1.4 Thiết kế hướng chức năng

Thiết kế hướng chức năng có nghĩa là tập trung trên thuật toán để giải quyết vấn đề.

Hãy tưởng tượng một thuật toán như một hàm tính toán mà tính kết quả từ những tham số cơ bản được cho. Tại thời điểm bắt đầu giai đoạn thiết kế, thuật toán như là hộp đen mà nội dung thì không được biết. Những tác nhiệm càng khó để giải quyết là thuật toán giải quyết của nó. Như vậy, rõ ràng thực hiện mô đun hóa để phân rã những tác nhiệm thành tác nhiệm con độc lập nhau, nhờ đó những thuật toán cho những giải quyết của tác nhiệm con được xem như là những hộp đen. Kết quả chung của những giải pháp trở thành mạng những thuật toán con gộp lại.

1.5.Thiết kế hướng đối tượng

Thiết kế hướng đối tượng là tổ chức xoay quanh những đối tượng và mối liên hệ giữa chúng

Thiết kế lớp đối tượng: mô tả các lớp đối tượng (thuộc tính, hành động

Thiết kế giao diện: Mô tả giao diện của lớp đối tượng trong từng trách nhiệm của chúng

Thiết kế dữ liệu: Mô tả cách thức tổ chức lưu trữ các đối tượng trên bộ nhớ phụ (chỉ có khi không sử dụng cơ sở dữ liệu hướng đối tượng)

Khả năng dùng lại đóng vai trò quan trong trong lập trình hướng đối tượng đối với chuên viên tin học (phải thực hiện nhiều phần mèm). Với tiếp cận mới việc tái sử dụng sẽ rất dễ dàng, nhanh chóng và tốn ít chi phí nhất có thể có (các phần mềm trong cùng lớp phần mềm bao gồm các đối tượng tương tự như nhau, cách xây dựng đối tượng tương tự như nhau cho các phần mềm khác nhau).

3. Kiến trúc phần mềm

Kiến trúc phần mềm bao gồm các thành phần cơ bản: thành phần giao diện, thành phần xử lý, phần dữ liệu. Khi thiết kế một phần mềm cụ thể, người kỹ sư tin học phải chọn lựa và ra quyết định về các "vật liệu" được dùng trong các thành phần. Sau khi đã quyết định xong, kết quả sẽ được mô tả lại hay đặc tả dưới dạng các bản vẽ phần mềm, dưới dạng sưu liệu.

Kết quả của thiết kế là các mô hình phần mềm. Mô hình cung cấp các thông tin chi tiết về 3 thành phần

- Thành phần giao diện
- Thành phần xử lý
- Thành phần dữ liệu

Thông tin về các thành phần giao diện bao gồm các thông tin sau:

- Nội dung và hình thức trình bày các màn hình giao tiếp của phần mềm. Ý niệm về màn hình giao tiếp sẽ được trình bày chi tiết trong phần thiết kế giao diện.
- Hệ thống các thao tác mà người dùng có thể thực hiện trên màn hình giao tiếp và xử lý tương ứng của phần mềm. Các ý niệm về thao tác và xử lý trên màn hình giao tiếp sẽ được trình bày chi tiết trong phần thiết kế giao diện.

Thông tin về thành phần xử lý bao gồm các thông tin sau:

- Hệ thống các kiểu dữ liệu được sử dụng trong phần mềm. Các kiểu dữ liệu này được mô tả cách tổ chức lưu trữ dữ liệu trong bộ nhớ chính của phần mềm
- Hệ thống các hàm được sử dụng trong phần mềm. Các hàm này sẽ thể hiện tương ứng việc thực hiện một công việc nào đó của thế giới thực trên máy tính (kiểm tra tính hợp lệ việc cho mượn sách, ghi vào sổ việc cho mượn sách…v.v)

Thông tin về các thành phần dữ liệu bao gồm các thông tin liên quan đến cách thức tổ chức lưu trữ các dữ liệu (nội dung của việc ghi chép vào sổ sách trong thế giới thực) trên bộ nhớ phụ.

- Dạng lưu trữ được sử dụng của phần mềm. Ý niệm về dạng lưu trữ (tập tin, cơ sở dữ liệu,..v.v) sẽ được trình bày chi tiết trong phần thiết kế dữ liệu

Hệ thống các thành phần lưu trữ cùng với quan hệ giữa chúng. Ý niệm về thành phần lưu trữ cùng với quan hệ giữa các thành phần này cũng sẽ được trình bày chi tiết trong phần thiết kế dữ liệu

4. Phương pháp thiết kế phần mềm

Tùy thuộc vào qui trình được chọn khi thực hiện phần mềm, việc thiết kế có thể được tiến hành theo 2 phương pháp chính:

- Phương pháp trực tiếp
- Phương pháp gián tiếp

Phương pháp trực tiếp được áp dụng khi thực hiện phần mềm không thông qua giai đoạn phân tích. Với phương pháp này việc thiết kế sẽ nhận kết quả chuyển giao trực tiếp từ giai đoạn xác định yêu cầu. Mô hình phần mềm sẽ được xây dựng trực tiếp từ các yêu cầu. Cách tiếp cận này sẽ rất khó khăn cho người thực hiện với các phần mềm có qui mô lớn (nhiều yêu cầu, yêu cầu phức tạp. v.v).

Với phương pháp trực tiếp, thiết kế phần mềm là quá trình cho phép chuyển đổi từ các yêu cầu (kết quả giai đoạn xác định yêu cầu) đến mô hình phần mềm tương ứng. Mục tiêu chính của việc thiết kế là mô tả các thành phần của phần mềm (thành phần giao diện, thành phần xử lý, thành phần dữ liệu) tương ứng với các yêu cầu của phần mềm (yêu cầu chức năng nghiệp vụ, yêu cầu chức năng hệ thống, yêu cầu phi chức năng.

Phương pháp gián tiếp được áp dụng với các qui trình có giai đoạn phân tích. Với phương pháp này việc thiết kế sẽ chỉ nhận một phần các kết quả chuyển giao trực tiếp từ giai đoạn xác định yêu cầu, phần chính yếu sẽ được nhận gián tiếp qua giai đoạn phân tích.

Mô hình phần mềm sẽ được xây dưng tương ứng theo các mô hình trong giai đoạn phân tích. Cách tiếp cận này sẽ rất thuận lợi trong đa số trường hợp với các phần mềm qui mô lớn.

Với phương pháp gián tiếp, thiết kế phần mềm là quá trình cho phép chuyển từ mô hình thế giới thực (kết quả giai đoạn phân tích) đến mô hình phần mềm tương ứng. Mục tiêu chính của việc thiết kế là mô tả các thành phần của phần mềm (thành phần giao diện, thành phần xử lý, thành phần dữ liệu) tương ứng với các mô hình của thế giới thực (mô hình xử lý, mô hình dữ liệu).

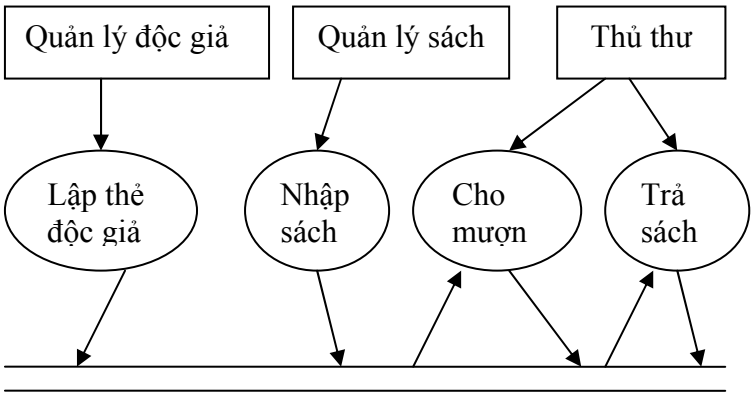
5. Ví dụ minh họa

Các ví dụ sau đây chỉ nhằm **minh họa** quá trình thiết kế phần mềm sau khi thực hiện giai đoạn mô hình hóa yêu cầu, các kết quả chỉ chú trọng chủ yếu **tính đúng đắn** và bỏ qua các yêu cầu chất lượng khác (tiến hóa, hiệu quả, tiện dụng). Kết quả thực tế khi xem xét đầy đủ các yêu cầu chất lượng là quá phức tạp và không thích hợp cho việc min họa

Ví dụ 1: Xét phần mềm quản lý thư viện với 4 yêu cầu

- 1. Lập thẻ đọc giả
- 2. Nhận sách
- 3. Cho mượn sách
- 4. Trả sách

a. Mô hình hóa các yêu cầu



b.Thiết kế phần mềm

Hệ thống các màn hình giao diện

- Màn hình chính:

- Nộidung:
  + Thông tin về thư viện
  + Thông tin về các độc giả trong thư viện
  + Thông tin về các sách trong thư viện
- Thao tác người dùng
  + Tra cứu và chọn độc giả
  + Tra cứu và chọn sách

- Màn hình Lập thẻ

- Nộidung:
  + Thông tin về thẻ độc giả
- Thao tác người dùng
  + Nhập thông tin về thẻ
  + Yêu cầu lập thẻ

- Màn hình Cho mượn sách:

- Nộidung:
  + Thông tin về thẻ độc giả

+ Ngày mượn sách

+ Danh sách các sách muốn mượn

- Thao tác người dùng

+ Nhập thông tin về việc cho mượn sách

+ Yêu cầu cho mượn sách

- Màn hình Nhận sách:

- Nộidung:

+ Ngày nhận sách

+ Danh sách các sách nhận cùng thông tin liên quan

- Thao tác người dùng

+ Nhập thông tin về việc cho nhận sách

+ Yêu cầu cho nhận sách

- Màn hình Trả sách:

- Nộidung:

+ Ngày trả sách

+ Thông tin về việc trả sách

- Thao tác người dùng

+ Nhập thông tin về việc trả sách

+ Yêu cầu trả sách

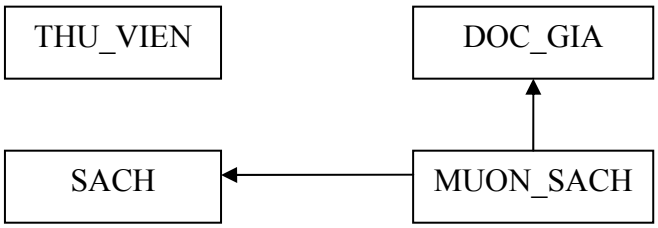
c. Hệ thống các hàm xử lý

- Hàm Lập thẻ: Kiểm tra tính hợp lệ và ghi nhận thẻ trên bộ nhớ phụ
- Hàm Tra cứu độc giả: Tìm thẻ độc giả theo các tiêu chuẩn khác nhau để cho phép cập nhật hay xóa thẻ
- Hàm Xóa thẻ: Xóa thẻ trên bộ nhớ phụ
- Hàm Nhập sách: Kiểm tra tính hợp lệ của sách và ghi nhận sách trên bộ nhớ phụ
- Hàm Xóa sách: Xóa sách trên bộ nhớ phụ
- Hàm Cho mượn sách: Kiểm tra tính hợp lệ của việc cho mượn sách và ghi nhận các thông tin cho mượn sách trên bộ nhớ phụ
- Hàm Tra cứu sách: Tìm sách theo các tiêu chuẩn khác nhau để cho phép cập nhật hay xóa sách
- Hàm Tính số sách độc giả đang mượng: Tính số lượng sách độc giả đang mượn và chưa trả

- Hàm Kiểm tra độc giả có sách mượn quá hạn: Kiểm tra độc giả có sách mượn quá hạn và trả về 1 nếu đúng, 0 nếu sai
- Hàm Kiểm tra tình trạng sách: Kiểm tra sách đang được mượn trả về 1 nếu đúng , 0 nếu sai
- Hàm Tra cứu phiếu cho mượn sách: Tra cứu các phiếu mượn sách theo nhiều tiêu chuẩn để cập nhật hay xó phiếu cho mượn
- Hàm Xóa phiếu cho mượn sách: Xóa thông tin về việc cho mượn sách trên bộ nhớ phụ
- Hàm Trả sách: Ghi nhận việc trả sách trên bộ nhớ phụ
- Hàm Tính tiền phạt: Tính tiền phạt khi độc giả trả sách trễ hạn

d. Hệ thống các bảng dữ liệu:

- **Sơ đồ logic**

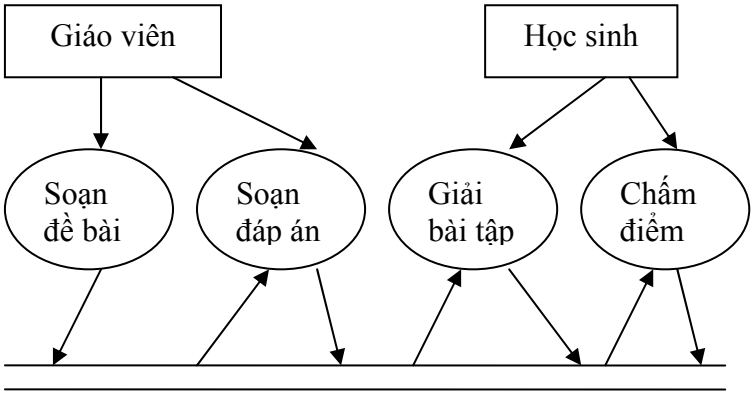


- Bảng THU_VIEN: các thông tin về thư viện
- Bảng DOC_GIA: Các thông tin về độc giả
- Bảng SACH: Các thông tin về sách
- Bảng MUON_SACH: Các thông tin về mượn trả sách

Ví dụ 2: Xét phần mềm hỗ trợ giải bài tập phương trình đại số với 4 yêu cầu

- 1. Soạn đề bài
- 2. Soạn đáp án
- 3. Giải bài tập
- 4. Chấm điểm

a. Mô hình hóa yêu cầu



b. Thiết kế phần mềm

- Màn hình chính:

- Nộidung:

  + Thông tin về sách bài tập

  + Thông tin về các bài tập của sách

- Thao tác người dùng: Tra cứu và chọn bài tập

- Màn hình Soạn đề bài:

- Nộidung:

  + Thông tin về đề bài

- Thao tác người dùng

  + Nhập thông tin về đề bài

  + Yêu cầu phát sinh đề

  + Yêu cầu ghi nhận đề

- Màn hình Soạn đáp án:

- Nộidung:

  + Thông tin về đáp án

- Thao tác người dùng

  + Nhập thông tin về đáp án

  + Yêu cầu ghi nhận đáp án

- Màn hình Nhận bài giải:

- Nộidung:

  + Thông tin về bài giải

- Thao tác người dùng

  + Nhập thông tin về bài giải

  + Yêu cầu ghi nhận bài giải

+ Yêu cầu chấm điểm

- Màn hình Chấm điểm:

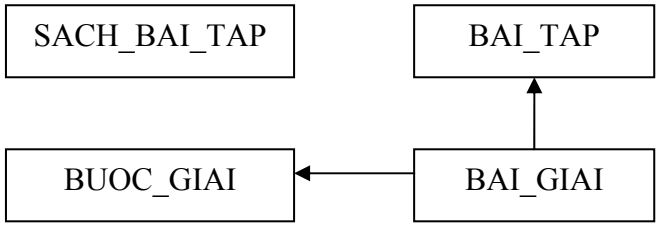
- Nộidung:
  + Thông tin về bài giải
  + Thông tin về việc chấm điểm
  + Thông tin về đáp án
- Thao tác người dùng
  + Xem thông tin điểm
  + Yêu cầu xem đáp án

c. Hệ thống các hàm xử lý:

- Hàm Soạn thảo đề bài: Ghi nhận đề bài của giáo viên trên bộ nhớ phụ (giới hạn không kiểm tra tính hợp lệ của đề bài)
- Hàm Tra cứu bài tập: Tìm kiến bài tập theo nhiều tiêu chuẩn khác nhau để có thể cập nhật, xóa hay soạn đáp án
- Hàm Xóa bài tập : Xóa bài tập trên bộ nhớ phụ
- Hàm Soạn đáp án: Kiểm tra tính hợp lệ của đáp án của giáo viên và ghi nhận đáp án trên bộ nhớ phụ
- Hàm Xóa đáp án: Xóa bài tập trên bộ nhớ phụ
- Hàm Ghi nhận bài giải: Kiểm tra tính hợp lệ bài giải của học sinh và ghi nhận bài giải trên bộ nhớ phụ
- Hàm Biến đổi: Biến đổi một biểu thức thành một đa thức
- Hàm Khai triển: Nhân đa thức
- Hàm Rút gọn: Cộng 2 đa thức
- Hàm so sánh: So sành đa thức
- Hàm Xóa bài giải: Xóa bài giải của học sinh trên bộ nhớ phụ
- Hàm Chấm điểm: Tính điểm số bài giải của học sinh
- Hàm Xem đáp án: Trình bày các bước giải của đáp án cho học sinh xem

d. Hệ thống lưu trữ

- Sơ đồ logic



- Bảng SACH_BAI_TAP: các thông tin về sách bài tập
- Bảng BAI_TAP: Các thông tin về các bài tập của sách
- Bảng BUOC_GIAI: Các thông tin về các bước giải trong một bài giải
- Bảng BAI_GIAI: Các thông tin về đáp án và các bài giải của một bài tập

Chương 4: THIẾT KẾ DỮ LIỆU

1. Tổng quan

Mục tiêu chính của thiết kế dữ liệu là mô tả cách thức tổ chức lưu trữ các dữ liệu của phần mềm. Có hai dạng lưu trữ chính mà người thiết kế cần phải cân nhắc và lựa chọn.

- Lưu trữ dưới dạng tập tin
- Lưu trữ dưới dạng cơ sở dữ liệu

Lưu trữ dưới dạng tập tin thường chỉ thích hợp với một số phần mềm đặc thù (cờ tướng, trò chơi, v.v.) đặc điểm chung của các phần mềm này là chú trọng rất nhiều vào xử lý, hình thức giao diện và không chú trọng nhiều đến việc lưu trữ lại các thông tin được tiếp nhận trong quá trình sử dụng phần mềm (thông thường các thông tin này được tiếp nhận và xử lý ngay).

Cách tiếp cận dùng cơ sở dữ liệu rất thông dụng và giáo trình này sẽ giới hạn trình bày chi tiết các phương pháp kỹ thuật liên quan đến việc tổ chức lưu trữ dữ liệu dùng cơ sở dữ liệu quan hệ. Giáo trình này sẽ không nhắc lại các khái niệm cơ bản về cơ sở dữ liệu và giả sử rằng người xem đã biết qua các khái niệm này. Tuy nhiên chúng ta cũng nên xem lại các bước để hình thành nên mô hình dữ liệu quan hệ trong quá trình thiết kế dữ liệu

2. Kết quả của thiết kế

Cách thức tổ chức lưu trữ dữ liệu của phần mềm được mô tả thông qua 2 loại thông tin sau:

- Thông tin tổng quát

  Cung cấp góc nhìn tổng quan về các thành phần lưu trữ
  - Danh sách các bảng dữ liệu: Việc lưu trữ cần sử dụng bao nhiêu bảng dữ liệu và đó là các bảng nào ?
  - Danh sách các liên kết: Các bảng dữ liệu có quan hệ, có mối liên kết giữa chúng ra sao?

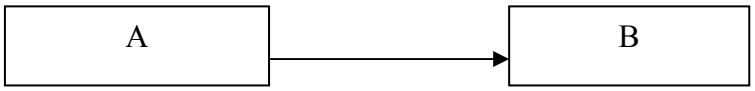
- Thông tin chi tiết:
  - Danh sách các thuộc tính của từng thành phần: Các thông tin cần lưu trữ của thành phần ?
  - Danh sách các Miền giá trị toàn vẹn: Các qui định về tính hợp lệ của các thông tin được lưu trữ

Có nhiều phương pháp, nhiều đề nghị khác nhau về việc mô tả các thông tin trên. Giáo trình này chọn sơ đồ logic để biểu diễn các thông tin tổng quát và bảng thuộc tính. Miền giá trị để mô tả chi tiết các thành phần trong sơ đồ logic.

Sơ đồ logic là sơ đồ cho phép thể hiện hệ thống các bảng dữ liệu cùng với quan hệ mối nối liên kết giữa chúng. Các ký hiệu được dùng trong sơ đồ rất đơn giản như sau:

Bảng: hình chủ nhật

Liên kết: (xác định duy nhất một): Mũi tên



Mũi tên hình trên có ngữ nghĩa: 1 phần tử A sẽ xác định duy nhất 1 phần tử B, ngược lại 1 phần tử B có thể tương ứng với nhiều phần tử A.

Ví dụ: Với phần mềm quản lý thư viện có sơ đồ logic sau:

Theo sơ đồ này việc lưu trữ các dữ liệu của phần mềm quản lý thư viện được tổ chức 3 bảng (DOCGIA, MUONSACH, SACH) vùng với 2 liên kết giữa chúng

Tất nhiên sơ đồ trên chỉ là một trong các cách thức tổ chức lưu trữ dữ liệu còn nhiều cách khác có thể có. Chi tiết các cách này sẽ được trình bày trong phương pháp thiết kế cơ sở dữ liệu.

Bảng thuộc tính cho phép mô tả chi tiết thành phần trong sơ đồ logic theo dạng như sau:

- Thành phần
- Ý nghĩa

STT	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1					
2					
…					

Bảng miền giá trị cho phép mô tả các Miền giá trị giữa các thuộc tính cùng một thành phần hay nhiều thành phần khác nhau.

MSố	Mô tả miền giá trị	Thành phần liên quan	Ghi chú
RB1			
RB2			
…			

Ví dụ:

Ghi chú:

- Bảng thuộc tính cho phép mô tả chi tiết thanh phần cần lưu trữ và sẽ được dùng trong báo cáo về thiết kế dữ liệu của phần mềm. Tuy nhiên cách mô tả trên khá dài dòng, trong giáo trình này sẽ sử dụng một dạng trình bày cô đọng hơn theo dạng lược đồ quan hệ. Với dạng trình bày này gồm tên bảng và thuộc tính đi kèm, các thuộc tính khóa được gạch chân.

Ví dụ:

DOC_GIA(MDG,Hoten,Loaidg,Ngsinh, Nglapthe, Diachi)

SACH(MSACH,Tensach,Theloai, NgNhap, Tacgia, Nhaxb, Namxb)

MUON(MDG,MSACH,NgMuon,Ngtra)

3. Quá trình thiết kế

Có 2 cách tiếp cận chính để thiết kế dữ liệu:

- **Phương pháp trực tiếp:**

Từ các yêu cầu đã xác định, tạo lập trực tiếp sơ đồ logic cùng với bảng thuộc tính, bảng miền giá trị. Các tiếp cận này rất khó thực hiện đối với sơ đồ logic phức tạp.

- **Phương pháp gián tiếp:**

Từ các yêu cầu đã xác định, tạo lập mô hình quan niệm dữ liệu, và sau đó đưa vào mô hình này sẽ tạo lập sơ đồ logic, bảng thuộc tính, bảng miền giá trị. Các tiếp cận này dễ thực hiện hơn vì mô hình quan niệm dữ liệu thường đơn giản (chứa các thành phần dữ liệu bản chất nhất của phần mềm). Khái niệm chi tiết về mô hình quan niệm dữ liệu cùng với các bước cụ thể sẽ được trình bày chi tiếp trong phần sau.

Tương ứng với 3 yêu cầu của phần mềm, quá trình thiết kế dữ liệu bao gồm 3 bước lớn:

- Thiết kế với tính đúng đắn
- Thiết kế với yêu cầu chất lượng
- Thiết kế với yêu cầu hệ thống

- Thiết kế với tính đúng đắn

- Đảm bảo đầy đủ và chính xác về mặt ngữ nghĩa các thông tin liên quan đến các công việc trong yêu cầu.
- Các thông tin phục vụ cho các yêu cầu chất lượng sẽ không được xét đến trong bước thiết kế này.

▪ Thiết kế với yêu cầu chất lượng

- Vẫn đảm bảo tínhh đúng đắn nhưng thỏa mãn thêm các yêu cầu chất lượng khác (tiến hóa, tốc độ nhanh, lưu trữ tối ưu).
- Cần chú ý bảo đảm tính đúng đắn khi cải tiến sơ đồ logic.

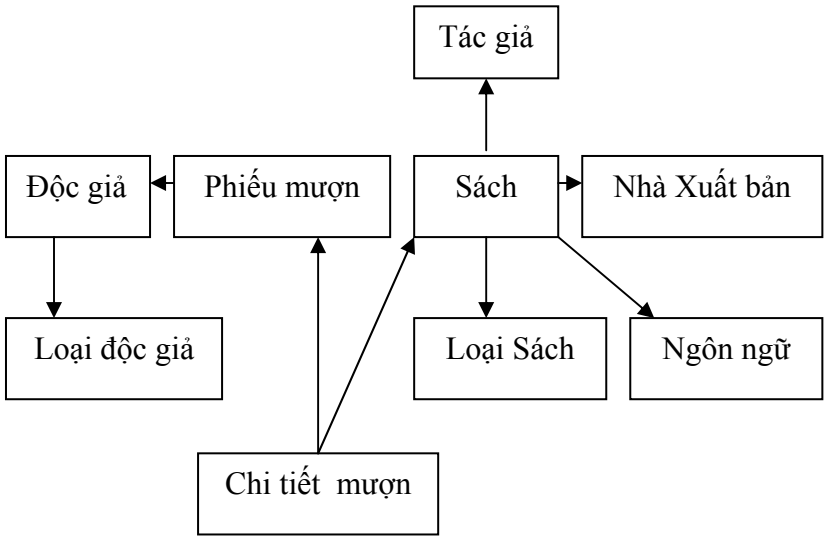
▪ Thiết kế với yêu cầu hệ thống

- Vẫn đảm bảo tính đúng đắn và các yêu cầu chất lượng khác nhưng thỏa mãn thêm các yêu cầu hệ thống (phân quyền, cấu hình phần cứng, môi trường phần mềm, v.v)

Ví dụ: phần mềm quản lý thư viện:

Với phương pháp trực tiếp sẽ cho kết qủa như sau:

Sơ đồ logic:



Các bảng thuộc tính:

DOC_GIA(MDG,MLDG,HoTen,NgaySinh,DiaChi,DienThoai)

SACH(MSACH,MTG,MNXB,MLSACH,MNN,TenSach, Ngàymua, SoTrang)

PHIEU_MUON(MPHM, NgayMuon)

CHITIETMUON(MPHM, MSACH, NgayTra)

LOAISACH(MLSACH,TenLS,GhiChu)

LOAIDOCGIA(MLDG,TenLDG,GhiChu)

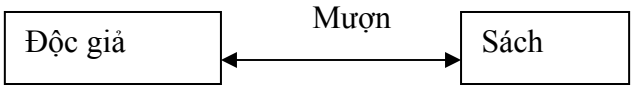
NHAXB(MNXB,TenNXB, GhiChu)

TACGIA(MTG,Ten, Ghichu)

NGONNGU(MNN,Ten,Ghichu)

Với phương pháp gián tiếp, ngoài kết qủa cuối cùng tương tự như phương pháp trực tiếp, còn có kết quả trung gian là mô hình quan niệm dữ liệu như sau:

+ Sơ đồ lớp đối tượng với 2 đối tượng chính Sách, Độc giả và 1 quan hệ Mượn giữa 2 lớp đối tượng trên

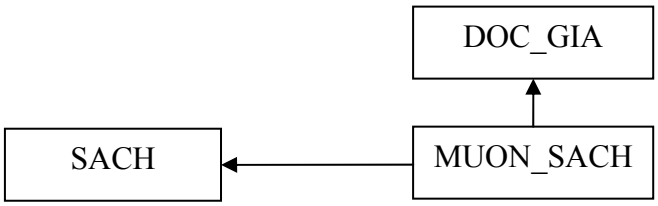


+ Mô hình chi tiết các thành phần trong sơ đồ lớp: **Xem chi tiết ở phụ lục B**

Ví dụ : Xét phần mềm với 4 yêu cầu: Lập thẻ độc giả, Nhận sách, Cho mượn sách, Trả sách

Thiết kế dữ liệu với tính đúng đắn

Sơ đồ logic



Chi tiết các bảng

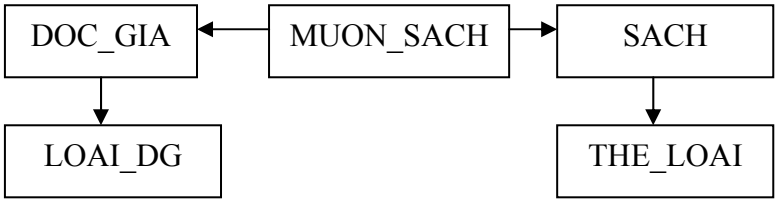
DOCGIA(MDG,MLDG,HoTen,NgaySinh,DiaChi,DienThoai)

SACH(MSACH,MTG,MNXB,MLSACH,MNN,TenSach, Ngàymua, SoTrang)

MUON_SACH(MDG,MSACH,NgMuon,NgTra,Tienphat)

Thiết kế dữ liệu với tính tiến hóa

Sơ đồ logic



Chi tiết các bảng:

DOC_GIA(MDG,MLDG,HoTen,NgaySinh,DiaChi,DienThoaiNg_lapthe,Ng_hethan)

SACH(MSACH,Tensach,MTL,ng_Nhap, Tacgia,NamXB, NhaXB)

MUON_SACH(MDG,MSACH,NgMuon,NgTra,Tienphat

THE_LOAI(MTL,Tentheloai,GhiChu)

LOAI_DG(MLDG,TenLDG,GhiChu)

Thiết kế với tính hiệu quả (truy xuất nhanh)

Sơ đồ logic

Cũng với sơ đồ logic như trên nhưng ta có các bảng thuộc tính:

DOC_GIA(MDG,MLDG,HoTen,NgaySinh,DiaChi,DienThoaiNg_lapthe,Ng_hethan, SosachMuon, TinhTrangtra)

SACH(MSACH,Tensach,MTL,ng_Nhap, Tacgia,NamXB, NhaXB, TinhTrangMuon)

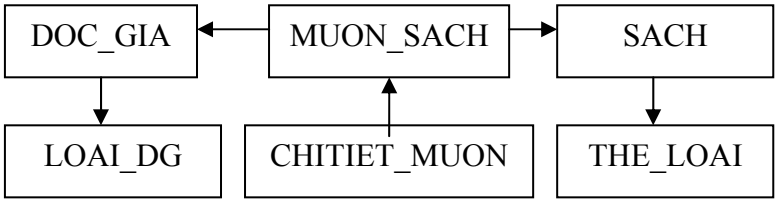
MUON_SACH(MDG,MSACH,NgMuon,NgTra,Tienphat)

THE_LOAI(MTL,Tentheloai,GhiChu)

LOAI_DG(MLDG,TenLDG,GhiChu)

Thiết kế dữ liệu với tính hiệu quả (lưu trữ tối ưu)

Sơ đồ logic



Chi tiết các bảng thuộc tính

DOC_GIA(MDG,MLDG,HoTen,NgaySinh,DiaChi,DienThoaiNg_lapthe,Ng_hethan, SosachMuon, TinhTrangtra)

SACH(MSACH,Tensach,MTL,ng_Nhap, Tacgia,NamXB, NhaXB, TinhTrangMuon)

MUON_SACH(MDG,MSACH,NgMuon,NgTra,Tienphat)

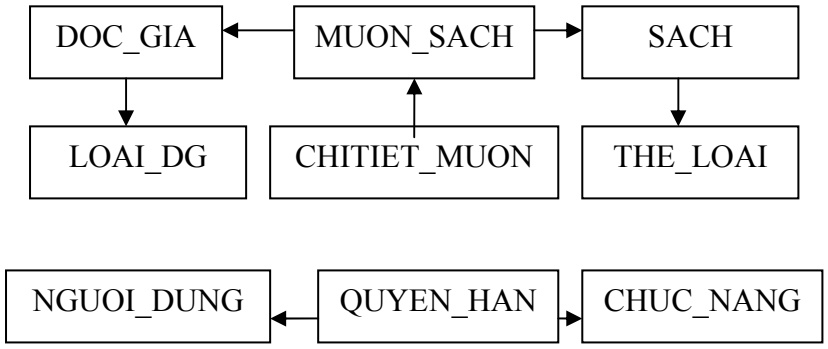
CHITIET_MUON(MMUON,MSACH,NgTra,Tienphat)

THE_LOAI(MTL,Tentheloai,GhiChu)

LOAI_DG(MLDG,TenLDG,GhiChu)

Thiết kế dữ liệu với yêu cầu phân quyền hệ thống (phân quyền)

Sơ đồ logic



Chi tiết các bảng

DOC_GIA(MDG,MLDG,HoTen,NgaySinh,DiaChi,DienThoaiNg_lapthe,Ng_hethan, SosachMuon, TinhTrangtra)

SACH(MSACH,Tensach,MTL,ng_Nhap, Tacgia,NamXB, NhaXB, TinhTrangMuon)

MUON_SACH(MDG,MSACH,NgMuon,NgTra,Tienphat)

CHITIET_MUON(MMUON,MSACH,NgTra,Tienphat)

THE_LOAI(MTL,Tentheloai,GhiChu)

LOAI_DG(MLDG,TenLDG,GhiChu)

NGUOI_DUNG(MND,HoTen, Ghichu)

CHUC_NANG(MCN,Ten_Chucnang, Ghichu)

QUYEN_HAN(MND,MCN)

4. Phương pháp thiết kế dữ liệu

4.1 Phương pháp trực tiếp

Bước 1:

- Lập sơ đồ với 1 thành phần duy nhất
- Đánh giá tính đúng đắn so với các yêu cầu và chuyển sang bước 2 nếu cần thiết

Bước 2:

- Tách 1 số thuộc tính để tạo ra các thành phần mới

- Xác định liên kết giữa các thành phần
- Đánh giá tính đúng đắn so với các yêu cầu và lặp lại bước 2 nếu cần thiết

Ví dụ: phần mềm quản lý thư viện

Cách 1: Chỉ dùng 1 thành phần SÁCH

Masach, Ten, Theloai, Ngaymua, Tacgia, NhaXB, NamXB

HotenDG, LoaiDG, Ngaylamthe, Ngaymuon, Ngaytra

Cách 2: Dùng 2 thành phần SACH,DOCGIA

Cách 2.1 : Chỉ lưu trừ lần mượn sách cuối cùng

SACH

MSACH, MADG, Ten, Theloai, NgayMua, TacGia, NhaXB, NamXB, Ngaymuon, NgayTra.

DOCGIA

MDG, HoTen, LoaiDG,Ngaylamthe,

Cách 2.2: Chỉ cho phép độc giả mượn tối đa 1 quyển sách

SACH

MSACH, Ten, Theloai, NgayMua, TacGia, NhaXB, NamXB, Ngaymuon, NgayTra.

DOCGIA

MDG, MSACH,HoTen, LoaiDG,Ngaylamthe, Ngaymuon

Cách 3: Dùng 3 thành phần SACH,DOCGIA, MUONSACH

SACH

MSACH, Ten, Theloai, NgayMua, TacGia, NhaXB, NamXB, Ngaymuon, NgayTra.

DOCGIA

MDG, HoTen, LoaiDG,Ngaylamthe,

MUONSACH

Mmuon,MDG,MSACH, Ngaymuon, Ngaytra

Ví dụ: Phần mềm quản lý học sinh

Cách 1: Dùng 1 thành phần HOCSINH

HOCSINH

MAHS, HoTen, Ngaysinh, GioiTinh, Lop, Monhoc, LoaiKT, HocKy,Diem, Ngayvang, Lydo

Cách 2: Dùng 3 bảng HOCSINH, KIEMTRA, DIEMDANH

HOCSINH

MAHS, Hoten, Ngaysinh, GioiTinh, Lop

KIEMTRA

MAKT,MAHS, Monhoc,LoaiKT,Hocky, Diem

DIEMDANH

MADD,MAHS,Ngayvang, Lydo

4.2 Phương pháp gián tiếp

Bước 1:

- Lập sơ đồ lớp
- Xác định các lớp đối tượng
- Xác định quan hệ giữa các lớp đối tượng và lập sơ đồ

Bước 2:

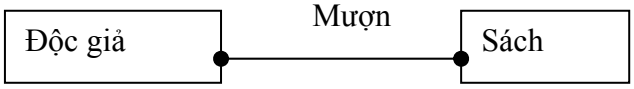
- Ánh xạ từ sơ đồ lớp vào sơ đồ logic
- Ánh xạ các lớp đối tượng
- Ánh xạ các quan hệ giữa các lớp đối tượng

Bước 3:

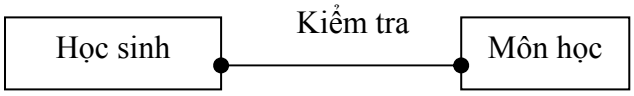
- Hoàn chỉnh sơ đồ logic
- Bổ sung các thành phần theo yêu cầu
- Mô tả chi tiết các thuộc tính của các thành phần

4.2.1 Lập sơ đồ lớp

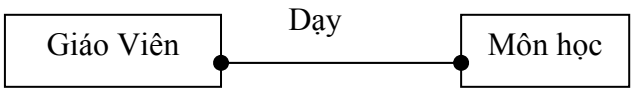
Ví dụ: Với phần mềm quản lý thư viện 2 đối tượng chính là Độc giả, Sách và quan hệ giữa chúng là quan hệ mượn sách



Với phần mềm quản lý học sinh trường phổ thông trung học 2 đối tượng chính là Học sinh, Môn học và quan hệ giữa chúng là quan hệ kiểm tra



Với phần mềm xếp thời khóa biểu trường trung học phổ thông 2 đối tượng chính là Giáo Viên, Môn học và quan hệ giữa chúng là quan hệ dạy.



4.2.2 Ánh xạ sơ đồ lớp

Ánh xạ lớp đối tượng. Mỗi đối tượng trong sơ đồ lớp tương ứng với 1 thành phần trong sơ đồ logic

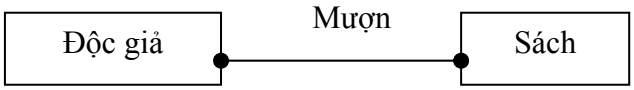
Sơ đồ lớp:

Sơ đồ logic:

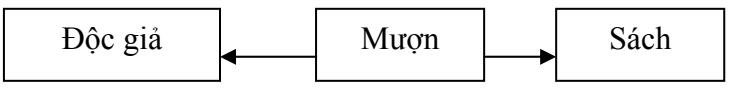
4.2.3 Ánh xạ quan hệ

- Quan hệ 1-n: Quan hệ 1-n trong sơ đồ lớp giữa 2 lớp đối tượng A,B (1 A nhiều B) tương ứng với liên kết xácđịnh duy nhất vừ A sang B trong sơ đồ logic.
- Quan hệ m-n: Quan hệ m-n C trong sơ đồ lớp giữa 2 lớp đối tượng A,B tương ứng với 1 thành phần C trong sơ đồ logic. Thành phần này có liên hệ xác định duy nhất A,B.

Sơ đồ lớp:



Sơ đồ logic:

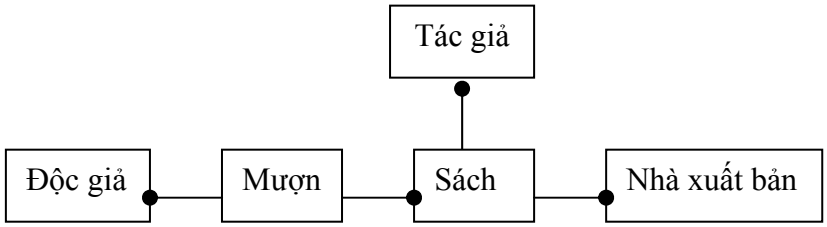


4.2.4 Hoàn chỉnh sơ đồ logic

1. Bổ sung các thành phần

+ Đối tượng phụ: Mỗi đối tượng phụ tương ứng với 1 thành phần trong sơ đồ logic

+ Các thành phần khác: Xem xét lại tính đúng đắn và bổ sung thêm nếu cần thiết



2. Mô tả chi tiết thuộc tính các thành phần

+ Thuộc tính khóa chính:

- Mỗi thành phần ứng với đối tượng (chính, phụ) cần 1 thuộc tính khóa riêng)

- Các thành phần còn lại, tùy theo ý nghĩa sử dụng sẽ có thuộc tính khóa riêng hay dùng tổ hợp thuộc tính khóa của các thành phần khác

Ví dụ: Các thành phần Độcgiả, Sách, Nhà xuất bản, Tác giả sẽ có thuộc tính khóa chính tương ứng là MDG, MSACH, MNXB, MTG.

Thành phần mượn cũng sẽ có khóa chính là MMUON (không dùng tổ hợp các thuộc tính khóa ngoại được ?)

+ Thuộc tính khóa ngoại:

- Thể hiện đúng liên kết giữa các thành phần trong sơ đồ logic: nếu A xác định duy nhất B thì A có thuộc tính là khoá chính của B (đó là khóa ngoại của A)

Ví dụ:

Thành phần Mượn có 2 khóa ngoại: MDG,MSACH

Thành phần Sách có 2 khoá ngoại: MNXB, MTG, MDG

+ Các thuộc tính khác:

Dựa vào yêu cầu lưu trữ, chú ý các loại thuộc tính sau:

- Định danh: Tên
- Loại: Sự phân loại
- Thời gian: Ngày tháng
- Không gian: vị trí
- Định lượng: độ đo, tính chất, v.v.v

Ví dụ: Độc giả sẽ có thuộc tính khác như:

HoTen (định danh)

LoaiDG (loại)

Ngaysinh (thời gian)

Ngayhethan (thời gian)

Diachi (không gian)

Sách sẽ có thuộc tính khác như:

TenSach (định danh)

LoaiSach (loại)

NgayMua (thời gian)

GiaTien (định lượng)

5. Thiết kế dữ liệu với tính đúng đắn

- Các bước thực hiện:

Bước 1: Chọn một yêu cầu và xác định sơ đồ logic cho yêu cầu đó

Bước 2: Bổ sung thêm một yêu cầu và xem xét lại sơ đồ logic

+ Nếu sơ đồ logic vẫn đáp ứng được thì tiếp tục bước 3

+ Nếu sơ đồ logic không đáp ứng được thì bổ sung vào sơ đồ thuộc tính mới (ưu tiên 1) hoặc thành phần mới (ưu tiên 2) cùng với các thuộc tính và liên kết tương ứng

Bước 3: Quay lại bước 2 cho đến khi đã xem xét đầy đủ các yêu cầu

Ghi chú:

- Với mỗi yêu cầu cần xác định rõ cần lưu trữ các thông tin gì? dựa vào luồng dữ liệu đọc/ghi trong sơ đồ luồng dữ liệu tương ứng) và tìm cách bổ sung các thuộc tính để lưu trữ các thông tin này
- Chỉ xem xét tính đúng đắn
- Cần chọn các yêu cầu theo thứ tự từ đơn giản đến phức tạp (thông thường yêu cầu tra cứu là đơn giản nhất)
- Với yêu cầu phức tạp có thể phải bổ sung vào sơ đồ logic nhiều thành phần mới

Khóa của các thành phần phải dựa trên ngữ nghĩa tương ứng trong thế giới thực

6. Thiết kế dữ liệu và yêu cầu chất lượng

- **Mục tiêu**

Xem xét đánh giá sơ đồ logic theo các yêu cầu về chất lượng và tiến hành cập nhật lại sơ đồ để bảo đảm các tiêu chuẩn về chất lượng. Ngoài tính đúng đắn cần ưu tiên hàng đầu xem xét sự hơn kém nhau giữa các phần mềm chính là mức độ thỏa mãn các tiêu chuẩn chất lượng còn lại (đặc biệt là tính tiến hóa).

6.1 Xem xét tính tiến hóa

Để bảo đảm tính tiến hóa, sơ đồ logic sẽ còn bổ sung cập nhật lại nhiều thành phần qua các bước thiết kế chi tiết. Trong các bước đầu tiên là thiết kế dữ liệu, chúng sẽ giới hạn xem xét đến các thuộc tính có giá trị rời rạc.

Thuộc tính có giá trị rời rạc là các thuộc tính mà miền giá trị chỉ bao gồm một số giá trị nhất định. Các giá trị này thông thường thuộc về tập hợp có độ biến động rất ít trong quá trình sử dụng phần mềm.

Ví dụ:

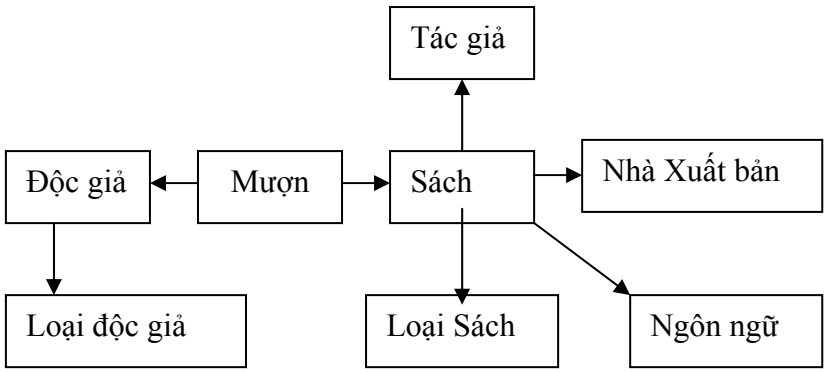
LOAIDG (thành phần độc giả): Thư viện hiện tại chỉ có 3 loại độc giả là ‘A’, ‘B’,’C’ và khả năng có thêm loại độc giả mới rất thấp.

Ngôn ngữ (thành phần Sách): Các sách trong thư viện hiện tại chỉ có 3 loại ngôn ngữ ‘Việt’, ‘Anh’, ‘Pháp’ và khả năng thêm sách thuộc ngôn ngữ mới rất thấp.

Tuy nhiên cần lưu ý rằng khả năng biến động trên tập hợp giá trị của thuộc tính rời rạc là thấp nhưng không phải là không có. Và khi xảy ra biến động (thêm loại độc giả, thêm sách thuộc ngôn ngữ mới) nếu không chuẩn bị trước trong thiết kế thì người dùng sẽ không thể khai báo được các biến động này với phần mềm, và do đó có thể một số chức năng sẽ không thực hiện được (ví dụ khong thể thêm sách mới với ngôn ngữ tiếng Hoa).

Để chuẩn bị tốt cho biến động về sau (nếu có) trong tập hợp các giá trị của thuộc tính rời rạc. Chúng ta sẽ tách các thuộc tính này thành một thành phần trong sơ đồ logic. Khi đó người dùng trong quá trình sử dụng hoàn toàn có thể cập nhật lại tập hợp các giá trị này tương ứng với các biến động thực tế trong thế giới thực.

Sơ đồ logic khi tách các thuộc tính rời rạc như sau:



6.2 Xem xét tính hiệu quả (tốc độ)

- Phạm vi xem xét:

  - Chỉ giới hạn xem xét việc tăng tốc độ thực hiện của phần mềm bằng cách bổ sung thêm các thuộc tính vào các bảng dùng lưu trữ các thông tin đã tính toán trước (theo qui tắc nào đó từ các thông tin gốc đã được lưu trữ)

    Ví dụ: số sách đang mượn của độc giả

  - Các thông tin này phải được tự động cập nhật khi có bất kỳ thay đổi thông tin gốc liên quan

    Ví dụ độc giả mượn thêm hoặc trả sách

    Học sinh có thêm cột điểm

- Các bước tiến hành:

- Bước 1: Chọn một yêu cầu và xem xét cần bổ sung thông tin gì trên bộ nhớ phụ để tăng tốc độ thực hiện của xử lý liên quan (các thông tin xử lý phải đọc mà không cần thực hiện việc tính toán)

- Bước 2: Quay lại bước 1 cho đến khi đã xem xét đầy đủ các yêu cầu

Ghi chú:

- Sau mỗi bước nhất thiết phải lập bảng danh sách các thuộc tính tính toán cùng với thông tin liên quan

  + Thông tin gốc

  + Xử lý tự động cập nhật thông tin gốc (chi tiết về các xử lý này sẽ được mô tả trong phần thiết kế xử lý)

- Nếu thông tin gốc thường xuyên bị thay đổi, việc bổ sung thuộc tính tính toán để tăng tốc độ thực hiện sẽ mất ý nghĩa (thậm chí theo chiều ngược lại)

- Việc tăng tốc độ truy xuất có thể sẽ dẫn đến việc lưu trữ không tối ưu
- Thứ tự xem xét các yêu cầu theo thứ tự từ đầu đến cuối (không cần chọn như các bước trong thiết kế dữ liệu)

  Ví dụ: Phần mềm quản lý giải vô địhc bóng đá quốc gia với bảng thuộc tính tính toán

  Thuộc tính: Tong_ban_thang, Tong_the_phat, Diem_so là những thuộc tính có thể xử lý tự động cập nhật

6.3 Xem xét tính hiệu quả (lưu trữ)

Tính hiệu quả trong thiết kế dữ liệu sẽ được xem xét dưới góc độ lưu trữ tối ưu. Vấn đề đặt ra là xây dựng sơ đồ logic sao cho vẫn bảo đảm lưu trữ đầy đủ thông tin theo yêu cầu nhưng với dung lượng lưu trữ nhỏ nhất có thể có. Vấn đề này đặc biệt quan trọng với các phần mềm với hệ thống lưu trữ lớn và nhiều phát sinh thông tin cần lưu trữ theo thời gian.

Khi đó cần đặc biệt quan tâm đến các thành phần mà dữ liệu tương ứng được phát sinh nhiều theo thời gian. Chúng ta sẽ tìm cách bố trí lại sơ đồ logic sao cho vẫn đảm bảo thông tin mà dung lượng lưu trữ lại ít hơn.

- Các bước tiến hành:

Bước 1:Lập danh sách các bảng cần được xem xét để tối ưu hóa việc lưu trữ

- Xem xét và xác định các công việc có tần suất thực hiện thường xuyên và bổ sung vào danh sách chọn các bảng được sử dụng tương ứng của công việc này
- Xem xét các bảng mà khóa của bảng bao gồm nhiều thuộc tính và bổ sung bảng này vào danh sách được chọn

Bước 2: Tối ưu hóa việc lưu trữ các bảng có khối lượng dữ liệu lưu trữ lớn thông qua việc tối ưu hóa lưu trữ từng thuộc tính trong bảng

- Xác định các thuộc tính mà việc lưu trữ chưa tối ưu. Ưu tiên xem xét các thuộc tính có kiểu chuỗi
- Tối ưu hóa việc lưu trữ tùy theo từng trường hợp cụ thể
- Một trong các trường hợp thông dụng nhất là chuỗi có kích thước lớn và giá trị được sử dụng nhiều lần trong các mẫu tin khác nhau (ví dụ: thuộc tính tác giả, Nha_xb trong bảng SACH của phần mềm quản lý sách)

- Với trường hợp trên việc tối ưu hoá có thể thực hiện thông qua việc bổ sung các bảng mới (bảng TAC_GIA, NHA_XB) và tổ chức cấu trúc bảng SACH (thay thuộc tính TAC_GIA bằng MTG, thay thuộc tính NHA_XB bằng MNXB)

Bước 3: Tối ưu hóa các bảng mà khóa của bảng bao gồm nhiều thuộc tính.

Phân rã bảng đang xét thành hai bảng. Trong đó, một bảng chứa các thuộc tính mà giá trị được lặp lại nhiều lần trong cùng một lần thực hiện công việc tương ứng trong thế giới thực. Bảng này cần có khóa riêng (sẽ được bảng còn lại sử dụng để tham chiếu đến)

Ghi chú:

- Việc phân rã giúp cho lưu trữ tối ưu tuy nhiên:
  - Tốc độ truy xuất có thể sẽ chậm hơn
  - Việc thực hiện xử lý khó khăn hơn (thuật giải phức tạp hơn)
- Cần cân nhắc trước khi thực hiện phân rã
- Việc đánh giá khóa riêng cho bảng đã phân ra có thể cần kiểm tra thêm số phụ thuộc hàm

Ví dụ minh họa: Phần mềm quản lý bán sách

Bước 1: Các bảng cần xem xét

NHAP_SACH(MSACH, Ng_Nhap, So_luong, Don_gia, Thanh_tien)

HOA_DON(MHD,MSACH, Khach_hang, Ng_lap_hd, So_Luong, Don_gia, Thanh_tien)

Bước 2:

- Bổ sung bảng KHACH_HANG

  KHACH_HANG(MKH,Ho_ten,Ghi_chu)

- Tổ chức lại bảng HOA_DON

  HOA_DON (MHD,MSACH,MKH,Ng_lap_hd,So_luong, Don_gia, Thanh_tien)

Bước 3:

- Phân rã bảng NHAP_SACH thành 2 bảng NHAP_SACH, CT_NHAP

  NHAP_SACH(MNHAP,Ng_Nhap)

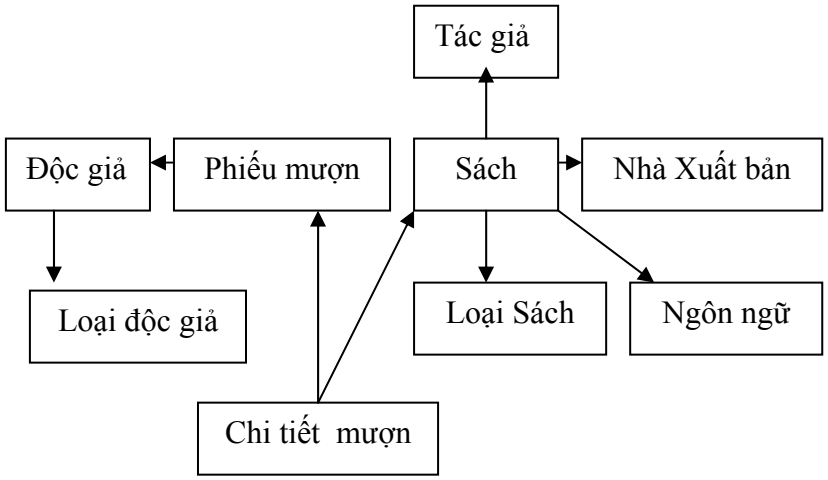
  CT_NHAP(MNHAP,MSACH,So_luong, Don_gia, Thanh_tien)

- Phân rã bảng HOA_DON thành 2 bảng HOA_DON, CT_HOA_DON

HOA_DON(MHD,MKH, Ng_lap_hd)

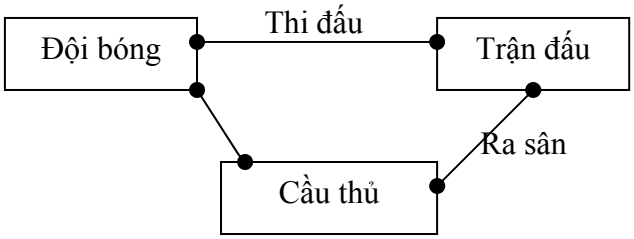
CT_HD(MHD, MSACH,So_luong, Don_gia, Thanh_tien)

Ví dụ: Xét phần mềm quản lý thư viện.



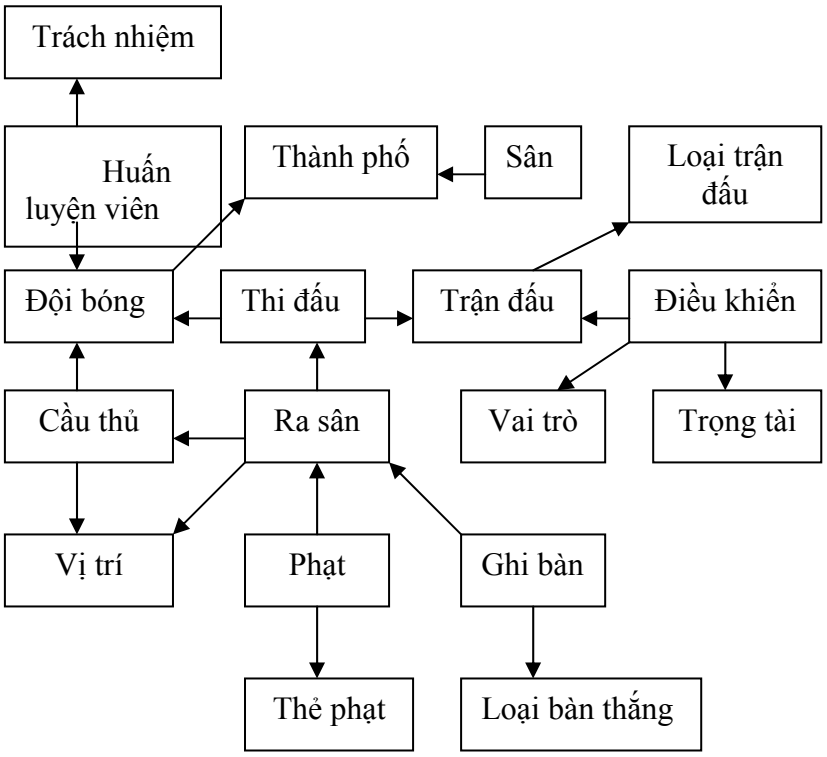
Ví dụ: Xét phần mềm quản lý giải bóng đá

Sơ đồ lớp



Mô tả chi tiết các thuộc tính: **Xem chi tiết phụ lục B**

- **Sơ đồ logic**



- **Mô tả chi tiết thuộc tính:** Xem chi tiết phụ lục B

Chương 5 : THIẾT KẾ GIAO DIỆN

1. Tổng quan

Chương này đi sâu tìm hiểu cách thức thiết kế giao diện là công đoạn không kém phần quan trọng trong quá trình làm phần mềm, đây cũng có thể xem là công đoạn phác thảo đồ hình hay prototype cho phần mềm và để sau đó nhận phản hồi yêu cầu của khách hàng đối với chương trình và để người thiết kế có thể điều chỉnh theo yêu cầu đề ra. Tùy theo mục đích yêu cầu, theo độ phức tạp của chương trình, người thiết kế giao diện có thể làm theo các những thiết kế sau và kết qủa thiết kế tương ứng. Thiết kế giao diện phải nắm bắt các điều chính yếu sau:

1. Hồ sơ cá nhân người dùng: Biết họ là ai, mục đích của người dùng là gì, Kỹ năng và kinh nghiệm của người dùng, nhu cầu của họ
2. Mượn những ứng xử từ những hệ thống quen thuộc đối với người dùng.
3. Cho người dùng thấy rõ các chức năng một cách sẵn sàng
4. ứng xử của chương trình từ trong ra ngoài phải kết dính, gắn kết
5. Thay đổi trong ứng xử nên tương phản với diện mạo của chương trình
6. Shortcut: Cung cấp cả cách thức cụ thể và tóm tắt tác vụ được làm
7. Tính tập trung: Một số giao diện GUI được tập trung chú ý nhiều hơn
8. Ngữ pháp: thông qua giao diện biết số luật thao tác
9. Trợ giúp, độ an toàn, hạn chế ngữ cảnh người dùng, giao diện đẹp,…

1.1 Kết quả thiết kế

- Màn hình giao diện:

  Màn hình giao diện là một trong các hình thức giao tiếp giữa người sử dụng và phần mềm khi họ thực hiện các công việc của mình trên máy tính. Mục tiêu chính của thiết kế giao diện là mô tả hệ thống các màn hình giao diện này

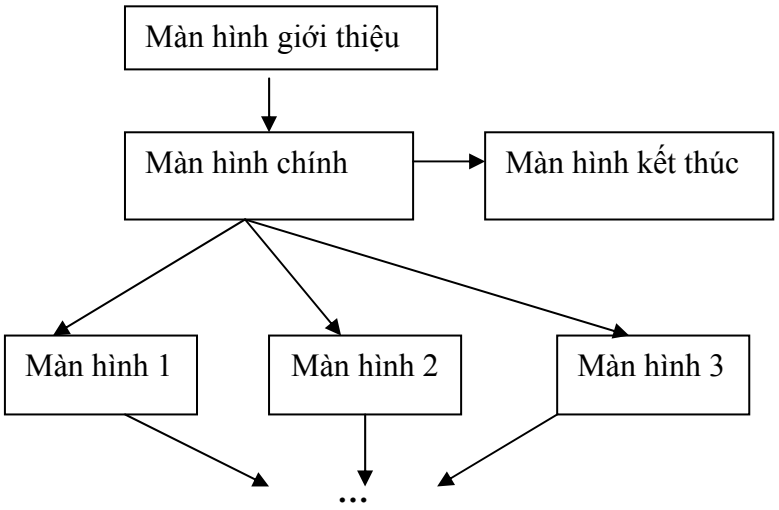
- Kết quả thiết kế giao diện: bao gồm 2 phần
  - Sơ đồ màn hình: Mô tả các thông tin tổng quát về hệ thống các màn hình cùng với quan hệ về việc chuyển điều khiển giữa chúng
  - Mô tả chi tiết từng màn hình: Mô tả chi tiết nội dung, hình thức trình bày và các thao tác mà người dùng có thể thực hiện trên từng màn hình

Ví dụ: Liệt kê các phần sau: màn hình, ý nghĩa sử dụng

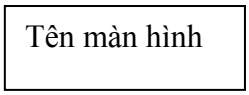
Danh sách các thao tác có thể thực hiện

STT	Thao tác	Ý nghĩa	Xử lý liên quan	Ghi chú
1				
2				

- Sơ đồ màn hình



Ký hiệu:



Màn hình với tên tương ứng

Chuyển điều khiển đến màn hình khác

- Mô tả màn hình giao diện:

  Các thông tin cần mô tả môt màn hình giao diện bao gồm

  o Tên màn hình:

  Tên của công việc tương ứng muốn thực hiện trên máy tính.

  o Nội dung:

  Cấu trúc các thành phần bên trong màn hình. Các thành phần này có thể chia làm 2 loại: Thanh phần dữ liệu, thành phần xử lý.

  + Thành phần dữ liệu

  Các thông tin liên quan đến công việc đang xét như sau:

  a. Thông tin nhập liệu:loại thông tin người dùng chịu trách nhiệm cung cấp giá trị (ngày lập, hóa đơn, hàng bán,..) có thể là nhập liệu trực tiếp, nhập liệu với giá trị định sẵn(có thể sửa nếu muốn) hoặc chọn trong danh sách có trước.

b. Thông tin kết xuất: loại thông tin này phần mềm chịu trách nhiệm cung cấp giá trị (ví dụ lượng hàn tồn hiện nay, tổng tiền trả,…).

+Thành phần xử lý:

Các nút điều khiển cho phép người dùng yêu cầu phần mềm thực hiện một xử lý nào đó.

+ Hình thức trình bày:

Cách thức bố trí sắp xếp các thành phần trong màn hình (ví trí, màu sắc, kích thước,…)

Với màn hình có biểu mẫu liên quan, tốt nhất là trình bày đúng với biểu mẫu tương ứng hoặc trình bày đúng như yêu cầu của khách hàng. Tuy nhiên cần lưu ý trong trường hợp biểu mẫu liên quan chỉ là kết quả cuối cùng cần ghi nhận trước khi đạt đến kết quả đó cần phải thực hiện một số công việc trung gian không có biểu mẫu rõ ràng. Với những trường hợp này cần bổ sung, sáng tạo hình thức trình bày các màn hình trung gian thể hiện các công việc trung gian.

Với màn hình không có biểu mẫu liên quan hình thức trình bày màn hình hoàn toàn là sự sáng tạo khi thiết kế.

+ Các thao tác có thể thực hiện

Mô tả hệ thống các thao tác mà người dùng có thể thực hiện trên màn hình cùng với ý nghĩa của chúng. Có rất nhiều loại thao tác khác nhau có thể cung cấp cho người dùng trên một màn hình giao diện, tuy nhiên giáo trình này chỉ giới hạn xem xét việc mô tả thao tác khi người dùng nhấn vào nút điều khiển hay nút lệnh hoặc kết thúc việc nhập liệu tại một thành phần nhập liệu nào đó.

1.2 Phân loại màn hình giao diện

Quá trình sử dụng phần mềm bao gồm các bước sau:

- Chọn công việc muốn thực hiện trên máy tính.
- Cung cấp các thông tin cần thiết tương ứng với công việc đã chọn.
- Yêu cầu phần mềm thực hiện.
- Xem xét kết quả thực hiện.

Dựa trên quá trình trên các màn hình giao diện có thể được chia thành nhiều loại tùy theo ý nghĩa sử dụng.

- **Màn hình chính**: Cho phép người dùng sử dụng chọn lựa công việc mong muốn thực hiện trên máy tính từ danh sách các công việc

- **Màn hình nhập liệu** lưu trữ: Cho phép người dùng thực hiện lưu trữ các thông tin được phát sinh trong thế giới thực.
- **Màn hình nhập liệu** xử lý: Cho phép người sử dụng cung cấp các thông tin cần thiết cho việc thực hiện một công việc nào đó
- **Màn hình kết quả**: Trình bày cho người sử dụng các kết quả việc thực hiện của một công việc nào đó
- **Màn hình thông báo**: Thông báo, nhắc nhở người sử dụng trong quá trình thực hiện một công việc nào đó
- **Màn hình tra cứu**: Cho phép tìm kiếm thông tin đã được lưu trữ với các tiêu chuẩn tìm kiếm

Một màn hình giao diện có thể thuộc một trong các loại trên hay cũng có thể tích hợp từ nhiều màn hình cơ sở thuộc vào các loại trên tùy theo bản chất công việc liên quan.

Trong thực tế còn có rất nhiều màn hình khác, tuy nhiên giáo trình chỉ giới hạn xem xét chủ yếu đến các loại màn hình đã trình bày phía trên, giáo trình sẽ chú trọng trình bày chi tiết 3 loại màn hình quan trọng và thông dụng nhất: màn hình chính, màn hình tra cứu, màn hình nhập liệu lưu trữ.

1.3 Quá trình thiết kế

Qui trình chung: Dựa trên yêu cầu chức năng, đầu tiên người thiết kế sẽ xem xét thiết kế các giao diện từ tính đúng đắn, đến tính tiện dụng thõa yêu cầu về tiện dụng, và xét đến tính hiệu quả nếu yêu cầu về hiệu quả được đưa ra và một số thiết kế theo yêu cầu khác, v.v

❖ **Thiết kế giao diện với tính đúng đắn**

- **Sơ đồ màn hình**

Giả sử cần thực hiện n công việc trên máy tính, sơ đồ màn hình trong trường hợp này chỉ bao gồm n+1 màn hình sau:

+ Một màn hình chính cho phép chọn công việc

+ n màn hình liên quan trực tiếp đén n công việc muốn thực hiện

- **Mô tả chi tiết từng màn hình**

1. Màn hình chính:

Xác định chính xác nội dung dựa trên danh sách các công việc được yêu cầu và chọn hình thức trình bày đơn giản nhất.

Ví dụ 1: Phần mềm thự viện

Màn hình chính	
1. Cho mượn sách	8. Lập báo cáo về độc giả
2. Nhận trả sách	9.Nhận sách mới
3.Tìm sách	10.Thanh lý sách
4.Lập báo cáo mượn trả	11.Lập báo cáo sách
5.Lập thẻ độc giả	12.Thay đổi qui đinh tổ chức
6.Gian hạn thẻ độc giả	13.Thay đổi quy định mượn trả
7.Tìm độc giả	14. Thoát

Đây là thiết kế cho ứng dụng chạy độc lập có thể hiển thị tất cả danh sách các màn hình, còn đối với ứng dụng lập trình mạng web có thể tùy theo quyền hạn sử dụng màn hình chính hạn chế bởi các màn hình tương tác cho người sử dụng đó.

Ví dụ 2: Phần mềm quản lý học sinh cấp 3

Màn hình chính	
1. Tiếp nhận hồ sơ	8. Thay đổi qui đinh tổ chức
2. Xếp lớp	9. Thay đổi qui đinh xếp loại
3.Tìm học sinh	10.Thoát
4. Nhận bảng điểm danh	
5.Nhận bảng điểm kiểm tra	
6.Xếp loại học sinh	
7.Lập báo cáo tổng kết	

Ví dụ 3: Phần mềm quản lý giải bóng đá

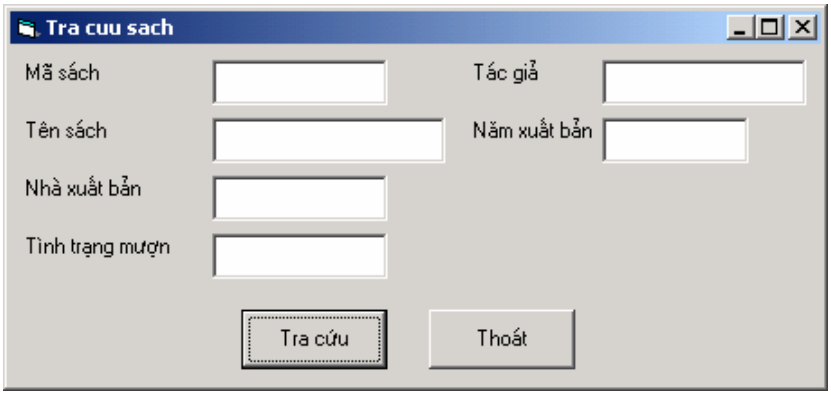
Màn hình chính

1. Tiếp nhận hồ sơ đội bóng
2. Xếp lịch thi đấu
3. Phân công trọng tài
4. Ghi nhận kết quả thi đấu
5. Lập bảng xếp hạng tạm thời
6. Tra cứu cầu thủ
7. Lập báo cáo tổng kết giải
8. Thay đổi qui đinh tổ chức
9. Thay đổi qui đinh xếp hạng
10. Thoát

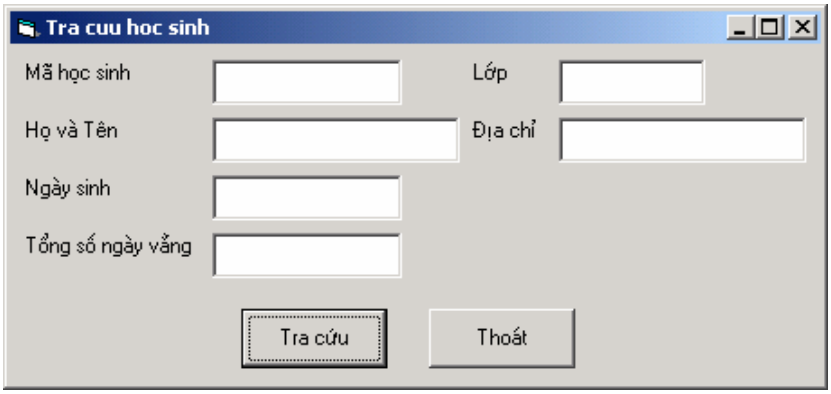
2. Màn hình tra cứu

Chọn tiêu chuẩn tra cứu đơn giản nhất (chỉ có mã số) và kết quả tìm kiếm đơn giản (cho biết có hay không có mã số trên).

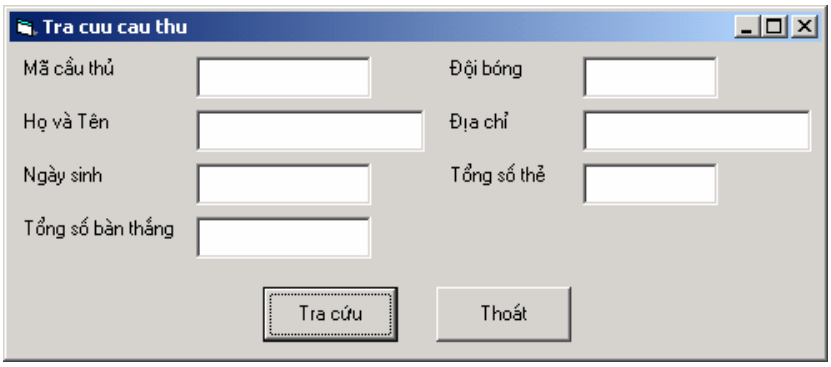
Ví dụ 1: Tra cứu sách với phần mềm quản lý thư viện



Ví dụ 2: Tra cứu học sinh với phần mềm quản lý học sinh



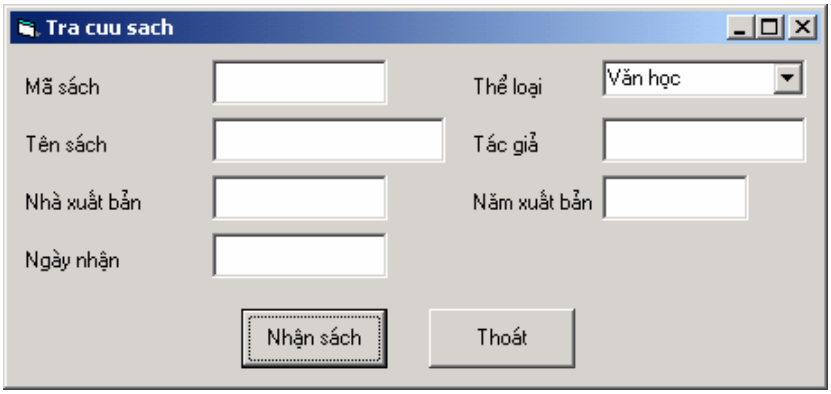
Ví dụ 3: Tra cứu cầu thủ với phần mềm quản lý giải bóng đá

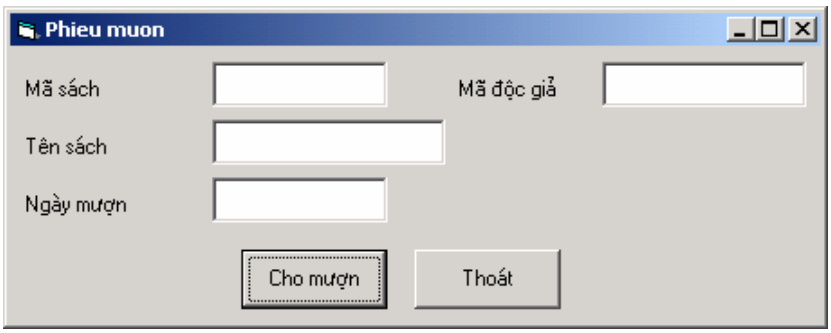


3. Màn hình nhập liệu

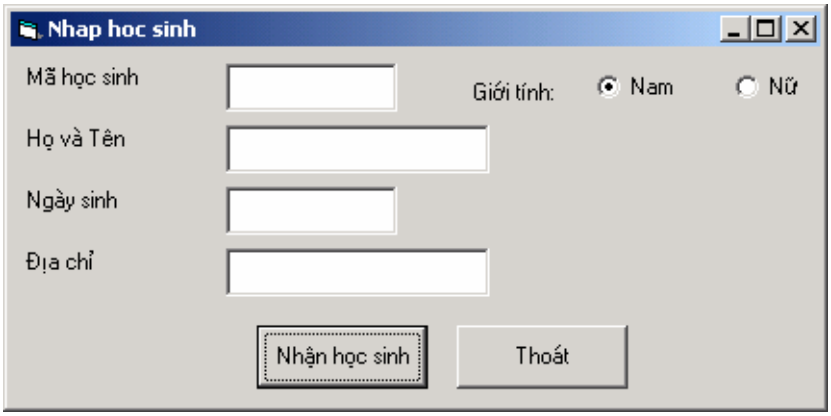
Xác định chính xác nội dung dựa trên biểu mẫu hoặc thông tin liên quan đến công việc tương ứng và chọn hình thức trình bày đơn giản nhất có thể có (liệt kê tuần tự các nội dung)

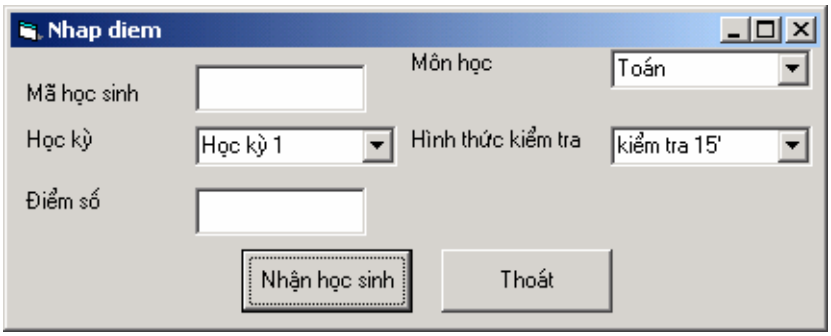
Ví dụ: Nhập sách, mượn sách với quản lý thư viện



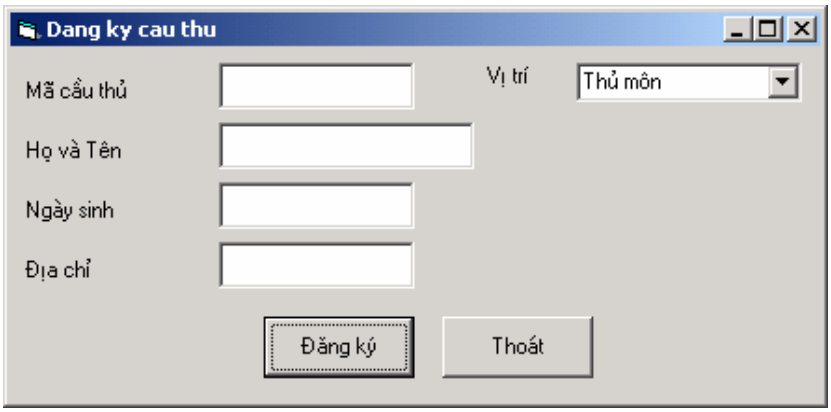


Ví dụ: Nhập học sinh, điểm số với phần mềm quản lý học sinh





Ví dụ: Đăng ký cầu thủ với phần mềm quản lý giải bóng đá



❖ **Thiết kế giao diện với tính tiện dụng**

- **Sơ đồ màn hình**

Bổ sung vào sơ đồ các màn hình công việc trung gian giúp cho việc sử dụng các màn hình công việc chính dễ dàng hơn, tự nhiên hơn.

- **Mô tả chi tiết từng màn hình**

1. Màn hình chính

Phân chia các công việc theo từng nhóm tùy theo ý nghĩa và chọn hình thức trình bày tự nhiên nhất có thể có (menu, sơ đồ,…)

Ví dụ 1: Màn hình chính phần mềm quản lý giải bóng đá. Có các nút điều khiển sau:

Tổ chức: Sân, Trọng tài, Loại thẻ phạt, Loại bàn thắng, Qui chế tổ chức.

Kế hoạch: Đăng ký đội bóng, Xếp lịch thi đấu, Phân trọng tài

Thi đấu: Ghi nhận kết quả, tra cứu cầu thủ, xếp loại tạm thời

Tổng kết: Xếp hạng chính thức, Lập báo cáo tổng kết

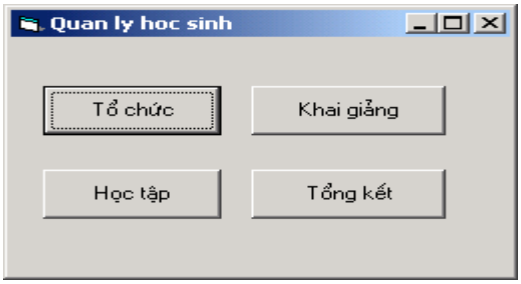
Ví dụ 2: Màn hình quản lý học sinh. Có các nút điều khiển sau

Tổ chức: Học kỳ, Lớp, Môn học, hình thức kiểm tra, danh hiệu, qui định chung, Thoát

Khai giảng: Tiếp nhận hồ sơ, Xếp lớp

Học tập: Điểm danh, Bảng điểm, tra cứu học sinh

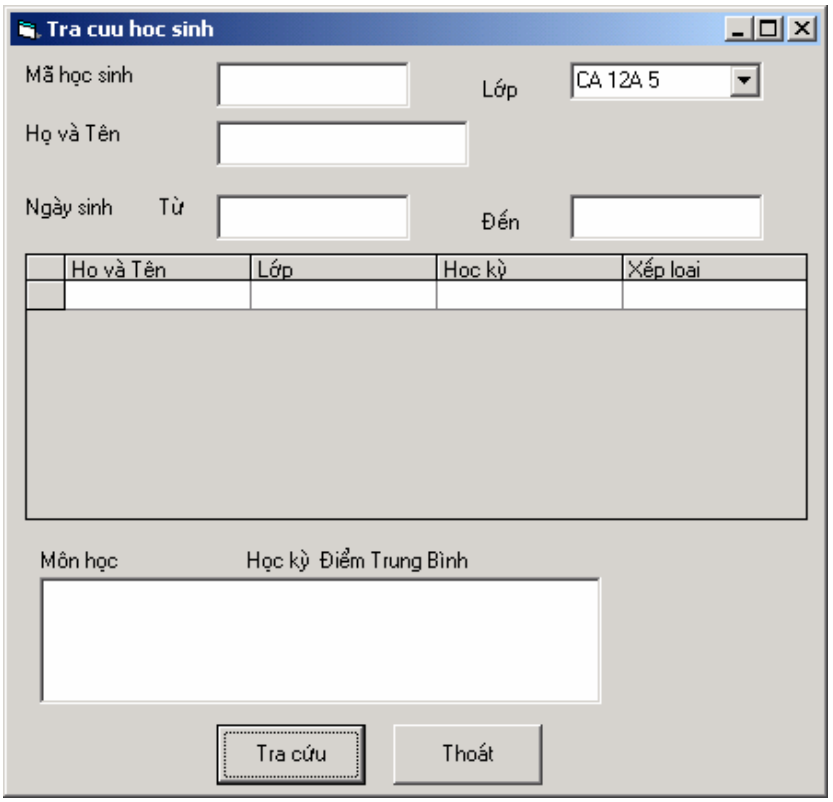
Tổng kết: Xếp loại học sinh, Báo cáo tổng hợp



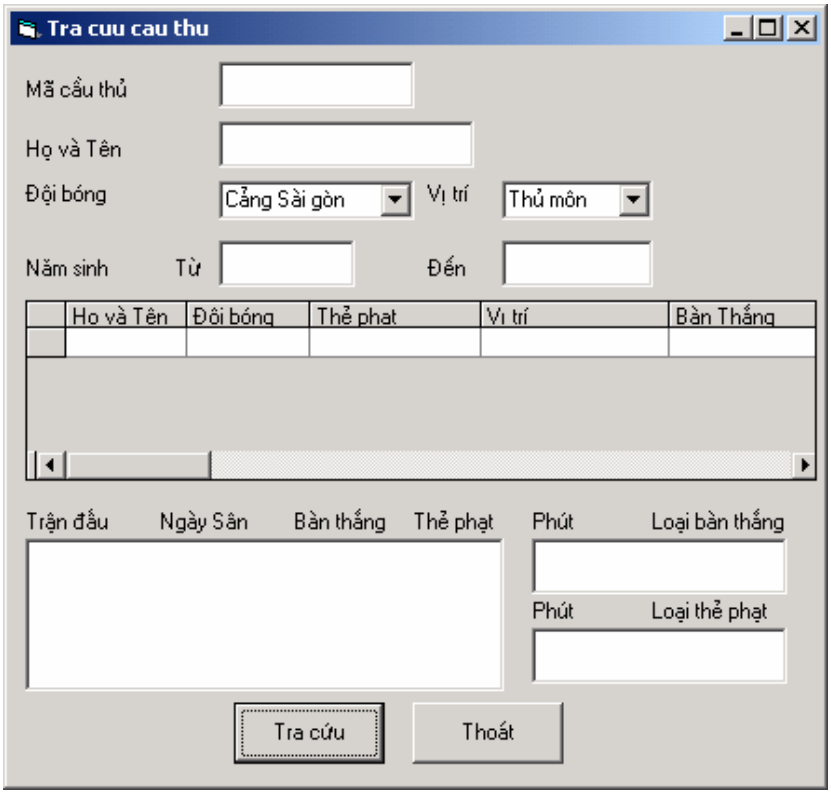
2. Màn hình tra cứu

Mở rộng các tiêu chuẩn tra cứu (các thông tin khác về đối tượng cần tìm). Mở rộng kết quả tìm kiếm (các thông tin liên quan đến đối tượng khi tìm thấy). Cho phép người dung xem các kết quả tra cứu dưới nhiều hình thức trình bày khác nhau (các thứ tự khác nhau với một danh sách, các dạng thể hiện biểu đồ, hình ảnh, v.v.)

Ví dụ 1: Tra cứu học sinh



Ví dụ 2: Tra cứu cầu thủ



3. Màn hình nhập liệu

Chọn dạng trình bày là biểu mẫu liên quan (nếu có) và bổ sung vào đó các thông tin giúp việc sử dụng thuận tiện hơn. Nếu không có biểu mẫu liên quan, cố gắng thiết kế hình thức trình bày tự nhiên nhất có thể có.

2. Thiết kế màn hình

2.1 Mô tả màn hình chính

- **Ý nghĩa sử dụng:**

Màn hình chính là màn hình cho phép ngươi dùng chọn được công việc mà họ muốn thực hiện với phần mềm. Thông thường mỗi phần mềm chỉ có một màn hình chính duy nhất.

- **Nội dung:** Danh sách các công việc có thể thực hiện với phần mềm
- **Hình thức trình bày**

Phím nóng: Hình thức này cho phép chọn nhanh một công việc cần thực hiện đối với người sử dụng chuyên nghiệp. Thông thường không được sử dụng riêng rẻ mà phải kết hợp với các hình thức khác.

Thực đơn: nhóm từng công việc theo chức năng (ví dụ lưu trữ, kết xuất). Đây là dạng sử dụng thông dụng nhất.

Biểu tượng: công việc thể trực quan qua biểu tượng (ký hiệu hay hình ảnh tượng trưng cho công việc. Tương tự như phím nóng nhưng thông dụng hơn và thường kết hợp với các hình thức khác.

Sơ đồ: Dùng sơ đồ để hiển thị trực quan các đối tượng chính, được thể hiện qua các thao tác trực tiếp trên sơ đồ.

Tích hợp: Sử dụng đồng thời nhiều hình thức, thông thường hình thức thực đơn sẽ được ưu tiên trước và kết hợp với nhiều hình thức khác.

- **Thao tác người dùng**

Trên màn hình này thao tác chính của ngừoi dùng là chọn công việc trong danh sách các công việc được đưa ra bởi phần mềm.

2.2 Thiết kế màn hình chính dùng thực đơn (menu)

- **Tổ chức của thực đơn**

Thực đơn bao gồm nhiều nhóm chức năng (tương ứng nhóm các công việc) mỗi nhóm chức năng bao gồm nhiều chức năng, mỗi chức năng tương ứng với một công việc.

- **Phân loại thực đơn: có 3 loại**

- Thực đơn hướng chức năng: Các nhóm chức năng tương ứng với các loại yêu cầu

Ví dụ:

Tổ chức: các công việc liên quan đến tổ chức

Lưu trữ: Các công việc liên quan đến lưu trữ

Tra cứu: Các công việc liên quan đến tra cứu tìm kiếm

- Thực đơn hướng đối tượng: Các nhóm chức năng tương ứng với các lớp đối tượng. Với sơ đồ lớp gồm n lớp đối tượng, thực đơn sẽ bao gồm (n+1) nhóm chức năng. Trong đó:

+ Một nhóm chức năng tương ứng với đối tượng thế giới thực.

+ n nhóm chức năng tương ứng n lớp đối tượng.

- Thực đơn hướng qui trình: Các nhóm chức năng tương ứng với các giai đoạn trng hoạt động của thế giới thực. Thông thường thế giới thực bao gồm các giai đoạn sau như Tổ chức, Kế hoạch, Tiếp nhận, Hoạt động, Tổng kết.

3. Thiết kế màn hình tra cứu

3.1 Mô tả màn hình tra cứu

- **Ý nghĩa sử dụng**

  Màn hình tra cứu là màn hình cho phép người dùng tìm kiếm và xem các thông tin về các đối tượng.

- **Nội dung**

  + Tiêu chuẩn tra cứu: Các thông tin được sử dụng cho việc tìm kiếm (thông thường là các thuộc tính).

  + Kết quả tra cứu: Cho biết có tìm thấy hay không. Các thông tin cơ bản về đối tượng tìm kiếm (các thuộc tính). Các thông tin về quá trình hoạt động của đối tượng (quan hệ với các đối tượng khác).

- **Hình thức trình bày**

  +Tiêu chuẩn tra cứu: Biểu thức logic, Cây, tích hợp

  + Kết quả tra cứu: Thông báo, danh sách đơn, xâu các danh sách, cây danh sách.

- **Thao tác người dùng:** Nhập các giá trị cho các tiêu chuẩn tra cứu, yêu cầu bắt đầu tra cứu, xem chi tiết các kết quả tra cứu.

3.2 Thể hiện tiêu chuẩn tra cứu

3.2.1 Tra cứu với biểu thức logic

Tiêu chuẩn tra cứu được thể hiện dưới dạng một biểu thức logic có dạng như sau:

<Biểu thức logic>=<biểu thức cơ sở>

Phép toán logic …

Phép toán AND, OR, NOT, phép so sánh

3.2.2 Tra cứu với hình thức cây

Tiêu chuẩn tra cứu được thể hiện qua cây mà các nút chính là các bộ phận trong tổ chức của thế giới thực. Hình thức này rất thích hợp với các thế giới thực có cấu trúc tổ chức phân cấp.

3.2.3 Tích hợp

Sử dụng đồng thời cả hai hình thức trên

3.3 Thể hiện kết quả tra cứu

3.3.1 Kết quả tra cứu dùng thông báo

Với hình thức này kết quả tra cứu chỉ đơn giản là câu thông báo cho biết có hay không đối tượng cần tìm. Đây là hình thức đơn giản nhất và có tính tiện dụng thấp nhất. Với hình thức này người sử dụng không biết thêm thông tin gì của đối tượng tìm thấy.

3.3.2 Kết quả tra cứu dùng danh sách đơn

Với hình thức này kết quả tra cứu là danh sách các đối tượng tìm thấy cùng với một số thông in cơ bản về đối tượng. Hình thức này cho phép người dùng biết thêm thông tin cơ bản về đối tượng tìm thấy nhưng không biết chi tiết về các hoạt động của đối tượng qua các quan hệ với đối tượng khác.

3.3.3 Kết quả tra cứu dùng xâu các danh sách

Với hình thức này kết quả tra cứu bao gồm nhiều danh sách mà trong đó danh sách thứ k dsk chứa các mô tả cho một phẩn tử trong danh sách thứ k-1 dsk-1. Danh sách đầu tiên chính là danh sách đơn trong hình thức trên.

Hình thức này không những cho phép xem các thông tin cơ bản về đối tượng tìm thấy mà còn cho biết chi tiết về hoạt động của đối tượng qua các quan hệ với các đối tượng khác.

3.3.4 Cây các danh sách

Với hình thức này kết quả tra cứu là cây mà các nút chính là các danh sách. Danh sách tương ứng trong một nút con sẽ là các thông tin mô tả chi tiết về một phần tử được chọn trong danh sách của nút cha. Danh sách đầu tiên chính là danh sách đơn trong hình thức phía trên.

Hình thức trình bày này cho phép xem được quá trình hoạt động của đối tượng với nhiều quan hệ, nhiều loại hoạt động khác.

3.4 Thao tác người dùng và xử lý của phần mềm

- Nhập giá trị cho các tiêu chuẩn tra cứu:

  - Có thể nhập một số hoặc tất cả tiêu chuẩn tra cứu

  -Với các tiêu chuẩn thường dùng có thể dùng giá trị định sẳn (loại sách thường tìm, loại hàng thường mua, …) để tiện dụng hơn cho người dùng.

  - Trong quá trình nhập liệu thông thường phần mềm sẽ không có xử lý tính toán nào ngoài việc chờ nhập giá trị cho các tiêu chuẩn tra cứu.

- Yêu cầu bắt đầu tra cứu:

- Nhấn vào nút tra cứu.

- Dựa vào giá trị các tiêu chuẩn tra cứu phần mềm sẽ tiến hành đọc và xuất các kết quả tra cứu tương ứng (xử lý tra cứu).

- Xem xét chi tiết các kết quả tra cứu

- Chọn đối tượng cần xem chi tiết trong danh sách của kết quả tra cứu..

- Nhập phạm vi thời gian cần quan sát (thông thường là thời gian từ ngày … đến này … hoặc đơn vị thời gian cụ thể tháng … năm …).

- Dựa vào đối tượng được chọn và dựa vào phạm vị thời gian, phần mềm sẽ đọc và xuất các kết quả tra cứu cấp chi tiết hơn theo từng loại hoạt động.

- Yêu cầu kết xuất

- Có thể bổ sung các nút điều khiển tương ứng với việc in ấn hoặc ghi lên tập tin các kết quả tra cứu. Thông thường mỗi kết quả tra cứu sẽ có một nút riêng, nhưng cũng có thể dùng chung một nút cho mọi kết quả tra cứu (dựa vào kết quả hiện hành).

- Việc kết xuất thông thường là qua máy in, tuy nhiên cũng có thể cho phép người dùng xác định lại đích của kết xuất (tập tin Excel, trang web,…) tùy theo mục đích sử dụng.

4. Thiết kế màn hình nhập liệu

4.1 Mô tả màn hình nhập liệu

- **Ý nghĩa sử dụng:**

Màn hình nhập liệu là màn hình cho phép người dùng thực hiện các công việc có liên quan đến ghi chép trong thế giới thực.

- **Nội dung:**

- Các thông tin nhập liệu: Với loại thông tin này, người dùng chịu trách nhiệm nhập trực tiếp các giá trị, phần mềm sẽ tiến hành kiểm tra tính hợp lệ các giá trị nhập dựa vào qui định liên quan.

- Các thông tin tính toán: Với thông tin này, phần mềm chịu trách nhiệm tính toán và xuất trên màn hình. Thông thường loại thông tin này giúp việc nhập liệu thuận tiện hơn (nhập số lượng hàng bán khi biết số lượng đang tồn tương ứng, nhập sách mượn khi biết số sách độc giả đang mượn …).

- **Hình thức trình bày:** Một số hình thức thông dụng

- Danh sách: Màn hình nhập liệu có dạng một danh sách trong thế giới thực (danh sách các thể loại sách, danh sách các lớp học).

- Hồ sơ: Màn hình nhập liệu có dạng một hồ sơ với nhiều thông tin chi tiết (hồ sơ học sinh, hồ sơ cầu thủ).

- Phiếu: Màn hình nhập liệu có dạng phiếu với nhiều dòng chi tiết (hóa đơn bán hàng, phiếu nhập hàng, …).

- Tích hợp: Sử dụng đồng thời các hình thức trên.

- **Thao tác người dùng:** Có 3 thao tác cơ bản trên màn hình nhập liệu.

  - Nút Ghi: Lưu trữ thông tin.
  - Nút Xóa: Xóa các thông tin đã lưu trữ.
  - Nút Tìm: Tìm và cập nhật lại thông tin đã lưu trữ.

  Ngoài ra để tăng tính tiện dụng có bổ sung các thao tác khác.

  - Tạo phím nóng: Định nghĩa các phím nóng tương ứng với các giá trị nhập liệu thường dùng, điều này cho phép tăng tốc độ nhập liệu.
  - Tạo các nút chuyển điều khiển: Chuyển điều khiển trực tiếp đến màn hình khác có liên quan đến việc nhập liệu hiện hành (bổ sung thể loại sách mới, nhà xuất bản mới, …).

4.2 Các hình thức trình bày màn hình nhập liệu

4.2.1 Thiết kế màn hình nhập liệu dạng danh sách

- **Sử dụng:** Dạng danh sách thích hợp khi cần nhập liệu các bảng danh sách với kích thước nhỏ (danh sách các thể loại sách, các môn học,…).

- **Thành phần nhập liệu:**

  - Thông tin nhập liệu: Các thuôc tính các bảng liên quan

  - Thông tin tính toán: Thông thường các mã số được tự động phát sinh.

- **Thành phần xử lý:**

  - Ghi: Ghi nhận các thay đổi trên danh sách (thêm mới, sửa đổi.

  - Xóa: Xóa 1 dòng trong danh sách.

  - Thoát: Quy về màn hình trước đó.

- **Các thao tác:**

  Người dùng có thể tùy ý sửa đổi các thông tin trên các dòng hoặc thêm dòng mới (nhập vào cuối danh sách), xóa dòng sau khi đã chọn dòng cần xóa và cuối cùng yêu cầu ghi các thay đổi trên bộ nhớ phụ.

  Tuy nhiên trong một số trường hợp đặc biệt một số thao tác có thể bị cấm (không cho xóa, không cho thay đổi một số thuộc tính …) tùy vào ý nghĩa cụ thể của danh sách.

4.2.2 Thiết kế màn hình nhập liệu dạng hồ sơ

- **Sử dụng**: Dạng hồ sơ thích hợp khi cần nhập liệu các hồ sơ các đối tượng trong thế giới thực (hồ sơ học sinh, đội bóng).
- **Thành phần dữ liệu**:

  - Thông tin nhập liệu: Các thuộc tính các bảng liên quan

  - Thông tin tính toán: Thông thường các mã số được tự động phát sinh.

- **Thành phần xử lý**: Thêm, Ghi, Xóa, Tìm, Thoát
- **Các thao tác**: Người dùng có thể thêm hồ sơ mới, tìm lại hồ sơ đã lưu trữ và sau đó tuỳ ý sửa đổi, các thông tin trên hồ sơ tìm thấy, xóa hồ sơ tìm thấy, và cuối cùng yêu cầu lưu trữ hồ sơ. Tuy nhiên để tăng tính tiện dụng một số thao tác chuyển điều khiển có thể được bổ sung cho phép di chuyển nhanh đến các màn hình nhập liệu liên quan khi cần thiết.

4.2.3 Thiết kế màn hình nhập liệu dạng phiếu

- **Sử dụng**: Dạng phiếu thích hợp khi cần nhập liệu các phiếu ghi nhận thông tin về hoạt động các đối tượng trong thế giới thực.
- **Thành phần dữ liệu**:

  - Thông tin nhập liệu: Các thông tin liên quan đến bảng.

  - Thông tin tính toán: Thông thường các mã số được tự động phát sinh.

  Thành phần xử lý: Thêm, Thêm chi tiết, Ghi, Xóa, Xóa chi tiết, Tìm, Sửa chi tiết, Thoát.

Chương 6: CÀI ĐẶT

1. Tổng quan

Trong cài đặt, chúng ta xuất phát từ kết quả của thiết kế và cài đặt hệ thống dưới dạng các thành phần, tức là các mã nguồn, các kịch bản, các tập tin nhị phân, các tập tin thực thi, thư viện, bảng, dữ liệu

May thay, phần lớn kiến trúc của hệ thống đã được nắm bắt trong quá trình thiết kế. Mục tiêu chủ yếu của cài đặt là bổ sung thêm cho kiến trúc và hệ thống để trở thành một khối hoàn chỉnh. Cụ thể hơn, các mục tiêu cài đặt là:

- Lên kế hoạch tích hợp hệ thống (system integration) trong mỗi bước lặp một các tăng cường. Điều này có nghĩa là một hệ thống được cài đặt bởi một dãy các bước nhỏ liên tiếp và có thể quản lý được.
- Phân phối hệ thống bằng cách ánh xạ các thành phần thi hành được vào các nút trong mô hình triển khai. Công việc này chủ yếu dựa vào các lớp động được tìm thấy trong quá trình thiết kế.
- Cài đặt các lớp thiết kế và các hệ thống con đã tìm được trong quá trình thiết kế. Đặc biệt, các lớp thiết kế được cài đặt thành các thành phần file chứa mã nguồn.
- Kiểm thử đơn vị các thành phần, rồi sau đó tích hợp chúng bằng các biên dịch chúng và liên kết chúng lại với nhau thành một hoặc nhiều thành phần thi hành được trước khi kiểm thử tích hợp và kiểm thử hệ thống

Để đạt được những mục tiêu trên, chương này sẽ đưa ra những bàn luận chung quá trình cài đặt phần mềm. Trước tiên chúng ta đưa ra khái quát chung của tiêu chí chất lượng và yếu tố then chốt của ngôn ngữ lập trình và chúng ta sẽ thảo luận trên những phần chính của kiểu lập trình như cấu trúc, những diễn giải, hình thức, và hiệu qủa. Cuối cùng chúng ta sẽ bàn đến thuộc tính chính của phần mềm như khả năng uyển chuyển và dùng lại.

Một cài đặt tốt phản ánh những quyết định của thiết kế

Cài đặt nên đảm bảo theo các mục sau:

- Cấu trúc, cấu trúc dữ liệu và những định nghĩa được chọn lựa và thiết lập trong suốt thủ tục thiết kế cần được tổ chức dễ dàng nhận biết trong quá trình cài đặt.

- Mức trừu tượng của thiết kế (các lớp (class), mô đun (module), thuật toán (algorithm), cấu trúc dữ liệu (data structure), và kiểu dữ liệu (data type)) cũng phải linh động trong thực hiện.
- Giao diện giữa các thành phần (components) của hệ thống phần mềm được mô tả rõ ràng trong thực hiện.
- Quá trình thực hiện cũng có thể được kiểm tra độ tin cậy của đối tượng và thao tác với trình biên dịch (trước khi qua giai đoạn kiểm tra chương trình thực sự).

Đảm bảo những đặc trưng ở trên phụ thuộc vào việc chọn lựa ngôn ngữ thực hiện và kiểu lập trình.

2. Môi trường lập trình

Câu hỏi cho việc chọn lựa "đúng" ngôn ngữ lập trình luôn là chủ đề được đưa ra trong qui trình lập trình.Việc chọn lựa ngôn ngữ lập trình trong công đoạn thực thi của một dự án luôn đóng vai trò quan trọng.

Trong trường hợp lý tưởng, thiết kế nên đảm trách mà không có bất cứ kiến thức liên quan đến ngôn ngữ thực hiện sau đó sao cho thiết kế có thể thực hiện được trên bất kỳ ngôn ngữ lập trình nào.

2.1 Chất lượng đòi hỏi cho một ngôn ngữ lập trình:

- Tính mô đun hóa
- Giá trị của tài liệu
- Cấu trúc dữ liệu
- Control flow (luồng điều khiển)
- Tính hiệu quả
- Khả năng tích hợp (Integrity)
- Tính khả chuyển (Portability)
- Hỗ trợ hộp thoại
- Yếu tố ngôn ngữ chuyên biệt

2.2 Khả năng Mô đun hóa của ngôn ngữ lập trình

Khả năng mô đun hóa là mức độ hỗ trợ những khả năng mô đun hóa chương trình. Phác thảo một chương trình lớn thành nhiều mô đun là điều kiện tiên quyết để thực thi trong dự án phần mềm.

Không có khả năng mô đun hóa thì phân chia công việc trong giai đoạn thực hiện trở nên không thể được. Những chương trình đơn nhất trở nên không thể quản lý: chúng khó có thể bảo trì và sưu liệu và chúng thực hiện với thời gian biên dịch dài.

Ngôn ngữ như Pascal chuẩn (không hỗ trợ mô đun, nhưng so sánh với Turbo Pascal và Mođun 2) để chứng minh tính không thích hợp cho những dự án lớn.

Nếu một ngôn ngữ hỗ trợ phát thảo một chương trình thành những phần nhỏ, chúng phải đảm bảo những thành phần phải hoạt động với nhau. Nếu một thủ tục được thực thi ở mô đun khác, cũng được kiểm tra thủ tục có thực sự tồn tại và nó có được sử dụng chính xác hay không (nghĩa là số tham số và kiểu dữ liệu là chính xác).

Những ngôn ngữ với việc biên dịch độc lập (ví dụ như C) nơi việc kiểm tra của ngôn ngữ chỉ thay thế ở quá trình run-time.

Ngôn ngữ với việc biên dịch tách biệt (ví dụ Ada và Modula-2) mỗi mô đun có một mô tả giao diện cung cấp những phương thức cơ bản cho việc kiểm tra những thành phần của mô đun dùng tại thời điểm chạy chương trình (run time).

2.3 Giá trị sưu liệu của ngôn ngữ lập trình

Ảnh hưởng của khả năng có thể đọc và bảo trì của chương trình. Điều quan trọng của giá trị sưu liệu được nâng lên đối với những chương trình lớn và cho những phần mềm mà khách hàng vẫn tiếp tục phát triển.

Giá trị của sưu liệu cao mang lại kết quả hơn. Vì chương trình nói chung chỉ được 1 lần nhưng việc đọc nó có thể lặp lại, hiệu quả tối thiểu thêm vào trong cách viết sẽ chịu ảnh hưởng không đâu nhiều hơn là trong quá trình bảo trì. Giống như phạm vi ngôn ngữ ảnh hưởng đến khả năng đọc chương trình

Nhiều ngôn ngữ mở rộng với quá nhiều chức năng chuyên biệt sẽ khó để hiểu thấu tất cả chi tiết, vì vậy dẫn đến giải thích sai.

2.4 Cấu trúc dữ liệu trong ngôn ngữ lập trình

Dữ liệu phức tạp phải được xử lý, sự sẵn sàng trong cấu trúc dữ liệu trong ngôn ngữ lập trình đóng vai trò quan trọng.

Ngôn ngữ như C cho phép khai báo con trỏ đối với cấu trúc dữ liệu. Điều này cho phép cấu trúc dữ liệu phức tạp, và phạm vi và cấu trúc của chúng có thể thay đổi ở thời điểm run-time. Tuy nhiên, việc drawback những cấu trúc dữ liệu chúng được mở và được phép truy xuất không nghiêm ngặt (nhưng so sánh với Java).

Trọng tâm trong dự án lớn với nhiều nhóm dự án, dữ liệu trừu tượng mang nghĩa cụ thể. Mặc dù dữ liệu trừu tượng có thể phân biệt với bất kỳ mô đun ngôn ngữ, bởi khả năng đọc tốt hơn.

Ngôn ngữ lập trình hướng đối tượng có những đặc trưng mở rộng loại kiểu dữ liệu trừu tượng cho phép hiện thực hoá những hệ thống phần mềm phức tạp. Đối với những giải pháp mở rộng và uyển chuyển, ngôn ngữ lập trình hướng đối tượng cung cấp tuỳ chọn đặc biệt tốt.

2.5 Ví dụ minh họa

Ví dụ: Giai đoạn thực hiện phần mềm quản lý thư viện, các giai đoạn trước đã được minh họa ở các chương trước

Giai đoạn 5: Thực hiện phần mềm

- Hệ thống lớp đối tượng: Tạo lập các lớp đối tượng (THU_VIEN, DOC_GIA, SACH) theo mô tả của phần thiết kế trong một môi trường cụ thể nào đó (Visual Basic, Visual C++, Java,…)
- Hệ thống giao diện: Tạo lập (vẽ) các màn hình giao diện (màn hình chính, màn hình lập thẻ, màn hình cho mượn sách, màn hình nhận sách, màn hình trả sách) theo mô tả của phần thiết kế trong một môi trường cụ thể nào đó (Visual Basic, Viusal C++, Java)
- Hệ thống lưu trữ: Tạo lập cấu trúc cơ sở dữ liệu (các bảng THU_VIEN, DOC_GIA, SACH, MUON_SACH) theo mô tả của phần thiết kế trong môi trường cụ thể nào đó(Access, SQL Server, Oracle,…)

3. Phong cách lập trình

Sau khi thực hiện và kiểm tra, hệ thống phần mềm hiếm khi được sử dụng một thời gian dài mà không có sửa đổi điều chỉnh. Thực vậy, điều này luôn là đúng: khi yêu cầu được cập nhật hoặc mở rộng sau khi hoàn chỉnh sản phẩm và trongsuốt quá trình thực hiện thao tác, không phát hiện ra lỗi hay những thiếu sót phát sinh. Giai đoạn thực hiện chắc chắn phải được sửa đổi và mở rộng, đòi hỏi lặp lại việc đọc và hiểu chương trình nguồn. Trong trường hợp lý tưởng, chức năng của một thành phần chương trình được hiểu mà không có kiến thức từ tài liệu thiết kế mà chỉ từ chương trình nguồn. Chương trình nguồn chỉ là tài liệu luôn phản ánh hiện trạng của thực thị.

Khả năng đọc được một chương trình phụ thuộc vào ngôn ngữ lập trình được dùng và vào phong cách lập trình của người thực hiện. Việc viết một chương trình có thể đọc được là tiến trình sáng tạo. Phong cách lập trình của người thực hiện ánh hưởng đến khả năng đọc được chương trình hơn là ngôn ngữ lập trình được sử dụng.

Yếu tố quan trọng nhất của phong cách lập trình tốt là:

- Tính cấu trúc
- Sự trình bày diễn đạt
- Cách thức trình bày bên ngoài
- Hiệu suất

3.1 Tính cấu trúc

Việc phân rã một hệ thống phần mềm dựa trên mục đích chính là độ phức tạp thông qua mức trừu tượng phần thành phần cô đọng rõ nét (cấu trúc chương trình lớn).

Chọn lựa những thành phần chương trình phù hợp trong việc định ra những thuật toán của thủ tục con.(cấu trúc chương trình nhỏ).

3.2 Thế mạnh của diễn đạt

Qui trình thực hiện một hệ thống phần mềm chứa đựng việc đặt tên đối tượng và mô tả các công việc thực thi của đối tượng này.

Chọn lựa tên đặc biệt quan trọng trong việc viết thuật toán

Một số đề nghị:

- Nếu dùng chữ viết tắt, thì sử dụng tên đặt này người đọc chương trình có thể hiểu mà không cần bất cứ sự giải thích nào. Việc sử dụng những từ viết tắt chỉ bao gồm ngữ cảnh.
- Với một hệ thống gán tên chỉ nên một ngôn ngữ (ví dụ đừng dùng lẫn lộn tiếng Anh và tiết Việt).
- Dùng chữ hoa chữ thường để phân biệt những loại định nghĩa khác nhau (ví dụ chữ hoa đầu tiên cho kiểu dữ liệu, lớp, mô đun, chữ thường đầu tiên cho biến) và đặt tên dài hơn có thể đọc (ví dụ CheckInputValue).
- Dùng danh từ cho giá trị, động từ cho hoạt động, và thuộc tính cho điều kiện để làm rõ ý nghĩa nhận diện (ví dụ width, ReadKey, valid).
- Thiết lập những qui luật cho chính bạn sử dụng theo chúng một cách thích hợp.

Phong cách lập trình tốt được tìm thấy trong diễn giải sử dụng ghi chú: đóng góp cho khả năng đọc được chương trình và như vậy nó là thành phần quan trọng của chương trình. Hiệu chỉnh việc ghi chú chương trình không dễ dàng và đòi hỏi kinh nghiệm, sáng tạo và khả năng diễn đạt thông điệp gọn gàng và chính xác.

Một số luật cho việc viết những ghi chú:

- Mỗi thành phần hệ thống (mỗi mô đun và lớp) nên bắt đầu với ghi chú chi tiết cho người đọc những thông tin với một vài vấn đề liên quan đến thành phần của hệ thống:
    - Thành phần này làm gì?
    - Thành phần này được sử dụng như thế nào trong những ngữ cảnh gì?
    - Những phương thức đặc biệt được sử dụng.
    - Ai là Tác giả của thành phần này?
    - Thành phần này được viết khi nào?
    - Những sửa đổi cập nhật nó được thực hiện.
- Mỗi thủ tục và phương thức cung cấp ghi chú mô tả công việc (có thể có). Điều này ứng dụng đặt biệt cho đặc tả giao diện.
- Giải thích ý nghĩa của biến với ghi chú.
- Những thành phần của chương chịu trách nhiệm cho những tác nhiệm riêng nên được đánh nhãn với những ghi chú.
- Những khối lệnh khó để hiểu (ví dụ thủ tục rắc rối hay những thành phần mà đặc trưng cho một máy tính cụ thể) nên được mô tả ghi chú sao cho người đọc dễ dàng hiểu chúng.
- Hệ thống phần mềm nên chứa mà một vài ghi chú gãy gọn súc tích như nếu có thể nhưng nhiều ghi chú chi tiết tương xứng nếu cần thiết.
- Đảm bảo những thay đổi chương trình không chỉ có tác động phần khai báo và khối lệnh mà còn phản ánh những cập nhật trong phần ghi chú. Những ghi chú không chính xác thì sẽ tệ hơn.

Lưu ý: những luật trên tuân thủ cân nhắc bởi vì không có luật áp dụng đồng nhất cho tất cả các hệ thống phần mềm và mỗi phạm vi ứng dụng. Việc ghi chú hệ thống phần mềm là một nghệ thuật cũng giống như phần thiết kế cài đặt hệ thống phần mềm.

3.3 Cách thức trình bày bên ngoài

Ngoài sự chọn tên và ghi chú, khả năng đọc được của hệ thống phần mềm cũng phụ thuộc vào cách thức trình bày bên ngoài.

Một số luật đề nghị cho hình thức trình bày chương trình:

- Mỗi thành phần của chương trình (components), những khai báo (của kiểu dữ liệu, hằng biến, …) nên được tách biệt mỗi phần của khối lệnh.

- Phần khai báo nên có một cấu trúc đồng nhất khi có thể như thứ tự sau: hằng, kiểu dữ liệu, lớp, mô đun, phương thức và thủ tục.
- Mô tả giao diện (danh sách tham số cho phương thức và thủ tục) nên tách tham số nhập liệu, kết xuất và nhập/xuất.
- Phần ghi chú và chương trình nguồn nên tách bạch.
- Cấu trúc của chương trình nên được nhấn mạnh ở phần canh chỉnh lề (sử dụng phím tab cho từ mỗi đầu khối lệnh đến khối lệnh theo sau).

4. Đánh giá chất lượng công việc

4.1 Hiện thực tăng cường

Ý tưởng cơ bản của việc hiện thực tăng cường gần với việc trộn giai đoạn thiết kế và cài đặt hơn là tách biệt hai giai đoạn này mà mô hình qui trình phát triển phần mềm tuần tự cổ điển đề ra.

Điểm nhấn mạnh của phương pháp này được tìm thấy dựa trên thực nghiệm rằng những quyết định trong thiết kế và cài đặt có tác động lẫn nhau và vì vậy nếu tách bạch thiết kế khắt khe sẽ không đạt được mục tiêu. Trong nhiều trường hợp, chỉ có cài đặt mới quyết định việc phân rã cấu trúc của thiết kế chứng minh sự thõa mãn đầy đủ vấn đề.

Hiện thực tăng cường nghĩa là sau mỗi bước thiết kế có liên quan đến kiến trúc, kiến trúc phần mềm hiện hành được thẩm định dựa trên những trường hợp thực. Sự tác động qua lại giữa các thành phần hệ thống cụ thể trong thiết kế (trong hình thức đặc tả giao diện) được thẩm định. Để có thể làm được điều này, những thành phần hệ thống (hành vi xuất/ nhập của chúng) được mô phỏng hay thực tế hóa như khuôn mẫu. Nếu có những nghi ngờ liên quan đến tính khả thi của thành phần thì tiến trình thiết kế được ngắt và những thành phần được thực hiện. Chỉ khi hiện thực và nhúng chúng vào trong kiến trúc hệ thống trước đó được kiểm tra thì tiến trình thiết kế tiếp tục hay kiến trúc được chấp nhận tương ứng kiến thức thu được trong hiện thực thành phần.

Hiệu quả của phương pháp này phụ thuộc vào việc mở rộng vào khả năng tích hợp thành phần hệ thống mà được viết trong chuẩn mực khác nhau và được hoàn chỉnh ở nhữn cấp độ khác nhau, đối với toàn bộ hệ thống để thực hiện gần với thực tế. Một vài thành phần hệ thống, ví dụ giao diện người dùng và mô hình dữ liệu được thể hiện dưới dạng khuôn mẫu, những thành phần khác từ thư viện thành phần có sẵn hay tồn tại như hiện thực hoàn chỉnh được thể hiện dưới dạng mã nguồn thực thi còn các thành phần hệ thống khác có sẵn như đặc

tả giao diện. Đối với sự hợp lệ của thiết kế hệ thống hiện hành, bất kỳ lúc nào giao diện người dùng triển khai thì tương ứng khuôn mẫu cần được kích hoạt.

4.2 Đánh giá lại thiết kế và chương trình (Design and Code Review)

Với việc xem lại thiết kế và chương trình, sẽ giúp hoàn chỉnh chất lượng hiệu quả của công việc hơn là chúng ta chỉ điều chỉnh những thay đổi đơn lẻ trong quá trình phát triển phần mềm.Trong những chương trình lớn, đòi hỏi xem xét lại những yêu cầu, đặc tả, thiết kế, và cả chương trình của chúng ta. Giúp điều chỉnh thiếu sót, logic, cấu trúc, tính sáng tỏ. Khi chương trình không rõ hay mơ hồ xáo trộn, thêm những ghi chú thì tốt hơn hay viết lại nó một cách đơn giản hơn sẽ làm cho chương trình dễ đọc và dễ hiểu. Việc làm này sẽ tạo cho chúng ta sự tự tin xuất bản hay trình bày cho bạn bè hay tập thể.

Mục đích của review để đảm bảo chương trình tạo ra đạt chất lượng cao nhất. Một trong việc review là kiểm duyệt, duyệt qua, xem xét mục riêng từ thiết kế đến từng dòng lệnh. Review có thể được dùng trên yêu cầu, thiết kế, hồ sơ tài liệu, hay bất kỳ yếu tố của sản phẩm.

Nhiều dự án phần mềm trãi qua nữa qúa trình phát triển ở giai đoạn kiểm thử. Điều này không hiệu quả. Review thiết kế và chương trình là những cách thức hiệu quả tìm và sửa chữa thiếu sót. Với review, chúng ta có thể tìm ra những thiếu sót trực tiếp, trong khi chúng ta chỉ có thể tìm ra những dấu hiệu. Khi chúng ta xem lại chương trình, chúng ta biết nơi và những logic gì giả sử phải được làm.Những sửa lỗi của chúng ta sẽ hoàn chỉnh và chính xác.

Công việc review cho phép quay trở lại bất cứ việc gì chúng ta đã làm. Nhóm phát triển nên ngồi lại với nhau để đọc lại mọi thiết kế và chương trình, nghiên cứu, hiểu nó. Sửa những sai sót: logic, cấu trúc, tính rõ ràng. Sau khi đã được xem xét và đánh giá, viết lại chương trình. Chỗ nào không rõ ràng hay lộn xộn, thêm ghi chú và viết lại hoàn chỉnh làm cho dễ đọc và dễ hiểu.

5. Ví dụ minh họa

Ví dụ: Xét phần mềm hỗ trợ giải bài tập phương trình đại só với 4 yêu cầu: Soạn đề bài, Soạn đáp án, Giải bài tập, Chấm điểm.

Nhằm thể hiện các giai đoạn thực hiện trong qui trình

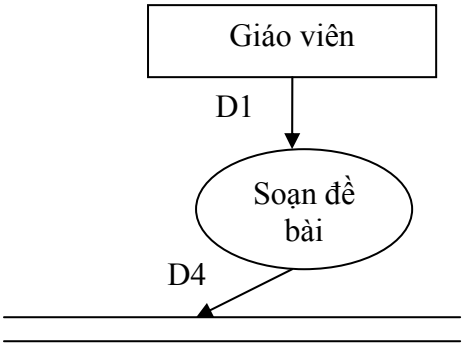
Giai đoạn 1: Xác định yêu cầu

- Yêu cầu 1: Soạn đề bài với qui định về Soạn đề bài
- Yêu cầu 2: Soạn đáp án với qui định về Soạn đáp án và biểu mẫu Soạn đáp án

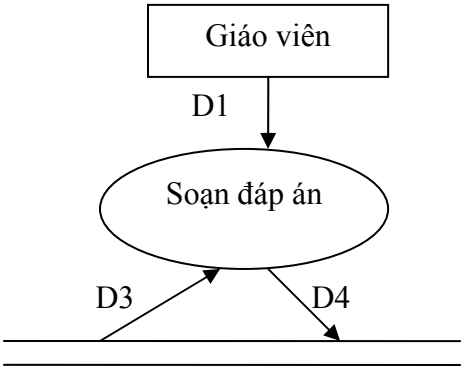
- Yêu cầu 3: Giải bài tập với qui định về Giải bài tập và biểu mẫu Giải bài tập
- Yêu cầu 4: Chấm điểm với qui định về Chấm điểm

Giai đoạn 2:

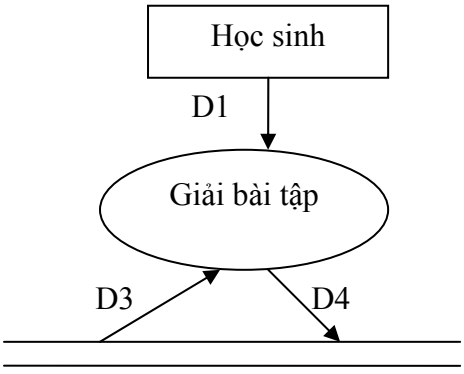
- Sơ đồ luồng dữ liệu cho công việc Soạn đề bài



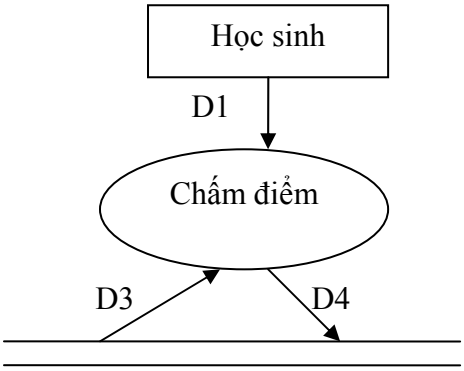
- Sơ đồ luồng dữ liệu cho công việc Soạn đáp án



- Sơ đồ luồng dữ liệu cho công việc Giải bài tập



- Sơ đồ luồng dữ liệu cho công việc Chấm điểm



Giai đoạn 3: Phân tích yêu cầu chức năng

Giai đoạn 4: Thiết kế phần mềm (đã trình bày trong phần kiến trúc phần mềm của chương Thiết kế phần mềm)

Giai đoạn 5: Thực hiện phần mềm

- Hệ thống Lớp đối tượng: Tạo lập các lớp đối tượng SACH_BAI_TAP, BAI_TAP theo mô tả phần thiết kế trong môi trường cụ thể nào đó (VIsual Basic, Visual C++, Java
- Hệ thống giao diện: Tạo lập (vẽ) các màn hình giao diện (màn hình chính, màn hình soạn đề bài, màn hình soạn đáp án, màn hình giải bài tập, màn hình chấm điểm) theo mô tả của phần thiết kế trong một môi trường cụ thể nào đó (Visual Basic, Visual C++, Java, v.v.)

Hệ thông lưu trữ: Tạo cấu trúc cơ sở dữ liệu (các bảng SACH_BAI_TAP, BAI_TAP, BAI_GIAI, BUOC_GIAI) theo mô tả của phần thiết kế trong một môi trường cụ thể nào đó (Access, SQL Server, Oracle,. v.v)

Giai đoạn 6: Kiểm chứng phần mềm xem ở chương Kiểm thử

Chương 7: KIỂM THỬ PHẦN MỀM

1. Tổng quan

Kiểm thử phần mềm là tiến hành thí nghiệm để so sánh kết quả thực tế với lý thuyết nhằm mục đích phát hiện lỗi.

Bộ thử nghiệm (test cases) là dữ liệu dùng để kiểm tra hoạt động của chương trình. Một bộ kiểm thử tốt là bộ có khả năng phát hiện ra lỗi của chương trình. Khi tiến hành kiểm thử, chúng ta chỉ có thể chứng minh được sự tồn tại của lỗi nhưng không chứng minh được rằng trong chương trình không có lỗi.

Nội dung của bộ thử nghiệm:

- Tên môđun/chức năng muốn kiểm thử
- Dữ liệu vào
  - Dữ liệu của chương trình: số, xâu ký tự, tập tin,
  - Môi trường thử nghiệm: phần cứng, hệ điều hành,
  - Thứ tự thao tác (kiểm thử giao diện)
- Kết quả mong muốn
  - Thông thường: số, xâu ký tự, tập tin, …
  - Màn hình, thời gian phản hồi
- Kết quả thực tế

Không gian thử nghiệm là tập các bộ số thử nghiệm. Không gian này nói chung là rất lớn. Nếu có thể vét cạn được không gian thử nghiệm thì chắc chắn qua phép kiểm tra đơn vị sẽ không còn lỗi. Tuy nhiên điều này không khả thi trong thực tế. Do đó khi đề cập đến tính đúng đắn của phần mềm chúng ta dùng khái niệm độ tin cậy.

Phương pháp kiểm thử là cách chọn bộ số thử nghiệm để tăng cường độ tin cậy của đơn vị cần kiểm tra. Hay nói cách khác phương pháp kiểm thử là cách phân hoạch không gian thử nghiệm thành nhiều miền rồi chọn bộ số liệu thử nghiệm đại diện cho miền đó. Như vậy cần tránh trường hợp mọi bộ thử nghiệm đều rơi vào một miền kiểm tra.

2. Yêu cầu đối với kiểm thử

- Tính lặp lại:
  - Kiểm thử phải lặp lại được (kiểm tra xem lỗi đã được sửa hay chưa)

- Dữ liệu/trạng thái phải mô tả được

- Tính hệ thống: phải đảm bảo đã kiểm tra hết các trường hợp.
- Được lập tài liệu: phải kiểm soát được tiến trình/kết quả.

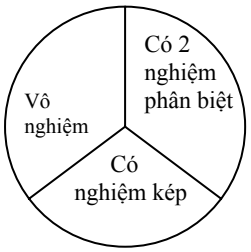
3. Các kỹ thuật kiểm thử

3.1 Phương pháp hộp đen (Kiểm thử chức năng)

Phương pháp kiểm thử này chỉ dựa trên bản đặc tả các chức năng. Do đó, chúng ta chỉ chú tâm đến phát hiện các sai sót về chức năng mà không quan tâm đến cách cài đặt cụ thể. Với phương pháp này chúng ta có khả năng phát hiện các sai sót, thiếu sót về mặt chức năng; sai sót về giao diện của môđun, kiểm tra tính hiệu quả; phát hiện lỗi khởi tạo, lỗi kết thúc.

Do không thể kiểm thử mọi trường hợp trên thực tế, chúng ta sẽ chia không gian thử nghiệm dựa vào giá trị nhập xuất của đơn vị cần kiểm tra. Ứng với mỗi vùng dữ liệu chúng ta sẽ thiết kế những bộ thử nghiệm tương ứng và đặc biệt là các bộ thử nghiệm tại các gía trị biên của vùng dữ liệu.

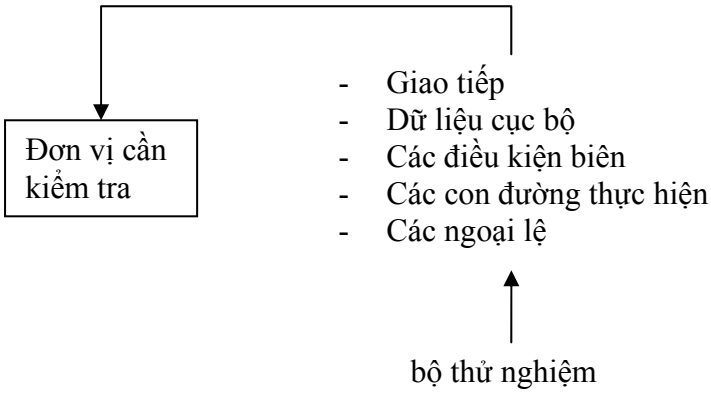
Để kiểm chứng chương trình giải phương trình bậc 2 theo phương pháp hộp đen, chúng ta sẽ phân chia không gian thử nghiệm thành 3 vùng như sau:



Sau khi đã thử kiểm tra với các bộ thử nghiệm đã thiết kế, chúng ta cần mở rộng bộ thử nghiệm cho các trường hợp đặc biệt như: biên của số trong máy tính (32767,-32768), số không, số âm, số thập phân, dữ liệu sai kiểu, dữ liệu ngẫu nhiên.

a. Phương pháp hộp trắng (Kiểm thử cấu trúc)

Theo phương pháp này, chúng ta sẽ chia không gian thử nghiệm dựa vào cấu trúc của



đơn vị cần kiểm tra.

Kiểm tra giao tiếp của đơn vị là để đảm bảo dòng thông tin vào ra đơn vị luôn đúng (đúng giá trị, khớp kiểu...)

Kiểm tra dữ liệu cục bộ để đảm bảo dữ liệu được lưu trữ trong đơn vị toàn vẹn trong suốt quá trình thuật giải được thực hiện.

Ví dụ: nhập dữ liệu sai, tên biến không đúng, kiểu dữ liệu không nhất quán, các ràng buộc hoặc ngoại lệ.

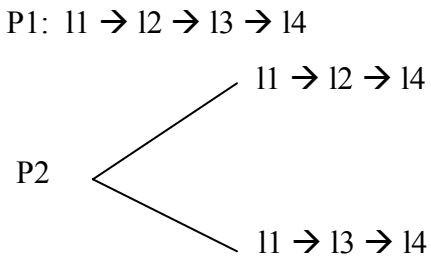
Kiểm tra các điều kiện biên của các câu lệnh if, vòng lặp để đảm bảo đơn vị luôn chạy đúng tại các biên này.

Kiểm tra để đảm bảo mọi con đường thực hiện phải được đi qua ít nhất một lần. Con đường thực hiện của một đơn vị chương trình là một dãy có thứ tự các câu lệnh bên trong đơn vị đó sẽ được thực hiện khi kích hoạt đơn vị.

Ví dụ:

P1	P2
l1	l1
l2	if (đk) l2
l3	else l3
l4	l4

Con đường thực hiện của p1 và p2 như sau:



4. Các giai đoạn và chiến lược kiểm thử

Đối với những dự án phần mềm lớn, những người tham gia được chia thành 2 nhóm:

- Nhóm thứ nhất: gồm những người tham gia trong dự án phát triển phần mềm. Nhóm này chịu trách nhiệm kiểm tra các đơn vị của chương trình để chắc chắn chúng thực hiện đúng theo thiết kế.
- Nhóm thứ hai: độc lập gồm các chuyên gia tin học nhưng không thuộc nhóm thứ nhất. Nhóm này có nhiệm vụ phát hiện các lỗi do nhóm thứ nhất chủ quan còn để lại.

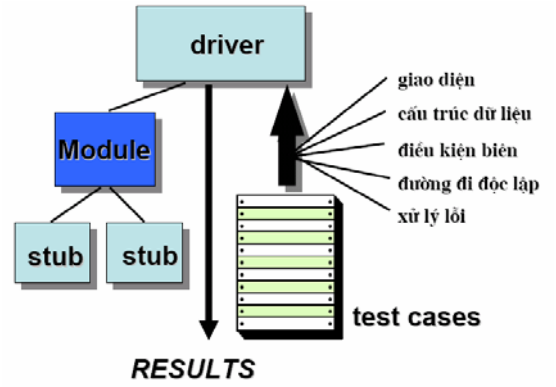
4.1 Kiểm thử đơn vị

Sử dụng kỹ thuật hộp trắng và dựa vào hồ sơ thiết kế để xây dựng các bộ thử nghiệm sao cho khả năng phát hiện lỗi là lớn nhất.

Vì đơn vị được kiểm tra không là 1 chương trình đầy đủ, hơn nữa đơn vị này có thể được gọi bởi những đơn vị khác hoặc gọi đến những đơn vị khác nên dù chương trình đã được hoàn tất đầy đủ các đơn vị, chúng ta cũng không nên giả thuyết sự tồn tại hoặc tính đúng đắn của các đơn vị khác mà phải xây dựng các module giả lập đơn vị gọi tên là driver và đơn vị bị gọi là stub.

Driver đóng vai trò như một chương trình chính nhập các bộ số thử nghiệm và gởi chúng đến đơn vị cần kiểm tra đồng thời nhận kết quả trả về của đơn vị cần kiểm tra.

Stub là chương trình giả lập thay thế các đơn vị được gọi bởi đơn vị cần kiểm tra. Stub thực hiện các thao tác xử lý dữ liệu đơn giản như in ấn, kiểm tra dữ liệu nhập và trả kết quả ra.



4.2 Kiểm thử tích hợp

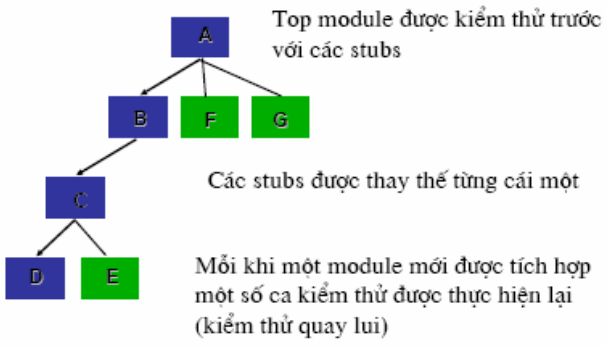
Giai đoạn này được tiến hành sau khi đã hoàn tất công việc kiểm thử từng môđun riêng lẻ bằng cách tích hợp các môđun này lại với nhau. Mục đích của giai đoạn này là kiểm tra giao diện của các đơn vị, kiểm tra tính đúng đắn so với đặc tả, kiểm tra tính hiệu quả.

Phương pháp thực hiện chủ yếu sử dụng kiểm tra chức năng. Các đơn vị có thể được tích hợp theo một trong hai chiến lược: từ trên xuống (top-down) hoặc từ dưới lên (bottom-up).

4.2.1 Trên xuống

Thuật giải của hướng tiếp cận này gồm những bước sau:

- Sử dụng Module chính như 1 driver và các stub được thay cho tất cả các module là con trực tiếp của module chính.
- Lần lượt thay thế các stub mỗi lần 1 cái bởi các module thực sự.
- Tiến hành kiểm tra tính đúng đắn.
- Một tập hợp bộ thử nghiệm được hoàn tất khi hết stub.
- Kiểm tra lùi có thể được tiến hành để đảm bảo rằng không phát sinh lỗi mới.



a) Ưu điểm

Kiểm thử trên xuống kết hợp với phát triển trên xuống sẽ giúp phát hiện sớm các lỗi thiết kế và làm giảm giá thành sửa đổi.

Nhanh chóng có phiên bản thực hiện với các chức năng chính.

Có thể thẩm định tính dùng được của sản phẩm sớm.

b) Nhược điểm

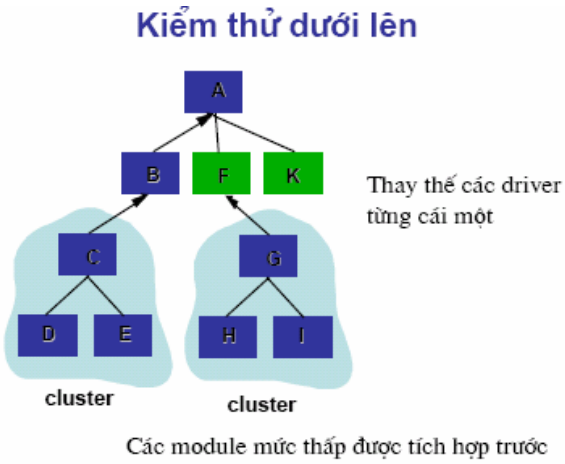
Nhiều môđun cấp thấp rất khó mô phỏng: thao tác với cấu trúc dữ liệu phức tạp, kết quả trả về phức tạp…

4.2.2 Dưới lên

Kiểm ta module lá trước do đó không cần phải viết stub.

Thuật giả của hướng này là:

- Các module cấp thấp được nhóm thành từng nhóm (thực hiện cùng chức năng)
- Viết driver điều khiển tham số nhập xuất.
- Bỏ driver và gắn chùm vào module cao hơn.



a)Ưu điểm

Tránh xây dựng các môđun tạm thời phức tạp.

Tránh sinh các kết quả nhân tạo (nhập từ bàn phím)

Thuận tiện cho phát triển các môđun để dùng lại

b) Nhược điểm

Chậm phát hiện các lỗi kiến trúc

Chậm có phiên bản thực hiện

4.3 Kiểm thử chấp nhận

Kiểm thử chấp nhận được tiến hành bởi khách hàng, còn được gọi là alpha testing. Mục đích là nhằm thẩm định lại xem phần mềm có những sai sót, thiếu sót so với yêu cầu người sử dụng không.

Trong giai đoạn này dữ liệu dùng để kiểm thử do người sử dụng cung cấp.

4.4 Kiểm thử beta

Đây là giai đoạn mở rộng của alpha testing. Công việc kiểm thử được thực hiện bởi một số lượng lớn người sử dụng.

Công việc kiểm thử được tiến hành một cách ngẫu nhiên mà không có sự hướng dẫn của các nhà phát triển. Các lỗi nếu được phát hiện sẽ được thông báo lại cho nhà phát triển.

4.5 Kiểm thử hệ thống

Đến giai đoạn này, công việc kiểm thử được tiến hành với nhìn nhận phần mềm như là một yếu tố trong một hệ thống thông tin phức tạp hoàn chỉnh.

Công việc kiểm thử nhằm kiểm tra khả năng phục hồi sau lỗi, độ an toàn, hiệu năng và giới hạn của phần mềm.

5. Ví dụ minh họa

Ví dụ 1: Phần mềm quản lý thư viện trong giai đoạn kiểm thử, các giai đoạn trước đã được trình bày ở các chương trước

Giai đoạn 6: Kiểm chứng phần mềm hướng đối tượng

- Kiểm tra tính đúng đắn của cá lớp đối tượng

- Chuẩn bị dữ liệu thử nghiệm: Nhập dữ liệu thử nghiệm cho các bảng THU_VIEN, SACH, DOC_GIA, MUON_SACH
- Kiểm tra:

  + Kiểm tra từng lớp đối tượng:

    - Kiểm tra lớp THU_VIEN (Tra cứu độc giả, Tra cứu sách)
    - Kiểm tra lớp DOC_GIA (Lập thẻ, cho mượn sách)

- Kiểm trả lớp SACH (Nhận sách, Trả sách)

+ Kiểm tra phối hợp các lớp đối tượng

- Kiểm tra phối hợp giữa lớp THU_VIEN và lớp DOC_GIA (Lập thẻ và sau đó Tra cứu độc giả)
- Kiểm tra phối hợp giữa lớp THU_VIEN và lớp SACH (Nhận sách và sau đó Tra cứu sách)
- Kiểm tra phối hợp giữa lớp DOC_GIA và lớp SACH (Lập thẻ, Nhận sách, Cho mượn sách và Tra sách)
- Kiểm tra phối hợp giữa 3 lớp THU_VIEN, DOC_GIA và lớp SACH

Xác nhận của khách hàng: Khách hàng sử dụng phần mềm để thực hiện các công việc của mình và so sánh kết quả khi sử dụng phần mềm với kết quả khi thực hiện trong thế giới thực

Ví dụ 2: Minh họa giai đoạn kiểm chứng của phần mềm hỗ trợ giải bài tập phương trình đại số

Giai đoạn 6: Kiểm chứng phần mềm

- Kiểm tra tính đúng đắn của các lớp đối tượng
- Chuẩn bị dữ liệu thử nghiệm: Chuẩn bị các đề tài, đáp án, bài giải, điểm số đã có trong thế giới thực và nhập điểm cho các bảng SACH_BAI_TAP, BAI_TAP, BAI_GIAI, BUOC_GIAI
- Kiểm tra:

+ Kiểm tra từng lớp đối tượng:

- Kiểm tra lớp SACH_BAI_TAP (Tra cứu bài tập)
- Kiểm tra lớp BAI_TAP (Soạn đề, Phát sinh đề, Soạn đáp án, Giải bài tập, Xem đáp án, Chấm điểm

- Ghi chú: Cần phải kiểm tra từng công việc và sau đó kiểm tra phối hợp giữa các công việc

+ Kiểm tra phối hợp các lớp đối tượng: Kiểm tra phối hợp giữa các lớp SACH_BAI_TAP và lớp BAI_TAP (Soạn đề thi và sau đó tra cứu bài tập

- Xác nhận của khách hàng: Khách hàng sử dụng phần mềm để thực hiện các công việc của mình và so sánh kết quả khi sử dụng phần mềm với kết quả khi thực hiện trong thế giới thực.

Chương 8: SƯU LIỆU

6. Tổng quan

Chương này xem xét sưu liệu là một phần của hệ thống phần mềm. Cấu trúc của sưu liệu người dùng và hệ thống được mô tả và điều quan trọng của việc tạo ra những sưu liệu đạt chất lượng phải được nhấn mạnh. Phần cuối của chương này chú ý đến khả năng bảo trì, tính khả chuyển của sưu liệu.

Có hai lớp sưu liệu kết hợp với hệ thống máy tính. Lớp sưu liệu người dùng được mô tả làm thế nào để sử dụng hệ thống và sưu liệu hệ thống được mô tả thiết kế và thực hiện hệ thống

Sưu liệu được cung cấp cùng với hệ thống có thể hữu dụng trong bất kỳ giai đoạn sống của hệ thống

Tất cả sưu liệu cần có chỉ mục hiệu quả. Một chỉ mục tốt, cho phép người dùng tìm kiếm thông tin họ cần, và đó là đặc tính hữu dụng nhất được cung cấp nhưng cũng thường là phần rối nhất trong khi tạo sưu liệu. Một chỉ mục cặn kẻ có thể làm cho sưu liệu được viết tệ có thể sử dụng được, nhưng không có chỉ mục, cho dù sưu liệu viết tốt thì không chắc người đọc sưu liệu có hiệu quả không.

7. Sưu liệu người dùng

Sưu liệu người dùng là sưu liệu mô tả chức năng của hệ thống, mà không tham chiếu đến chức năng được thực hiện như thế nào

Sưu liệu người dùng nên được cấu trúc sao cho không nhất thiết phải đọc hết tất cả sưu liệu trước khi bắt đầu dùng hệ thống. Nó phải được hợp nhất với on-line help nhưng nó quá đơn giản để in văn bản trong help như sưu liệu người dùng

Có 5 loại sưu liệu cho người dùng

- Mô tả chức năng, giải thích hệ thống có thể làm gì
- Sưu liệu cài đặt, giải thích làm thế nào để install hệ thống và chi tiết cho từng cấu hình phần cứng cụ thể
- Giới thiệu, giải thích thuật ngữ đơn giản, và làm thế nào để bắt đầu hệ thống
- Tham chiếu, mô tả chi tiết tất cả tiện ích của hệ thống, chúng được sử dụng như thế nào

- Hướng dẫn người quản trị hệ thống (nếu cần), giải thích làm thể nào để ứng xử với những trường phát sinh khi hệ thống đang sử dụng và làm thể nào để thực hiện bảo quản hệ thống như backup hệ thống

7.1. Mô tả chức năng

- Phác thảo yêu cầu của hệ thống.
- Phác thảo mục đích của người thiết kế hệ thống.
- Mô tả hệ thống có thể làm được gì?
- Mô tả hệ thống không thể làm được gì?
- Giới thiệu ví dụ minh họa nhỏ bất kỳ chỗ nào có thể.
- Vẽ sơ đồ thì tốt nhất.

7.2. Bảng Giới thiệu

- Cung cấp cái nhìn tổng quan của hệ thống.
- Cho phép người dùng quyết định nếu hệ thống phù hợp với nhu cầu của họ.
- Trình bày giới thiệu thông tin đối với hệ thống.
- Mô tả làm thế nào để bắt đầu với hệ thống và làm thế nào người thực hiện sử dụng những tiện ích chung của hệ thống.
- Bảo người dùng hệ thống làm thể nào để tránh những rắc rối khi họ làm sai.

7.3. Bảng tham khảo

- Bảng tham khảo hệ thống là Sưu liệu được định nghĩa cho cách sử dụng hệ thống.
- Bảng tham khảo nên hoàn chỉnh.
- Kỹ thuật mô tả chuẩn nên được dùng để đảm bảo độ hoàn chỉnh đạt được.
- Người viết bảng này có thể giả sử:
    - Người đọc quen với cả mô tả hệ thống và phần giới thiệu.
    - Người đọc dùng vài hệ thống và hiểu được khái niệm và thuật ngữ.
- Phần tham khảo hệ thống cũng nên mô tả:
    - Những báo cáo lỗi phát sinh trong hệ thống.
    - Những tình huống lỗi phát sinh, nếu phù hợp, hướng người dùng đến những mô tả tiện ích đã gây ra lỗi.
- Chỉ mục cặn kẽ đặc biệt quan trọng trong phần Sưu liệu.

7.4. Sưu liệu cài đặt

Sưu liệu cài đặt nên được cung cấp đầy đủ chi tiết làm thế nào để install hệ thống trong mô trường cụ thể.

Phải bao gồm mô tả thiết bị có thể đọc của máy mà hệ thống cung cấp như định dạng, mã ký tự dùng, làm thế nào thông tin được viết, và những tập tin được tạo của hệ thống.

Sưu liệu này gồm các mô tả:

- Cấu hình tối thiểu đòi hỏi để có thể chạy của hệ thống.

- Tập tin cố định phải được thiết lập.

- Làm thế nào để bắt đầu của hệ thống.

- Những tập tin phụ thuộc cấu hình phải thay đổi để thích ứng với hệ thống đối với hệ thống máy chủ cụ thể.

Hướng dẫn cho quản trị hệ thống (cho hệ thống đòi hỏi người theo dõi tương tác)

- Mô tả những thông điệp phát sinh ở màn hình hệ thống, và làm thể nào để ứng phó với những thông điệp này
- Giải thích những nhiệm vụ của người theo dõi trong duy trì phần cứng đó

Những tài liều dễ đế đọc khác

- Danh sách các tham khảo nhanh sẵn có của tiện ích của hệ thống và làm thể nào để sử dụng chúng
- Hệ thống on-line help

8. Sưu liệu hệ thống

Sưu liệu hệ thống chứa tất cả những sưu liệu mô tả quá trình thực hiện của hệ thông từ những sưu liệu đặc tả đến bản kế hoạch test cuối cùng

-Tài liệu mô tả thiết kế

- Sưu liệu mô tả thực hiện

- Sưu liệu mô tả việc kiểm thử

Sưu liệu hệ thống thì cần thiết để hiểu và bảo trì hệ thống phần mềm.

Sưu liệu nên được cấu trúc và được mô tả tổng quan hướng người đọc đến mô tả hình thức và chi tiết với mỗi khía cạnh của hệ thống.

Một trong những khó khăn của sưu liệu hệ thống là duy trì tính kiên định qua những sưu liệu khác mô tả hệ thống. Lưu vết những thay đổi, cân nhắc những sưu liệu nên được thay thế dưới những kiểm soát của hệ thống quản lý cấu hình.

Những thành phần của sưu liệu hệ thống:

1.Định nghĩa và đặc tả yêu cầu và kết hợp.

2.Trình bày đặc tả tất cả hệ thống làm thế nào những yêu cầu được phân rã thành những nhóm các chương trình tương tác với nhau. Sưu liệu này không được yêu cầu khi hệ thống được thực hiện chỉ với chương trình đơn lẻ.

3.Mỗi chương trình của hệ thống, một mô tả làm thế nào chương trình được phân rã thành những thành phần và khẳng định của những đặc tả của thành phần.

4.Mỗi đơn vị, một mô tả của những thao tác. Điều này không cần mở rộng để mô tả hoạt động của chương trình.

5.Mô tả kế hoạch kiểm thử (test plan) chi tiết làm thế nào mỗi đơn vị chương trình được kiểm thử.

6.Một kế hoạch kiểm thử chỉ ra những kiểm thử hợp nhất như là kiểm tra tất cả đơn vị chương trình kết hợp với nhau được thực hiện.

7.Một kế hoạch kiểm thử được chấp nhận, vạch ra sự nối kết những người dùng hệ thống. Tài liệu này nên mô tả những kiểm thử phải được thõa mãn trước khi hệ thống được chấp nhận.

9. Chất lượng của sưu liệu

Chất lượng của sưu liệu quan trọng như chất lượng của chương trình.

Khi không có thông tin làm thế nào sử dụng hệ thống và làm thế nào để hiểu nó thì những tiện ích của hệ thống sẽ bị giảm chất lượng.

Tạo những sưu liệu tốt không dễ dàng không rẻ và tiến trình cũng khó như tạo một chương trình tốt.

Tiêu chuẩn sưu liệu nên được mô tả chính xác như sưu liệu bao gồm những gì và nên mô tả hệ thống các ký hiệu dùng trong sưu liệu.

Với một tổ chức, cần thiết lập một chuẩn cho sưu liệu và yêu cầu tất cả các sưu liệu phải tuân thủ theo những định dạng đó.

Những tiêu chuẩn của sưu liệu có thể bao gồm:

+ Mô tả định dạng trước được chấp bởi tất cả tài liệu
+ Đánh số trang và cách thức ghi chú trang.
+ Phương thức tham khảo tài liệu khác

+ Số đề mục và đề mục con

Phong cách viết là yếu tố nền tảng ảnh hưởng đến chất lượng của sưu liệu và đó là khả năng của người viết để xây dưng một cách rõ ràng kỹ thuật soạn thảo chính xác. Một số cách viết nên tránh như dùng câu quá dài, mô tả phức tạp, lặp lại, thông tin tham chiếu chỉ toàn là số không những chi tiết gơi nhớ cho người đọc v.v

10. Bảotrì sưu liệu

Bởi vì hệ thống phần mềm được cập nhật, sưu liệu kết hợp với hệ thống cũng phải được cập nhật tướng ứng với những thay đổi của hệ thống.

Tất cả những sưu liệu kết hợp nên được cập nhật khi một thay đổi được làm bởi chương trình. Giả sử những thay đổi này được nhìn thấy bởi người dùng, chỉ mô tả thực hiện hệ thống cần phải thay đổi. Nếu hệ thống thay đổi nhiều hơn sự chính xác của lỗi chương trình thì điều này có nghĩa là xem xét lại sưu liệu thiết kế và kiểm thử và có thể sưu liệu ở thiết kế mức cao mô tả đặc tả và yêu cầu.

Một trong những vấn đề chính của bảo trì sưu liệu là lưu những thể hiện khác nhau của hệ thống từng bước với nhau. Giải pháp tốt nhất cho vấn đề này là hỗ trợ bảo trì sưu liệu với công cụ phần mềm mà ghi nhận mối liên hệ sưu liệu, nhắc nhỡ những kỹ sư phần mềm khi thay đổi một sưu liệu có tác động đến sưu liệu khác, và ghi nhận những thay đổi trong sưu liệu

Nếu sự thay đổi của hệ thống tác động giao diện người dùng một cách trực tiếp hoặc thêm mới một tiện ích hoặc mở rộng tiện ích tồn tại.

11. Các mẫu sưu liệu cho qui trình làm phần mềm

11.1. Xác định yêu cầu (SRS)

Software Requirements Specifications (w/o Use Cases)

Chuẩn IEE 830-1984

1. Giới thiệu

    1.1 Mục đích

    1.2 Phạm vi

    1.3 Định nghĩa (định nghĩa, từ viết tắt)

    1.4 Tài liệu tham khảo

1.5 Mô tả cấu trúc tài liệu

2. Mô tả chung

2.1 Tổng quan về sản phẩm

2.2 Chức năng sản phẩm

2.3 Đối tượng người dùng

2.4 Ràng buộc tổng thể

2.5 Giả thiết và sự lệ thuộc

3. Yêu cầu chi tiết

3.1 Yêu cầu chức năng

3.1.1 Yêu cầu chức năng 1

3.1.1.1 Giới thiệu

3.1.1.2 Dữ liệu vào

3.1.1.3 Xử lý

3.1.1.4 Kết quả

3.1.2 Yêu cầu chức năng 2

3.1.n Yêu cầu chức năng n

…

b. Thiết kế

Sưu liệu cho giai đoạn thiết kế có các mẫu thiết kế sau:

- Thiết kế cơ sở dữ liệu (Database Design)
- Thiết kế ràng buộc (Design Criteria)
- Sưu liệu kiến trúc phần mềm (Software Architecture Document)
- Thiết kế Thành phần (Components Design)

11.2. Mô tả thiết kế phần mềm (SDD)

Software Design Descriptions Chuẩn IEEE 1016-1998

1. Introduction (Giới thiệu)

- Purpose (mục đích)

- Scope (Phạm vi)
- Definitions, acronyms, and abbreviations (Định nghĩa, viết tắt)

2 References (Tham khảo)

3 Decomposition description (Mô tả phân rã)

4 Dependency description (Mô tả phụ thuộc)

5 Interface description (mô tả giao diện)

6 Detailed design (thiết kế chi tiết)

11.3. System Design Rationale Document (SDRD)

1. Introduction (Giới thiệu)

    1.1 Purpose of the document (Mục đích của sưu liệu)

    1.2 Design goals (Mục tiêu của thiết kế đạt được)

    1.3 Definitions, acronyms, and abbreviations

    1.4 References (Tham khảo)

    1.5 Overview (Tổng quan)

2. Rationale for Current Software Architecture

3. Rationale for Proposed Software Architecture

    3.1 Overview (Tổng quan)

    3.2 Rationale for Subsystem decomposition

    3.3 Rationale for Hardware/software mapping

    3.4 Rationale for Persistent data management

    3.5 Rationale for Access control and security

    3.6 Rationale for Global software control

    3.7 Rationale for Boundary conditions

4. Subsystem Services

Glossary

Phụ Lục A

1. Câu hỏi lý thuyết

Chương (1->4)

1. Trình bày sự khác biệt của giai đoạn thiết kế trong các qui trình khác nhau
2. Trình bày sự khác biệt của giai đoạn lập trình trong các qui trình khác nhau
3. Khi tiến hành thực hiện phần mềm qua các giai đoạn (trong qui trình 5 giai đoạn) có thể phát sinh lỗi trong một giai đoạn nào đó (kết quả chuyển giao không chính xác, thiếu sót, v.v…). Theo các anh chị lỗi (nếu phát sinh) của giai đoạn nào là nghiêm trọng nhất
4. Theo các anh chị trong các giai đoạn của qui trình công nghệ phần mềm

- Giai đoạn nào là quan trọng nhất (tại sao)
- Giai đoạn nào dễ thực hiện nhất (tại sao)
- Giai đoạn nào là tốn nhiều thời gian và chi phí nhất (tại sao)
- Giai đoạn nào là có thể bỏ qua (trong trường hợp nào và tại sao)

Chương 2 (5-10)

5. Cho biết sự khác biệt cơ bản giữa yêu cầu chức năng (yêu cầu nghiệp vụ, yêu cầu hệ thống) và phi chức năng (yêu cầu chất lượng). Theo anh chị thì loại yêu cầu nào là quan trọng hơn
6. Xác định tất cả các yêu cầu chức năng hệ thống có thể có trong các phần mềm sau (chi tiết về qui định, biểu mẫu liên quan có trong mô tả của đề tài)
    1) Phần mềm quản lý bán sách
    2) Phần mềm quản lý học sinh trường cấp 3
    3) Phần mềm đánh cờ gánh
    4) Phần mềm hỗ trợ giải bài tập phương trình đại số
    5) Phần mềm quản lý giải vô địch bóng đá quốc giá
7. Nhận xét về phát biểu sau “Mọi phần mềm đều có yêu cầu về tính tiện dụng”.

- Nếu đúng: giải thích
- Nếu sai: giải thích và ví dụ minh họa

8. Nhận xét về phát biểu sau: “Mọi phần mềm đề có yêu cầu về tính hiệu quả

- Nếu đúng: giải thích

- Nếu sai: giải thích và cho ví dụ minh họa

9. Nhận xét về phát biểu sau: "Mọi phần mềm đề có yêu cầu chức năng hệ thống

- Nếu đúng: giải thích
- Nếu sai: giải thích và cho ví dụ minh họa

Chương (3-14)

10. Nhận xét về phát biểu sau "Việc mô hình hóa yêu cầu không cung cấp thêm thông tin mới về yêu cầu của phần mềm mà chỉ giúp trình bày lại yêu cầu của phần mềm dưới dạng trực quan hơn

- Nếu đúng: giải thích
- Nếu sai: giải thích và cho ví dụ minh họa

11. Nếu không thực hiện qua bước mô hình hóa yêu cầu thì việc lập mô hình đối tượng sẽ có các khó khăn gì? tại sao?

12. Cho biết các kết quả của việc mô hình hóa yêu cầu có được sử dụng trong bước thiết kế giao diện của đối tượng hay không ?

- Nếu đúng: giải thích
- Nếu sai: giải thích và cho ví dụ minh họa

13. Cho biết các kết quả của việc mô hình hóa yêu cầu có được sử dụng trong bước xác định thuộc tính đối tượng (giai đoạn thiết kế) hay không

- Nếu đúng: giải thích
- Nếu sai: giải thích và cho ví dụ minh họa

14. Cho biết các kết quả của việc mô hình hóa yêu cầu có được sử dụng trong bước xác định hàm xử lý của đối tượng (giai đoạn thiết kế) hay không

- Nếu đúng: giải thích
- Nếu sai: giải thích và cho ví dụ minh họa

2. Nội dung và yêu cầu bài tập

2.1 Quản lý thuê bao điện thoại

Lưu trữ: Các thông tin về

- Các hợp đồng thuê bao điện thoại (Khách hàng, loại thuê bao, máy điện thoại)
- Các cuộc gọi (Máy điện thoại, Ngày, Giờ, Thời gian, Nơi gọi đến).

Tính toán:

- Số tiền phải trả của từng máy điện thoại trong từng tháng:
    - Tiền thuê bao hàng tháng (phụ thuộc vào từng loại thuê bao với các định mức riêng).
    - Tiền cước phí trả thêm (hụ thuộc vào thời gian gọi, số phút gọi, nơi gọi đến)
- Tính công nợ khách hàng đối với các khách hàng chưa thanh toán tiền điện thoại.

Kết xuất:

- Hóa đơn tính tiền điện thoại cho từng khách hàng trong từng tháng.
- Danh sách khách hàng chưa thanh toán tiền điện thoại.
- Thống kê về nơi gọi đến, thời điểm gọi theo từng khu vực trong từng tháng.

2.2 Quản lý học sinh trường phổ thông trung học

Lưu trữ: Các thông tin về

- Học sinh: Họ, tên, lớp, ngày sinh, giới tính, địa chỉ, thành phần, kết quả học tập, điểm danh.

Tra cứu: Thông tin về học sinh

Tính toán:

- Điểm trung bình từng môn học theo từng học kỳ: Tính theo các điểm của từng hình thức kiểm tra (15 phút: hệ số 1, 1 tiết hệ số 2, thi học kỳ: hệ số 3)
- Điểm trung bình học kỳ 1, học kỳ 2, cả năm (học kỳ 1 hệ số 1, học kỳ 2 hệ số 2) .
- Xếp loại: Xuất sắc nếu điểm trung bình niên khóa ≥ 9.0 và không có môn nào có điểm trung bình dưới 7.5. Tiên tiến nếu điểm trung bình niên khóa ≥ 7.5 và không có môn nào có điểm trung bình dưới 6.0. Đạt yêu cầu nếu điểm trung bình niên khóa ≥ 5 và không có môn nào có điểm trung bình dưới 5. Không đạt yêu cầu nếu có ít nhất 1 môn dưới 5.

Ghi chú: Nếu tổng số ngày vắng vượt quá 20 sẽ bị xếp vào loại không đạt yêu cầu. Nếu số ngày vắng vượt quá 10 hay số ngày vắng không phép vượt quá 5 thì sẽ bị hạ xuống một bậc (chỉ áp dụng với loại xuất sắc và tiên tiến).

Kết xuất:

- Danh sách học sinh theo từng lớp.

- Phiếu điểm cho mỗi học sinh.
- Bảng điểm các môn và bảng điểm tổng kết cho từng lớp.
- Thống kê về xếp loại học sinh của tòan trường trong 1 niên khóa.

2.2 Quản lý các tài khoản trong ngân hàng

Lưu trữ:

- Các tài khoản: Khách hàng, loại tài khỏan, số tiền, loại tiền, ngày gởi, tình trạng
- Quá trình gửi và rút tài khỏan: Khách hàng, ngày số tiền, hình thức.
- Các qui định về lãi suất và tỷ giá.

Tra cứu: Tài khoản theo các tiêu chuẩn

- Mã số
- Khách hàng
- Loại tài khoản
- Ngày mở, ngày đóng.

Tính toán:

- Lãi suất cho từng tài khoản khi đến kỳ hạn hay khi khách hàng rút trước kỳ hạn (chỉ được không kỳ hạn).

Kết suất:

- Danh sách các biến động trên 1 tài khoản
- Danh sách tài khoản cùng số dư hiện tại theo từng loại tài khoản.
- Tình hình gửi, rút tiền theo từng loại tài khoản.
- Số dư của ngân hàng theo từng ngày của tháng.

2.3 Theo dõi kế hoạch sản lượng cao su

Lưu trữ: Các thông tin về

- Nông trường: Tên, diện tích các lô cao su theo từng năm.
- Sản lượng kế hoạch theo tháng, năm của từng loại mủ.
- Sản lượng thực tế theo ngày của từng loại mủ.

Tính toán:

- Tỷ lệ đạt của từng loại mủ theo từng nông trường theo kế hoạch.
- Kế hoạch dự kiến cho năm tới.

Kết xuất:

- Báo cáo nhanh hàng ngày.
- Báo cáo tháng.
- Kế hoạch năm cho từng nông trường cho từng loại mủ.

2.4 Quản lý giải vô địch bóng đá

Lưu trữ: Các thông tin về

- Các đội bóng tham gia giải: Tên đội bóng, tên huấn luyện viên, các cầu thủ, sân nhà.
- Lịch thi đấu: đội tham dự, sân, thời gian
- Kết quả các trận đấu: Trọng tài, tỷ số, khán giả, các cầu thủ ra sân của 2 đội cùng vị trí tương ứng, việc ghi bàn, phạt thẻ.

Tra cứu: Cầu thủ, đội bóng

Tính toán:

- Tính điểm cho từng đội: mỗi trận thắng được 3 điểm, mỗi trận hòa được 1 điểm, mỗi trận thua được 0 điểm.
- Xếp hạng cho từng đội: Dựa vào các tiêu chuẩn: tổng số điểm, tổng số bàn thắng, hiệu số, đối kháng trực tiếp, bốc thăm.

Kết xuất:

- Danh sách các cầu thủ theo từng đội, vị trí.
- Lịch thi đấu.
- Bảng xếp hạng các đội bóng.
- Tổng kết việc ghi bàn của giải.
- Tình hình phạt thẻ các đội bóng.

2.4 Thi trắc nghiệm trên máy tính

Lưu trữ: Các thông tin về

- Thí sinh dự thi: Họ và tên, môn thi, ngày thi, địa chỉ, đề thi, bài làm, phòng thi.
- Câu hỏi trắc nghiệm: Nội dung câu hỏi, các cầu trả lời có thể có, đáp án, mức độ khó, thang điểm, môn tương ứng.

Tính toán:

- Phát sinh các đề thi tương đương cho một đề thi đã chọn cho một môn thi nào đó (đề thi tương có cùng các câu hỏi trắc nghiệm nhưng có số thứ tự khác nhau và trật tự các câu trả lời cũng khác nhau).
- Tính điểm thi cho từng thí sinh: Tổng điểm các câu hỏi với thang điểm tương ứng.

Kết xuất:

- Danh sách các thí sinh theo từng phòng thi.
- Đề thi.
- Bài làm của từng thí sinh cùng với điểm số.
- Danh sách kết quả thi của mỗi môn thi.
- Thống kê kết quả thi theo từng mức theo từng môn thi.
- Thống kê kết quả thi theo từng câu hỏi.

2.5 Quản lý trung tâm giới thiệu việc làm sinh viên

Lưu trữ: Các thông tin về

- Sinh viên đăng ký tìm việc: Họ và tên, ngày sinh, địa chỉ, tình hình sức khỏe, quá trình học tập và bằng cấp, các công việc có thể đảm nhận, các yêu cầu khi tìm việc.
- Đơn vị đăng ký tìm người: Tên, địa chỉ, người đại diện, các công việc cùng yêu cầu tuyển dụng.
- Giới thiệu việc làm: Sinh viên, đơn vị, công việc, tình trạng.

Tra cứu:

Sinh viên tra cứu công việc

- Loại công việc.
- Mức lương.
- Hình thức làm việc.
- Nơi làm việc

Đơn vị tuyển dụng tra cứu các sinh viên

- Bằng cấp chuyên môn.
- Sức khỏe.
- Phương tiện làm việc.

Tính toán:

- Các công việc thích hợp cho sinh viên đăng ký làm việc.

- Các sinh viên thích hợp cho công việc cần tuyển dụng của 1 đơn vị.

Kết xuất:

- Danh sách sinh viên đăng ký theo từng công việc.
- Danh sách số lượng sinh viên đăng ký theo từng loại công việc.
- Danh sách các đơn vị tuyển dụng theo từng công việc.
- Danh sách số lượng đơn vị tuyển dụng theo từng công việc.
- Thống kê tình hình giới thiệu việc làm thực hiện trong năm.

c. Phần mềm quản lý bán sách

Khảo sát thực tế và rút ra yêu cầu cần phải làm cho đề tài

d. Phần mềm quản lý bán vé chuyến bay

Khảo sát thực tế và rút ra yêu cầu cần phải làm cho đề tài

e. Phần mềm quản lý phòng mạch

Khảo sát thực tế và rút ra yêu cầu cấn phải làm cho đề tài

3. Bài tập nâng cao

3.1 Đăng ký môn học và học phí

Một trường đại học có nhu cầu tin học hóa khâu quản lý việc đăng ký môn học và học phí của sinh viên. Một sinh viên sau khi hoàn thành thủ tục nhập học phải cho biết họ và tên, ngày sinh, giới tính, quê quán gồm tên huyện và tên tỉnh. Nếu sinh viên thuộc đối tượng (con liệt sỹ, con thương binh, con gia đình có công với nước, vùng sâu, vùng xa,) thì phải có xác nhận của địa phương. Mỗi đối tượng có một tỷ lệ tương ứng về việc giảm học phí. Để thuận tiện trong việc quản lý người ta gán cho mỗi sinh viên một mã số gọi là mã số sinh viên, mã số này là duy nhất, không thay đổi trong suốt quá trình sinh viên học tại trường. Căn cứ ngành học mà sinh viên thi đậu vào mà sinh viên đó sẽ thuộc sự quản lý của một khoa nào đó: nghĩa là mỗi sinh viên thuộc một ngành, và một khoa có thể gồm nhiều ngành học khác nhau; dĩ nhiên không tồn tại một ngành thuộc sự quản lý của hai khoa khác nhau.

Vào đầu học kỳ mới sinh viên đến phòng Giáo vụ đăng ký các môn học. Việc đăng ký môn học được thể hiện qua một phiếu đăng ký. Trên phiếu đăng ký có một số phiếu, thông tin về sinh viên (mã số, họ và tên), ngày đăng ký, học kỳ và niên khóa đăng ký. Một phiếu đăng ký có thể có nhiều môn học (mã môn, tên môn và số đơn vị học trình tương ứng của môn đó).

Tất nhiên là các môn học đó sẽ được dạy trong học kỳ cho sinh viên đăng ký mà phòng Giáo vụ đã có kế hoạch trong thời khóa biểu đã thông báo cho sinh viên biết trước khi đăng ký.

Mỗi môn học ngoài việc định danh bằng tên còn kèm theo số tín chỉ học trình và được gán cho một mã số môn học. Số tín chỉ của mỗi môn học tùy thuộc vào thời gian giảng dạy (thường 15 tiết lý thuyết hoặc bài tập hay 30 tiết thực hành tương đương 1 tín chỉ). Để đơn giản người ta phân thành hai loại môn: môn lý thuyết (hoặc bài tập) và môn thực hành. Nếu đăng ký môn lý thuyết sinh viên sẽ phải trả 27000 đồng/ tín chỉ, còn với môn thực hành là 37000 đồng/tín chỉ. Có một số môn, muốn đăng ký học, sinh viên phải học và đạt trên điểm trung bình một số môn trước để làm cơ sở cho việc học môn đó (gọi là các môn tiên quyết của môn học đó). Mỗi ngành học bao gồm một hệ thống nhiều môn mà sinh viên thuộc ngành đó phải theo học nằm trong nội dung chương trình giảng dạy của ngành đó; có thể có nhiều môn thuộc chương trình giảng dạy của nhiều ngành học khác nhau. Mỗi học kỳ, căn cứ vào việc đăng ký các môn học và đối tượng của từng sinh viên mà người ta xác định được số tiền học phí mà mỗi sinh viên sẽ phải đóng.

Sau khi đăng ký xong các môn học, sinh viên phải đến Phòng Tài vụ của trường để đóng học phí. Mỗi lần khi một sinh viên đến nộp học phí, một phiếu thu được lập, trên đó ghi nhận mã số sinh viên, ngày lập, số tiền mà sinh viên đóng và được đánh số thứ tự để tiện việc theo dõi. Mỗi phiếu thu chỉ thu tiền học phí của một sinh viên tại một học kỳ. Một phiếu thu được in thành hai liên, một liên gửi cho sinh viên như một biên lai, liên còn lại để lưu. Nhân viên của Phòng Tài vụ lập phiếu phải nhận tiền học phí của sinh viên để cuối buổi nộp cho thủ quỹ. Mỗi học kỳ, nhà trường khống chế thời điểm cuối cùng (một ngày nào đó) mà sinh viên phải hoàn thành thủ tục trên, nếu quá hạn đó phòng Tài vụ khóa sổ không thu nữa, và như vậy những sinh viên không đóng, không kịp đóng hoặc đóng không đủ học phí sẽ không được tham dự kỳ thi cuối học kỳ đó. Mỗi học kỳ, sau khi cho sinh viên đăng ký môn học, để khuyến khích sinh viên đóng học phí sớm nhà trường cũng qui định một ngày mà nếu sinh viên đóng học phí trước ngày đó sẽ được giảm một tỷ lệ nào đó (thường là 5% số tiền học phí mà sinh viên phải đóng cho học kỳ đó). Mỗi học kỳ sinh viên có thể đóng học phí làm nhiều lần tùy theo tình hình tài chính của mình và phải đóng trước ngày hết hạn đóng học phí của học kỳ đó.

Khi hết hạn đóng học phí Phòng Tài vụ sẽ tổng kết số tiền học phí mà mỗi sinh viên đã đóng, kết hợp với số tiền học phí mà sinh viên phải đóng xác định danh sách những sinh viên đang còn nợ học phí của học kỳ đó để gửi cho bộ phận quản lý của Phòng Giáo vụ loại những sinh viên đó ra khỏi danh sách dự thi.

3.2 Quản lý đồ án – Niên luận

Bộ môn Hệ thống thông tin và toán ứng dụng khoa Công Nghệ Thông Tin muốn quản lý tất cả các đồ án - niên luận của sinh viên tin học chính quy cũng như tại chức. Để dễ dàng trong việc quản lý, ngay sau khi vào trường mỗi sinh viên ngoài họ tên, ngày sinh, giới tính đều được gán một mã số gọi là mã số sinh viên. Sinh viên chính quy thuộc sự quản lý của trường còn đối với sinh viên tại chức sẽ thuộc sự quản lý của một đơn vị đào tạo (thường là trung tâm giáo dục thường xuyên) của một tỉnh nào đó.

Trong chương trình đào tạo sinh viên phải thực hiện một số loại đồ án (niên luận 1 - lập trình chuyên ngành, niên luận 2 - lập trình quản lý, niên luận 3 – lập trình ứng dụng, tiểu luận tốt nghiệp, và luận văn tốt nghiệp cho một số sinh viên xuất sắc khi ra trường). Mỗi loại đồ án - niên luận có một số đơn vị học trình tương ứng gọi là số tín chỉ.

Theo chương trình học, đến kỳ triển khai đồ án - niên luận bộ môn yêu cầu các giáo viên ra đề tài cho sinh viên chọn. Mỗi một đề tài giáo viên yêu cầu những điều mà sinh viên sẽ phải làm, cung cấp các tài liệu để sinh viên tham khảo. Sau khi giáo viên nộp đề tài bộ môn sẽ gán cho mỗi đề tài một mã số. Việc định danh (đặt tên) do các giáo viên ra đề tài quyết định. Mỗi đề tài chỉ thuộc một loại đồ án - niên luận duy nhất, và được ra bởi ít nhất một giáo viên trong bộ môn.

Mỗi một giáo viên được nhận biết qua mã số giáo viên, họ tên, ngày sinh, phái và một chức danh. Mỗi chức danh có một hệ số chức danh, và căn cứ vào chức danh này để sau này tính tiền cho giáo viên ra đề tài hay giáo viên hướng dẫn đồ án - niên luận.

Đến học kỳ mà sinh viên phải thực hiện loại đồ án nào đó, bộ môn sẽ triển khai việc thực hiện đồ án - niên luận cho sinh viên. Trước hết bộ môn cung cấp danh sách các đề tài mà các giáo viên đã ra thuộc loại đó đểø sinh viên lựa chọn thực hiện. Đối với các loại niên luận, tiểu luận, các sinh viên tự lập nhóm, tối đa hai sinh viên một nhóm, nhóm này chọn làm chung một quyển đồ án và một quyển đồ án như vậy làm về một đề tài duy nhất trong danh sách các đề tài được bộ môn cung cấp. Riêng trường hợp đối với luận văn tốt nghiệp, chỉ có một số sinh viên xuất sắc được chọn và mỗi sinh viên làm một đồ án tốt nghiệp riêng rẽ.

Sau khi sinh viên lựa chọn đề tài, bộ môn sẽ phân công giáo viên hướng dẫn cho từng nhóm sinh viên làm chung một đề tài và viết chung một quyển đồ án - niên luận. Nói chung giáo viên ra đề tài là người hướng dẫn những sinh viên thực hiện đề tài đó, tuy nhiên có khi giáo viên ra đề tài bận đi công tác, bộ môn có thể cử người khác hướng dẫn. Đến hạn sinh viên phải hoàn thành và nộp các quyển đồ án. Quyển đồ án phải được soạn theo mẫu mà bộ môn đã quy định để dễ dàng trong việc quản lý và đánh giá. Cán bộ trực bộ môn phải chịu

trách nhiệm thu nhận các quyển đồ án mà sinh viên nộp. Để đơn giản trong quản lý, mỗi quyển đồ án - niên luận được cán bộ trực bộ môn gán cho một số thứ tự, ghi nhận lại ngày mà sinh viên nộp.

Ngay sau ngày hết hạn nộp trưởng hoặc phó bộ môn sẽ phân công giáo viên đánh giá và chấm điểm cho từng quyển đồ án. Bộ môn cũng yêu cầu các giáo viên nộp kết quả đúng kỳ hạn để tổng kết điểm. Các sinh viên thực hiện chung một đề tài sẽ được chung một điểm kết quả qua sự cho điểm đó. Khi đến hạn, bộ môn sẽ tổng kết điểm, lập danh sách báo cáo cho phòng Giáo vụ.

Cuối học kỳ bộ môn tổng kết số đề tài mà mỗi giáo viên đã ra (mà được sinh viên chọn làm đồ án - niên luận), số đồ án - niên luận mà mỗi giáo viên đã hướng dẫn, đã chấm để làm cơ sở cho việc tính tiền giảng dạy.

3.3 Quản lý cơ sở sản xuất và chất lượng sản phẩm

Chi cục tiêu chuẩn đo lường chất lượng sản phẩm một tỉnh cần quản lý chất lượng các sản phẩm của những cơ sở sản xuất trong tỉnh.

Trên địa bàn tỉnh quản lý có nhiều cơ sở sản xuất. Để thuận tiện trong quản lý người ta gán mỗi cơ sở một mã số cơ sở duy nhất. Mỗi một cơ sở có một địa chỉ, một người chịu trách nhiệm gọi là chủ cơ sở, được biết bằng họ và tên, có thể không có, có một hoặc có một vài số điện thoại để tiện liên hệ.

Cơ sở muốn sản xuất một mặt sản phẩm nào phải đăng ký thông qua một phiếu đăng ký chất lượng cho nó. Một phiếu đăng ký có một số đăng ký hay số thứ tự và chỉ cấp cho một sản phẩm duy nhất, tuy nhiên một cơ sở sản xuất có thể đăng ký nhiều sản phẩm khác nhau. Mỗi phiếu đăng ký có một thời hạn (từ một ngày đến một ngày nào đó) và số lượng đăng ký sẽ sản xuất trong thời hạn đó. Mỗi sản phẩm được gán cho một mã số sản phẩm, một định danh rõ ràng và một đơn vị tính tương ứng. Một sản phẩm thường phải đăng ký nhiều chỉ tiêu, mỗi chỉ tiêu có một đơn vị tính cho chỉ tiêu đó, và khi đăng ký thì chỉ số đăng ký cho chỉ tiêu tương ứng là bao nhiêu.

Trong thời hạn đăng ký, về nguyên tắc sản phẩm đã đăng ký sản xuất được bán trên thị trường phải bảo đảm các chỉ tiêu đã đăng ký. Theo định kỳ hoặc có gì nghi vấn chi cục tiêu chuẩn đo lường chất lượng sản phẩm sẽ bốc mẫu sản phẩm của cơ sở về để kiểm nghiệm, đánh giá. Khi đánh giá xong một phiếu kiểm nghiệm được lập. Một phiếu kiểm nghiệm chỉ kiểm một sản phẩm theo một số chỉ tiêu với chỉ số kiểm nghiệm tương ứng. Hơn nữa một phiếu kiểm nghiệm có một số thứ tự, ngày đánh giá và chỉ dùng cho một cơ sở duy nhất đã

sản xuất sản phẩm đã đăng ký đó. Dựa vào kết quả kiểm nghiệm mà người có trách nhiệm cho đánh giá là đạt hay không đạt chất lượng theo mức đăng ký. Sản phẩm của cơ sở nào không đạt chất lượng sẽ không được phép tiếp tục sản xuất và lưu hành trên thị trường, và bị rút giấy phép kinh doanh. Nếu sản phẩm gây nguy hại cho người dùng thì chủ cơ sở có thể bị truy tố trước pháp luật.

Đến lúc nào đó cục tiêu chuẩn đo lường chất lượng sản phẩm muốn biết các sản phẩm của cơ sở nào hết thời hạn đăng ký, những sản phẩm nào không đạt chất lượng, vv...

3.4 Quản lý lương sản phẩm

Một công ty sản xuất muốn quản lý tiền lương của tất cả các nhân viên. Các nhân viên thuộc hai loại: nhân viên hành chánh và công nhân. Mỗi một nhân viên có một mã số, họ tên, phái, ngày sinh, và ngày bắt đầu tham gia công tác. Mỗi nhân viên sẽ thuộc một đơn vị quản lý nào đó.

Đối với công nhân hưởng lương sản phẩm. Các sản phẩm này thường được các công ty khác đặt hàng thông qua một hợp đồng với một số lượng tương ứng cùng những yêu cầu về kỹ thuật và thẩm mỹ kèm theo. Một sản phẩm có một mã số và mang một tên để gọi và đơn vị tính của nó.

Các hợp đồng được đánh số thứ tự, tên hợp đồng, ngày bắt đầu và ngày kết thúc. Một hợp đồng ít nhất về một sản phẩm, nếu liên qua đến nhiều sản phẩm thì tất cả các sản phẩm này đều cùng kết thúc cùng một thời điểm ghi trên hợp đồng để giao hàng và thanh lý hợp đồng.

Quá trình sản xuất một sản phẩm gồm nhiều công đoạn tùy theo sản phẩm. Do đặc tính kỷ thuật, thẩm mỹ và môi trường làm việc mà mỗi công đoạn được trả một đơn giá tương ứng. Các công đoạn sản xuất một sản phẩm được gọi bằng tên công đoạn và thường được đánh số thứ tự.

Hàng ngày, bộ phận quản lý sẽ ghi nhận kết quả làm việc của công nhân ngày hôm trướùc do đơn vị sản xuất báo lên. Kết quả làm việc của mỗi công nhân trong ngày thể hiện việc công nhân đó thực hiện được những công đoạn nào của sản phẩm được hợp đồng với số lượng tương ứng của công đoạn đó là bao nhiêu trong ca làm việc nào. Làm việc ở ca 3 hoặc các ca của ngày chủ nhật được hưởng thêm một hệ số cao hơn làm việc các ca khác trong ngày làm việc bình thường. Kết quả này sẽ xác định thu nhập của công nhân trong ngày hôm đó.

Đối với việc tính lương cho nhân viên hành chánh căn cứ vào hệ số lương và số ngày làm việc trong tháng của người đó. Nếu nghỉ có lý do (bệnh đột xuất, thai sản, ...) sẽ được hưởng tiền bảo hiểm xã hội tùy theo số ngày nghỉ có lý do trong tháng. Nếu nghỉ không lý do thì không được tính lương. Hệ số lương thường căn cứ vào trình độ chuyên môn, trình độ ngoại ngữ, và thâm niên công tác và do lãnh đạo công ty xem xét và quyết định. Đối với những người có đảm trách chức vụ thì được hưởng phụ cấp chức vụ tùy theo đặc thù của chức vụ. Do nhu cầu của công tác, có thể các nhân viên hành chánh có thể làm việc ngoài giờ. Bộ phân theo dõi lương sẽ tổng kết số buổi làm thêm ngoài giờ của từng nhân viên trong tháng để tính lương ngoài giờ cho nhân viên.

3.5 Quản lý công tác thực hành tin học

Khoa Công nghệ thông tin muốn quản lý công tác thực hành tin học của các phòng thực hành. Khoa có nhiều phòng máy tính phục vụ các môn học thực hành và làm niên luận, luận văn cho sinh viên. Mỗi phòng có số phòng, cùng hệ thống các máy tính trong đó. Các máy tính được đánh số và có thể có cấu hình (các phụ tùng: Mainboard, Ram, Harddisk, ... với đặc tính kỷ thuật liên quan) khác nhau. Mỗi phòng thực hành do một cán bộ phụ trách. Người ta quan tâm đến họ tên, phái, ngày sinh, địa chỉ của cán bộ và để cho đơn giản người ta cho mỗi cán bộ một mã số để phân biệt.

Dựa vào việc đăng ký các môn thực hành của sinh viên vào đầu học kỳ mà phòng Giáo vụ chuyển danh sách cho, trợ lý giáo vụ của khoa sẽ phân thành các nhóm thực hành. Các sinh viên cùng một nhóm sẽ có cùng một lịch thực hành. Lịch thực hành của mỗi môn học tại một học kỳ được bố trí thành các buổi tại các phòng thực hành. Mỗi buổi thực hành chỉ dành cho một môn thực hành của một nhóm nào đó. Phòng Giáo vụ dựa vào việc đăng ký môn học đầu học kỳ của sinh viên mà cung cấp danh sách các nhóm thực hành cho từng môn, căn cứ vào đó cán bộ coi thi thực hành điểm danh và kiểm tra.

Khi tiến hành mỗi buổi thực hành, cán bộ phụ trách sẽ bố trí vị trí của sinh viên (ngồi vào máy nào của phòng máy). Nói chung sinh viên tham dự các buổi thực hành theo lịch thực hành mà trợ lý giáo vụ hay trưởng phòng thí nghiệm đã sắp xếp. Cũng như đối với cán bộ, người ta quan tâm đến họ tên, phái, ngày sinh, địa chỉ của sinh viên và để cho đơn giản người ta cho mỗi sinh viên một mã số gọi lã mã sinh viên để phân biệt. Những thông tin về sinh viên được ghi nhận tại Phòng Giáo vụ khi sinh viên nhập học sau khi trúng tuyển qua kỳ tuyển sinh.

Một buổi thực hành tại một phòng máy chỉ thực hành một môn học nào đó. Chú ý là một ngày làm việc có thể có 3 buổi thực hành (sáng, chiều, và tối). Sau khi trợ lý giáo vụ công

bố lịch thực hành, bộ môn sẽ phân công cán bộ giảng dạy hướng dẫn sinh viên thực hiện các bài tập cho thực hành này. Cùng một môn nhưng có thể có nhiều cán bộ coi thực hành tại một buổi thi.

Xong mỗi đợt thực hành cán bộ phụ trách phòng thực hành kiểm tra sinh viên nào đủ tiêu chuẩn thi, sinh viên nào không tham dự đầy đủ số buổi thực hành sẽ bị cấm thi.

Cuối học kỳ bộ môn sẽ tổng kết số giờ coi thực hành của từng cán bộ để giáo vụ khoa tổng hợp công tác giảng dạy.

3.6 Công tác tổ chức thi học kỳ

Trợ lý giáo vụ khoa công nghệ thông tin muốn tin học hóa việc tổ chức thi cử ở các đơn vị đào tạo mà khoa phụ trách. Hàng năm khoa phải tổ chức nhiều đợt thi cho sinh viên các đơn vị đào tạo: thường là thi cuối học kỳ của mỗi năm học, mỗi học kỳ có hai lần thi, mỗi lần thi lại tổ chức nhiều đợt khác nhau do có thể có nhiều môn thi trong một lần thi. Tùy từng lần thi có thể có 1, 2 hoặc thậm chí 3 đợt cho một lần thi.

Trước hết trợ lý giáo vụ phải dự kiến thời gian tổ chức cho mỗi đợt trong mỗi lần của kỳ thi, danh sách các môn thi, số sinh viên tham dự cho các lớp tương ứng tại mỗi đơn vị đào tạo. Mỗi đợt thi tại các đơn vị đào tạo khác nhau có hệ thống các môn thi khác nhau. Có một số đơn vị đào tạo do điều kiện khách quan có thể không có một số khóa học nào đó. Nếu tất cả sinh viên của lớp tại một đơn vị đào tạo đã đạt lần thi thứ nhất thì không cần tổ chức thi môn đó cho lớp đó trong lần 2; nhưng cũng tại đợt thi của lần thi đó ở đơn vị đào tạo khác lại phải bố trí do có sinh viên chưa đạt kết quả trong lần thi trước. Số sinh viên trong lần thi thứ nhất thường là tất cả các sinh viên của lớp đủ điều kiện dự thi.

Nói chung một đợt thi có lịch thi thống nhất áp dụng cho một số đơn vị đào tạo nào đó. Tuy nhiên do hoàn cảnh và nhiều nguyên nhân chủ quan cũng như khách quan, nên có thể thời gian thi áp dụng cho đơn vị đào tạo này khác với thời gian thi áp dụng cho đơn vị đào tạo kia là điều có thể xảy ra. Việc dự kiến trước thời gian tổ chức thi nhằm có kế hoạch trước để các bộ môn phân công cán bộ ra đề thi. Cùng một môn thi trong một đợt thi, nhưng ở những đơn vị đào tạo khác nhau có thể do những giáo viên khác nhau ra đề. Giáo viên ra đề tự quyết định thời gian làm bài của thí sinh cho đề mình ra. Giáo viên ra đề phải hoàn thành trước thời gian bắt đầu tổ chức đợt thi một tuần.

Sau đó trợ lý giáo vụ khoa làm lịch thi và cử các cán bộ làm giám sát đợt thi tại các đơn vị đào tạo có tổ chức thi. Có thể có nhiều cán bộ tham gia làm giám sát đợt thi tại cùng một đơn vị đào tạo (thường đầu tuần một người, cuối tuần lại người khác và sang tuần sau lại có thể là cán bộ khác nữa) do một đợt thi có thể kéo dài đến vài tuần. Khi làm lịch thi trợ lý giáo

vụ khoa dự kiến ngày, giờ bắt đầu cho mỗi môn thi tại từng đơn vị đào tạo. Người có trách nhiệm tại các đơn vị đào tạo chịu trách nhiệm phân công cán bộ coi thi cho mỗi môn. Theo qui định thường có 2 cán bộ coi thi cho mỗi môn. Tùy theo thời gian làm bài của sinh viên đối với mỗi môn thi đã được qui định bởi giáo viên ra đề mà tính tiền coi thi cho cán bộ coi thi môn đó. Thời gian thi càng dài thì tiền coi thi càng nhiều.

Mỗi đợt thi, sau khi phòng đào tạo xử lý bài thi (cắt phách) xong các trưởng bộ môn phân công cán bộ đến phòng đào tạo nhận bài thi về chấm. Chú ý là trong cùng một đợt thi, cùng một môn thi, bài thi các lớp tại các đơn vị đào tạo khác nhau, có thể do các cán bộ khác nhau chấm. Sau khi chấm xong các cán bộ chấm thi nộp kết quả cho phòng đào tạo và báo số sinh viên còn thiếu điểm để trợ lý giáo vụ khoa ghi nhận, làm cơ sở xác định số lượng đề cần photocopy cho việc ra đề lần sau.

Mỗi đợt thi tại mỗi đơn vị đào tạo, trợ lý giáo vụ khoa cần biết các thông tin cụ thể như: ngày, giờ, môn, lớp. thời gian thi, cán bộ ra đề, cán bộ chấm, số sinh viên còn nợ sau khi chấm, các cán bộ coi thi tương ứng đối với từng môn thi.

4. Biểu mẫu thực hiện đồ án môn học

I. Yêu cầu chung

Mỗi sinh viên đăng ký thực hiện phần mềm. Kết quả gồm báo cáo viết, đĩa/CD (chương trình nguồn, EXE, báo cáo viết.

II. Cấu trúc báo cáo viết

1. Hiện trạng và yêu cầu

Hiện trạng

- Giới thiệu về thới giới thực liên quan
- Mô tả qui trình các công việc liên quan đến đề tài
- Mô tả các mẩu biểu có liên quan
- Mô tả các qui định ràng buộc có liên quan
- Mô tả các qui định công thức tính có liên quan

Yêu cầu

Danh sách các công việc sẽ được hỗ trợ thực hiện trên máy tính (dựa theo tóm tắt yêu cầu đã cho)

2. Mô hình hóa yêu cầu

Mô hình luồng dữ liệu theo yêu cầu

- Sơ đồ luồng dữ liệu cho từng yêu cầu
- Mô tả chi tiết cho từng sơ đồ

Mô hình luồng dữ liệu chung cho toàn bộ hệ thốn

Sơ đồ luồng dữ liệu chung cho toàn bọê hệ thống

3. Thiết kế phần mềm

Thiết kế dữ liệu

+ Sơ đồ logic

+ Danh sách các thành phần của sơ đồ

Stt	Tên	Loại	Ý nghĩa	Ghi chú

+ Danh sách các thuộc tính của từng thành phần

Tên thành phần:

Stt	Tên	Loại	Kiểu	Miền giá trị	Ý nghĩa

Thiết kế giao diện

Stt	Mã số	Loại	Ý nghĩa	Ghi chú

+ Mô tả chi tiết từng màn hình

- Nội dung
- Danh sách biến cố và xử lý tương ứng trên màn hình

Stt	Biến cố	Ý nghĩa	Xứ lý tương ứng	Mã số xử lý

Thiết kế xử lý

+ Danh sách các xử lý (Các xử lý quan trọng)

Stt	Mã số	Loại	Ý nghĩa	Ghi chú

+ Mô tả chi tiết từng xử lý

- Sơ đồ luồng dữ liệu
- Mô tả chi tiết sơ đồ

4. Cài đặt thử nghiệm

Cài đặt

+ Danh sách tình trạng cài đặt các chức năng (mức độ hoàn thành)

Stt	Chức năng	Mức độ hoàn thành	Ý nghĩa

Thử nghiệm

+ Nội dung các bảng dữ liệu
+ Một số test-case chạy thử nghiệm
+ Các báo biểu màn hình cùng các số liệu tương ứng

5. Tổng kết

+ Các kết quả đã thực hiện
+ Đánh giá ưu khuyết điểm
+ Hướng mở rộng tương lai

PHỤC LỤC B

1. Phần mềm quản lý thư viên

- Mô tả chi tiết các thuộc tính

1. **Độc giả:**

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MDG	Chuỗi	Khóa chính		
2	MLDG	Chuỗi	Khóa ngoại		
3	HoTen	Chuỗi			
4	NgaySinh	Ngày			
5	DiaChi	Chuỗi			
6	DienThoai	Chuỗi			

2. **Sách:**

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MSACH	Chuỗi	Khóa chính		
2	MTG	Chuỗi	Khóa ngoại		
3	MNXB	Chuỗi	Khóa ngoại		
4	MLSACH	Chuỗi	Khóa ngoại		
5	MNN	Chuỗi	Khóa ngoại		
6	TenSach	Chuỗi			
7	Ngàymua	Ngày			
8	SoTrang	Số	>0		

3. **Phiếu mượn**:

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MPHM	Chuỗi	Khóa chính		
2	NgayMuon	Ngày			

4. **Chi tiết mượn**:

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú

1	MPHM	Chuỗi	Khóa ngoại		
2	MSACH	Chuỗi	Khóa ngoại		
3	NgayTra	Ngày			

5.**Loại sách**:

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MLSACH	Chuỗi	Khóa ngoại		
2	TenLoai	Chuỗi			
3	Ghi Chu	Chuỗi			

6. **Loại độc giả:**

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MLDG	Chuỗi	Khóa chính		
2	Tenloại	Chuỗi			
3	GhiChu	Chuỗi			

7. **Nhà xuất bản**:

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MNXB	Chuỗi	Khóa chính		
2	Tenloai	Chuỗi			
3	GhiChu	Chuỗi			

8. **Tác giả**:

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MTG	Chuỗi	Khóa chính		
2	Ten	Chuỗi			
3	GhiChu	Chuỗi			

8. **Ngôn ngữ**

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MNN	Chuỗi	Khóa chính		
2	Ten	Chuỗi			

3	GhiChu	Chuỗi			

- **Mô hình chi tiết các thành phần trong sơ đồ lớp**

1. Đối tượng Độc Giả

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MDG	Chuỗi			
2	Loại độc giả	Số	giá trị rời rạc		
3	HoTen	Chuỗi			
4	NgaySinh	Ngày	từ 19 đến 90		
5	DiaChi	Chuỗi			
6	DienThoai	Chuỗi			

2. Đối tượng Sách

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MSACH	Chuỗi			
2	Loại sách	Số			
3	Tác giả	Chuỗi			
4	Nhà xuất bản	Chuỗi			
5	Ngày nhập	Chuỗi			
6	TenSach	Chuỗi			
7	Ngôn ngữ	Ngày			
8	SoTrang	Số	>0		

3. Quan hệ mượn

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	Ngày mượn	Ngay	>=Ngày nhập		
2	Ngày trả	Ngay	>=Ngày mượn		

2. Phần mềm quản lý giải vô địch bóng đá

- **Mô tả chi tiết các thuộc tính**

1. Đối tượng Đội bóng

Stt	Thuộc tính	Kiểu	Miền giá trị	Ghi chú
1	Tên	Chuỗi	Giá trị rời rạc	
2	Thành Phố	ĐT phụ		
3	Sân nhà	ĐT phụ		
4	Địa chỉ	Chuỗi		
5	Trạng thái	Số	Giá trị rời rạc	
6	Huấn luyện viên	ĐT phụ	Nhiều	

2. Đối tượng Cầu thủ

Stt	Thuộc tính	Kiểu	Miền giá trị	Ghi chú
1	Họ Tên	Chuỗi		
2	Ngày sinh	Ngày		
3	Vị trí	ĐT phụ		
4	Số Áo	Số	>=0	
5	Chiều cao	Số	>1.5	
6	Trạng thái	Số	Rời rạc	

3. Đối tượng Trận đấu

Stt	Thuộc tính	Kiểu	Miền giá trị	Ghi chú
1	Loại trận đấu	Số	Giá trị rời rạc	
2	Ngày	NGAY	>=0	
3	Giờ			
4	Thời gian	Số	>=0	
5	Sân	ĐT phụ		
6	Trọng tài	ĐT phụ	Nhiều	
7	Số khán giả	Số	Ít hơn sức chứa của sân	

4. Quan hệ Thi đấu

Stt	Thuộc tính	Kiểu	Miền giá trị	Ghi chú
1	Số bàn thắng	Số	Giá trị rời rạc	Tính toán

2	Số bàn thua	NGAY	>=0	Tính toán
3	Thẻ phạt	ĐT phụ	Nhiều	Tính toán

5. Quan hệ Ra sân

Stt	Thuộc tính	Kiểu	Miền giá trị	Ghi chú
1	Thời điểm	Số	>=0	
2	Vị trí	ĐT phụ		
3	Bàn thắng	ĐT phụ	Nhiều	
4	Thẻ phạt	ĐT phụ	Nhiều	

- **Mô tả chi tiết thuộc tính**

Đội bóng

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MDBTên	Chuỗi	Khóa chính	
2	MTP	Chuỗi	Khóa ngoại	
3	HoTen	Chuỗi		
4	Diachi	Chuỗi		
5	DienThoai	Chuỗi		

Cầu thủ:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MCT	Chuỗi	Khóa chính	
2	MDB	Chuỗi	Khóa ngoại	
3	MVT	Chuỗi	Khóa ngoại	
4	HoTen	Chuỗi		
5	Ngaysinh	NGAY		
6	SoAo	Số	>0	
7	TrangThai	Logic		

Trận đấu:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MTRD	Chuỗi	Khóa chính	
2	MLTRD	Chuỗi	Khóa ngoại	
3	MSAN	Chuỗi	Khóa ngoại	
4	Ngay	NGAY		

5	GIO	GIO		
6	Thoigian	Số	>0	
7	Sokhangia	Số	>0	

Thi đấu:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MTD	Chuỗi	Khóa chính	
2	MTRD	Chuỗi	Khóa ngoại	
3	MDB	Chuỗi	Khóa ngoại	
4	Ketqua	Số		

Ra Sân

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MRS	Chuỗi	Khóa chính	
2	MTD	Chuỗi	Khóa ngoại	
3	MCT	Chuỗi	Khóa ngoại	
4	MVT	Chuỗi	Khóa ngoại	
5	Thoidiem	Số		

Ghi bàn:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MRS	Chuỗi	Khóa chính, khóa ngoại	
2	MTD	Chuỗi	Khóa chính, Khóa ngoại	
3	Thoidiem	Số		

Phạt

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MRS	Chuỗi	Khóa chính, khóa ngoại	
2	MTHE	Chuỗi	Khóa chính, Khóa ngoại	
3	Thoidiem	Số		

Điều khiển:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú

1	MTRD	Chuỗi	Khóa chính, khóa ngoại	
2	MTTAI	Chuỗi	Khóa chính, Khóa ngoại	
3	MVTRO	Chuỗi	Khóa ngoại	

Loại trận đấu:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MLTRD	Chuỗi	Khóa chính	
2	Tên	Chuỗi		
3	Ghichu	Chuỗi		
4	Sobanthang	Số	>0	tính tóan
5	SoThe	Số	>0	tính toán

Vị trí:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MVT	Chuỗi	Khóa chính	
2	Tên	Chuỗi		
3	Ghichu	Chuỗi		

Trách nhiệm:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MTN	Chuỗi	Khóa chính	
2	Tên	Chuỗi		
3	Ghichu	Chuỗi		

Vai trò:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MVTRO	Chuỗi	Khóa chính	
2	Tên	Chuỗi		
3	Ghichu	Chuỗi		
4	Soluong	Số		tính toán

Loại bàn thắng:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MLBT	Chuỗi	Khóa chính	
2	Tên	Chuỗi		

3	Ghichu	Chuỗi		
4	Soluong	Số		tính toán

Thẻ phạt:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MTHE	Chuỗi	Khóa chính	
2	Tên	Chuỗi		
3	Ghichu	Chuỗi		
4	Soluong	Số		tính toán

Trọng tài:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MTTAI	Chuỗi	Khóa chính	
2	Tên	Chuỗi		
3	Ghichu	Chuỗi		

Huấn luyện viên:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MHLV	Chuỗi	Khóa chính	
2	MDB	Chuỗi	Khóa ngoại	
3	MTN	Chuỗi	Khóa ngoại	
4	Ten	Chuỗi		
5	Ghichu	Chuỗi		

Thành phố:

Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MTP	Chuỗi	Khóa chính	
2	Tên	Chuỗi		
3	Ghichu	Chuỗi		

Sân:

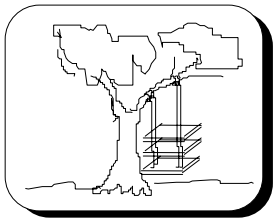
Stt	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	MSAN	Chuỗi	Khóa chính	
2	MTP	Chuỗi	Khóa ngoại	
2	Tên	Chuỗi		
3	Succhua	Số	>0	

Đại cương về công nghệ phần mềm

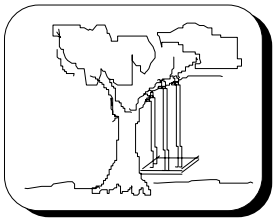
Mục lục

Thời kỳ thứ ba 1970 – 1990	Hệ thống xử lý phân bổ (Distributed processing system) Thông minh (Intelligence) Phần cứng giá thành hạ Hiệu quả tiêu thụ
Thời kỳ thứ tư 1990 trở đi	Hệ thống để bàn (Desktop – Personal – Notebook computers) Lập trình hướng đối tượng (Object oriented programming) Lập trình trực quan (Visual programming) Hệ chuyên gia (Expert system) Mạng thông tin toàn cầu (Worldwide communication network) Xử lý song song (Paralell processing) ...

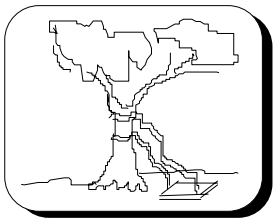
Sau đây là một tranh vui về quá trình tạo ra một sản phẩm phần mềm đã khá quen thuộc đối với những người làm Tin học từ hơn 20 năm nay (theo J. CLAVIER, "Diriger un projet informatique", Edition J. C. I. Inc, Canada 1993) :



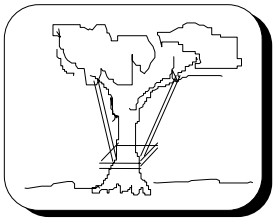
1. Người đặt hàng
Ví dụ : Công ty Công viên



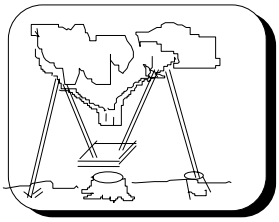
2. Thiết kế của chủ trì đề tài



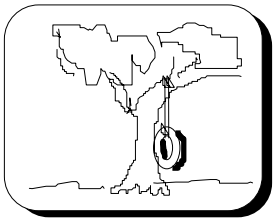
3. Sản phẩm của người lập trình



4. Sau khi sửa sai với nhiều sáng kiến cải tiến



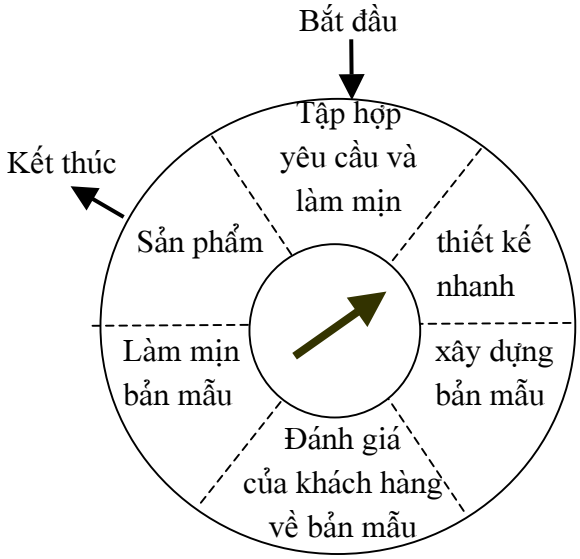
5. Triển khai cho khách hàng



6. Ước mơ của người sử dụng !

Hình 1.1. Quá trình tạo ra một sản phẩm phần mềm

1. *Tiếp cận thác nước* (the waterfall approach) : Bao gồm các giai đoạn đặc tả yêu cầu, thiết kế phần mềm, cài đặt, kiểm thử, v.v..., sau mỗi giai đoạn là sự kết thúc (signed-off) và tiếp tục giai đoạn tiếp theo.
2. *Lập trình thăm dò* (exloratory programming) : Cho phép tăng nhanh quá trình để dẫn đến tính thỏa đáng của hệ thống. Lập trình thăm dò thường được áp dụng trong lĩnh vực trí tuệ nhân tạo, khi NSD không thể định hình được các đặc tả yêu cầu. NSD quan tâm đến tính thỏa đáng của kết quả hơn là tính chính xác.
3. *Bản mẫu* (prototyping) : Tương tự tiếp cận lập trình thăm dò. Pha đầu tiên bao gồm phát triển một chương trình cho phép thử nghiệm. Tuy nhiên, mục đích của phát triển là thiết lập các yêu cầu hệ thống. Sau đó là sự cài đặt lại phần mềm để đưa đến hệ thống chất lượng - sản phẩm.



Hình 1.3. Tiếp cận kiểu bản mẫu

4. *Biến đổi hình thức* (formal transformation) : Là sự biến đổi các đặc tả hình thức (formal specification) của hệ thống phần mềm đang xét để thành một chương trình khả thi nhưng bảo toàn được tính chính xác (correctness - preserving transformations).
5. *Lắp ráp hệ thống từ các thành phần dùng lại được* (system assembly from reusable components). Kỹ thuật này cho phép xây dựng hệ thống từ các thành phần đã có. Tiến trình phát triển hệ thống là sự lắp ráp hơn là sự sáng tạo.

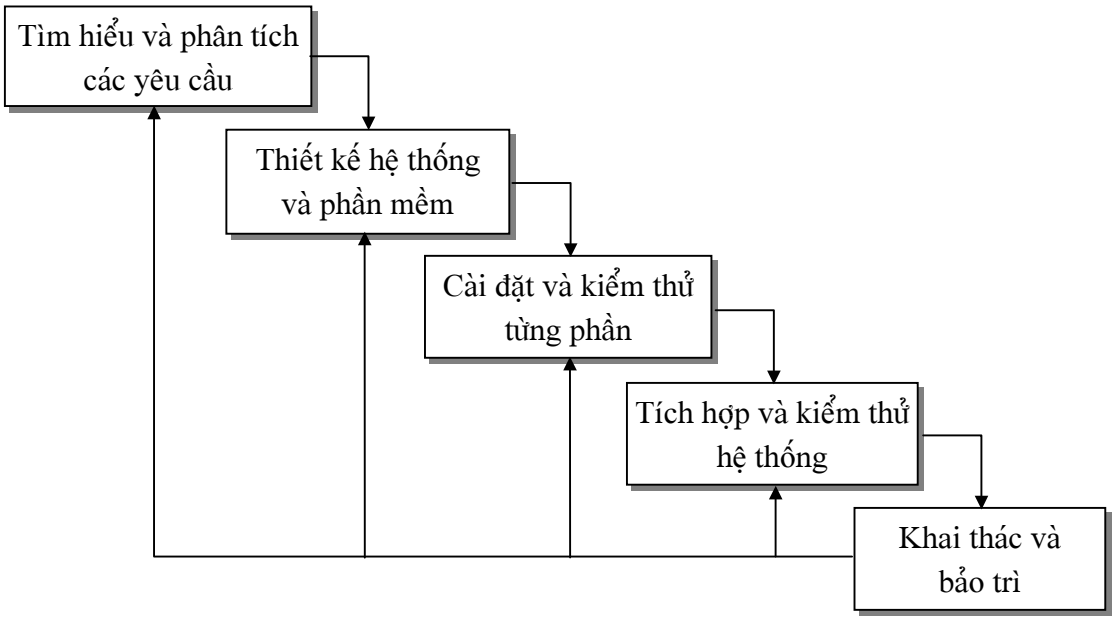
Hiện nay, các tiếp cận 1, 2, 3 được ứng dụng nhiều trong thực tiễn.

Trên thực tế, các giai đoạn phát triển phần mềm không phải rời riêng mà là gối lên nhau (overlap) và thông tin được cung cấp lẫn nhau.

Trong khi thiết kế, những vấn đề và các yêu cầu gắn bó với nhau, trong khi lập trình, những vấn đề thiết kế được tìm thấy, v.v... Lúc này, tiến trình phần mềm không đơn giản là một mô hình tuyến tính mà bao gồm một dãy các tương tác của các hoạt động phát triển.

Tuy nhiên, một mô hình chứa các vòng lặp sẽ làm khó khăn cho việc quản lý và báo cáo. Có nhiều dạng mô hình trong tiến trình phần mềm. Sau đây là một số mô hình :

1. Mô hình thác nước cải tiến

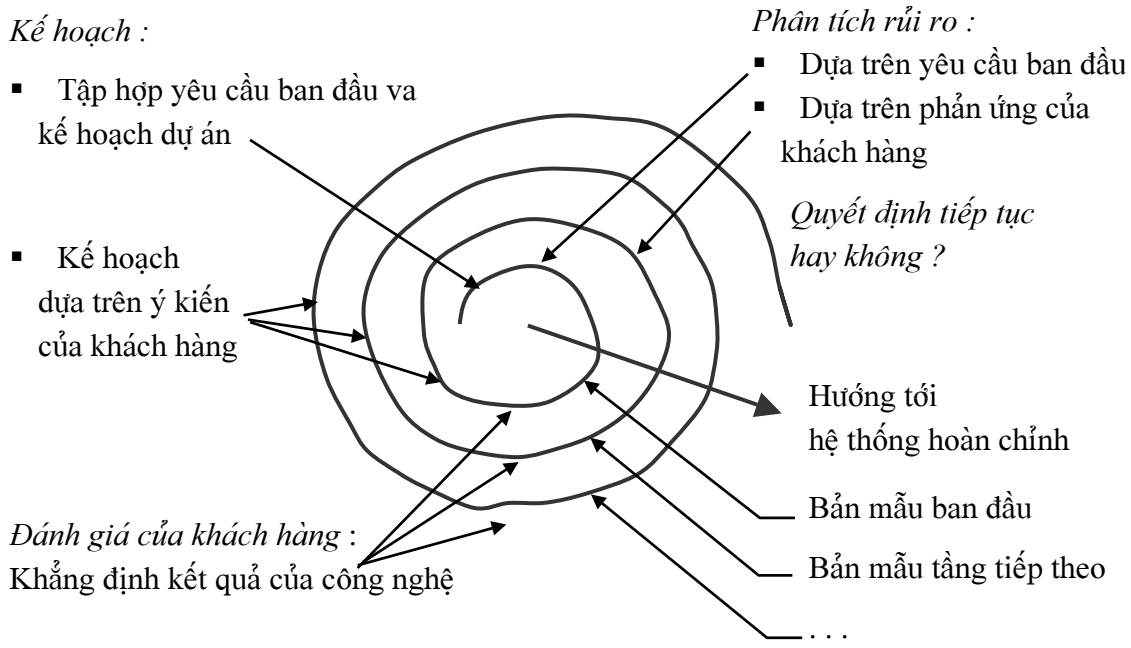


Hình 1.4. Mô hình thác nước cải tiến

1. *Tìm hiểu và phân tích các yêu cầu*: NSD hệ thống và người phát triển hệ thống bàn bạc, trao đổi (consultation) với nhau để thiết lập mục đích, ràng buộc và các dịch vu của hệ thống phần mềm, lĩnh hội được những đòi hỏi của bài toán.
2. *Thiết kế hệ thống và phần mềm* : Tiến trình thiết kế hệ thống phân chia các yêu cầu thành các hệ thống phần cứng, phần mềm và thiết lập một kiến trúc hệ thống toàn bộ (overall system architecture). Việc thiết kế phần mềm bao gồm việc thể hiện các chức năng hệ thống phần mềm (software system functions) để biến đổi thành các chương trình khả thi.
3. *Cài đặt và kiểm thử từng phần* : Trong giai đoạn này, các đơn vị chương trình hay tập hợp các chương trình được kiểm thử lần lượt sao cho thỏa mãn các đặc tả tương ứng.
4. *Tích hợp và kiểm thử hệ thống* : Các đơn vị chương trình được tích hợp và kiểm thử như là một hệ thống đầy đủ để đảm bảo các yêu cầu đặt ra ban đầu. Sau giai đoạn này, hệ thống phần mềm được giao cho khách hàng.
5. *Khai thác và bảo trì* (operation and maintenance) : Đây là một pha dài nhất của chu kỳ sống. Hệ thống được cài đặt và đưa vào sử dụng thực tế. Việc bảo trì bao gồm việc khắc phục những sai sót xảy ra đã không xuất hiện trong các giao đoạn trước đó của chu kỳ sống. Việc tối ưu hóa các dịch vụ của hệ thống được xem như là những yêu cầu mới được phát hiện.

2. Mô hình xoắn ốc

Phát triển trên tính ưu việt của vòng đời cổ điên và bản mẫu, bổ sung nhũng yếu tố còn thiếu và thêm các yếu tố mới, phân tích rủi ro.



Hình 1.5. Mô hình xoắn ốc

Ưu điểm :

Các phiên bản (hay sản phẩm) được hoàn thiện dần theo chiều xoáy ốc từ trong ra ngoài.

Nhược điểm :

- Khó đánh giá chính xác, nhất là khi gặp rủi ro, khó kiểm soát. Do đó khó thuyết phục được các khách hàng lớn
- Mô hình nàuy còn mới, chưa được kiểm nghiệm nhiều trong thực tiễn.

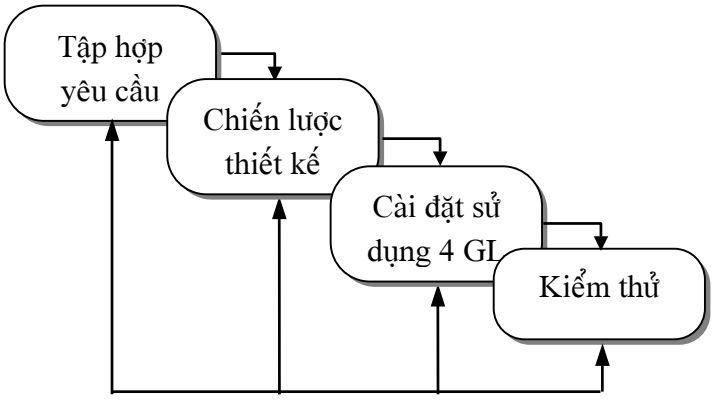
3. Kỹ thuật thế hệ 4 (4th Generation Technology)

Bao gồm các công cụ phần mềm trên cơ sở tự động sản sinh mã chương trình gốc theo nhu cầu của người phát triển :

- Ngôn ngữ phi thủ tục[2] (non procedural language) để truy cập cơ sở dữ liệu.
- Bộ sinh báo cáo.
- Bộ thao tác dữ liệu.

[2] là ngôn ngữ lập trình không tuân theo cách gọi thủ tục hay gọi chương trình con thông thường, không sử dụng các cấu trúc điều khiển, tuần tự, mà dựa trên tập hợp các yếu tố và quan hệ để dẫn về kết quả yêu cầu. Ví dụ ngôn ngữ vấn tin SQL thuộc loại này.

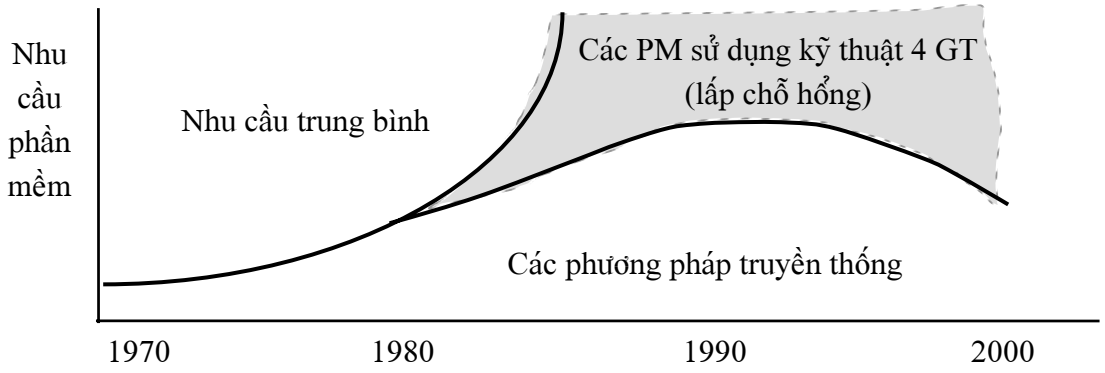
- Bộ tương tác và thiết kế màn hình.
- Bộ sinh chương trình.
- Bảng tính.
- Công cụ đồ họa.



Hình 1.6. Kỹ thuật thế hệ 4

Ưu điểm :

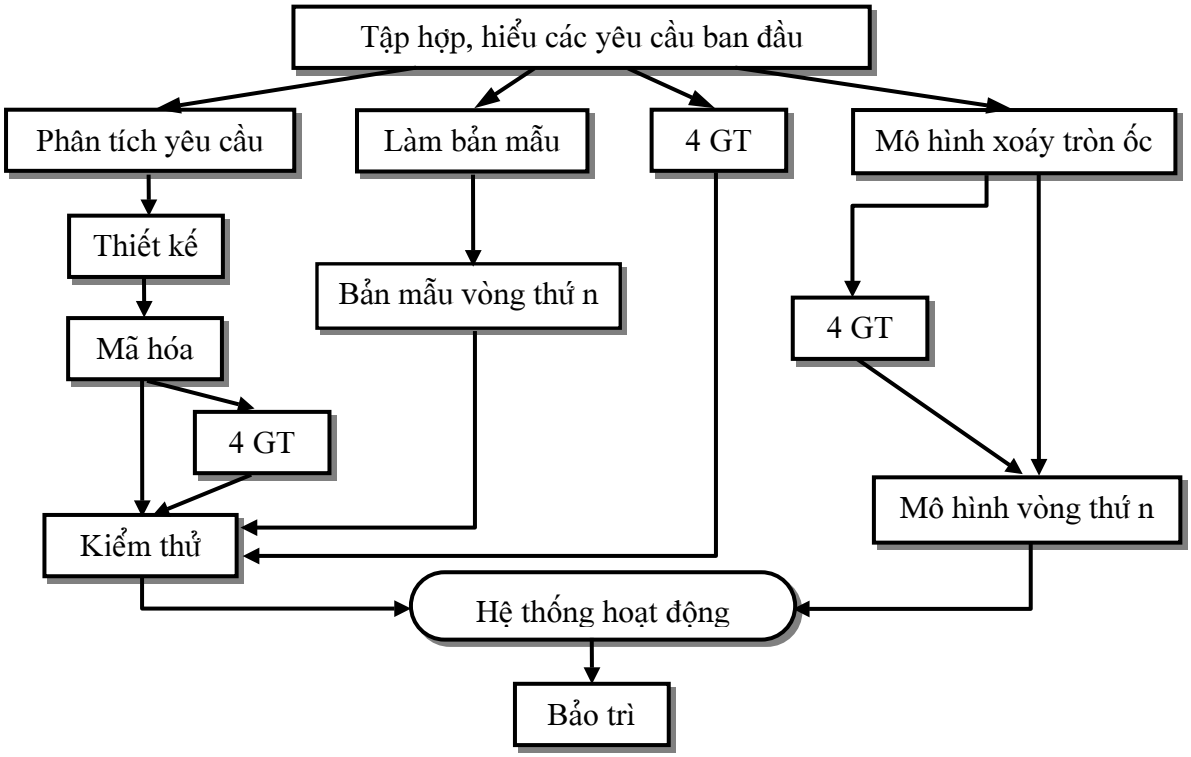
Thường được sử dụng để xây dựng các hệ thông tin và tương lai là các ứng dụng kỹ nghệ phát triển phần mềm thời gian thực.



Hình 1.7. Nhu cầu phần mềm

5. Tích hợp các kỹ thuật

Nhằm tăng cường tính tối ưu trong phát triển phần mềm, người ta có xu hướng tích hợp các kỹ thuật cổ điển, xoáy tròn ốc và 46T đã nêu.



Hình 1.8. Tích hợp các kỹ thuật

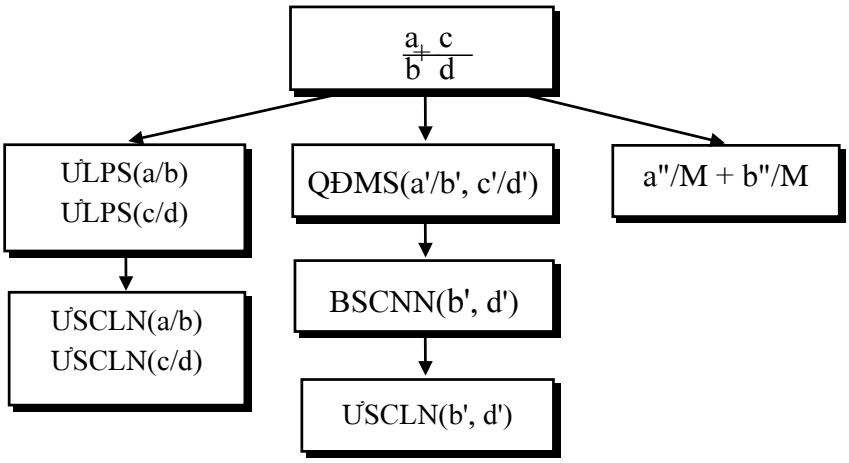
Ví dụ 2 :

Phân tích bài toán cộng hai phân số để đưa về bài toán tìm ước số chung lớn nhất.

Để cộng hai phân số, trước tiên cần ước lược chúng, tiếp đó quy đồng mẫu số để lấy mẫu số chung. Cuối cùng tiến hành cộng hai tử số của hai phân số đã có chung mẫu số. Việc ước lược phân số được đưa về tìm ước số chung lớn nhất của tử số và mẫu số (sử dụng thuật toán Euclide).

Để quy đồng mẫu số, cần tìm bội số chung nhỏ nhất. Việc tìm bội số chung nhỏ nhất của hai số lại được đưa về tìm ước số chung lớn nhất của chúng :

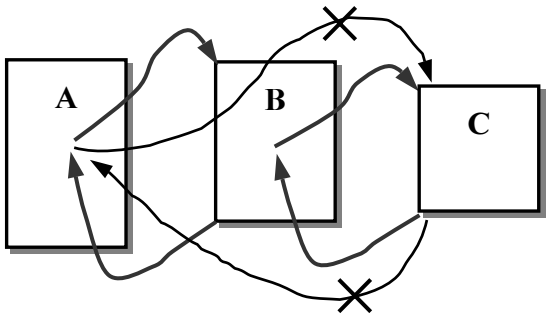
BSCNN(b', d') = b' * d' / ƯSCLN(b',d').



Hình 2.2. Phân tích bài toán cộng hai phân số

Như vậy, chương trình là một hệ thống gồm nhiều thành phần phân cấp, mỗi thành phần có nhiệm vụ giải quyết một vấn đề sơ cấp và có tính độc lập cao. Các thành phần nên tương tác với nhau tối thiểu. Giữa hai thành phần trong hệ thống chỉ nên có tối đa một đường tương tác là đường trao đổi thông tin để dễ quản lý và dễ kiểm soát.

- Giữa chương trình chính và chương trình con có đường giao tiếp là việc truyền tham biến. Không được gọi chương trình con theo kiểu vượt cấp.



Hình 2.3. A gọi B, B gọi C, nhưng A không gọi được C

- Hạn chế dùng biến toàn cục (global variables) trong chương trình con vì sẽ tạo thêm những đường giao tiếp khó quản lý. Chẳng hạn, một chương trình con nào đó làm thay đổi một biến toàn cục thì ở một nơi khác, trong một chương trình con khác hoặc ngay trong chương trình chính, cũng sẽ khó nhận biết sự thay đổi này.

b) Không sử dụng lệnh nhảy goto

Lệnh goto (jump statement) dùng để chuyển điều khiển đến một điểm khác trong chương trình. Lệnh goto làm khó quản lý và khó kiểm soát chương trình nên khó đọc, khó sửa sai (rối rắm như món mì sợi Spaghetti của Ý).

Các chương trình viết trên ngôn ngữ aassembly hoặc trên các ngôn ngữ bậc cao như Fortran, Algol, Cobol... thường sử dụng lệnh goto.

Ví dụ 3 :

Chương trình Algol sau đây sử dụng lệnh goto để điều khiển vòng lặp tính tổng các phần tử của mảng a gồm N số thực :

```
S := 0; I := 0;
Start : S := S + a[ I]; I := I + 1;
     if I <= N then goto Start;
...
```

c) Chương trình có tính cấu trúc

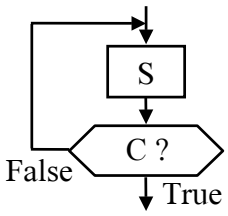
Chương trình chỉ sử dụng các cấu trúc điều kiện chuẩn, dễ hiểu, dễ thể hiện thuật toán. Cấu trúc của chương trình phản ánh được cấu trúc của vấn đề và cách giải quyết vấn đề (làm như thế nào ?). Phương pháp hay được sử dụng để thiết kế chương trình là *phân tích từ trên xuống* (Top-Down Analysis) và *tổng hợp từ dưới lên* (Bottom up Synthesis).

Nội dung phương pháp phân tích từ trên xuống là nhìn nhận xem xét tổng quát toàn bộ vấn đề, xuất phát từ mục tiêu (đỉnh) đi xuống các thành phần trong hệ thống, chia các thành phần thành các thành phần nhỏ hơn theo một cấu trúc phân cấp chặt chẽ.

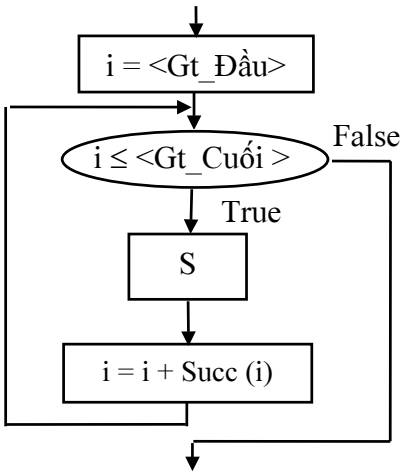
Nội dung phương pháp tổng hợp từ dưới lên là xuất phát từ các vấn đề cụ thể và cách giải quyết cụ thể, sau đó tích hợp chúng lại thành vấn đề lớn hơn và cách giải tổng quát hơn, hướng từ dưới lên trên để nhận được vấn đề cần phải giải quyết ban đầu

Cách thiết kế này gây khó khăn vì khó kiểm soát và dễ lạc hướng, khó đáp ứng đầy đủ các yêu cầu của vấn đề.

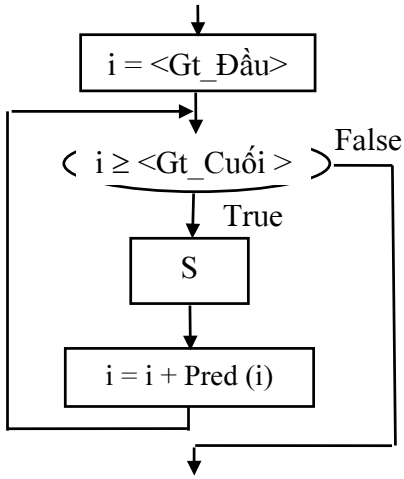
	while C **do** S		

Stt	*Cấu trúc điều khiển*	*Lưu đồ tương đương*	*Mô tả*
6	Lặp với kiểm tra điều kiện sau khi thực hiện xong thân vòng lặp : **do** S **until** C		Còn thực hiện S khi C còn chưa thoả mãn (sai). Ít nhất lặp được một lần. Dừng khi C đúng.

7 Lặp hết trước số lần (for)

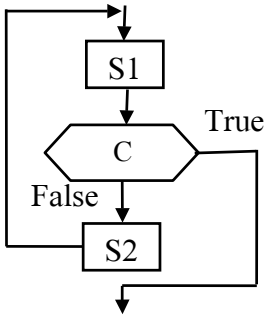


for I ← Gt_Đầu **to** Gt_Cuối **do** S



for I ← Gt_Đầu **downto** Gt_Cuối **do** S

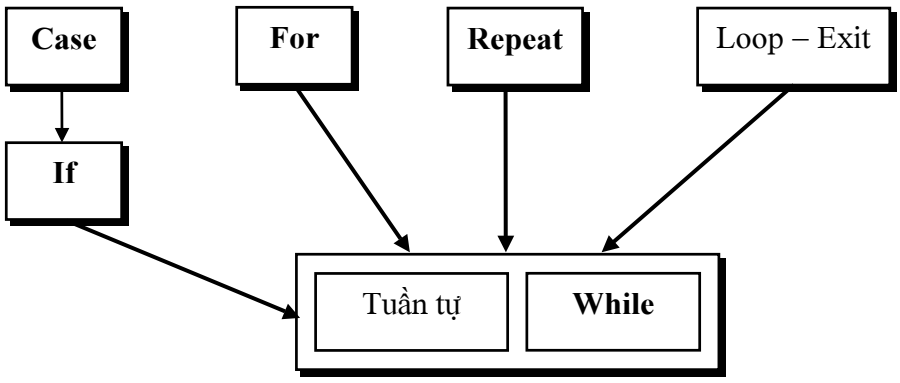
Ngoài các cấu trúc lặp hay gặp thông thường trên đây, người ta còn sử dụng các cấu trúc lặp có thoát (loop exit) như sau :




```
{ sử dụng khoá key để đánh dấu lối thoát,
  key không xuất hiện trong S và trong C }
key := False
While not key do begin
    S1
    if C then key := True else S2
End
```

- *Với mọi dữ liệu vào X thuộc miền xác định X, ta có P(x) = Q(x) : P và Q biến đổi những cái vào giống nhau thành các ra giống nhau.*
- *Các thao tác trên các biến của Q là giống như của P.*
- *Các biến của Q cũng là các biến của P, có thể Q chứa thêm một số biến logic.*
- *Q sử dụng hai cấu trúc điều khiển duy nhất là **tuần tự** và vòng lặp* **while** (SW: Sequence &While).

Sơ đồ chuyển cấu trúc như sau :



*Hình 2.4. Chuyển về cấu trúc **tuần tự** và lặp **while*** (SW)

Sau đây là cách chuyển đổi của các cấu trúc **Case**, **If**, **For**, **Repeat** và **Loop** – Exit :

1. **Case → if**

```
Case C of
    C1 : S1
     . . .
    Cn : Sn
End {Case}
```

→

```
If C = C1 then S1
else if C = C1 then S2
    else if . . .
    . . .
    else if C = Cn then Sn
```

2. **if → SW**

Dùng hai biến phụ kiểu logic để thực hiện vòng lặp **While** đúng một lần :

Var p, q : **Boolean**

```
If C then S
```

→

```
p := C
While p do begin
    S ; p := not p
End
```

```
If C then S1 else S2
```

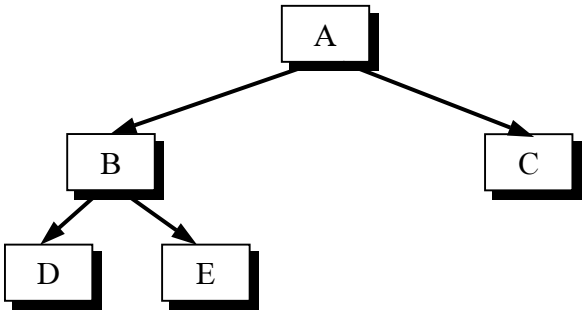
→

```
p := C ; q := p
While p do begin
    S1 ; p := not p
```

Các đơn thể được tổ chức phân cấp dạng cây nhưng phải thỏa mãn tính *cục bộ tham chiếu* (locality of Reference) : chỉ có đơn thể mức cao hơn (cha) mới có quyền tham chiếu (gọi) đến đơn thể mức thấp hơn kề đó (con).

Những cấu trúc cây thỏa mãn tính cục bộ tham chiếu được gọi là *cấu trúc cây thuần túy* (Pure tree Structure).

Ví dụ 9 :



– D chỉ phục vụ B, chỉ có B mới có quyền điều khiển D, C không thể gọi D.

– Giữa B và D chỉ có một đường tương tác duy nhất là trao đổi tham biến.

Hình 2.6. Cấu trúc cây thuần túy

a) Đặc điểm của cấu trúc cây thuần túy

– Mỗi đơn thể chỉ được quyền điều khiển đơn thể con trực tiếp, giảm được tính phức tạp của chương trình.

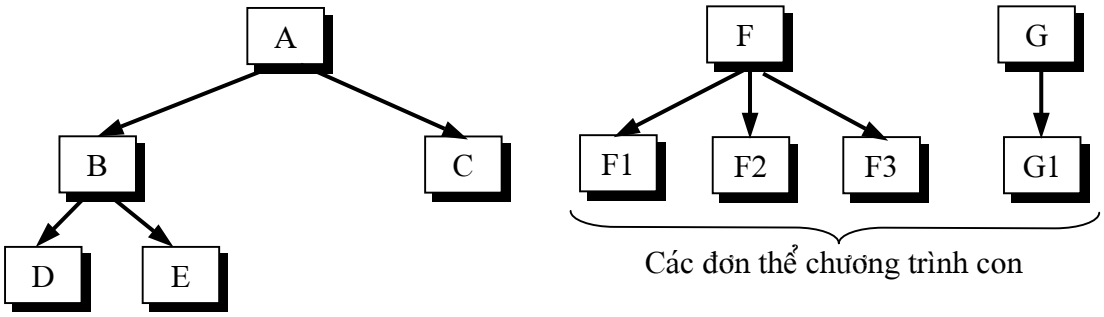
– Các nhánh hoàn toàn tách biệt nhau nên có tính tương tác tối thiểu.

Ngoại lệ : Nếu một công việc nào đó cần thực hiện nhiều lần, nhiều chỗ trong chương trình thì nên tổ chức thành một đơn thể chương trình con và vẽ riêng, không vẽ vào cấu trúc cây.

Như vậy, các đơn thể chương trình con chỉ đóng vai trò thư viện, một sự mở rộng của ngôn ngữ lập trình. Trường hợp đơn thể chương trình con phức tạp thì có thể tổ chức theo cấu trúc cây thuần túy.

Như vậy toàn bộ chương trình là một tập hợp các cấu trúc cây thuần túy.

Ví dụ 10 :



Hình 2.7. Cấu trúc cây thuần túy của chương trình và chương trình con

```
for c := '0' to '9' do
  writeln('số các chữ số',c,'đã đọc =',số [ c] :2);
```

Ta tinh chế bước *kí_tự_là_chữ_số* bằng cách chuyển ra dạng biểu thức Pascal như sau :

```
('0' < c) and (c < = '9')
```

Việc *đọc_một_kí_tự* được viết như sau : `Read (c);`

Dấu chấm có thể dùng hằng :

```
Const dấu chấm '=';
```

Ta thấy trước lúc đếm, các phần tử của mảng số phải bằng 0. Ta có :

```
for c := '0' to '9' do số [ c] := 0;
```

Bây giờ ta có chương trình hoàn chỉnh như sau :

```
Program Đếm chữ số;
Const dấu chấm = '.';
Type dãy = array [ '0'..'9'] of integer;
Var  số: dãy;
     c: char;
begin
   for c := '0' to '9' do số [ c] := 0;
   writeln ('Hãy gõ vào các kí tự');
   writeln ('và kết thúc bằng dấu chấm (.) :');
   Repeat
      Read (c);
      if ('0' < = c) and (c < = '9') then
      số [ c] := số [ c] + 1;
   Until c = dấu chấm;
   for c := '0' to '9' do
      writeln ('Số các chữ số', c, ' đã đọc =', số [ c] : 2)
   Readln
end.
```

Cho chạy chương trình ta được kết quả như sau :

```
Hãy gõ vào các kí tự
và kết thúc bằng dấu chấm (.) :
ytr7657g858450020820.
Số các chữ số 0_đã đọc = 4
Số các chữ số 1_đã đọc = 0
Số các chữ số 2_đã đọc = 2
Số các chữ số 3_đã đọc = 0
Số các chữ số 4_đã đọc = 1
Số các chữ số 5_đã đọc = 3
```

Đã_quay_lại_quá_cột_đầu: Tức là đã lùi quá cột đầu tiên : tình trạng bế tắc xảy ra : không tìm ra lời giải !

```
j < 1
```

Thử_cột : Tìm xem có thể đặt quân hậu tại hàng nào ở cột đang xét. Bước *Thử_cột* sẽ có dạng :

```
repeat
   Xét_một_hàng  ;  { là hàng thứ i }
   Kiểm_tra_an_toàn  ;  { khi đặt quân hậu vào hàng này }
Until An_toàn or Đã_xét_đến_hàng_cuối;
```

Lúc đầu I=0, việc *Xét_một_hàng* tức : `i:= i+1`

Từ đó ta có ngay *Đã_xét_đến_hàng_cuối* tức là : `i = 8`

Lúc này người ta tìm cách biểu diễn dữ liệu tương ứng vì các công việc đã có vẻ "mịn" rồi. Theo lời khuyên của Niclaus Wirth, sự biểu diễn dữ liệu càng trì hoãn lâu càng tốt (đến khi không thể trì hoãn được nữa) !

Vì bàn cờ có 8 x 8 ô nên có thể nghĩ ngay đến việc sử dụng một ma trận Boolean hai chiều để biểu diễn :

```
Var B : array [ 1..8, 1..8] of Boolean;
```

B [i, j] có giá trị true nếu có quân hậu ở hàng i, cột j.

Tuy nhiên, cách biểu diễn này gây khó khăn cho việc kiểm tra hai đường chéo có 2 quân hậu nào ăn nhau không theo luật cờ vua ?

Bây giờ ta dùng 3 dãy Boolean a, b, c với :

a [i] = true nếu không tồn tại quân hậu nào nằm trên hàng i.

b [k] = true nếu không tồn tại quân hậu nào nằm trên đường chéo thuận thứ k.

c [l] = true nếu không tồn tại quân hậu nào nằm trên đường chéo nghịch thứ l.

Với mỗi ô (i, j) hàng i cột j, ta có quan hệ như sau :

–đường chéo thuận thứ k thoả mãn i + j = k;

–đường chéo nghịch thứ l thoả mãn i - j = l;

Vì vậy, nếu : $1 \le i, j \le 8$ thì : $2 \le k \le 16$ và : $-7 \le l \le 7$.

Ta có các mảng a , b , c như sau :

```
var  a : array [ 1..8] of boolean;
     b : array [ 2..16] of boolean;
     c : array [ -7..7] of boolean;
```

Để biểu diễn sự kiện đặt quân hậu tại cột j vào hàng i, ta dùng dãy nguyên x sao cho x [j] = i nếu như có một quân hậu ở ô (i, j) :

```
var x : array [ 1..8] of integer;
```

Việc đặt quân hậu vào ô (i, j) sẽ làm cho :

a [i] = b [i+j] = c [i-j] = false

Kiểm_tra_an_toàn : Cho đến lúc này, chưa có hai quân hậu nào trong số những quân đã đặt lên bàn cờ có thể ăn lẫn nhau. Điều kiện *An_toàn* để đặt quân hậu vào ô (i, j) là :

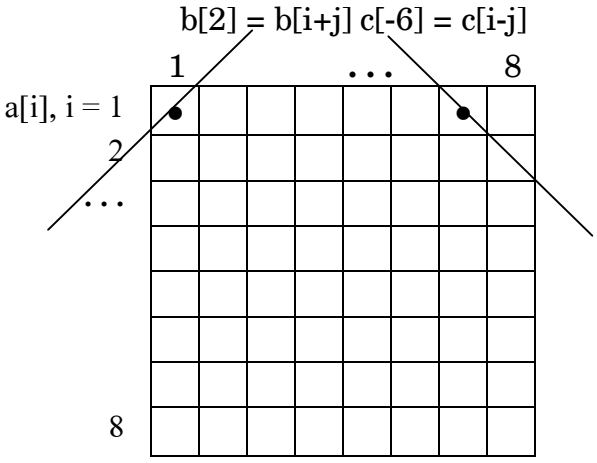
a [i] = b [i+j] = c [i-j] = true;

Bằng cách sử dụng một biến logic :

```
Var Antoan: Boolean;
```

Việc *Kiểm_tra_an_toàn* được dịch ra Pascal như sau :

```
An toàn := a [ i] and b [ i + j] and c [ i - j];
```



Hình 2.8. Bàn cờ vua cho bài toán tám quân hậu

Đặt quân hậu vào ô (i, j) *Đặt_quân_hậu_vào* sẽ là :

```
x[ j] := i;
a [ i] := false;
b [ i+j] := false;
c [ i-j] := false;
```

Tiếp tục tinh chế bước phức tạp nhất là *Quay_lại* :

Quay_lại : là quay lại một cột ở trước cột đang xét để đặt lại quân hậu cho cột đó khi tình thế hiện trạng là bế tắc.

Bước *Quay_lại* có dạng :

```
Xét_lại_cột_trước;
if not Đã_quay_lại_quá_cột_đầu then begin
   Bỏ_quân_hậu_ở_cột_đó; { tức cột trước cột đang xét, ô (i, j) }
   if Đang_ở_hàng_cuối_cùng then begin
```

```
        Xét_lại_cột_trước ;
        if not Đã_quay_lại_quá_cột_đầu then
           Bỏ_quân_hậu_ở_cột_đó
     end
  end;
```

Dễ dàng ta thấy *Xét_lại_cột_trước* tức là :

```
j = j - 1;
```

Còn *Đã_quay_lại_quá_cột_đầu* thì đã xét trước đây, tức là :

```
j < 1;
```

Thao tác *Bỏ_quân_hậu_ở_cột_đó* sẽ có dạng:

```
i:= x [ j] ; a [ i] := true; b[ i+j] := true; c[ i-j] := true;
```

Chương trình hoàn chỉnh như sau :

```
Program TamQuânHau;
Uses Crt;
Const Hau='Q ';ov='#';
Var x: array[ 1..8] of Integer;
    a: array[ 1..8] of Boolean;
    b: array[ 2..16] of Boolean;
    c: array[ -7..7] of Boolean;
    i,j: Integer;
    antoan: Boolean;
Begin
   for i:=1 to 8 do a[ i] :=true;
   for i:=2 to 16 do b[ i] :=true;
   for i:=-7 to 7 do c[ i] :=true;
   j:=1; i:=0;
   repeat
      repeat
         i:=i+1;
         antoan:=a[ i] and b[ i+j] and c[ i-j] ;
      until antoan or (i=8);
      if antoan then begin
         x[ j] :=i;
         a[ i] :=false; b[ i+j] :=false; c[ i-j] :=false;
         j:= j+1; i:= 0
      end else begin
         j:=j-1;
         if j>=1 then begin
            i:=x[ j] ;
            a[ i] :=true; b[ i+j] :=true; c[ i-j] :=true;
            if i=8 then begin
```

- Khi đã đến cột cuối cùng và đặt quân hậu cuối cùng vào bàn cờ, ta in lời giải ra nhưng chưa kết thúc chương trình ngay mà tiếp tục quay trở lại để tìm lời giải khác.
- Chương trình ngừng khi sự quay lại đã quá cột đầu.

Lời giải có dạng phác thảo như sau :

```
Xét_cột_đầu;
repeat
   Thử_cột ;
   if An_toàn then begin
      Đặt_quân_hậu_vào ;
      Xét_cột_kế ;
      if Cột_kế_vượt_quá_cột_cuối_cùng then begin
         In_ra_lời_giải;
         Quay_lại
      end
   end else Quay_lại
Until Đã_quay_lại_quá_cột_đầu;
```

Từ đây, với các bước làm mịn đã giải quyết ở mục trước, ta có thể viết lại thành chương trình hoàn chỉnh.

Bài toán mã đi tuần

Cho bàn cờ n × n ô và một quân mã đang ở toạ độ x_0, y_0. Hãy tìm cách cho quân mã đi theo luật cờ vua để qua hết tất cả các ô của bàn cờ, mỗi ô đi qua đúng một lần ?

Cách giải quyết để quân mã đi qua hết $n^2 - 1$ ô của bàn cờ là tại mỗi ô mà quân mã đang đứng, hãy xác định xem có thể thực hiện một nước đi kế tiếp nữa hay không ? Như vậy thuật toán để tìm nước đi kế tiếp có thể viết thành thủ tục đệ quy dạng phác thảo như sau :

```
Procedure Thử_nước_đi_kế;
Begin
   Khởi_động_nước_đi_có_thể
   Repeat
      Chọn_một_nước_đi
      if OK then begin
         Thực_hiện_nước_đi
         if Chưa_hết_nước then begin
            Thử_nước_đi_kế;
            if NotOK then Xoá_nước_trước
         end else Thành_công
      end
   Until Đi_được or (Hết_nước_đi);
   Kết_thúc
End;
```

Bây giờ ta cần tìm cấu trúc dữ liệu để biểu diễn bàn cờ n × n ô, mỗi ô có toạ độ (i, j), với 1 ≤ i,j ≤ n. Dễ dàng ta tìm được mô tả như sau :

```
Type Idx = 1..n;
Var  H: Array[ Idx, Idx] of Integer;
```

Trong mô tả trên, thay vì sử dụng giá trị kiểu `Bollean` để đánh dấu ô đó đã được đi qua chưa, ta đưa vào giá trị kiểu `Integer` để dò theo quá trình di chuyển của quân mã theo quy ước như sau :

H[x, y] = 0 ô <x, y> chưa được quân mã đi qua

H[x, y] = i ô <x, y> đã được quân mã đi qua ở nước thứ i, $1 \le i \le n^2$

Để chỉ một nước đi có thành công hay không, ta sử dụng biến `Bollean q` với quy ước như sau :

`q = true` nước đi thành công

`q = false` không có nước đi

Ta thấy điều kiện *Chưa_hết_nước* được biểu diễn bởi biểu thức : $i \le n^2$

Giả sử gọi u, v là toạ độ nước đi kế tiếp của quân mã theo luật cờ vua thì điều kiện OK phải thoả mãn :

- Ô mới <u, v> phải thuộc vào bàn cờ, nghĩa là 1 ≤ u ≤ n và 1 ≤ v ≤ n.
- Quân mã chưa đi qua ô <u, v>, nghĩa là H[u, v] = 0.

Bằng cách xây dựng tập hợp :

```
Var s: set of Idx;
```

biểu thức điều kiện OK bây giờ có thể viết :

```
(u in s) and (v in s) and H[ u, v] =0
```

Để ghi nhận nước đi hợp lệ *Thực_hiện_nước_đi*, ta sử dụng phép gán :

```
H[ u, v]  := i;
```

Từ đó, việc *Xoá_nước_trước* có thể sử dụng phép gán :

```
H[ u, v]  := 0
```

Để ghi nhận kết quả lời gọi đệ quy, ta sử dụng biến `Bollean q1` cho biểu thức điều kiện *Đi_được*. Như vậy, *Thành_công* sẽ là :

```
q1 := true
```

và *Kết_thúc* sẽ là :

```
q := q1
```

Bây giờ ta có lời giải mịn hơn như sau :

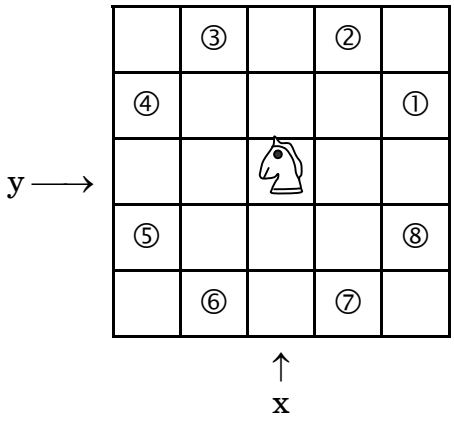
```
Procedure Try(i:Integer; x, y : Idx; Var q: Boolean);
```

```
Var  u, v:Integer;
     q1: Boolean;
Begin
   Khởi_động_nước_đi_có_thể
   Repeat
      Chọn_một_nước_đi
      if (u in s) and (v in s) and H[u, v]=0 then begin
         H[u, v] := i;
         if n < sqr(n) then begin
            Try(i+1, u, v, q1);
            if not q1 then H[ u, v] := 0
         end else q1:= true
      end
   Until q1 or (Hết_nước_đi);
   q := q1
End;
```

Cho đến lúc này, ta chưa xét đến luật đi của quân mã, nghĩa là chương trình xây dựng ở trên độc lập với luật cờ vua với chủ ý giảm nhẹ những chi tiết chưa cần thiết khi phát triển chương trình.

Như vậy ta vẫn còn hai việc chưa giải quyết là : *Khởi_động_nước_đi_có_thể* và *Chọn một_nước_đi*.

Cho trước một toạ độ bất kỳ <x, y> của quân mã trên bàn cờ, ta có thể có tám ô <u, v> được đánh số từ 1..8 (theo chiều ngược kim đồng hồ) mà quân mã có thể nhảy đến như hình dưới đây :



Hình 2.9. Các vị trí khác nhau của quân mã

Để có được <u, v>, từ <x, y>, ta cần xác định giá trị chênh lệch theo toạ độ. Ta sẽ dùng hai mảng một chiều a và b, mỗi mảng sẽ có kích thước 8 phần tử, để lưu giữ 8 giá trị chênh lệch theo toạ độ <x, y>, với quy ước chiều đi ↑ và → mang dấu +, chiều đi ← và ↓ mang dấu –. Ta có khai báo như sau :

```
Var a, b: Array[ 1..8] of integer;
```

Chẳng hạn nếu cho <x, y> = <x_0, y_0> với điều kiện n–2 ≥ x_0, y_0 ≥ 3 thì ta có thể có 8 cặp giá trị như sau :

```
① a[ 1] := 2;    b[ 1] := 1;
② a[ 2] := 1;    b[ 2] := 2;
③ a[ 3] := -1;   b[ 3] := 2;
④ a[ 4] := -2;   b[ 4] := 1;
⑤ a[ 5] := -2;   b[ 5] := -1;
⑥ a[ 6] := -1;   b[ 6] := -2;
⑦ a[ 7] := 1;    b[ 7] := -2;
⑧ a[ 8] := 2;    b[ 8] := -1;
```

Bây giờ, để đánh số các nước đi có thể, ta sử dụng một biến k nguyên, k sẽ nhận giá trị trong phạm vi 1..8. Như vậy, đầu thủ tục, việc *Khởi_động_nước_đi_có_thể* tương ứng với lệnh gán :

```
k:= 0;
```

Việc *Chọn một_nước_đi* tương ứng với các lệnh gán :

```
k:= k + 1;
q1:= false;
u:= x + a[ k] ; v:= y + b[ k] ;
```

Còn biểu thức điều kiện *Hết_nước_đi* sẽ tương ứng với : `k = 8`

Thủ tục đệ quy được bắt đầu bởi toạ độ <x_0, y_0> = <1,1>, kể từ nước đi k=2, các ô của bàn cờ đều có thể là đích của quân mã với khởi động :

```
for i:=1 to n do for j:=1 to n do H[ i, j] := 0;
```

Lời gọi thủ tục như sau :

```
H[ 1, 1] := 1; Try(2, 1, 1, q);
```

Cuối cùng là một thay đổi nhỏ bằng cách thêm biến nguyên `nsq` để tính `sqr(n) ngoài thủ tục`. Chú ý rằng n ≥ 5. Sau đây là chương trình hoàn chỉnh :

Chương trình mã đi tuần :

```
PROGRAM KnightsTour;
Uses CRT, Dos;
Const NMax=50;
Type Idx = 1..Nmax;
Var  i, j: Idx;
     N, Nsq: integer;
     q: Boolean;
     s: set of Idx;
     H: Array[Idx, Idx] of Integer;
     a, b: Array[1..8] of integer;
     gio, phut, giay, hund : Word; { Để tính thời gian }
Procedure Try(i:Integer; x, y : Idx; Var q: Boolean);
```

```
Var k, u, v:Integer; q1: Boolean;
Begin
k:= 0;
  Repeat
    k:= k + 1; q1:= false; u:= x + a[k]; v:= y + b[k];
    if (u in s) and (v in s) and (H[u, v]=0) then begin
       H[u, v] := i;
       if i < Nsq then begin
          Try(i+1, u, v, q1);
          if not q1 then H[u, v] := 0
       end else q1:= true
    end
  Until q1 or (k=8);
  q := q1
End { Try };
Begin { KnightsTour main }
   ClrScr;
   a[1]:= 2;   b[1]:= 1;
   a[2]:= 1;   b[2]:= 2;
   a[3]:= -1;  b[3]:= 2;
   a[4]:= -2;  b[4]:= 1;
   a[5]:= -2;  b[5]:= -1;
   a[6]:= -1;  b[6]:= -2;
   a[7]:= 1;   b[7]:= -2;
   a[8]:= 2;   b[8]:= -1;
   Write('Cho N = '); Readln(N);
   While (N>1) and (N<Nmax) do begin
      { Giờ bắt đầu tính }
      GetTime(gio, phut, giay, hund);
      Writeln('Bắt đầu =', gio:2,':', phut:2,':', giay:2,':',
      hund);
      nsq:= sqr(N);
      s:=[1..n];
      for i:=1 to n do for j:=1 to n do H[i, j]:= 0;
      H[1, 1]:= 1; Try(2, 1, 1, q);
      if q then
         for i:=1 to N do begin
             for j:=1 to N do
                write(h[i, j]:5);
             Writeln
         end
      else Writeln('Không có lời giải !');
      { Giờ bắt đầu tính }
      GetTime(gio, phut, giay, hund);
      Writeln('Kết thúc=', gio:2,':', phut:2,':', giay:2,':',
      hund);
      Write('Cho N = '); Readln(N);
   End {While}
End.
```

Sau đây là kết quả với N = 5 :

Cho N = 5

Bắt đầu = 5:59:57:30

1	6	15	10	21
14	9	20	5	16
19	2	7	22	11
8	13	24	17	4
25	18	3	12	23

Kết thúc = 5:59:57:36

Kết quả với N = 6 :

Cho N = 6

Bắt đầu = 6: 0:40:80

1	16	7	26	11	14
34	25	12	15	6	27
17	2	33	8	13	10
32	35	24	21	28	5
23	18	3	30	9	20
36	31	22	19	4	29

Kết thúc = 6: 0:41:79

b) Phần giao tiếp (Interface Section)

```
Interface
        { Các khai báo sau đây không bắt buộc phải có }
        Uses <Ds các Units dùng cho Unit này>
        Const <Ds các hằng>
        Type <Ds các mô tả kiểu>
        Var <Ds các khai báo biến>
        <Ds các phần đầu của các thủ tục và hàm>
```

Các khai báo Const, Type và Var sau Interface cho Unit cũng dùng được tại nơi sử dụng đơn vị này. Ta nói chúng là "thấy được" (Visible).

Danh sách các tên thủ tục và hàm được các Unit khác dùng đến sẽ được khai báo đầy đủ trong phần tiếp theo :

c) Phần hiện thực (Implementation Section)

```
Implementation
        { Các khai báo sau đây không bắt buộc phải có }
        Uses <Ds các Uses sử dụng đến>
        Label <Ds các nhãn>
        Const <Ds các hằng>
        Type <Ds các mô tả kiểu>
        Var <Ds các khai báo biến>
        <Các mô tả hàm và/hoặc thủ tục>
```

Các mô tả hàm và/hoặc thủ tục trong phần này gồm có :

- Các hàm và /hoặc thủ tục đã mô tả ở phần giao tiếp.
- Các hàm và/hoặc thủ tục nội bộ dùng riêng trong phần hiện thực (không khai báo trong phần giao tiếp).

Các mô tả nhãn, hằng, kiểu dữ liệu, biến và các hàm, thủ tục nội bộ trong phần hiện thực của một Unit là không dùng được tại nơi sử dụng Unit này. Ta gọi chúng là "bị dấu" (Hidden).

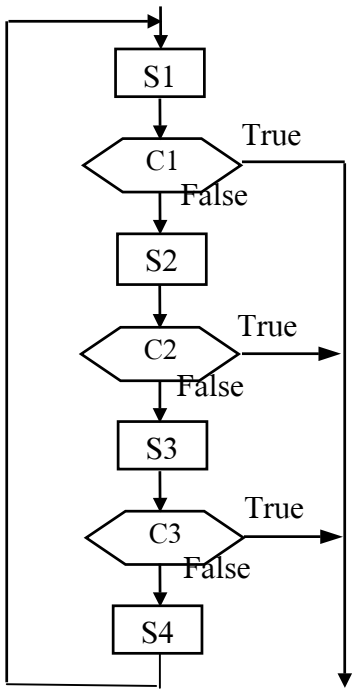
d) Phần khởi động (Initialization section)

```
Begin
    <Các lệnh Pascal>
```

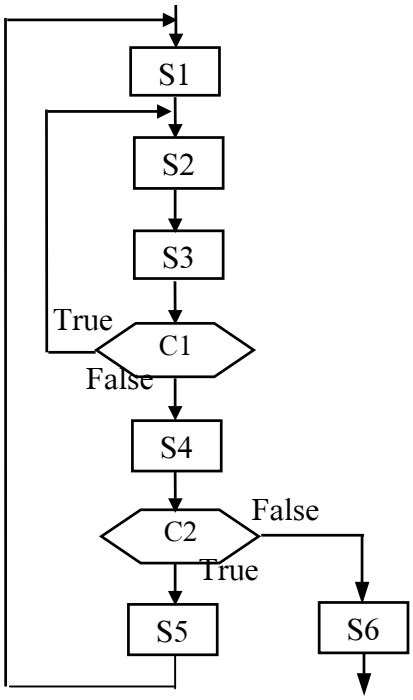
Phần này có thể vắng mặt nhưng **end.** phải có mặt !

```
  Writeln ('trong chương trình chính sẽ sử dụng Unit DTHTRON !')
End.
```

7.



8.



2. **Xử lý chuỗi** (string) : Viết chương trình đọc một câu (kết thúc bởi Enter ↵) sau đó tiến hành các công việc :

- Thống kê số từ, số ký tự trong câu.
- Tách câu ra thành các từ cách nhau bởi dấu cách (space). In ra các từ này.
- Nén câu (bỏ các dấu cách giữa các từ). In kết quả.
- Thay thế những xuất hiện của câu con S1 thành câu con S2 và in kết quả. (S1 và S2 là các câu con nhập vào)

Yêu cầu sử dụng Unit cho mỗi việc. Chương trình chính dùng menu để gọi.

3. Viết chương trình xử lý ma trận vuông A cấp n x n dưới dạng menu, gồm các việc sau :

- Đọc vào ma trận vuông cấp n x n.
- Tính định thức của ma trận.
- Kiểm tra tính đối xứng qua đường chéo chính của ma trận.
- Xác định xem ma trận có dạng tam giác trên không ? (các phần tử phía dưới đường chéo chính đều = 0)
- Xác định xem ma trận có dạng tam giác dưới không ? (các phần tử phía trên đường chéo chính đều = 0)

Yêu cầu dùng Unit. Bốn việc sau cùng chỉ có hiệu lực khi ma trận đã được đọc.

Quy tắc điều kiện sau :

$$\frac{E\ \{P\}\ S \qquad S \to S'}{\therefore\ E\ \{P\}\ S'}$$

Quy tắc "hoặc" :

$$\frac{E\ \{P\}\ S \qquad E'\ \{P\}\ S}{\therefore\ E \vee E'\ \{P\}\ S \wedge S'}$$

Ví dụ5 :

Chứng minh chương trình con P tính tích hai số nguyên *m* và *n* là đúng :

```
P ≡ function product(m, n: Integer): Integer;
      begin
      {P1 ≡} if n < 0 then a:= –n else a:= n;
      {P2 ≡} k:= 0; x:= 0;
      {P3 ≡} while k < a do begin
                  x:= x + m;
                  k:= k + 1
             end;
      {P4 ≡} if n < 0 then product:= –x
             else product:= x
      end;
```

Ta sẽ chứng minh rằng sau khi thực hiện P thì hàm trả về giá trị là *mn*.

Ta chia P gồm bốn đoạn CT là {P1; P2; P3; P4} như trên.

Gọi E là điều kiện đầu E ≡ «*m, n nguyên*» và S1 ≡ E ∧ (a = |n|).

Khi đó có thể chỉ ra E {P1} S1 là đúng.

Gọi S2 ≡ S1 ∧ (k = *0*) ∧ (x = *0*). Dễ dàng kiểm tra rằng S1 {P2} S2 là đúng.

Ta cũng thấy điều kiện ($x = mk$) ∧ ($k \le a$) là một bất biến trong vòng lặp P3 tương tự với lý luận quy nạp trong vòng lặp tính *n!*. Vòng lặp này kết thúc sau a bước lặp khi $k = a$, tức $x = ma$ tại điểm này.

Gọi S3 ≡ ($x = ma$) ∧ (a = [n]). Từ đó suy ra S2 {P3} S3 là đúng.

Cuối cùng có thể chỉ ra P4 là đúng với điều kiện đầu S3 và điều kiện cuối S, với S ≡ *product* = *mn*. Vậy S3 {S4} S đúng và hàm trả về giá trị là *mn*.

Từ các mệnh đề E {P1} S1, S1 {P2} S2, S2 {P3} S3 và S3 {S4} S là đúng, theo quy tắc hợp thành, ta có P cũng đúng, tức E { P } S đúng.

Ngoài ra do cả P1, P2, P3 và P4 đều dừng nên P cũng dừng $_{(qed)}$.

Theo các quy tắc điều kiện trước và điều kiện sau, chỉ cần chứng minh :

(I) $P1 \wedge (a\geq 0) \wedge (b>0)$ {Div} $P1 \wedge (a=bq+r) \wedge (q\geq 0) \wedge (r\geq 0) \wedge (r<b)$

Ta sẽ chứng minh hai giai đoạn :

1. P1 {Div} P1

2. $(a\geq 0) \wedge (b>0)$ {Div} $(a=bq+r) \wedge (q\geq 0) \wedge (r\geq 0) \wedge (r<b)$

❶ ***P1 {Div} P1***

Trước hết cần chỉ ra P1 là bất biến của vòng lặp. Theo quy tắc gán, ta có :

P1 { q :=q + 1 } P1

P1 { r := r – b } P1

như vậy, theo quy tắc tổ hợp :

P1 { r := r – b ; q := q + 1 } P1, nhưng :

$P1 \wedge (r\geq b) \rightarrow P1$

Theo quy tắc điều kiện trước :

$P1 \wedge (r\geq b)$ { r := r – b ; q :=q + 1 } P1

Theo quy tắc vòng lặp :

(1) P1 { while r≥b do begin r := r – b ; q :=q + 1 end } $P1 \wedge (r<b)$

Áp dụng lần nữa quy tắc gán và quy tắc tổ hợp, ta có :

(2) P1 { r := a ; q := 0 } P1

Từ (1) và (2), theo quy tắc tổ hợp, ta có :

P1 {Div} $P1 \wedge (r<b)$

Cuối cùng, theo quy tắc điều kiện sau :

P1 {Div} P1 (*đpcm*)

❷ ***$(a\geq 0) \wedge (b>0)$ {Div} $(a=bq+r) \wedge (q\geq 0) \wedge (r\geq 0) \wedge (r<b)$***

Đặt :

$P3 = (a=bq+r) \wedge (q\geq 0) \wedge (r\geq 0)$

$P4 = (a=b(q+1)+r) \wedge (q+1\geq 0) \wedge (r\geq 0)$ bằng cách thay q trong P3 bởi q+1

$P5 = (a=b(q+1)+r–b) \wedge (q+1\geq 0) \wedge (r–b \geq 0)$ bằng cách thay r trong P4 bởi r–b

(2.1.) Ta xem rằng P3 là một bất biến của vòng lặp. Theo quy tắc gán :

P4 { q := q + 1 } P3

và P5 { r := r – b } P4

Từ đó theo quy tắc tổ hợp :

(3) P5 { r := r – b ; q := q + 1 } P3

Xét điều kiện trước, ta có :

(i) P5 ~ (a=bq+r) ∧ (q≥ –1) ∧ (r≥b)

(ii) Vì rằng :

(q≥0) → (q≥ –1) và

(r≥0) ∧ (r≥b) → (r≥b), ta có :

P3 ∧ (r≥b) → (a=bq+r) ∧ (q≥ –1) ∧ (r≥b)

Theo quy tắc điều kiện trước, phát biểu (3) sẽ là :

P3 ∧ (r≥b) { r := r – b ; q := q + 1 } P3

Theo quy tắc vòng lặp dẫn đến :

(4) P3 { while r≥b do begin r := r – b ; q := q + 1 end } P3 ∧ (r<b)

(2.2.) Bây giờ sử dụng bất biến vòng lặp như là điều kiện sau cho những lệnh đầu tiên của dãy. Ta có quy tắc gán :

(a=b.0+r) ∧ (0≥0) ∧ (r≥0) { q := 0 } P3

và (a=b.0+r) ∧ (0≥0) ∧ (r≥0) ~ (a=r) ∧ (r≥0)

Áp dụng cho lần nữa quy tắc gán :

(a=a) ∧ (a≥0) { r := a } (a=r) ∧ (r≥0) và :

(a=a) ∧ (a≥0) ~ (a≥0)

Theo quy tắc tổ hợp, ta có :

(5) (a≥0) { r := a ; q := 0 } P3

Quy tắc tổ hợp áp dụng cho (4) và (5) cho ta :

(a≥0) {Div} P3 ∧ (r<b)

Quy tắc điều kiện sau dẫn đến kết luận :

(a≥0) ∧ (b>0) {Div} P3 ∧ (r<b)

Từ ❶ và ❷ đã chứng minh, suy ra (I) theo quy tắc "và", đồng thời kết thúc việc chứng minh tính đúng đắn từng phần của Div.

$$Q_{b,\,w,\,r} = E_1 \wedge (1 \le b \le N) \wedge (w-1 \le r \le N) \wedge P_{b,\,w,\,r}$$

Giả sử A_1 là vòng lặp while của chương trình (I) và A_2 là thân của vòng lặp này.

b) Chứng minh tính đúng đắn từng phần

Bổ đề : $x \le \alpha \le y \to A(\alpha)$ tương đương với $(x \le \alpha \le y-1 \to A(\alpha)) \wedge (x \le y \to A(y))$

Chứng minh :

$x \le \alpha \le y \to A(\alpha)$

$\sim (x \le y) \to A(x) \wedge \ldots \wedge A(y)$ theo định nghĩa

$\sim ((x < y) \to A(x) \wedge \ldots \wedge A(y)) \wedge ((x = y) \to A(y))$

$\sim ((x < y) \to A(x) \wedge \ldots \wedge A(y-1)) \wedge ((x < y) \to A(y)) \wedge ((x = y) \to A(y))$

$\sim x \le \alpha \le y-1 \to A(\alpha)) \wedge (x \le y \to A(y))$ đpcm

(α) Chứng minh của bất biến $Q_{b,\,w,\,r}$ trong A_2

Ta kiểm tra rằng $E_1 \wedge (1 \le i \le N) \wedge (1 \le j \le N)$ { hoán vị (i, j) } E_1

(i) Trường hợp W(w)

Rõ ràng ta có : $Q_{b,\,w+1,\,r}$ { w := w+1 } $Q_{b,\,w,\,r}$

Vả lại : $Q_{b,\,w,\,r} \wedge (w \le r) \wedge W(w) \sim E_1 \wedge (1 \le b \le w) \wedge (w \le r \le N) \wedge P_{b,\,w,\,r} \wedge W(w)$

và : $P_{b,\,w,\,r} \wedge W(w) \to P_{b,\,w+1,\,r}$

như vậy : $Q_{b,\,w,\,r} \wedge (w \le r) \wedge W(w) \to Q_{b,\,w+1,\,r}$

Điều này chứng minh : $\boxed{Q_{b,\,w,\,r} \wedge (w \le r) \wedge W(w)\ \{\ w := w+1\ \}\ Q_{b,\,w,\,r}}$

(ii) Trường hợp B(w)

Rõ ràng ta có : $Q_{b+1,\,w+1,\,r}$ { b := b+1 ; w := w+1 } $Q_{b,\,w,\,r}$

Vả lại :

$$P_{b+1,\,w+1,\,r} = (1 \le \alpha < b+1 \to B(\alpha)) \wedge (b+1 \le \alpha < w+1 \to W(\alpha)) \wedge (r \le \alpha < N \to R(\alpha))$$

Từ đó theo bổ đề :

$$P_{b+1,\,w+1,\,r} \sim (1 \le \alpha < b+1 \to B(\alpha)) \wedge (b+1 \le \alpha < w \to W(\alpha)) \wedge (r \le \alpha < N \to R(\alpha)) \wedge ((b \ge 1) \to B(b)) \wedge ((w \ge b+1) \to W(\alpha))$$

Áp dụng tiên đề định nghĩa bởi phép hoán vị cho $Q_{b+1,\,w+1,\,r}$:

$$F\ \{\ \text{hoán vị}(b, w)\ \}\ Q_{b+1,\,w+1,\,r}$$

với :

$F = E_1 \wedge (1 \le b+1 \le w+1) \wedge (w \le r \le N)$

$\wedge (1 \le \alpha < b \rightarrow B(\alpha)) \wedge (b+1 \le \alpha < w \rightarrow W(\alpha))$

$\wedge (r < \alpha \le N \rightarrow R(\alpha)) \wedge ((b \ge 1) \rightarrow B(w))$

$\wedge ((w \ge b+1) \rightarrow W(b)) \wedge (1 \le b \le N) \wedge (1 \le w \le N)$

Nhưng :

$(b+1 \le \alpha < w \rightarrow W(\alpha)) \wedge ((w \ge b+1) \rightarrow W(b)) \sim (b \le \alpha < w \rightarrow W(\alpha))$

Như vậy :

$F \sim E_1 \wedge (1 \le b \le w) \wedge (w \le r \le N) \wedge B(w) \wedge P_{b, w, r}$

Từ đó suy ra :

$Q_{b, w, r} (w \le r) \wedge \neg W(w) \wedge B(w) \rightarrow F$

Điều này chứng minh :

$$Q_{b, w, r} \wedge (w \le r) \wedge \neg W(w) \wedge B(w) \{ \text{hoánvị}(b, w) ; b := b+1 ; w := w+1 \} Q_{b, w, r}$$

(iii) Trường hợp R(w)

Ta có :

$Q_{b, w, r-1} \{ r := r -1 \} Q_{b, w, r}$

Mặt khác theo bổ đề :

$P_{b, w, r-1} \sim P_{b, w, r} \wedge ((r \le N) \rightarrow R(r))$

Áp dụng tiên đề định nghĩa bởi phép hoán vị cho $Q_{b, w, r-1}$:

$$G \{ \text{hoánvị}(r, w) \} Q_{b, w, r-1}$$

với :

$G = E_1 \wedge (1 \le b \le w) \wedge (w-1 \le r-1 \le N) \wedge P_{b, w, r} \wedge ((r \le N) \rightarrow R(w))$
$\wedge (1 \le r \le N) \wedge (1 \le w \le N)$

Thực tế ta có :

$P_{b, w, r} \wedge (b \le w) \wedge (w \le r) \wedge (1 \le w \le N) \wedge (1 \le r \le N)$

$$\{ \text{hoánvị}(r, w) \} P_{b, w, r} \wedge (b \le w) \wedge (w \le r)$$

Ta có :

$G \sim E_1 \wedge (1 \le b \le w) \wedge (w \le r \le N) \wedge R(w) \wedge P_{b, w, r}$

Vả lại :

$E_1 \wedge \neg W(w) \wedge \neg B(w) \wedge (1 \le w \le N) \rightarrow R(w)$

Như vậy :

$Q_{b, w, r} \wedge (w \le r) \wedge \neg W(w) \wedge \neg B(w) \rightarrow G$

Điều này chứng minh :

$$Q_{b, w, r} \wedge (w \le r) \wedge \neg W(w) \wedge \neg B(w) \{ \text{hoánvị}(r, w) ; r := r - 1 \} Q_{b, w, r}$$

Từ 3 kết quả trên ta suy ra :

$$Q_{b, w, r} \wedge (w \le r) \{ A_2 \} Q_{b, w, r}$$

(β) Chứng minh tính đúng đắn của chương trình (I)

Ta đã có :

$$Q_{b, w, r} \{ A_2 \} Q_{b, w, r} \wedge (w > r)$$

(i) Điều kiện sau

Ta có :

$$Q_{b, w, r} \wedge (w > r) \rightarrow (1 \le b \le w) \wedge (w=r+1) \wedge (r \le N) \wedge P_{b, w, r}$$

và :

$P_{b, w, r} \wedge (w=r+1) \rightarrow (1 \le \alpha < b \rightarrow B(\alpha)) \wedge (b \le \alpha < r \rightarrow W(\alpha)) \wedge (r < \alpha \le N \rightarrow R(\alpha))$

Như vậy :

$$Q_{b, w, r} \wedge (w > r) \rightarrow S_1$$

trong đó :

$$Q_{b, w, r} \{ A_1 \} S_1$$

(ii) Khởi gán

Ta có :

$$E_1 \wedge (0 \le n \le N) \wedge (n < \alpha \le N \rightarrow R(\alpha)) \{ w := 1; b := 1; r := n \} Q_{b, w, r}$$

vả lại :

$$(n=N) \rightarrow (0 \le n \le N) \wedge (n < \alpha \le N \rightarrow R(\alpha)),$$

do đó :

$$E_1 \wedge (n=N) \{ \text{chương trình (I)} \} S_1$$

Như vậy ta đã chứng minh xong tính đúng đắn từng phần.

c) Chứng minh dừng

Chương trình (I) chỉ có một vòng lặp A_1 và thân của vòng lặp là A_2.

Giả sử : $W_{Q_{bwr} \wedge (w \le r)} = X_1$ và hàm $m : X_1 \rightarrow N$ sao cho $m(x) = r - w$.

```
pfpre(if B then P else Q, S)
        ~if pfpre(if B then , true)
        then (B ∧ pfpre(P, S) ∨ (¬B ∧ pfpre(Q, S)) else false
```

Không có quy tắc đơn giản cho vòng lặp while.

Độ phức tạp của các quy tắc đưa ra để xác định dkt của một vòng lặp while có thể được minh hoạ như sau :

– Vấn đề dừng của vòng lặp while trừu tượng không là quyết định được.
Vì vậy, điều kiện

```
pfpre(while B do P, true)
```

không tính được cho mọi B, mọi P.

– Việc giải quyết một số bài toán nổi tiếng đưa về việc chứng minh một chương trình là dừng. Ví dụ, người ta không biết nếu :

```
pfpre(while n<>1 do
   n:=if not odd(n) then n div 2 else 3*n+1, true)~(n >= 1)
```

Đây là sự giả định (conjecture) của Collatz.

Chú ý :Các tiên đề và các quy tắc suy diễn xét ở mục I cho phép chứng minh tính đúng đắn của các phát biểu E {P} S, trong đó E ≠ pfpre(P, S). Ví dụ, ta đã chứng minh tính đúng đắn từng phần rằng :

$$(a \geq 0)\ \{Div\}\ (a = bq + r) \wedge (q \geq 0) \wedge (0 \leq r < b)$$

b) Trường hợp điều kiện sau mạnh nhất

Theo định nghĩa của pfpost, ta có : E {P} S ~ (pfpost(E, P) → S)

Như vậy, một hệ thống chứng minh tính đúng đắn dựa trên việc tính toán các điều kiện sau mạnh nhất cho phép chứng minh tính đúng đắn *từng phần*.

Đối với lệnh điều kiện, quy tắc suy diễn được cho bởi tính chất :

```
pfpost(E, if B then P else Q)
~ pfpost(E ∧ B, P) ∧ pfpost(E ∧ ¬B,Q)
```

với giả trị rằng B luôn luôn cho một kết quả.

Trong trường hợp tổng quát :

```
pfpost(E, if B then P else Q)
   ~ pfpost(E ∧ (if pfpre(if B then, true) then B else false), P)
     ∨ pfpost(E ∧ (if pfpre(if B then, true) then ¬B else
   false), Q)
```

Ta có thể dùng quy tắc để chứng minh chương trình theo chiều bắt đầu – kết thúc.

Ví dụ, để chứng minh $(x > 0)\ \{ x := x*2 \}\ (x > 0)$:

- Xét quy tắc từ điều kiện sau về điều kiện trước :
  $(2x > 0)\ \{ x := x*2 \}\ (x > 0)$

Sử dụng quy tắc điều kiện trước, ta nhận được kết quả vì :
$(2x > 0) \rightarrow (x > 0)$

- Xét quy tắc từ điều kiện trước về điều kiện sau :

  $(x > 0)\ \{ x := x*2 \}\ (\frac{x}{2} > 0)$

  Sử dụng quy tắc điều kiện sau, ta nhận được kết quả vì : $(\frac{x}{2} > 0) \rightarrow (x > 0)$

Chú ý : Các tiên đề gán đã đưa ra trong các ví dụ trên có thể nhận được từ điều kiện sau bởi tiên đề gán ở mục I. Trong các ví dụ này, hàm $f_{x := <bt>} : W \rightarrow W$ là đơn ánh.

Ví dụ sau đây chỉ ra rằng nếu hàm $f_{x := <bt>} : W \rightarrow W$ không là đơn ánh, người ta không còn có tính chất này.

Ví dụ 17 :

Hàm $f_{x := x \text{ div } 2}$ là toàn thể, nhưng không đơn ánh. Thu hẹp của nó vào $W_{x = 2y}$ là đơn ánh.

Áp dụng quy tắc tính pfpost, ta có tiên đề : $(x = 2y)\ \{ x := x \text{ div } 2 \}\ (2x = 2y)$

tương đương với : $(x = 2y)\ \{ x := x \text{ div } 2 \}\ (x = y)$

Tuy nhiên, xuất phát từ điều kiện sau $(2x = 2y)$ và sử dụng tiên đề gán ở mục I, ta có tiên đề :

$$(2\lfloor\frac{x}{2}\rfloor = 2y)\ \{ x := x \text{ div } 2 \}\ (2x = 2y)$$

tương đương với : $(2\lfloor\frac{x}{2}\rfloor = 2y)\ \{ x := x \text{ div } 2 \}\ (x = y)$

b) Trường hợp tổng quát

Nếu không tồn tại hàm ngược của $f_{x := <bt>}$, sẽ không thể tìm được giá trị ban đầu của x, xuất phát từ giá trị các biến sau khi thực hiện phép gán.

Một số kịch bản coi trọng việc tìm lỗi sai và khuyến khích việc chạy demo trực tiếp mã chương trình (hand made) nguồn : khảo sát mang đến cuộc họp cách tiến hành và các dữ liệu liên quan để mọi người tiến hành thử nghiệm. Người ta còn gọi cách thử nghiệm như vậy là *walk throughs* (chạy suốt).

Một số kịch bản lại coi trọng việc chứng minh không hình thức : khảo sát đề nghị xác minh các tính chất cho phép thử nghiệm tính đúng đắn của sản phẩm. Người ta nói đây là việc khảo sát căn cứ trên việc xác minh.

Việc kiểm lại (review) khác với khảo sát vì rằng việc kiểm lại không đòi hỏi phải họp : Sản phẩm được giao cho những người không tham gia vào việc lập trình, họ có những khuynh hướng đánh giá độc lập.

Có thể nói phương pháp khảo sát có hiệu quả đáng kể : những số liệu tìm thấy trong các văn bản ghi nhận khoảng 50% sai sót được phát hiện khi khảo sát. Những con số dưới đây (lấy từ tạp chí IEEE[3] năm 1992 của Dyer M. từ bài báo "Verification Based Inspection") cho thấy các sai số tìm thấy khi phát triển dự án 5 phần mềm của hãng IBM :

Dự án	*Khảo sát thiết kế toàn bộ*	*Khảo sát thiết kế chi tiết*	*Khảo sát mã*	*Thử nghiệm đơn vị*	*Thử nghiệm hệ thống*
1			50	25	25
2	4	13	49	17	17
3	20	27	10	20	23
4	20	26	22	18	36
5	10	18	24	24	24

Một phương pháp khác, gọi là *phương pháp phòng sạch* (Clean-room Methodology), thay vì thử nghiệm (testing), khuyến khích việc khảo sát (inspection) bằng cách xác minh (verification) trong quá trình sản xuất phần mềm. Sự phát triển phần là liên tiếp làm mịn (raffinement) sản phẩm. Mỗi giai đoạn, người ta tiến hành chứng minh tính đúng đắn (prouving) một cách chặt chẽ, đồng thời với các cuộc khảo sát, sao cho sản phẩm phần mềm không chứa sai sót.

Việc thử nghiệm chỉ được tiến hành khi xác minh hậu nghiệm (a posteriori) nhờ các phương pháp thống kê, nhằm đạt được mục đích đặt ra lúc đầu. Phương pháp phòng sạch do H.Mills xây dựng tại IBM, đã được áp dụng để sản xuất các phần mềm cỡ lớn.

[3] IEEE, đọc là eye-triple-ee, tên viết tắt của Institute of Elechtrric and Engineers.

- Mỗi *phép thử nghiệm* chỉ ra hoạt động từ việc thiết kế các tập dữ liệu thử, tiến hành thử nghiệm và đánh giá kết quả đến các giai đoạn khác nhau trong chu kỳ sống của phần mềm.
- *Người thử nghiệm* (hay *nhóm thử nghiệm*) có kiến thức chuyên môn Tin học có nhiệm vụ tiến hành phép thử nghiệm.
- Một *khiếm khuyết* (failure) xảy ra khi chương trình thực hiện cho ra kết quả không tương hợp với đặc tả của chương trình.
- Một lỗi sai (error) là một phần chương trình (lệnh) đã gây ra khiếm khuyết.

Người thử nghiệm có nhiệm vụ :

1. Tạo ra tập dữ liệu thử.
2. Triển khai các phép thử.
3. Lập báo cáo về kết quả thử nghiệm và lưu giữ.

Mục đích thử nghiệm là để :

1. Chứng minh rằng chương trình là đúng đắn

Để khẳng định tính đúng đắn của chương trình, cần tiến hành các thử nghiệm toàn bộ (exhaustive testing), đòi hỏi tập dữ liệu thử phải hữu hạn và có kích thước vừa phải sao cho đủ sức thuyết phục. Điều này trên thực tế rất khó thực hiện.

Sau đây là một tiêu chuẩn nổi tiếng của Dijkstra : "Các thử nghiệm cho phép chứng minh một chương trình là không đúng, bằng cách chỉ ra một phản ví dụ, tuy nhiên, không bao giờ có thể chứng minh được chương trình đó là đúng đắn".

2. Gây ra những khiếm khuyết của chương trình

Myers G. J. trong bài báo "The Art of Software Testing", Wiley 1979, đã định nghĩa thử nghiệm như sau :

"Phép thử nghiệm là cho chạy chương trình nhằm tìm ra những sai sót".

Từ đó, thường người ta nói về "thử nghiệm phá hủy" (destructive testings). Mục đích của những phép thử như vậy là tập trung tìm ra các lỗi sai từ những khiếm khuyết do người lập trình phạm phải. Người thử nghiệm tiến hành với mục đích nghịch (negative) : phép thử là thành công nếu tìm ra được khiếm khuyết, là thất bại trong trường hợp ngược lại. Việc thử nghiệm kiểu này thường được áp dụng trong quá trình viết chương trình.

3. Đưa ra đánh giá tĩnh (static evaluation - static benchmark) **về chất lượng của chương trình.**

Người ta sử dụng phương pháp "thử nghiệm tĩnh" (static testing) cho mục đích này. Trong phương pháp phòng trắng, người ta chỉ tiến hành những phép thử tĩnh,

c) Phủ các điều kiện

Ta xét một chương trình chứa cấu trúc rẽ nhánh logic gồm các lệnh not, end và or. Một phép thử phủ các điều kiện nếu việc thực hiện chương trình kéo theo sự tính giá trị của biểu thức này cho mọi giá trị logic có thê. Như vậy một biểu thức có hai toán hạng P, Q sẽ được tính toán với :

	A	B
	`true`	`true`
	`true`	`false`
	`false`	`true`
	`false`	`false`

Phép phủ các điều kiện cho phép củng cố phép phủ các quyết định. Ví dụ có thể phủ các quyết định bằng cách thực hiện phép lựa chọn P và Q với :

P = **`true`**, Q = **`true`** và P = **`false`**, Q = **`false`**,

điều này không cho phép phân biệt phép rẽ nhánh A or B.

d) Phủ các lộ trình thực hiện chương trình (path testing)

Một phép thử phủ các lộ trình chạy chương trình nếu gây ra việc thực thi mỗi lộ trình thực hiện chương trình. Không tồn tại phép thử như vậy nếu chương trình có vô hạn lộ trình thực hiện trong trường hợp tổng quát. Thông thường người ta xây dựng phép thử phủ các lộ trình thực hiện có số lượng ≤ một hằng đã cho.

e) Xác định dữ liệu cho phép phủ lộ trình thực hiện đặc biệt

Giả thiết rằng với mọi lệnh P của chương trình và mọi quyết định S, có thể tính ptpre (P, S), điều kiện đầu yếu nhất ứng với P và S. Người ta có thể với mọi lộ trình của chương trình, tính được một công thức E sao cho các dữ liệu của chương trình thỏa mãn E nếu và chỉ nếu việc thực hiện của chương trình đi theo lộ trình đã chọn.

Đặc biệt, E không là sai nếu và chỉ nếu lộ trình đã chọn là lộ trình thực thi. Như vậy chỉ cần tìm ra các dữ liệu làm thỏa mãn E để có phép thử phủ lộ trình đã chọn. Điều này có thể thực hiện bằng ta, hay chứng minh một cách sáng tạo công thức xE.

Phương pháp này được dùng để định nghĩa phép thử phủ các quyết định của một chương trình :

- Lựa chọn một tập hợp các lộ trình phủ các quyết định.
- Với mỗi lộ trình, tính điều kiện đầu yếu nhất tương ứng (hoặc một điều kiện đầu mạnh hơn).
- Tìm các dữ liệu thỏa mãn các điều kiện điều này.

Trong những điều kiện sử dụng đã cho, sự xuất hiện thường xuyên các khiếm khuyết do các sai sót khác nhau gây ra là rất biến động : một số sai sót gây ra thường xuyên các khiếm khuyết, những sai sót khác thì rất hiếm, có thể không bao giờ xảy ra trên thực tế.

Việc thực thi một phần mềm với một dữ liệu cố định trước là một quá trình có tính xác định gây ra hoặc là một kết quả đúng, hoặc là một khiếm khuyết. Nếu người ta ở trong những điều kiện sử dụng chương trình, mỗi dữ liệu có thể được của chương trình sẽ cho một xác suất nào đó.

Tập hợp các dữ liệu cùng xác suất sử dụng như vậy được gọi là một mẫu sử dụng (use pattern) của chương trình. Từ một mặt cắt sử dụng đã cho, người ta định nghĩa xác suất một lần chạy cho một kết quả đúng và xác suất một khiếm khuyết, còn được gọi là tỷ suất khiếm khuyết.

Với một mô hình độc lập với thời gian, người ta định nghĩa độ tin cậy (reliability) của một chương trình như là xác suất của sự kiện " lần chạy sau của chương trình là đúng ", nghĩa là 1,xác suất của một khiếm khuyết.

Với một mô hình phụ thuộc thời gian, người ta định nghĩa độ tin cậy như là một xác suất của sự kiện "chương trình chạy đúng đắn trong thời gian t". Lúc này độ tin cậy là một hàm của thời gian.

Các mô hình phụ thuộc vào thời gian thường được sử dụng cho các phần mềm tương hỗ (như là các hệ điều hành). Tiếp theo đây, người ta sẽ chỉ khai triển các mô hình độc lập với thời gian.

Khi xuất hiện một khiếm khuyết, nếu là một khiếm khuyết về phần cứng, thì phải sửa chữa, nếu là một khiếm khuyết về phần mềm thì phải chạy trình sửa lội debugger.

Sửa chữa các hư hỏng thuộc về phần cứng là để thay thế những chi tiết hư hỏng, thiết lập lại sự vận hành ổn định của thiết bị như trước. Còn chạy trình debugger là để sửa các lỗi về thiết kế, tăng độ tin cậy của phần mềm.

Thường người ta sử dụng đại lượng liên quan đến độ tin cậy là số lần sử dụng trung bình cho đến khi xảy ra khiếm khuyết (đối với mô hình độc lập với thời gian), hoặc sử dụng sau một thời gian trung bình nào đó đến khi xảy ra khiếm khuyết (đối với mô hình phụ thuộc vào thời gian).

Đại lượng liên quan đến độ tin cậy MTTF (Mean Time To Failure : thời gian trung bình để xảy ra khiếm khuyết) được tính như sau :

Trong mô hình độc lập với thời gian :

Độ ổn định = xác suất một lần chạy đúng.

= 1 - xác suất một khiếm khuyết.

a) Cách giải phi hình thức

Chuyển n - 1 đĩa từ A qua C lấy B làm cột trung gian, sau đó chuyển đĩa dưới cùng từ A sang B. Tiếp tục chuyển n - 1 đĩa từ C qua B lấy A làm cột trung gian theo cách trên.

b) Cách giải hình thức bằng đặc tả

Gọi thủ tục chuyển n đĩa từ A qua B lấy C làm trung gian (n > 0) là :

Hà_nội (n, A, B, C)

và thủ tục chuyển một đĩa từ A qua B là :

Chuyển_một_đĩa(A, B).

Khi đó, ta có đặc tả :

```
Hà_nội (n, A, B, C) = if n > 0 then begin
                        Hà_nội (n - 1, A, C, B);
                        Chuyển_một_đĩa (A, B);
                        Hà_nội (n - 1, C, B, A)
                      End
```

ta dễ dàng viết các thao tác trên thành thủ tục như sau :

```
Procedure ChuyểnCột(n, A, B, C: TênCột);
  Begin
  if n>0 then begin
     ChuyểnCột(n-1, A, C, B);
     Chuyển_một_đĩa_từ_A_sang_C;
     ChuyểnCột(n-1, B, A, C);
     End
  End;
```

Thao tác *Chuyển_một_đĩa_từ_A_sang_C;* được viết thành lệnh :

```
Writeln('Chuyển một đĩa từ ', A:1, ' -> ', C:1);
```

Thêm biến đếm i để tính số bước chuyển đĩa, chương trình đầy đủ như sau :

```
Program HanoiTower;
Type TênCột = 1 .. 3;
Var i, N: Integer;
Procedure ChuyểnCột(n, A, B, C: TênCột);
  Begin
  if n>0 then begin
     ChuyểnCột(n-1, A, C, B);
     i:=i+1;
     Writeln(i:3,'Chuyển một đĩa từ ',A:1,' -> ',C:1);
     ChuyểnCột(n-1, B, A, C);
     End
  End;
Begin
  Write('Số đĩa cần chuyển : ');
```

```
begin
    i:= 1; { phântử ∉ V[ 1..i - 1] , i ≤ n}
    while (V[ i] <> phântử) and (i < 1) do
    { phântử ∉ V[ 1..i] , i < n}
        i := i + 1; { phântử ∉ V[ 1..i - 1] , i ≤ n}
    { ((V[ i] = phântử) ∨ (i = n), phântử ∉ V[ 1..i - 1] ,
    i ≤ n) ⇒ (V[ i] = phântử, phântử ∉ V)
    ∨ (V[ i] ≠ phântử, phântử ∉ V)}
    Check := (V[ i] = phântử)
    { (check, phântử ∉ V) ∨ (Not check, phântử ∉ V)}
end;
```

Ta có thể viết lại thuật toán dưới dạng đệ quy như sau :

```
function check(V:Vectơ, i,n:integer; phântử: T): Boolean;
{ n ≥ 0 ⇒ check, phântử ∈ V[ i..n] )
∨ (Not check, phântử ∉ V[ i..n] )}
begin
    if i > n then check := false
    else if V[ i] = phântử then check := true
    else check := check (V, i + 1, n, phântử)
end;
```

Khi gọi hàm, i có thể nhận giá trị bất kỳ, từ 1..n, đặc biệt i = 1

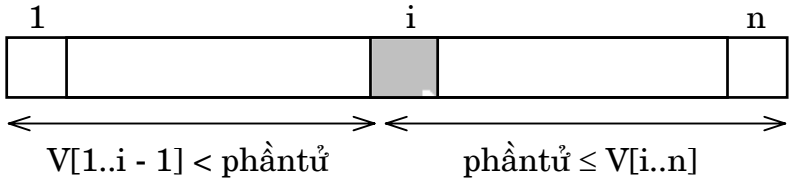
Trường hợp duyệt vectơ từ phải qua trái, ta không cần dùng biến i nữa :

```
function check (V:Vectơ; n: integer; phântử: T): Boolean;
{ n ≥ 0 ⇒ (check, phântử ∈ V) V (Not check, phântử ∉ V)}
begin
if n = 0 then check := false
else if V[ n] = phântử then check := true
    else check := check (V, n - 1, phântử)
end;
```

b) Vectơ được sắp xếp thứ tự

Ta cần tìm chỉ số i ∈ [1..n] sao cho thỏa mãn :

V[1..i - 1] < phântử ≤ V[i..n]



Hình 5.3. Vectơ được sắp xếp thứ tự

Vấn đề là kiểm tra đẳng thức phântử = V[i] không trong V đã được sắp xếp ?

Lập luận :

lúc này ta trở lại bài toán đã xét : tìm phân tử trong vectơ V[1..m - 1] hoặc V[m +1..n]. Kết thúc nếu phân tử = V[m]

Một cách tổng quát, lần lượt xác định dãy các vectơ có V_1, V_2, ..., V_k sao cho mỗi V_i có kích thước nhỏ hơn kích thước của vectơ con trước đó V_{i-1}

Để ý rằng nếu chọn V_1 = V[1..n], V_2 = V[2..n], ..., V_k = V[k..n], ta đi đến phép tìm kiếm tuần tự đã xét ở trên.

Ta sẽ chọn m là vị trí giữa (nếu n lẻ) để cho V[1..m - 1] và V[m + 1..n] có kích thước bằng nhau, hoặc chọn m sao cho chúng hơn kém nhau một phân tử.

Khi đó kích thước của các vectơ thuộc dãy V_1, V_2, ..., V_k sẽ lần lượt được chia đôi tại mỗi bước : n, n/2, ..., $n/2^{k-1}$.

Như vậy, sẽ có tối đa [$\log_2 n$] vectơ con khác rỗng.

Ví dụ : nếu n = 9000, số vectơ con khác rỗng tối đa sẽ là 13, vì 2^{13} = 8192

Xây dựng thuật toán :

Sau một số bước, ta có vectơ con V[inf.. sup] sao cho :

V[1..inf - 1] < phầntử < V[sup + 1..n]

Xảy ra hai trường hợp :

- inf > sup (inf = sup + 1)

  (V[1..inf-1] < phầntử < V[sup+1..n], inf = sup+1) $\Rightarrow$ (phân tử $\notin$ V, kết thúc)

- inf $\leq$ sup : m = (inf + sup) div 2

khi đó ta có V[inf..m - 1] $\leq$ V[m] $\leq$ V[m + 1..sup]

Tồn tại 3 khả năng như sau :

- Phần tử = V[m] : kết thúc, phần tử $\in$ V
- Phần tử < V[m] : tiếp tục tìm kiếm trong V[inf..m - 1]
  lấy sup := m - 1 đê có lại khẳng định phần tử < V[sup + 1..m]
- Phần tử > V[m] : tiếp tục tìm kiếm trong V[m + 1..sup]
  lấy inf := m + 1 để có lại khẳng định V[1..inf - 1] < phần tử

Như vậy cả hai trường hợp : V[1..inf - 1] < phần tử < V[sup + 1..n]

Khởi đầu, lấu inf := 1 và sup := n

Ta có thuật toán như sau :

```
function binary (V:vectơ; n:integer; phântử:T): Boolean;
{ V được SXTT ⇒ (binary, phântử∈V)∨(not binary, phântử∉V)}
Var inf, sup, m : integer ;
OK : Boolean ;
begin
   OK : false ; { not OK, phân tử ∉ V}
```

```
    inf := 1 ; sup := n ;
    { V[ 1..inf - 1] < phântử < V[ sup+1..n] }
    while (inf ≤ sup) and (not OK) do begin
      m := (inf + sup) div 2 ;
      if V[ m] = phântử then OK := true { OK, phântử ∈ V}
      else { not OK}
         if V[ m] <phântử then inf:= m+1 { V[ 1..inf-1] <phântử}
         else sup := m - 1 ; { V[ sup + 1..n] > phântử}
         { (V[ 1..inf - 1] < phântử < V[ sup + 1..n] , not OK)
         ∨ (tìm thấy, phântử ∈ V)}
    end;
    { (inf = sup + 1) ∨ (tìm thấy),
    (¬ tìm thấy, V[ 1..inf - 1] < phântử < V[ sup + 1..n] ) ∨
    (tìm thấy, phântử ∈ V) ⇒
    (¬ tìm thấy, V[ 1..inf - 1] < phântử < V[ inf..n] ) ∨
    (tìm thấy, phântử ∈ V) ⇒
    (¬ tìm thấy, phântử ∉ V) ∨ (tìm thấy, phântử ∈ V)}
    Nhị phân := OK
end ;
```

Viết chương trình trên dưới dạng đệ quy :

```
function NhịPhân(V:Vectơ;inf,sup:integer;phântư:T):boolean;
{ (V được SXTT ⇒ (NhịPhân, phântử∈V) ∨ ¬ NhịPhân,phântử∉V)}
Var m : integer ;
begin
   if inf > sup then NhịPhân:= false
   else begin
     m := (inf + sup) div 2 ;
     if V[ m] = phântử then NhịPhân:= true
     else if V[ m] < phântử then
        NhịPhân:= NhịPhân(V, m+1, sup, phântử)
     else NhịPhân:= NhịPhân(V, inf, m - 1, phântử)
   end
end ;
```

Hàm này có thể được gọi với các giá trị inf, sup bất kỳ, thông thường được gọi bởi dòng lệnh :

```
NhịPhân (V, 1, n, phântử)
```

b) Phương án 2

Có thể tìm ra những phương án khác cho thuật toán tìm kiếm nhị phân. Chẳng hạn, thay vì kiểm tra đẳng thức V[m] = phântử, ta kiểm tra khẳng định :

```
V[ 1..inf - 1] < phântử ≤ V[ inf..n]
```

Ssau đó kiểm tra phântử = V[inf] để có câu trả lời.

Mặt khác, có thể thay đổi giá trị trả về của hàm tìm kiếm nhị phân bởi vị trí của phân tử trong vectơ, bằng 0 nếu phân tử ∉ V.

Nếu inf = 1, khẳng định có dạng V[1..0] < phầntử ≤ V[sup..n] và được viết gọn phầntử ≤ V[1..n].

```
function NhịPhân(V:vectơ;n:integer;phântử: T): integer ;
{ (V được SXTT, n > 0) ⇒ (m ∈ [ 1..n]
NhịPhân = m, V[ m] = phântử) V (NhịPhân = 0, phân tử ∉ V)}
Var m, inf, sup : integer ;
begin
   if phân tử > V[ n] then nhị phân := 0
   else begin
      inf := 1 ; sup := n ;
      { V[ 1..inf - 1] < phântử ≤ V[ sup..n]}
      while inf < sup do begin
         m := (inf + sup) div 2 ;
         if phầntữ ≤ V[ m] do sup := m { phâtử ≤ V[ sup..n]}
         else inf := m + 1 { V[ 1..inf - 1] < phân tử}
         { V[ 1..inf - 1] < phân tử ≤ V[ sup..n]}
      end ;
       { (inf = sup, V[ 1..inf - 1] < phân tử ≤ V[ inf..n]
       ⇒ (V[ 1..inf - 1] < phântử ≤ V[ inf..n]}
       if phântử = V[ inf] then NhịPhân:= inf
       else NhịPhân:= 0
    end
end ;
```

Tên các tập hợp và các phép toán trên tập hợp xác định một ký dấu (signature). Như vậy một dấu kí được xây dựng từ :

- Tên các kiểu đặc tả

- Tên các phép toán với việc chỉ rõ miền xác định (domain) và miền trị (range) như sau :

tên phép toán : miền xác định → miền trị

Ta xây dựng dấu kí từ kiểu string như sau

```
Adt String ;
Use char, Not, Bool ;
Sorts string ;
Operations
new : → string ;
append _ _ : String, string → string ;
add _ to _ : char, string → string ;
# _ : String → not ;
is empty ? _ string → bool ;
_ = _ : string, string → bool ;
frist _ : string → char ;
```

Tên xuất hiện trong một dấu kí gồm hai loại là có ích (internest) và bổ trợ (auxiliary) tùy theo vai trò của chúng. Ví dụ :

- String là có ích

- Char, not và bool là bổ trợ

Cú pháp (cp)

Cp đặc tả đại số sử dụng trong ví dụ trên được chia ra thành các khối : đầu, giao tiếp và thân của đặc tả. Mỗi khối gồm một số khai báo ngăn cách nhau bởi các từ khóa (có gạch chân)

Đối với khối giao tiếp (interface), người ta sử dụng các khái niệm tiền tố (prefix), trung tố (infix) và hậu tố (postfix) như sau :

Tiền tố : tên của phép toán được đặt trước dãy các tham biến

Ví dụ : appenend _ _ : string, string → string ;

Từ đó người ta có thể viết các hạng dưới dạng :

append x y hay

append (x y) hay

(append x y)

Trung tố : cho phép định nghĩa toán tử hay vị từ

Ví dụ _ = _ : string, string → bool ;

từ đó có thể viết các hạng dưới dạng :

FV (x) = {x}

Ví dụ : FV (append (is empty ? (new), add c to x)) = {x, c}

Phép thế trong một hạng t cho các thành phần chứa biến x bởi hạng u, ký hiệu t [u /x], được định nghĩa như sau :

Với $x \in FV(t)$ thì

$(f\ t_1\ t_2 \ldots t_n)\ [u/x] = (f\ t_1\ [u/x]\ t_2\ [u/x] \ldots t_n\ [u/x])$

y [u/ x] = u y = x

= y $y \neq x$

Ví dụ : append (is empty ? (new), (add c to x)) [(add c to y) / x]

= append (is empty ? (new), (new), (add c to (add c' to y)))

Mô tả các thuộc tính qua các phương trình

Các tiên đề sử dụng trong đặc tả được xây dựng theo logic vị trí bậc 1 dạng phương trình (pt)

Một phương trình hợp thức có vế trái và vế phải cùng kiểu hạng :

AX spec = $\{t = t' \mid t : s \wedge t' : s\}$

Trong ví dụ về đa kiểu string, phép toán is empty ? được định nghĩa theo phương trình :

is empty ? (new) = true ;

Có nghĩa một xâu vừa mới tạo ra là rỗng - sau đó, việc thêm một ký tự mới vào xâu sẽ cho kết quả là false :

is empty ? (add c to x) = false ;

Tính đệ quy của phương trình :

append (x, add c to y) = add c to (append (x, y)) ;

chỉ ra rằng việc ghép một xâu với xâu được tạo ra bằng cách thêm một ký tự vào xâu này thì cũng có giá trị như ghép hai xâu trước rồi sau đó thêm một ký tự vào xâu kết quả. Điều đó hợp lý vì ta có tính chất của phương trình : append (x, new) = x ;

nghĩa là ghép một xâu nào đó với xâu rỗng cũng cho ra kết quả chính xâu đó

Ta có các tiên đề về xâu ký tự như sau :

Axioms

is empty ? (new) = true ;

is empty ? (add c to x) = false ;

new = 0 ;

(add c to x) = x (x) = + 1;

not (false) ??? true

true and b ??? b

false and b ??? false

true or b ??? true

false or b ??? b

false xor b ??? not (b)

Ví dụ : sử dụng các quy tắc trên để viết lại hạng sau đây :

not (false or (true and false))

not (true and false) not (false or false)

not (false)

true

Tuy nhiên, nguyên lý hướng về viết lại không đủ để chứng minh mọi thuộc tính tương đương. Ta có thể minh họa điều đó trong đặc tả các số tự nhiên một cách đơn giản như sau :

Interface

Sort not ;

Operations

0 : → not ;

_ _ : not → not ;

_ + _ : not not → not ;

Body

Axions

a x 1 0 + x = x ;

a x 2 : x + (- x) = 0 ;

...

Từ đặc tả trên, ta có thể xây dựng các quy tắc :

0 + x ??? x

x + (- x) ??? 0

- 0 ??? 0

Ta nói lệnh **chg** (x, y) đã chuyển bộ nhớ q thành q'.

Định nghĩa ngữ nghĩa của lệnh **chg** chính là đặc tả, nhờ viết điều kiện tương đương với điều kiện (1.1)

Ví dụ sự kết hợp các điều kiện sau đây để đặc tả phép toán thay đổi **chg** :

$q'(x) = y$

$q'(a) = q(a)$ nếu $a \neq x$ (1.2)

Như vậy ta đã ngầm ẩn thừa nhận rằng q và q' chỉ định các hàm. Trong ví dụ này, với bộ nhớ 3 địa chỉ, miền xác định của hàm là tập hợp {1, 2, 3} tổng quát các địa chỉ và miền giá trị là các giá trị có thể lưu trữ được trong bộ nhớ đang xét, chẳng hạn là tập hợp các số nguyên $\{-2^{31}, ..., 2^{31}\}$ (1.3)

Một cách tổng quát, gọi A là tập hợp các địa chỉ, V là tập hợp các giá trị, ta có:

$q \in A \rightarrow V$ (1.4)

Như vậy một biểu thức dạng q (a) chỉ có nghĩa nếu $a \in A$, từ đó q (a) chỉ có nghĩa nếu $a \in A$, từ đó $q(a) \in V$. Do bộ nhớ q' nhận được từ q sau khi thực hiện lệnh **chg**, q' cũng thỏa mãn điều kiện (1.4)

$q' \in A \rightarrow V$

Tuy nhiên điều kiện (1.2) không phải luôn luôn có nghĩa vì rằng biểu thức q' (x) đòi hỏi $x \in A$. Ta thấy điều kiện (1.1) dẫn đến (1.2) nhưng ngược lại không hoàn tàon đúng.

Ta có thể hạn chế các điều kiện (1.2) để có được điều kiện tương đương như sau

$x \in A$

$y \in V$

$q'(x) = y$ (1.5)

$q'(a) = q(a)$ với $x \neq a$ và $x \in A$

Trong (1.5), hai điều kiện đầu được gọi là điều kiện đầu (preconditions), hai điều kiện sau được gọi là các điều kiện sau (postconditions)

Ta cần kiểm tra (1.5) là chấp nhận được, nghĩa là có điều kiện (1.4) là bất hiến (invariant). Muốn vậy ta cần chứng minh định lý sau đây :

((1.4) và (1.5)) kéo theo (1.4)' (1.6)

Định lý về tính chấp nhận được (plausibility)

Ta có thể nối hai điều kiện tương đương (1.1) và (1.5)

q **chg** (x, y) q'

$x \in A$

$y \in V$ (1.7)

Ta cũng có thể xây dựng tập hợp con của đích chứa các giá trị xác định từ tập hợp con của nguồn, gọi là ran (f) (range)

Khi một hàm được định nghĩa, ta có thể liệt kê các thành phần của hàm. Ví dụ, ta có :

1 0 0

2 0 chg (2, 5) 5

3 0 0

tương ứng với hai hàm :

q = { 1→ 0, 2 → 0, 3 → 0 }

q' = { 1 → 0, 2 → 5, 3 → 0 }

Để nhân được các dom và các ran từ các hàm, người ta tập hợp các phần tử đặt ở bên trái và bên phải của mũi tên tương ứng :

dom (q) = {1, 2, 3} dom (q') = 1, 2, 3}

ran (q) = { 0 } ran (q') = { 0, 5 }

Một cách tổng quan, ta có kết quả sau :

dom ({ x → y }) = { x }

ran ({ x → y }) = { y } (2.1)

f ∈ X → Y kéo theo dom (f) = X

Cho hàm f ∈ X → Y và một tập hợp con S ⊆ X, người ta ký hiệu f \ S là hàm nhân được bằng cách loại khỏi dom (f) các phần tử của S. Đây là phép hạn chế tương ứng với định nghĩa sau :

dom (f \ S) = dom (f) - S

(f \ S) (x) = f (x) với x ∈ dom (f) - S

HÌNH VẼ

Ta có : { 1 → 5, 2 → 8, 3 → 6 } \ { 1, 2} = { 3 → 6 } \ { 1, 2 } = { 3 → 6 }

Cho hai hàm có cùng nguồn và cùng đích nhưng có các dom rời nhau. Ký hiệu f ∪ g là hợp của hai hàm theo định nghĩa sau :

dom (f ∪ g) = dom (f) ∪ dom (g)

$$(f \cup g)(x) = \begin{cases} f(x) \text{ nếu } x \in \text{dom}(f) \\ g(x) \text{ nếu } x \in \text{dom}(g) \end{cases} \quad (2.4)$$

Bằng cách dùng hai phép toán trên đây, ta có thể định nghĩa sự chồng lên (overload) của một hàm bởi một hàm khác. Ta ký hiệu f + g tác động lên hai hàm f và g có cùng nguồn và cùng đích mà lần này, các dom không nhất thiết rời nhau :

Ta có : $f + g = (f \setminus dom(g)) \cup g$ (2.5)

Từ (2.3) ta có :

$$(f + g)(x) = \begin{cases} (f \setminus dom(g))(x) & \text{nếu } x \in dom(f \setminus dom(g)) \\ g(x) & \text{nếu } x \in dom(g) \end{cases} \quad (2.6)$$

hay

$$(f + g)(x) = \begin{cases} f(x) & \text{nếu } x \in dom(f) - dom(g) \\ g(x) & \text{nếu } x \in dom(g) \end{cases}$$

HÌNH VẼ

Ví dụ : $\{ 1 \to 0, 2 \to 0, 3 \to 0 \} + \{ 2 \to 5 \}$

$= (\{ 1 \to 0, 2 \to 0, 3 \to 0 \} \setminus \{ 2 \} \cup \{ 2 \to 0 \}$

$= \{ 1 \to 0, 3 \to 0 \} \cup \{ 2 \to 5 \}$

$= \{ 1 \to 0, 2 \to 5, 3 \to 0 \}$

Lý do cơ bản để đưa vào các phép toán \, ∪ và + thay vì sử dụng một cách hệ thống những định nghĩa của chúng (2.3), (2.4) và (2.7) trong việc hình thức hóa là ở chỗ tâ có thể chứng minh dễ dàng dãy các tính chất đại số sẽ sử dụng về sau.

Sau đây là một số tính chất :

$(f \cup g) \setminus S = (f \setminus G) \cup (g \setminus S)$

$(f + g) \setminus S = (f \setminus S) + (g \setminus S)$

$(f \setminus S) \setminus T = f \setminus (S \cup T)$

$f \cup g = g \cup f$

$(f \cup g) \cup h = f \cup (g \cup h)$

$dom(f \cup g) = dom(f) \cup dom(g)$ (2.8)

$ran(f \cup g) = ran(f) \cup ran(g)$

$(f + g) + h = f + (g + h)$

$dom(f + g = dom(f) \cup dom(g)$

Bây giờ ta có thể sửa chữa đặc tả đã nhận được ở cuốn mục trước (1.7). Đặc tả này rõ ràng đơn giản hơn :

$q\ chg\ (x, y)\ q'$

$x \in A$

$y \in V$

$q' = q + \{ x \to y \}$

Định lý về tính chấp nhận được (plausilility) bây giờ dễ dàng được chứng minh bởi phép tính hình thức đơn giản (simple formal calculus).

$q \in A \rightarrow V$ (3.1)

Bộ nhớ q đóngvai trò bộ nhớ trước đó, còn bộ nhớ p dùng để khôi phục trạng thái cũ khi cần khởi động lại. Sau đây là đặc tả của 3 thao tác cho hệ thống với mod thay thế chg :

(p, q) mod (x, y) ((p', q')

p' = p (3.2)

q chg (x, y) q'

Ta thấy thao tác thay đổi bộ nhớ xuất hiện như một sự mở rộng, ở mức đặc tả, của phép toán chg :

(p, q) rst (p', q')

q' = p

q' = q (3.3)

(p, q) vld (p', q')

p' = q

q' = q (3.4)

Dễ dàng chứng minh rằng cả ba phép toán này đều chấp nhận được, nghĩa là sau khi thực hiện chúng, ta có (3.1)'. Nói cách khác, p và q được thay thế bởi p' và q' trong (3.1), với giả thiết rằng (3.1) đã được kiểm chứng trước khi thực hiện chúng.

Rốt cuộc, ta phải chứng minh rằng việc kết hợp các điều kiện sau đây :

(p, q) vld (p_1, q_1)

(p_1, q_1) op (p_2, q_2)

...

(p_{n-1}, q_{n-1}) op (p_n, q_n)

(p_n, q_n) rst (p', q')

dẫn đến q' = q

Thật vậy, theo (3.4), ta có :

p_1 = q

và theo (3.2) và (3.4) do op là một trong hai phép toán mod (x, y) hoặc rst, ta có :

p_n = ... p_1

Cuối cùng, theo (3.3) :q' = p_n

Rõ ràng phép hợp thức hóa là trong suốt vì dẫn đến q' = q theo (3.4)

Sau đây là một ví dụ về các phép toán trên :

```
     q      p
1    0      0
2    0      0
3    0      0
mod (1, 1)
1    1      0
2    0      0
3    0      0
mod (2, 2)
1    1      0
2    2      0
3    0      0
mod (1, 3)
1    3      0
2    2      0
3    0      0
vld
1    3      3
2    2      2
3    0      0
mod (3, 1)        q      q
1    3      3
2    2      2
2    1      0
not
1    3      3
2    2      2
3    0      0
```

Phần tiếp theo sẽ làm mịn mô hình này, nghĩa là đưa vào các biến trạng thái mới để thể hiện các ràng buộc về phần cứng và phần mềm.

p (x) = m (a (x))

q (x) = m (n (x)) với x ∈ A (4.2)

Ta có thể kiểm chứng ngay được rằng những thay đổi của biến là có ý nghĩa (rõ ràng vì các hàm a, n và m là toàn phần) và chặt chẽ với bất biến (3.1) chỉ rõ rằng p và q dều thuộc tập hợp A → V

Phép biến đổi mod dặc tả bởi các điều kiện (3.2) sẽ được triển khai bởi một phép toán mới mod_1. Phép mod_1 sẽ ghi một giá trị mới, một địa chỉmà không thuộc vào miền trị (range) của n cũng như miền trị của a, nói cách khác, địa chỉ này chỉ phụ thuộc vào tập hợp :

ran (n) ∪ ran (a)

hay rõ hơn, thuộc tập hợp D - (ran (n) ∪ ran (0)), tập hợp được đặt tên là L (Liberty).

Nếu L ≠ ∅, một điều kiện trước dược đặt ra, thì ta có thể chọn một địa chỉ bất kỳ u, với đặc tả sau :

(a, n, m) mod_1 (x, y) (a', n', m')

x ∈ A

y ∈ V

L ≠ ∅

a' = a

n' = n + { x → u }

m' = m + { u → y } (4.3)

trong đó

L = ran (n) ∪ ran (a)

u ∈ L

Ta cần chứng minh rằng đặc tả (4.3) là chấp nhận được, nghĩa là :

((4.1) và (4.3)) kéo theo (4.1)' (4.4)

Mệnh đề (4.4) trên đây là hiển nhiên. Tiếp theo ta cần chứng minh phép mod_1 phù hợp (đúng) với phép mod đã đặc tả ở (3.2) như sau :

((4.2), (4.2)' và (4.3)) kéo theo (3.2) (4.5)

Mệnh đề này không hiển nhiên, tương ứng với sơ đồ giao hoán dưới đây :

HÌNH VẼ

Nói cách khác, nếu các giá trị của các biến (a, n, m) và (a', n', m') thỏa mãn đặc tả (4.3), thì khi trở về các biến cũ (p, q) và (p', q') (các biến đã bị thay đổi trở thành các biến (a, n, m) và (a', n', m') bởi các phép biến đổi (4.2) và 4.2)') các giá trị của các biến (p, q) và (p', q') thỏa mãn đặc tả (3.2).

Nói gọn lại, (4.3) hiện thực (3.2)

Các phép toán mới khởi tạo lại các hợp thức hóa mô tả trong các đặc tả sau đây rõ ràng chấp nhận được và phù hợp :

(a, n, m) rst_1 (a', n', m')

a' = a

n' = n (4.6)

m' = m

(a, n, m) vld_1 (a', n', m')

a' = n

n' = n (4.7)

m' = m

Nhìn vào các công thức, ta thấy đã không chép lại các giá trị nhưng chỉ có các địa chỉ có thể có theo một sự tiết kiệm đáng kể về thời gian nhưng không lớn lắm về không gian nhớ. Giả thiết với 4096 trang và địa chỉ của D là 2 bytes, mỗi bảng a và n sẽ chiến 8 Kbytes.

Hệ thống mới hoạt động qua ví dụ sau :

	n	a	m		
1	1	1	1	0	
2	2	2	2	0	
3	3	3	3	0	
	4	0			
	5	0			
	6	0			

mod_1 (1, 1)

1	4	1	1	0	L = { 4, 5, 6 }
2	2	2	2	0	u = 4 ∈ L
3	3	3	3	0	(chọn u = min (L))
	4	1			
	5	0			
	6	0			

Tùy theo một địa chỉ d của D thuộc về một trong bốn tập hợp trên, ta nói trạng thái tương ứng sẽ là :

old (cũ) $d \in ran(a)$

common (chung) $d \in ran(a)$ và $d \in ran(n)$

new (mới) $d \in ran(a)$

free (tự do) $d \notin ran(a)$ và $d \notin ran(n)$

Khi một địa chỉ tự do được chọn, khi một thay đổi xảy ra, địa chỉ đó chuyển qua trạng thái mới ; về địa chỉ quá tải trong bảng n, nếu địa chỉ đó không phân chia bên trong bảng n (nghĩa là nếu hàm nlà đơn ánh và điều này được giả thiết là luôn đúng), khi đó, địa chỉ sẽ trở nên tự do nếu đạng ở trạng thái mới hoặc chuyển sang trạng thái cũ nếu đang ở trạng thái chung.

Khi một phép hợp thức hóa hay khởi động lại các địa chỉ tự do hay chung vẫn như cũ. Các địa chỉ mới chuyển thành tự do khi một sự khởi động lại và là trường hợp chung khi hợp thức hóa.

Cuối cùng, các địa chỉ cũ chuyển thành tự do khi hợp thức hóa và trở thành chung khi khởi động lại.Chú ý rằng các địa chỉ cũ không liên quan đến sự thay đổi. Sơ đồ dưới đây tóm tắt một cách phi hình thức những chuyển đổi khác nhau này.

Mục đích để hình thức hóa phương pháp này, ta đưa vào một biến mới s định nghĩa trạng thái của mỗi địa chỉ của D.

$s \in D \rightarrow \{fr, nw, cm, ol\}$ (6.1)

Ta có bất biến sau đây :

$RA - RN = adr(ol)$
$RA \cap RN = adr(cm)$ (6.2)
$RN - RA = adr(nw)$
$RA \cup RN = adr(fr)$

HÌNH VẼ

Trong đó :

$RA = ran(a)$
$RN = ran(n)$
$adr(z) = \{x \in D / s(x) = Z\}$

với $Z \in \{fr, nw, cm, ol\}$

Cuối cùng bất biến thứ ba chỉ rõ rằng cả hai hàm n và a đều đơn ánh, nghĩa là hai địa chỉ của A phân biệt sẽ luôn luôn tương ứng với các địa chỉ của D phân biệt qua các hàm này.

Tập hợp các hàm từ A vào D như vậy được ký hiệu

bởi A ⤆ D như vậy

$n \in A \setminus D$

$a \in A \setminus D$

Bây giờ sẽ là định nghĩa ba hàm chuyển tiếp lần lượt là f, g và h sử dụng khi thay đổi (cho các địa chỉ của D liên quan), khởi động lại thay cho hợp thức hóa (cho mọi địa chỉ của D) :

f = {fr → nw,

nw → fr, thay đổi

cm → ol}

g = {fr → fr,

nw → fr,

cm → cm,khởi động lại

ol → cm}

h = {fr → fr,

nw → cm,

cm → cm,hợp thức hóa

ol → fr}

Ta có đặc tả của 3 phép toán mới mod_2, rst_2, và vld_2 xuất hiện như là các mở rộng tương ứng từ mục 4 :

(a, n, m, s) mod_2 (x, y) (a', n', m', s')

(a, n, m) mod_1 (x, y) (a', n', m') (6.5)

s' = s + {u → f (s (u)), v → f (s) (v))}

xem (4.3)

trong đó :

L = { Z ∈ D | s (z) = fr }

u ∈ L

v = n (x)

(a, n, m, s) rst_2 (a', n', m', s') (6.6)

s' = gos xem (4.6)

(a, n, m, s) vld_2 (a', n', m', s') (6.7)

(a, n, m) vld_1 (a', n', m') xem (4.7)

s' = hos

Sau đây là một quá trình chuyển đổi của hệ thống

CHỪA

Chứng minh

Mặc dù trong mục trước ta đã kiểm chứng kỹ lưỡng đặc tả hệ thống và nhận được kết quả thõa mãn, nhưng chưa đảm bảo được tính đúng đắn của đặc tả trong mọi trường hợp.

Để đi đến một kết quả tổng quát, ta cần phải chứng minh không phải cho một trường hợp đặc biệt nào đó mà phải cho các dữ liệu tượng trưng thỏa mãn những giả thiết nào đó, các phép toán đã đặc tả là phù hợp và chấp nhận được.

Việc chứng minh tính phù hợp của các phép toán đặc tả ở (6.5), (6.6) và (6.7) so với các phép toán đặc tả ở (4.3), (4.6) và (4.7) là hiển nhiên vì rằng trong cách lập các công thức thì các phép toán (4.3), (4.6) và (4.7) một cách tương ứng.

Trái lại, việc chứng minh tính chấp nhận được phức tạp hơn. Trước hết ta cần chứng minh ba nhóm định lý bất biến sau đây :

((6.1) và (6.5)) kéo theo (6.1)' (7.1)

((6.2) và (6.5)) kéo theo (6.2)' (7.2)

((6.3) và (6.5)) kéo theo (6.3)' (7.3)

((6.1) và (6.6)) kéo theo (6.1)' (7.4)

((6.2) và (6.6)) kéo theo (6.2)' (7.5)

((6.3) và (6.6)) kéo theo (6.3)' (7.6)

((6.1) và (6.7)) kéo theo (6.1)' (7.7)

((6.2) và (6.7)) kéo theo (6.2)' (7.8)

((6.3) và (6.7)) kéo theo (6.3)' (7.9)

Đối với 3 định lý ở nhóm 1, ta có thể dẫn đến các giả thiết cho các điều kiện sau đây :

$a \in A \cup D$

$n \in A \cup D$

$u \notin RN$ (7.10)

$u \notin RA$

$v \in RN$

Trong đó u và được định nghĩa ở (6.5) và RA, RN được định nghĩa ở (6.2)

chứng minh (7.1)

Một khó khăn nhỏ là hàm chuyển tiếp f được định nghĩa ở (6.4) là hàm bộ phận. Cần chứng minh rằng s' được định nghĩa đúng, nghĩa là các biểu thức f (s (u)) và f (s (v)) trong (6.5) có nghĩa, nói cách khác ta có :

s (u) ∈ dom (f)

s (v) ∈ dom (f)

điều này hiển nhiên vì rằng theo (7.10), ta có :

s (u) = fr

s (v) = { nw, cm}

và theo (6.4) ta có

dom (f) = {fr, nw, cm}

Để chứng minh (7.2) và (7.3) ta cần kết quả sau đây liên quan đến sự quá tải của một hàm đơn ánh thừa nhận mà không chứng minh :

f ∈ X ⇀ Y kéo theo f' ∈ X ⇀ Y

x ∈ dom (f) ran (f') = r'

y ∉ Y - ran (f) y ≠ f (x)

Trong đó :

f' = f + {x → y}

r' = (ran (f) - {f (x)}) ∪ {y}

chứng minh (7.2) : Theo (7.10), (7.11) và (6.5) ta có

RA' = RA (vì rằng a' = a theo (4.3))

RN' = (RN {V}) ∪ {u} theo 7.11

u ≠ v theo 7.10

HÌNH VẼ

Xảy ra hai trường hợp :

1/ v ∈ RA, nghĩa là s (v) = cm

Khi đó :

RA' - RN' = (RA - RN) ∪ {v}
RA' ∩ RN' = (RA ∩ RN) - {v}
RN' - RA' = (RN - RA) ∪ {u}
RA' ∪ RN' = (RA ∪ RN) - {u}

HÌNH VẼ

2/ v ∉ RA, nghĩa là s (v) = nw

Khi đó :

RA' - RN' = RA - RN

t' = t

(a, n, m, s, t) vld_3 (a', n', m', s', t') (8.5)

(a, n, m, s) vld_2 (a', n', m', s') xem (6.7)

t' = s'

Dễ dàng chứng minh rằng cả ba đặc tả trên là phù hợp và chấp nhận được. Mặt khác ta thấy rằng phép khởi động lại tái sinh bộ nhớ trong từ đĩa và chỉ từ đĩa mà thôi.

Triển khai lần thứ tư và lần thứ năm

Ta sẽ mã hóa các giá trị fr, nw, cm và ol như hàm đơn ánh k như sau :

k = {(0, 0) → fr,
(0, 1) → ol,
(1, 0) → cm, (9.1)
(1, 1) → nw}

Sau đó ta biểu diễn các hàm s và t nhờ ba chuỗi bit như sau :

b ∈ D → {0, 1} để biểu diễn s

c ∈ D → {0, 1}

d → {0, 1}để biểu diễn t (9.2)

Cuối cùng, thay đổi các biến tương ứng với các điều kiện sau :

s (x) = k (b (x), c (x))
t (x) = k (d (x), 0) với x ∈ D (9.3)

Giả sử là phép bù (đảo ngược bít) như sau

0 = 1,1 = 0 (9.4)

Ta thấy có thể mã hóa hàm f đã định nghĩa ở (6.4) nhờ hai phép bù và hàm h cũng đã dịnh nghĩa ở (6.4) nhờ phép sao chép và đặt lề 0 tương ứng với hàm :

Z ∈ D → {0} (9.5)

Ta có 3 đặc tả mới như sau :

(a, n, m, b, c, d) mod_4 (x, y) (a', n', m', b', c', d')
(a, n, m) vld_1 (a', n',m')
b' = b (9.8)
c' = z xem (4.7)
d' = b

Bây giờ ta chỉ cần tóm tắt lại những gì đã làm cho đến lúc này, nghĩa là một mặt, sao chép lại các đặc tả (4.3), (4.6) và (4.7) vào bên trong của (9.6), (9.7) và (9.8), mặt khác, nhóm các bất biến (4.1), (6.1), (6.2), (6.3), (8.1), (8.2) và (9.2)

Điều này làm được bằng cách khử các biến trở thành dư thừa (chứa s và t) bởi các phép thay đổi biến (9.3).

Khi sao chép, ta thấy rằng đặc tả (4.3) chứa điều kiện trước $L \neq \varnothing$ không dễ gì tính được. Để khắc phục nhược điểm này ta đưa vào một biến mới w là một số nguyên

$w \in N$ (9.9)

w chứa các phần tử của tập hợp L (cardinality)

$w = | RA \cup RN |$

Ta thừa nhận ngầm rằng các tập hợp D và A đều là hữu hạn. Khi hợp thức hóa và khởi động lại, bộ đếm w được khởi tạo giá trị $|D| - |A|$ (vì rằng a và n đều đơn ánh) là một hằng số dương của hệ thống. Khi có sự thay đổi, bộ đếm w tăng lên khi và chỉ khi địa chỉ v, quá tải trong n, đang ở trạng thái cm, nghĩa là nếu b (v) = 1 và nếu c (v) = 0

Với sự mở rộng mới này, bất biến của hệ thống lúc này sẽ là :

$a \in A \rightarrow D$ (6.3)
$n \in A \rightarrow D$ (6.3)
$m \in D \rightarrow V$ (4.1)
$b \in D \rightarrow \{0, 1\}$ (9.2)
$c \in D \rightarrow \{0, 1\}$ (9.2)
$d \in D \rightarrow \{0, 1\}$ (9.2)
$d \in D \rightarrow \{0, 1\}$ (9.2)
$w \in N$ (9.9)
$RA - RN = \{x \in D / b (x) = 0 \text{ và } c (x) = 1\}$ (6.2)
$RA \cap RN = \{x \in D / b (x) = 1 \text{ và } c (x) = 0\}$ (6.2)
$RN - RA = \{x \in D / b (x) = 1 \text{ và } c (x) = 1\}$ (6.2)
$RA \cup RA \{x \in D / b (x) = 0 \text{ và } c (x) = 0\}$ (6.2)
$RA = \{x \in D / d (x) = 1\}$(9.2)
$W = | RA \cup RN |$ (9.10)

Trong đó

RA = ran (a)
RN = ran (n)

Sau đây là các đặc tả được tóm tắt bằng cách thay thế các danh sách dài các biến bởi hai biến trạng thái state và trạng thái có dấu nháy (') state'

$state\ mod_5\ (x, y)\ state'$
$x \in A$
$y \in V$
$w \neq 0$
$a' = a$
$n' = n + \{x \rightarrow u\}$

ngữ giả Pascal. Ta đưa vào các quy tắc để tiết lập mối liên hệ giữa đặc tả và lập trình.

Quy tắc 1 :

Khi một đặc tả chứa các điều kiện trước, chương trình tương ứng sẽ là một hàm. Theo nghĩa của Pascal, mỗi giá trị sai của điều kiện trước sẽ trả về biến trạng thái state một giá trị phân biệt.

Chương trình tương ứng với đặc tả (9.12) là như sau :

```
if not (x in A) then
state := bad - x
else if not (y in v) then
state := bad - y
else if w = 0 then
state := no more place
else begin
State := OK ;
Modification {gọi thủ tục}
end ;
```

Quy tắc 2 :

Khi một đặc tả chứa các biến phụ, ta có thể mở mọi thủ tục chứa các biến này như là các biến cục bộ. Thủ tục này bắt đầu bởi các lệnh khởi động

Thủ tục Modification như sau :

```
procedure Modification ;
var u, v : D
begin
    u := 1 ; { chọn u là địa chỉ bé nhất của L}
    while (b (u) ≠ 0) or (c (u) ≠ 0) do
    u := u + 1 ;  (10.2)
    v := n (x)
    { tiếp tục thân thủ tục
end ;
```

Quy tắc 3 :

Các điều kiện sau khác nhau nếu có dạng a' = ... (trong đó dấu ba chấm ... chỉ định một biểu thức khởi động chứa các biến có đánh dấu nháy), thì có thể được chuyển thành phép gán qua các quy tắc bổ trợ như sau :

- Loại bỏ các điều kiện sau dạng đẳng thức.

Ví dụ : d' = d

- Thay thế dấu = bởi dấu gán bằng :=

- Thực hiện phép tối ưu khi một hàm là quá tải

- Loại bỏ các dấu nháy '

- Thay thế một điều kiện sau bởi cấu trúc điều kiện if... else :

Các điều kiện sau không bị loại bỏ của đặc tả (9.12) như sau :

```
if not (x in A) then
      state := bad - x
   else if not (y in V) then
      state := bad - y(10.1)
   else if w = 0 then
      state := no-more-place
   else begin
      state := O.K ;
      Modication { gọi giá trị thủ tục}
      end ;
```

Quy tắc 2 :

Khi một đặc tả chứa các biến phụ, ta có thể mở một thủ tục chứa các biến này như là các biến cục bộ. Thủ tục này bắt đầu bởi các lệnh khởi động.

Thủ tục Modication như sau :

```
Procedure Modication ;
var u, v : D
begin
   u := 1 ; { chọn u là địa chỉ bé nhấtcủa L}
   uhile (b (u) ≠ 0) or (c (u) ≠ 0) do
   u := u + 1 ;  (10.2)
   v := n (x)
   { tiếp tục thân thủ tục
end ;
```

Quy tắc 3 :

Các điều kiện sau khác nhau nếu đều có dạng a' = ... (trong đó dấu chấm ... chỉ dịnh một biểu thức không chứa các biến có đánh dấu nháy), thì có thể được chuyển thành phép gán qua các quy tắc bổ trợ như sau :

- Loại bỏ các điều kiện sau dạng đẳng thức.

Ví dụ : d' = d

- Thay thế dấu = bởi dấu gán bằng :=

- thực hiện phép tối ưu khi một hàm là quá tải.

- Loại bỏ các dấu nháy '

- Thay thế một điềukiện sau bởi cấu trúc điều kiện if ... else :

Các điều kiện sau không loại bỏ của đặc tả (9.12) như sau :

n (x) := u ;

m (u) := y ; (10.3)

b (u) := b (u) ; b (v) := b (v) ;

c (u) := c (u) ; c (v) = c (v) ;

if (b (v) = 1) and (c (v) = 0) then w := w - 1

Quy tắc 4 :

Một cách hệ thống các hệ chương trình đã viết được bởi các quy tắc đảm bảo tính "song song" đưa vào từ đặc tả. Từ các đoạn chương trình (10.1), (10.2) và (10.3) ta nhận được công thức đấy đủ hơn như sau :

Một số đề thi

I. Đặc tả (Specification)

1. Cho ma trận vuông A cấp $n \times n$. Viết đặc tả thể hiện : ***a)*** Mỗi phần tử trên đường chéo chính là phần tử lớn nhất trên cùng hàng đi qua phần tử đó. ***b)*** Mỗi phần tử trên đường chéo phụ là phần tử nhỏ nhất trên cùng cột đi qua phần tử đó.
2. Một xâu (string) w được gọi là *đối xứng* (palindrome) nếu $w = w^R$ hay đọc xuôi ngược đọc ngược đều như nhau (w^R là xâu đảo ngược của w). Ví dụ các xâu *omo*, *mannam*, ... đều là *đối xứng*. Viết đặc tả thể hiện các xâu *đối xứng*.
3. Đa thức cấp n được viết dưới dạng Toán học là :

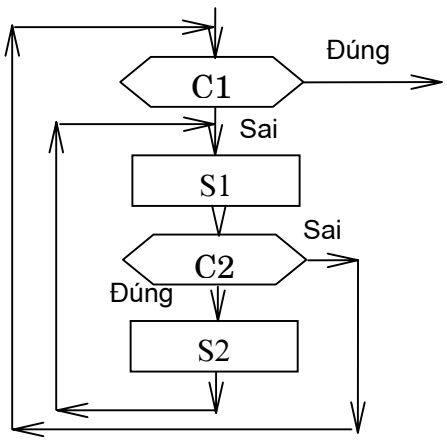
$$P_n(x) = a_0 + a_1x^1 + a_2x^2 + ... + a_nx^n$$

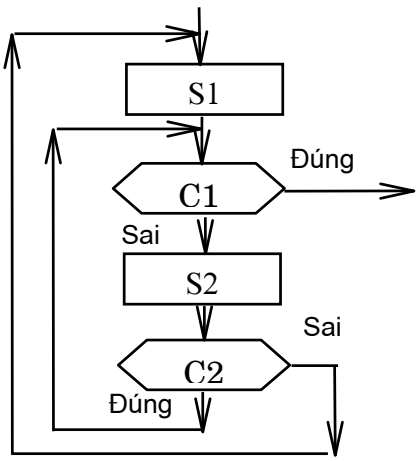
Viết đặc tả thể hiện phép cộng, so sánh hai đa thức $P_n(x)$ và $Q_m(x)$, nhân đa thức với một hằng số $a \times P_n(x)$ và nhân hai đa thức $P_n(x) \times Q_m(x)$.

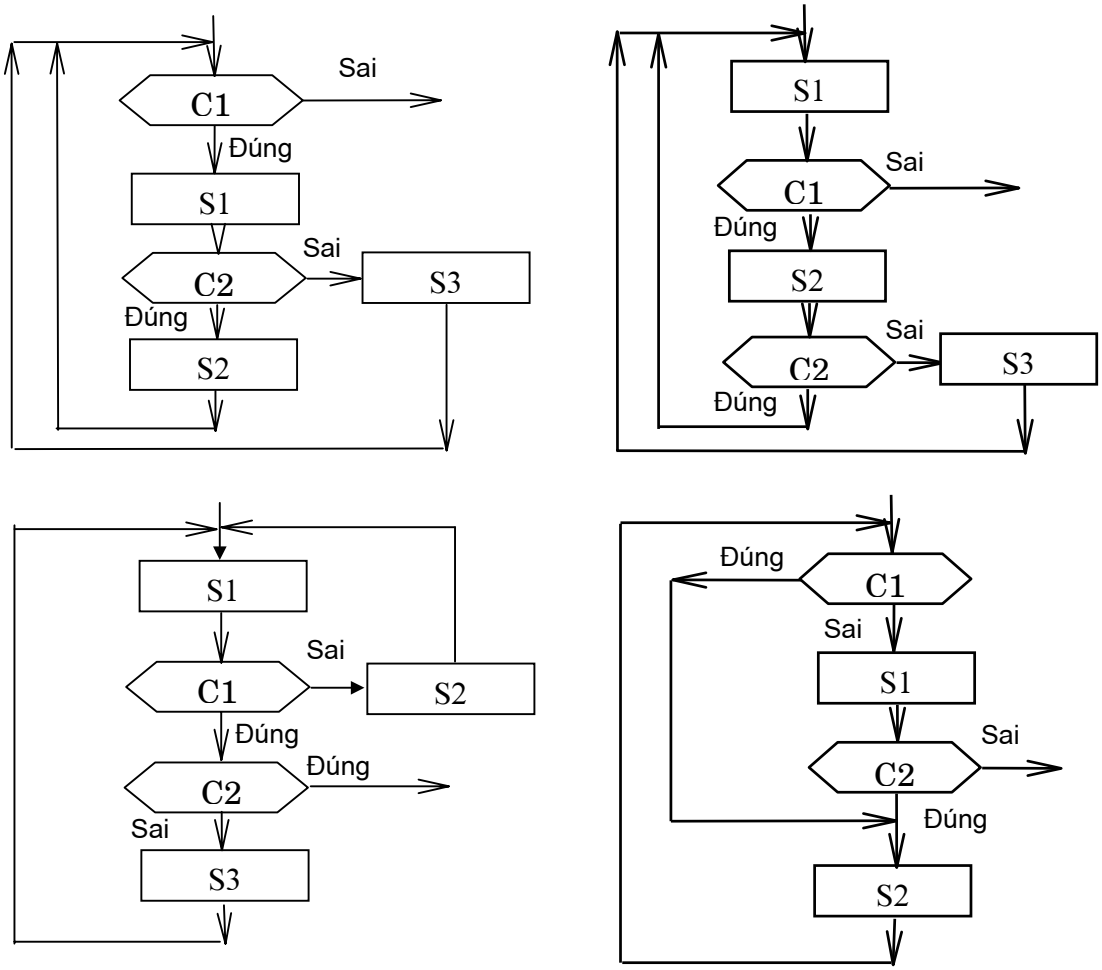
4. Các phân số (hay số hữu tỷ) được biểu diễn bởi danh sách (n, d), với n là tử số và d là mẫu số, là những số nguyên ($d \neq 0$). Viết đặc tả xây dựng các hàm xử lý phân số: rút gọn, trừ, chia và so sánh hai phân số, cộng, nhân hai phân số và chuyển đổi phân số thành số thực.

II. Lập trình cấu trúc (Structured programming)

1. Viết lệnh bằng giả ngữ (phỏng Pascal), chỉ sử dụng tối đa ba cấu trúc tuần tự, điều kiện if và lặp (while-repeat), theo các sơ đồ khối dưới đây :







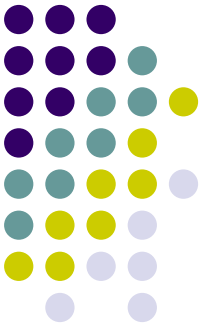
2. Viết lệnh bằng giả ngữ (phỏng Pascal), chỉ sử dụng tối đa ba cấu trúc tuần tự, điều kiện if và lặp (while□repeat), theo sơ đồ khối dưới đây :

III. Thử nghiệm chương trình (Testing)

Giả sử các chương trình đã cho ở phần trên đây là các đơn thể gọi đến các đơn thể con S1, S2 và S3). Trình bày một phương pháp để thử nghiệm đơn thể gọi.

NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

GIỚI THIỆU



Phan Phương Lan

Nội dung

- Phần I: Tổng quan về Công nghệ phần mềm
  - Chương 1: Giới thiệu về Công nghệ phần mềm
  - Chương 2: Các mô hình về tiến trình phần mềm
  - Chương 3: Quản lý phần mềm
    - Quản lí nhân sự và tổ chức
    - Quản lí chất lượng
    - Quản lí cấu hình
    - Quản lí dự án
  - Chương 4: Ước lượng giá thành
- Phần II: Tiến trình phần mềm
  - Chương 5: Đặc tả yêu cầu
  - Chương 6: Thiết kế
  - Chương 7: Lập trình
  - Chương 8: Kiểm thử
  - Chương 9: Triển khai hệ thống
  - Chương 10: Bảo trì

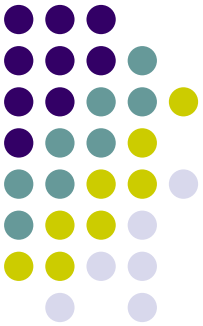
Tài liệu tham khảo

- *Sách tham khảo chính:*
  - Shari Lawrence Pleeger, Joanne M.Atlee, Software Engineering theory and practice, 3th edition, 2006.
  - Ian Sommerville, Software Engineering, 8th edition, 2006.

- *Sách đọc thêm:*
  - Hans Van Vliet, Software Engineering principles and practice, John Wiley, 2000.
  - Pressman, Roger S., Software Engineering: A Practitioner's Approach, McGraw-Hill, 5th edition, 2003.

NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

CHƯƠNG 1 – GIỚI THIỆU VỀ CÔNG NGHỆ PHẦN MỀM



Nội dung

- Định nghĩa về CNPM
- Các giai đoạn trong phát triển phần mềm
- Những người tham gia trong dự án phát triển phần mềm
- Các yếu tố chính làm thay đổi sự phát triển phần mềm

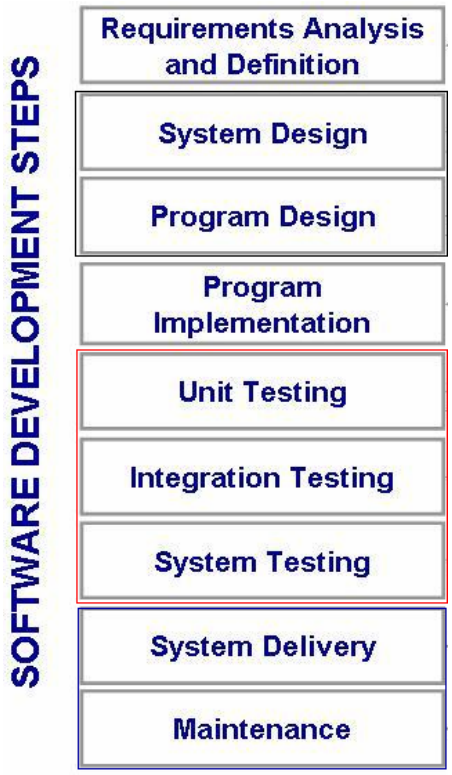
Định nghĩa về CNPM

- IEEE: CNPM là
  - (1) Việc áp dụng phương pháp tiếp cận có hệ thống, bài bản và được lượng hóa trong phát triển, vận hành và bảo trì phần mềm;
  - (2) Nghiên cứu các phương pháp tiếp cận được dùng trong (1)
- NATO: CNPM là việc thiết lập và dùng các nguyên tắc công nghệ đúng đắn để thu được phần mềm một cách kinh tế nhất và chạy hiệu quả trên các máy thật.

Định nghĩa về CNPM

- Mục tiêu của CNPM là làm sao để tạo ra phần mềm:
  - Có chất lượng cao
    - Đúng, thỏa yêu cầu khách hàng
    - Dễ khai thác, vận hành
    - Dễ bảo trì
  - Đúng kế hoạch thời gian
  - Trong phạm vi ngân sách dự kiến
  - Giá thành ngày càng hạ

Các giai đoạn phát triển phần mềm



Định nghĩa & Phân tích yêu cầu

Thiết kế

Cài đặt

Kiểm thử

Phát hành

Bảo trì

Các giai đoạn phát triển phần mềm

- **Định nghĩa & Phân tích yêu cầu**: thu thập mô tả đầy đủ của bài toán
  - Chức năng/tính năng của PM
  - Khả năng mở rộng
  - Các loại tài liệu đòi hỏi
  - Thời gian đáp ứng hoặc các yêu cầu về chất lượng của hệ thống
  - Nghiên cứu khả thi
- **Thiết kế**: thiết kế hệ thống và thiết kế chi tiết

Các giai đoạn phát triển phần mềm

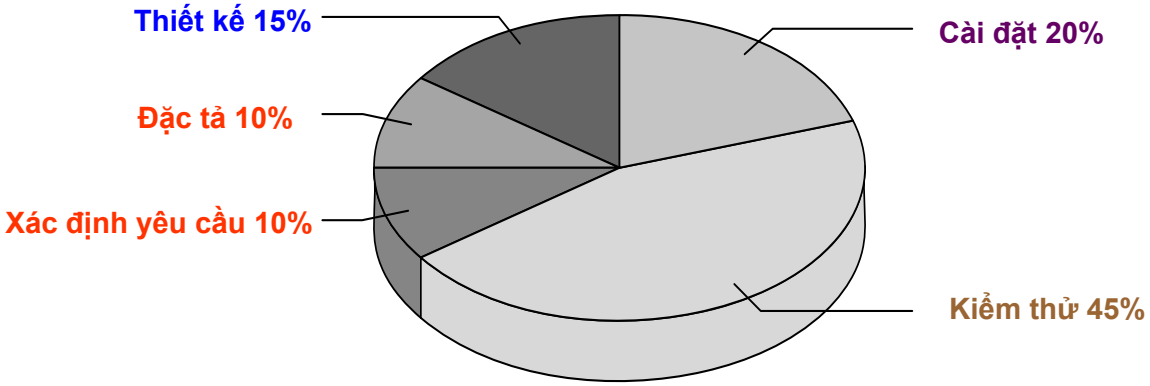
- **Cài đặt**: tập trung vào từng module riêng lẻ:
  - Giải thuật
  - Tài liệu
  - Coding
- **Kiểm thử** (kiểm thử đơn vị, kiểm thử tích hợp và kiểm thử hệ thống): thử và xác nhận tính đúng đắn của
  - Tài liệu đặc tả
  - Thiết kế
  - Module
  - Chuyển tiếp giữa các giai đoạn

Các giai đoạn phát triển phần mềm

- **Bảo trì**
  - Sửa lỗi sau khi phần mềm đã được triển khai
  - Đáp ứng sự thay đổi yêu cầu, sự thay đổi về môi trường, v.v

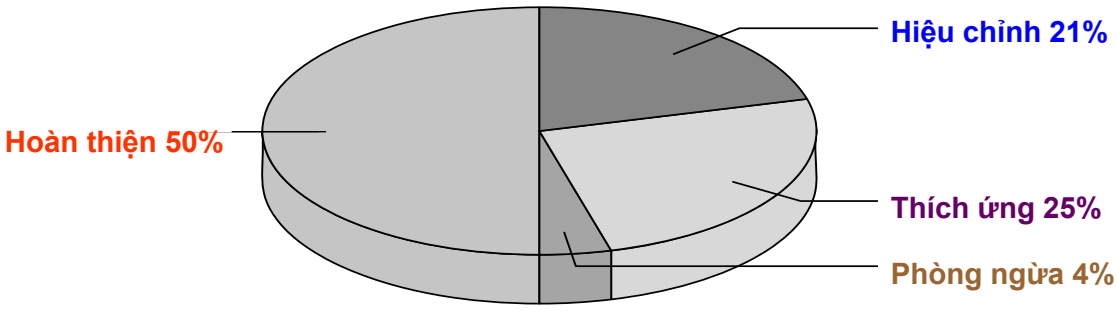
Các giai đoạn phát triển phần mềm

- Công sức của từng giai đoạn: 40 – 20 – 40



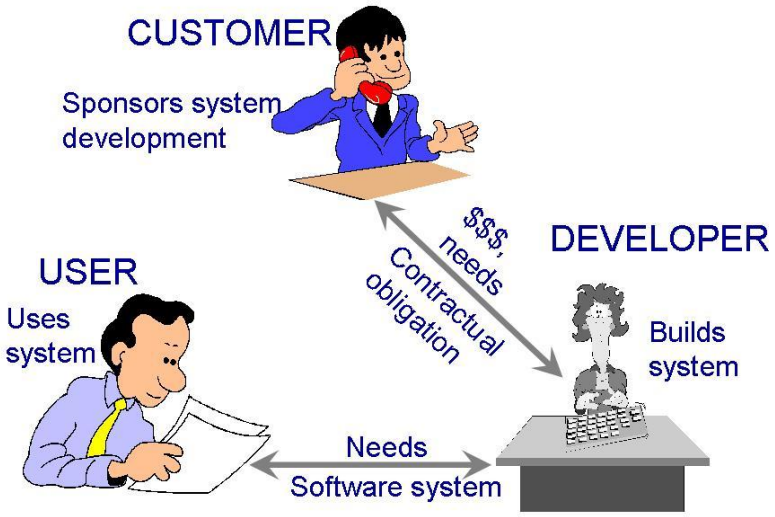
Các giai đoạn phát triển phần mềm

- Công sức của từng giai đoạn – Giai đoạn bảo trì
  - Hoạt động bảo trì chiếm khoảng 50 – 70% toàn bộ công sức
  - Các loại bảo trì: Hoàn thiện, Phòng ngừa, Hiệu chỉnh và Thích ứng
  - Sự phân phối của các loại bảo trì



Những người tham gia trong dự án phát triển phần mềm

- Những người tham gia: Khách hàng, Nhà phát triển và Người sử dụng.

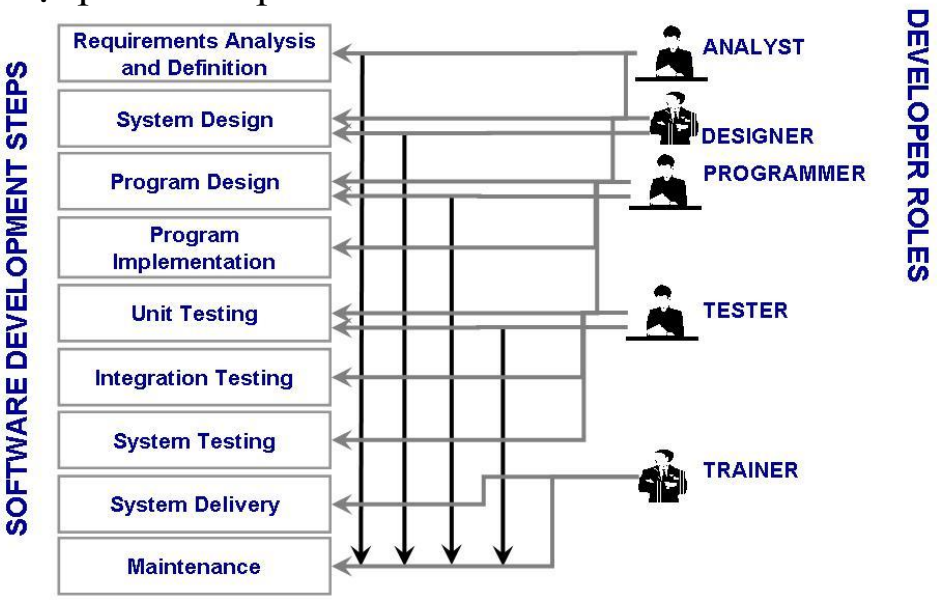


Những người tham gia trong dự án phát triển phần mềm

- Các thành viên trong đội phát triển phần mềm:
  - **Nhà phân tích yêu cầu**: làm việc với khách hàng để xác định và tư liệu hóa các yêu câu
  - **Nhà thiết kế**: tạo ra bản mô tả mức hệ thống về cái mà hệ thống phải thực hiện
  - **Lập trình viên**: viết mã lệnh cài đặt sự thiết kế
  - **Nhà kiểm thử**: bắt các lỗi
  - **Người hướng dẫn**: chỉ dẫn người dùng cách sử dụng hệ thống
  - **Bảo trì viên**: chỉnh sửa các lỗi khi hệ thống đã được phát hành và đáp ứng các thay đổi
  - **Thủ thư**: chuẩn bị và lưu giữ các tài liệu chẳng hạn như các đặc tả yêu cầu
  - **Nhóm quản lý cấu hình**: duy trì sự phù hợp giữa các thành phần được tạo ra

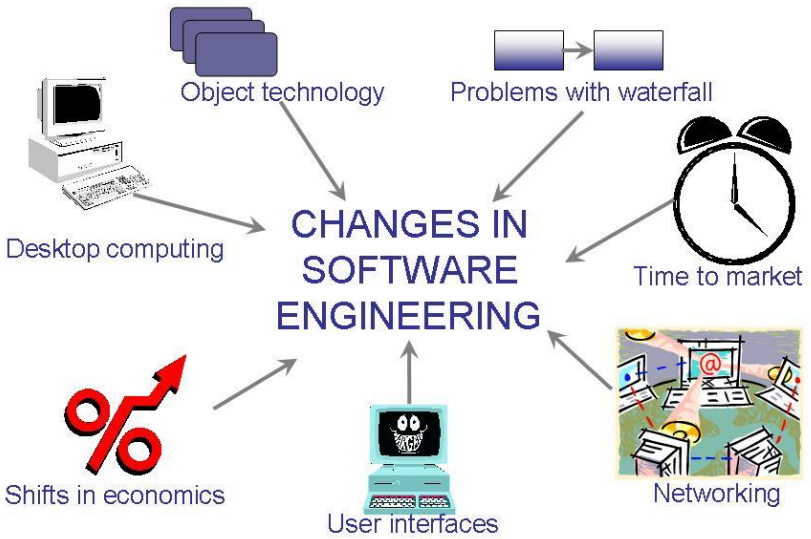
Những người tham gia trong dự án phát triển phần mềm

- Các vai trò tiêu biểu được thực hiện bởi những thành viên trong đội phát triển phần mềm



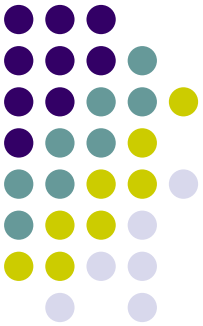
Các yếu tố chính làm thay đổi sự phát triển phần mềm

- Các yếu tố chính:



NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

CHƯƠNG 2 – CÁC MÔ HÌNH VỀ TIẾN TRÌNH PHẦN MỀM



Nội dung

- Tiến trình
- Các mô hình về tiến trình phần mềm
  - Mô hình thác nước
  - Mô hình chữ V
  - Mô hình bản mẫu
  - Mô hình phát triển ứng dụng nhanh
  - Mô hình gia tăng
  - Mô hình xoắn ốc
  - Mô hình RUP

Tiến trình (Process)

- **Tiến trình**: một chuỗi các bước bao gồm các hoạt động, các ràng buộc và các tài nguyên mà chúng tạo ra kết quả được mong đợi
- **Tiến trình**: bao gồm một bộ các công cụ và các kỹ thuật

Tiến trình

- **Các đặc trưng của tiến trình**
  - Quy định tất cả các hoạt động chính của tiến trình
  - Sử dụng các nguồn tài nguyên, phụ thuộc vào tập các ràng buộc (chẳng hạn như kế hoạch làm việc)
  - Tạo ra các sản phẩm cuối cùng hoặc trung gian
  - Có thể được tạo thành từ các tiến trình con bằng hệ thống phân cấp hay các liên kết

Tiến trình

- **Các đặc trưng của tiến trình**
  - Mỗi hoạt động của tiến trình có tiêu chuẩn vào và ra
  - Các hoạt động được tổ chức theo trình tự vì thế sự tính toán về thời gian là rõ ràng
  - Mỗi tiến trình có các nguyên tắc hướng dẫn, bao gồm các mục tiêu của từng hoạt động
  - Các ràng buộc có thể áp dụng vào một hoạt động, tài nguyên hay sản phẩm

Tiến trình

- **Tầm quan trọng của tiến trình**
  - Áp đặt cấu trúc và tính bền vững lên một tập các hoạt động
  - Hướng dẫn ta hiểu, điều khiển, kiểm tra và cải thiện các hoạt động
  - Cho phép ta có được các kinh nghiệm

Tiến trình

- **Lý do để mô hình hóa một tiến trình**
  - Hình thành một cách hiểu chung
  - Tìm ra sự không nhất quán, sự dư thừa hay sự bỏ sót
  - Tìm ra và đánh giá các hoạt động phù hợp để đạt được các mục tiêu của tiến trình
  - Cụ thể hóa một tiến trình chung cho một hoàn cảnh cụ thể

Tiến trình

- **Chu kỳ sống của phần mềm**
  - Khi một tiến trình liên quan tới việc xây dựng một phần mềm, tiến trình có thể được xem như chu kỳ sống của phần mềm.

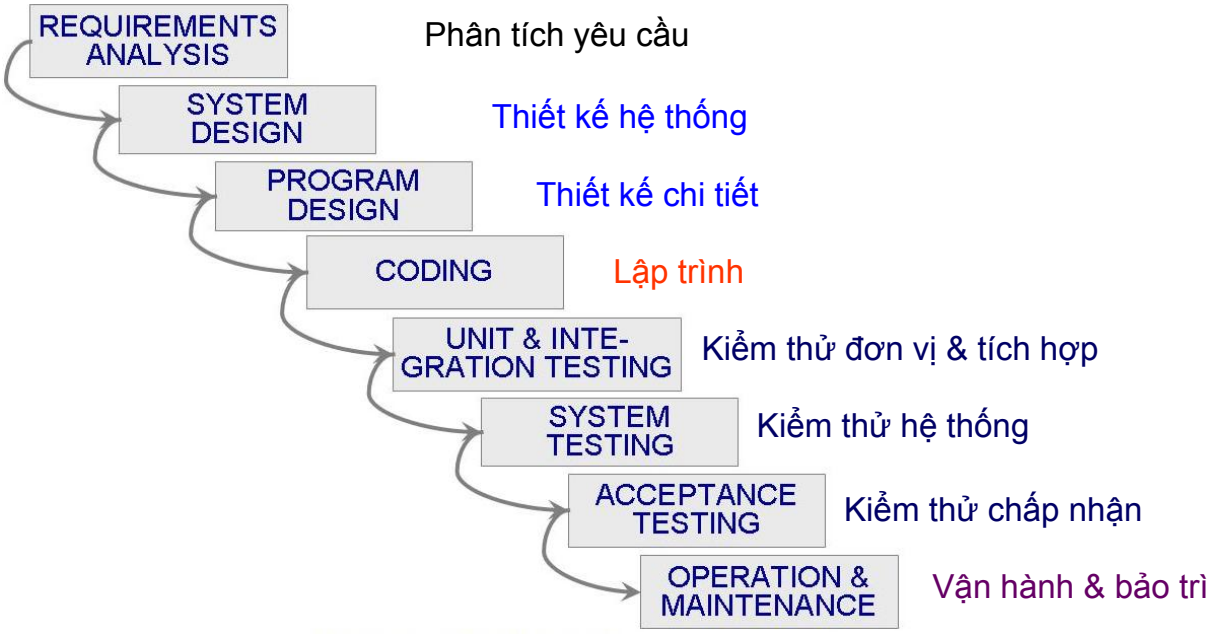
Các mô hình về tiến trình phần mềm

- Mô hình thác nước
- Mô hình chữ V
- Mô hình bản mẫu
- Mô hình phát triển ứng dụng nhanh
- Mô hình gia tăng
- Mô hình xoắn ốc
- Mô hình RUP

Mô hình thác nước (Waterfall Model)

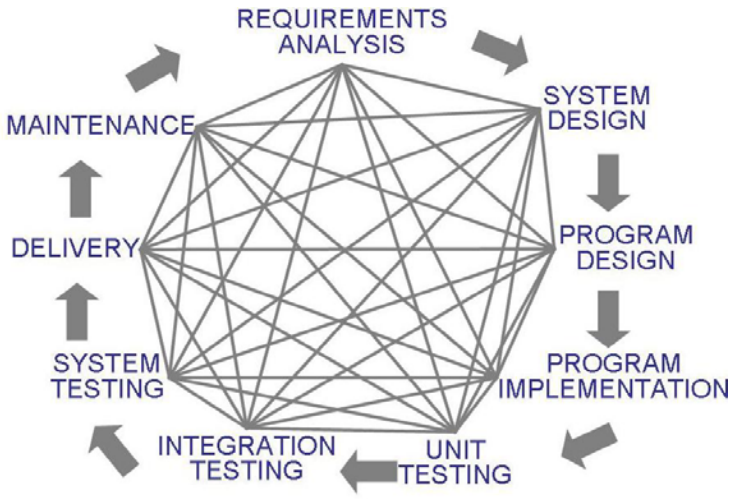
- Một trong các mô hình đầu tiên về tiến trình phần mềm
- Phù hợp với những bài toán được hiểu kỹ có rất ít hay không có các thay đổi về yêu cầu
- Đơn giản và dễ giải thích với khách hàng
- Nó biểu diễn
  - Một tổng quan mức rất cao của tiến trình phát triển
  - Một chuỗi tuần tự các hoạt động của tiến trình

Mô hình thác nước



Mô hình thác nước

- Không có sự lặp lại trong mô hình thác nước
- Thực tế, các dự án ít khi tuân theo dòng tuần tự của mô hình, mà thường có lặp lại



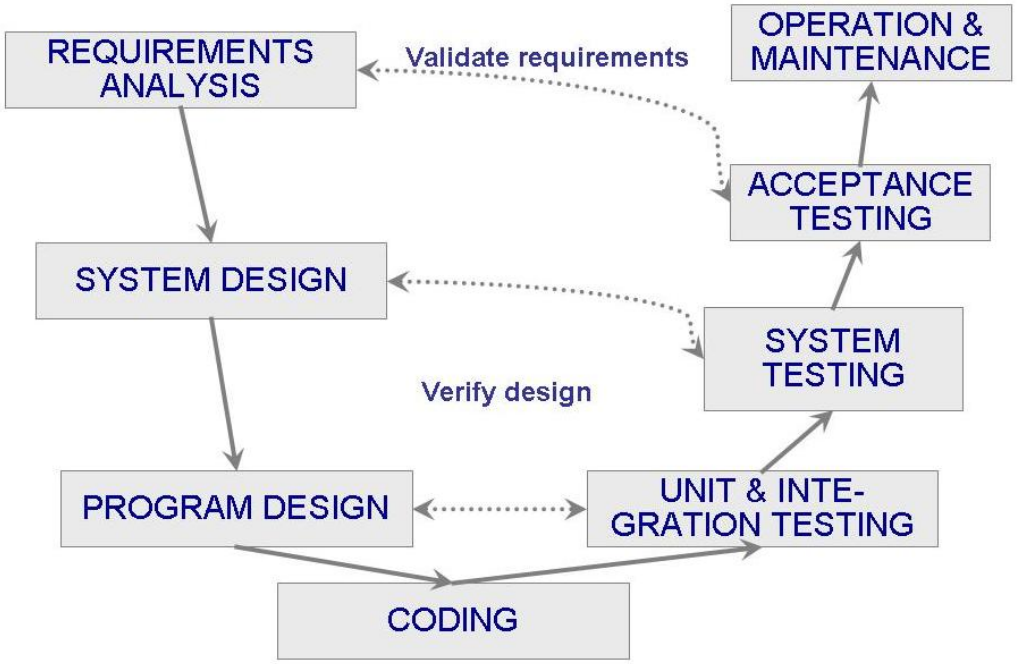
Mô hình thác nước

- Hạn chế của mô hình thác nước
  - Không có các hướng dẫn về cách thức xử lý những thay đổi về sản phẩm và hoạt động trong suốt sự phát triển
  - Xem sự phát triển phần mềm như một tiến trình sản xuất hơn là tiến trình sáng tạo
  - Không có các hoạt động lặp để tạo ra sản phẩm cuối
  - Phải chờ đợi lâu trước khi có sản phẩm cuối

Mô hình chữ V (V Model)

- Một sự biến đổi của mô hình thác nước
- Sử dụng kiểm thử đơn vị để xác minh (verify) thiết kế chi tiết
- Sử dụng kiểm thử tích hợp để xác minh thiết kế hệ thống
- Sử dụng kiểm thử chấp nhận để thẩm định (validate) các yêu cầu
- Nếu các vấn đề được tìm thấy trong suốt sự xác minh và thẩm định, phần bên trái của mô hình chữ V có thể được tái thực hiện trước khi việc kiểm thử phần bên phải được tái thực hiện

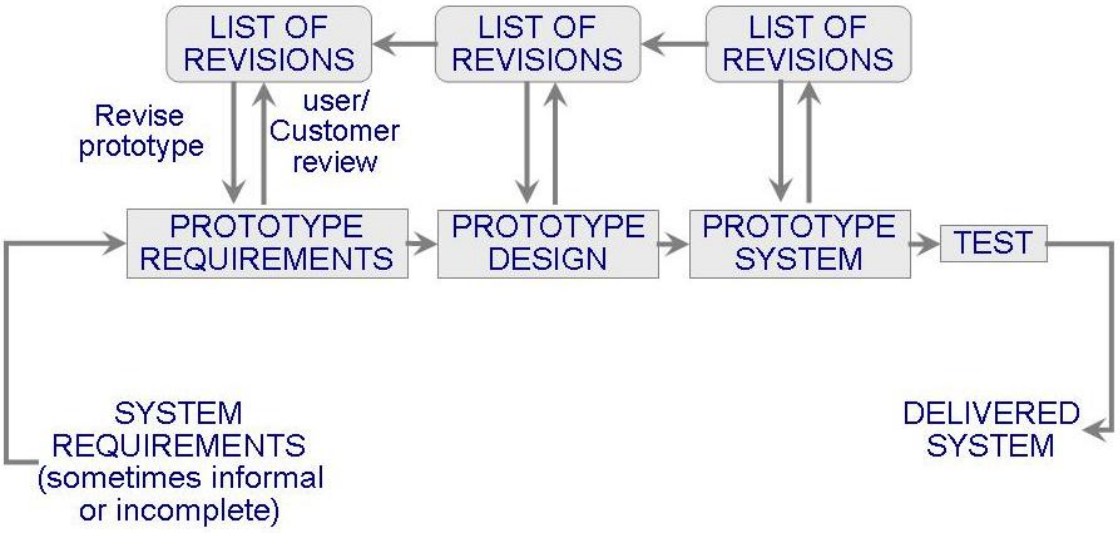
Mô hình chữ V



Mô hình bản mẫu (Prototyping Model)

- Cho phép sự nghiên cứu về các yêu cầu và thiết kế được lặp lại
- Giảm sự rủi ro và sự không chắc chắn trong phát triển
- Sử dụng mô hình bản mẫu khi các yêu cầu chưa rõ ràng

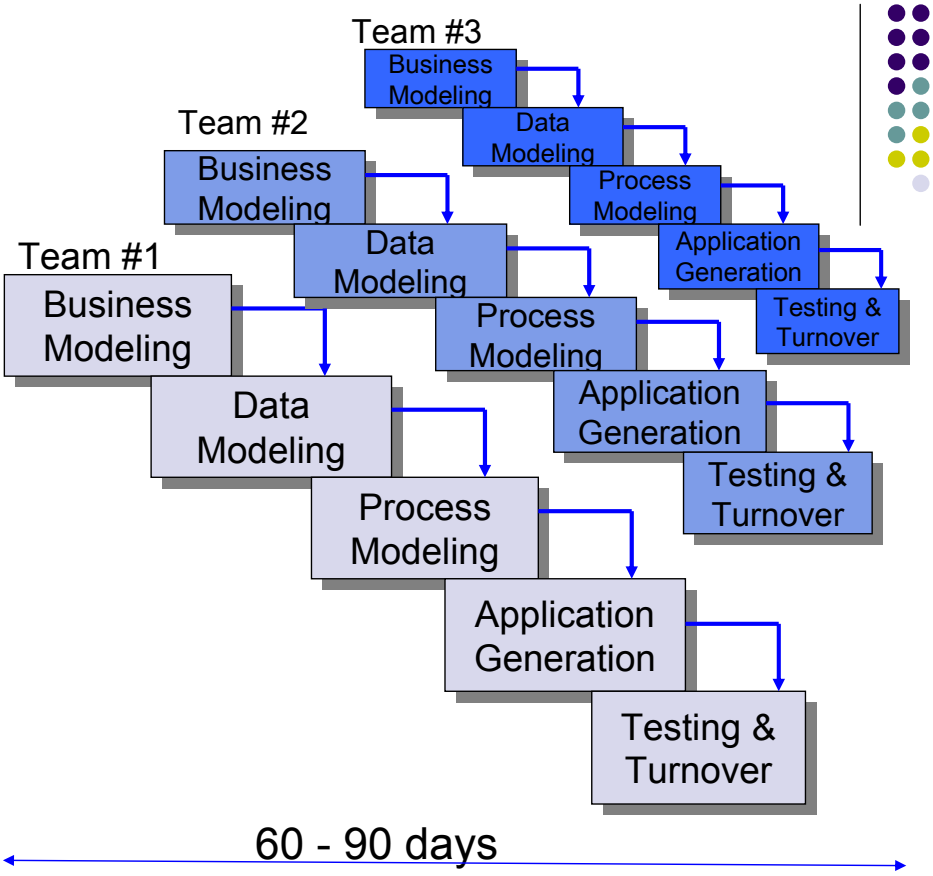
Mô hình bản mẫu



Mô hình phát triển ứng dụng nhanh (Rapid Application Development: RAD)

- Là tiến trình phát triển phần mềm gia tăng
- Thời gian phát triển phần mềm rất ngắn
- Sử dụng các kỹ thuật thế hệ thứ tư
- Xây dựng dựa trên hướng thành phần với khả năng tái sử dụng
- Gồm một số nhóm, mỗi nhóm làm 1 RAD theo các pha: Mô hình hóa nghiệp vụ, Mô hình hóa dữ liệu, Mô hình hóa xử lý, Tạo ứng dụng, Kiểm thử và đánh giá (Business, Data, Process, Application Generation, Testing)

Mô hình phát triển ứng dụng nhanh

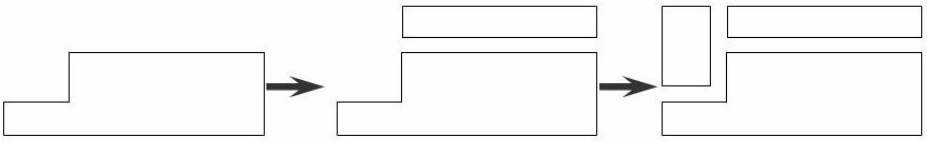


Mô hình phát triển ứng dụng nhanh

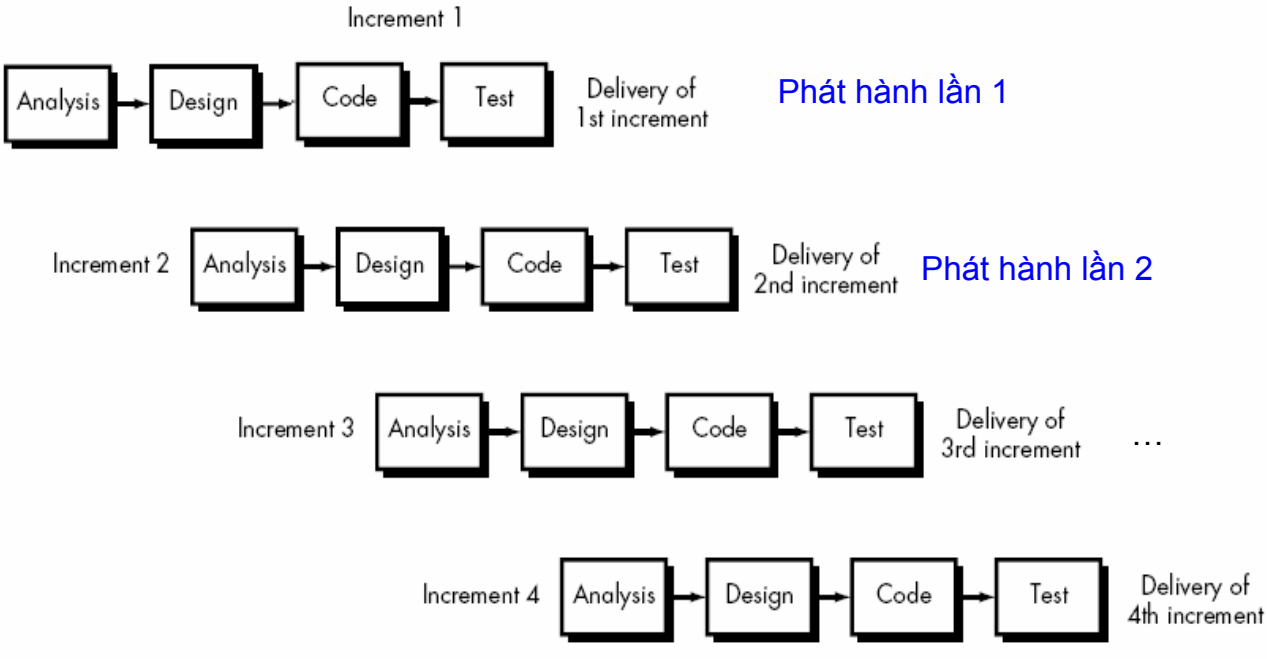
- Cần nguồn nhân lực dồi dào để tạo các nhóm cho các chức năng chính
- Yêu cầu hai bên cam kết trong thời gian ngắn phải có phần mềm hoàn chỉnh, thiếu trách nhiệm của một bên dễ làm dự án đổ vỡ
- RAD không phải tốt cho mọi ứng dụng, nhất là với ứng dụng không thể module hóa hoặc đòi hỏi tính năng cao
- RAD không phù hợp khi các rủi ro kỹ thuật cao

Mô hình gia tăng (Incremental Model)

- Kết hợp mô hình tuần tự và ý tưởng lặp lại của chế bản mẫu
- Sản phẩm lõi cho những yêu cầu cơ bản nhất của hệ thống được phát triển
- Các chức năng cho những yêu cầu khác được phát triển thêm sau (gia tăng)
- Lặp lại quy trình để hoàn thiện dần



Mô hình gia tăng



Mô hình xoắn ốc (Spiral Model)

- Được đề nghị bởi Boehm (1988)
- Kết hợp các hoạt động phát triển với sự quản lý rủi ro để giảm đến mức tối thiểu và kiểm soát các rủi ro
- Thích hợp với các hệ lớn
- Mô hình được trình bày ở dạng xoắn ốc trong đó mỗi lần lặp được biểu diễn bởi một đường vòng gồm bốn hoạt động chính
  - Lập kế hoạch
  - Xác định các mục tiêu, các lựa chọn và các ràng buộc
  - Đánh giá các lựa chọn và các rủi ro
  - Phát triển và kiểm thử

Mô hình xoắn ốc

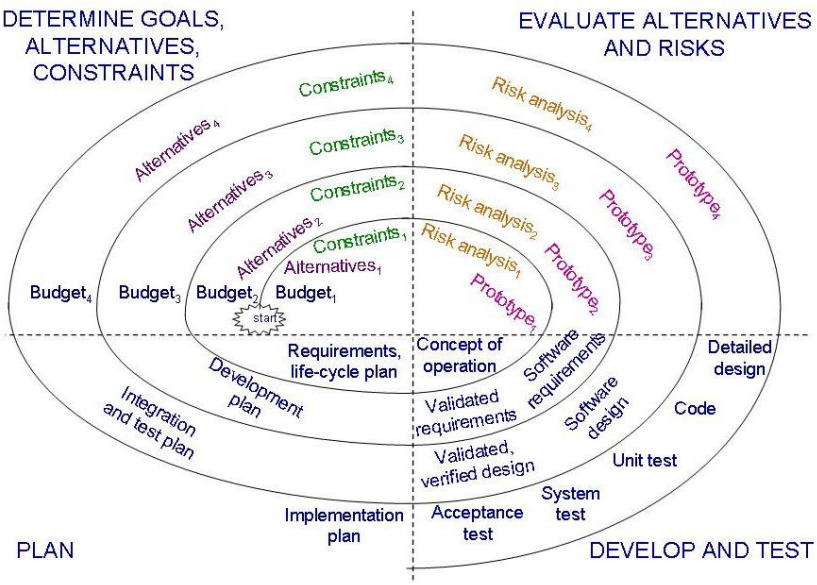
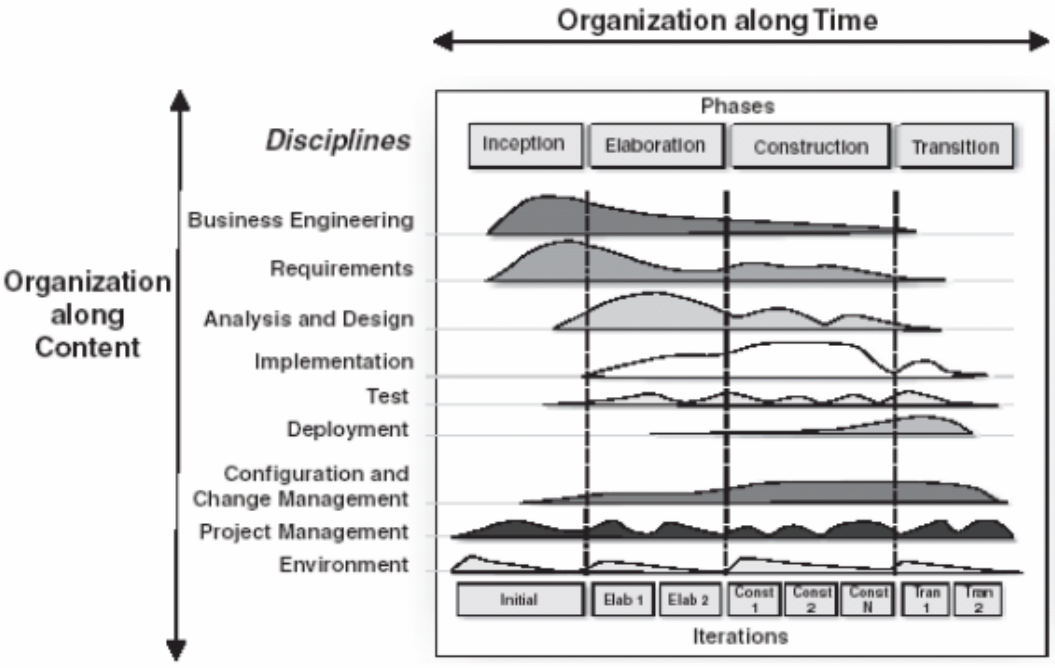


Figure 2.10 the spiral model.

RUP – Rational Unified Process

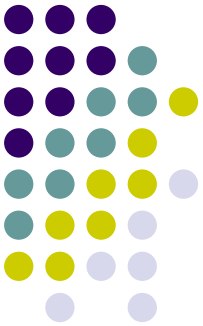
- Bổ sung cho UML
- Cách tiếp cận lặp cho các hệ thống hướng đối tượng, bao gồm các use case để mô hình hóa các yêu cầu
- Các giai đoạn của RUP
  - *Inception*: thiết lập phạm vi, giới hạn, các use case quan trọng, các kiến trúc ứng viên, các dự đoán về chi phí và kế hoạch làm việc
  - *Elaboration*: đạt được kiến trúc hoàn chỉnh, thiết lập sự hỗ trợ công cụ, có tất cả các use case, giải quyết tất cả các rủi ro chính
  - *Construction*: xây dựng tiến trình, một hay nhiều sự phát hành
  - *Transition*: phát hành ra cộng đồng người dùng, thường là một số phát hành

RUP



NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

CHƯƠNG 3 - QUẢN LÝ



NỘI DUNG

- Quản lý nhân sự
- Quản lý chất lượng
- Quản lý cấu hình
- Quản lý dự án (Lập kế hoạch và kiểm soát dự án)

Nội dung – Quản lý nhân sự

- Chọn nhân sự
- Thúc đẩy nhân sự
- Quản lý nhóm

Chọn nhân sự

- Nhân sự là thành phần quan trọng nhất của tổ chức
- Việc quản lý nhân sự kém sẽ dẫn đến sự thất bại của dự án
- Các yếu tố quản lý nhân sự
  - Không phân biệt đối xử
  - Tôn trọng
  - Lắng nghe
  - Trung thực

Chọn nhân sự

- Một công việc quản lý dự án quan trọng là chọn nhóm làm việc
- Các thông tin cần cho sự lựa chọn nhân sự gồm:
  - Thông tin được cung cấp bởi ứng viên
  - Thông tin do phỏng vấn và nói chuyện với ứng viên
  - Thông tin từ thư tiến cử hay sự giới thiệu của những người biết hay những người làm việc với ứng viên

Chọn nhân sự

- Một số lưu ý trong việc chọn nhân sự
  - Các nhà quản lý trong công ty không muốn mất người cho các dự án mới. Vì vậy, ta phải chấp nhận những người chỉ có thể làm việc bán thời gian trong dự án.
  - Các kỹ năng cần thiết cho dự án là khan hiếm => không có được nhiều ứng viên để chọn.
  - Những sinh viên mới ra trường không có nhiều kinh nghiệm cụ thể nhưng họ thường nhiệt tình và dễ học công nghệ mới
  - Sự thành thạo về kỹ thuật có thể ít quan trọng hơn các kỹ năng xã hội

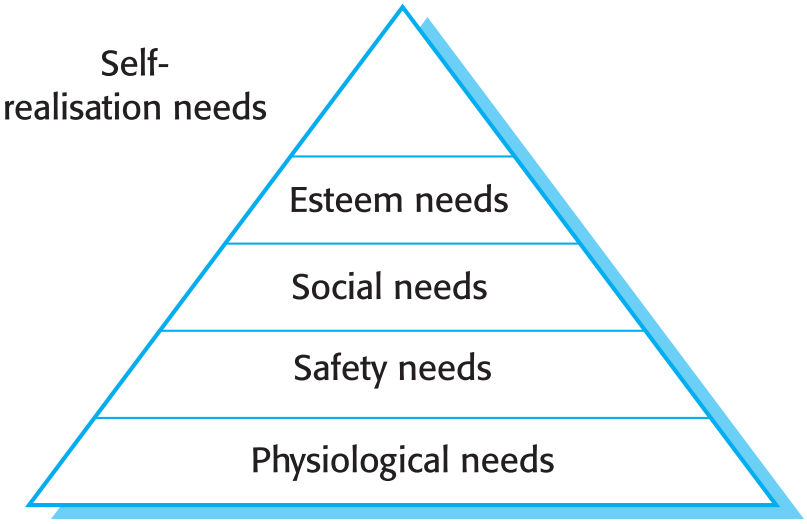
Chọn nhân sự

- Các yếu tố tác động lên việc chọn nhân sự
  - Kinh nghiệm về lĩnh vực ứng dụng
  - Kinh nghiệm về nền tảng
  - Kinh nghiệm về ngôn ngữ lập trình
  - Khả năng giải quyết vấn đề
  - Nền tảng giáo dục
  - Khả năng giao tiếp
  - Tính thích ứng
  - Thái độ
  - Tính cách

Thúc đẩy nhân sự

- Một vai trò quan trọng của nhà quản lý là thúc đẩy nhân sự làm việc trong dự án
- Động cơ thúc đẩy là một vấn đề phức tạp.
- Các loại động cơ thúc đẩy được dựa trên:
  - Các nhu cầu cơ bản (lương thực, thời gian ngủ, v.v);
  - Các nhu cầu cá nhân (sự tôn trọng, lòng tự trọng, v.v);
  - Các nhu cầu xã hội (được chấp nhận là một thành viên của nhóm, v.v)

Thúc đẩy nhân sự



Sự phân cấp các nhu cầu của con người

Thúc đẩy nhân sự

- Đảm bảo thỏa mãn các nhu cầu về:
  - Xã hội
    - Cung cấp các phương tiện giao tiếp
    - Cho phép các giao tiếp không hình thức
  - Sự quý trọng
    - Công nhận các thành tích
    - Các phần thưởng tương xứng
  - Sự phát triển năng khiếu bản thân
    - Đào tạo — những người muốn học nhiều hơn
    - Trách nhiệm

Thúc đẩy nhân sự

- Động cơ thúc đẩy còn quan tâm tới các kiểu tính cách:
  - Hướng tới công việc
  - Hướng tới bản thân
  - Hướng tới sự tương tác

Thúc đẩy nhân sự

- Tính cách hướng tới công việc
  - Động cơ thúc đẩy làm việc chính là công việc
- Tính cách hướng tới bản thân
  - Công việc là một phương tiện để đạt được các mục tiêu cá nhân
- Tính cách hướng tới sự tương tác
  - Động cơ thúc đẩy chủ yếu là sự hiện diện và các hoạt động của những người cùng làm việc

Thúc đẩy nhân sự

- Cân bằng động cơ thúc đẩy
  - Các động cơ thúc đẩy cá nhân được tạo thành từ nhiều yếu tố
  - Sự cân bằng có thể thay đổi tùy thuộc vào hoàn cảnh cá nhân và các sự kiện bên ngoài
  - Con người không chỉ được thúc đẩy bởi các yếu tố cá nhân mà còn bởi việc trở thành một phần của nhóm hay văn hóa
  - Con người làm việc vì họ được thúc đẩy bởi những người mà họ làm cùng

Quản lý nhóm

- Hầu hết hoạt động phần mềm là hoạt động nhóm vì một dự án phần mềm không thể hoàn thành bởi duy nhất một người
- Sự tương tác nhóm là yếu tố quyết định then chốt sự thực hiện của nhóm
- Tính linh động trong kết cấu nhóm bị hạn chế do các nhà quản lý phải làm cái tốt nhất mà họ có thể với nhân sự hiện có

Quản lý nhóm

- Các yếu tố chi phối đến công việc nhóm
  - Kết cấu nhóm
  - Sự gắn kết nhóm
  - Các giao tiếp nhóm
  - Tổ chức của nhóm

Quản lý nhóm

- Kết cấu nhóm
  - Nhóm được tạo thành từ những thành viên có cùng động cơ thúc đẩy có thể có vấn đề
    - Hướng công việc — mỗi người muốn làm công việc của chính họ
    - Hướng bản thân — mỗi người đều muốn lãnh đạo
    - Hướng tương tác — nói quá nhiều

Quản lý nhóm

- Kết cấu nhóm
  - Một nhóm hiệu quả phải có sự cân bằng của tất cả các tính cách
    - Người hướng tới công việc thường mạnh về kỹ thuật
    - Người hướng tới bản thân thường thúc đẩy sự hoàn thành công việc
    - Người hướng tới sự tương tác giúp cho sự giao tiếp trong nhóm thuận tiện hơn

Quản lý nhóm

- Kết cấu nhóm
  - Lãnh đạo nhóm
    - Trách nhiệm của lãnh đạo nhóm là:
      - Cung cấp các chỉ dẫn kỹ thuật và các quản lý dự án
      - Phải nắm được công việc hàng ngày của nhóm để đảm bảo mọi người làm việc hiệu quả và làm việc với nhà quản lý dự án theo kế hoạch của dự án
    - Trong một nhóm, có thể có cả 1 lãnh đạo kỹ thuật và 1 lãnh đạo quản lý
    - Sự lãnh đạo nhóm dựa trên sự tôn trọng, sự lãnh đạo dân chủ hiệu quả hơn sự lãnh đạo chuyên quyền

Sự gắn kết nhóm

- Trong một nhóm gắn kết, các thành viên xem nhóm là quan trọng hơn bất cứ cá nhân nào trong nhóm
- Thuận lợi của nhóm gắn kết:
  - Chuẩn về chất lượng nhóm được phát triển
  - Các thành viên trong nhóm làm việc cùng với nhau vì thế các hạn chế về sự không hiểu biết giảm
  - Các thành viên trong nhóm học lẫn nhau và biết được công việc của nhau

Sự gắn kết nhóm

- Phát triển tính gắn kết
  - Tính gắn kết bị ảnh hưởng bởi các yếu tố như văn hóa của tổ chức, các tính cách trong nhóm
  - Tính gắn kết có thể được khuyến khích thông qua
    - Tổ chức các sự kiện xã hội
    - Phát triển một tên riêng và một lĩnh vực của nhóm
    - Thực hiện các hoạt động xây dựng nhóm
  - Tính mở của thông tin là một cách đơn giản để đảm bảo tất cả các thành viên trong nhóm cảm thấy là một phần của nhóm

Sự gắn kết nhóm

- Những vấn đề có thể xảy ra trong các nhóm gắn kết:
  - Chống lại người lãnh đạo mới được chỉ định từ bên ngoài nhóm.
    - Giải quyết: chọn lãnh đạo mới là người trong nhóm
  - Suy nghĩ nhóm => các khả năng phê bình bị xói mòn
    - Giải quyết: tổ chức các buổi họp chính thức và mời chuyên gia từ bên ngoài phê bình các quyết định của nhóm

Giao tiếp nhóm

- Các giao tiếp tốt là cần thiết giúp cho làm việc nhóm hiệu quả
- Các thông tin phải được trao đổi: tình trạng công việc, các quyết định thiết kế, những thay đổi đối với các quyết định trước đó
- Các giao tiếp tốt còn làm gia tăng tính gắn kết nhóm vì nó thúc đẩy sự hiểu biết

Giao tiếp nhóm

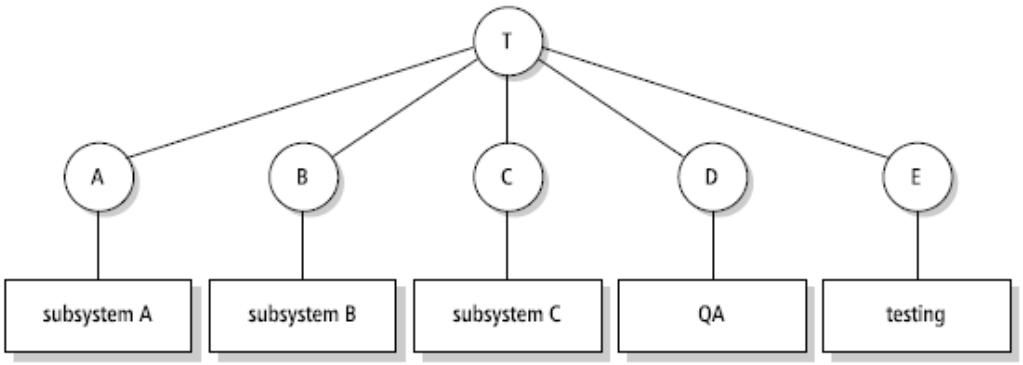
- Các yếu tố tác động lên tính hiệu quả trong giao tiếp
  - Qui mô nhóm
    - Nhóm càng lớn, mọi người càng khó giao tiếp hiệu quả với các thành viên khác trong nhóm
  - Cấu trúc nhóm
    - Sự giao tiếp là tốt hơn trong các nhóm được tổ chức tự do hơn là trong các nhóm được cấu trúc phân cấp
  - Kết cấu nhóm
    - Sự giao tiếp là tốt hơn nếu nhiều thành viên trong nhóm có tính cách khác nhau và có cả nam và nữ
  - Môi trường làm việc tự nhiên
    - Việc tổ chức nơi làm việc tốt có thể khuyến khích sự giao tiếp

Tổ chức nhóm

- Các tổ chức nhóm
  - Tổ chức phân cấp
  - Tổ chức ma trận
  - Nhóm lập trình viên chính
  - Nhóm SWAT
  - Nhóm lập trình nhanh

Tổ chức nhóm

- Tổ chức phân cấp
  - Phân nhóm chức năng
  - Phản ánh cấu trúc tổng thể của dự án
  - Nhược điểm: Khoảng cách giao tiếp xa, thông tin nhiễu



Tổ chức nhóm

- Tổ chức ma trận
  - Các nhóm bộ phận có chuyên môn sâu
  - Tham gia bán thời gian vào các dự án khác nhau
  - Người quản lí dự án chịu trách nhiệm cho sự thành công toàn bộ dự án và nâng cao kiến thức chuyên môn của nhóm.

	real-time programming	graphics	databases	QA	testing
project C	X			X	X
project B	X		X	X	X
project A		X	X	X	X

Tổ chức nhóm

- Nhóm lập trình viên chính
  - Hạt nhân của mỗi nhóm gồm 3 người:
    - Trưởng nhóm: thiết kế và cài đặt phần chính của hệ thống
    - Trợ lý: giúp việc cho trưởng nhóm
    - Người quản lí tài liệu
  - Cần có một trưởng nhóm giỏi kỹ thuật và năng lực quản lí tốt
  - Lưu ý: công việc không có cấu trúc, không có người phân tích, người thiết kế, lập trình viên…

Tổ chức nhóm

- Nhóm SWAT(Skilled With Advanced Tools)
  - Nhóm nhỏ: 4-5 người
  - Thường áp dụng trong tiến trình theo mô hình gia tăng
  - Dùng ngôn ngữ cấp cao, các gói dùng lại, các phần mềm sinh mã.
  - Nhóm trưởng là người giỏi

Tổ chức nhóm

- Nhóm lập trình nhanh
  - Làm việc theo cặp
    - Người chính
    - Người phụ
    - Đổi vai trò

Nội dung – Quản lý chất lượng

- Quản lý chất lượng phần mềm
- Đảm bảo chất lượng và các chuẩn
- Lập kế hoạch chất lượng
- Kiểm soát chất lượng

Quản lý chất lượng phần mềm

- Quản lý chất lượng phần mềm
  - Liên quan tới việc đảm bảo một sản phẩm phần mềm đạt được mức chất lượng được yêu cầu
  - Liên quan đến việc định nghĩa các thủ tục và các chuẩn chất lượng phù hợp và đảm bảo rằng tất cả các chuẩn và thủ tục này được tuân theo
  - Hướng tới phát triển một 'văn hóa chất lượng' nơi chất lượng được xem là trách nhiệm của mọi người

Quản lý chất lượng phần mềm

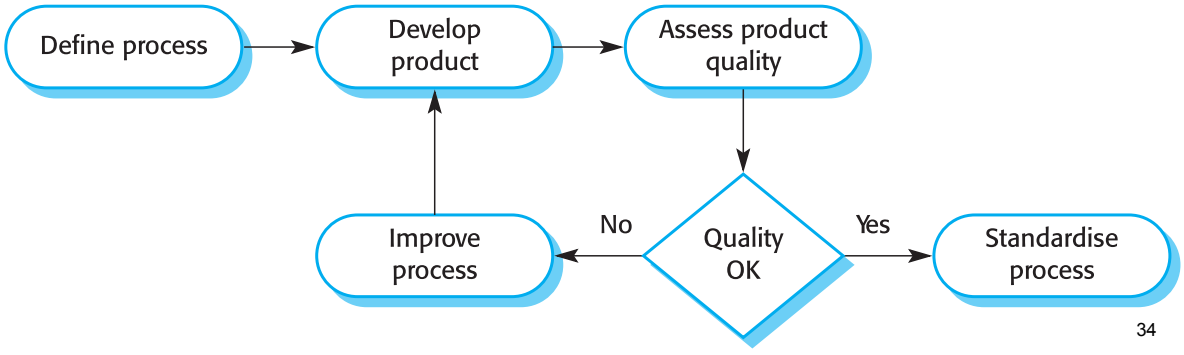
- Phạm vi của quản lý chất lượng
  - Quản lý chất lượng là đặc biệt quan trọng đối với các hệ thống phức tạp và lớn. Tài liệu chất lượng là hồ sơ về tiến trình và hỗ trợ tính liên tục phát triển khi nhóm phát triển thay đổi.
  - Đối với các hệ thống nhỏ hơn, quản lý chất lượng cần ít tài liệu hơn và nên tập trung vào việc củng cố văn hóa chất lượng.

Quản lý chất lượng phần mềm

- Các hoạt động chính của quản lý chất lượng
  - Đảm bảo chất lượng
    - Thiết lập thủ tục tổ chức và các chuẩn về chất lượng
  - Lập kế hoạch chất lượng
    - Chọn các thủ tục và các chuẩn phù hợp với một dự án cụ thể và hiệu chỉnh chúng khi cần
  - Kiểm soát chất lượng
    - Đảm bảo rằng nhóm phát triển phần mềm tuân theo các thủ tục và chuẩn
- Quản lý chất lượng nên tách biệt khỏi quản lý dự án để đảm bảo sự độc lập

Quản lý chất lượng phần mềm

- Chất lượng sản phẩm và quy trình
  - Chất lượng sản phẩm được phát triển bị ảnh hưởng bởi chất lượng quy trình sản xuất
  - Một cách tiếp cận dựa trên quy trình để đạt được chất lượng sản phẩm



Quản lý chất lượng phần mềm

- Chất lượng của sản phẩm và quy trình
  - Trong phát triển phần mềm, mối quan hệ giữa chất lượng sản phẩm và chất lượng quy trình là phức tạp vì
    - Việc áp dụng các kinh nghiệm và các kỹ năng cá nhân là đặc biệt quan trọng trong phát triển phần mềm
    - Các yếu tố bên ngoài như tính mới lạ của ứng dụng hay kế hoạch phát triển gấp có thể làm suy giảm chất lượng sản phẩm
  - Một số thuộc tính chất lượng phần mềm khó đo lường => khó đánh giá được cách mà các đặc điểm của quy trình tác động đến các thuộc tính đó

Quản lý chất lượng phần mềm

- Quản lý chất lượng quy trình liên quan tới:
  - Định nghĩa các chuẩn quy trình như khi nào và bằng cách nào các xem lại (review) được quản lý, quản lý cấu hình, v.v
  - Giám sát quy trình phát triển để đảm bảo các chuẩn được tuân theo
  - Báo cáo quy trình phần mềm với quản lý dự án và khách hàng mua phần mềm

Đảm bảo chất lượng và các chuẩn

- Các chuẩn
  - Là chìa khóa của sự quản lý chất lượng hiệu quả
  - Có thể là các chuẩn của tổ chức, của quốc gia hay của quốc tế
  - Các loại chuẩn:
    - Chuẩn sản phẩm
      - Các chuẩn áp dụng cho sản phẩm phần mềm đang được phát triển.
      - Chúng gồm các chuẩn như các chuẩn tư liệu, các chuẩn lập trình

Đảm bảo chất lượng và các chuẩn

- Các chuẩn
  - Các loại chuẩn
    - Chuẩn quy trình:
      - Các chuẩn định nghĩa các quy trình mà chúng nên được tuân theo trong suốt sự phát triển phần mềm.
      - Chúng bao gồm các định nghĩa về những quy trình đặc tả, thiết kế, thẩm định và sự mô tả về các tài liệu được viết trong các quy trình đó

Đảm bảo chất lượng và các chuẩn

- Các chuẩn quy trình và sản phẩm

Product standards	Process standards
Design review form	Design review conduct
Requirements document structure	Submission of documents to CM
Method header format	Version release process
Java programming style	Project plan approval process
Project plan format	Change control process
Change request form	Test recording process

Đảm bảo chất lượng và các chuẩn

- Tầm quan trọng của các chuẩn
  - Là sự tóm lược thực tiễn tốt nhất
  - Cung cấp một khung để thực hiện quy trình đảm bảo chất lượng
  - Hỗ trợ tính liên tục nơi công việc được thực hiện bởi một người nay được giao cho người khác

Đảm bảo chất lượng và các chuẩn

- Các vấn đề về chuẩn
  - Chúng có thể được xem là không liên quan và không được cập nhật bởi các kỹ sư phần mềm
  - Chúng thường đòi hỏi quá nhiều thực hiện rườm rà và có thể buồn tẻ

Đảm bảo chất lượng và các chuẩn

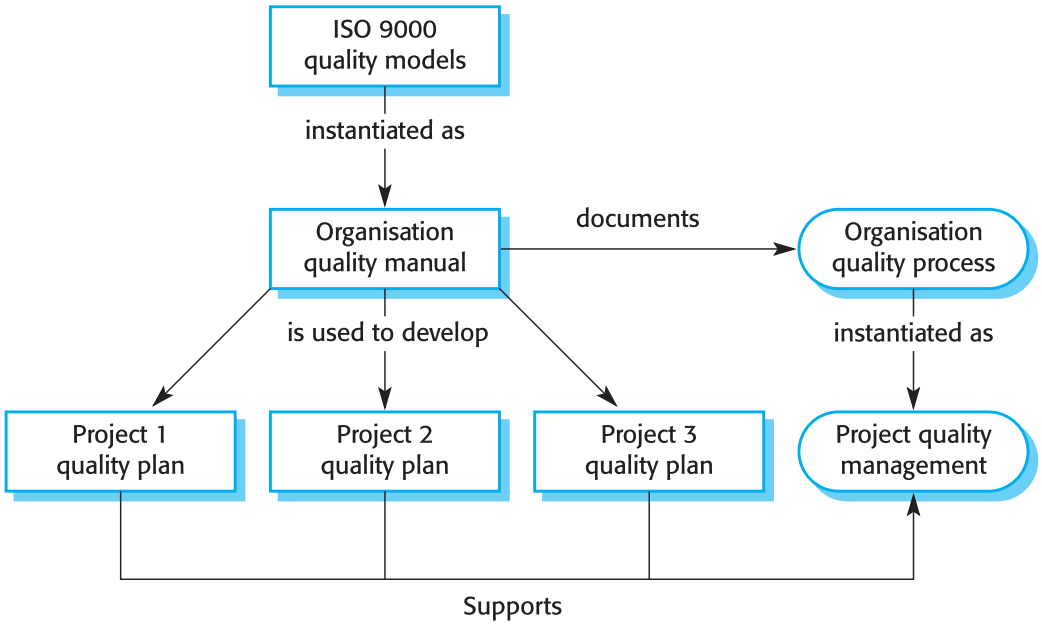
- Để tránh các vấn đề về chuẩn, nhà quản lý chất lượng nên thực hiện:
  - Mời các kỹ sư phần mềm tham gia vào việc chọn các chuẩn sản phẩm
  - Xem lại và hiệu chỉnh các chuẩn để phản ánh các công nghệ đang thay đổi
  - Cung cấp các công cụ phần mềm để hỗ trợ các chuẩn nếu có thể

Đảm bảo chất lượng và các chuẩn

- ISO 9000
  - Một tập chuẩn quốc tế cho quản lý chất lượng
  - Phù hợp với nhiều tổ chức từ công nghiệp sản xuất tới kinh doanh dịch vụ

Đảm bảo chất lượng và các chuẩn

- ISO 9000 và quản lý chất lượng



Đảm bảo chất lượng và các chuẩn

- ISO 9001
  - ISO 9001 phù hợp với các tổ chức thiết kế, phát triển và bảo trì sản phẩm
  - ISO 9001 là một mô hình chung của quy trình chất lượng mà nó phải được cụ thể hóa cho từng công ty bằng cách sử dụng các thủ tục và các chuẩn tổ chức mà công ty nên định nghĩa

Đảm bảo chất lượng và các chuẩn

- ISO 9001 bao phủ các phạm vi sau

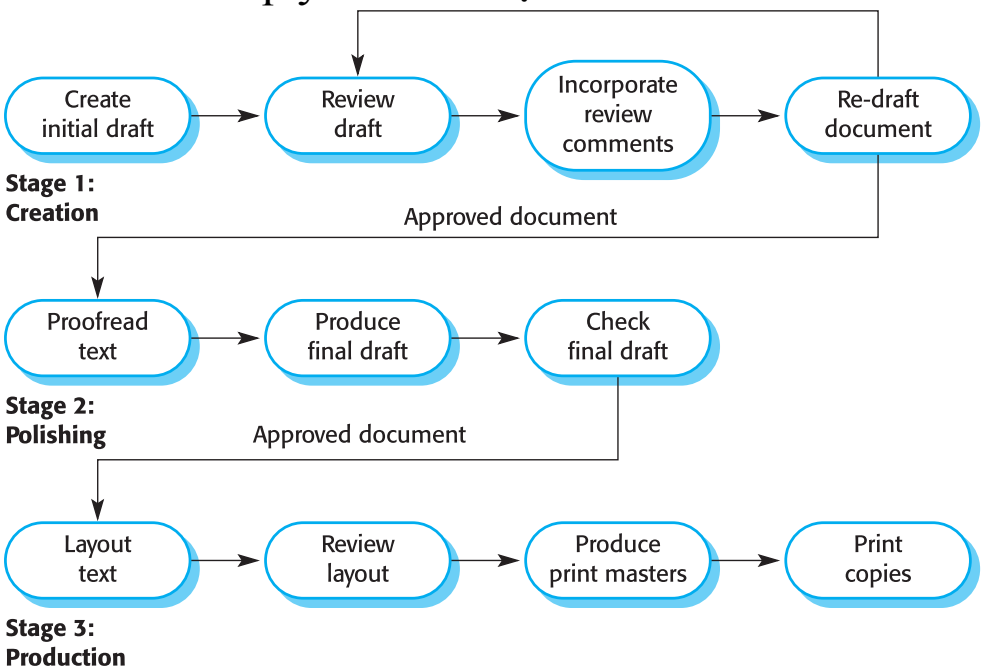
Management responsibility	Quality system
Control of nonconforming products	Design control
Handling, storage, packaging and delivery	Purchasing
Purchaser-supplied products	Product identification and traceability
Process control	Inspection and testing
Inspection and test equipment	Inspection and test status
Contract review	Corrective action
Document control	Quality records
Internal quality audits	Training
Servicing	Statistical techniques

Đảm bảo chất lượng và các chuẩn

- Các chuẩn tư liệu
  - Đặc biệt quan trọng vì tài liệu là cách hữu hình duy nhất để biểu diễn phần mềm và quy trình phần mềm
  - Ba loại chuẩn tư liệu
    - Chuẩn quy trình tư liệu: liên quan tới cách các tài liệu nên được phát triển, thẩm định và duy trì
    - Chuẩn tài liệu: chi phối nội dung, cấu trúc và sự trình bày của các tài liệu
    - Chuẩn trao đổi tài liệu: đảm bảo rằng tất cả các bản sao điện tử của các tài liệu là tương thích

Đảm bảo chất lượng và các chuẩn

- Một mô hình về quy trình tư liệu



Đảm bảo chất lượng và các chuẩn

- Chuẩn tài liệu
  - Các chuẩn nhận dạng tài liệu: cách các tài liệu được nhận biết là duy nhất
  - Các chuẩn cấu trúc tài liệu: cấu trúc chuẩn cho các tài liệu của dự án
  - Các chuẩn trình bày tài liệu: định nghĩa các font chữ, kiểu chữ, sử dụng các logo, v.v.
  - Chuẩn cập nhật tài liệu: định nghĩa cách các thay đổi so các phiên bản trước được phản ánh trong tài liệu

Đảm bảo chất lượng và các chuẩn

- Chuẩn trao đổi tài liệu
  - Các chuẩn trao đổi cho phép các tài liệu điện tử được nhận, được gửi, v.v.
  - Các tài liệu được tạo ra bằng cách sử dụng các hệ thống khác nhau và trên các máy tính khác nhau. Thậm chí khi các công cụ chuẩn được sử dụng, các chuẩn được cần đến để định nghĩa các quy tắc sử dụng chúng

Lập kế hoạch chất lượng

- Kế hoạch chất lượng trình bày các chất lượng sản phẩm được mong đợi và mô tả cách mà chúng được đánh giá cũng như định nghĩa các thuộc tính chất lượng quan trọng nhất
- Kế hoạch chất lượng nên định nghĩa quy trình đánh giá chất lượng
- Nó trình bày những chuẩn tổ chức nào nên được áp dụng và, khi cần thiết, định nghĩa các chuẩn mới

Lập kế hoạch chất lượng

- Cấu trúc của kế hoạch chất lượng
  - Giới thiệu sản phẩm
  - Các kế hoạch cho sản phẩm
  - Các mô tả quy trình
  - Mục tiêu chất lượng
  - Rủi ro và quản lý rủi ro
- Kế hoạch chất lượng nên là tài liệu ngắn gọn và súc tích

Lập kế hoạch chất lượng

- Các thuộc tính chất lượng phần mềm

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

Kiểm soát chất lượng

- Kiểm soát chất lượng đòi hỏi việc giám sát quy trình phát triển phần mềm để đảm bảo các thủ tục và các chuẩn đang được tuân theo
- Hai cách tiếp cận kiểm soát quy trình
  - Các xem lại chất lượng
  - Sự đo lường phần mềm và sự đánh giá phần mềm tự động

Kiểm soát chất lượng

- Xem lại chất lượng
  - Đây là một phương pháp cơ bản công nhận chất lượng của quy trình hay sản phẩm
  - Một nhóm kiểm tra một phần hay toàn bộ quy trình hay hệ thống và các tư liệu của nó để tìm ra các vấn đề tiềm ẩn
  - Mục đích của xem lại chất lượng là phát hiện ra các nhược điểm của hệ thống và các mâu thuẫn
  - Bất cứ tài liệu nào được tạo ra trong quy trình đều có thể được xem lại
  - Các nhóm xem lại nên tương đối nhỏ và các buổi xem lại nên khá ngắn

Kiểm soát chất lượng

- Các kiểu xem lại

Review type	Principal purpose
Design or program inspections	To detect detailed errors in the requirements, design or code. A checklist of possible errors should drive the review.
Progress reviews	To provide information for management about the overall progress of the project. This is both a process and a product review and is concerned with costs, plans and schedules.
Quality reviews	To carry out a technical analysis of product components or documentation to find mismatches between the specification and the component design, code or documentation and to ensure that defined quality standards have been followed.

Nội dung – Quản lý cấu hình

- Quản lý cấu hình (Configuration Management - CM)
- Lập kế hoạch quản lý cấu hình
- Quản lý sự thay đổi
- Quản lý phiên bản và phát hành
- Xây dựng hệ thống

Quản lý cấu hình

- CM là sự phát triển và ứng dụng của các thủ tục và chuẩn để quản lý một sản phẩm phần mềm đang tiến hóa
- CM có thể được xem là một phần của quy trình quản lý chất lượng tổng quan hơn
- Khi được phát hành tới CM, các hệ thống phần mềm đôi khi được gọi là các baseline vì chúng là điểm bắt đầu cho sự phát triển xa hơn

Quản lý cấu hình

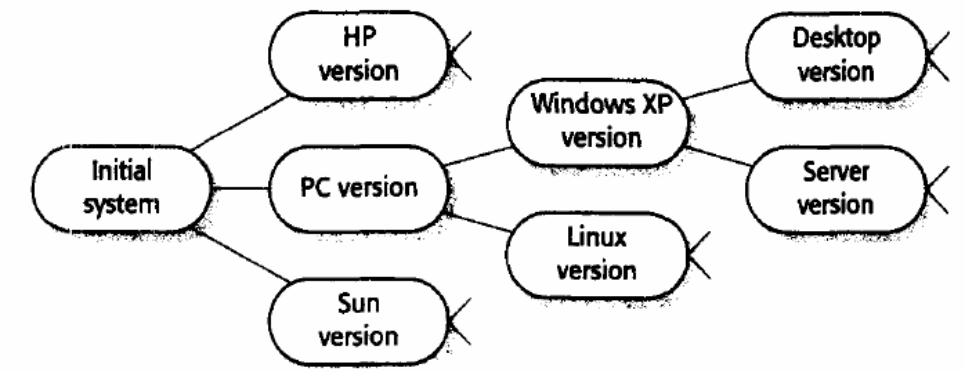
- Thủ tục CM định nghĩa:
  - Cách lưu giữ và xử lý các thay đổi hệ thống được đề nghị
  - Cách liên kết các thay đổi này với các bộ phận phần mềm và các phương thức được sử dụng để nhận dạng các phiên bản khác nhau của hệ thống

Quản lý cấu hình

- Các chuẩn của CM
  - Định nghĩa và sử dụng các chuẩn CM là rất cần thiết để xác nhận chất lượng
  - Các chuẩn có thể được dựa trên các chuẩn CM bên ngoài tổng quát và được điều chỉnh cho phù hợp với với môi trường cụ thể của tổ chức
  - Các chuẩn nên định nghĩa các thành phần (item) được nhận dạng, cách các thay đổi được kiểm soát và cách các phiên bản mới được quản lý

Quản lý cấu hình

- Các phiên bản mới của các hệ thống phần mềm được tạo ra khi chúng:
  - Dùng cho các máy/ hệ điều hành khác nhau
  - Cung cấp các chức năng khác
  - Đáp ứng các yêu cầu đặc biệt của người dùng



Lập kế hoạch quản lý cấu hình

- Kế hoạch quản lý cấu hình
  - Định nghĩa những cái được quản lý (thành phần cấu hình) và một sơ đồ được dùng để nhận dạng những thành phần đó
  - Định nghĩa người có trách nhiệm đối với các thủ tục CM và gửi các thành phần cấu hình tới nhóm quản lý cấu hình
  - Định nghĩa các chính sách để quản lý phiên bản và kiểm soát sự thay đổi
  - Xác định các công cụ mà ta nên sử dụng để quản lý cấu hình và quy trình sử dụng chúng
  - Định nghĩa cơ sở dữ liệu CM để lưu thông tin cấu hình và những thông tin khác nên được lưu trong CSDL đó

Lập kế hoạch quản lý cấu hình

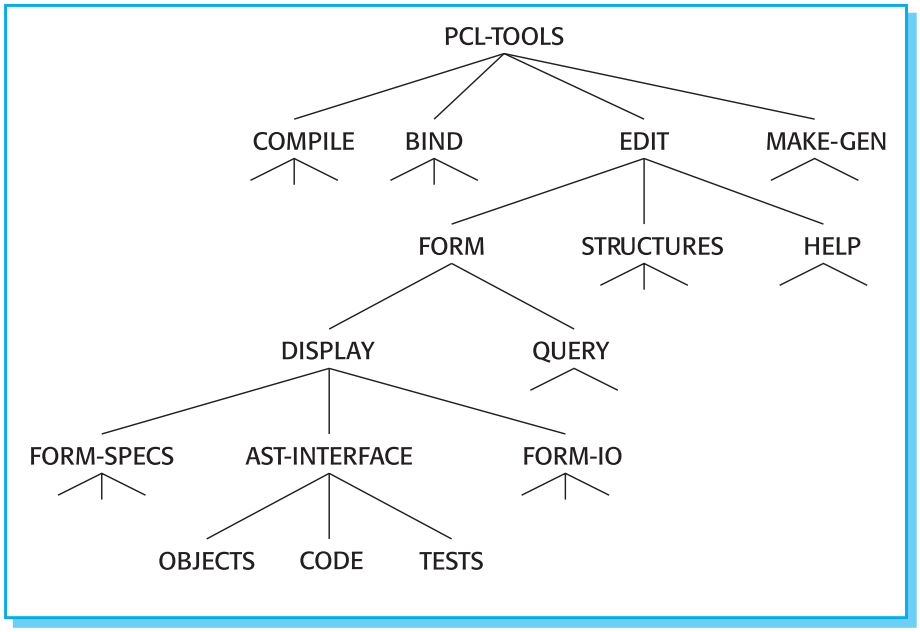
- Nhận dạng các thành phần cấu hình
  - Các dự án lớn thường tạo ra hàng ngàn tài liệu mà chúng phải được nhận dạng là duy nhất
  - Một số tài liệu này phải được bảo quản trong suốt thời gian sống của phần mềm
  - Sơ đồ phân cấp với với các tên đa mức có thể là một phương pháp uyển chuyển nhất

Lập kế hoạch quản lý cấu hình

- Nhận dạng các thành phần cấu hình
  - Các thành phần cấu hình:
    - Các đặc tả
    - Các thiết kế
    - Các chương trình
    - Dữ liệu kiểm thử
    - Tài liệu hướng dẫn người sử dụng

Lập kế hoạch quản lý cấu hình

- Phân cấp cấu hình



Lập kế hoạch quản lý cấu hình

- Cơ sở dữ liệu của quản lý cấu hình
  - Tất cả các thông tin CM nên được lưu trong cơ sở dữ liệu cấu hình
  - Nó còn cho phép các truy vấn về quản lý cấu hình như:
    - Ai có một phiên bản hệ thống cụ thể?
    - Phần cứng và hệ điều hành nào được yêu cầu cho một phiên bản cụ thể?
    - Những phiên bản nào bị ảnh hưởng bởi sự thay đổi của thành phần X?
    - Có bao nhiêu lỗi được báo cáo trong phiên bản T?

Lập kế hoạch quản lý cấu hình

- Cơ sở dữ liệu của quản lý cấu hình
  - Có thể là một phần của môi trường được tích hợp nhằm hỗ trợ phát triển phần mềm
    - Cơ sở dữ liệu CM và các tài liệu được quản lý tất cả được lưu giữ trong cùng hệ thống
  - Các công cụ CASE có thể được tích hợp để liên kết một cách trực tiếp các thay đổi với các tài liệu và các bộ phận bị ảnh hưởng bởi sự thay đổi
  - Một cách phổ biến hơn, cơ sở dữ liệu CM được lưu tách biệt vì nó rẻ hơn và linh động hơn

Quản lý sự thay đổi

- Quản lý sự thay đổi
  - Các yêu cầu thay đổi đối với hệ thống phần mềm có thể bắt nguồn từ
    - Người dùng
    - Nhà phát triển
    - Áp lực thị trường
  - Quản lý sự thay đổi liên quan tới việc theo dõi các thay đổi này và đảm bảo rằng chúng được thực hiện theo cách hiệu quả nhất về chi phí

Quản lý sự thay đổi

- Qui trình quản lý sự thay đổi

```
Request change by completing a change request form
Analyze change request
if change is valid then
    Assess how change might be implemented
    Assess change cost
    Record change request in database
    Submit request to change control board
    if change is accepted then
        repeat
            make changes to software
            record changes and link to associated change request
            submit changed software for quality approval
        until software quality is adequate
        create new system version
    else
        reject change request
else
    reject change request
```


Quản lý sự thay đổi

- Biểu mẫu yêu cầu thay đổi (change request form)
  - Sự định nghĩa của một biểu mẫu yêu cầu thay đổi là một phần của quy trình lập kế hoạch CM
  - Biểu mẫu này lưu sự thay đổi được đề nghị, người yêu cầu thay đổi, lý do tại sao sự thay đổi này được đề nghị và tính cấp bách của sự thay đổi
  - Nó còn lưu ước lượng về sự thay đổi, phân tích ảnh hưởng, chi phí thay đổi và các đề nghị
  - …

Quản lý sự thay đổi

Change Request Form

Project: Proteus/PCL-Tools **Number:** 23/02
Change requester: I. Sommerville **Date:** 1/12/02
Requested change: When a component is selected from the structure, display the name of th file where it is stored.

Change analyser: G. Dean **Analysis date:** 10/12/02
Components affected: Display-Icon.Select, Display Icon.Display

Associated components: FileTable

Change assessment: Relatively simple to implement as a file name table is available. Requires the design and implementation of a display field. No changes to associated components are required.

Change priority: Low
Change implementation
Estimated effort: 0.5 days
Date to CCB: 15/12/02 **CCB decision date:** 1/2/03
CCB decision: Accept change. Change to be implemented in Release 2.1.
Change implementor: **Date of change:**
Date submitted to QA **QA decision:**
Date submitted to CM:
Comments

Quản lý sự thay đổi

- Ban kiểm soát sự thay đổi
  - Các thay đổi nên được xem lại bởi một nhóm những người quyết định xem chúng có mang lại lợi nhuận hay không theo quan điểm chiến lược và tổ chức hơn là theo quan điểm kỹ thuật
  - Ban kiểm soát sự thay đổi nên là một nhóm độc lập của dự án

Quản lý phát hành và phiên bản

- Phát triển một sơ đồ nhận dạng các phiên bản của hệ thống
- Lập kế hoạch khi một phiên bản của hệ thống mới được tạo ra
- Đảm bảo rằng các công cụ và thủ tục quản lý phiên bản được áp dụng một cách đúng đắn
- Lập kế hoạch phân phối các phát hành của hệ thống mới

Quản lý phát hành và phiên bản

- Phiên bản / Biến thể / Phát hành
  - Phiên bản (version): Một thể hiện của hệ thống mà nó khác biệt chức năng với các thể hiện khác của hệ thống theo cách nào đó
  - Biến thể (variant): Một thể hiện của hệ thống mà nó giống về chức năng nhưng khác về phi chức năng với các thể hiện khác của hệ thống
  - Phát hành (release): Một thể hiện của hệ thống mà nó được phân phối cho người dùng bên ngoài nhóm phát triển

Quản lý phát hành và phiên bản

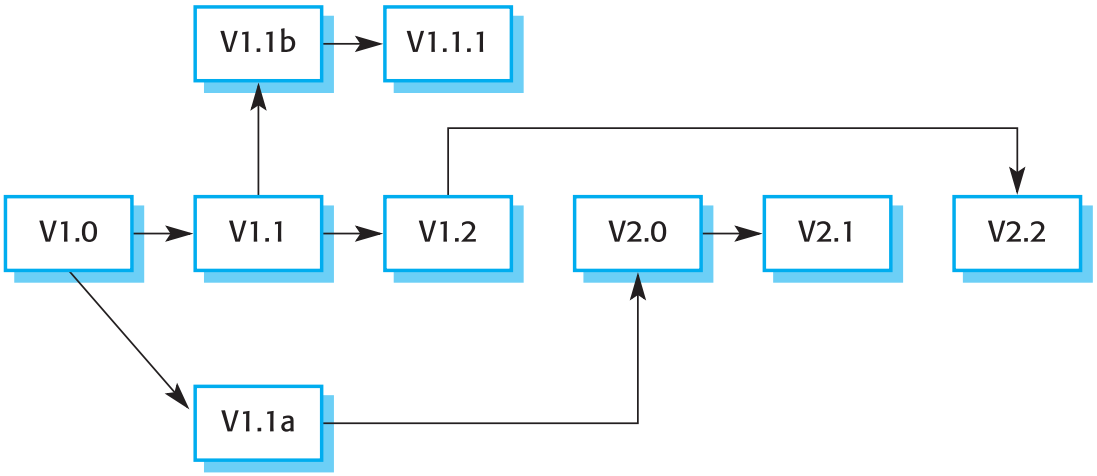
- Nhận dạng phiên bản
  - Các thủ tục nhận dạng phiên bản nên định nghĩa một cách rõ ràng việc nhận dạng các phiên bản của thành phần
  - Ba kỹ thuật cơ bản để nhận dạng phiên bản của thành phần
    - Đánh số phiên bản
    - Nhận dạng dựa vào thuộc tính
    - Nhận dạng hướng thay đổi

Quản lý phát hành và phiên bản

- Nhận dạng phiên bản - Đánh số phiên bản
  - Sơ đồ đánh số đơn giản nhất sử dụng sự tiến hóa tuyến tính
    - V1, V1.1, V1.2, V2.1, v.v
  - Cấu trúc tiến hóa thực tế là một cây hay một mạng hơn là một sự liên tục
  - Các tên không có ý nghĩa

Quản lý phát hành và phiên bản

- Nhận dạng phiên bản - Đánh số phiên bản



Quản lý phát hành và phiên bản

- Nhận dạng phiên bản - Nhận dạng dựa vào thuộc tính
  - Các thuộc tính có thể được sử dụng để nhận dạng phiên bản
    - Các thuộc tính có thể là ngày, người tạo ra, ngôn ngữ lập trình, khách hàng, trạng thái, v.v
  - Cách làm này có thể gây ra các vấn đề: tập các thuộc tính phải được chọn để tất cả các phiên bản có thể được định danh duy nhất
  - Trong thực tiễn, một phiên bản còn cần một tên kết hợp để tham khảo dễ dàng

Quản lý phát hành và phiên bản

- Nhận dạng phiên bản - Nhận dạng dựa vào thuộc tính
  - Một thuận lợi quan trọng của nhận dạng dựa vào thuộc tính là nó có thể hỗ trợ các truy vấn
  - Truy vấn chọn ra một phiên bản phụ thuộc vào các giá trị thuộc tính

Quản lý phát hành và phiên bản

- Nhận dạng phiên bản - Nhận dạng hướng thay đổi
  - Tích hợp các phiên bản và các thay đổi được thực hiện để tạo ra các phiên bản đó
  - Được sử dụng để nhận dạng phiên bản của hệ thống hơn là phiên bản của thành phần
  - Mỗi một thay đổi được đề nghị có một tập thay đổi kết hợp được tạo ra để thực hiện thay đổi đó

Quản lý phát hành và phiên bản

- Quản lý phát hành
  - Phát hành hệ thống: một phiên bản của hệ thống mà nó được phân phối tới khách hàng
  - Nhà cung cấp sản phẩm phần mềm thường chỉ đưa ra các phát hành mới cho các nền mới hay thêm các chức năng mới rất cần thiết
  - Các hệ thống hiện nay thường được phát hành trên đĩa quang hoặc các tập tin cài đặt có thể tải xuống từ trang web

Quản lý phát hành và phiên bản

- Quản lý phát hành

  Phát hành hệ thống
  - Không chỉ là một tập các chương trình có thể thực thi được
  - Mà có thể bao gồm
    - Các tập tin cấu hình định nghĩa cách thức phát hành được cấu hình cho một sự cài đặt cụ thể
    - Các tập tin dữ liệu cần cho sự vận hành hệ thống
    - Một chương trình cài đặt hay một script tiện ích để cài đặt hệ thống lên phần cứng đích
    - Các tài liệu ở dạng giấy hay dạng điện tử
    - Đóng gói và quảng cáo liên quan

Quản lý phát hành và phiên bản

- Quản lý phát hành

  Các vấn đề phát hành
  - Khách hàng có thể không muốn bản phát hành mới của hệ thống
  - Quản lý phát hành không nên giả sử rằng tất cả các phát hành trước đó được chấp nhận. Tất cả những tập tin cần cho một phát hành nên được tạo lại khi một phát hành mới được cài đặt

Quản lý phát hành và phiên bản

- Ra quyết định phát hành
  - Việc chuẩn bị và phân phối một bản phát hành hệ thống là một quy trình tốn kém
  - Các yếu tố như chất lượng kỹ thuật, sự cạnh tranh, v.v tác động đến việc quyết định khi nào đưa ra một phát hành của hệ thống mới

Factor	Description
Technical quality of the system	If serious system faults are reported which affect the way in which many customers use the system, it may be necessary to issue a fault repair release. However, minor system faults may be repaired by issuing patches (often distributed over the Internet) that can be applied to the current release of the system.
Platform changes	You may have to create a new release of a software application when a new version of the operating system platform is released.
Lehman's fifth law (see Chapter 21)	This suggests that the increment of functionality that is included in each release is approximately constant. Therefore, a system release with significant new functionality may have to be followed by a repair release.
Competition	A new system release may be necessary because a competing product is available.
Marketing requirements	The marketing department of an organisation may have made a commitment for releases to be available at a particular date.
Customer change proposals	For customised systems, customers may have made and paid for a specific set of system change proposals, and they expect a system release as soon as these have been implemented.

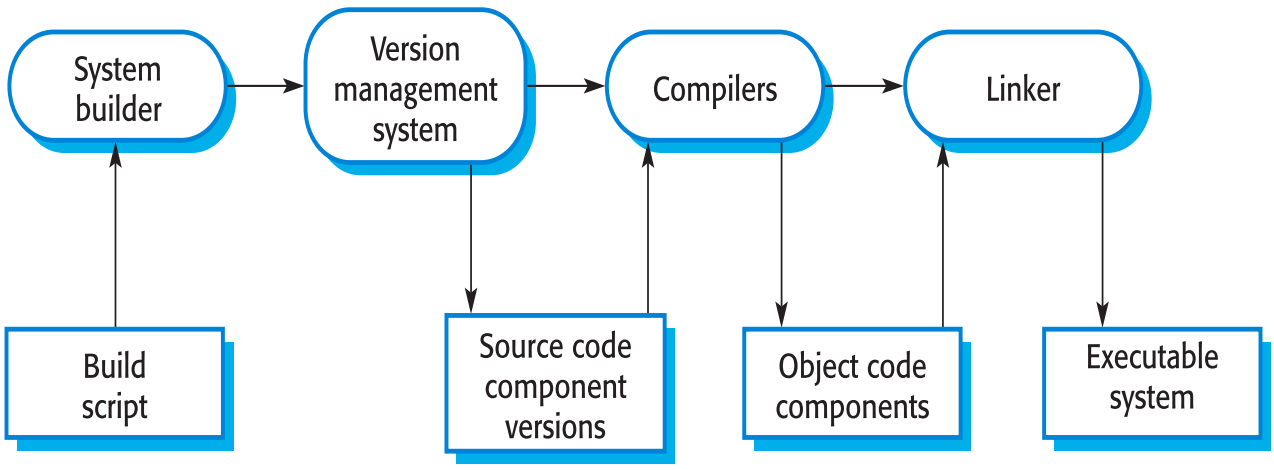
Quản lý phát hành và phiên bản

- Tư liệu phát hành
  - Ghi lại các phiên bản cụ thể của các thành phần mã nguồn được sử dụng để tạo ra mã thực thi
  - Lưu bản sao của mã nguồn, mã thực thi, tất cả dữ liệu và các tập tin cấu hình
  - Ghi lại phiên bản của hệ điều hành, thư viện, bộ biên dịch và những công cụ được sử dụng để xây dựng phần mềm

Xây dựng hệ thống

- Xây dựng hệ thống là quy trình biên dịch và liên kết các thành phần của phần mềm vào một chương trình mà nó thực hiện trên một cấu hình đích cụ thể
- Các hệ thống khác nhau được xây dựng từ các kết hợp khác nhau về các thành phần của phần mềm
- Qui trình này hiện nay luôn được hỗ trợ bởi các công cụ tự động

Xây dựng hệ thống



Nội dung – Quản lý dự án

- Các đặc trưng của dự án
- Quản lý rủi ro
- Các kỹ thuật kiểm soát và lập kế hoạch dự án

Các đặc trưng của dự án

- Các lớp đặc trưng của dự án:
  - Đặc trưng của sản phẩm
  - Đặc trưng của qui trình
  - Đặc trưng của nguồn lực
- Các đặc trưng có mức độ chắn chắn xác định

Các đặc trưng của dự án

- Độ chặc chắn của sản phẩm:
  - Các yêu cầu rõ ràng, được biết trước: độ chắc chắn của sản phẩm cao
  - Các yêu cầu của người dùng thay đổi thường xuyên: độ chắc chắn của sản phẩm thấp
- Độ chặc chắn của quy trình:
  - Biết nhiều về sự ảnh hưởng của các hoạt động điều khiển: cao
  - Sử dụng các công cụ không biết: thấp
- Độ chặc chắn của nguồn lực:
  - Phụ thuộc vào sự sẵn có của nhân viên có phẩm chất phù hợp, tiềm lực tài chính

Các đặc trưng của dự án

- Các trạng thái kiểm soát điển hình
  - Realization: tất cả các độ chắc chắn đều cao
  - Allocation: độ chắn chắn của tài nguyên thấp còn lại đều cao
  - Design: độ chắc chắn của sản phẩm cao còn lại đều thấp
  - Exploration: tất cả các độ chắc chắn đều thấp

	Realization	Allocation	Design	Exploration
Product certainty	high	high	high	low
Process certainty	high	high	low	low
Resource certainty	high	low	low	low

Các đặc trưng của dự án

- Trạng thái kiểm soát Realization
  - Mục đích cơ bản trong kiểm soát:
    - Tối ưu hóa việc sử dụng tài nguyên, hiệu suất và kế hoạch
  - Kiểu quản lý / phối hợp:
    - Kiểu tách biệt, phân cấp, chuẩn hóa

Các đặc trưng của dự án

- Trạng thái kiểm soát Allocation
  - Mục đích cơ bản trong kiểm soát:
    - Thu nhận và đào tạo nhân sự
  - Kiểu quản lý / phối hợp:
    - Sự chuẩn hóa sản phẩm và quy trình

Các đặc trưng của dự án

- Trạng thái kiểm soát Design
  - Mục đích cơ bản trong kiểm soát:
    - Kiểm soát quy trình
  - Kiểu quản lý / phối hợp:
    - Sự chuẩn hóa quy trình

Các đặc trưng của dự án

- Trạng thái kiểm soát Exploration
  - Mục đích cơ bản trong kiểm soát:
    - Cực đại kết quả và giảm thiểu rủi ro
  - Kiểu quản lý / phối hợp:
    - Kiểu quan hệ, giao phó và điều chỉnh lẫn nhau

Quản lý rủi ro

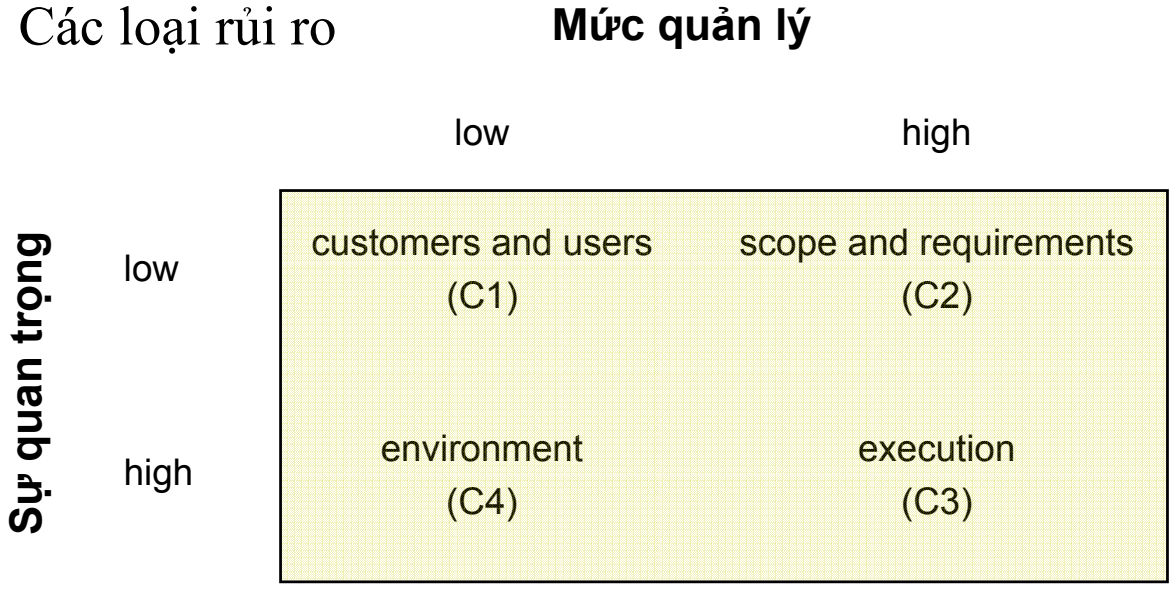
- Các yếu tố rủi ro hàng đầu
  - Nhân sự thiếu
  - Lịch biểu / ngân sách phi hiện thực
  - Chức năng phần mềm sai
  - Giao diện người dùng sai
  - Phát triển những chức năng không được khách hàng yêu cầu
  - Các yêu cầu không ổn định
  - Các thành phần được cung cấp từ bên ngoài không đạt yêu cầu
  - Công việc đối ngoại dở
  - Yêu cầu thời gian thực không được đáp ứng
  - Môi trường không ổn định hay công nghệ mới, chưa được thử nghiệm

Quản lý rủi ro

- Chiến lược quản lý rủi ro
  - Nhận dạng các yếu tố rủi ro
  - Xác định mức độ rủi ro
  - Phát triển các chiến lược giảm nhẹ các rủi ro
  - Quản lý các rủi ro

Quản lý rủi ro

- Các loại rủi ro



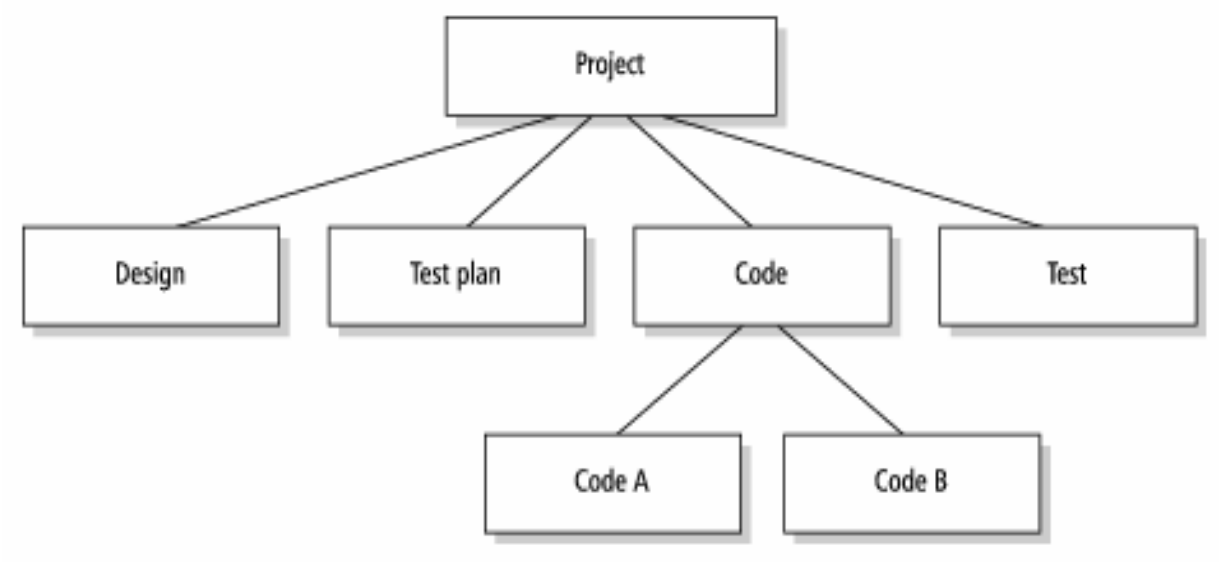
Thứ tự quản lý: Đầu tiên C3, sau đó C2, sau đó C4 và C1

Các kỹ thuật kiểm soát và lập kế hoạch dự án

- Cấu trúc phân chia công việc (Work breakdown structure - WBS)
- Sơ đồ Pert
- Sơ đồ Gantt
- …

Các kỹ thuật kiểm soát và lập kế hoạch

- Cấu trúc phân chia công việc (Work breakdown structure - WBS)



Lập kế hoạch quản lý

- Nguyên tắc chung
  - Chia nhỏ dự án thành các công việc kiểm soát được
  - Mỗi công việc có một mốc thời gian và nguồn lực có thể kiểm soát được tiến độ.
  - Các công việc thường được thực hiện theo một trật tự nào đó
- Ta có thể lập bảng công việc & các biểu đồ như PERT, GANTT

Sơ đồ PERT theo công việc

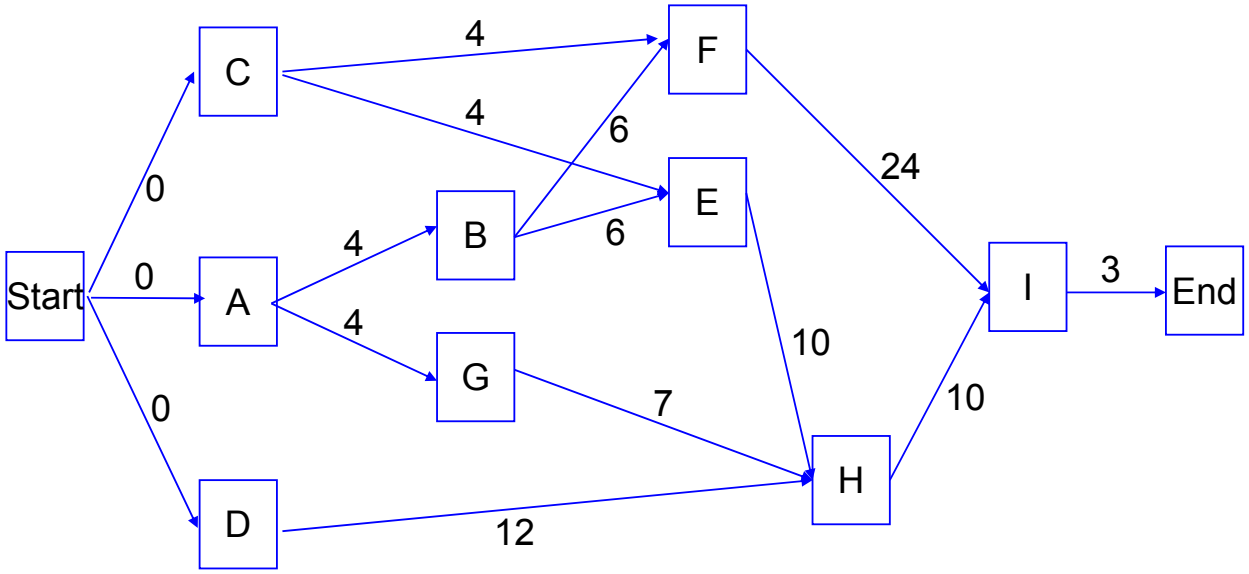
- Pert sử dụng hai yếu tố cơ bản là công việc và thời gian thực hiện công việc.
  - Công việc được biểu thị bằng một đỉnh
  - Thời gian thực hiện công việc được biểu thị bằng một cung.
- Để vẽ sơ đồ PERT theo công việc ta phải sử dụng 2 nút giả là bắt đầu (Start) và kết thúc (End).

Sơ đồ PERT theo công việc

- Ví dụ: Giả sử sau khi phân chia và ước lượng công việc ta có bảng sau

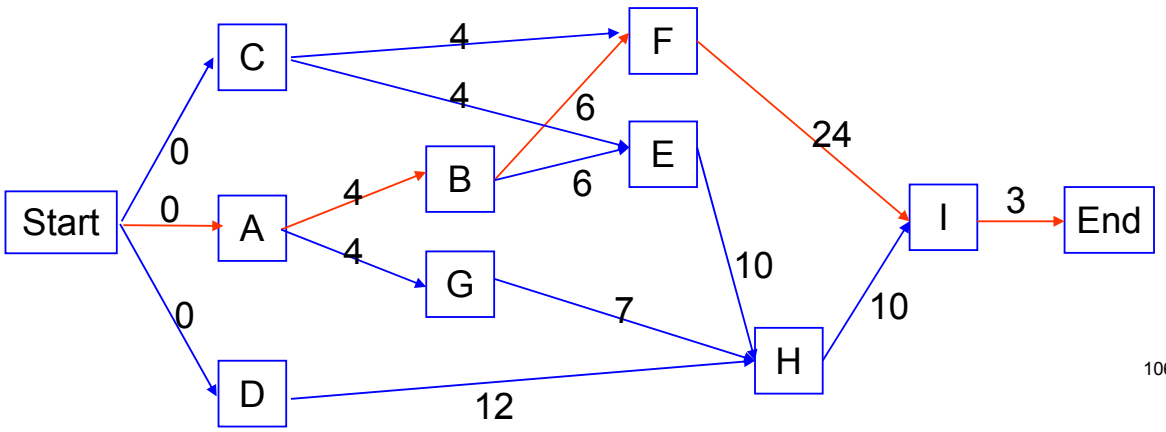
Công việc	Công việc trước đó	Thời gian (tháng)	Chi phí (triệu đồng)
A	-	4	5
B	A	6	11
C	-	4	3
D	-	12	150
E	B, C	10	10
F	B, C	24	147
G	A	7	18
H	D, E, G	10	4
I	F, H	3	2

Sơ đồ PERT theo công việc



Sơ đồ PERT theo công việc

- Đường găng
  - Đường dài nhất (theo thời gian) trong sơ đồ Pert đi từ Start tới End.
  - Thời gian thực hiện dự án được tính bằng cách cộng dồn thời gian theo đường này



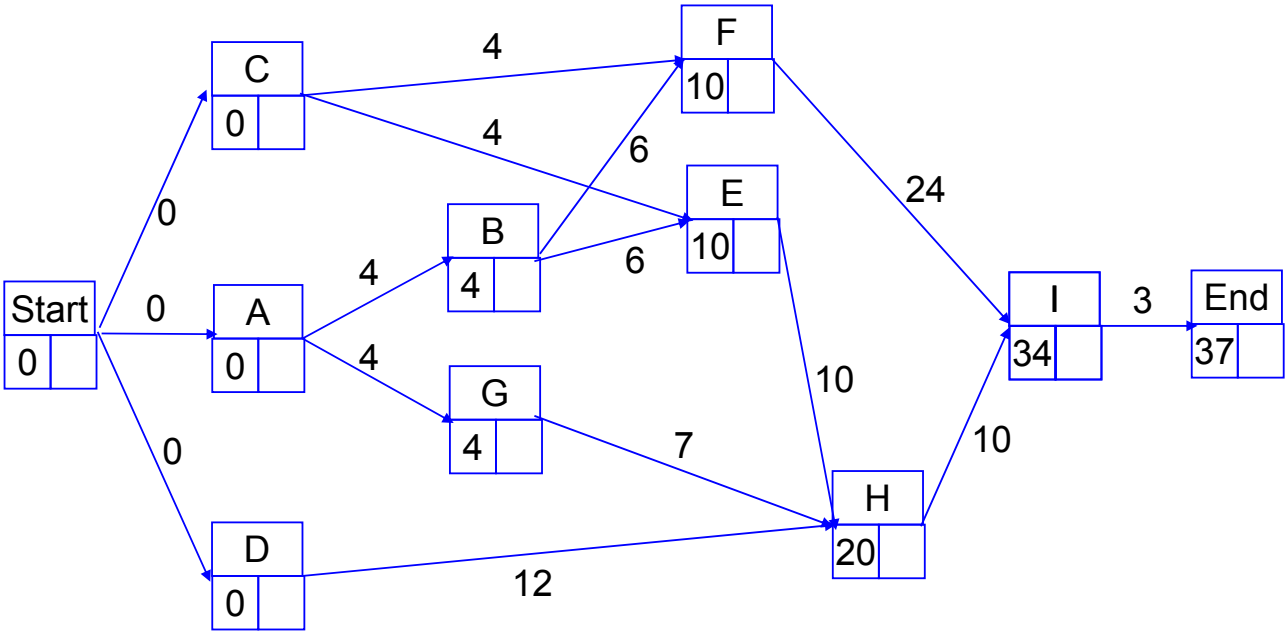
Sơ đồ PERT theo công việc

- Công việc găng
  - Công việc nằm trên đường găng
  - Công việc mà thực hiện chúng chậm đi bao nhiêu thì toàn bộ dự án sẽ bị đẩy lùi đi thời gian đúng bằng bấy nhiêu

Sơ đồ PERT theo công việc

- Thời gian sớm nhất để bắt đầu thực hiện công việc i được ký hiệu là t_i
  - $t_i = \max_{j \in P(i)} \{t_j + t_{ji}\}$ trong đó
    - P(i) là tập hợp tất cả các đỉnh j đứng trước i
    - t_{ji} là giá trị hay độ dài của cung (j, i)

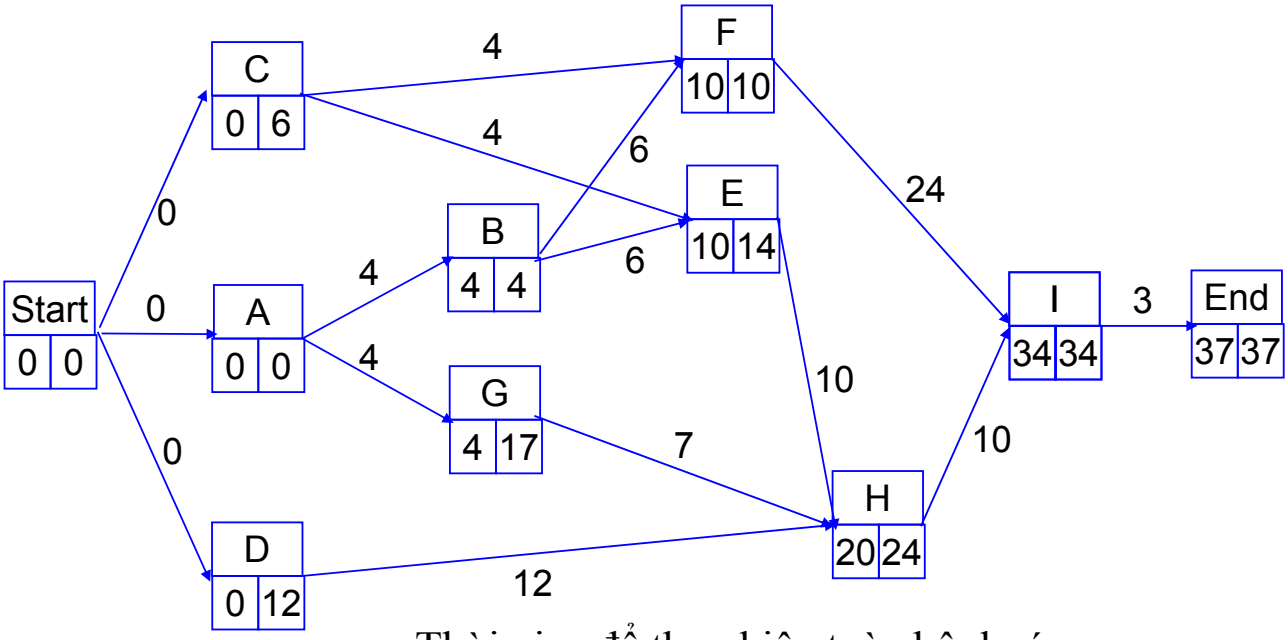
Sơ đồ PERT theo công việc



Sơ đồ PERT theo công việc

- Thời gian trễ nhất để bắt đầu thực hiện công việc i được ký hiệu là T_i
  - $T_i = \min_{j \in S(i)} \{T_j - t_{ij}\}$ trong đó
    - S(i) là tập hợp tất cả các đỉnh j đứng sau i
    - t_{ij} là giá trị hay độ dài của cung (i,j)

Sơ đồ PERT theo công việc



Thời gian để thực hiện toàn bộ dự án là 37 tháng và kinh phí là 350 triệu

Sơ đồ PERT theo công việc

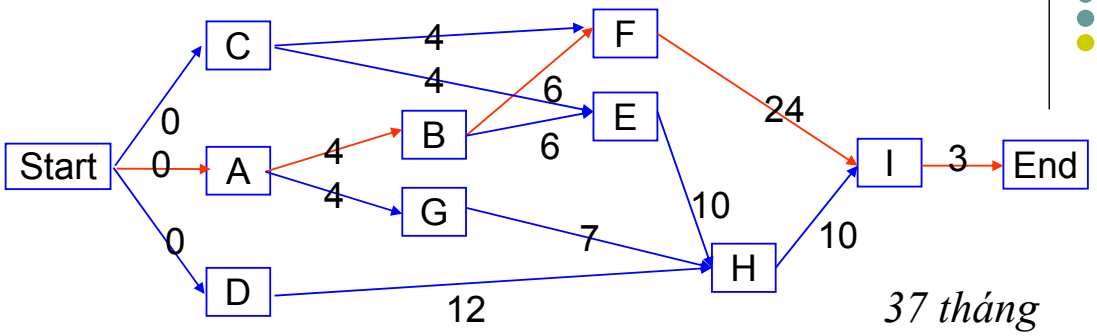
Công việc	Công việc trước đó	Thời gian (tháng)	Chi phí (triệu đồng)	Thời gian thực hiện khẩn trương có thể	Chi phí bỏ thêm khi rút ngắn 1 tháng
A	-	4	5	2	5
B	A	6	11	5	19
C	-	4	3	2	4
D	-	12	150	9	10
E	B, C	10	10	8	5
F	B, C	24	147	19	13
G	A	7	18	6	12
H	D, E, G	10	4	7	7
I	F, H	3	2	2	3

Hãy rút ngắn thời gian thực hiện dự án xuống còn 28 tháng?

Sơ đồ PERT theo công việc

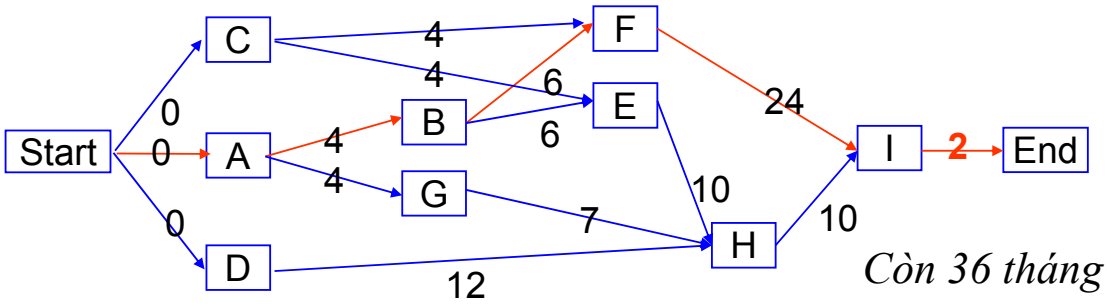
- Rút ngắn thời gian thực hiện dự án
  - Lặp lại việc *chọn công việc găng với chi phí cần bổ sung để đẩy nhanh thêm một đơn vị thời gian là rẻ nhất* và giảm thời gian thực hiện công việc này tới tối đa cho đến khi:
    - Đạt được thời gian tối thiểu cần thiết để thực hiện công việc hay
    - Xuất hiện công việc găng mới
  - Nếu công việc găng cần rút ngắn nằm trên chu trình gồm nhiều công việc găng khác thì *rút ngắn tối đa hai công việc găng nằm trên hai nhánh khác nhau của chu trình sao cho tổng chi phí bỏ thêm của chúng là ít nhất* (so với các công việc găng còn lại và các cặp công việc găng trên các nhánh của chu trình)

Sơ đồ PERT theo công việc



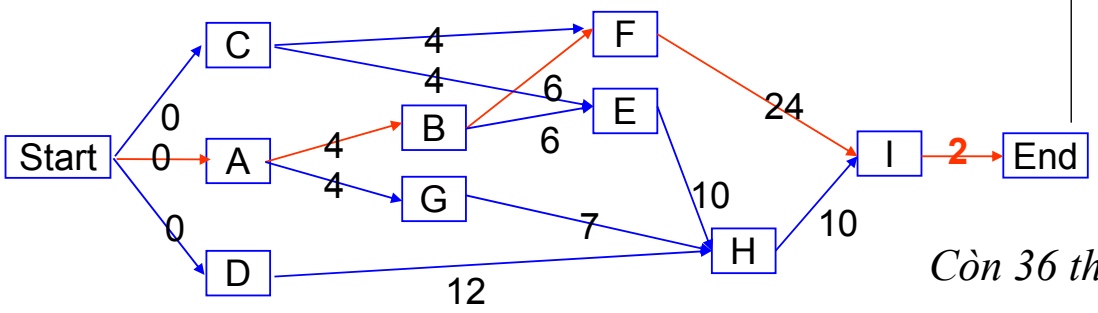
37 tháng

- Chọn các công việc găng A, B, F, I để rút ngắn
  - Chọn I đầu tiên vì chi phí bỏ thêm cho I là thấp nhất và rút ngắn I một tháng.



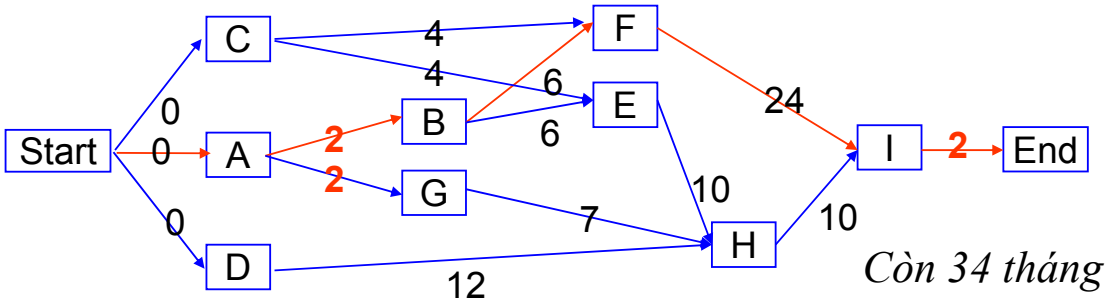
Còn 36 tháng

Sơ đồ PERT theo công việc



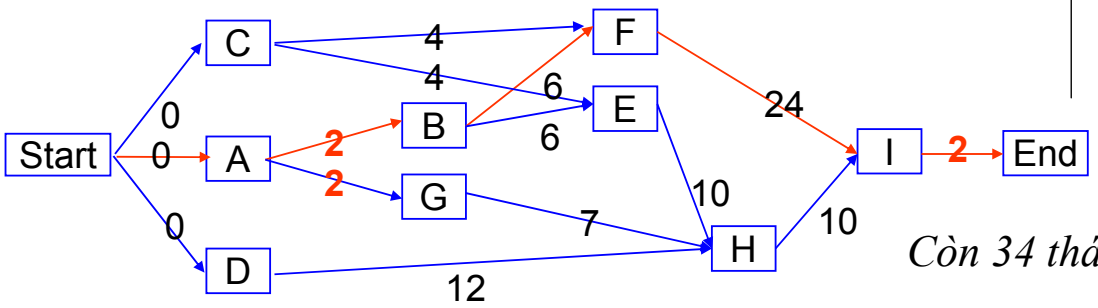
Còn 36 tháng

- Chọn các công việc găng A, B, F để rút ngắn
  - Chọn A tiếp theo vì chi phí bỏ thêm cho A là thấp thứ hai và rút ngắn A hai tháng.



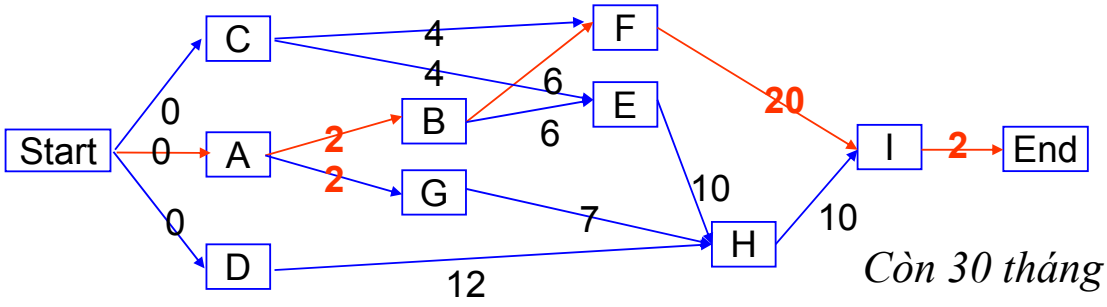
Còn 34 tháng

Sơ đồ PERT theo công việc



Còn 34 tháng

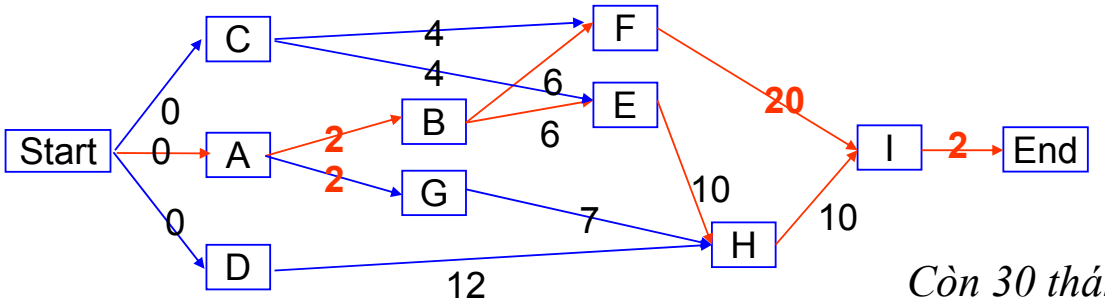
- Chọn các công việc găng B, F để rút ngắn
  - Chọn F tiếp theo vì chi phí bỏ thêm cho F là thấp thứ ba và rút ngắn F bốn tháng.



Còn 30 tháng

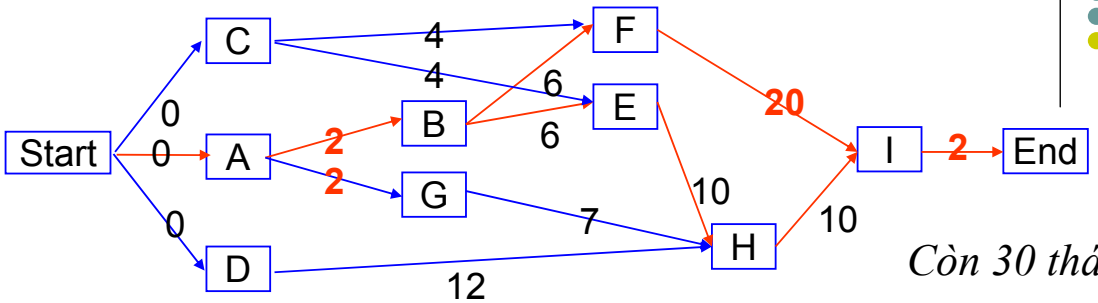
Sơ đồ PERT theo công việc

- Khi rút ngắn F bốn tháng, ta có chu trình các công việc găng



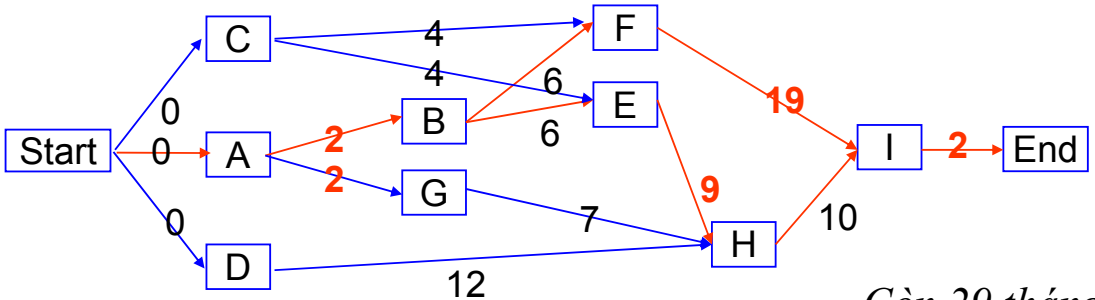
Còn 30 tháng

Sơ đồ PERT theo công việc



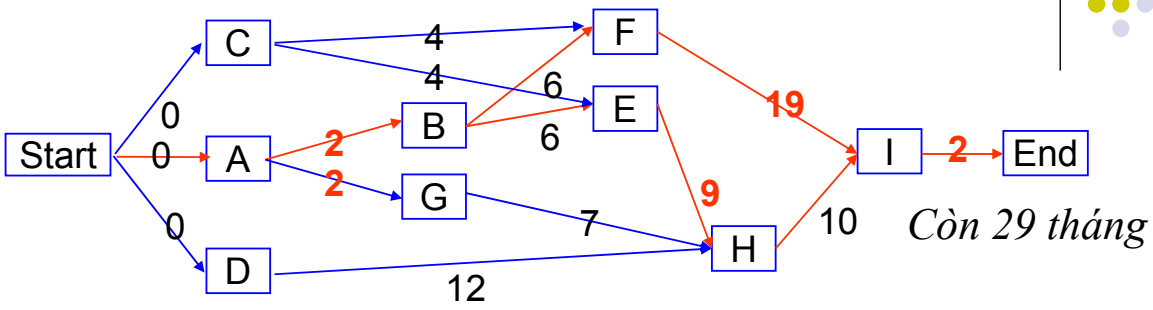
Còn 30 tháng

- Trong số các công việc găng còn lại và các cặp công việc găng trên các nhánh thì cặp F+E có chi phí thấp nhất nên ta rút F+E 1 tháng.



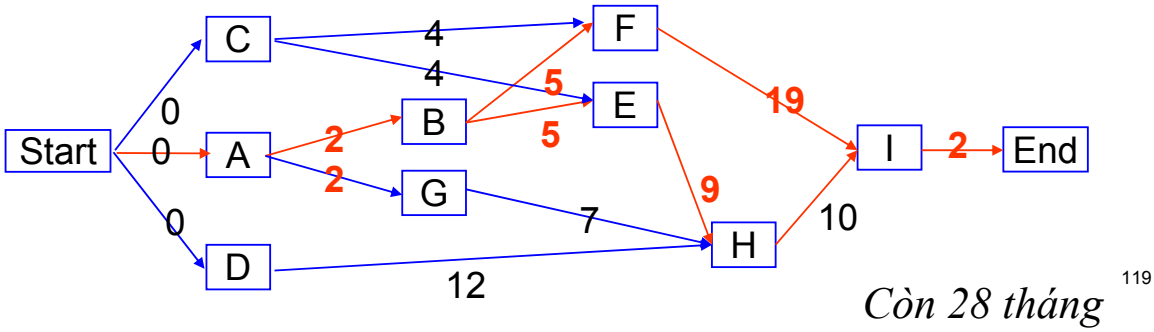
Còn 29 tháng

Sơ đồ PERT theo công việc



Còn 29 tháng

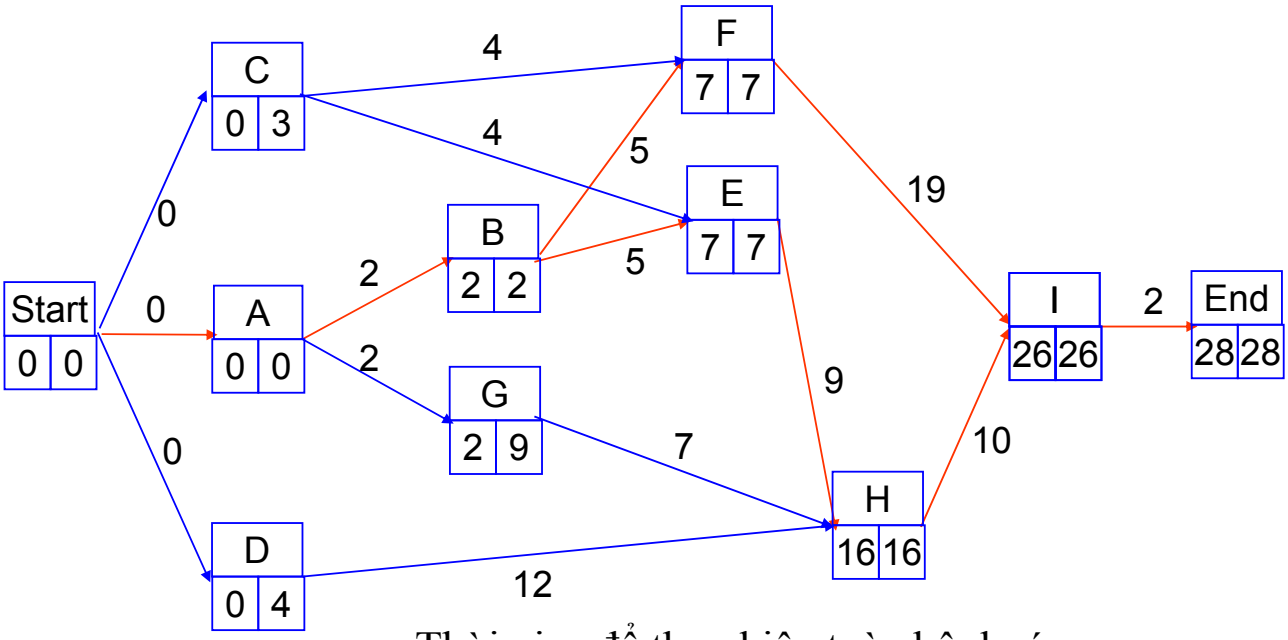
- Cuối cùng, ta rút ngắn B một tháng.



Còn 28 tháng

Sơ đồ PERT theo công việc

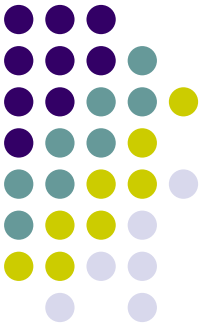
- Sơ đồ Pert cho kế hoạch khẩn trương



Thời gian để thực hiện toàn bộ dự án là 28 tháng và kinh phí là 452 triệu

NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

CHƯƠNG 4 – ƯỚC LƯỢNG CHI PHÍ PHẦN MỀM



Nội dung

- Giới thiệu
- Ước lượng kích thước phần mềm
- Ước lượng chi phí phần mềm

Giới thiệu

- Các yếu tố cần ước lượng
  - Kích thước phần mềm
  - Công sức phát triển
  - Thời gian thực hiện
- Nguyên tắc ước lượng
  - Phân rã dự án theo các chức năng chính và ước lượng từng chức năng
  - Dựa trên kinh nghiệm, dữ kiện quá khứ

Ước lượng kích thước phần mềm

- Ước lượng kích thước phần mềm
  - Qua dòng lệnh: Ước lượng trực tiếp với từng module
  - Qua điểm chức năng: Ước lượng gián tiếp thông qua ước lượng input/output, yêu cầu,…

Ước lượng kích thước phần mềm

- Qua dòng lệnh
  - Theo đơn vị một dòng lệnh **LOC** (Lines Of Code)
  - Theo đơn vị một ngàn dòng lệnh **KDSI / KLOC** (Thousand Delivered Source of Code / Kilo Lines of Code)
  - Phụ thuộc ngôn ngữ lập trình

Ước lượng kích thước phần mềm

- *Các vấn đề gặp phải với các phương pháp LOC và KDSI*
  - Tính toán kích thước tại các giai đoạn khác nhau: phân tích yêu cầu, …
  - Cài đặt trên các ngôn ngữ lập trình khác nhau: C, Java, Lisp,…
  - Cách tính số dòng mã lệnh: mã lệnh thực thi, định nghĩa dữ liệu,…
  - Sinh mã tự động, thiết kế giao diện trực tiếp (GUI)
  - *Giá thành của sản phẩm phụ thuộc vào ước lượng LOC*

Ước lượng kích thước phần mềm

- Qua điểm chức năng (**FP** - Functional Points)
  - Độc lập với ngôn ngữ lập trình
  - Các điểm chức năng:
    - Input Item (Inp): Số input của người dùng mà chúng thực hiện các thay đổi trong cấu trúc dữ liệu (xử lý), không điều khiển sự thực hiện chương trình
    - Output Item (Oup): Số output được kết xuất cho người dùng
    - Inquiry (Inq): Số input (truy vấn) điều khiển sự thực hiện chương trình, không thay đổi trong cấu trúc dữ liệu
    - Master File (Maf) : Số tập tin được sinh ra và được sử dụng, được duy trì bởi hệ thống
    - Interface (Inf): Số output đi vào ứng dụng khác hay được chia sẻ với ứng dụng nào đó

Ước lượng kích thước phần mềm

- Bảng giá trị các điểm chức năng theo độ phức tạp từ thấp, trung bình đến cao

	Level of Complexity		
Component	Simple	Average	Complex
Input item *(Inp)*	3	4	6
Output item *(Out)*	4	5	7
Inquiry *(Inq)*	3	4	6
Master file *(Maf)*	7	10	15
Interface *(inf)*	5	7	10

- Công thức tính số điểm chức năng thô

$$UFP = a_1 \times Inp + a_2 \times Oup + a_3 \times Inq + a_4 \times Maf + a_5 \times Inf$$

với a_1, a_2, a_3, a_4, a_5 là giá trị các điểm chức năng theo độ phức tạp cho trong bảng trên.

Ước lượng kích thước phần mềm

Table 2. FP Counting Weights

For Internal Logical Files and External Interface Files

	Data Elements		
Record Elements	**1 - 19**	**20 - 50**	**51+**
1	Low	Low	Avg.
2 - 5	Low	Avg.	High
6+	Avg.	High	High

For External Output and External Inquiry

	Data Elements		
File Types	**1 - 5**	**6 - 19**	**20+**
0 or 1	Low	Low	Avg.
2 - 3	Low	Avg.	High
4+	Avg.	High	High

For External Input

	Data Elements		
File Types	**1 - 4**	**5 - 15**	**16+**
0 or 1	Low	Low	Avg.
2 - 3	Low	Avg.	High
3+	Avg.	High	High

Ước lượng kích thước phần mềm

- Công thức tính điểm chức năng FP

  FP = Điểm chức năng thô x (0.65 + 0.01 x
  Tổng các mức độ ảnh hưởng của các hệ số kỹ thuật)

- 14 hệ số kỹ thuật (có mức độ ảnh hưởng nằm trong phạm vi từ 0 (không quan trọng hay không thích hợp hay không ảnh hưởng) tới 5 (cực kỳ quan trọng hay cần thiết tuyệt đối hay ảnh hưởng nhất)

Ước lượng kích thước phần mềm

- Các hệ số kỹ thuật
  1. Data communication
  2. Distributed function
  3. Performance
  4. Heavily used configuration
  5. Transaction rate
  6. Online data entry
  7. End-user efficiency
  8. Online update
  9. Complex processing
  10. Reusability
  11. Installation ease
  12. Operation ease
  13. Multiple site
  14. Facilitate change

Ước lượng kích thước phần mềm

- Điểm chức năng FP có thể được dùng để dự đoán số dòng lệnh LOC

  LOC = AVC * số điểm chức năng FP

  với AVC : yếu tố phụ thuộc vào ngôn ngữ lập trình được sử dụng

Ước lượng kích thước phần mềm

	Programming Statements per Function Point		
Language	Minimum (Minus 1 Standard Deviation)	Mode (Most Common Value)	Maximum (Plus 1 Standard Deviation)
Ada 83	45	80	125
Ada 95	30	50	70
C	60	128	170
C#	40	55	80
C++	40	55	140
Cobol	65	107	150
Fortran 90	45	80	125
Fortran 95	30	71	100
Java	40	55	80
Macro Assembly	130	213	300
Perl	10	20	30
Second generation default (Fortran 77, Cobol, Pascal, etc.)	65	107	160
Smalltalk	10	20	40
SQL	7	13	15
Third generation default (Fortran 90, Ada 83, etc.)	45	80	125
Microsoft Visual Basic	15	32	41

Source: Adapted from *Estimating Software Costs* (Jones 1998), *Software Cost Estimation with Cocomo II* (Boehm 2000), and *Estimating Software Intensive Systems* (Stutzke 2005).

Ước lượng kích thước phần mềm

- Bài tập: Một sản phẩm phần mềm được ước lượng có 4 đầu vào đơn giản, 9 đầu vào phức tạp, 17 đầu ra trung bình, 20 truy vấn phức tạp, 13 tập tin chính đơn giản, 15 giao diện trung bình và 7 giao diện phức tạp.
  1. Hãy tính số lượng điểm chức năng thô?
  2. Nếu hầu hết các nhân tố kỹ thuật đều không quan trọng trừ hai nhân tố dễ cài đặt và dễ sử dụng là rất quan trọng thì số lượng điểm chức năng là bao nhiêu?
  3. Hãy qui đổi ra số lượng dòng mã lệnh theo ngôn ngữ lập trình C?

Ước lượng chi phí phần mềm

- Ước lượng chi phí
  - Dựa trên kích thước, độ phức tạp
  - Dựa vào dữ liệu quá khứ
  - Chi phí tỉ lệ với công sức (effort) phát triển phần mềm
  - Chi phí tính dựa theo công sức cho các giai đoạn phát triểm phần mềm: khởi đầu, phân tích, thiết kế, cài đặt, kiểm thử nhưng chưa tính đến giai đoạn bảo trì.
  - Đơn vị tính của công sức: người-tháng (hoặc người-ngày)

Ước lượng chi phí phần mềm

- Các phương pháp ước lượng chi phí phần mềm
  - Theo ý kiến của chuyên gia
  - Theo giải thuật
  - Bằng sự tương tự
  - Bằng luật Parkinson
  - Đấu giá (Pricing to win)

Tự đọc

Ước lượng chi phí phần mềm

- Ý kiến của chuyên gia (Expert judgment)
  - Một hay nhiều chuyên gia trong cả lĩnh vực ứng dụng và phát triển phần mềm sử dụng kinh nghiệm của họ để dự tính chi phí phần mềm. Qui trình này được lặp đi lặp lại cho đến khi đạt được sự nhất trí.
  - Thuận lợi: Phương pháp dự đoán chi phí thấp một cách tương đối. Có thể chính xác nếu các chuyên gia có kinh nghiệm trực tiếp trong các hệ thống tương tự.
  - Khó khăn: Rất thiếu chính xác nếu không có các chuyên gia thực sự!

Ước lượng chi phí phần mềm

- Ước lượng chi phí theo giải thuật
  - Một số mô hình ước lượng chi phí theo giải thuật
    - Mô hình Walston và Felix
    - Mô hình Bailey và Basili
    - Mô hình COCOMO

Mô hình Walston và Felix

- Một trong các mô hình giải thuật sớm nhất (1977)
- Mô hình được đề nghị sau khi nghiên cứu 60 dự án của IBM
- Có 29 yếu tố ảnh hưởng tới hiệu suất
- Công thức ước lượng *công sức E*

  $E = 5.2 \times S^{0.91}$ (người - tháng)

  Với S là kích thước được ước lượng của hệ thống (*theo KDSI*)
- Công thức ước lượng *thời gian thực hiện T*

  $T = 2.5 \times E^{0.35}$ (tháng)

Mô hình Walston và Felix

- Mỗi yếu tố sẽ nhận 1 trong 3 giá trị tùy thuộc vào sự tác động của nó tới hiệu suất
  - 1 (cao): làm tăng hiệu suất
  - 0 (trung bình): không ảnh hưởng tới hiệu suất
  - -1 (thấp): làm giảm hiệu suất

Mô hình Walston và Felix

1. Customer interface complexity	16. Use of design and code inspections
2. User participation in requirements definition	17. Use of top-down development
3. Customer-originated program design changes	18. Use of a chief programmer team
4. Customer experience with the application area	19. Overall complexity of code
5. Overall personnel experience	20. Complexity of application processing
6. Percentage of development programmers who participated in the design of functional specifications	21. Complexity of program flow
7. Previous experience with the operational computer	22. Overall constraints on program's design
8. Previous experience with the programming language	23. Design constraints on the program's main storage
9. Previous experience with applications of similar size and complexity	24. Design constraints on the program's timing
10. Ratio of average staff size to project duration (people per month)	25. Code for real-time or interactive operation or for execution under severe time constraints
11. Hardware under concurrent development	26. Percentage of code for delivery
12. Access to development computer open under special request	27. Code classified as nonmathematical application and input/output formatting programs
13. Access to development computer closed	28. Number of classes of items in the database per 1000 lines of code
14. Classified security environment for computer and at least 25% of programs and data	29. Number of pages of delivered documentation per 1000 lines of code
15. Use of structured programming	

Mô hình Bailey và Basili

- Mô hình được đề nghị năm 1981 bởi Bailey và Basili
- Mô hình này sử dụng cơ sở dữ liệu của 18 dự án viết bằng ngôn ngữ Fortran tại trung tâm Goddard Space Flight của NASA
- Các nhóm yếu tố ảnh hưởng tới công sức: phương pháp, độ phức tạp và kinh nghiệm
- Công thức ước lượng công sức ban đầu E

$$E = 5.5 + 0.73 \times S^{1.16} \text{ (người - tháng)}$$

Với S là kích thước được ước lượng của hệ thống (*theo KDSI*)

Mô hình Bailey và Basili

- Mỗi yếu tố ảnh hưởng tới công sức nhận một trong các giá trị từ 0 đến 5

Total methodology (METH)	*Cumulative complexity (CPLX)*	*Cumulative experience (EXP)*
Tree charts	Customer interface complexity	Programmer qualifications
Top-down design	Application complexity	Programmer machine experience
Formal documentation	Program flow complexity	Programmer language experience
Chief programmer teams	Internal communication complexity	Programmer application experience
Formal training	Database complexity	Team experience
Formal test plans	External communication complexity	
Design formalisms	Customer-initiated program design changes	
Code reading		
Unit development folders		

Mô hình COCOMO 81

- Mô hình COCOMO 81 được đề nghị bởi Boehm
  - Dạng cơ bản: áp dụng cho nhóm nhỏ, môi trường quen thuộc
  - Dạng trung bình: áp dụng cho dự án khá lớn, có một ít kinh nghiệm
  - Dạng lớn: áp dụng cho dự án lớn, môi trường mới
- Bảng mức độ khó khi phát triển sản phẩm (sử dụng mô hình COCOMO 81 trung gian)

Software project	a_b	b_b	c_b	d_b
Dạng cơ bản (organic)	3.2	1.05	2.50	0.38
Dạng trung bình (semi-detached)	3.0	1.12	2.50	0.35
Dạng lớn (embedded)	2.8	1.20	2.50	0.32

Mô hình COCOMO 81 trung gian

- Công sức $E = a_b \times S^{b_b} \times EAF$
  - a_b và b_b: được xác định dựa vào bảng mức độ khó khi phát triển phần mềm
  - EAF (effort adjustment factor): hệ số hiệu chỉnh công sức. Nó được tính bằng tích của các hệ số phát triển
  - S là kích thước được ước lượng của hệ thống (*theo KDSI*)
- Thời gian $T = c_b \times E^{d_b}$

Mô hình COCOMO 81 trung gian

- Các hệ số phát triển

Cost Drivers	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
Product Attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Database size		0.94	1.00	1.08	1.16	
Product complexity	0.70	0.85	1.00	1.15	1.30	1.65
Computer Attributes						
Execution time constraint			1.00	1.11	1.30	1.66
Main storage constraint			1.00	1.06	1.21	1.56
Virtual machine volatility*		0.87	1.00	1.15	1.30	
Computer turnaround time		0.87	1.00	1.07	1.15	
Personnel Attributes						
Analyst capabilities	1.46	1.19	1.00	0.86	0.71	
Applications experiences	1.29	1.13	1.00	0.91	0.82	
Programmer capabilitiy	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience*	1.21	1.10	1.00	0.90		
Programming language experiences	1.14	1.07	1.00	0.95		
Project Attributes						
Use of modern programming pratices	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

Mô hình COCOMO 81 trung gian

- Bài tập: Bạn được giao trách nhiệm phát triển một phần mềm dạng lớn có 128000LOC. Các thông số nhìn chung đều ảnh hưởng ở mức bình thường nhưng yêu cầu về độ tin cậy của phần mềm là rất cao, sản phẩm có tính chất phức tạp và ràng buộc về vấn đề lưu trữ chính là vô cùng cao cũng như yêu cầu rất chặt về lịch biểu phát triển. Ngoài ra, đội ngũ lập trình viên được đánh giá có kinh nghiệm và có khả năng tốt. Sử dụng mô hình COCOMO 81 trung gian để tính E vàT?

Mô hình COCOMO 2

- Mô hình COCOMO 81 được phát triển trên giả thiết rằng tiến trình phát triển phần mềm thác nước được sử dụng và tất cả các phần mềm được phát triển từ đầu.
- Mô hình COCOMO 2 được thiết kế để thích ứng với những cách tiếp cận khác nhau đối với sự phát triển phần mềm

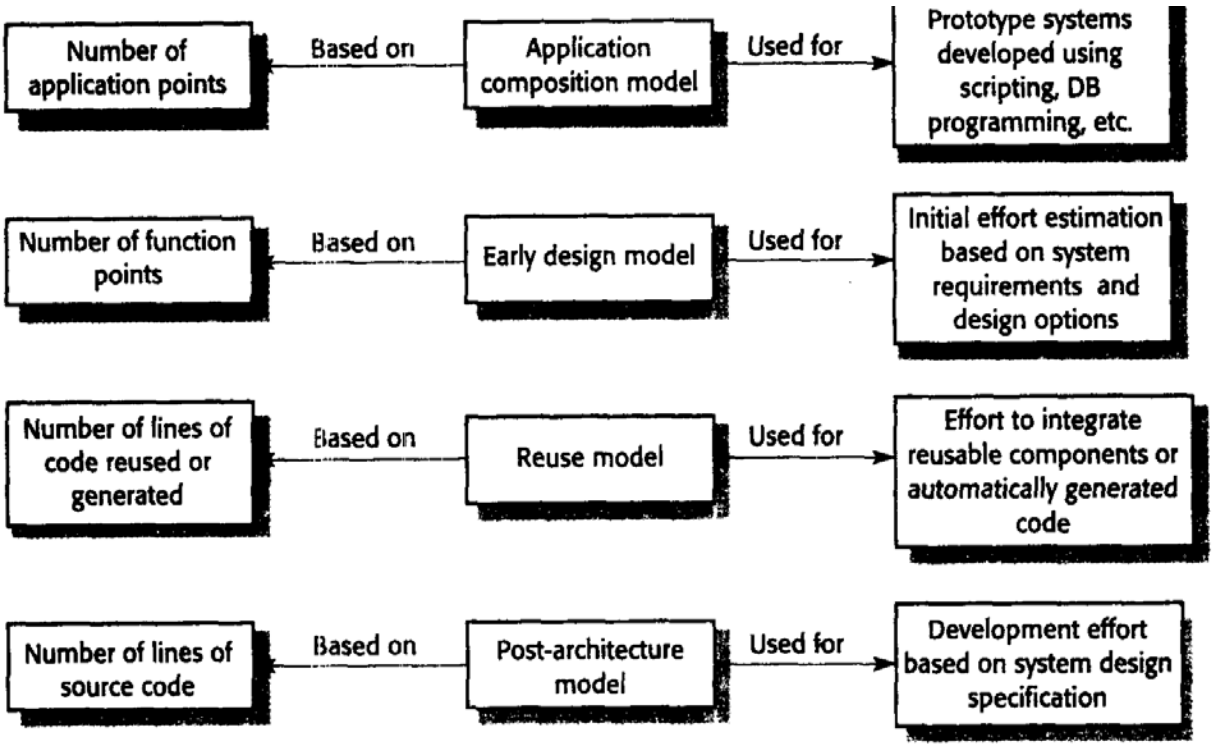
Mô hình COCOMO 2

- COCOMO 2 kết hợp chặt chẽ các mô hình con nhằm đưa ra các dự đoán phần mềm chi tiết
- Các mô hình con trong COCOMO 2:
  - Mô hình Application composition. Mô hình này giả sử rằng hệ thống được tạo thành từ các thành phần có thể tái sử dụng. Mô hình này được thiết kế để ước lượng công sức phát triển bản mẫu.
  - Mô hình Early design: được sử dụng tại giai đoạn thiết kế kiến trúc khi các yêu cầu là sẵn (và thiết kế chi tiết vẫn chưa được bắt đầu)

Mô hình COCOMO 2

- Các mô hình con trong COCOMO 2:
  - Mô hình Reuse: được sử dụng để tính công sức tích hợp các thành phần có thể dùng lại được và / hoặc mã chương trình mà nó được tự động sinh ra bởi các công cụ dịch chương trình hay thiết kế. Nó thường được sử dụng kết hợp với mô hình Post - architecture.
  - Mô hình Post-architecture: được sử dụng ngay khi kiến trúc hệ thống đã được thiết kế và các thông tin chi tiết hơn về hệ thống là sẵn có.

Mô hình COCOMO 2



Mô hình Application composition

- Để ước lượng công sức cần để lập bản mẫu các dự án và cho các dự án được phát triển bằng cách kết hợp các thành phần sẵn có
- Được dựa trên số lượng điểm ứng dụng (đối tượng)
- Công thức ước lượng công sức

  E = (NAP x (1 - %reuse/100)) / PROD (người — tháng)

  - NAP: số lượng điểm ứng dụng
  - %reuse: % mã lệnh được tái sử dụng từ các dự án khác
  - PROD: hiệu suất. Nó phụ thuộc vào kinh nghiệm và khả năng của nhà phát triển cũng như tính trưởng thành và khả năng của công cụ

Mô hình Application composition

- Bảng xác định hiệu suất PROD

Developer's experience and capability	Very low	L w	Nominal	High	Very high
CASE maturity and capability	Very low	Low	Nominal	High	Very high
PROD (NOP/month)	4	7	13	25	50

Mô hình Application composition

- Số lượng điểm ứng dụng NAP phụ thuộc vào:
  - Các màn hình riêng biệt được hiển thị (giao diện người dùng)
  - Các báo cáo được sinh ra bởi hệ thống
  - Các thành phần thư viện

Mô hình Application composition

- Tính số lượng điểm ứng dụng
  - Đếm số lượng màn hình, báo cáo và module
  - Xác định độ phức tạp cho từng thành phần (1 màn hình hay 1 báo cáo hay 1 module) theo bảng sau

For Screens				*For Reports*			
	Number and source of data tables				*Number and source of data tables*		
Number of views contained	Total < 4 (<2 server, <3 client)	Total < 8 (2-3 server, 3-5 client)	Total 8+ (>3 server, >5 client)	*Number of sections contained*	Total < 4 (<2 server, <3 client)	Total < 8 (2-3 server, 3-5 client)	Total 8+ (>3 server, >5 client)
<3	simple	simple	medium	0 or 1	simple	simple	medium
3 - 7	simple	medium	difficult	2 or 3	simple	medium	difficult
8 +	medium	difficult	difficult	4 +	medium	difficult	difficult

Mô hình Application composition

- Tính số điểm ứng dụng cho từng thành phần khi đã biết độ phức tạp theo bảng dưới đây

Object type	*Simple*	*Medium*	*Difficult*
Screen	1	2	3
Report	2	5	8
3GL component	-	-	10

- Tính tổng số điểm ứng dụng cho tất cả các thành phần

Mô hình Early design

- Ước lượng công sức khi các yêu cầu đã được chấp nhận và thiết kế hệ thống đang được thực hiện
- Công thức ước lượng
  - Công sức $E = a \times S^b \times M$ với
    - M: tích của 7 hệ số nhân
    - a = 2.94
    - S kích thước được ước lượng của hệ thống (*theo KDSI*)
    - b thay đổi trong khoảng 1.1 tới 1.24 tùy thuộc vào tính mới của hệ thống, tính linh động trong phát triển, các phương pháp quản lý rủi ro và tính trưởng thành của tiến trình

Mô hình Early design

- Các hệ số nhân phản ánh khả năng của nhà phát triển, các yêu cầu phi chức năng, sự hiểu biết rõ về nền tảng phát triển, v.v.
  - RCPX - product reliability and complexity
  - RUSE - the reuse required
  - PDIF - platform difficulty
  - PREX - personnel experience
  - PERS - personnel capability
  - SCED - required schedule
  - FCIL - the team support facilities
- Giá trị của các hệ số này nằm trong khoảng từ 1 (rất thấp) đến 6 (vô cùng cao)

Mô hình Reuse

- Ước lượng công sức tích hợp mã lệnh được sinh ra hay có thể tái sử dụng
- COCOMO 2 xem mã lệnh được tái sử dụng ở một trong hai dạng:
  - Mã tái sử dụng hộp đen: mã lệnh được tái sử dụng mà không cần hiểu hay sửa đổi nó => Công sức phát triển cho mã hộp đen được tính là 0
  - Mã tái sử dụng hộp trắng: mã lệnh phải được chỉnh sửa để tích hợp với mã lệnh mới hay các thành phần được tái sử dụng khác => Cần tính đến công sức phát triển

Mô hình Reuse

- Nhiều hệ thống còn bao gồm các mã lệnh được sinh ra một cách tự động từ các bộ dịch chương trình (một dạng tái sử dụng)
- Đối với mã lệnh được sinh ra tự động, dự đoán công sức để tích hợp mã lệnh này:

$$E = (ASLOC * AT/100)/ATPROD$$

  - ASLOC: số dòng mã lệnh trong các thành phần mà chúng được chỉnh sửa (tái sử dụng hộp trắng)
  - AT: tỷ lệ phần trăm mã lệnh được chỉnh sửa mà nó được sinh tự động
  - ATPROD: hiệu suất của các kỹ sư trong việc tích hợp mã lệnh đó (khoảng 2400 câu lệnh nguồn trong 1 tháng)

Mô hình Reuse

- Dự đoán số dòng mã nguồn mới tương đương ESLOC với số mã lệnh được tái sử dụng
  - ESLOC tính đến công sức hiểu phần mềm, tạo ra các thay đổi đối với mã lệnh được tái sử dụng, tạo ra các thay đổi đối với hệ thống để tích hợp mã lệnh đó. Nó còn tính đến số mã lệnh được sinh ra tự động.

    ESLOC = ASLOC * (1-AT/100) * AAM

    - ASLOC và AT như trước
    - AAM: hệ số nhân hiệu chỉnh sự thích ứng. Nó được tính từ các chi phí sửa mã lệnh được tái sử dụng, các chi phí hiểu cách tích hợp mã lệnh và các chi phí để ra quyết định tái sử dụng

Mô hình Post-architecture

- Sử dụng cùng một công thức như mô hình early design nhưng có tới 17 hệ số nhân
  - Công sức $E = a \times S^b \times M$ với
    - M: tích của 17 hệ số nhân
    - a = 2.94
    - S kích thước được ước lượng của hệ thống (*theo KDSI*)
    - $b = 1.01 + 0.01 \times \sum W_i$

Mô hình Post-architecture

- 17 hệ số nhân M

Cost drivers	Rating: Very low	Low	Nominal	High	Very high	Extra high
Product factors						
Reliability required	0.75	0.88	1.00	1.15	1.39	
Database size		0.93	1.00	1.09	1.19	
Product complexity	0.75	0.88	1.00	1.15	1.30	1.66
Required reusability		0.91	1.00	1.14	1.29	1.49
Documentation needs	0.89	0.95	1.00	1.06	1.13	
Platform factors						
Execution time constraints			1.00	1.11	1.31	1.67
Main storage constraints			1.00	1.06	1.21	1.57
Platform volatility		0.87	1.00	1.15	1.30	
Personnel factors						
Analyst capability	1.50	1.22	1.00	0.83	0.67	
Programmer capability	1.37	1.16	1.00	0.87	0.74	
Application experience	1.22	1.10	1.00	0.89	0.81	
Platform experience	1.24	1.10	1.00	0.92	0.84	
Language and tool experience	1.25	1.12	1.00	0.88	0.81	
Personnel continuity	1.24	1.10	1.00	0.92	0.84	
Project factors						
Use of software tools	1.24	1.12	1.00	0.86	0.72	
Multi-site development	1.25	1.10	1.00	0.92	0.84	0.78
Required development schedule	1.29	1.10	1.00	1.00	1.00	

Mô hình Post-architecture

- Kích thước mã lệnh S trong mô hình này được tính bằng cách cộng ba ước lượng dưới đây:
  - Sự ước lượng về số dòng mã nguồn mới được phát triển
  - Sự ước lượng về số dòng mã nguồn tương đương được tính bằng cách sử dụng mô hình reuse
  - Sự ước lượng về số dòng mã lệnh phải được sửa đổi do các thay đổi về yêu cầu

Mô hình Post-architecture

- Các hệ số W

Precedentedness	Reflects the previous experience of the organisation with this type of project. Very low means no previous experience; Extra high means that the organisation is completely familiar with this application domain.
Development flexibility	Reflects the degree of flexibility in the development process. Very low means a prescribed process is used; Extra high means that the client sets only general goals.
Architecture/risk resolution	Reflects the extent of risk analysis carried out. Very low means little analysis; Extra high means a complete and thorough risk analysis.
Team cohesion	Reflects how well the development team know each other and work together. Very low means very difficult interactions; Extra high means an integrated and effective team with no communication problems.
Process maturity	Reflects the process maturity of the organisation. The computation of this value depends on the CMM Maturity Questionnaire, but an estimate can be achieved by subtracting the CMM process maturity level from 5.

Mô hình Post-architecture

- Các hệ số W

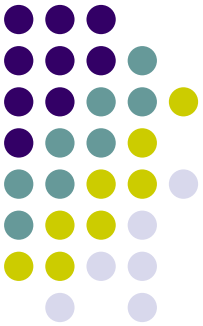
W(i)	Very Low	Low	Nominal	High	Very High	Extra High
Precedentedness	4.05	3.24	2.43	1.62	0.81	0.00
Development Flexibility	6.07	4.86	3.64	2.43	1.21	0.00
Architecture / Risk Resolution	4.22	3.38	2.53	1.69	0.84	0.00
Team Cohesion	4.94	3.95	2.97	1.98	0.99	0.00
Process Maturity	4.54	3.64	2.73	1.82	0.91	0.00

Mô hình COCOMO 2

- Bài tập: Nhóm của bạn được giao trách nhiệm ước lượng công sức thực hiện một phần mềm có khoảng 100KDSI bằng mô hình COCOMO 2. Đây là một dạng phần mềm mới đối với công ty bạn. Để thực hiện nó, công ty đã tiến hành thành lập các nhóm. Nhóm của bạn thực hiện việc ước lượng khi kiến trúc hệ thống đã được thiết kế và các thông tin chi tiết hơn về hệ thống là sẵn có. Nhìn chung, các hệ số W và hệ số nhân đều ở mức độ bình thường. Tuy nhiên, công ty đặc biệt yêu cầu rất cao về sự cần thiết của tài liệu và kinh nghiệm trong việc sử dụng công cụ và ngôn ngữ.

NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

CHƯƠNG 5 – ĐẶC TẢ YÊU CẦU



Nội dung

- Quy trình xác định các yêu cầu
- Thu thập các yêu cầu
- Phân loại các yêu cầu
- Các đặc trưng của yêu cầu
- Các ký hiệu mô hình hóa
- Các ngôn ngữ đặc tả và yêu cầu
- Lập bản mẫu cho các yêu cầu
- Tài liệu yêu cầu
- Xác minh và thẩm định

Quy trình xác định các yêu cầu

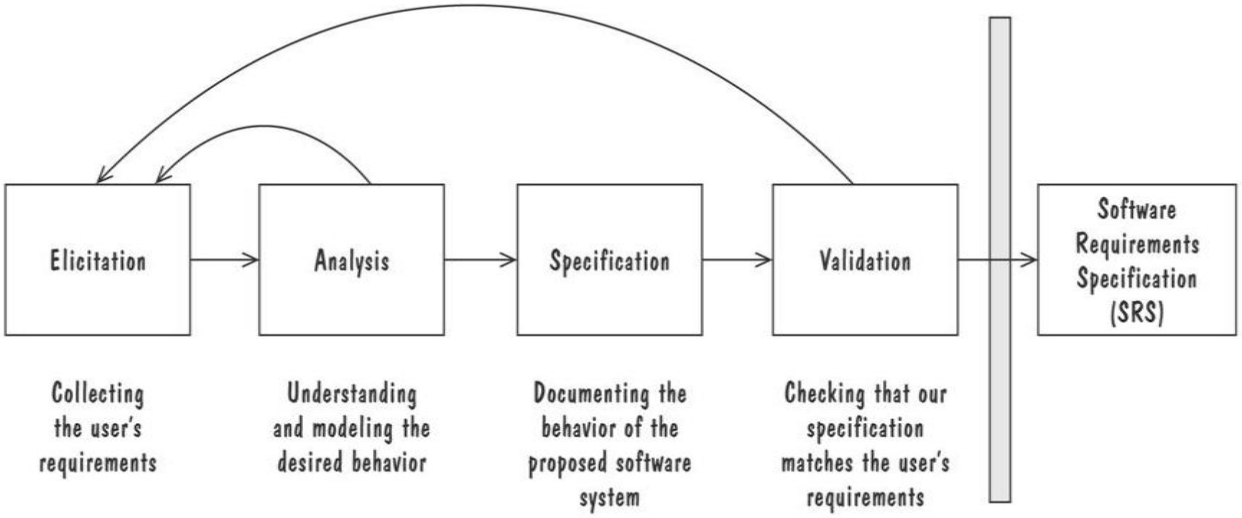
- Một ***yêu cầu*** là sự diễn đạt hành vi (behaviour) mong muốn
- Một yêu cầu đề cập đến
  - Các đối tượng hay thực thể
  - Trạng thái của đối tượng hay thực thể
  - Các chức năng được thực hiện để thay đổi trạng thái hay các đặc trưng của đối tượng
- Các yêu cầu tập trung vào nhu cầu của khách hàng chứ không phải tập trung vào giải pháp hay sự thực hiện
  - Chỉ định rõ hành vi là gì mà không nói cách thức mà hành vi đó sẽ được thực hiện

Quy trình xác định các yêu cầu

- Các yêu cầu là quan trọng
  - Các yếu tố hàng đầu làm cho dự án thất bại
    - Các yêu cầu không hoàn chỉnh
    - Thiếu sự tham gia của người sử dụng
    - Các mong muốn không thực tế
    - Thiếu sự hỗ trợ về quản lý
    - Thay đổi các yêu cầu và các đặc tả
    - Thiếu việc lập kế hoạch
    - Hệ thống không được cần nữa
  - Một phần nào đó trong quy trình xác định các yêu cầu bị liên quan đến hầu hết các nguyên nhân này
  - Các lỗi về yêu cầu có thể tốn kém nếu không được phát hiện sớm

Quy trình xác định các yêu cầu

- Được thực hiện bởi nhà phân tích yêu cầu hay nhà phân tích hệ thống
- Kết quả cuối cùng là tài liệu đặc tả các yêu cầu phần mềm



Thu thập các yêu cầu

- Khách hàng không luôn hoàn toàn mô tả một cách chính xác họ cần cái gì và vấn đề của họ là gì
- Nhà phân tích không luôn hoàn toàn hiểu nghiệp vụ của khách hàng
- Sự thảo luận với các thành viên tham gia vào hoạt động thu thập yêu cầu là quan trọng
- Sự thảo luận đưa đến sự đồng ý giữa các bên về các yêu cầu

Thu thập các yêu cầu

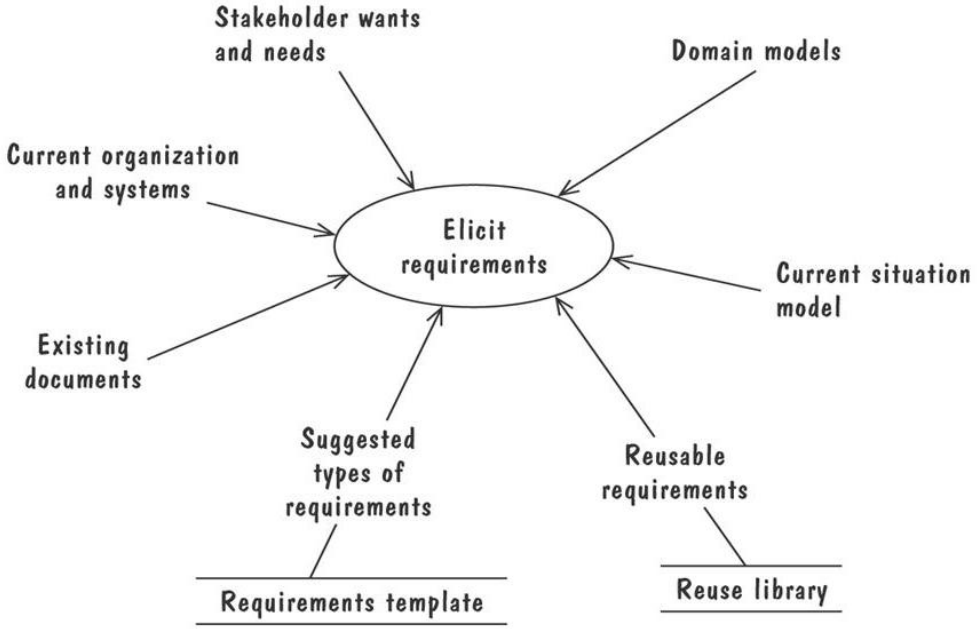
- Các thành viên tham gia trong hoạt động thu thập yêu cầu
  - **Clients**: trả tiền cho phần mềm được phát triển
  - **Customers**: mua phần mềm sau khi nó được phát triển
  - **Người dùng**: sử dụng hệ thống
  - **Chuyên gia về lĩnh vực**: biết rõ vấn đề mà phần mềm phải tin học hóa
  - **Nhà nghiên cứu thị trường**: thực hiện các cuộc khảo sát để xác định các xu hướng tương lai và những khách hàng tiềm năng
  - **Luật sư và kiểm toán viên**: biết rõ các yêu cầu của luật pháp, chính phủ hay sự an toàn
  - **Kỹ sư phần mềm và các chuyên gia công nghệ khác**: đảm bảo phần mềm là khả thi về kinh tế và công nghệ

Thu thập các yêu cầu

- Các cách thu thập yêu cầu
  - Phỏng vấn những người tham gia trong hệ thống
  - Phỏng vấn những người tham gia vào hệ thống theo nhóm
  - Xem lại các tài liệu có sẵn
  - Quan sát hệ thống hiện hành (nếu hệ thống tồn tại)
  - Theo người dùng để học về nghiệp vụ của họ một cách chi tiết hơn
  - Sử dụng các chiến lược xác định vấn đề như thiết kế ứng dụng chung (Joint Application Design)
  - Vận dụng trí tuệ (brainstorming) của người dùng hiện tại và tiềm năng để có được các yêu cầu

Thu thập các yêu cầu

- Các nguồn của các yêu cầu có thể có



Phân loại các yêu cầu

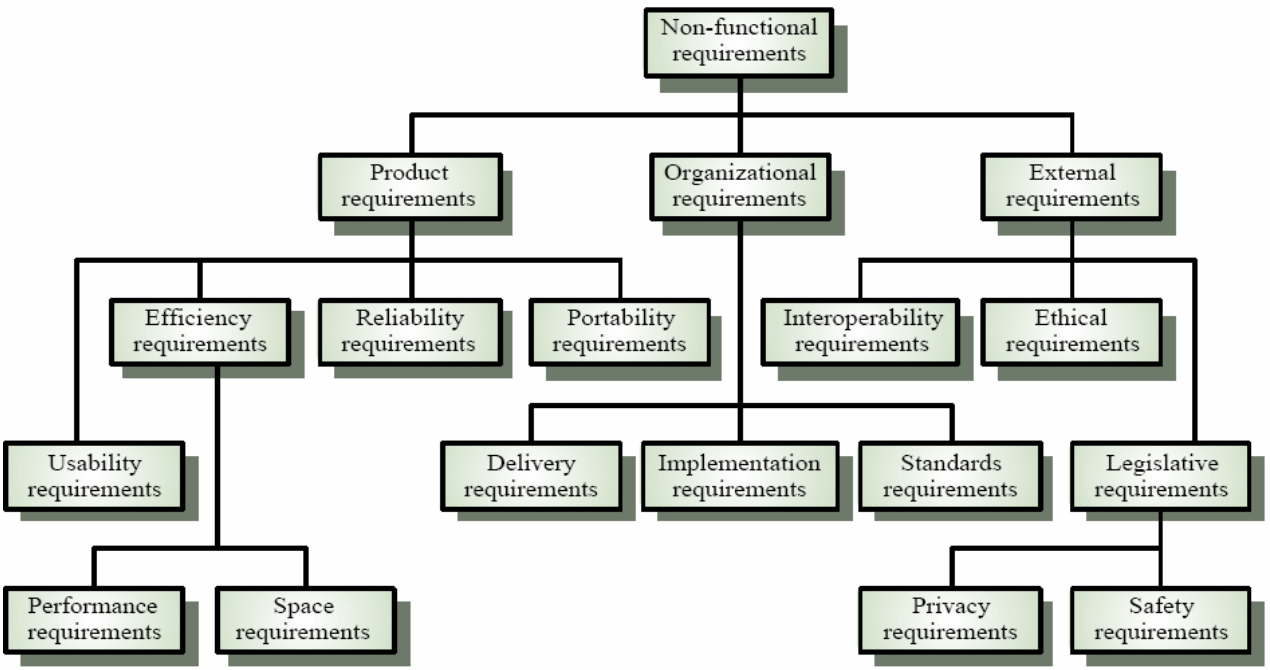
- Phân loại các yêu cầu
- Giải quyết các xung đột
- Các loại tài liệu yêu cầu

Phân loại các yêu cầu

- **Các yêu cầu chức năng**: mô tả các chức năng và dịch vụ mà hệ thống phải cung cấp
- **Các yêu cầu phi chức năng**: mô tả một đặc trưng nào đó về chất lượng nào mà phần mềm phải có; các ràng buộc thiết kế như chọn nền hay các thành phần của giao diện; các ràng buộc quy trình như sự hạn chế về các kỹ thuật và các tài nguyên được sử dụng để xây dựng hệ thống

Phân loại các yêu cầu

- Các loại yêu cầu phi chức năng



Phân loại các yêu cầu

- Giải quyết sự xung đột
  - Các thành viên khác nhau có các yêu cầu khác nhau => các yêu cầu xung đột tiềm ẩn
  - Giải quyết sự xung đột bằng cách sắp thứ tự ưu tiên cho các yêu cầu
  - Sự ưu tiên hóa phân tách các yêu cầu vào ba hạng mục sau
    - *Cần thiết*: phải được đáp ứng một cách hoàn toàn
    - *Mong muốn*: mong được đáp ứng nhưng không nhất thiết
    - *Tùy chọn*: có thể được đáp ứng nhưng có thể bị loại trừ

Phân loại các yêu cầu

- Các loại tài liệu yêu cầu
  - **Định nghĩa các yêu cầu**: một danh sách hoàn chỉnh về những thứ mà khách hàng muốn đạt được
    - Mô tả các thực thể trong môi trường nơi hệ thống sẽ được cài đặt
    - Mô tả các phép biến đổi hay các ràng buộc lên các thực thể đó
  - **Đặc tả các yêu cầu**: diễn tả lại các yêu cầu như một đặc tả về cách mà hệ thống được đề nghị sẽ hoạt động
    - Chỉ tham khảo tới các thực thể mà hệ thống có thể truy xuất chúng qua giao diện của hệ thống

Các đặc trưng của yêu cầu

- Chính xác (Correct)
- Nhất quán (Consistent)
- Không mơ hồ (Unambigious)
- Hoàn chỉnh (Complete)
- Khả thi (Feasible)
- Có liên quan (Relevant)
- Có thể kiểm thử (Testable)
- Có thể theo vết (Traceable)

Các ký hiệu mô hình hóa

- Việc có các ký hiệu chuẩn để mô hình hóa các quyết định là quan trọng
- Việc mô hình hóa giúp ta hiểu thấu đáo các yêu cầu
  - Các khuyết điểm trong mô hình bộc lộ cách hoạt động mơ hồ hay không được biết
  - Các kết xuất nhiều, xung đột nhau đối với cùng một nguồn vào bộc lộ sự không nhất quán trong các yêu cầu

Các ký hiệu mô hình hóa

- Các ký hiệu mô hình hóa
  - Lưu đồ thực thể quan hệ (Entity Relationship Diagram - ERD)
  - Dò theo sự kiện (Event Traces)
  - Máy trạng thái (State Machines)
  - Lưu đồ dòng dữ liệu (Data Flow Diagram)
  - Các chức năng và quan hệ (Functions and Relations)
  - Logic

Lưu đồ thực thể quan hệ

- Một lưu đồ ký hiệu dạng đồ thị được ưa chuộng dùng để biểu diễn các mô hình mức quan niệm
- Các thành phần cốt lõi trong lưu đồ
  - Thực thể: được vẽ như một hình chữ nhật, biểu diễn cho tập các đối tượng trong thế giới thực mà chúng có chung các đặc điểm và cách hoạt động
  - Quan hệ: được vẽ như một cạnh nối hai thực thể, với hình thoi ở chính giữa cạnh xác định loại quan hệ
  - Thuộc tính: một diễn giải trong thực thể. Nó mô tả dữ liệu hay các đặc tính được kết hợp với thực thể

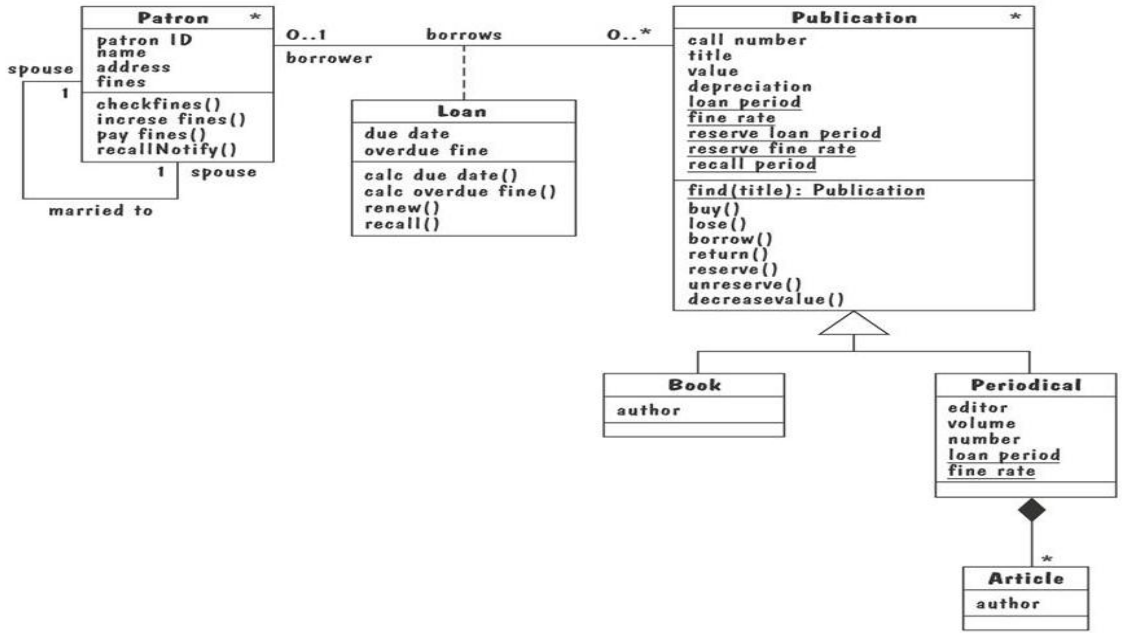
Lưu đồ thực thể quan hệ

- Ví dụ:

- Lược đồ thực thể quan hệ là phổ biến vì
  - Nó cung cấp một cái nhìn tổng quan về vấn đề đã được xác định
  - Tính tổng quan là tương đối ổn định khi có thay đổi trong các yêu cầu
- Lược đồ thực thể quan hệ có vẻ phù hợp để mô hình hóa vấn đề sớm trong quy trình xác định các yêu cầu

Lưu đồ thực thể quan hệ

- Ví dụ: sơ đồ lớp của UML (UML Class Diagram) là một lược đồ thực thể quan hệ phức tạp



Lưu đồ thực thể quan hệ

- Ví dụ: sơ đồ lớp của UML là lược đồ thực thể quan hệ phức tạp
  - Các thuộc tính và các phương thức được kết hợp với lớp hơn là với các thể hiện của lớp
  - Thuộc tính phạm vi lớp, được biểu diễn như một thuộc tính được gạch chân, là một giá trị dữ liệu mà nó được chia sẻ bởi tất cả các thể hiện của lớp
  - Phương thức phạm vi lớp, được biểu diễn như một phương thức được gạch chân, là một phương thức được thực hiện bởi lớp trừu tượng (hơn là bởi các thể hiện của lớp) trên một thể hiện mới hay trên toàn bộ
  - Liên kết (association), được đánh dấu như một đường nối giữa hai lớp, chỉ ra mối quan hệ giữa các thực thể của các lớp

Lưu đồ thực thể quan hệ

- Ví dụ: sơ đồ lớp của UML là lược đồ thực thể quan hệ phức tạp
  - Liên kết kết tập (aggregation association) là một mối quan hệ "chứa đựng" về cấu trúc hoặc hành vi của một phần tử trong một tập hợp. Hình thoi rỗng được ký hiệu trên liên kết về phía lớp biểu diễn tập hợp chứa đựng.
  - Liên kết cấu thành (composition association) là một liên kết kết tập đặc biệt. Nó mô tả một sự chứa đựng về cấu trúc giữa các thể hiện. Hình thoi đặc được ký hiệu trên liên kết ở phía lớp chứa.
  - Liên kết tổng quát hóa
  - …

Dò theo sự kiện

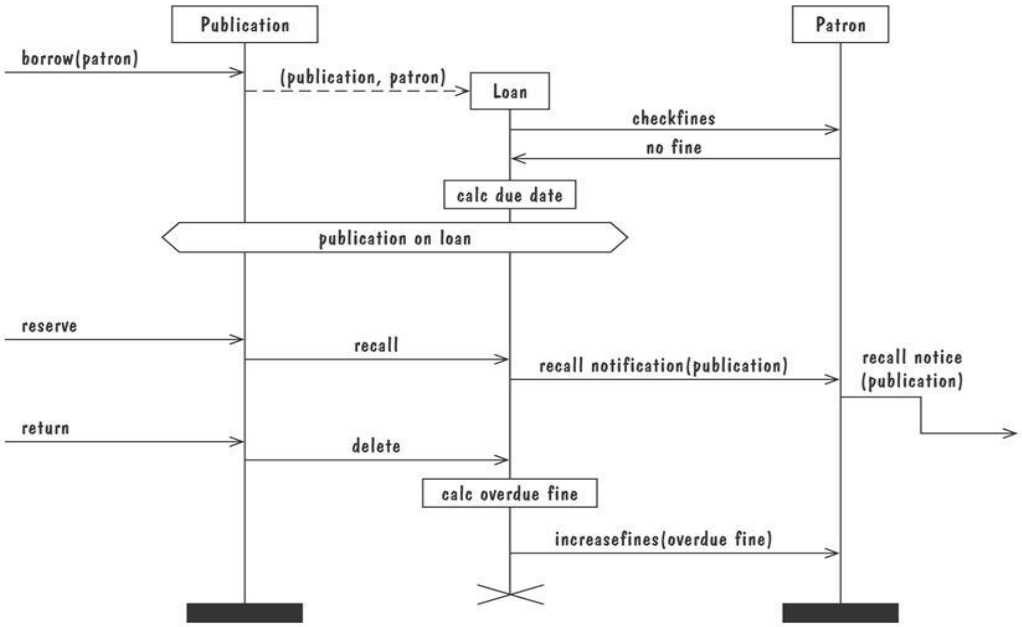
- Mô hình thực thể quan hệ không nói gì về cách mà các thực thể hành xử
- Dò theo sự kiện là sự mô tả ở dạng đồ họa của một chuỗi các sự kiện mà chúng được trao đổi giữa các thực thể của thế giới thực
  - *Đường dọc*: đường thời gian của một thực thể riêng biệt; tên của nó xuất hiện trên đỉnh của đường
  - *Đường ngang*: một sự kiện hay sự tương tác giữa hai thực thể
  - Thời gian tiến triển theo hướng từ trên xuống
- Mỗi đồ thị mô tả một đơn dò theo sự kiện, đại diện cho một trong một vài cách hoạt động có thể có
- Các dò theo sự kiện phổ biến đối với nhà phát triển và khách hàng vì chúng tương đối chính xác, đơn giản và dễ hiểu)

Dò theo sự kiện

- Ví dụ: biểu đồ trình tự thông điệp là ký hiệu dò theo sự kiện được cải tiến, với các phương tiện cho phép tạo ra hay hủy bỏ các thực thể, xác định các hoạt động và định thời, và tạo ra các dò theo
  - *Đường dọc* biểu diễn cho một thực thể tham gia
  - *Thông điệp* được vẽ bằng một mũi tên hướng từ thực thể gửi sang thực thể nhận.
  - *Hoạt động* được vẽ bằng hình chữ nhật được gán nhãn và được đặt trên đường thực thi của thực thể
  - *Điều kiện* là các trạng thái quan trọng trong sự tiến hóa của thực thể, được biểu diễn bằng hình sáu cạnh có gán nhãn

Dò theo sự kiện

- Ví dụ biểu đồ trình tự thông điệp là dò theo sự kiện được cải tiến: biểu đồ trình tự thông điệp cho giao dịch mượn tài liệu của thư viện



Máy trạng thái

- Sự mô tả ở dạng đồ họa của tất cả các cuộc đối thoại giữa hệ thống và môi trường của nó
  - Nút (*trạng thái*) biểu diễn một tập ổn định các điều kiện mà nó tồn tại giữa các lần xuất hiện của sự kiện
  - Cạnh (*dịch chuyển*) biểu diễn cho sự thay đổi về hành vi hay điều kiện do sự xuất hiện của một sự kiện
- Hữu ích cả cho việc xác định hành vi động và cho việc mô tả cách mà hành vi nên thay đổi để đáp ứng được lịch sử của các sự kiện mà chúng đã xuất hiện

Máy trạng thái

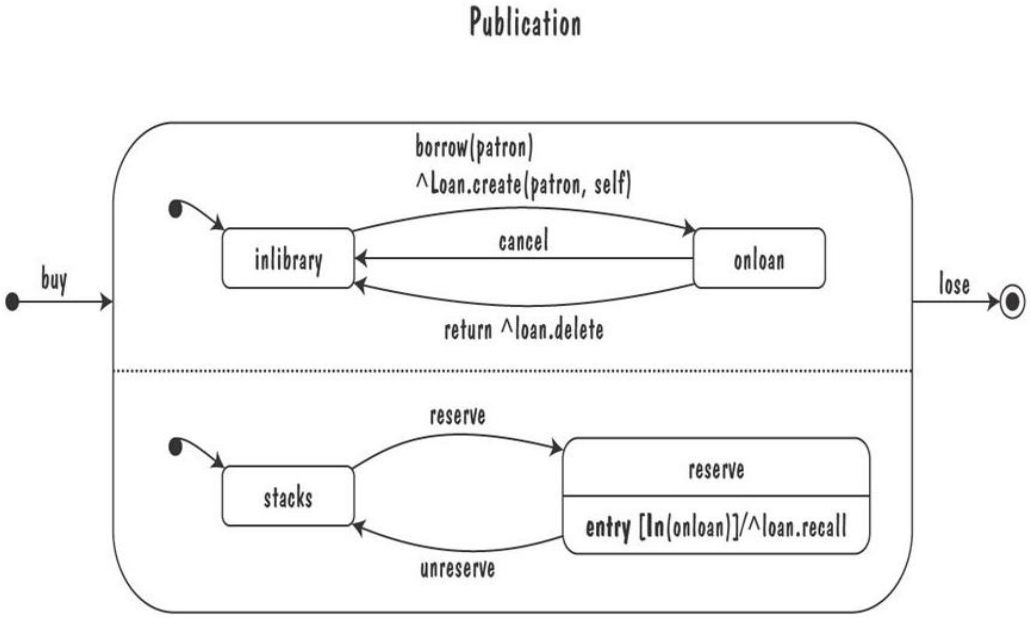
- Đường đi: bắt đầu từ trạng thái bắt đầu của máy và đi theo các dịch chuyển từ trạng thái này sang trạng thái khác
- Máy trạng thái hữu hạn: với mỗi trạng thái hay sự kiện, có một đáp ứng duy nhất

Máy trạng thái

- Ví dụ: sơ đồ trạng thái của UML
  - Mô tả hành vi động của các đối tượng trong một lớp UML
  - Có một cú pháp phong phú, bao gồm sự phân cấp trạng thái, sự đồng thời và giao tiếp giữa các máy

Máy trạng thái

- Ví dụ sơ đồ trạng thái của UML cho lớp `Publication`

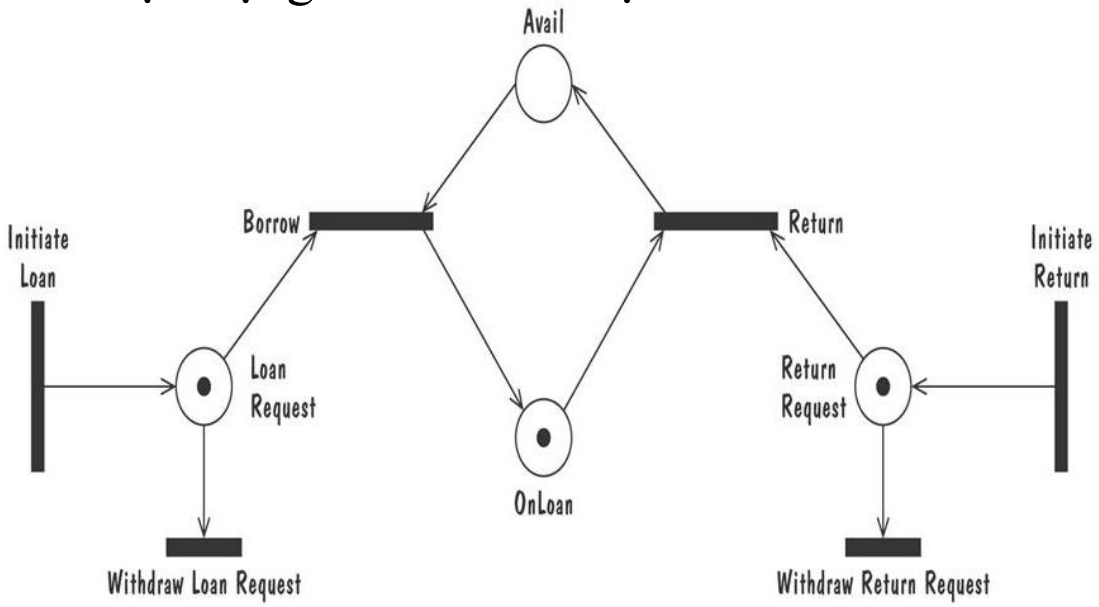


Máy trạng thái

- Ví dụ: mạng Petri
  - Ký pháp dịch chuyển trạng thái được sử dụng để mô hình hóa các hoạt động đồng thời và sự tương tác của chúng
    - Vòng tròn biểu diễn cho các hoạt động hay các điều kiện
    - Thanh ngang/ dọc biểu diễn cho các dịch chuyển
    - Cung nối kết một dịch chuyển với các hoạt động và điều kiện vào và ra của nó

Máy trạng thái

- Ví dụ: mạng Petri cho mượn sách



Lưu đồ dòng dữ liệu

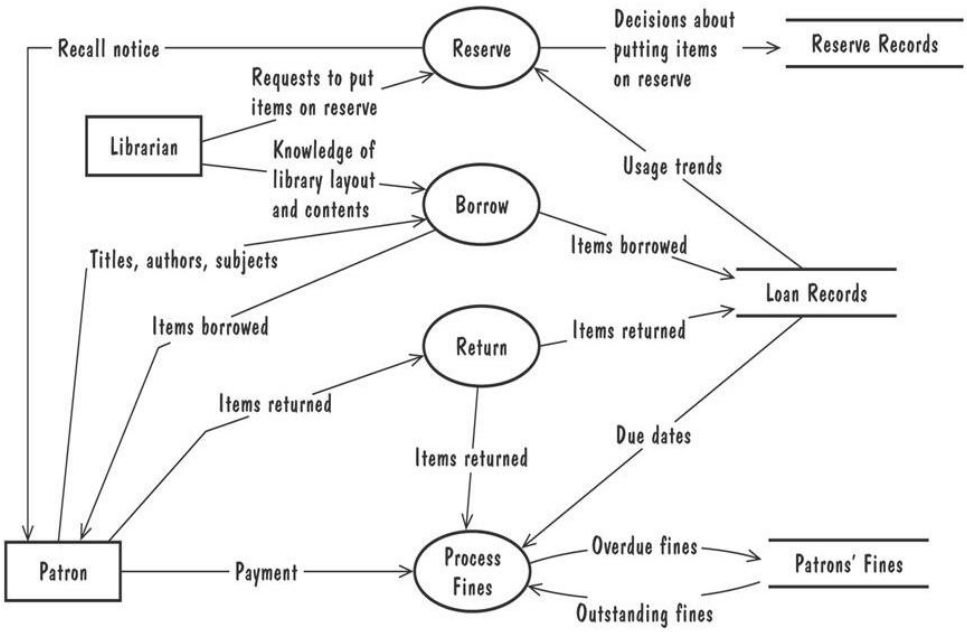
- Lược đồ thực thể quan hệ phân rã vấn đề theo các thức thể
- Dò theo sự kiện phân rã vấn đề theo kịch bản
- Máy trạng thái phân rã vấn đề theo trạng thái điều khiển
- Cả ba chỉ mô tả các hành vi mức thấp => rất khó nhìn thấy chức năng mức cao của mô hình

Lưu đồ dòng dữ liệu

- Lưu đồ dòng dữ liệu mô hình hóa chức năng và dòng dữ liệu từ chức năng này sang chức năng khác
  - Hình elip biểu diễn cho quy trình hay chức năng: biến đổi dữ liệu
  - Mũi tên biểu diễn cho dòng dữ liệu (vào hay ra của chức năng)
  - Hai đường song song biểu diễn cho kho dữ liệu: lưu giữ các dữ liệu
  - Hình chữ nhật biểu diễn cho các tác nhân: các thực thể cung cấp dữ liệu vào và nhận kết quả kết xuất

Lưu đồ dòng dữ liệu

- Lưu đồ dòng dữ liệu mức cao biểu diễn cho bài toán thư viện



Lưu đồ dòng dữ liệu

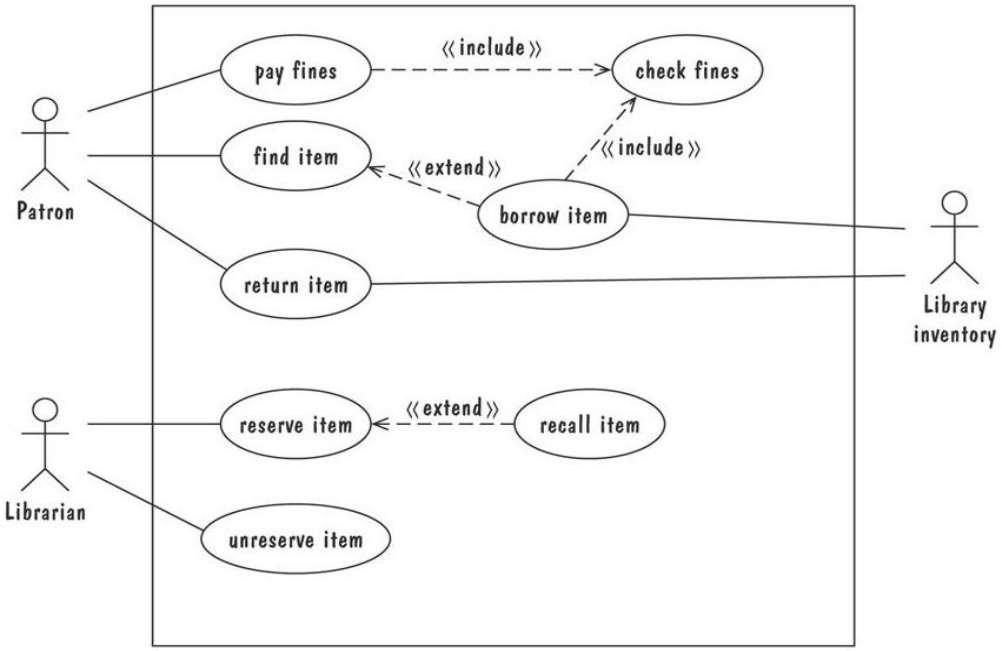
- Thuận lợi:
  - Cung cấp một mô hình trực quan về chức năng mức cao của hệ thống được đề nghị và của các phụ thuộc dữ liệu giữa các quy trình khác nhau
- Khó khăn :
  - Có thể làm tăng tính mơ hồ đối với nhà phát triển phần mềm người chưa quen với vấn đề đang được mô hình hóa

Lưu đồ dòng dữ liệu

- Ví dụ: sơ đồ use case của UML
  - Các thành phần
    - Đường biên của hệ thống (được ký hiệu bởi hình chữ nhật)
    - Tác nhân (được ký hiệu bởi hình người hay << >>)
    - Trường hợp sử dụng – use case - (được ký hiệu bằng hình oval). Một use case biểu diễn một chức năng được yêu cầu nào đó và biến thể của nó
    - Quan hệ giữa tác nhân và use case hay giữa các use case (được biểu diễn bằng các đường hay đường nét đứt)

Lưu đồ dòng dữ liệu

- Ví dụ lược đồ use case của UML: một số use case của thư viện



Các ngôn ngữ đặc tả và yêu cầu

- Ngôn ngữ mô hình hóa hợp nhất (Unified Modeling Language - UML)
- Ngôn ngữ mô tả và đặc tả (Specification and Description Language – SDL)

Các ngôn ngữ đặc tả và yêu cầu

- UML (Unified Modeling Language)
  - Kết hợp nhiều sơ đồ ký hiệu
  - Các sơ đồ UML được sử dụng trong suốt sự định nghĩa và đặc tả các yêu cầu
    - Sơ đồ use case (Lưu đồ dòng dữ liệu mức cao)
    - Sơ đồ lớp (Lưu đồ thực thể quan hệ)
    - Sơ đồ tuần tự (Dò theo sự kiện)
    - Sơ đồ cộng tác (Dò theo sự kiện)
    - Sơ trạng thái (Mô hình trạng thái máy)

Các ngôn ngữ đặc tả và yêu cầu

- Ngôn ngữ mô tả và đặc tả (SDL)
  - Xác định hành vi của các quy trình phân tán, đồng thời và thời gian thực mà chúng giao tiếp với nhau thông qua các hàng đợi thông điệp không giới hạn
  - Bao gồm
    - Sơ đồ hệ thống SDL (Lưu đồ dòng dữ liệu)
    - Sơ đồ khối SDL (Lưu đồ dòng dữ liệu)
    - Sơ đồ quy trình SDL (Mô hình máy trạng thái)
    - Kiểu dữ liệu SDL (Đặc tả đại số)
  - Thường được đi kèm bởi một tập biểu đồ trình tự của thông điệp

Lập bản mẫu cho các yêu cầu

- Xây dựng bản mẫu
  - Để thu thập các chi tiết của hệ thống được đề nghị
  - Để cố gắng lấy được thông tin phản hồi từ người dùng tiềm năng về
    - Những khía cạnh nào mà họ muốn cải tiến
    - Những đặc tính nào là không quá hữu ích
    - Chức năng nào đang thiếu
  - Giúp xác định xem vấn đề của khách hàng có giải pháp khả thi hay không
  - Hỗ trợ trong việc thăm dò các chọn lựa để tối ưu hóa các yêu cầu về chất lượng

Lập bản mẫu cho các yêu cầu

- Các cách tiếp cận để làm bản mẫu
  - Cách tiếp cận được làm ra để sử dụng một lần duy nhất (throwaway prototype)
    - Được phát triển để nghiên cứu thêm về vấn đề hay giải pháp được đề nghị , và không bao giờ được xem là một phần của phần mềm được phân phối
  - Cách tiếp cận tiến hóa (evolutionary prototype)
    - Được phát triển không chỉ giúp chúng ta trả lời các câu hỏi mà còn được kết hợp vào sản phẩm cuối cùng
    - Bản mẫu cuối cùng phải biểu thị các yêu cầu về chất lượng của sản phẩm cuối
  - Cả hai kỹ thuật này đôi khi được gọi là làm bản mẫu nhanh

Lập bản mẫu cho các yêu cầu

- Sự khác nhau giữa lập bản mẫu và mô hình hóa
  - Lập bản mẫu
    - Tốt khi trả lời những câu hỏi về giao diện người dùng
  - Mô hình hóa
    - Trả lời nhanh các câu hỏi về các ràng buộc theo thứ tự mà theo đó các sự kiện nên xuất hiện, về sự đồng bộ của các hoạt động

Tài liệu yêu cầu

- Định nghĩa các yêu cầu - Các bước của quy trình
  - Phác thảo mục đích chung và phạm vi của hệ thống, bao gồm các lợi ích liên quan, các mục tiêu và mục đích
  - Mô tả nền tảng và nhân tố cơ bản ở sau sự đề xuất cho hệ thống mới
  - Mô tả các đặc trưng cần thiết của một giải pháp có thể chấp nhận được
  - Mô tả môi trường trong đó hệ thống sẽ vận hành
  - Phác thảo sự mô tả về đề xuất, nếu khác hàng có một đề xuất cho giải quyết vấn đề
  - Liệt kê bất cứ giả định nào về cách mà môi trường hoạt động

Tài liệu yêu cầu

- Đặc tả các yêu cầu - Các bước của quy trình
  - Mô tả chi tiết tất cả các đầu vào, đầu ra, bao gồm
    - Các nguồn của đầu vào
    - Các đích của đầu ra
    - Các miền giá trị
    - Định dạng dữ liệu cho dữ liệu vào/ra
    - Các giao thức của dữ liệu
    - Tổ chức và định dạng của cửa sổ
    - Ràng buộc thời gian
  - Diễn đạt lại chức năng được yêu cầu dưới dạng các đầu vào/ra của giao diện
  - Đưa ra tiêu chuẩn phù hợp cho từng yêu cầu về chất lượng của khách hàng

Tài liệu yêu cầu

- Chuẩn IEEE cho đặc tả các yêu cầu phần mềm

1.Introduction to the Document
- 1.1 Purpose of the Product
- 1.2 Scope of the Product
- 1.3 Acronyms, Abbreviations, Definitions
- 1.4 References
- 1.5 Outline of the rest of the SRS

2.General Description of Product
- 2.1 Context of Product
- 2.2 Product Functions
- 2.3 User Characteristics
- 2.4 Constraints
- 2.5 Assumptions and Dependencies

Tài liệu yêu cầu

3. Specific Requirements
   - 3.1 External Interface Requirements
     - 3.1.1 User Interfaces
     - 3.1.2 Hardware Interfaces
     - 3.1.3 Software Interfaces
     - 3.1.4 Communications Interfaces
   - 3.2 Functional Requirements
     - 3.2.1 Class 1
     - 3.2.2 Class 2
     - …
   - 3.3 Performance Requirements
   - 3.4 Design Constraints
   - 3.5 Quality Requirements
   - 3.6 Other Requirements
4. Appendices

Xác minh & Thẩm định

- Thẩm định (validation) của các yêu cầu: ta kiểm tra xem định nghĩa các yêu cầu có phản ánh chính xác nhu cầu của khách hàng
- Xác minh (verification): ta kiểm tra xem một tài liệu được tạo ra có tuân theo tài liệu khác

Xác minh & Thẩm định

- Các kỹ thuật thẩm định

Walkthroughs
Reading
Interviews
Reviews
Checklists
Models to check functions and relationships
Scenarios
Prototypes
Simulation
Formal inspections

Xác minh & Thẩm định

- Xác minh
  - Kiểm tra xem tài liệu đặc tả các yêu cầu có tương đương với định nghĩa các yêu cầu
  - Đảm bảo rằng nếu ta thực hiện một hệ thống mà nó đáp ứng sự đặc tả thì hệ thống sẽ đáp ứng các yêu cầu của khách hàng
  - Đảm bảo rằng mỗi yêu cầu trong tài liệu định nghĩa là có thể theo dõi qua dấu vết của đặc tả

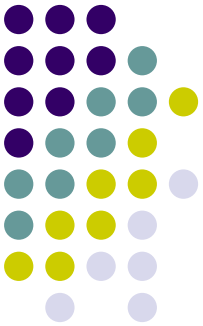
Kiểm tra và xác nhận tính hợp lệ

- Các kỹ thuật xác minh

Cross-referencing
Simulation
Consistency checks
Completeness checks
Check for unreachable states or transitions
Model checking
Mathematical proofs

NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

CHƯƠNG 6 – THIẾT KẾ



Nội dung

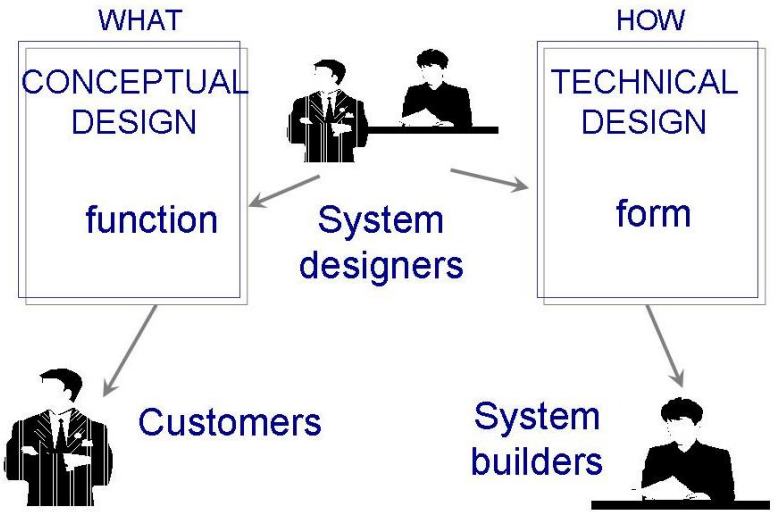
- Định nghĩa về thiết kế
- Các nội dung thiết kế
- Một số vấn đề trong sự tạo ra thiết kế
- Đặc trưng của thiết kế hoàn thiện
- Tài liệu thiết kế

Định nghĩa về thiết kế

- Thiết kế là quá trình sáng tạo thực hiện việc biến đổi vấn đề sang giải pháp
- Sự mô tả của một giải pháp còn được biết như là thiết kế
  - Đặc tả các yêu cầu định nghĩa vấn đề
  - Tài liệu thiết kế xác định một giải pháp cụ thể cho vấn đề

Định nghĩa về thiết kế

- Thiết kế là một quá trình tương tác gồm hai phần
  - Thiết kế mức khái niệm (Thiết kế hệ thống)
  - Thiết kế kỹ thuật



Định nghĩa về thiết kế

- Thiết kế mức khái niệm nói với khách hàng những cái mà hệ thống phải làm:
  - Dữ liệu đến từ đâu?
  - Điều gì sẽ xảy ra với dữ liệu trong hệ thống?
  - Hệ thống trông giống cái gì?
  - Những lựa chọn nào sẽ được cung cấp cho người dùng?
  - Các báo cáo và màn hình giống cái gì?
  - Định thời gian cho các sự kiện

Định nghĩa về thiết kế

- Thiết kế mức khái niệm

  Các đặc trưng của một thiết kế mức khái niệm hoàn thiện
  - Theo ngôn ngữ mà khách hàng có thể hiểu
  - Không có các từ kỹ thuật
  - Mô tả các chức năng của hệ thống
  - Độc lập với sự thực hiện
  - Được liên kết với các yêu cầu

Định nghĩa về thiết kế

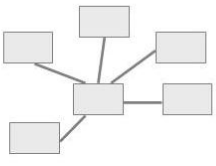
- Thiết kế kỹ thuật nói với lập trình viên những cái mà hệ thống phải làm như:
  - Các thành phần phần cứng chính và chức năng của chúng
  - Sự phân cấp và các chức năng của các thành phần phần mềm
  - Các cấu trúc dữ liệu và dòng dữ liệu

Định nghĩa về thiết kế

- Sự khác nhau trong tài liệu thiết kế

"The user will be able to route messages to any other user on any other network computer."

CONCEPTUAL DESIGN



Network topology
Protocol used
Prescribed bps rate
. . .

TECHNICAL DESIGN

Định nghĩa về thiết kế

- Mỗi phương pháp thiết kế đều liên quan đến một dạng phân rã
  - Bắt đầu bằng sự mô tả mức cao về các thành phần chính của hệ thống
  - Tạo ra các mô tả mức thấp hơn về cách mà các đặc trưng và chức năng của hệ thống phù hợp với nhau

Định nghĩa về thiết kế

- Những phương pháp tạo ra thiết kế
  - Phân rã theo mô đun
  - Phân rã hướng dữ liệu
  - Phân rã hướng sự kiện
  - Thiết kế trong ngoài
  - Thiết kế hướng đối tượng
- Dù là phương pháp thiết kế nào, nó cũng phân rã hệ thống thành các thành phần hay mô đun

Các nội dung thiết kế

- Nhà thiết kế thực hiện các công việc:
  - Thiết kế kiến trúc
  - Thiết kế dữ liệu
  - Thiết kế giao diện
  - Thiết kế thủ tục (thuật toán)

Thiết kế kiến trúc

- Thiết kế kiến trúc
  - Phân rã hệ thống thành một tập các thành phần
  - Xác định các giao diện giữa các thành phần
  - Định nghĩa các hoạt động tạo ra hệ thống từ các thành phần
- Cung cấp một cái nhìn tổng thể về hệ thống

Thiết kế kiến trúc

- Các thành phần của hệ thống có thể được kết hợp với nhau theo các kiểu kiến trúc:
  - Đường dẫn và bộ lọc
  - Thiết kế hướng đối tượng
  - Sự dẫn chứng ẩn
  - Phân lớp
  - Kho chứa
  - Bộ phiên dịch
  - Kiểm soát quy trình
  - Khách – chủ

Thiết kế dữ liệu

- Thiết kế dữ liệu
  - Các thành phần dữ liệu và bảng để tạo CSDL
  - Các cấu trúc dữ liệu
- Các mức thiết kế dữ liệu
  - Thiết kế cấu trúc logic: các quan hệ chuẩn, các khóa, các tham chiếu, các cấu trúc thao tác dữ liệu
  - Thiết kế cấu trúc vật lý: các file, các kiểu, kích cỡ

Thiết kế giao diện

- Thiết kế giao diện
  - Các form nhập liệu
  - Các reports và những kết xuất mà hệ thống mới phải sinh ra

Thiết kế giao diện

- Các vấn đề then chốt được xem xét khi thiết kế giao diện
  - Các vấn đề về văn hóa (dân tộc, giới tính, nghề nghiệp, tuổi tác, vùng miền)
  - Sở thích của người dùng
- Một số lưu ý khi thiết kế giao diện
  - Nên có sự đồng nhất giữa các giao diện (menu, lệnh, hiển thị...)
  - Đặt tên nhãn ngắn gọn, dễ nhớ
  - Tối ưu trong trình bày hộp thoại và di chuyển chuột

Thiết kế giao diện

- Một số lưu ý khi thiết kế giao diện
  - Hạn chế nhập dữ liệu trực tiếp, nếu có thể, nên cho người dùng chọn lựa từ một số dữ liệu có sẵn
  - Yêu cầu xác nhận những tác vụ mang tính phá hủy (xoá dữ liệu)
  - Chấp nhận lỗi từ phía người sử dụng
  - Nên cung cấp feedback cho người dùng
  - Tạo ra thông báo lỗi có ý nghĩa
  - Cung cấp trợ giúp

Thiết kế thủ tục

- Thiết kế thủ tục (thuật toán)
  - Giải thích quá trình xử lý từ input đến output.
- Phương pháp biểu diễn
  - Lưu đồ giải thuật
  - Ngôn ngữ giả

Một số vấn đề trong việc tạo ra thiết kế

- Thiết kế cộng tác
  - Hầu hết các dự án làm việc cộng tác
  - Các vấn đề trong thiết kế cộng tác
    - Ai là người phù hợp nhất để thiết kế từng bộ phận của hệ thống
    - Viết tài liệu thiết kế như thế nào
    - Làm thế nào để kết hợp các thành phần thiết kế thành một thiết kế hợp nhất
  - Các vấn đề trong việc thực hiện thiết kế cộng tác
    - Sự khác nhau về kinh nghiệm cá nhân, sự hiểu biết, sở thích

Một số vấn đề trong việc tạo ra thiết kế

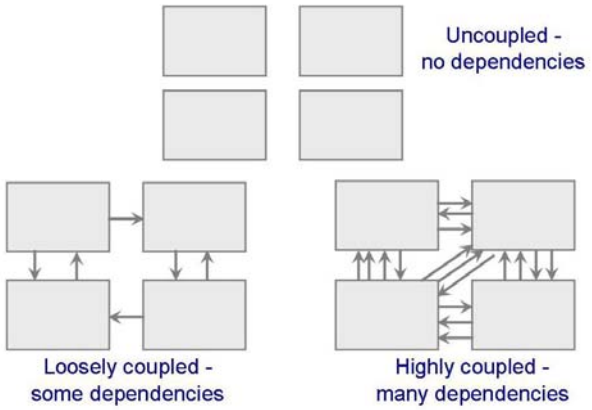
- Sự đồng thời
  - Các vấn đề
    - Tính nhất quán của dữ liệu được chia sẻ giữa các thành phần mà chúng thực thi tại cùng thời điểm
    - Đảm bảo rằng một hoạt động không can thiệp vào các hoạt động khác
  - Các giải pháp
    - *Sự đồng bộ*: phương pháp cho phép hai hoạt động thực hiện đồng thời mà không can thiệp vào nhau
    - *Loại trừ lẫn nhau*: một quy trình truy xuất một phần tử dữ liệu, không có quy trình nào khác ảnh hưởng tới phần tử
    - *Giám sát*: một đối tượng trừu tượng kiểm soát sự loại trừ lẫn nhau của một quy trình cụ thể

Các đặc trưng của một thiết kế hoàn thiện

- Sự độc lập của thành phần
  - Sự ghép nối (coupling)
  - Sự gắn kết (cohesion)
- Nhận dạng và xử lý các ngoại lệ
- Ngăn chặn và chấp nhận các lỗi trong giới hạn cho phép
  - Chủ động
  - Bị động

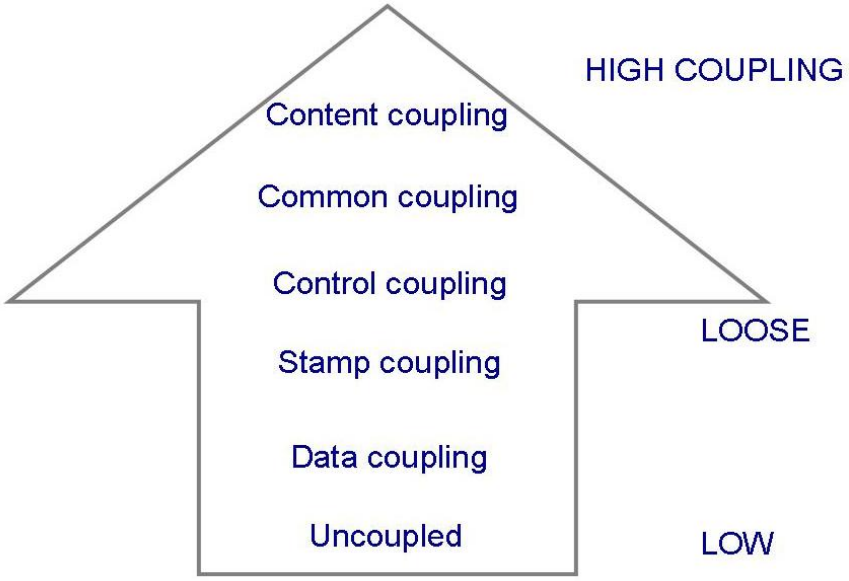
Sự ghép nối

- Các thành phần *được ghép nối cao* khi có một lượng lớn các phụ thuộc
- Các thành phần *được ghép nối lỏng lẻo* khi có một sự phụ thuộc nào đó nhưng các quan hệ giữa các thành phần yếu
- Các thành phần *không được ghép nối* khi không có các quan hệ nào cả



Sự ghép nối

- Ta có thể đo sự ghép nối theo mức độ phụ thuộc

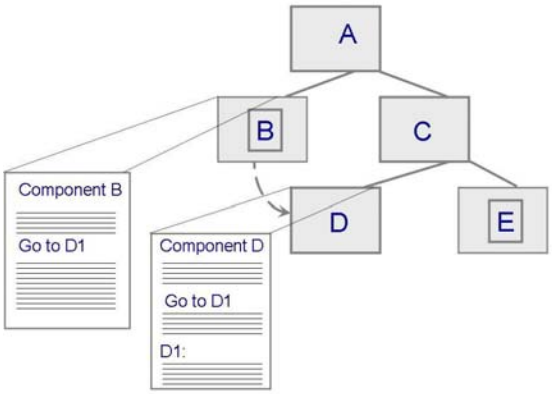


Sự ghép nối

- Các loại ghép nối
  - Ghép nối nội dung: một thành phần sửa dữ liệu nội bộ của một thành phần khác hay một thành phần rẽ nhánh sang một thành phần khác
  - Ghép nối chung: các thành phần truy xuất và làm thay đổi dữ liệu chung
  - Ghép nối kiểm soát: một thành phần truyền các tham số để điều khiển hoạt động của một thành phần khác
  - Ghép nối cấu trúc dữ liệu: cấu trúc dữ liệu được sử dụng để truyền thông tin từ một thành phần này sang một thành phần khác
  - Ghép nối dữ liệu: chỉ có dữ liệu được truyền từ từ một thành phần này sang một thành phần khác

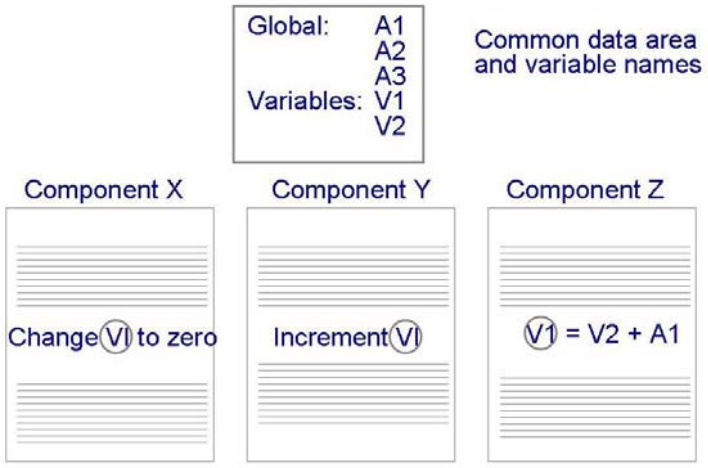
Sự ghép nối

- Ví dụ



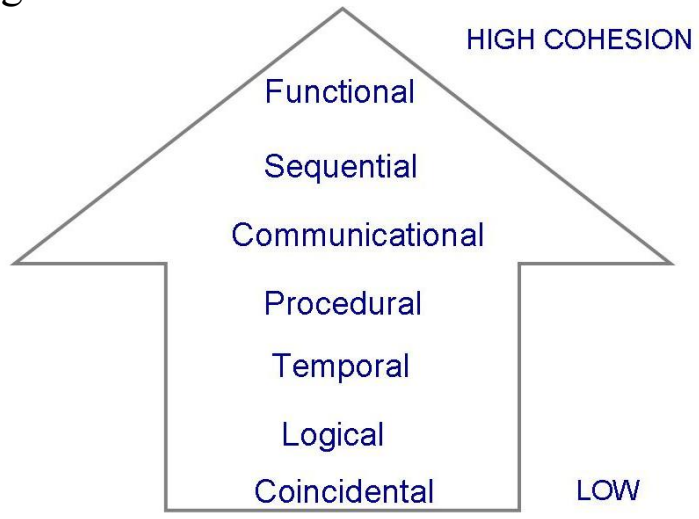
Ghép nối nội dung

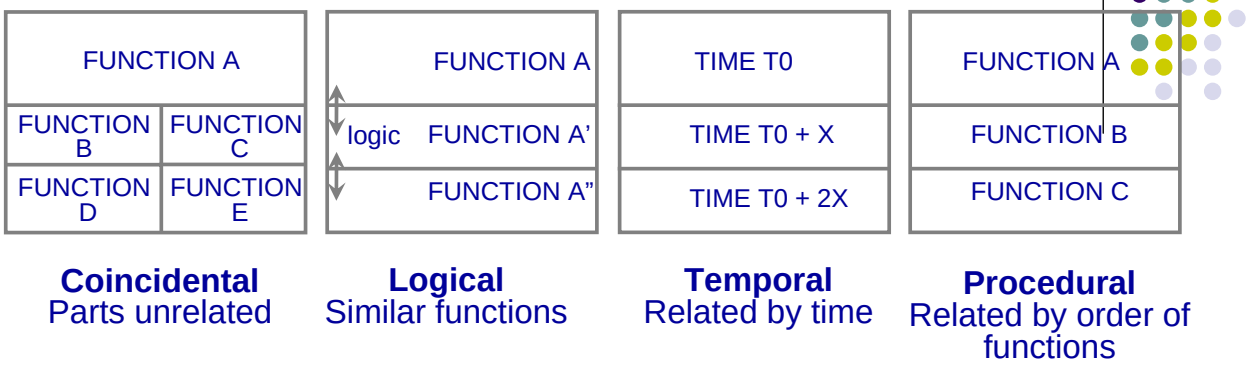
Ghép nối chung



Sự gắn kết

- Một thành phần là gắn kết nếu tất cả các thành viên của thành phần được hướng tới và cần thiết để thực hiện cùng một công việc
- Một số dạng gắn kết



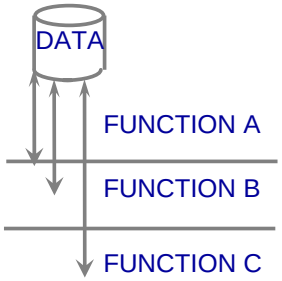


Coincidental
Parts unrelated

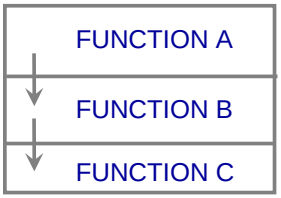
Logical
Similar functions

Temporal
Related by time

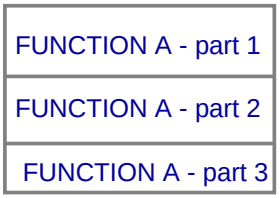
Procedural
Related by order of functions



Functional
Access same data



Sequential
Output of one part is input to next



Functional
Sequential with complete, related functions

Nhận dạng và xử lý ngoại lệ

- Các ngoại lệ: những tình huống mà nó ngược lại với cái mà ta thực sự muốn hệ thống làm
  - Không thực hiện được việc cung cấp một dịch vụ
  - Cung cấp dữ liệu hay dịch vụ sai
  - Làm hư dữ liệu
- Các ngoại lệ có thể được xử lý theo một trong ba cách sau
  - Thử lại
  - Hiệu chỉnh
  - Báo cáo

Ngăn chặn và chấp nhận các lỗi

- Phát hiện lỗi chủ động: định kỳ kiểm tra các dấu hiệu về lỗi hoặc cố gắng giải quyết trước khi lỗi xuất hiện
- Phát hiện lỗi bị động: chờ cho đến khi lỗi xuất hiện trong suốt sự thực thi
- Hiệu chỉnh lỗi: sự đền bù của hệ thống cho sự hiện diện của lỗi
- Chấp nhận lỗi: cô lập những thiệt hại được gây ra bởi lỗi

Viết tài liệu thiết kế

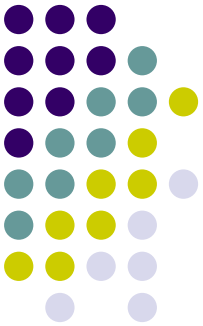
- Tài liệu thiết kế gồm:
  - Mục nêu lý do cơ bản của thiết kế
    - Phác thảo những vấn đề then chốt và các thỏa hiệp
  - Mục mô tả về các thành phần của hệ thống
  - Mục xác định cách mà người dùng tương tác với hệ thống
  - Tập các biểu đồ và ký pháp hình thức mô tả toàn bộ tổ chức và cấu trúc của hệ thống

Viết tài liệu thiết kế

- Mục xác định cách mà người sử dụng tương tác với hệ thống
  - Các menu và các định dạng màn hình hiển thị
  - Giao diện người dùng: các phím chức năng, v.v.
  - Kết nhập: dữ liệu đến từ đâu, cách mà chúng được định dạng, chúng được lưu giữ trên phương tiện nào
  - Kết xuất: dữ liệu đến từ đâu, cách mà chúng được định dạng, chúng được lưu giữ trên phương tiện nào
  - Các đặc trưng chức năng chung
  - Các ràng buộc về sự thực thi
  - Các thủ tục lưu giữ
  - Cách phương pháp xử lý lỗi

NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

CHƯƠNG 7 – LẬP TRÌNH



Nội dung

- Các chuẩn và thủ tục lập trình
- Chọn ngôn ngữ lập trình
- Nguyên tắc lập trình
- Viết tài liệu

Các chuẩn và thủ tục lập trình

- Chuẩn và thủ tục giúp lập trình viên
  - Tổ chức các ý định và tránh các lỗi
    - Các tài liệu theo chuẩn giúp ta quay lại công việc mà không mất dấu những gì đã làm
    - Các tài liệu theo chuẩn giúp ta định vị các lỗi và tạo ra các thay đổi
  - Dịch các thiết kế sang mã lệnh

Các chuẩn và thủ tục lập trình

- Chuẩn và thủ tục giúp các thành viên khác
  - Giúp các thành viên khác hiểu được mã lệnh (do lập trình viên viết ra) làm gì và làm như thế nào nhằm thực hiện việc:
    - Tái sử dụng (lập trình viên khác)
    - Kiểm thử (kiểm thử viên)
    - Hiệu chỉnh hay hoàn thiện hệ thống (bảo trì viên)
  - Ví dụ: Mỗi chương trình được tạo ra bởi một công ty phần mềm đều bắt đầu bằng đoạn chú thích mô tả các chức năng của chương trình và giao diện với các chương trình khác => giúp ích rất nhiều cho các thành viên khác.

Các chuẩn và thủ tục lập trình

- Chuẩn và thủ tục giúp tạo ra sự tương ứng trực tiếp giữa các thành phần thiết kế và các thành phần cài đặt
  - Các đặc trưng của chương trình nên giống như các đặc trưng của thiết kế: ghép nối thấp, gắn kết cao và các giao diện rõ ràng =>
  - Các giải thuật, chức năng, giao diện và cấu trúc dữ liệu có thể được dò theo từ thiết kế sang chương trình và ngược lại một cách dễ dàng

Chọn ngôn ngữ lập trình

- Theo loại phần mềm
  - Phần mềm hệ thống: C, C++
  - Phần mềm thời gian thực: C, C++, Assembly
  - Phần mềm nhúng: C++, Java
  - Phần mềm nghiệp vụ (HTTT):
    - CSDL: Oracle, MySQL, SQL Server
    - NNLT: VB, Foxpro, VC++
  - Phần mềm trí tuệ nhân tạo: Prolog, Lisp
  - Phần mềm (dịch vụ) Web: PHP, ASP, Java Script
- Theo đặc trưng của ngôn ngữ
- Theo năng lực và kinh nghiệm của nhóm phát triển phần mềm
- Theo yêu cầu của khách hàng

Nguyên tắc lập trình

- Dù bất cứ ngôn ngữ lập trình nào được sử dụng, mỗi thành phần chương trình đều liên quan tới ít nhất 3 khía cạnh chính sau:
  - Cấu trúc điều khiển
  - Giải thuật
  - Cấu trúc dữ liệu

Cấu trúc điều khiển

- Tạo ra mã lệnh dễ đọc

Ví dụ: Cho đoạn chương trình

```
            benefit = minimum;
            if (age < 75) goto A;
            benefit = maximum;
            goto C;
            if (AGE < 65) goto B;
            if (AGE < 55) goto C;
A:if (AGE < 65) goto B;
            benefit = benefit * 1.5 + bonus;
            goto C;
B:          if (age < 55) goto C;
            benefit = benefit * 1.5;
C:          next statement
```

Điều khiển của đoạn chương trình này bỏ qua một số câu lệnh => Rất khó đọc. Ta nên viết lại như thế nào?

Cấu trúc điều khiển

- Ví dụ: Đoạn chương trình

```
            benefit = minimum;
            if (age < 75) goto A;
            benefit = maximum;
            goto C;
            if (AGE < 65) goto B;
            if (AGE < 55) goto C;
A:          if (AGE < 65) goto B;
            benefit = benefit * 1.5 + bonus;
            goto C;
B:          if (age < 55) goto C;
            benefit = benefit * 1.5;
C:          next statement
```

Được viết lại:

```
if (age < 55) benefit = minimum;
elseif (AGE < 65) benefit = minimum + bonus;
elseif (AGE < 75) benefit = minimum * 1.5 + bonus;
else benefit = maximum;
```

Cấu trúc điều khiển

- Xây dựng chương trình từ các khối mô đun nhằm làm cho hệ thống dễ hiểu, dễ kiểm thử và dễ bảo trì
- Viết mã lệnh có tính tổng quát nhưng không làm ảnh hưởng đến sự thực hiện và sự hiểu biết
- Sử dụng các tên tham số và các chú thích để biểu thị sự kết hợp giữa các thành phần
- Tạo sự phụ thuộc giữa các thành phần một cách rõ ràng

Giải thuật

- Mục tiêu chung: tăng hiệu quả thực hiện (tốc độ)
- Tính hiệu quả trong sự thực hiện có thể có các chi phí ẩn
  - Tốn thời gian để viết mã lệnh thực thi nhanh hơn
  - Tốn thời gian để kiểm thử mã lệnh
  - Tốn thời gian để người dùng hiểu mã lệnh
  - Tốn thời gian để hiệu chỉnh mã lệnh, nếu cần

  => Cân bằng thời gian thực thi với chất lượng thiết kế, chuẩn và các yêu cầu của khách hàng

Cấu trúc dữ liệu

- Ta nên định dạng và lưu trữ dữ liệu sao cho việc quản lý và thực hiện trên dữ liệu là không phức tạp
- Các cấu trúc dữ liệu: danh sách, ngăn xếp, cây, v.v
- Một số kỹ thuật sử dụng cấu trúc dữ liệu để tổ chức chương trình
  - Làm cho chương trình đơn giản
  - Sử dụng cấu trúc dữ liệu để xác định cấu trúc chương trình

Cấu trúc dữ liệu

- Làm cho chương trình đơn giản

Ví dụ: xác định thuế thu nhập

- Với $10,000 thu nhập đầu tiên, thuế là 10%
- Với $10,000 thu nhập tiếp theo (trên $10,000), thuế là 12%
- Với $10,000 thu nhập tiếp theo (trên $20,000), thuế là 15%
- Với $10,000 thu nhập tiếp theo(trên $30,000), thuế là 18%
- Với bất cứ thu nhập nào trên $40,000, thuế là 20%

Cấu trúc dữ liệu

```
tax = 0.
if (taxable_income == 0) goto EXIT;
if (taxable_income > 10000) tax = tax + 1000;
else {
      tax = tax + .10*taxable_income;
      goto EXIT;                                    }
if (taxable_income > 20000) tax = tax + 1200;
else {
      tax = tax + .12*(taxable_income-10000):
      goto EXIT;                                    }
if (taxable_income > 30000) tax = tax + 1500;
else {
      tax = tax + .15*(taxable_income-20000);
      goto EXIT;                                    }
if (taxable_income < 40000){
      tax = tax + .18*(taxable_income-30000);
      goto EXIT;                                    }
else
      tax = tax + 1800. + .20*(taxable_income-40000);
EXIT;
```

Cấu trúc dữ liệu

- Xác định bảng thuế

Bracket	Base	Percent
0	0	10
10,000	1000	12
20,000	2200	15
30,000	3700	18
40,000	55000	20

- Giải thuật được đơn giản hóa

```
for (int i=2; level=1; i <= 5; i++)
        if (taxable_income > bracket[i])
                level = level + 1;
 tax = base[level]+percent[level]*(taxable_income-bracket[level]);
```

Cấu trúc dữ liệu

- Sử dụng cấu trúc dữ liệu để xác định cấu trúc chương trình
  - Cấu trúc dữ liệu chi phối tổ chức chương trình
  - Cấu trúc dữ liệu có thể tác động đến việc chọn ngôn ngữ
    - Nếu cấu trúc dữ liệu đệ quy, chọn ngôn ngữ cho phép lập trình các thủ tục đệ quy
    - Nếu cấu trúc dữ liệu là danh sách, chọn ngôn ngữ Lisp
    - …

Nguyên tắc chung

- Cục bộ hóa input và output
  - Các phần đọc input và sinh ra output là những phần có khả năng phải thay đổi nhiều nhất khi phần cứng và phần mềm được sửa đổi
  - Chúng được tách riêng khỏi phần còn lại của mã lệnh nhằm giúp ta dễ hiểu và dễ thay đổi hệ thống hơn
- Tái sử dụng
  - Tái sử dụng của nhà sản xuất: tạo ra các thành phần mà chúng được tái sử dụng trong các ứng dụng nối tiếp
  - Tái sử dụng của người tiêu thụ: tái sử dụng các thành phần mà ban đầu chúng được phát triển cho các dự án khác

Nguyên tắc chung

- Một số lưu ý khi lập trình
  - Bắt ngoại lệ
  - Quên cấp phát và thu hồi bộ nhớ
  - Vấn đề làm tròn số
  - Hạn chế gọi nhiều lần các hàm có qui mô nhỏ
  - Truyền tham số
  - Tránh dùng mảng nhiều chiều
  - Sử dụng các phép toán nhanh

Tài liệu chương trình

- Tài liệu chương trình: một tập các mô tả giải thích với người đọc về các chương trình làm gì và chúng làm như thế nào
- Các loại tài liệu chương trình
  - Tài liệu nội là mô tả được viết trực tiếp trong mã lệnh và được sử dụng bởi những người đọc mã lệnh
  - Tài liệu ngoại gồm những tài liệu khác và được sử dụng bởi những người không trực tiếp đọc mã lệnh

Tài liệu chương trình

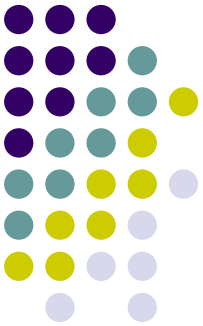
- Tài liệu nội
  - Khối chú thích ở phần đầu của từng thành phần: ai, cái gì, tại sao, khi nào, như thế nào và ở đâu
  - Các chú thích khác giúp ta hiểu chương trình rõ hơn: thêm thông tin mới, không phải diễn đạt lại những gì đã rõ ràng
  - Nhãn cho câu lệnh và tên biến phải có nghĩa
  - Định dạng để gia tăng sự hiểu biết: thụt lề, khoảng trắng

Tài liệu chương trình

- Tài liệu ngoại
  - Mô tả vấn đề: mô tả các lựa chọn được xem xét để giải quyết vấn đề và tại sao một giải pháp cụ thể được chọn
  - Mô tả giải thuật: giải thích từng giải thuật được sử dụng trong thành phần (công thức, ranh giới, các điều kiện đặc biệt, bắt lỗi, v.v)
  - Mô tả dữ liệu: từ điển dữ liệu

NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

CHƯƠNG 8 – KIỂM THỬ



Nội dung

- Phần 1 – Kiểm thử chương trình
- Phần 2 – Kiểm thử hệ thống
- Phần 3 – Ước lượng số lỗi và thời gian kiểm thử

Phần 1 - Nội dung

- Các lỗi phần mềm
- Các vấn đề trong kiểm thử
- Kiểm thử đơn vị
- Kiểm thử tích hợp
- Lập kế hoạch kiểm thử
- Các công cụ kiểm thử tự động

Lỗi phần mềm

- Các nguyên nhân làm phần mềm thất bại
  - Đặc tả sai
  - Đặc tả thiếu
  - Yêu cầu không thể thực thi
  - Thiết kế không chính xác
  - Mã lệnh có thiếu sót

Lỗi phần mềm

- Mục đích của kiểm thử: phát hiện ra các lỗi
- Một kiểm thử là thành công chỉ khi lỗi được phát hiện
  - Nhận dạng lỗi là quá trình xác định lỗi nào đã gây ra sự thất bại
  - Hiệu chỉnh lỗi là quá trình tạo ra các thay đổi đối với hệ thống nhằm loại bỏ các lỗi

Lỗi phần mềm

- Các loại lỗi
  - Lỗi giải thuật: các lỗi giải thuật điển hình
    - Lỗi giải thuật xuất hiện khi logic hay giải thuật của thành phần không tạo ra kết xuất đúng cho kết nhập xác định
      - Rẽ nhánh quá sớm
      - Rẽ nhánh quá trễ
      - Kiểm thử cho điều kiện sai
      - Quên khởi tạo giá trị ban đầu của biến hay đặt vòng lặp bất biến
      - Quên kiểm thử cho điều kiện đặc biệt
      - So sánh giá trị của các kiểu dữ liệu không phù hợp
    - Lỗi cú pháp

Lỗi phần mềm

- Các loại lỗi
  - Lỗi về độ chính xác và tính toán
    - Sự thực hiện của một công thức là sai hoặc không tính ra kết quả với độ chính xác mong muốn
  - Lỗi tài liệu
    - Tài liệu không phù hợp với cái mà chương trình làm
  - Lỗi về quá tải hay dung lượng
    - Sự thực hiện của hệ thống là không thể chấp nhận khi đạt đến các ranh giới nào đó

Lỗi phần mềm

- Các loại lỗi
  - Lỗi hợp tác và thời gian
  - Lỗi thực hiện
    - Hệ thống không thực hiện với tốc độ được mô tả
  - Lỗi quy trình và chuẩn
  - Lỗi phần cứng, phần mềm

Bài tập – Câu 1

- Cho đoạn chương trình sau

```
int n;
float x[1000];
…
x[i] = (1 - 2/(n-1)) * x[i - 1] + 2/(n-1) * x[i];
```

Đoạn chương trình này:

a. Lỗi đường biên
b. Lỗi thiếu khởi tạo
c. Lỗi về tính toán / chính xác
d. Lỗi b và c
e. Lỗi a và b
f. Không có lựa chọn nào đúng

Bài tập – Câu 2

```
List::~List(){
/* Xóa tất cả các phần tử trong danh sách */
for (int i=1; i < count; i++)
    delete list[i];
}
```

Đoạn chương trình này:

a. Lỗi thiếu khởi tạo
b. Lỗi tài liệu
c. Lỗi về tính chính xác
d. Lỗi b và c
e. Lỗi a và b
f. Không có lựa chọn nào đúng

Bài tập – Câu 3

```
int list[10];
for (int i=0; i<=10; i++) list[i] = i;
```

Đoạn chương trình này:

a. Lỗi thiếu khởi tạo

b. Lỗi về tính chính xác

c. Lỗi về dung lượng / quá tải

d. Lỗi a, b và c

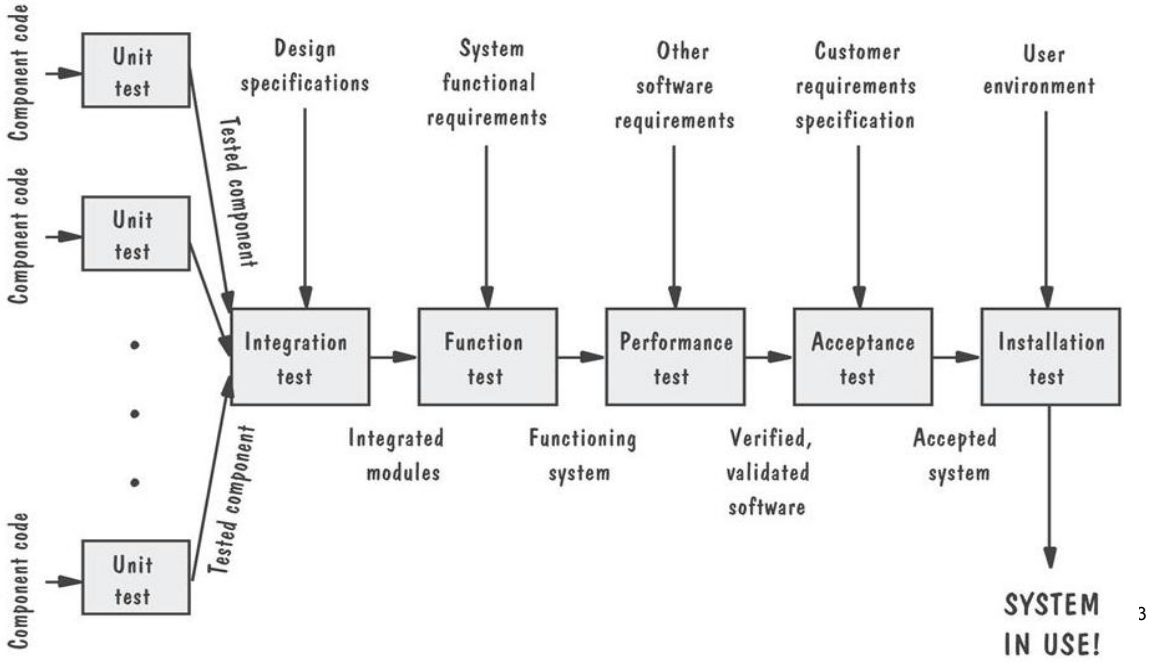
e. Không có lựa chọn nào đúng

Các vấn đề trong kiểm thử

- Tổ chức kiểm thử
  - Kiểm thử đơn vị
  - Kiểm thử tích hợp
  - Kiểm thử chức năng
  - Kiểm thử sự thực hiện
  - Kiểm thử chấp nhận
  - Kiểm thử cài đặt

Các vấn đề trong kiểm thử

- Các bước kiểm thử



Các vấn đề trong kiểm thử

- Quan điểm kiểm thử
  - Lập trình bản ngã: các chương trình được xem như những thành phần của một hệ thống lớn hơn, không như tài sản của những người đã viết ra chúng

Các vấn đề trong kiểm thử

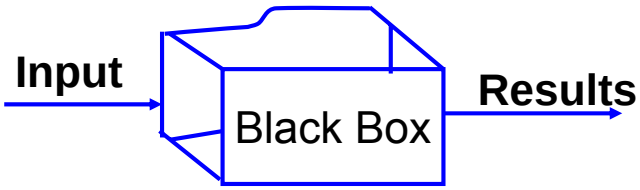
- Người kiểm thử: nhóm kiểm thử độc lập
  - Tránh sự mâu thuẫn
  - Tăng tính khách quan
  - Cho phép kiểm thử và lập trình đồng thời

Các vấn đề trong kiểm thử

- Tổng quan về các phương pháp kiểm thử
  - Kiểm thử hộp đóng hay hộp đen sử dụng chức năng của đối tượng kiểm thử để kiểm thử theo nhiều cách
  - Kiểm thử hộp mở hay hộp trắng sử dụng cấu trúc của đối tượng kiểm thử để kiểm thử theo nhiều cách

Các vấn đề trong kiểm thử

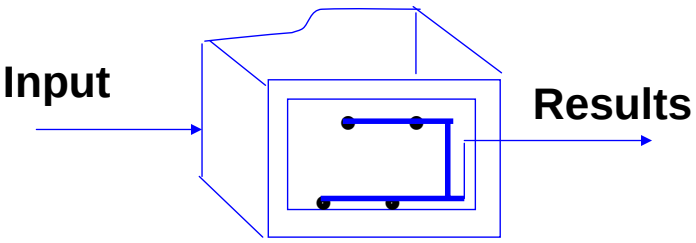
- Kiểm thử hộp đen
  - Đặc điểm
    - Đối tượng kiểm thử như một hộp đen, thông qua giao diện để đưa dữ liệu vào và nhận dữ liệu ra



  - Độc lập với các ràng buộc bị tác động bởi cấu trúc bên trong và tính logic của đối tượng

Các vấn đề trong kiểm thử

- Kiểm thử hộp trắng
  - Phương pháp thiết kế trường hợp kiểm thử sử dụng cấu trúc điều khiển của thiết kế thuật toán để dẫn ra các trường hợp kiểm thử



Các vấn đề trong kiểm thử

- Các yếu tố ảnh hưởng đến việc chọn phương pháp kiểm thử:
  - Số đường dẫn logic có thể có
  - Trạng thái của dữ liệu input
  - Số lượng tính toán có liên quan
  - Độ phức tạp của giải thuật

Kiểm thử đơn vị

- Kiểm thử đơn vị: kiểm thử từng thành phần hay từng mô đun của phần mềm.
- Các bước kiểm thử đơn vị
  - Xem lại mã lệnh
  - Biên dịch mã lệnh và loại bỏ bất cứ lỗi cú pháp nào còn lại
  - Phát triển các trường hợp kiểm thử để chỉ ra rằng input được chuyển đổi chính xác thành ouput mong đợi

Kiểm thử đơn vị

- Xem lại (kiểm tra) mã lệnh
  - Đi qua mã lệnh (Code walkthrough)
  - Xem xét kỹ mã lệnh (Code inspection)

Kiểm thử đơn vị

- Các bước chọn trường hợp kiểm thử
  - Xác định các mục tiêu kiểm thử
  - Lựa chọn các trường hợp kiểm thử
  - Định nghĩa một kiểm thử

Kiểm thử đơn vị

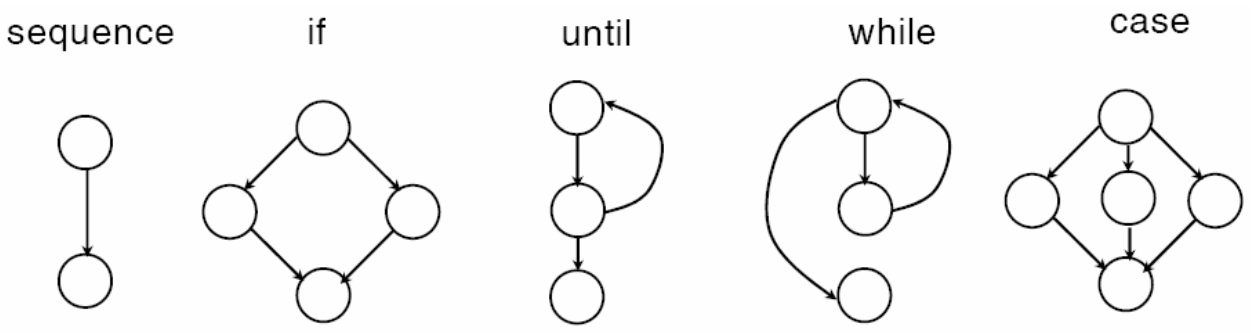
- Tính toàn diện của kiểm thử
  - Kiểm thử câu lệnh
  - Kiểm thử rẽ nhánh
  - Kiểm thử đường đi
  - Kiểm thử sử dụng sự định nghĩa
  - Kiểm thử tất cả sử dụng
  - Kiểm thử sử dụng tất cả vị từ / kiểm thử sử dụng một số tính toán
  - Kiểm thử sử dụng một số vị từ / kiểm thử sử dụng tất cả tính toán

Kiểm thử đơn vị - Kiểm thử đường đi

- Kiểm thử đường đi: Kiểm thử tất cả các hướng đi có thể
  - Thiết lập đồ thị dòng chảy
  - Liệt kê các đường thực thi độc lập cơ bản
  - Sinh các trường hợp kiểm thử cho các đường thực thi đó

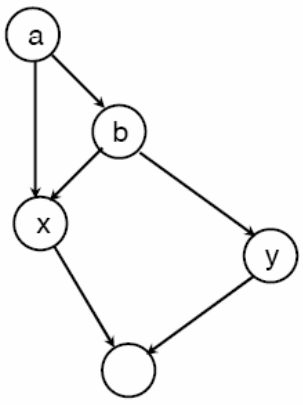
Kiểm thử đường đi

- Thiết lập đồ thị dòng chảy
  - Mỗi nút hình tròn biểu diễn một hoặc một vài tác vụ
  - Cạnh có hướng miêu tả đường thực thi
  - Đồ thị dòng chảy được xây dựng từ lưu đồ thuật giải

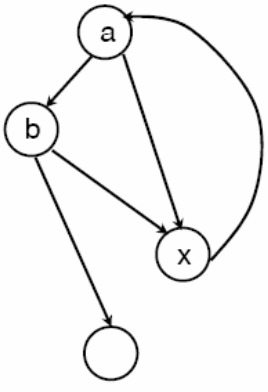


Kiểm thử đường đi

- Thiết lập đồ thị dòng chảy
  - Phải phân rã tất cả các điều kiện phức trở thành các điều kiện đơn



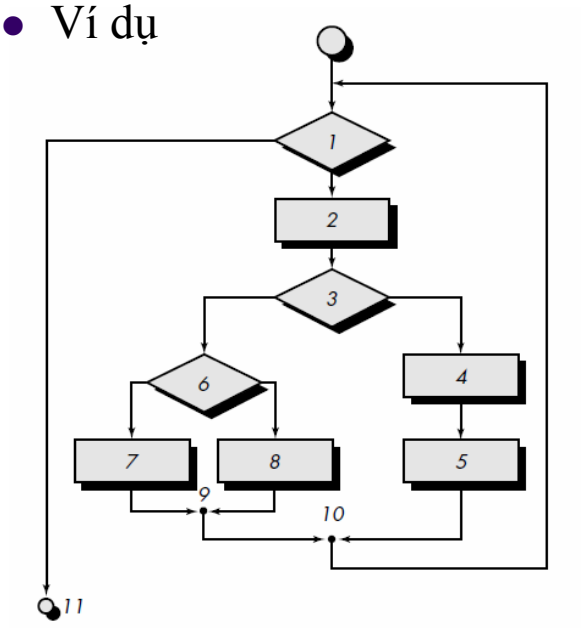
`if a and b then y else x`



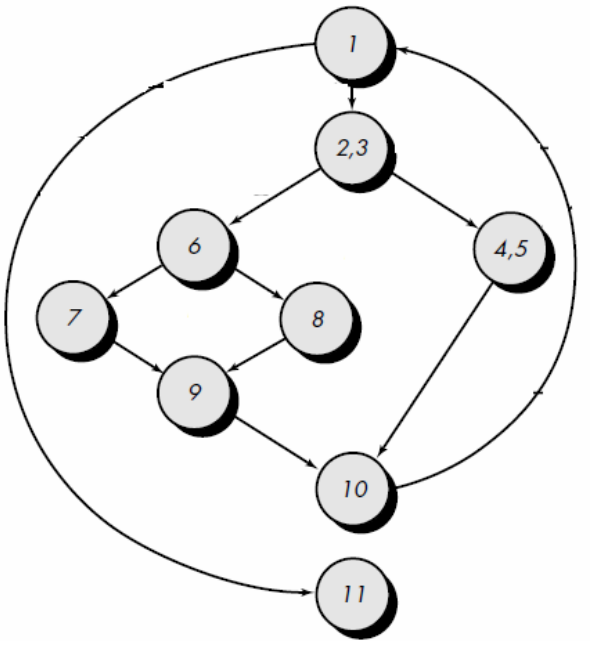
`while a or b do x`

Kiểm thử đường đi

- Ví dụ



Lưu đồ giải thuật



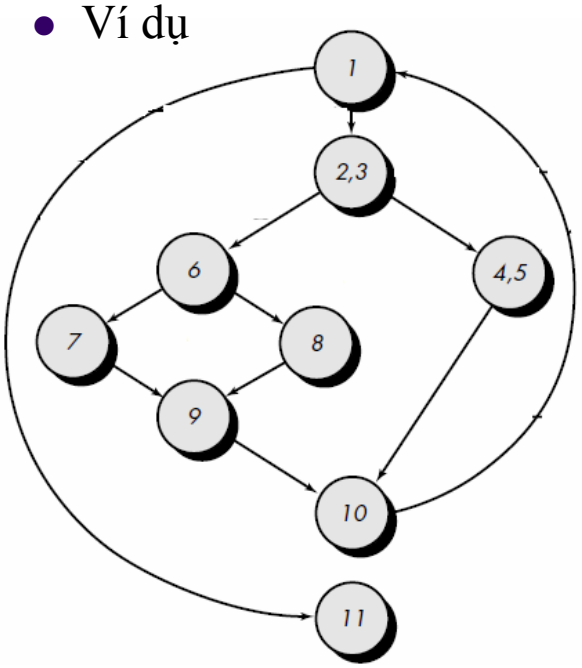
Đồ thị dòng chảy

Kiểm thử đường đi

- Liệt kê các đường độc lập cơ bản
  - Từ nút bắt đầu đến nút kết thúc, các đường thực thi cơ bản được liệt kê theo một thứ tự nào đó để đảm bảo rằng đường đang liệt kê ít nhất đi qua một cạnh chưa được duyệt qua bởi các đường đã liệt kê trước đó
  - Tổng số đường thực thi cơ bản độc lập nhau được tính bằng **V = P + 1** trong đó P là số nút phân nhánh

Kiểm thử đường đi

- Ví dụ



Đồ thị dòng chảy

Các đường độc lập cơ bản:

1. 1 - 11
2. 1 – 2,3 – 4,5 – 10 – 1 - 11
3. 1 – 2,3 – 6 – 7 – 9 – 10 – 1 - 11
4. 1 – 2,3 – 6 – 8 – 9 – 10 – 1 -11

Kiểm thử đường đi

- Sinh ra các trường hợp kiểm thử (test-case)
  - Thiết lập một test-case cho mỗi đường thực thi cơ bản
  - Dựa vào giải thuật để
    - Đưa ra dữ liệu input
    - Tính ra dữ liệu output hay đáp ứng mong đợi của giải thuật
  - Có thể không tạo ra được test-case cho một đường thực thi nào đó

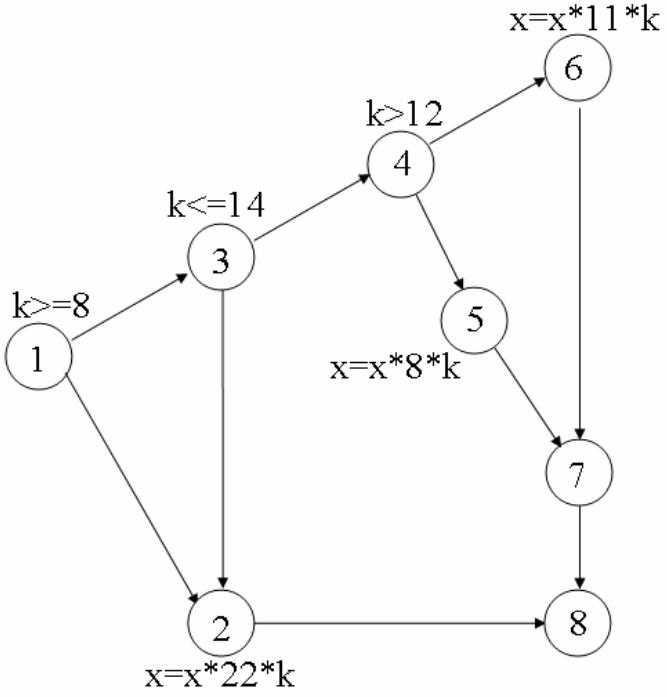
Kiểm thử đường đi

- Ví dụ: Cho đoạn chương trình sau, hãy xác định các trường hợp kiểm thử?

```
if (k>=8 && k <=14)
{
        if (k>12)
                x=x*11*k;
        else
                x=x*8*k;
}
else
        x=x*22*k;
```


Kiểm thử đường đi

```
if (k>=8 && k <=14)
{
    if (k>12)
        x=x*11*k;
    else
        x=x*8*k;
}
else
    x=x*22*k;
```



Kiểm thử đường đi

- Các đường độc lập cơ bản:
  - 1 – 2 – 8
  - 1 – 3 – 2 – 8
  - 1 – 3 – 4 – 5 – 7 – 8
  - 1 – 3 – 4 – 6 – 7 – 8

Kiểm thử đường đi

- Các trường hợp kiểm thử kiểm thử:
  - Đường 1: 1 – 2 – 8
    - Input: k=6; x=3
    - Output mong đợi: x=396
  - Đường 2: 1 – 3 – 2 – 8
    - Input: k=15; x=2
    - Output mong đợi: x=660
  - Đường 3: 1 – 3 – 4 – 5 – 7 – 8
    - Input: k=9; x=1
    - Output mong đợi: x=99
  - Đường 4: 1 – 3 – 4 – 6 – 7 – 8
    - Input: k=13; x=2
    - Output mong đợi: x=208

Kiểm thử tích hợp

- Kiểm thử tích hợp nhằm phát hiện các lỗi liên quan đến sự giao tiếp giữa các thành phần
- Các dạng kiểm thử tích hợp
  - Kiểm thử từ dưới lên
  - Kiểm thử từ trên xuống
  - Kiểm thử từ trên xuống được hiệu chỉnh
  - Kiểm thử sandwich (kẹp)
  - Kiểm thử sandwich được hiệu chỉnh
  - Kiểm thử Big-bang

Tích hợp dần

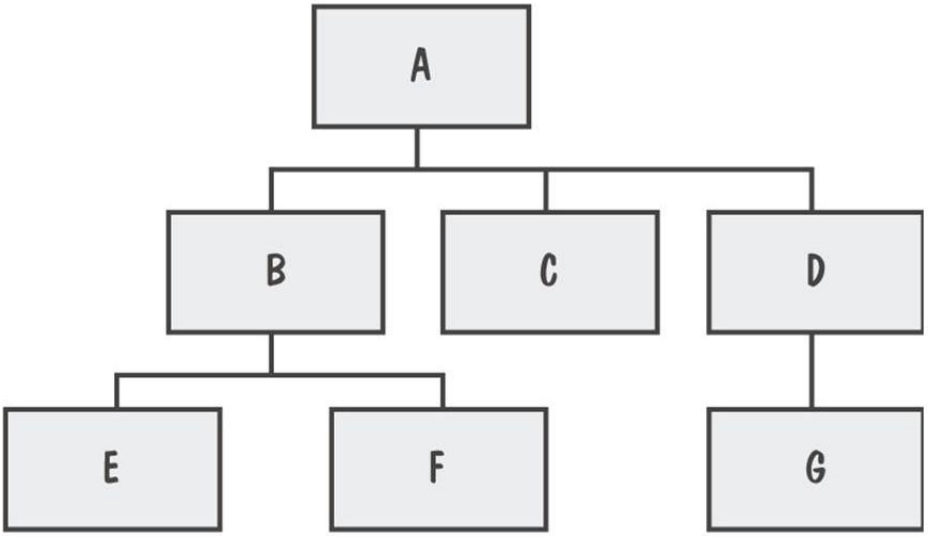
Tích hợp đồng thời một lúc

Kiểm thử tích hợp

- Thuật ngữ
  - Driver: một chương trình mà nó gọi một thành phần (component) cụ thể và truyền một trường hợp kiểm thử cho nó
  - Stub: một chương trình (có mục đích đặc biệt để) mô phỏng hoạt động của thành phần bị thiếu

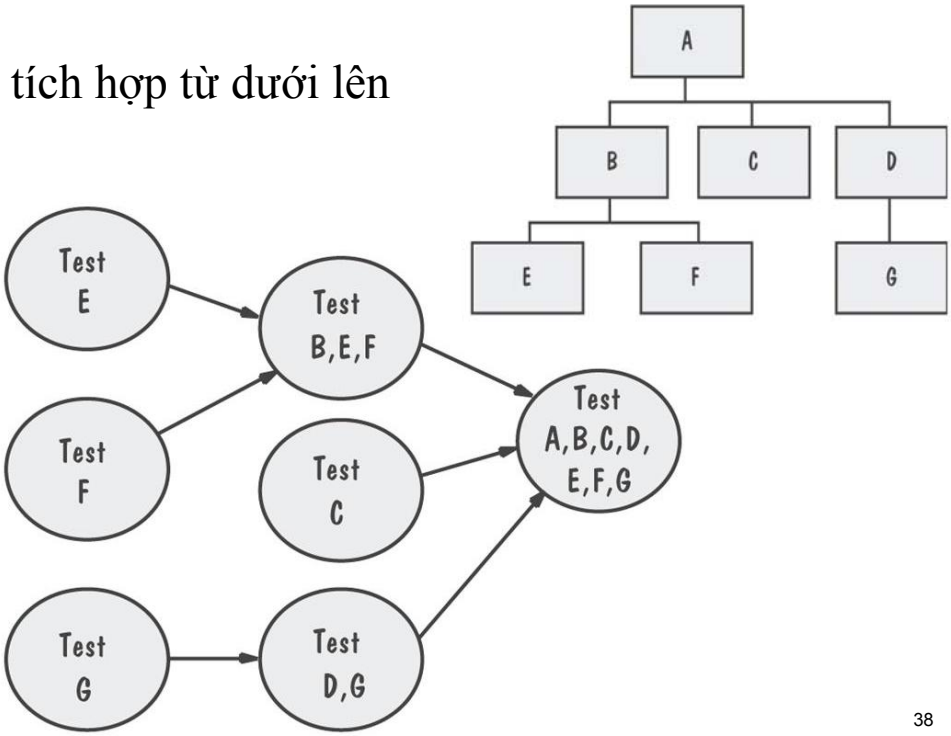
Kiểm thử tích hợp

- Ví dụ: Cho một hệ thống với các thành phần được phân cấp



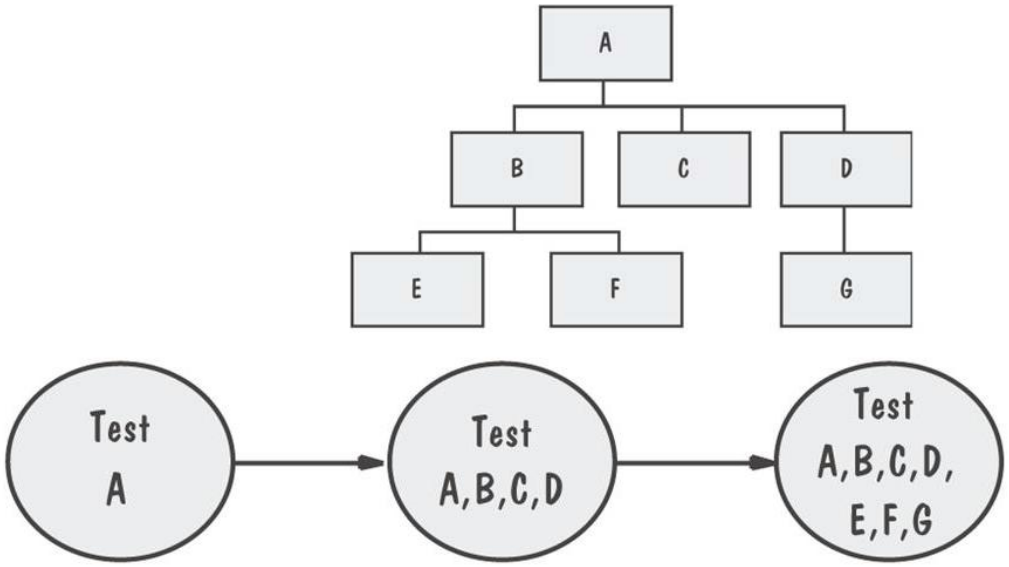
Kiểm thử tích hợp

- Kiểm thử tích hợp từ dưới lên



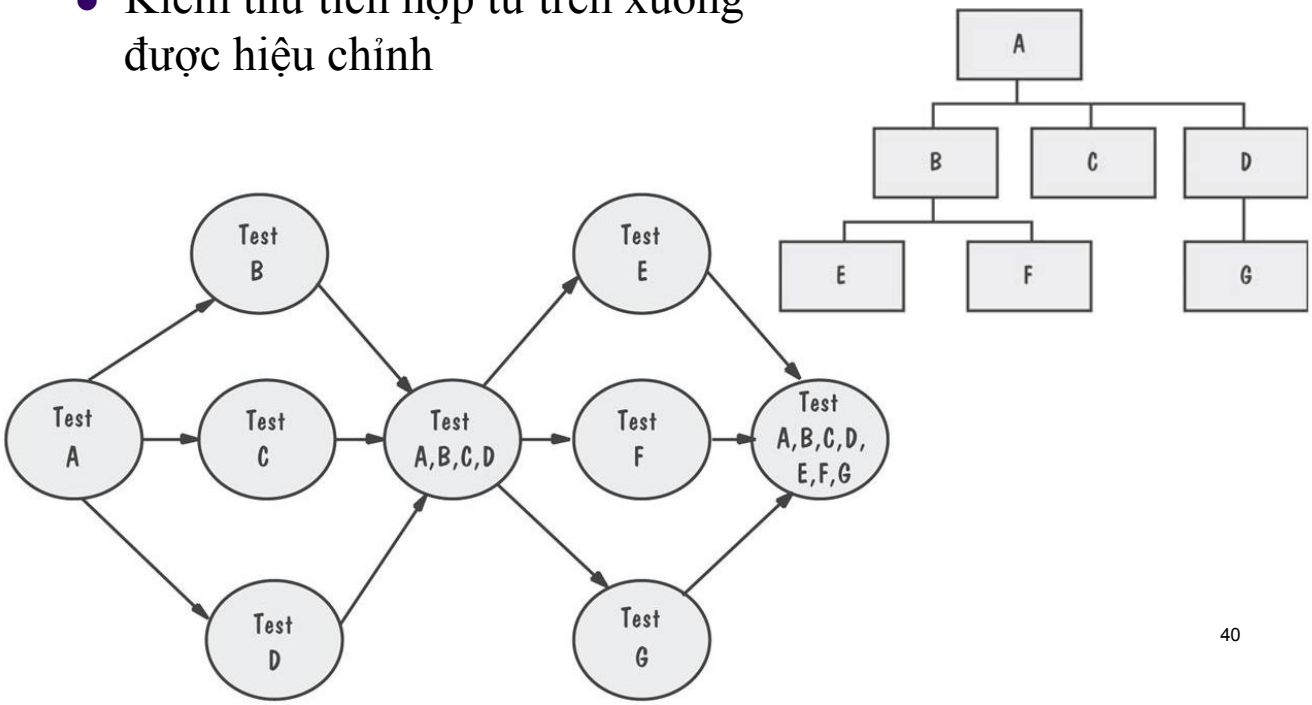
Kiểm thử tích hợp

- Kiểm thử tích hợp từ trên xuống



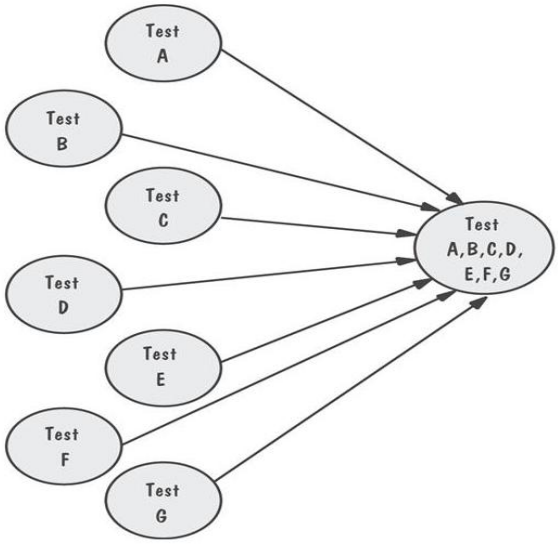
Kiểm thử tích hợp

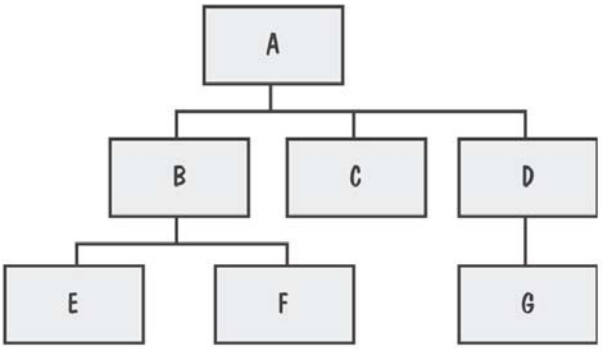
- Kiểm thử tích hợp từ trên xuống được hiệu chỉnh



Kiểm thử tích hợp

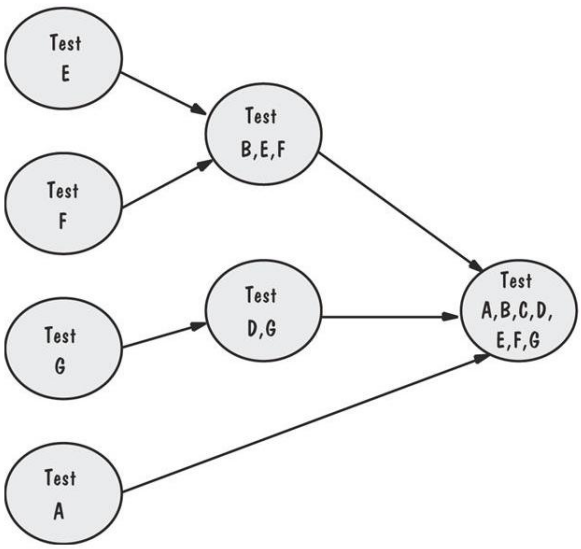
- Kiểm thử tích hợp Big Bang

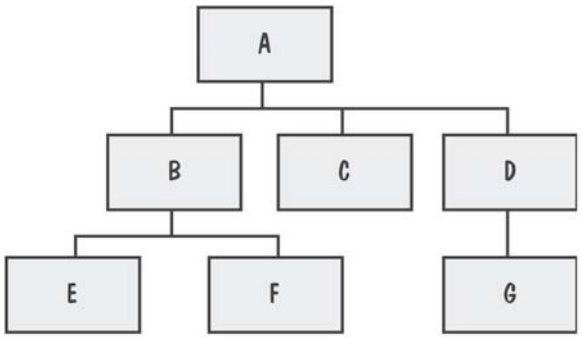




Kiểm thử tích hợp

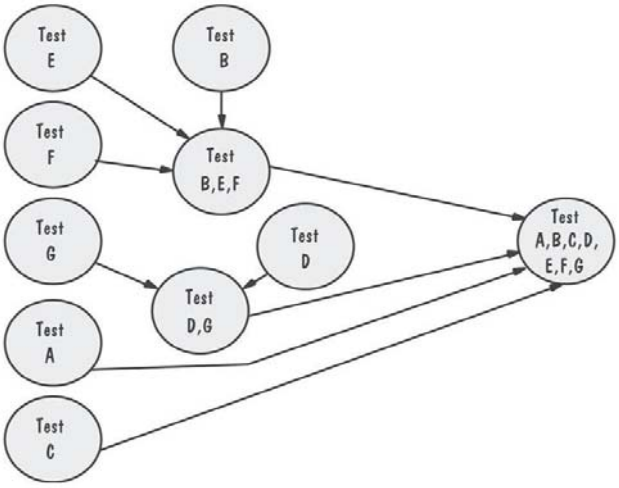
- Kiểm thử tích hợp sandwich

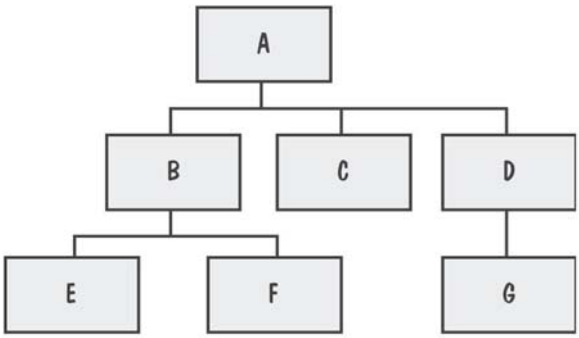




Kiểm thử tích hợp

- Kiểm thử tích hợp sandwich được hiệu chỉnh





Kiểm thử tích hợp

- So sánh các dạng kiểm thử tích hợp

	Từ dưới lên	Từ trên xuống	Từ trên xuống được hiệu chỉnh	Big-bang	Sandwich	Sandwich được hiệu chỉnh
Sự tích hợp	Early	Early	Early	Late	Early	Early
Thời gian cho lập trình công việc cơ bản	Late	Early	Early	Late	Early	Early
Các driver thành phần được cần đến	Yes	No	Yes	Yes	Yes	Yes
Các Stub được cần	No	Yes	Yes	Yes	Yes	Yes
Quan hệ song song của công việc lúc bắt đầu	Medium	Low	Medium	High	Medium	High
Khả năng kiểm thử các đường dẫn cụ thể	Easy	Hard	Easy	Easy	Medium	Easy
Khả năng lập kế hoạch và kiểm soát chuỗi sự kiện	Easy	Hard	Hard	Easy	Hard	hard

Lập kế hoạch kiểm thử

- Thiết lập các mục tiêu kiểm thử
- Thiết kế các trường hợp kiểm thử
- Viết các trường hợp kiểm thử
- Kiểm tra các trường hợp kiểm thử
- Thực thi các kiểm thử
- Đánh giá các kết quả kiểm thử

Lập kế hoạch kiểm thử

- Mục đích của kế hoạch

  Kế hoạch kiểm thử giải thích:
  - Ai thực hiện kiểm thử
  - Tại sao các kiểm thử được thực hiện
  - Cách thức các kiểm thử được kiểm soát
  - Khi nào các kiểm thử được thực hiện

Lập kế hoạch kiểm thử

- Nội dung của kế hoạch:
  - Các mục tiêu của kiểm thử
  - Cách thực hiện kiểm thử
  - Chuẩn được sử dụng để xác định khi nào kiểm thử hoàn thành

Công cụ kiểm thử tự động

- Công cụ phân tích mã lệnh
  - Phân tích tĩnh
    - Bộ phân tích mã lệnh
    - Bộ kiểm tra cấu trúc
    - Bộ phân tích dữ liệu
    - Bộ kiểm tra trình tự
  - Phân tích động
    - Các bộ giám sát chương trình: xem và báo cáo lại các hoạt động của chương trình

Công cụ kiểm thử tự động

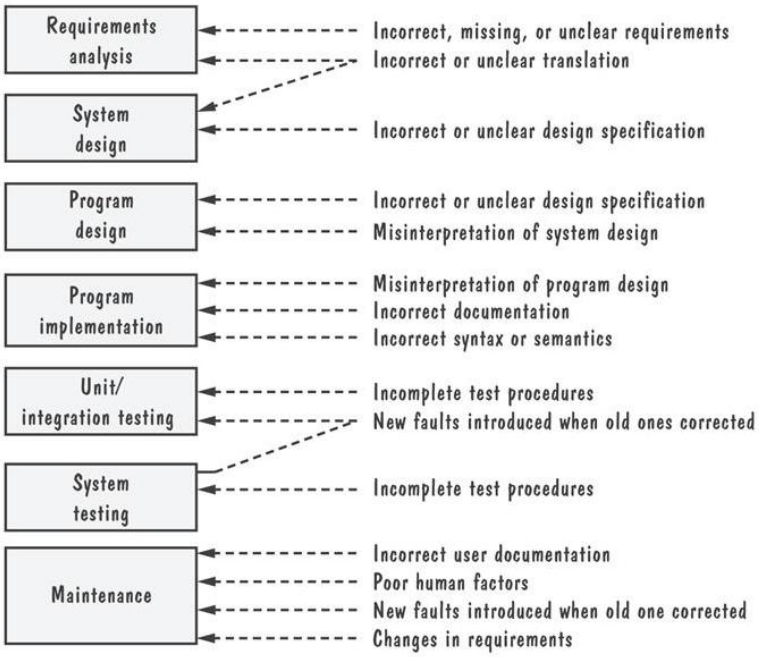
- Công cụ thực hiện kiểm thử
  - Capture và replay
  - Các stub và driver
  - Môi trường kiểm thử tự động
- Bộ sinh các trường hợp kiểm thử

Phần 2 - Nội dung

- Các nguyên lý của kiểm thử hệ thống
- Kiểm thử chức năng
- Kiểm thử sự thực thi
- Tính tin cậy, tính sẵn có và tính có thể bảo trì
- Kiểm thử chấp nhận
- Kiểm thử sự cài đặt
- Tài liệu kiểm thử

Các nguyên lý của kiểm thử hệ thống

- Nguồn gốc của các lỗi phần mềm trong suốt quá trình phát triển

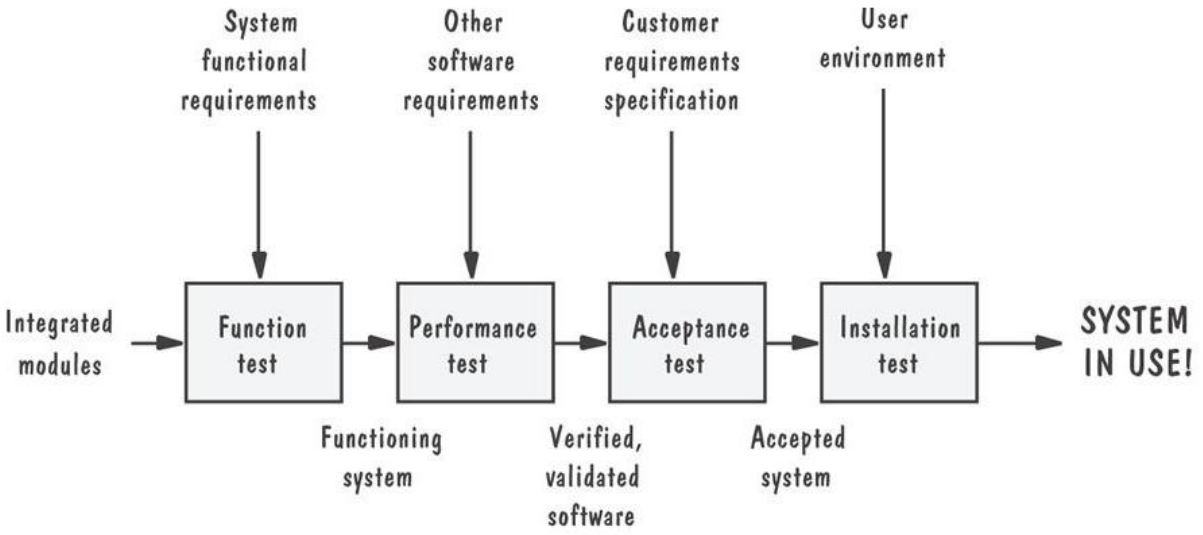


Các nguyên lý của kiểm thử hệ thống

- Quy trình kiểm thử hệ thống
  - Kiểm thử chức năng: hệ thống được tích hợp có thực hiện như được cam kết trong đặc tả yêu cầu?
  - Kiểm thử thực hiện: các yếu tố phi chức năng có được đáp ứng?
  - Kiểm thử chấp nhận: hệ thống có phải là cái mà khách hàng mong đợi?
  - Kiểm thử sự cài đặt: hệ thống có vận hành ở chỗ khách hàng không?

Các nguyên lý của kiểm thử hệ thống

- Các bước trong quy trình kiểm thử hệ thống



Các nguyên lý của kiểm thử hệ thống

- Những kỹ thuật được sử dụng trong kiểm thử hệ thống
  - Kế hoạch tích hợp hay xây dựng
  - Kiểm thử hồi quy
  - Quản lý cấu hình

Các nguyên lý của kiểm thử hệ thống

- Kế hoạch tích hợp hay xây dựng
  - Mô tả các hệ thống con (spin) được kiểm thử
  - Mô tả cách thức, nơi chốn, thời gian và người thực hiện các kiểm thử

Các nguyên lý của kiểm thử hệ thống

- Ví dụ về kế hoạch xây dựng cho hệ thống viễn thông

Spin	Chức năng	Bắt đầu kiểm thử	Kết thúc kiểm thử
0	Exchange	1 September	15 September
1	Area code	30 September	15 October
2	State/province/district	25 October	5 November
3	Country	10 November	20 November
4	International	1 December	15 December

Các nguyên lý của kiểm thử hệ thống

- Kiểm thử hồi quy
  - Nhận dạng các lỗi mới mà chúng được gây ra bởi sự hiệu chỉnh các lỗi hiện tại
  - Kiểm tra phiên bản hay phát hành mới để xác nhận rằng nó vẫn thực hiện cùng các chức năng theo cùng cách như phiên bản hay phát hành cũ

Các nguyên lý của kiểm thử hệ thống

- Các bước kiểm thử hồi quy
  - Chèn vào mã lệnh mới
  - Kiểm thử các chức năng sẽ bị ảnh hưởng bởi mã lệnh mới
  - Kiểm thử các chức năng cần thiết của phiên bản m để xác nhận rằng chúng vẫn hoạt động chính xác
  - Tiếp tục kiểm thử chức năng của phiên bản m + 1

Các nguyên lý của kiểm thử hệ thống

- Nhóm kiểm thử
  - Kiểm thử viên chuyên nghiệp: tổ chức và thực hiện các kiểm thử
  - Nhà phân tích: người đã tạo ra các đặc tả
  - Nhà thiết kế hệ thống: hiểu giải pháp được đề nghị
  - Chuyên gia quản lý cấu hình: giúp kiểm soát các sửa chữa
  - Người dùng: đánh giá các phát hành

Kiểm thử chức năng

- Mục đích và vai trò
  - So sánh sự thực hiện thực tế của hệ thống với các yêu cầu của nó
  - Phát triển các trường hợp kiểm thử dựa trên tài liệu yêu cầu

Kiểm thử chức năng

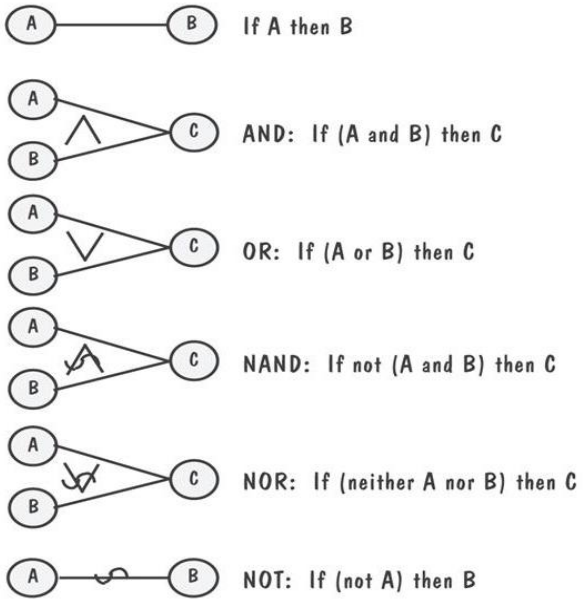
- Tạo ra các trường hợp kiểm thử chức năng
  - Vẽ đồ thị nhân quả từ các yêu cầu
  - Chuyển đồ thị thành bảng quyết định
  - Mỗi cột trong bảng quyết định tương ứng với một trường hợp kiểm thử chức năng

Kiểm thử chức năng

- Đồ thị nhân quả
  - Một đồ thị logic phản ánh quan hệ logic giữa nguyên nhân (các input) và kết quả (output hay sự biến đổi)
  - Vẽ đồ thị nhân quả từ các yêu cầu:
    - Các yêu cầu được phân tách sao cho mỗi yêu cầu mô tả một chức năng
    - Mô tả các nguyên nhân, các kết quả (đó là các nút trong đồ thị)
    - Vẽ mối quan hệ giữa chúng

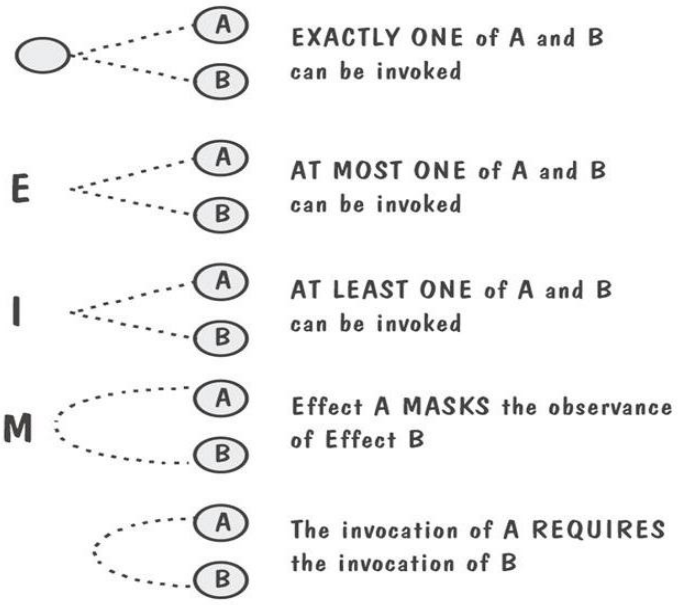
Kiểm thử chức năng

- Ký hiệu của đồ thị nhân quả



Kiểm thử chức năng

- Ký hiệu bổ sung của đồ thị nhân quả



Kiểm thử sự thực thi

- Mục đích và vai trò
  - Được sử dụng để kiểm tra
    - Sự tính toán
    - Tốc độ đáp ứng
    - Độ chính xác của kết quả
    - Khả năng truy cập dữ liệu
  - Được thiết kế và quản lý bởi nhóm kiểm thử

Kiểm thử sự thực thi

- Các loại kiểm thử sự thực thi
  - Kiểm thử ứng suất
  - Kiểm thử dung lượng CTDL
  - Kiểm thử cấu hình
  - Kiểm thử tính tương thích
  - Kiểm thử hồi quy
  - Kiểm thử sự bảo mật
  - Kiểm thử sự điều hòa thời gian
  - Kiểm thử môi trường
  - Kiểm thử chất lượng
  - Kiểm thử sự hồi phục
  - Kiểm thử bảo trì
  - Kiểm thử tài liệu
  - Kiểm thử tính dễ sử dụng đ/v con người

Tính tin cậy, tính sẵn có và tính có thể bảo trì

- Định nghĩa
  - Tính tin cậy của phần mềm: vận hành mà không có lỗi dưới một điều kiện xác định trong một khoảng thời gian cho trước
  - Tính sẵn có của phần mềm: vận hành thành công theo sự đặc tả tại một điểm thời gian xác định
  - Tính có thể bảo trì của phần mềm: với một điều kiện sử dụng xác định, một hoạt động bảo trì có thể được thực hiện trong khoảng thời gian, thủ tục và tài nguyên xác định

Tính tin cậy, tính sẵn có và tính có thể bảo trì

- Các mức độ khác nhau về tính khốc liệt của lỗi
  - Thảm khốc: gây ra sự đổ vỡ hoặc sự mất dữ liệu
  - Then chốt: gây ra tổn thương rất xấu hay sự thiệt hại hệ thống chính
  - Lề: gây ra tổn thương phụ hay sự thiệt hại hệ thống phụ
  - Thứ yếu: không gây ra tổn thương hay sự thiệt hại hệ thống

Tính tin cậy, tính sẵn có và tính có thể bảo trì

- Đo lường tính tin cậy, tính sẵn có và tính có thể bảo trì
  - Thời gian trung bình đối với sự thất bại (Mean time to failure - MTTF)
  - Thời gian trung bình để sửa một thành phần phần mềm bị lỗi (Mean time to repair - MTTR)
  - Thời gian trung bình giữa các lần thất bại (Mean time between failures - MTBF)
    - MTBF = MTTF + MTTR
  - Tính tin cậy
    - R = MTTF/(1+MTTF)
  - Tính sẵn có
    - A = MTBF (1+MTBF)
  - Tính có thể bảo trì
    - M = 1/(1+MTTR)

Kiểm thử chấp nhận

- Mục đích và vai trò
  - Cho phép khách hàng và người dùng xác định xem hệ thống được xây dựng có đáp ứng được yêu cầu và sự mong đợi của họ hay không
  - Được viết, quản lý và đánh giá bởi khách hàng

Kiểm thử chấp nhận

- Các loại kiểm thử chấp nhận
  - Kiểm thử thử nghiệm (pilot): cài đặt hệ thống trên cơ sở thí nghiệm
    - Kiểm thử alpha: kiểm thử của người dùng trong tổ chức hay công ty phát triển phần mềm
    - Kiểm thử beta: kiểm thử của khách hàng
  - Kiểm thử song song: một hệ thống mới vận hành song song với hệ thống cũ

Kiểm thử sự cài đặt

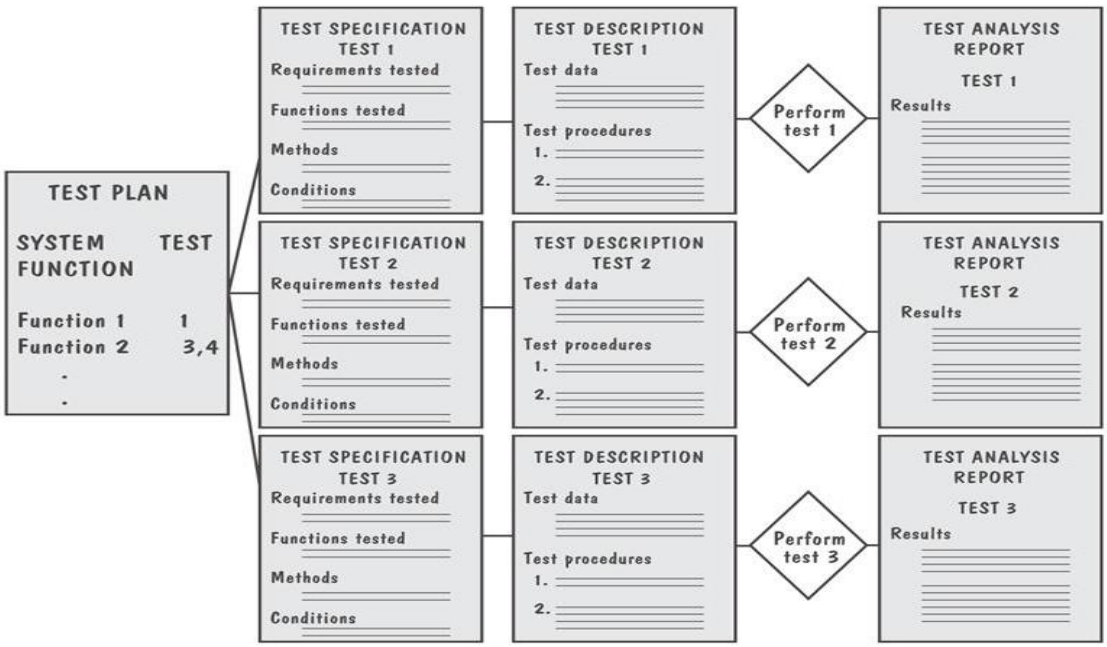
- Trước khi kiểm thử
  - Cấu hình hệ thống
  - Gắn số và loại thiết bị
  - Thiết lập sự giao tiếp với hệ thống khác
- Kiểm thử
  - Kiểm thử hồi quy: kiểm tra rằng hệ thống được cài đặt một cách chính xác và hoạt động

Tài liệu kiểm thử

- Kế hoạch kiểm thử: mô tả hệ thống và kế hoạch sử dụng tất cả các chức năng và các đặc trưng
- Đặc tả và đánh giá kiểm thử: chi tiết từng kiểm thử và định nghĩa tiêu chuẩn để đánh giá từng đặc điểm
- Mô tả kiểm thử: thủ tục và dữ liệu kiểm thử cho từng kiểm thử
- Báo cáo phân tích kiểm thử: các kết quả của từng kiểm thử

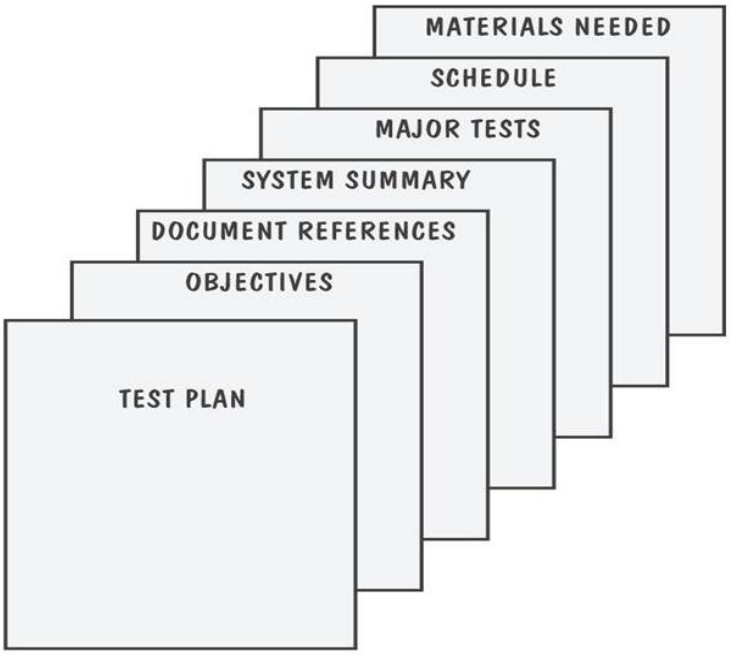
Tài liệu kiểm thử

- Các tài liệu được tạo ra trong suốt sự kiểm thử



Tài liệu kiểm thử

- Các phần của một kế hoạch kiểm thử



Tài liệu kiểm thử

- Kế hoạch kiểm thử: kế hoạch bắt đầu bằng việc nói rõ các mục tiêu, chúng nên:
  - Hướng dẫn việc quản lý kiểm thử
  - Hướng dẫn công sức kỹ thuật cần trong suốt sự kiểm thử
  - Thiết lập kế hoạch và lịch biểu kiểm thử
  - Giải thích bản chất và sự mở rộng của từng kiểm thử
  - Giải thích cách thức kiểm thử sẽ đánh giá đầy đủ chức năng và sự thực hiện của hệ thống
  - Ghi thành tài liệu các input của kiểm thử, các thủ tục kiểm thử cụ thể, các kết quả mong đợi

Tài liệu kiểm thử

- Sự mô tả kiểm thử gồm:
  - Phương tiện kiểm soát
  - Dữ liệu
  - Thủ tục

Tài liệu kiểm thử

- Báo cáo phân tích kiểm thử
  - Ghi vào tài liệu kết quả kiểm thử
  - Cung cấp thông tin cần để sao chép lại sự thất bại, định vị và sửa tận gốc vấn đề
  - Cung cấp thông tin cần thiết để xác định xem dự án có hoàn thành hay không
  - Thiết lập sự tin cậy trong sự thực thi của hệ thống

Tài liệu kiểm thử

- Biểu mẫu báo cáo vấn đề
  - Vị trí: vấn đề đã xuất hiện ở đâu?
  - Thời gian: nó đã xuất hiện khi nào?
  - Dấu hiệu: cái được quan sát?
  - Kết quả cuối cùng: các hệ quả?
  - Kỹ thuật: nó đã xuất hiện như thế nào?
  - Nguyên nhân: tại sao nó xuất hiện?
  - Tính khốc liệt: mức độ mà người dùng hay doanh nghiệp bị ảnh hưởng?
  - Chi phí: nó tốn bao nhiêu?

Tài liệu kiểm thử

- Ví dụ về biểu mẫu báo cáo vấn đề trong thực tế

Fault Number	Week In	System Area	Fault Type	Week Out	Hours to Repair
...	...	...	...	...	...
F254	92/14	C2	P	92/17	5.5

Tài liệu kiểm thử

- Ví dụ về biểu mẫu báo cáo vấn đề trong thực tế

FAULT REPORT S.P0204.6.10.3016

ORIGINATOR:	Joe Bloggs
BRIEF TITLE:	Exception 1 in dps_c.c line 620 raised by NAS
FULL DESCRIPTION	Started NAS endurance and allowed it to run for a few minutes. Disabled the active NAS link (emulator switched to standby link), then re-enabled the disabled link and CDIS exceptioned as above. (I think the re-enabling is a red herring.) (during database load)
ASSIGNED FOR EVALUATION TO:	DATE:

CATEGORISATION: 0 (1) 2 3 Design Spec Docn
SEND COPIES FOR INFORMATION TO:
EVALUATOR: DATE: 8/7/92

CONFIGURATION ID	ASSIGNED TO	PART
dpo_s.c		

COMMENTS: dpo_s.c appears to try to use an invalid CID, instead of rejecting the message. AWJ

ITEMS CHANGED

CONFIGURATION ID	IMPLEMENTOR/DATE	REVIEWER/DATE	BUILD/ISSUE NUM	INTEGRATOR/DATE
dpo_s.c v.10	AWJ 8/7/92	MAR 8/7/92	6.120	RA 8-7-92

COMMENTS:

CLOSED

FAULT CONTROLLER: DATE: 9/7/92

Phần 3

- Ước lượng số lỗi
- Ước lượng thời gian kiểm thử

Ước lượng số lỗi

- Công thức dự đoán số lỗi của Halstead

  $((N_1+N_2)\log_2(n_1+n_2))/3000$ (/KDSI)

  với
  - N_1: Số toán tử
  - N_2: Số toán hạng
  - n_1: Số toán tử khác nhau
  - n_2: Số toán hạng khác nhau

Ví dụ

- Cho đoạn chương trình sắp xếp sau:

```
1     procedure sort(var x: array; n: integer);
2     var i, j, save: integer;
3     begin
4         for i:= 2 to n do
5             for j:= 1 to i do
6                 if x[i] < x[j] then
7                 begin save:= x[i];
8                     x[i]:= x[j];
9                     x[j]:= save
10                end
11    end;
```

Ví dụ

Operator	Number of occurrences
procedure	1
sort()	1
var	2
:	3
array	1
;	6
integer	2
,	2
begin . . . end	2
for . . . do	2
if . . . then	1
:=	5
<	1
[ ]	6
$n_1 = 14$	$N_1 = 35$

Operand	Number of occurrences
x	7
n	2
i	6
j	5
save	3
2	1
1	1
$n_2 = 7$	$N_2 = 25$

Ví dụ

- Dự đoán số lỗi

  $((N_1+N_2)\log_2(n_1+n_2))/3000 =$

  $((35+25)\log_2(14+7))/3000 = 60\log_2 21 =$

  0.0046 (/KDSI)

Ước lượng thời gian kiểm thử

- Công thức (Brettschneider) xác định thời gian kiểm thử

$$\frac{\ln\left(\frac{f_{target}}{0.5 + f_{target}}\right)}{\ln\left(\frac{0.5 + f_{target}}{f_{total} + f_{target}}\right)} \times t_h$$

với

- f*target* : Số lượng lỗi dự đoán
- f*total*: Số lỗi thực sự xảy ra sau đó
- t*h*: Thời gian kiểm thử xảy ra lỗi

Ví dụ

- Giả sử sản phẩm có 50000 LOC, hợp đồng qui định mỗi KDSI có ít hơn 0.02 lỗi. Sản phẩm được kiểm thử 400 giờ, trong thời gian này có 20 lỗi xảy ra và đã thực thi 50 giờ kể từ lỗi cuối cùng. Xác định thời gian cần kiểm thêm?

Ví dụ

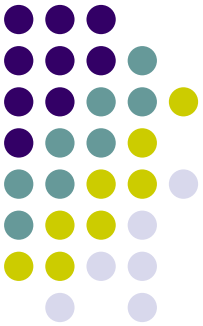
- Giả sử sản phẩm có 50000 LOC, hợp đồng qui định mỗi KDSI có ít hơn 0.02 lỗi. Sản phẩm được kiểm thử 400 giờ, trong thời gian này có 20 lỗi xảy ra và đã thực thi 50 giờ kể từ lỗi cuối cùng.
  - f*target* = 0.02 x 50 = 1,
  - f*total* = 20,
  - t*h* = 400 - 50 =350 giờ

$$\frac{\ln\left(\frac{f_{target}}{0.5 + f_{target}}\right)}{\ln\left(\frac{0.5 + f_{target}}{f_{total} + f_{target}}\right)} \times t_h$$ = (ln(1/1.5)/ln(1.5/21)) x 350 = 54 giờ

=> Phải kiểm thử thêm 4 giờ nữa

NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

CHƯƠNG 9– TRIỂN KHAI



Nội dung

- Huấn luyện
- Tài liệu

Triển khai hệ thống

- Triển khai hệ thống:
  - Không chỉ đặt hệ thống vào vị trí
  - Còn giúp người dùng hiểu và cảm thấy thoải mái với hệ thống
    - Huấn luyện
    - Tài liệu

Huấn luyện

- Phân loại người sử dụng hệ thống
  - Người dùng: sử dụng các chức năng của hệ thống chính
  - Điều hành viên: thực hiện các chức năng bổ sung
    - Tạo ra các bản sao dự phòng của các tập tin dữ liệu
    - Xác định những người truy xuất vào hệ thống

Huấn luyện

- Các chức năng của người dùng và điều hành viên

Chức năng của người dùng	Chức năng của điều hành viên
Thao tác trên các tập tin dữ liệu	Chấp nhận truy xuất của người dùng
Kích thích các hoạt động	Chấp nhận truy xuất tập tin
Phân tích dữ liệu	Thực hiện sao lưu
Giao tiếp dữ liệu	Cài đặt các thiết bị mới
Vẽ đồ thị và biểu đồ	Cài đặt phần mềm mới
	Phục hồi các tập tin hư

Huấn luyện

- Các kiểu huấn luyện
  - Huấn luyện người dùng
  - Huấn luyện điều hành viên
  - Yêu cầu huấn luyện đặc biệt

Huấn luyện

- Huấn luyện người dùng
  - Giới thiệu các chức năng cơ bản
    - Quản lý hồ sơ: tạo, xóa, truy lục, sắp xếp hồ sơ
    - Sự điều hướng qua hệ thống
    - Không cần kỹ thuật bên trong (giải thuật, cấu trúc dữ liệu, …)
  - Liên hệ cách các chức năng được thực hiện hiện tại, cách thực hiện sau đó bằng hệ thống mới
    - Cần tính đến khó khăn của học hỏi chuyển tiếp

Huấn luyện

- Huấn luyện điều hành viên
  - Tập trung vào việc huấn luyện điều hành viên quen với các chức năng hỗ trợ của hệ thống và xác định cách thức hệ thống làm việc (hơn là cái mà hệ thống làm)
  - Thực hiện huấn luyện theo hai mức
    - Cách đưa ra và thực thi hệ thống mới
    - Cách hỗ trợ người dùng

Huấn luyện

- Yêu cầu huấn luyện đặc biệt
  - Người sử dụng thường xuyên và không thường xuyên
  - Người sử dụng mới

Huấn luyện

- Phương tiện trợ giúp huấn luyện
  - Tài liệu
    - Tài liệu hình thức, sách hướng dẫn
    - Một tỷ lệ phần trăm nhỏ người dùng đọc chúng
  - Trợ giúp trực tuyến và biểu tượng
    - Biểu tượng cho đối tượng và chức năng
    - Sách hướng dẫn trực tuyến cung cấp các liên kết siêu văn bản
  - Các thể hiện và các lớp
    - Tổ chức lớp học; Sử dụng thiết bị đa phương tiện (nghe, nhìn)
  - Người dùng thành thạo (và người được huấn luyện)

Huấn luyện

- Nguyên tắc huấn luyện
  - Hiểu sở thích cá nhân, kiểu làm việc, và áp lực tổ chức
  - Cần xem xét các kiểu học viên khác nhau
    - Hệ thống được đặc thù hóa
  - Chia nội dung huấn luyện thành các bài ở dạng ngắn
  - Xác định kiểu huấn luyện dựa trên vị trí của học viên

Tài liệu

- Hiểu thính giả
  - Phân nhóm thính giả
    - Người dùng
    - Điều hành viên
    - Nhân viên của khách hàng
    - Các thành viên khác của nhóm phát triển
  - Thiết kế tài liệu khác nhau cho các nhóm thính giả khác nhau

Tài liệu

- Các loại tài liệu
  - Sách hướng dẫn người dùng
  - Sách hướng dẫn điều hành viên
  - Hướng dẫn hệ thống chung
  - Hướng dẫn học và các khái quát được tự động
  - Tài liệu khác: hướng dẫn lập trình viên

Tài liệu

- Sách hướng dẫn người dùng
  - Bắt đầu bằng mục tiêu chung, tiến tới mô tả chức năng chi tiết
    - Mục tiêu và mục đích của hệ thống
    - Các chức năng và khả năng của hệ thống
    - Các đặc điểm, đặc trưng và thuận lợi của hệ thống

Tài liệu

- Sách hướng dẫn điều hành viên
  - Cấu hình phần cứng, phần mềm
  - Các phương pháp chấp nhận và từ chối truy xuất đối với người dùng
  - Các thủ tục thêm để bổ sung và loại bỏ các ngoại vi ra khỏi hệ thống
  - Các kỹ thuật để sao chép và dự phòng các tập tin và tài liệu

Tài liệu

- Hướng dẫn hệ thống chung
  - Hệ thống chi tiết hóa theo cách mà khách hàng có thể hiểu
  - Cấu hình phần cứng và phần mềm của hệ thống
  - Triết lý ẩn dưới cấu trúc của hệ thống

Tài liệu

- Hướng dẫn học và các khái quát được tự động
  - Hướng dẫn học được tự động hóa, từng bước một, dựa trên đa phương tiện

Tài liệu

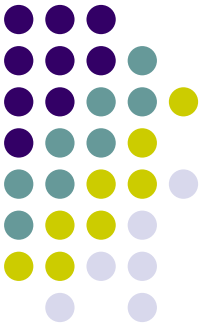
- Hướng dẫn lập trình viên
  - Khái quát về cách thức phần cứng, phần mềm được cấu hình
  - Các thành phần của phần mềm được chi tiết hóa và các chức năng của chúng được thực hiện
  - Các chức năng hỗ trợ hệ thống
  - Các cải tiến của hệ thống

Tài liệu

- Gỡ rối và giúp đỡ người dùng
  - Thông điệp chỉ dẫn khi hệ thống không thực hiện
  - Các tài liệu tham khảo
  - Các tập tin giúp đỡ trực tuyến

NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

CHƯƠNG 10– BẢO TRÌ HỆ THỐNG



Nội dung

- Hệ thống thay đổi
- Trạng thái tự nhiên của bảo trì
- Các vấn đề bảo trì
- Đo các đặc trưng của bảo trì
- Các kỹ thuật và công cụ bảo trì

Hệ thống thay đổi

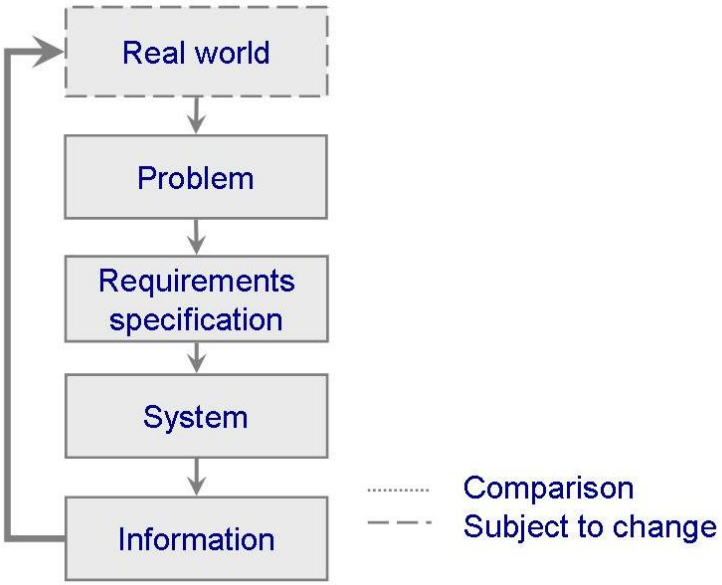
- Bảo trì: bất cứ công việc nào được thực hiện để thay đổi hệ thống sau khi nó được đưa vào vận hành
- Phần mềm liên tục tiến hóa => Quy trình bảo trì có thể khó khăn

Hệ thống thay đổi

- Các kiểu hệ thống của Lehman
  - Hệ thống S (S-system)
  - Hệ tống P (P-system)
  - Hệ thống E (E-system)

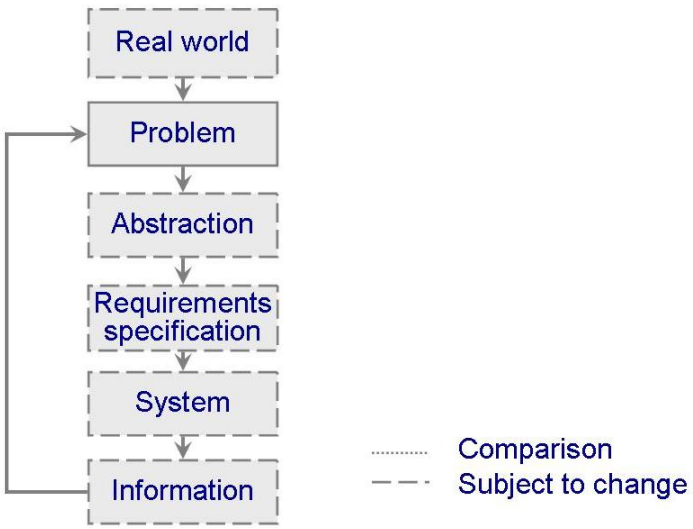
Hệ thống thay đổi

- Hệ thống S: được định nghĩa một cách hình thức và được dẫn ra từ sự đặc tả



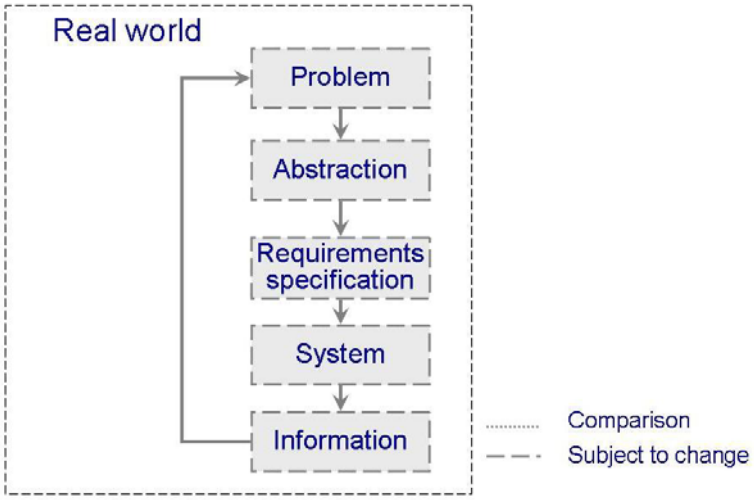
Hệ thống thay đổi

- Hệ thống P: các yêu cầu được dựa trên giải pháp gần với vấn đề (thực tế hơn để xây dựng và sử dụng)



Hệ thống thay đổi

- Hệ thống E: được nhúng vào thế giới thực và thay đổi và thay đổi theo nó



Hệ thống thay đổi

- Các thay đổi trong suốt chu kỳ sống của hệ thống
  - Hệ thống S: không thay đổi
  - Hệ thống P: thay đổi gia tăng
  - Hệ thống E: thay đổi hằng số

Bản chất của bảo trì

- Các loại bảo trì
  - Hiệu chỉnh (Corrective): duy trì sự kiểm soát trên các chức năng hàng ngày
  - Thích ứng (Adaptive): duy trì sự kiểm soát trên các sửa đổi của hệ thống
  - Hoàn thiện (Perfective): hoàn thiện các chức năng
  - Ngăn ngừa (Preventive): ngăn sự thực thi của hệ thống bị suy thoái tới các mức không thể chấp nhận

Bản chất của bảo trì

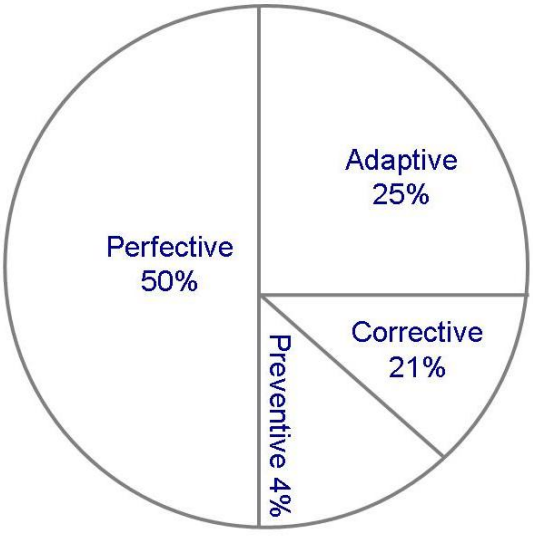
- Người thực hiện bảo trì
  - Nhóm bảo trì độc lập
    - Có thể khách quan hơn
    - Dễ dàng hơn trong phân biệt cách thức hệ thống nên thực hiện với cách thức hệ thống thực hiện
  - Một phần của nhóm phát triển
    - Xây dựng hệ thống theo cách làm cho việc bảo trì dễ dàng hơn
    - Quá tự tin, lờ đi tài liệu giúp cho việc bảo trì

Bản chất của bảo trì

- Trách nhiệm của nhóm bảo trì
  - Hiểu hệ thống
  - Định vị thông tin trong tài liệu hệ thống
  - Tiếp tục cập nhật tài liệu hệ thống
  - Mở rộng các chức năng hiện có để đáp ứng các yêu cầu đang thay đổi hay yêu cầu mới
  - Thêm các chức năng mới vào hệ thống
  - Tìm ra nguồn gốc của vấn đề hay sự thất bại của hệ thống
  - Định vị và hiệu chỉnh các lỗi
  - Trả lời các câu hỏi về cách thức hệ thống hoạt động
  - Tái tổ chức các thành phần mã lệnh và thiết kế
  - Viết lại các thành phần mã lệnh và thiết kế
  - Xóa các thành phần mã lệnh và thiết kế mà chúng không còn có ích
  - Quản lý các thay đổi của hệ thống khi chúng được tạo ra

Bản chất của bảo trì

- Sử dụng thời gian bảo trì:



Các vấn đề bảo trì

- Các vấn đề nhân sự
  - Sự hiểu biết hạn chế
  - Các ưu tiên trong quản lý
  - Đạo đức
- Các vấn đề kỹ thuật
  - Các mô hình
  - Các khó khăn của kiểm thử

Các vấn đề bảo trì

- Cần sự thỏa hiệp
  - Cân bằng nhu cầu thay đổi với nhu cầu giữ cho hệ thống sẵn có với người dùng
  - Sửa chữa vấn đề nhanh nhưng giải pháp thiếu sự tinh tế
  - Vấn đề giải quyết chỉ liên quan đến sự hiệu chỉnh tức thời của lỗi

Các vấn đề bảo trì

- Các yếu tố tác động đến phương pháp bảo trì
  - Kiểu không hoạt động
  - Tính khốc liệt hay nguy kịch của sự không hoạt động
  - Mức độ khó của các thay đổi được yêu cầu
  - Phạm vi của các thay đổi được yêu cầu
  - Tính phức tạp của các thành phần sẽ được thay đổi
  - Số vị trí vật lý mà tại đó các thay đổi phải được tạo ra

Các vấn đề bảo trì

- Các yếu tố tác động đến phương pháp bảo trì
  - Loại ứng dụng
  - Tính mới lạ của hệ thống
  - Khả năng của nhân viên bảo trì và tốc độ thay thế
  - Nhịp sống của hệ thống
  - Sự phụ thuộc vào môi trường hay thay đổi
  - Các đặc trưng của phần cứng
  - Chất lượng thiết kế
  - Chất lượng lập trình
  - Chất lượng tài liệu
  - Chất lượng kiểm thử

Các vấn đề bảo trì

- Mô hình hóa công sức bảo trì (Belady và Lehman)

$$M = p + K^{c-d}$$

  - M : toàn bộ công sức bảo trì
  - p : công sức sản xuất
  - c: tính phức tạp
  - d : mức độ quen thuộc
  - K : hằng số kinh nghiệm

Các vấn đề bảo trì

- Mô hình hóa công sức bảo trì (COCOMO 2)

$$\text{Size} = ASLOC\,(AA + SU + 0.4DM + 0.3CM + 0.3IM)/100$$

  - *ASLOC*: số dòng mã nguồn được sửa lại
  - *AA*: công sức đánh giá và đồng hóa
  - *SU*: mức hiểu biết phần mềm
  - *DM*: tỷ lệ % thiết kế được hiệu chỉnh
  - *CM*: tỷ lệ % mã lệnh được hiệu chỉnh
  - *IM*: tỷ lệ % mã lệnh bên ngoài được tích hợp

Các vấn đề bảo trì

Bảng đánh giá SU

	Very low	Low	Nominal	High	Very high
Structure	Very low cohesion, high coupling, spaghetti code	Moderately low cohesion, high coupling	Reasonably well structured, some weak areas	High cohesion, low coupling	Strong modularity, information hiding in data and control structure
Application clarity	No match between program and application worldviews	Some correlation between program and application	Moderate correlation between program and application	Good correlation between program and application	Clear match between program and application worldviews
Self descriptiveness	Obscure code; documentation missing, obscure, or obsolete	Some code commentary headers; some useful documentation	Moderate level of code commentary headers, and documentation	Good code commentary and headers; useful documentation; some weak areas	Self descriptive code; documentation up-to-date , well organized, with design rationale
SU increment	50	40	30	20	10

Các vấn đề bảo trì

Assessment and Assimilation Increment	Level of Assessment and Assimilation Effort
0	None
2	Basic component search and documentation
4	Some component test and evaluation documentation
6	Considerable component test and evaluation documentation
8	Extensive component test and evaluation documentation

Công sức AA

Đo các đặc trưng của bảo trì

- Tính dễ bảo trì không chỉ liên quan đến mã lệnh mà còn liên quan đến sự đặc tả, thiết kế và kế hoạch kiểm thử
- Tính dễ bảo trì có thể được nhìn theo hai cách
  - Nhìn bên ngoài phần mềm
  - Nhìn bên trong phần mềm

Đo các đặc trưng của bảo trì

- Dữ liệu cần thiết
  - Thời điểm mà vấn đề được báo cáo
  - Thời gian uổng phí do sự trì hoãn quản lý
  - Thời gian cần để phân tích vấn đề
  - Thời gian cần để xác định những thay đổi nào sẽ được tạo ra
  - Thời gian cần để tạo ra sự thay đổi
  - Thời gian cần để kiểm thử sự thay đổi
  - Thời gian cần để ghi sự thay đổi vào tài liệu
- Dữ liệu mong muốn
  - Tỷ lệ thời gian thực hiện thay đổi hoàn toàn trên số các thay đổi được thực hiện
  - Số vấn đề chưa được giải quyết
  - Thời gian dùng cho các vấn đề chưa được giải quyết
  - Tỷ lệ phần trăm các thay đổi gây ra lỗi mới
  - Số thành phần được sửa đổi để thực hiện sự thay đổi

Các thuộc tính bên ngoài tác động đến tính dễ bảo trì

Đo các đặc trưng của bảo trì

- Các thuộc tính bên trong tác động đến tính dễ bảo trì (theo McCabe)
  - Tính phức tạp về cấu trúc của mã nguồn
    - Đường độc lập tuyến tính
  - Dựa trên khái niệm lý thuyết đồ thị

Đo các đặc trưng của bảo trì

- Các phép đo khác
  - Phân tích cây phân lớp
  - Chỉ mục Fog
  - Tính dễ đọc của De Young và Kampen

Kỹ thuật và công cụ bảo trì

- Quản lý cấu hình
  - Ban kiểm soát cấu hình
  - Kiểm soát sự thay đổi
- Phân tích tác động
- Công cụ bảo trì được tự động hóa

Kỹ thuật và công cụ bảo trì

- Quy trình kiểm soát cấu hình
  - Vấn đề được phát hiện bởi hay sự thay đổi được yêu cầu bởi người dùng/khách hàng/nhà phát triển và được ghi lại
  - Sự thay đổi được báo cáo tới ban kiểm soát cấu hình (CCB)
  - CCB thảo luận về vấn đề: xác định bản chất của sự thay đổi, người trả chi phí
  - CCB thảo luận về nguồn gốc của vấn đề, phạm vi của sự thay đổi, thời gian sửa đổi; họ đưa ra độ ưu tiên và giao cho nhà phân tích
  - Nhà phân tích tạo ra sự thay đổi trên bản sao của kiểm thử
  - Nhà phân tích làm việc với thủ thư để kiểm soát sự thực hiện sự thay đổi
  - Nhà phân tích đưa ra báo cáo về sự thay đổi

Kỹ thuật và công cụ bảo trì

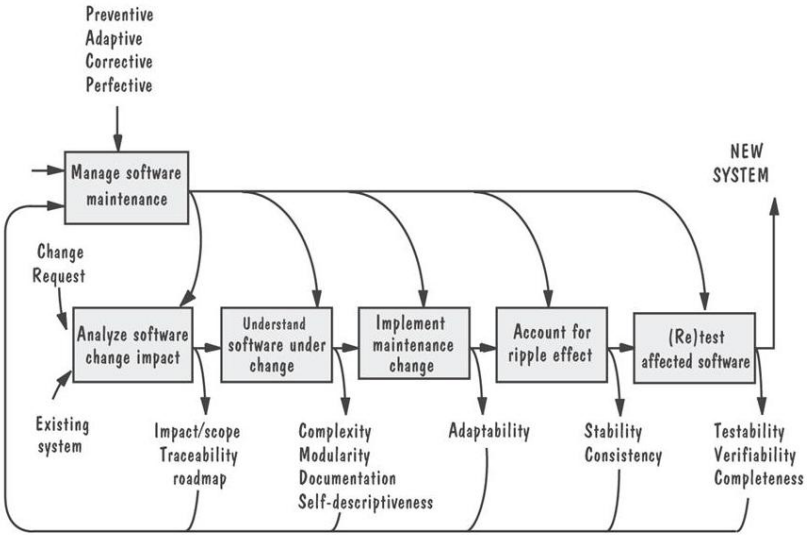
- Các vấn đề kiểm soát sự thay đổi
  - Sự đồng bộ hóa: khi sự thay đổi được tạo ra?
  - Nhận dạng: ai tạo ra sự thay đổi?
  - Đặt tên: thành phần nào của hệ thống được thay đổi?
  - Sự xác thực: thay đổi được tạo ra một cách chính xác?
  - Sự cho phép: ai được cho phép với sự thay đổi?
  - Lộ trình: ai được thông báo về sự thay đổi?
  - Sự hủy bỏ: ai có thể hủy bỏ yêu cầu thay đổi?
  - Sự ủy thác: ai có trách nhiệm đối với sự thay đổi?
  - Sự đánh giá: độ ưu tiên của sự thay đổi?

Kỹ thuật và công cụ bảo trì

- Phân tích sự tác động
  - Sự đánh giá các rủi ro liên quan tới sự thay đổi, bao gồm các dự đoán về các ảnh hưởng lên tài nguyên, công sức và lịch biểu
  - Giúp kiểm soát chi phí bảo trì

Kỹ thuật và công cụ bảo trì

- Các hoạt động bảo trì phần mềm: hình minh họa các hoạt động được thực hiện khi một thay đổi được yêu cầu



Kỹ thuật và công cụ bảo trì

- Công cụ bảo trì được tự động
  - Trình soạn thảo văn bản
  - Trình so tập tin
  - Trình biên dịch và trình tải
  - Công cụ gỡ rối
  - Bộ sinh tham chiếu chéo
  - Bộ phân tích mã lệnh tĩnh
  - Kho quản lý cấu hình

Nhập môn Công nghệ học Phần mềm

(Introduction to Software Engineering)

Department of Software Engineering

Faculty of Information Technology

Hanoi University of Technology

TEL: 04-8682595 FAX: 04-8692906

Email: cnpm@it-hut.edu.vn

Cấu trúc môn học

- 45 tiết + 1 Đồ án môn học
- Cần những kiến thức căn bản về CNTT
- Cung cấp những nguyên lý chung về Công nghệ học Phần mềm (CNHPM)
- Cung cấp kiến thức để học các môn chuyên ngành hẹp như Phân tích và thiết kế phần mềm, Xây dựng và đánh giá phần mềm, Quản trị dự án phần mềm,...

Cấu trúc môn học (tiếp)

- **Nội dung: gồm 6 phần với 11 chương**
  - **Giới thiệu chung về CNHPM (3 buổi)**
  - **Quản lý dự án PM (2b)**
  - **Yêu cầu người dùng (1b)**
  - **Thiết kế và lập trình (2b)**
  - **Kiểm thử và bảo trì (2b)**
  - **Chủ đề nâng cao và tổng kết (1b+1b)**
- **Đánh giá: Thi hết môn + Đồ án môn học**

Tài liệu tham khảo

- R. Pressman, *Software Engineering: A Practioner's Approach*. 5th Ed., McGraw-Hill, 2001
- R. Pressman, Kỹ nghệ phần mềm. Tập 1, 2, 3. NXB Giáo dục, Hà Nội, 1997 (Người dịch: Ngô Trung Việt)
- I. Sommerville, *Software Engineering*. 5th Ed., Addison-Wesley, 1995
- K. Kawamura, *Nhập môn Công nghệ học Phần mềm*. NXB Kinki-Kagaku, Tokyo, 2001 (Tiếng Nhật)

Phần I
Giới thiệu chung về CNHPM

Chương 1: Bản chất phần mềm

1.1 Định nghĩa chung về phần mềm
1.2 Kiến trúc phần mềm
1.3 Các khái niệm
1.4 Đặc tính chung của phần mềm
1.5 Thế nào là phần mềm tốt ?
1.6 Các ứng dụng phần mềm

1.1. Định nghĩa chung về phần mềm

- Phần mềm (Software - SW) như một khái niệm đối nghĩa với phần cứng (Hardware - HW), tuy nhiên, đây là 2 khái niệm tương đối
- Từ xưa, SW như thứ được cho không hoặc bán kèm theo máy (HW)
- Dần dần, giá thành SW ngày càng cao và nay cao hơn HW

Các đặc tính của SW và HW

HW	SW
• Vật "cứng"	• Vật "mềm"
• Kim loại	• Kỹ thuật sử dụng
• Vật chất	• Trừu tượng
• Hữu hình	• Vô hình
• Sản xuất công nghiệp bởi máy móc là chính	• Sản xuất bởi con người là chính
• Định lượng là chính	• Định tính là chính
• Hỏng hóc, hao mòn	• Không hao mòn

Định nghĩa 1: Phần mềm là

- Các lệnh (chương trình máy tính) khi được thực hiện thì cung cấp những chức năng và kết quả mong muốn
- Các cấu trúc dữ liệu làm cho chương trình thao tác thông tin thích hợp
- Các tư liệu mô tả thao tác và cách sử dụng chương trình

SW đối nghĩa với HW

- Vai trò SW ngày càng thể hiện trội
- Máy tính là . . . chiếc hộp không có SW
- Ngày nay, SW quyết định chất lượng một hệ thống máy tính (HTMT), là chủ đề cốt lõi, trung tâm của HTMT
- Hệ thống máy tính gồm HW và SW

Định nghĩa 2

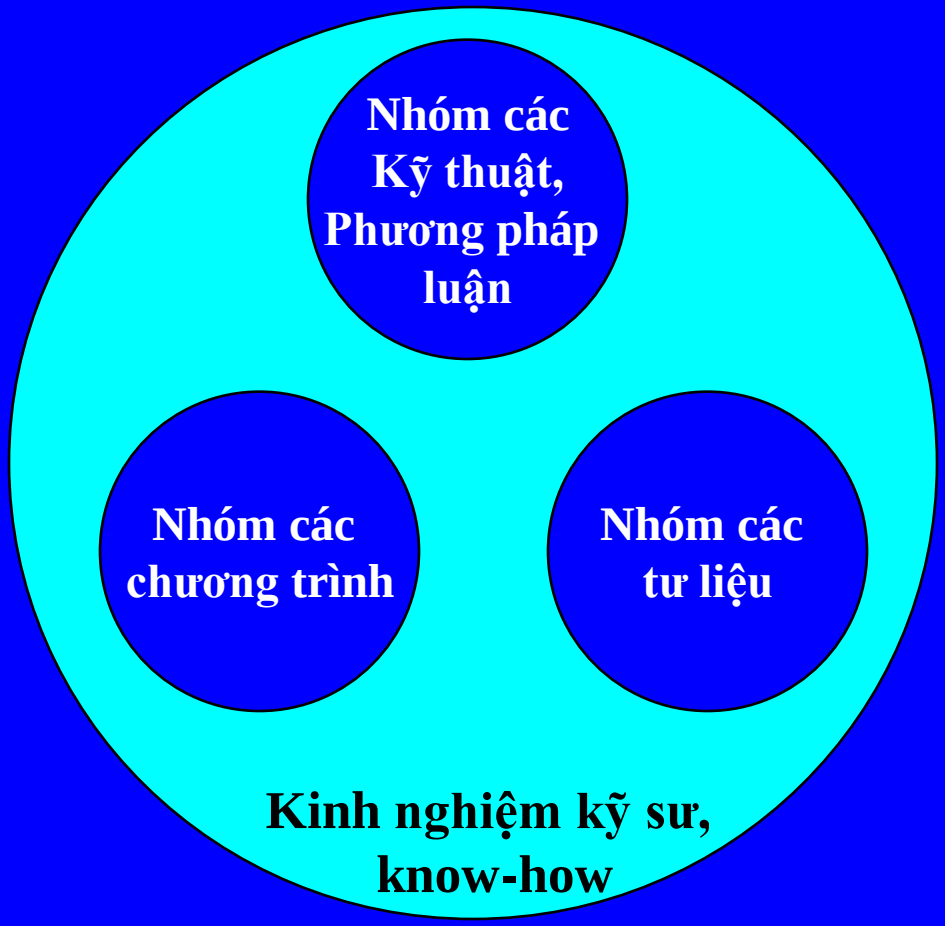
Trong một hệ thống máy tính, nếu trừ bỏ đi các thiết bị và các loại phụ kiện thì phần còn lại chính là phần mềm (SW)

- Nghĩa hẹp: SW là dịch vụ chương trình để tăng khả năng xử lý của phần cứng của máy tính (như hệ điều hành - OS)
- Nghĩa rộng: SW là tất cả các kỹ thuật ứng dụng để thực hiện những dịch vụ chức năng cho mục đích nào đó bằng phần cứng

SW theo nghĩa rộng

- Không chỉ SW cơ bản và SW ứng dụng
- Phải gồm cả khả năng, kinh nghiệm thực tiễn và kỹ năng của kỹ sư (người chế ra phần mềm): Know-how of Software Engineer
- Là tất cả các kỹ thuật làm cho sử dụng phần cứng máy tính đạt hiệu quả cao

Phần mềm là gì ?



Nhóm các kỹ thuật, phương pháp luận

- Các khái niệm và trình tự cụ thể hóa một hệ thống
- Các phương pháp tiếp cận giải quyết vấn đề
- Các trình tự thiết kế và phát triển được chuẩn hóa
- Các phương pháp đặc tả yêu cầu, thiết kế hệ thống, thiết kế chương trình, kiểm thử, toàn bộ quy trình quản lý phát triển phần mềm

Nhóm các chương trình

- Là phần giao diện với phần cứng, tạo thành từ các nhóm lệnh chỉ thị cho máy tính biết trình tự thao tác xử lý dữ liệu
- Phần mềm cơ bản: với chức năng cung cấp môi trường thao tác dễ dàng cho người sử dụng nhằm tăng hiệu năng xử lý của phần cứng (ví dụ như OS là chương trình hệ thống)
- Phần mềm ứng dụng: dùng để xử lý nghiệp vụ thích hợp nào đó (quản lý, kế toán, . . .), phần mềm đóng gói, phần mềm của người dùng, . . .

Nhóm các tư liệu

- Những tư liệu hữu ích, có giá trị cao và rất cần thiết để phát triển, vận hành và bảo trì phần mềm
- Để chế ra phần mềm với độ tin cậy cao cần tạo ra các tư liệu chất lượng cao: đặc tả yêu cầu, mô tả thiết kế từng loại, điều kiện kiểm thử, thủ tục vận hành, hướng dẫn thao tác

Những yếu tố khác

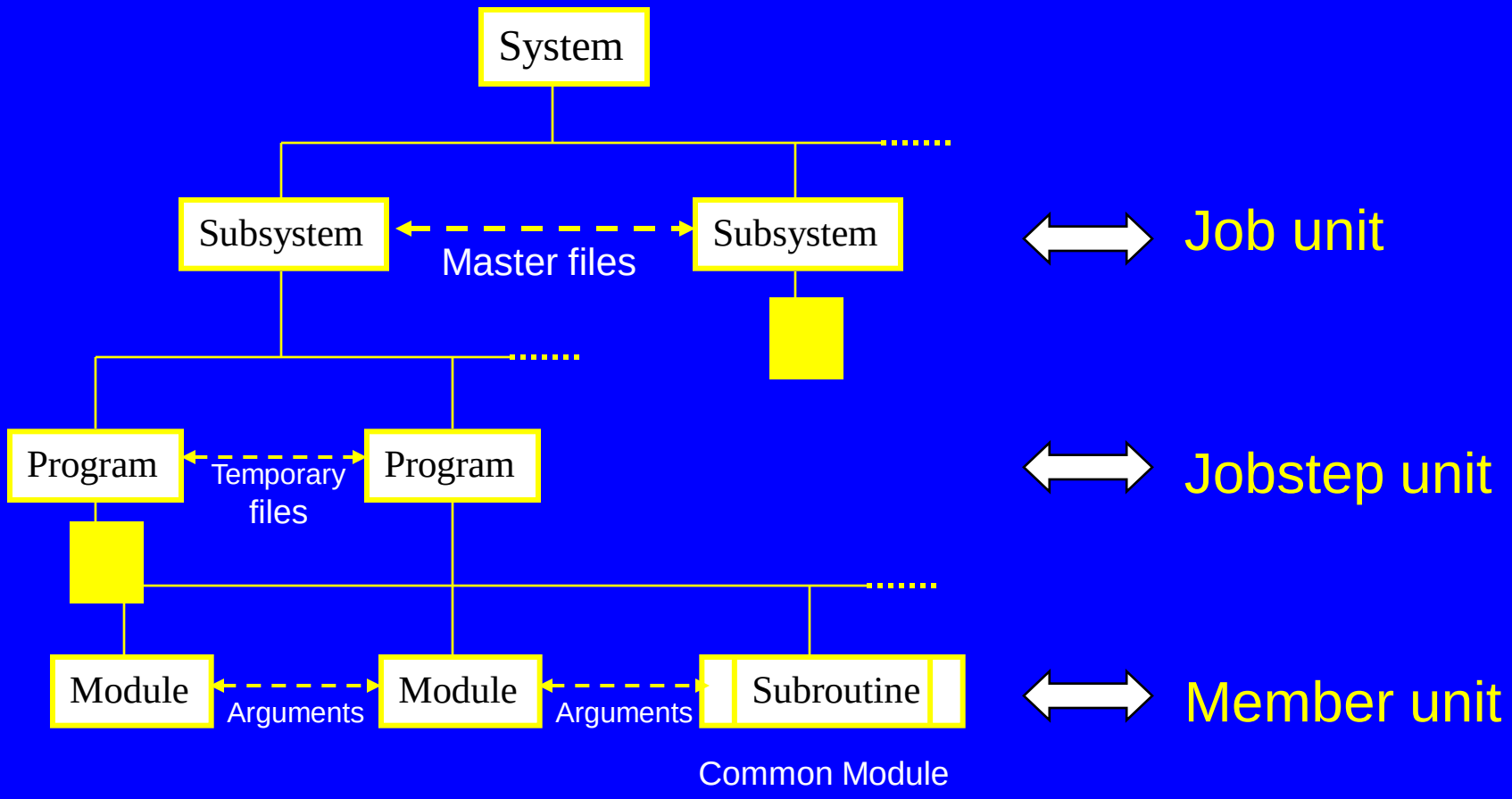
- Sản xuất phần mềm phụ thuộc rất nhiều vào con người (kỹ sư phần mềm). Khả năng hệ thống hóa trừu tượng, khả năng lập trình, kỹ năng công nghệ, kinh nghiệm làm việc, tầm bao quát, . . .: khác nhau ở từng người
- Phần mềm phụ thuộc nhiều vào ý tưởng (idea) và kỹ năng (know-how) của người/nhóm tác giả

1.2 Kiến trúc phần mềm

1.2.1 Phần mềm nhìn từ cấu trúc phân cấp

- Cấu trúc phần mềm là cấu trúc phân cấp (hierarchical structure): mức trên là hệ thống (system), dưới là các hệ thống con (subsystems)
- Dưới hệ thống con là các chương trình
- Dưới chương trình là các Modules hoặc Subroutines với các đối số (arguments)

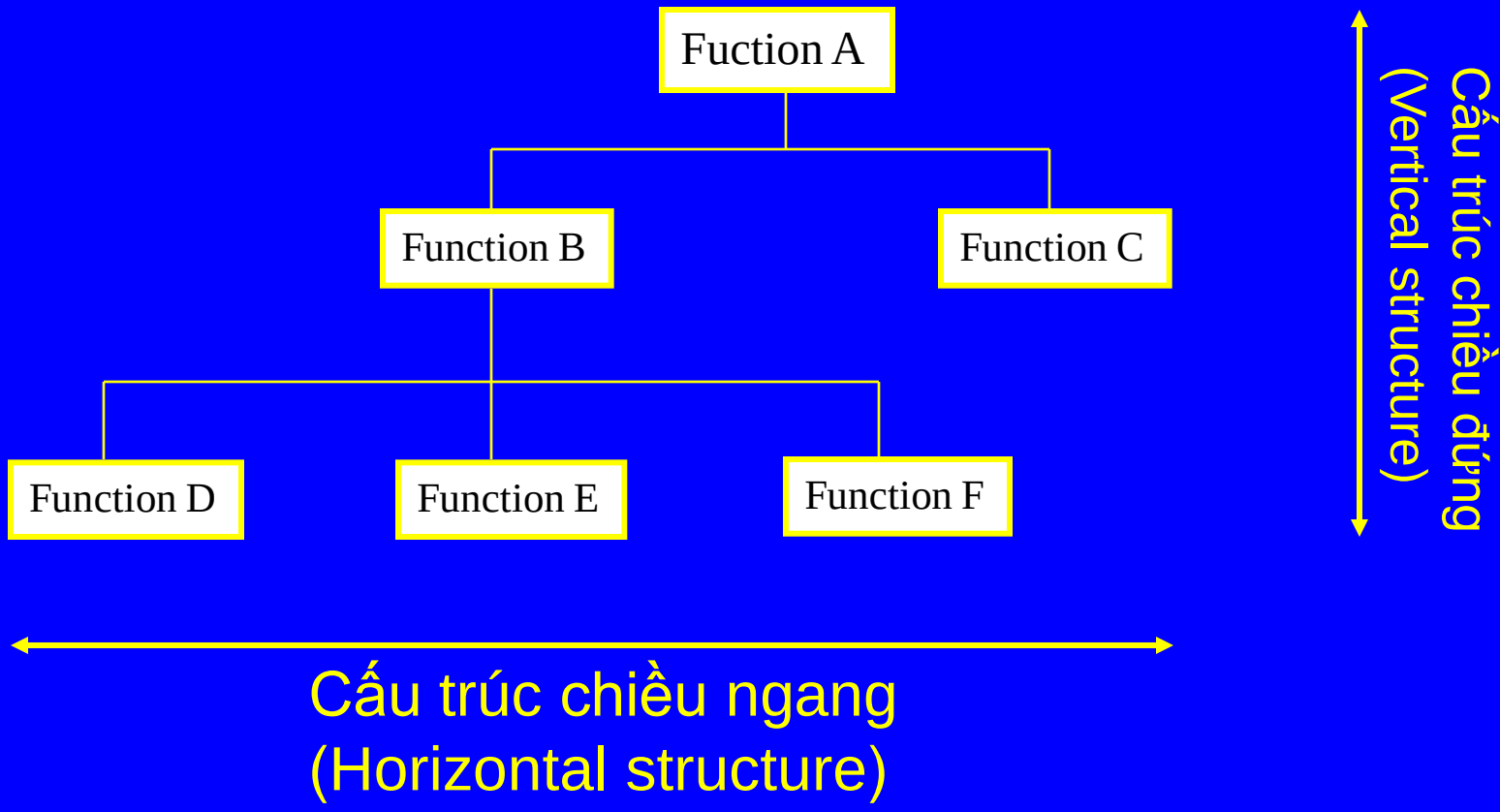
Kiến trúc phần mềm



1.2.2 Phần mềm nhìn từ cấu trúc và thủ tục

- Hai yếu tố cấu thành của phần mềm
  - Phương diện cấu trúc
  - Phương diện thủ tục
- Cấu trúc phần mềm: biểu thị kiến trúc các chức năng mà phần mềm đó có và điều kiện phân cấp các chức năng (thiết kế cấu trúc)
- Thiết kế chức năng: theo chiều đứng (càng sâu càng phức tạp) và chiều ngang (càng rộng càng nhiều chức năng, qui mô càng lớn)

Cấu trúc phần mềm



Thủ tục (procedure) phần mềm

- Là những quan hệ giữa các trình tự mà phần mềm đó có
- Thuật toán với những phép lặp, rẽ nhánh, điều khiển luồng xử lý (quay lui hay bỏ qua)
- Là cấu trúc lôgic biểu thị từng chức năng có trong phần mềm và trình tự thực hiện chúng
- Thiết kế cấu trúc trước rồi sang chức năng

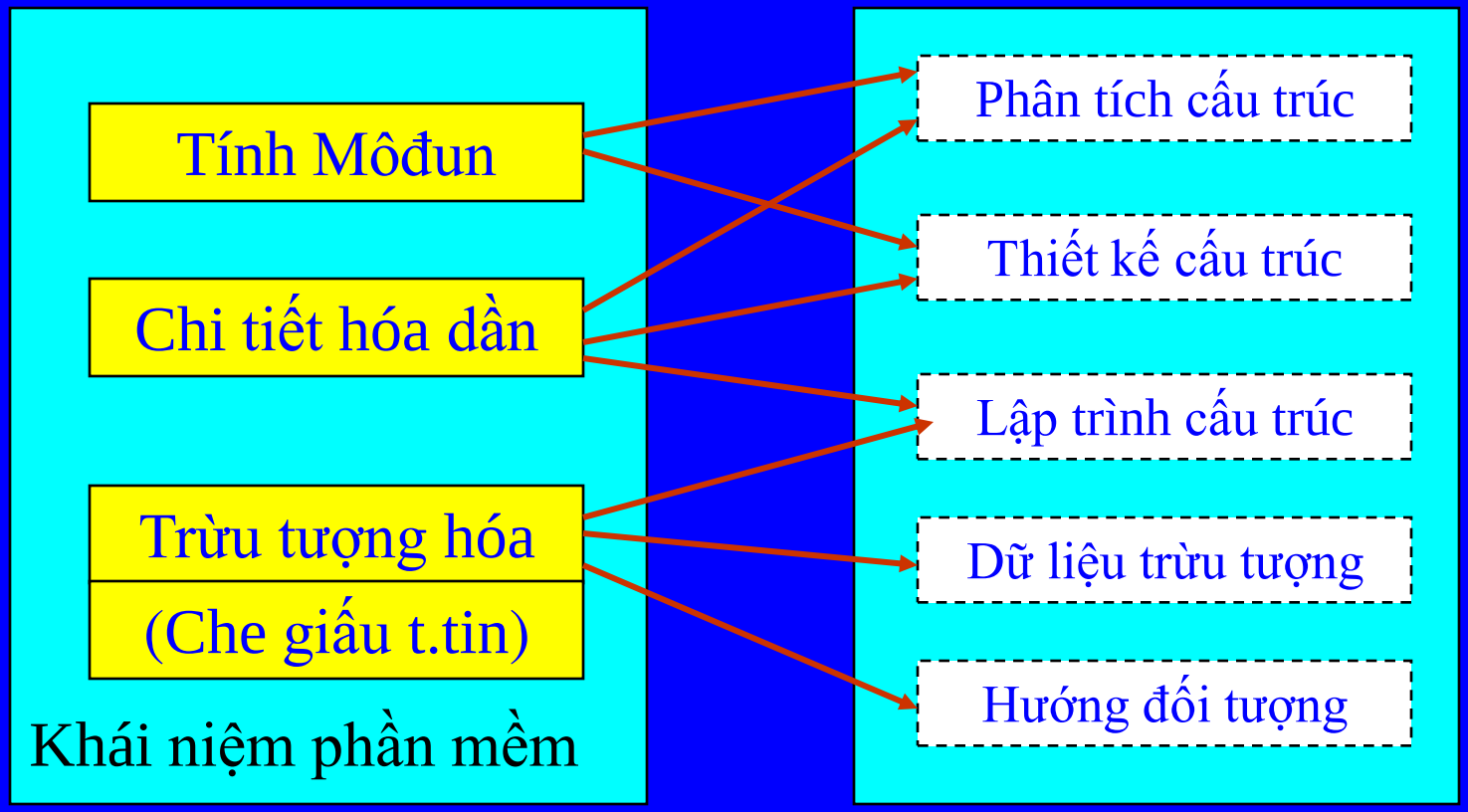
1.3 Các khái niệm

- Khi chế tác phần mềm cần nhiều kỹ thuật
  - Phương pháp luận (Methodology): những chuẩn mực cơ bản để chế tạo phần mềm với các chỉ tiêu định tính
  - Các phương pháp kỹ thuật (Techniques): những trình tự cụ thể để chế tạo phần mềm và là cách tiếp cận khoa học mang tính định lượng
- Từ phương pháp luận triển khai đến kỹ thuật

Các khái niệm (Software concepts)

- Khái niệm tính môđun (modularity concept)
- Khái niệm chi tiết hóa dần từng bước (stepwise refinement concept)
- Khái niệm trừu tượng hóa (abstraction concept): về thủ tục, điều khiển, dữ liệu
- Khái niệm che giấu thông tin (information hiding concept)
- Khái niệm hướng đối tượng (object oriented)

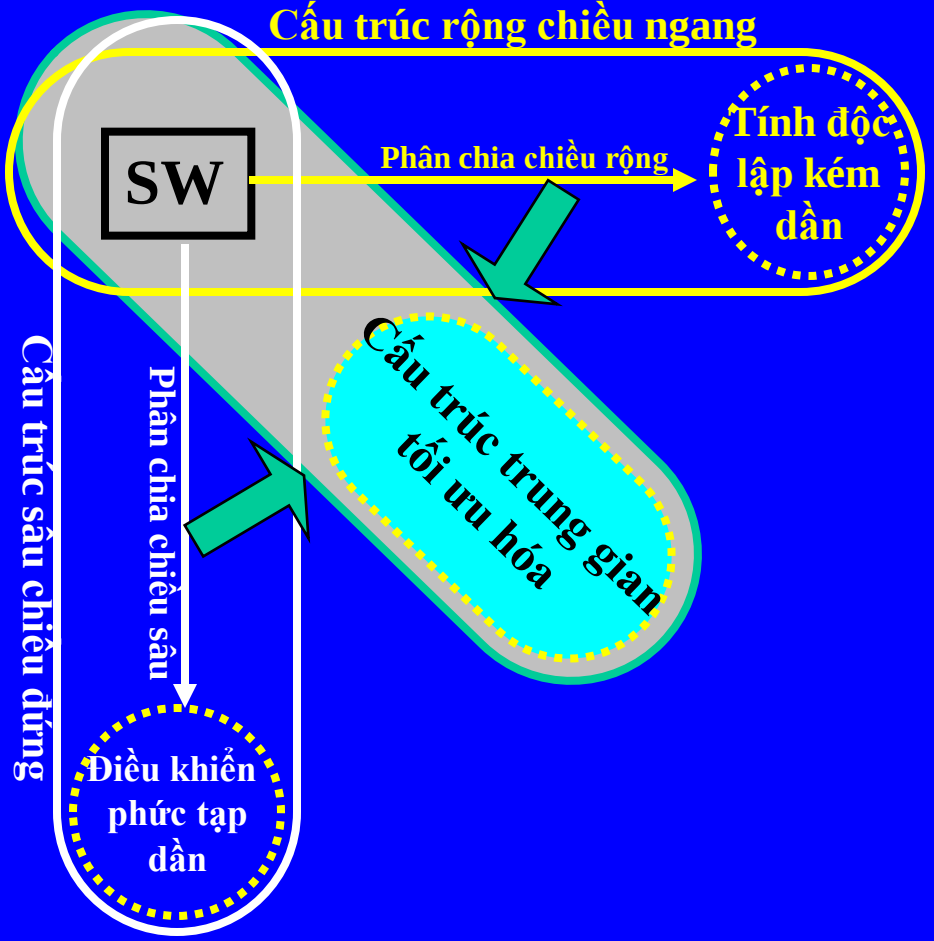
Từ phương pháp luận phần mềm sang kỹ thuật phần mềm



1.3.1 Tính môđun (Modularity)

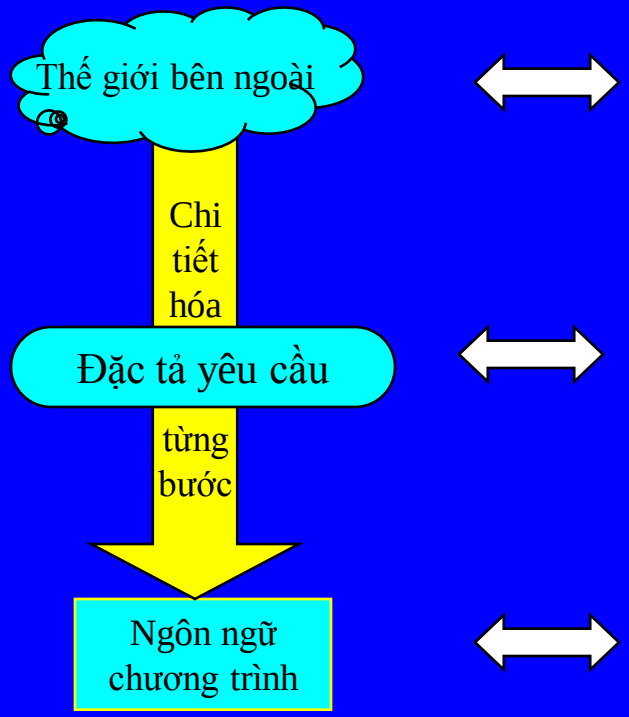
- Là khả năng phân chia phần mềm thành các môđun ứng với các chức năng, đồng thời cho phép quản lý tổng thể: khái niệm phân chia và trộn (partion and merge)
- Hai phương pháp phân chia môđun theo chiều
  - sâu (depth, thẳng đứng): điều khiển phức tạp dần
  - rộng (width, nằm ngang): môđun phụ thuộc dần
- Quan hệ giữa các môđun: qua các đối số (arguments)

Chuẩn phân chia môđun



1.3.2 Chi tiết hóa từng bước

Cách tiếp cận từ trên xuống (top-down approach)



Trừu tượng hóa mức cao:
Thế giới bên ngoài,
trạng thái chưa rõ ràng

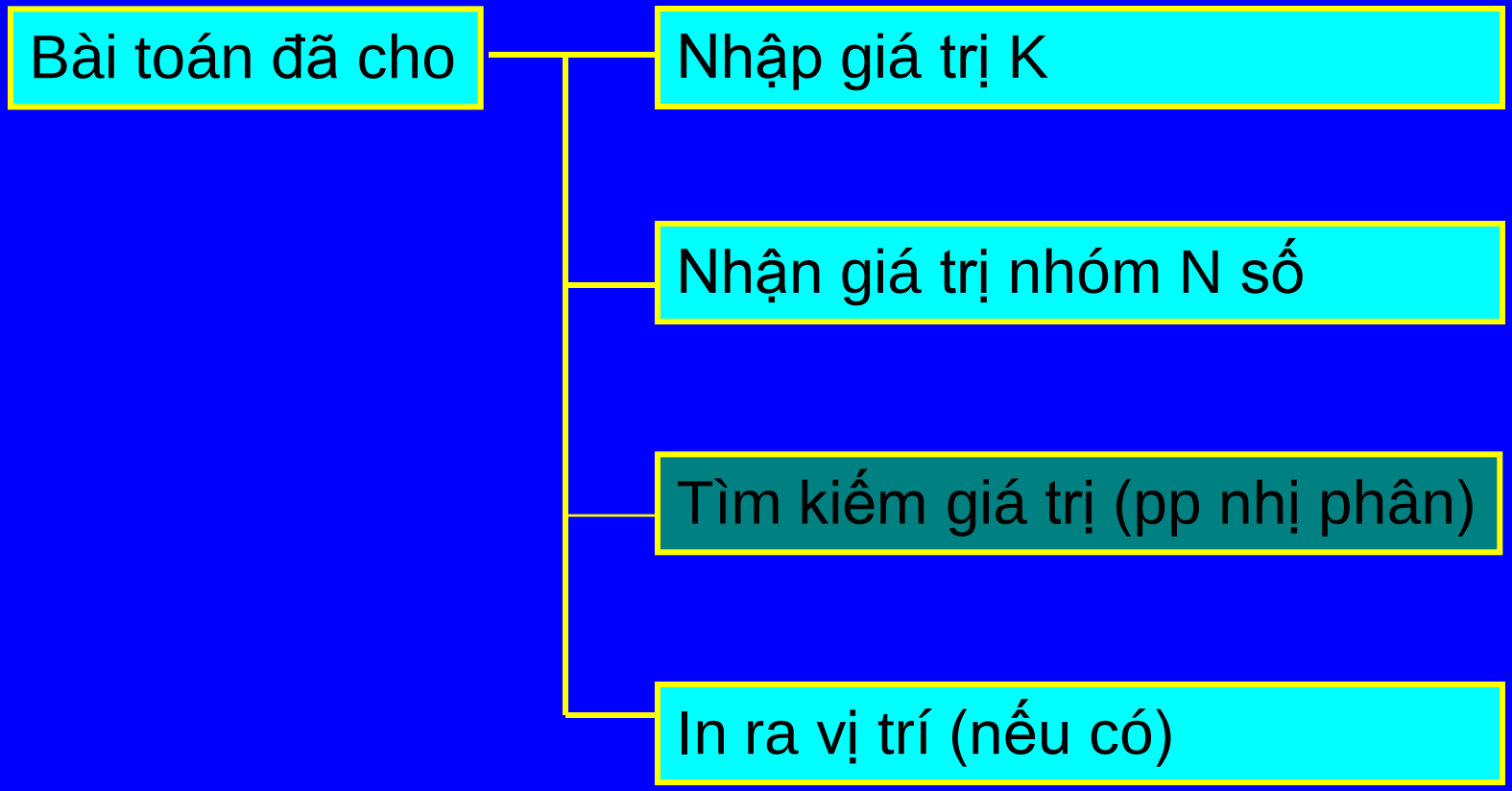
Trừu tượng hóa mức trung gian:
Xác định yêu cầu và đặc tả
những định nghĩa yêu cầu

Trừu tượng hóa mức thấp:
Từng lệnh của chương trình được
viết bởi ngôn ngữ thủ tục nào đó

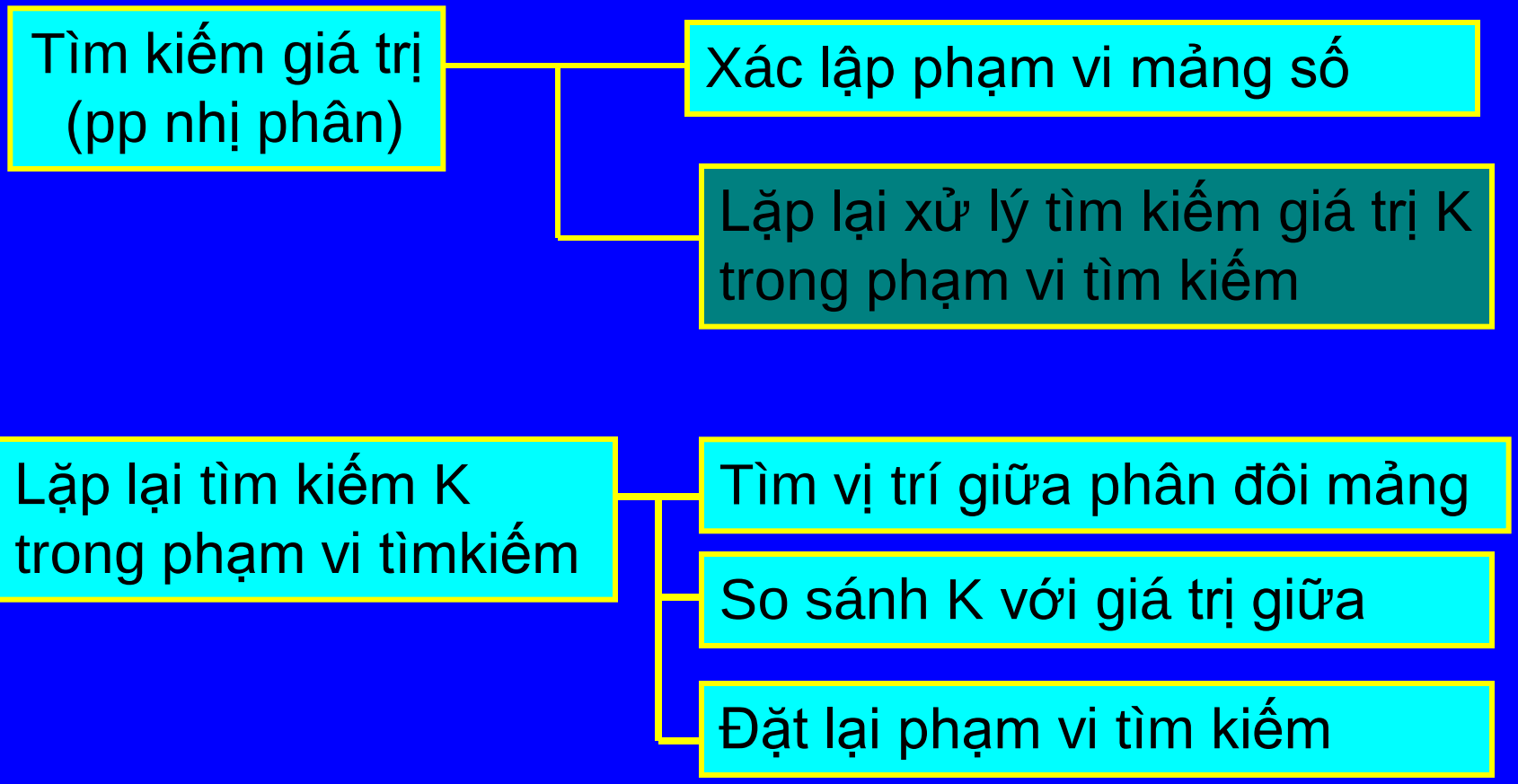
Ví dụ: Trình tự giải quyết vấn đề từ mức thiết kế chương trình đến mức lập trình

- Bài toán: từ một nhóm N số khác nhau tăng dần, hãy tìm số có giá trị bằng K (nhập từ ngoài vào) và in ra vị trí của nó
- Giải từng bước từ khái niệm đến chi tiết hóa từng câu lệnh bởi ngôn ngữ lập trình nào đó
- Chọn giải thuật tìm kiếm nhị phân (pp nhị phân)

Cụ thể hóa thủ tục qua các chức năng



Cụ thể hóa bước tiếp theo



Mức mô tả chương trình (bằng PDL)

```
Bắt đầu
    Đọc K
    Nhận giá trị cho mảng 1 chiều A(I), (I =1, 2,  . . . ,.N)
    MIN = 1
    MAX = N
    DO WHILE  (Có giá trị bằng K không, cho đến khi MIN > MAX)
        Lấy MID = (MIN + MAX) / 2
        IF  A(MID) > K THEN
                MAX = MID - 1
        ELSE
                IF A(MID) <  K THEN
                        MIN = MID + 1
                ELSE
                        In  giá trị  MID
                ENDIF
        ENDIF
    ENDDO
KếtThúc
```

1.3.3 Khái niệm Che giấu thông tin

- Để phân rã phần mềm thành các môđun một cách tốt nhất, cần tuân theo nguyên lý che giấu thông tin: "các môđun nên được đặc trưng bởi những quyết định thiết kế sao cho mỗi môđun ẩn kín đối với các môđun khác" [Parnas1972]
- Rất hữu ích cho kiểm thử và bảo trì phần mềm

Khái niệm Trừu tượng hóa

- Abstraction cho phép tập trung vấn đề ở mức tổng quát, gạt đi những chi tiết mức thấp ít liên quan
- 3 mức trừu tượng
  - Trừu tượng thủ tục: dãy các chỉ thị với chức năng đặc thù và giới hạn nào đó
  - Trừu tượng dữ liệu: tập hợp dữ liệu mô tả đối tượng dữ liệu nào đó
  - Trừu tượng điều khiển: Cơ chế điều khiển chương trình không cần đặc tả những chi tiết bên trong
- Ví dụ: *Mở cửa*. Thủ tục: Mở gồm . . .; Dữ liệu: Cửa là . . .

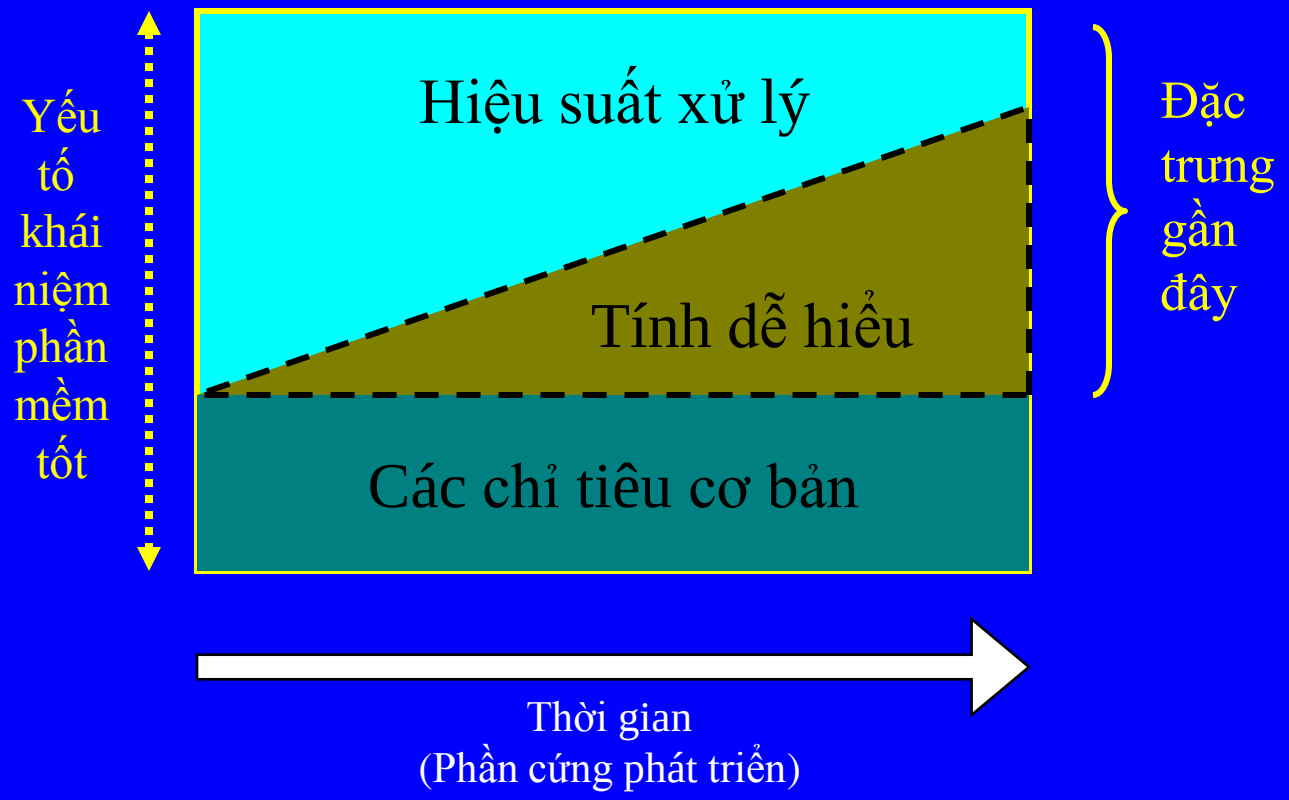
1.4 Đặc tính chung của phần mềm

- Là hàng hóa vô hình, không nhìn thấy được
- Chất lượng phần mềm: không mòn đi mà có xu hướng tốt lên sau mỗi lần có lỗi (error/bug) được phát hiện và sửa
- Phần mềm vốn chứa lỗi tiềm tàng, theo quy mô càng lớn thì khả năng chứa lỗi càng cao
- Lỗi phần mềm dễ được phát hiện bởi người ngoài

Đặc tính chung của phần mềm (tiếp)

- Chức năng của phần mềm thường biến hóa, thay đổi theo thời gian (theo nơi sử dụng)
- Hiệu ứng làn sóng trong thay đổi phần mềm
- Phần mềm vốn chứa ý tưởng và sáng tạo của tác giả/nhóm làm ra nó
- Cần khả năng "tư duy nhị phân" trong xây dựng, phát triển phần mềm
- Có thể sao chép rất đơn giản

1.5 Thế nào là phần mềm tốt ?



1.5.1 Các chỉ tiêu cơ bản

- Phản ánh đúng yêu cầu người dùng (tính hiệu quả - effectiveness)
- Chứa ít lỗi tiềm tàng
- Giá thành không vượt quá giá ước lượng ban đầu
- Dễ vận hành, sử dụng
- Tính an toàn và độ tin cậy cao

1.5.2 Hiệu suất xử lý cao

- **Hiệu suất thời gian tốt (efficiency):**
  - **Độ phức tạp tính toán thấp (Time complexity)**
  - **Thời gian quay vòng ngắn (Turn Around Time: TAT)**
  - **Thời gian hồi đáp nhanh (Response time)**
- **Sử dụng tài nguyên hữu hiệu: CPU, RAM, HDD, Internet resources, . . .**

1.5.3 Tính dễ hiểu

- Kiến trúc và cấu trúc thiết kế dễ hiểu
- Dễ kiểm tra, kiểm thử, kiểm chứng
- Dễ bảo trì
- Có tài liệu (mô tả yêu cầu, điều kiện kiểm thử, vận hành, bảo trì, FAQ, . . .) với chất lượng cao

Tính dễ hiểu: chỉ tiêu ngày càng quan trọng

1.6 Các ứng dụng phần mềm

- Phần mềm hệ thống (System SW)
- Phần mềm thời gian thực (Real-time SW)
- Phần mềm nghiệp vụ (Business SW)
- Phần mềm tính toán KH&KT (Eng.&Scie. SW)
- Phần mềm nhúng (Embedded SW)
- Phần mềm máy cá nhân (Personal computer SW)
- Phần mềm trên Web (Web-based SW)
- Phần mềm trí tuệ nhân tạo (AI SW)

Chương 2:
Khủng hoảng phần mềm
(Software Crisis)

2.1 Khủng hoảng phần mềm là gì ?

2.2 Những vấn đề (khó khăn) trong sản xuất phần mềm

2.1 Khủng hoảng phần mềm là gì?

- 10/1968 tại Hội nghị của NATO các chuyên gia phần mềm đã đưa ra thuật ngữ "Khủng hoảng phần mềm" (Software crisis). Qua hàng chục năm, thuật ngữ này vẫn được dùng và ngày càng mang tính cấp bách
- Khủng hoảng là gì ? [Webster's Dict.]
  - Điểm ngoặt trong tiến trình của bất kỳ cái gì; thời điểm, giai đoạn hoặc biến cố quyết định hay chủ chốt
  - Điểm ngoặt trong quá trình diễn biến bệnh khi trở nên rõ ràng bệnh nhân sẽ sống hay chết
- Trong phần mềm: Day dứt kinh niên (chronic affliation, by Prof. Tiechrow, Geneva, Arp. 1989)

Khủng hoảng phần mềm là gì? (tiếp)

***Là sự day dứt kinh niên (kéo dài theo thời gian hoặc thường tái diễn, liên tục không kết thúc) gặp phải trong phát triển phần mềm máy tính*, như**

- **Phải làm thế nào với việc giảm chất lượng vì những lỗi tiềm tàng có trong phần mềm ?**
- **Phải xử lý ra sao khi bảo dưỡng phần mềm đã có ?**
- **Phải giải quyết thế nào khi thiếu kỹ thuật viên phần mềm?**
- **Phải chế tác phần mềm ra sao khi có yêu cầu phát triển theo qui cách mới xuất hiện ?**
- **Phải xử lý ra sao khi sự cố phần mềm gây ra những vấn đề xã hội ?**

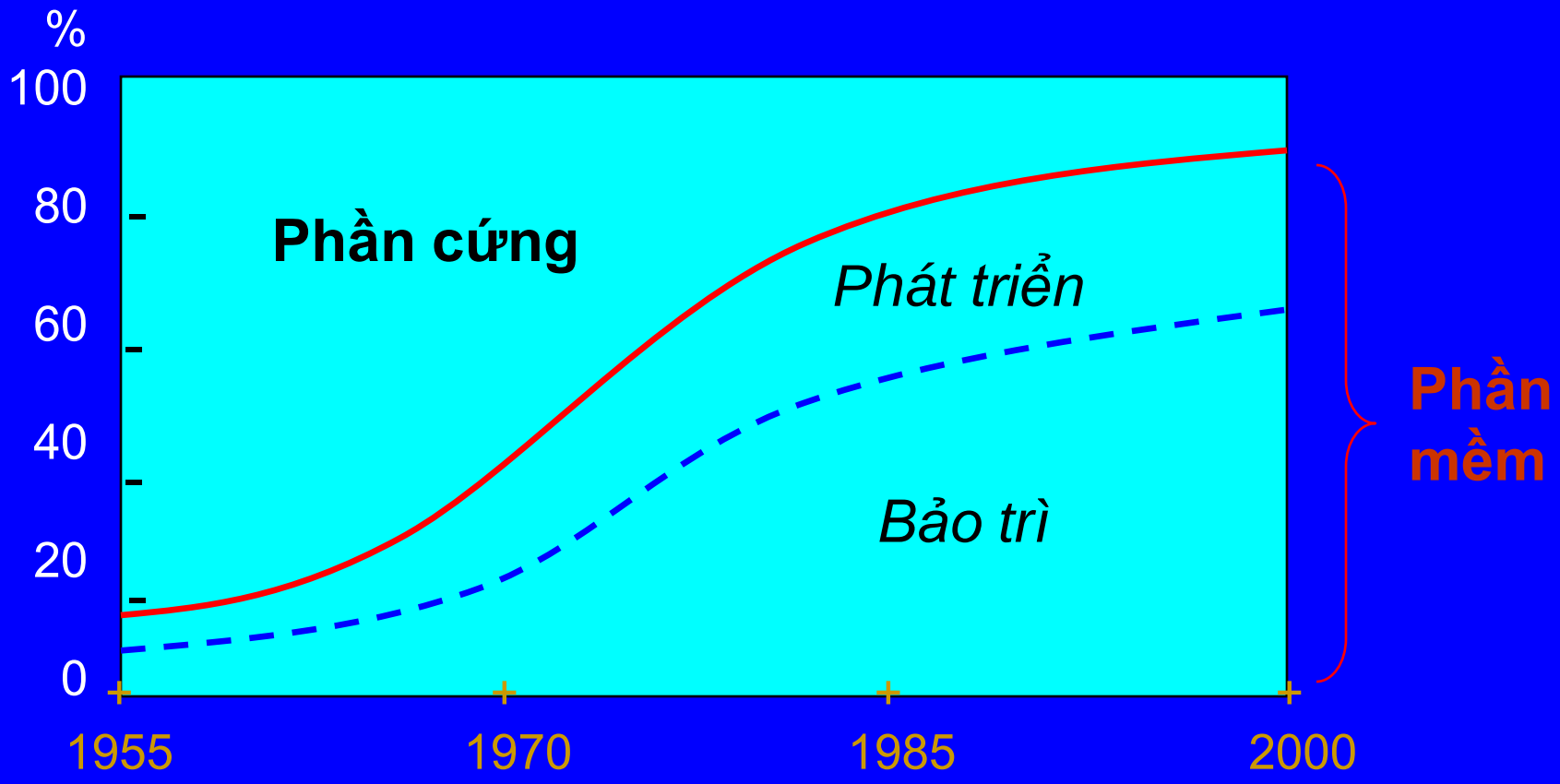
Một số yếu tố

- Phần mềm càng lớn sẽ kéo theo phức tạp hóa và tăng chi phí phát triển
- Đổi vai trò giá thành SW vs. HW
- Công sức cho bảo trì càng tăng thì chi phí cho Backlog càng lớn
- Nhân lực chưa đáp ứng được nhu cầu phần mềm
- Những phiền hà của phần mềm gây ra những vấn đề xã hội

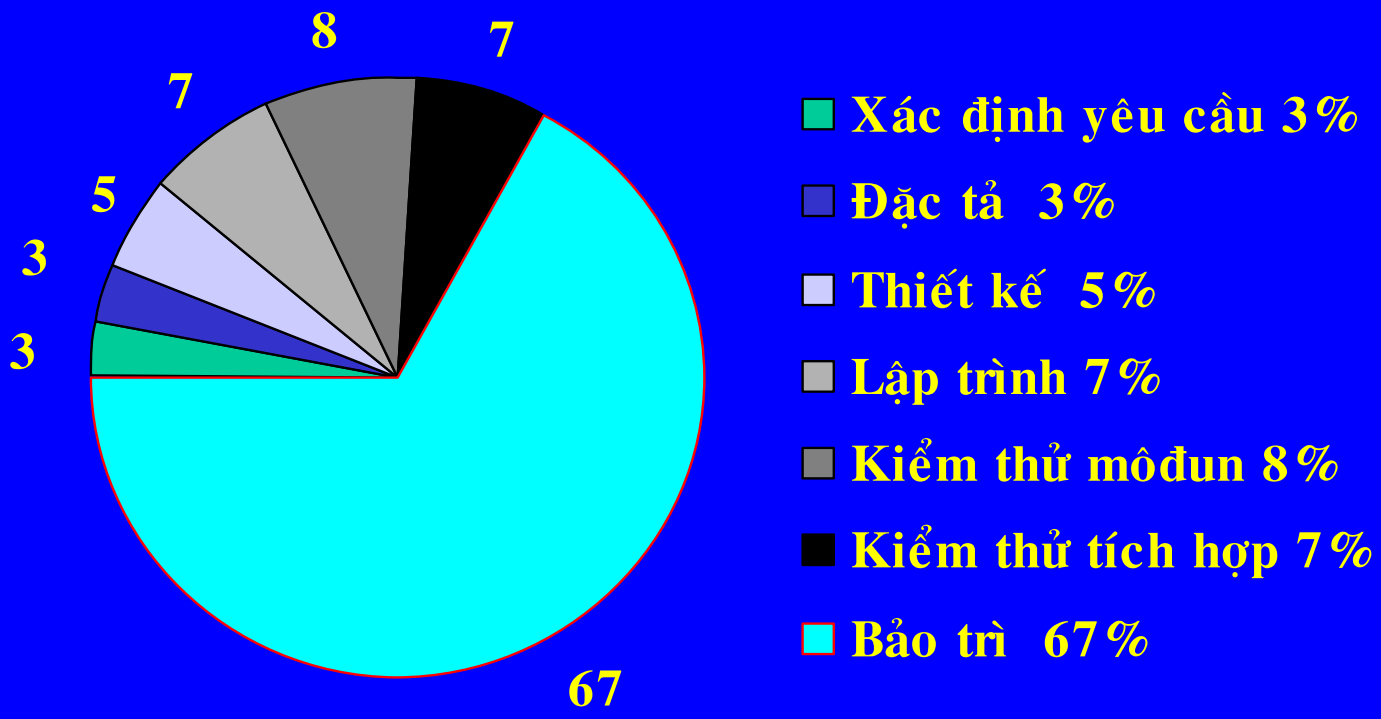
Những dự án lớn của NASA
(National Aeronautics and Space Administration)

Tên dự án	Thời điểm phát triển	Tổng số b- ớc (triệu)
GEMINI	Giữa 1960	6
APPOLO (1 Bill. $)	Đầu 1970	13
SPACE SHUTTLE	Cuối 1970	45

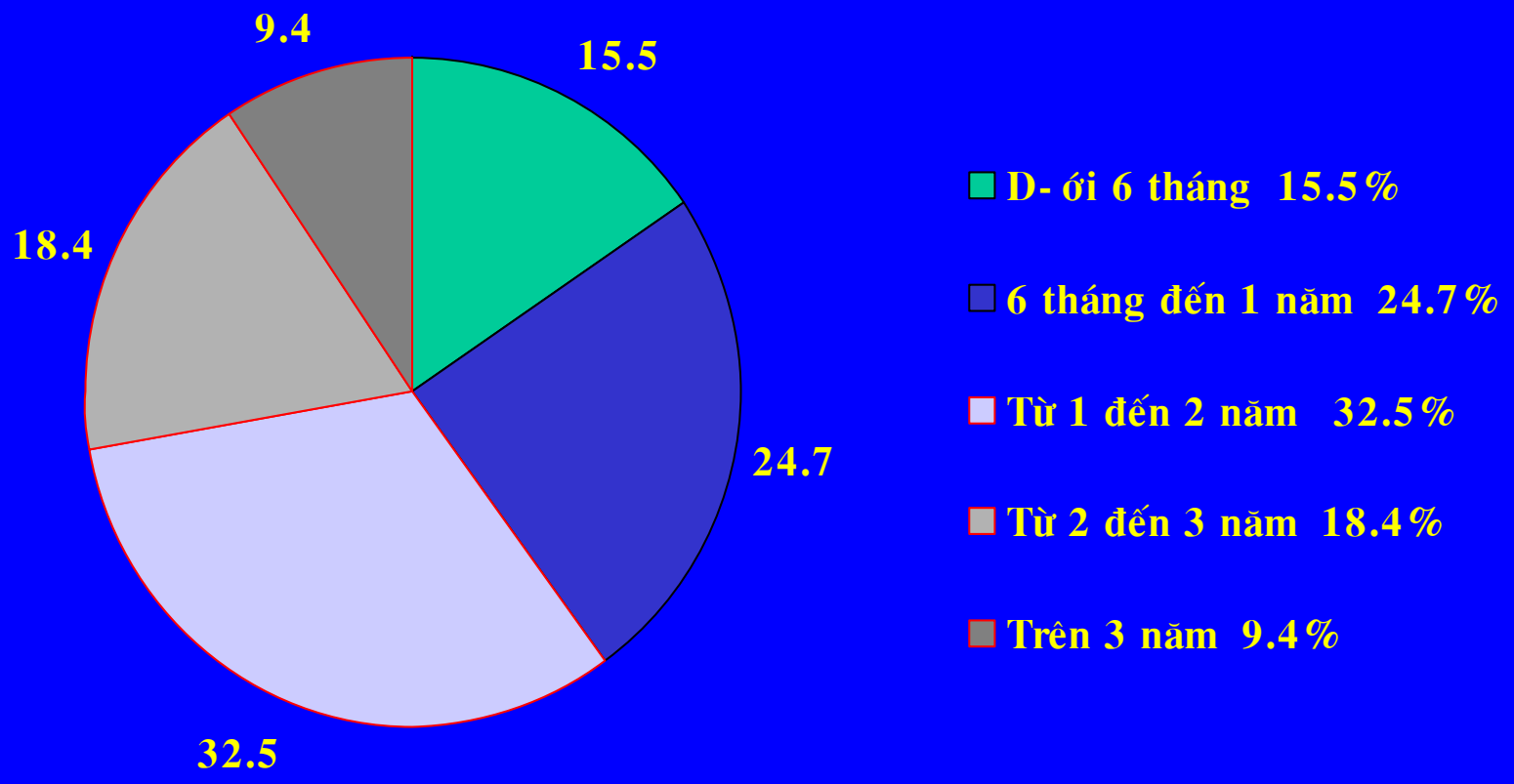
So sánh chi phí cho Phần cứng và Phần mềm



So sánh chi phí cho các pha



Backlog tại Nhật Bản năm 1985



Những vấn đề (khó khăn) trong sản xuất phần mềm

(1) Không có phương pháp mô tả rõ ràng định nghĩa yêu cầu của người dùng (khách hàng), sau khi bàn giao sản phẩm dễ phát sinh những trục trặc (troubles)

(2) Với những phần mềm quy mô lớn, tư liệu đặc tả đã cố định thời gian dài, do vậy khó đáp ứng nhu cầu thay đổi của người dùng một cách kịp thời trong thời gian đó

Những vấn đề trong sản xuất phần mềm (tiếp)

(3) Nếu không có Phương pháp luận thiết kế nhất quán mà thiết kế theo cách riêng (của công ty, nhóm), thì sẽ dẫn đến suy giảm chất lượng phần mềm (do phụ thuộc quá nhiều vào con người)

(4) Nếu không có chuẩn về làm tư liệu quy trình sản xuất phần mềm, thì những đặc tả không rõ ràng sẽ làm giảm chất lượng phần mềm

Những vấn đề trong sản xuất phần mềm (tiếp)

(5) Nếu không kiểm thử tính đúng đắn của phần mềm ở từng giai đoạn mà chỉ kiểm ở giai đoạn cuối và phát hiện ra lỗi, thì thường bàn giao sản phẩm không đúng hạn

(6) Nếu coi trọng việc lập trình hơn khâu thiết kế thì thường dẫn đến làm giảm chất lượng phần mềm

(7) Nếu coi thường việc tái sử dụng phần mềm (software reuse), thì năng suất lao động sẽ giảm

Những vấn đề trong sản xuất phần mềm (tiếp)

(8) Phần lớn trong quy trình phát triển phần mềm có nhiều thao tác do con người thực hiện, do vậy năng suất lao động thường bị giảm

(9) Không chứng minh được tính đúng đắn của phần mềm, do vậy độ tin cậy của phần mềm sẽ giảm

(10) Chuẩn về một phần mềm tốt không thể đo được một cách định lượng, do vậy không thể đánh giá được một hệ thống đúng đắn hay không

Những vấn đề trong sản xuất phần mềm (tiếp)

(11) Khi đầu tư nhân lực lớn vào bảo trì sẽ làm giảm hiệu suất lao động của nhân viên

(12) Công việc bảo trì kéo dài làm giảm chất lượng của tư liệu và ảnh hưởng xấu đến những việc khác

Những vấn đề trong sản xuất phần mềm (tiếp)

(13) Quản lý dự án lỏng lẻo kéo theo quản lý lịch trình cũng không rõ ràng

(14) Không có tiêu chuẩn để ước lượng nhân lực và dự toán sẽ làm kéo dài thời hạn và vượt kinh phí của dự án

Đây là những vấn đề phản ánh các khía cạnh khủng hoảng phần mềm, hãy tìm cách nỗ lực vượt qua để tạo ra phần mềm tốt!

Chương 3
Công nghệ học Phần mềm
(Software Engineering)

3.1 Lịch sử tiến triển Công nghệ học phần mềm

3.2 Sự tiến triển của các phương pháp thiết kế phần mềm

3.3 Định nghĩa Công nghệ học phần mềm

3.4 Vòng đời của phần mềm

3.5 Quy trình phát triển phần mềm

3.1 Lịch sử tiến triển của CNHPM

- **Nửa đầu 1960: ít quan tâm đến phần mềm, chủ yếu tập trung nâng cao tính năng và độ tin cậy của phần cứng**
- **Giữa những năm 1960: Phát triển hệ điều hành như phần mềm lớn (IBM OS/360, EC OS). Xuất hiện nhu cầu về quy trình phát triển phần mềm lớn và quy trình gỡ lỗi, kiểm thử trong phạm vi giới hạn**

Lịch sử tiến triển của CNHPM (tiếp)

- **Năm 1968: Tại Tây Đức, Hội nghị khoa học của NATO đã đưa ra từ "Software Engineering". Bắt đầu bàn luận về khủng khoảng phần mềm và xu hướng hình thành CNHPM như một chuyên môn riêng**
- **Nửa cuối 1960: IBM đưa ra chính sách phân biệt giá cả giữa phần cứng và phần mềm. Từ đó, ý thức về phần mềm ngày càng cao. Bắt đầu những nghiên cứu cơ bản về phương pháp luận lập trình**

Lịch sử tiến triển của CNHPM (tiếp)

- **Nửa đầu những năm 1970**: Nhằm nâng cao chất lượng phần mềm, không chỉ có các nghiên cứu về lập trình, kiểm thử, mà có cả những nghiên cứu đảm bảo tính tin cậy trong quy trình sản xuất phần mềm. Kỹ thuật: lập trình cấu trúc hóa, lập trình môđun, thiết kế cấu trúc hóa, vv
- **Giữa những năm 1970**: Hội nghị quốc tế đầu tiên về CNHPM được tổ chức (1975): International Conference on SE (ICSE)

Lịch sử tiến triển của CNHPM (tiếp)

- **Nửa sau những năm 1970: Quan tâm đến mọi pha trong quy trình phát triển phần mềm, nhưng tập trung chính ở những pha đầu. ICSE tổ chức lần 2, 3 và 4 vào 1976, 1978 và 1979**
  - **Nhật Bản có "Kế hoạch phát triển kỹ thuật sản xuất phần mềm" từ năm 1981**
  - **Cuộc "cách tân sản xuất phần mềm" đã bắt đầu trên phạm vi các nước công nghiệp**

Lịch sử tiến triển của CNHPM (tiếp)

- **Nửa đầu những năm 1980: Trình độ học vấn và ứng dụng CNHPM được nâng cao, các công nghệ được chuyển vào thực tế. Xuất hiện các sản phẩm phần mềm và các công cụ khác nhau làm tăng năng suất sản xuất phần mềm đáng kể**
  - **ICSE tổ chức lần 5 và 6 năm 1981 và 1982 với trên 1000 người tham dự mỗi năm**
  - **Nhật Bản sang "Kế hoạch phát triển các kỹ thuật bảo trì phần mềm" (1981-1985)**

Lịch sử tiến triển của CNHPM (tiếp)

- **Nửa cuối những năm 1980 đến nay: Từ học vấn sang nghiệp vụ! Chất lượng phần mềm tập trung chủ yếu ở *tính năng suất, độ tin cậy và tính bảo trì*. Nghiên cứa hỗ trợ tự động hóa sản xuất phần mềm**
  - **Nhật Bản có "Kế hoạch hệ thống công nghiệp hóa sản xuất phần mềm"(SIGMA: Software Industrialized Generator & Maintenance Aids, 1985-1990)**
  - **Nhiều trung tâm, viện nghiên cứu CNHPM ra đời. Các trường đưa vào giảng dạy SE**

Hiện nay

- Công nghiệp hóa sản xuất phần mềm bằng cách đưa những kỹ thuật công nghệ học (Engineering techniques) thành cơ sở khoa học của CNHPM
- Thể chế hóa lý luận trong sản xuất phần mềm và ứng dụng những phương pháp luận một cách nhất quán
- Tăng cường nghiên cứu và tạo công cụ trợ giúp sản xuất phần mềm

3.2 Sự tiến triển của các phương pháp thiết kế phần mềm

- Phương pháp luận trong CNHPM: bắt đầu từ những năm 1970
- Trong phát triển phần mềm: nâng cao năng suất, độ tin cậy, giá thành - tính năng (productivity, reliability, cost-performance)
- Tiến triển phương pháp thiết kế: Sơ khởi, Trưởng thành, Phát triển và Biến đổi

Sơ khởi: nửa đầu 1970

- Khái niệm về tính môđun, cụ thể hóa từng bước trong phương pháp luận thiết kế
- N. Wirth: Chi tiết hóa từng giai đoạn. Thiết kế trên xuống. Lập trình môđun

Trưởng thành: nửa cuối 1970

- Phương pháp luận về quy trình thiết kế phần mềm với phương pháp phân chia môđun và thiết kế trong từng môđun.
- L.L. Constantine, 1974: Thiết kế cấu trúc hóa (phân chia môđun);
- E.W. Dijkstra, 1972: Lập trình cấu trúc hóa (trong môđun) . Phương pháp M.A. Jackson (1975) và J.D. Warnier (1974)
- Trừu tượng hóa dữ liệu: B.H. Liskov (1974);D.L. Parnas (1972)

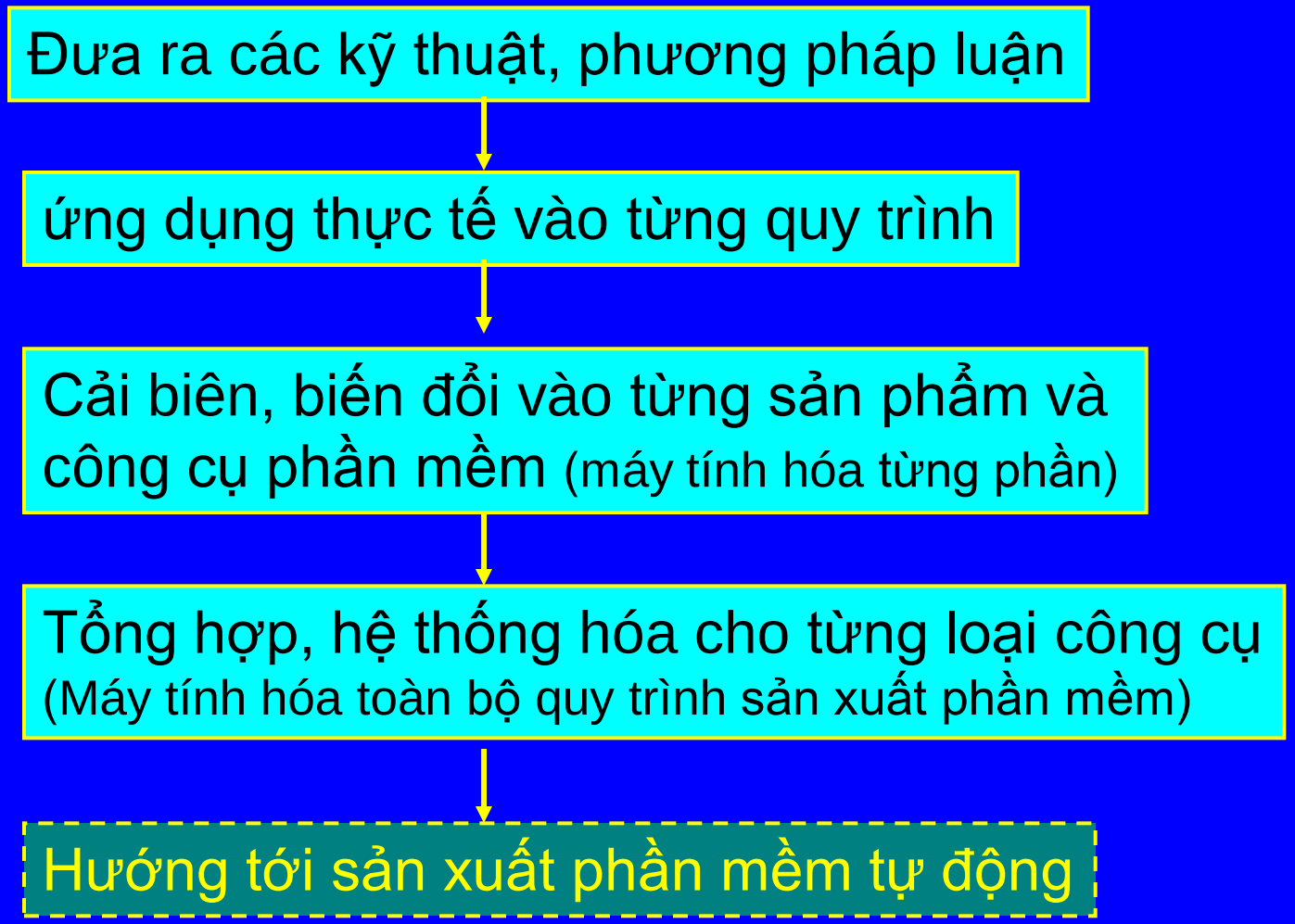
Phát triển: nửa đầu 1980

- Triển khai các công cụ hỗ trợ phát triển phần mềm dựa trên các phương pháp và kỹ thuật đưa ra những năm 1970
- Bộ khởi tạo chương trình (program generators: pre-compiler; graphics-input editors, etc.)
- Ngôn ngữ đối thoại đơn giản (4GL, DB SQL)
- Hệ trợ giúp: Hệ trợ giúp kiểm thử; Hệ trợ giúp quản lý thư viện; Hệ trợ giúp tái sử dụng

Biến đổi: nửa cuối 1980 đến nay

- Đưa ra các môi trường mới về phát triển phần mềm. Triển khai mới về kết hợp giữa CNHPM và CNH Tri thức (Knowledge Engineering)
- Triển khai những môi trường bậc cao về phát triển phần mềm; Tự động hóa sản xuất phần mềm; Chế phần mềm theo kỹ thuật chế thử (Prototyping); Lập trình hướng đối tượng - OOP; Hướng thành phần; Hỗ trợ phát triển phần mềm từ các hệ chuyên gia, vv

Hình thái sản xuất Phần mềm



3.3 Định nghĩa Công nghệ học phần mềm

- **Bauer [1969]: CNHPM là việc thiết lập và sử dụng các nguyên tắc công nghệ học đúng đắn dùng để thu được phần mềm một cách kinh tế vừa tin cậy vừa làm việc hiệu quả trên các máy thực**
- **Parnas [1987]: CNHPM là việc xây dựng phần mềm nhiều phiên bản bởi nhiều người**
- **Ghezzi [1991]: CNHPM là một lĩnh vực của khoa học máy tính, liên quan đến xây dựng các hệ thống phần mềm vừa lớn vừa phức tạp bởi một hay một số nhóm kỹ sư**

Định nghĩa CNHPM (tiếp)

- **IEEE [1993]: CNHPM là**
  - **(1) việc áp dụng phương pháp tiếp cận có hệ thống, bài bản và được lượng hóa trong phát triển, vận hành và bảo trì phần mềm;**
  - **(2) nghiên cứu các phương pháp tiếp cận được dùng trong (1)**
- **Pressman [1995]: CNHPM là bộ môn tích hợp cả quy trình, các phương pháp, các công cụ để phát triển phần mềm máy tính**

Định nghĩa CNHPM (tiếp)

- **Sommerville [1995]: CNHPM là lĩnh vực liên quan đến lý thuyết, phương pháp và công cụ dùng cho phát triển phần mềm**
- **K. Kawamura [1995]: CNHPM là lĩnh vực học vấn về các kỹ thuật, phương pháp luận công nghệ học (lý luận và kỹ thuật được hiện thực hóa trên những nguyên tắc, nguyên lý nào đó) trong toàn bộ quy trình phát triển phần mềm nhằm nâng cao cả chất và lượng của sản xuất phần mềm**

Định nghĩa CNHPM (tiếp)

Công nghệ học phần mềm là lĩnh vực khoa học về các phương pháp luận, kỹ thuật và công cụ tích hợp trong quy trình sản xuất và vận hành phần mềm nhằm tạo ra phần mềm với những chất lượng mong muốn **[Software Engineering is a scientìic field to deal with methodologies, techniques and tools integrated in software production-maintenance process to obtain software with desired qualities]**

Công nghệ học trong CNHPM ?

(1) Như các ngành công nghệ học khác, CNHPM cũng lấy các phương pháp khoa học làm cơ sở

(2) Các kỹ thuật về thiết kế, chế tạo, kiểm thử và bảo trì phần mềm đã được hệ thống hóa hóa thành phương pháp luận và hình thành nên CNHPM

(3) Toàn bộ quy trình quản lý phát triển phần mềm gắn với khái niệm vòng đời phần mềm, được mô hình hóa với những kỹ thuật và phương pháp luận trở thành các chủ đề khác nhau trong CNHPM

Công nghệ học trong CNHPM ? (tiếp)

(4) Trong vòng đời phần mềm không chỉ có chế tạo mà bao gồm cả thiết kế, vận hành và bảo dưỡng (tính quan trọng của thiết kế và bảo dưỡng)

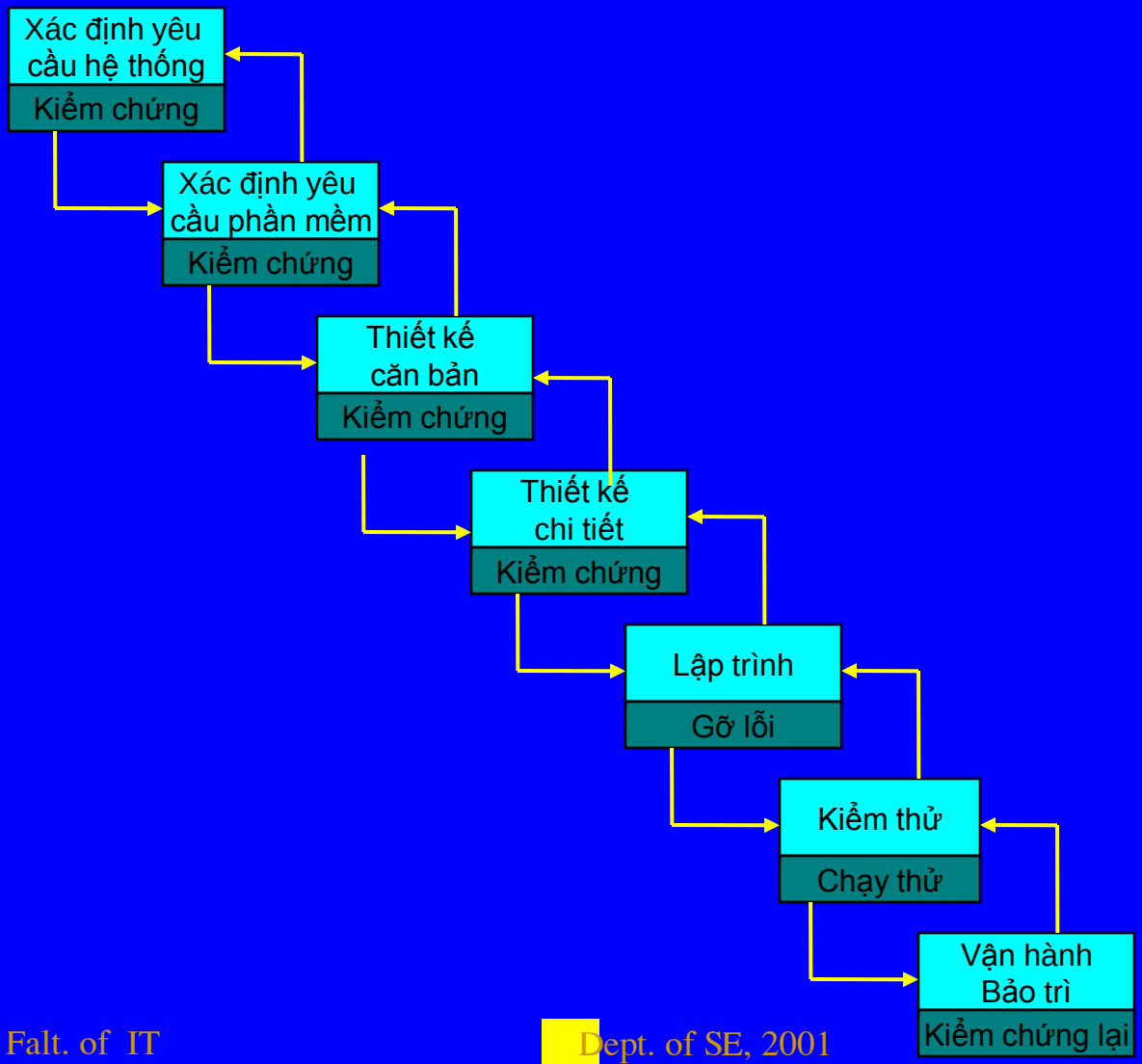
(5) Trong khái niệm phần mềm, không chỉ có chương trình mà cả tư liệu về phần mềm

(6) Cách tiếp cận công nghệ học (khái niệm công nghiệp hóa) thể hiện ở chỗ nhằm nâng cao năng suất (tính năng suất) và độ tin cậy của phần mềm, đồng thời giảm chi phí giá thành

3.4 Vòng đời phần mềm (Software life-cycle)

- Vòng đời phần mềm là thời kỳ tính từ khi phần mềm được sinh (tạo) ra cho đến khi chết đi (từ lúc hình thành đáp ứng yêu cầu, vận hành, bảo dưỡng cho đến khi loại bỏ không đâu dùng)
- Quy trình phần mềm (vòng đời phần mềm) được phân chia thành các pha chính: phân tích, thiết kế, chế tạo, kiểm thử, bảo trì. Biểu diễn các pha có khác nhau theo từng người

Mô hình vòng đời phần mềm của Boehm



Suy nghĩ mới về vòng đời phần mềm

(1) Pha xác định yêu cầu và thiết kế có vai trò quyết định đến chất lượng phần mềm, chiếm phần lớn công sức so với lập trình, kiểm thử và chuyển giao phần mềm

(2) Pha cụ thể hóa cấu trúc phần mềm phụ thuộc nhiều vào suy nghĩ trên xuống (top-down) và trừu tượng hóa, cũng như chi tiết hóa

(3) Pha thiết kế, chế tạo thì theo trên xuống, pha kiểm thử thì dưới lên (bottom-up)

Suy nghĩ mới về vòng đời phần mềm

(4) Trước khi chuyển sang pha kế tiếp phải đảm bảo pha hiện nay đã được kiểm thử không còn lỗi

(5) Cần có cơ chế kiểm tra chất lượng, xét duyệt giữa các pha nhằm đảm bảo không gây lỗi cho pha sau

(6) Tư liệu của mỗi pha không chỉ dùng cho pha sau, mà chính là đối tượng quan trọng cho kiểm tra và đảm bảo chất lượng của từng quy trình và của chính phần mềm

Suy nghĩ mới về vòng đời phần mềm

(7) Cần chuẩn hóa mẫu biểu, cách ghi chép tạo tư liệu cho từng pha, nhằm đảm bảo chất lượng phần mềm

(8) Thao tác bảo trì phần mềm là việc xử lý quay vòng trở lại các pha trong vòng đời phần mềm nhằm biến đổi, sửa chữa, nâng cấp phần mềm

Các phương pháp luận và kỹ thuật cho từng pha

Tên pha	Nội dung nghiệp vụ	Ph- ơng pháp, kỹ thuật
Xác định yêu cầu	Đặc tả yêu cầu ng- ời dùng Xác định yêu cầu phần mềm	Phân tích cấu trúc hóa
Thiết kế hệ thống	Thiết kế cơ bản phần mềm Thiết kế cấu trúc ngoài của phần mềm	Thiết kế cấu trúc hóa
Thiết kế ch- ơng trình	Là thiết kế chi tiết: Thiết kế cấu trúc bên trong của phần mềm (đơn vị ch- ơng trình hoặc môđun)	Lập trình cấu trúc Ph- ơng pháp Jackson Ph- ơng pháp Warnier
Lập trình	Mã hóa bởi ngôn ngữ lập trình	Mã hóa cấu trúc hóa
Đảm bảo chất l- ợng	Kiểm tra chất l- ợng phần mềm đã phát triển	Ph- ơng pháp kiểm thử ch- ơng trình
Vận hành Bảo trì	Sử dụng, vận hành phần mềm đã phát triển. Biến đổi, điều chỉnh phần mềm	Ch- a cụ thể

3.5 Quy trình phát triển phần mềm

3.5.1 Capability Maturity Model (CMM) by SEI: Mô hình thuần thục khả năng

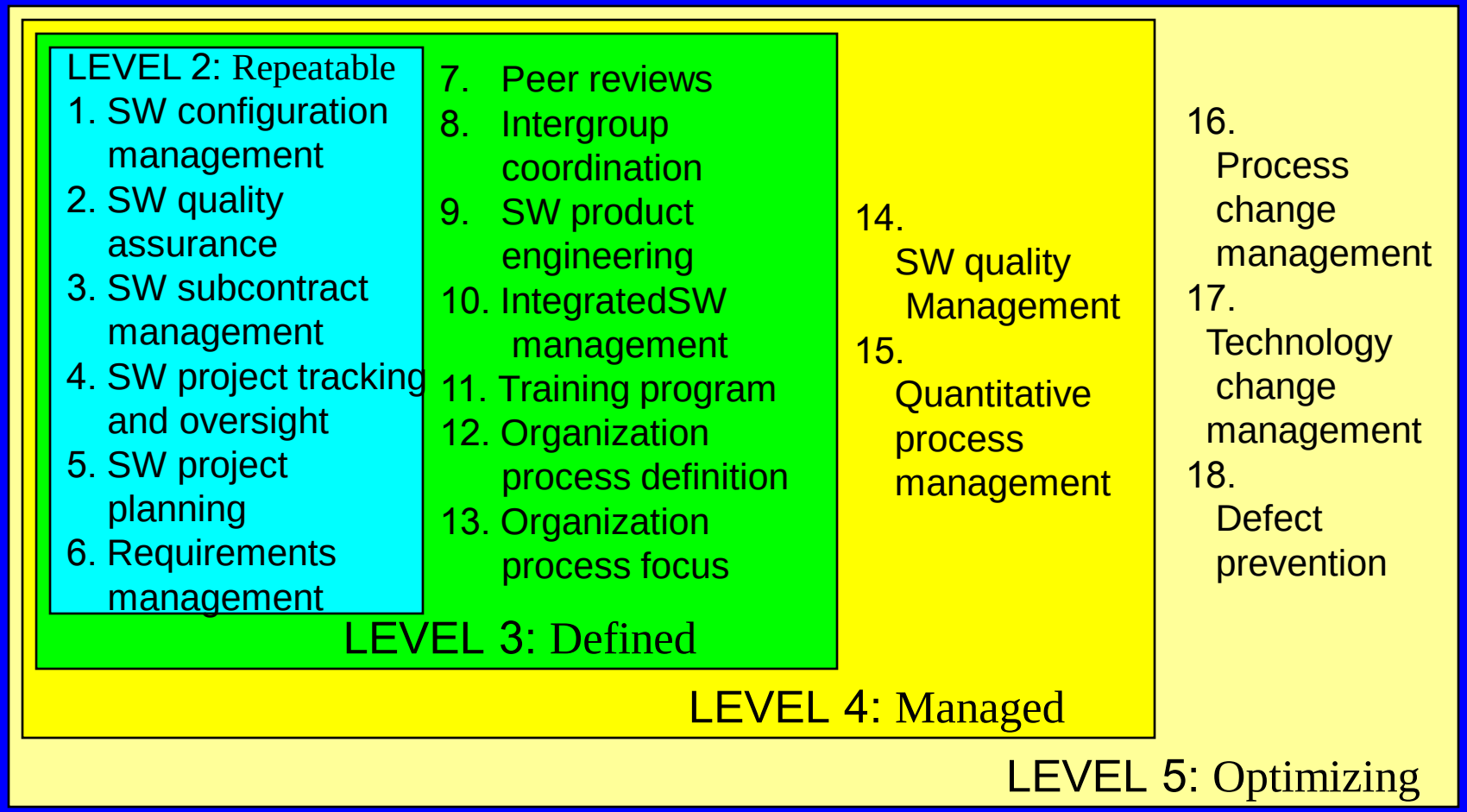
- **Level 1: Initial (Khởi đầu). Few processes are defined. Success depends on individual effort**
- **Level 2: Repeatable (Lặp lại). Basic project management processes. Repeat earlier succeses on projects with similar applications**
- **Level 3: Defined (Xác định). Use a documented and approved version of the organization's process for developing and supporting software**

CMM (cont.)

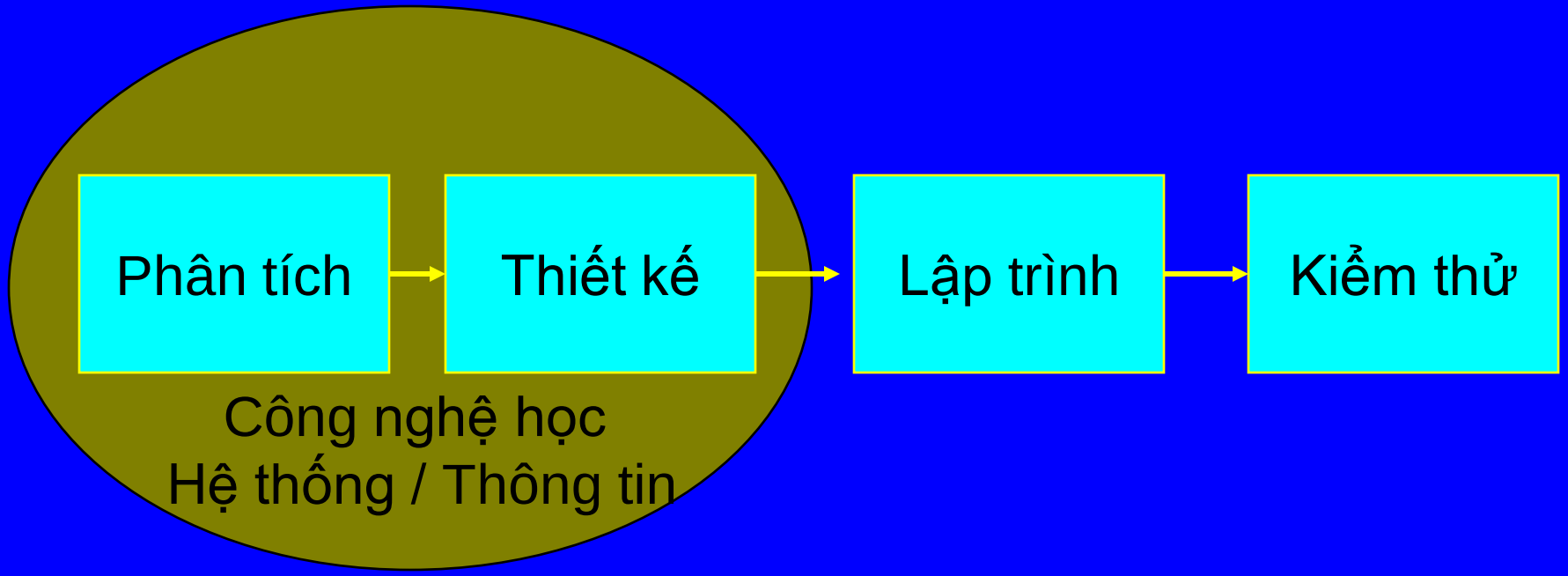
- **Level 4: Managed (Quản trị). Both SW process and products are quantitatively understood and controlled using detailed measures**
- **Level 5: Optimizing (Tối ưu). Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies**

18 key process areas (KPAs) for CMM

18 KPAs of CMM



3.5.2 Mô hình tuyến tính



Điển hình là mô hình vòng đời cổ điển (mô hình thác nước) Classic life cycle / waterfall model: là mô hình hay đựoc dùng nhất

Mô hình tuyến tính

- **Công nghệ học Hệ thống / Thông tin và mô hình hóa (System / Information engineering and modeling): thiết lập các yêu cầu, ánh xạ một số tập con các yêu cầu sang phần mềm trong quá trình tương tác giữa phần cứng, người và CSDL**
- **Phân tích yêu cầu (Requirements analysis): hiểu lĩnh vực thông tin, chức năng, hành vi, tính năng và giao diện của phần mềm sẽ phát triển. Cần phải tạo tư liệu và bàn thảo với khách hàng, người dùng**

Mô hình tuyến tính

- **Thiết kế (Design): là quá trình nhiều bước với 4 thuộc tính khác nhau của một chương trình: cấu trúc dữ liệu, kiến trúc phần mềm, biểu diễn giao diện và chi tiết thủ tục (thuật toán). Cần tư liệu hóa và là một phần quan trọng của cấu hình phần mềm**
- **Tạo mã / lập trình (Code generation / programming): Chuyển thiết kế thành chương trình máy tính bởi ngôn ngữ nào đó. Nếu thiết kế đã được chi tiết hóa thì lập trình có thể chỉ thuần túy cơ học**

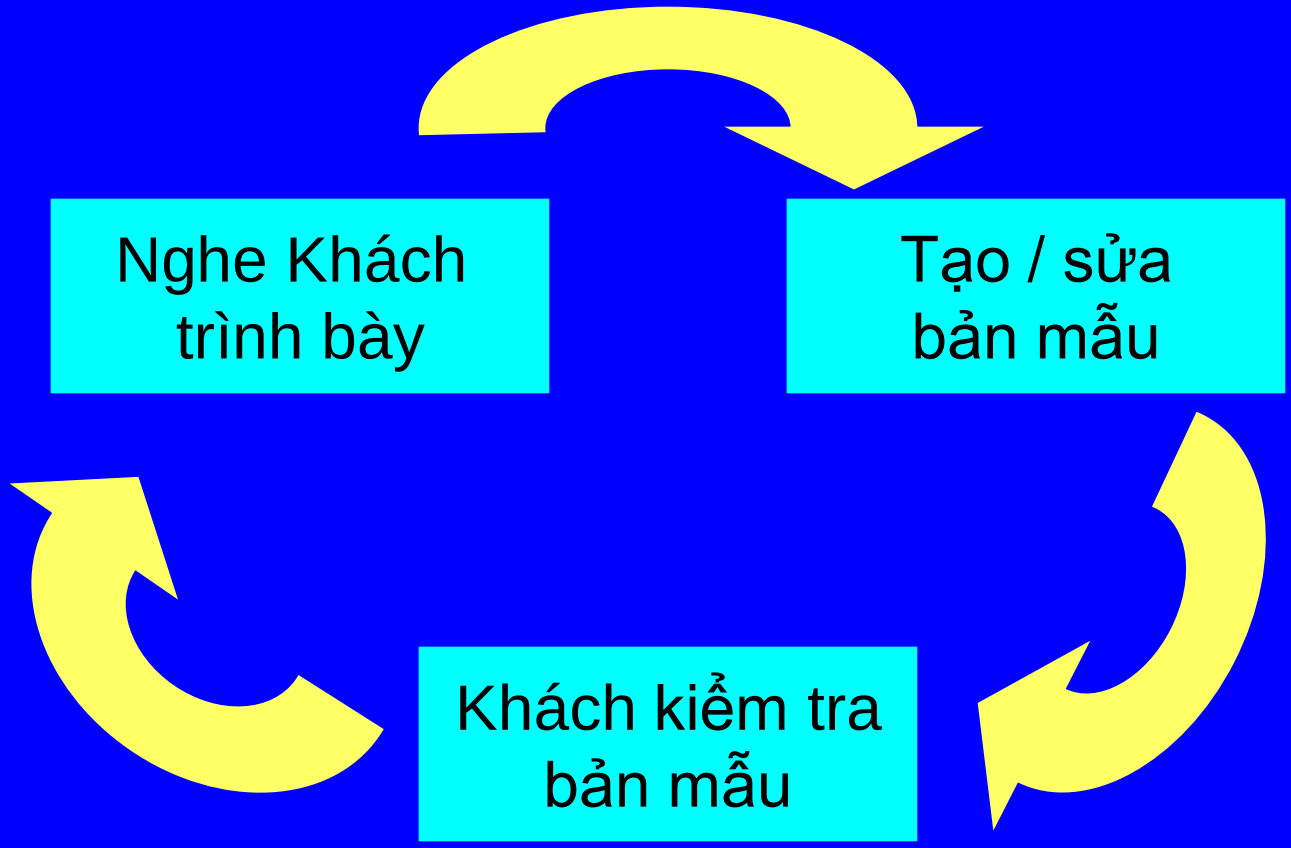
Mô hình tuyến tính

- **Kiểm thử (Testing): Kiểm tra các chương trình và môđun cả về lôgic bên trong và chức năng bên ngoài, nhằm phát hiện ra lỗi và đảm bảo với đầu vào xác định thì cho kết quả mong muốn**
- **Hỗ trợ / Bảo trì (Support / Maintenance): Đáp ứng những thay đổi, nâng cấp phần mềm đã phát triển do sự thay đổi của môi trường, nhu cầu**

Điểm yếu của Mô hình tuyến tính

- Thực tế các dự án ít khi tuân theo dòng tuần tự của mô hình, mà thường có lặp lại (như mô hình của Boehm)
- Khách hàng ít khi tuyên bố rõ ràng khi nào xong hết các yêu cầu
- Khách hàng phải có lòng kiên nhẫn chờ đợi thời gian nhất định mới có sản phẩm. Nếu phát hiện ra lỗi nặng thì là một thảm họa!

3.5.3 Mô hình chế thử (Prototyping model)



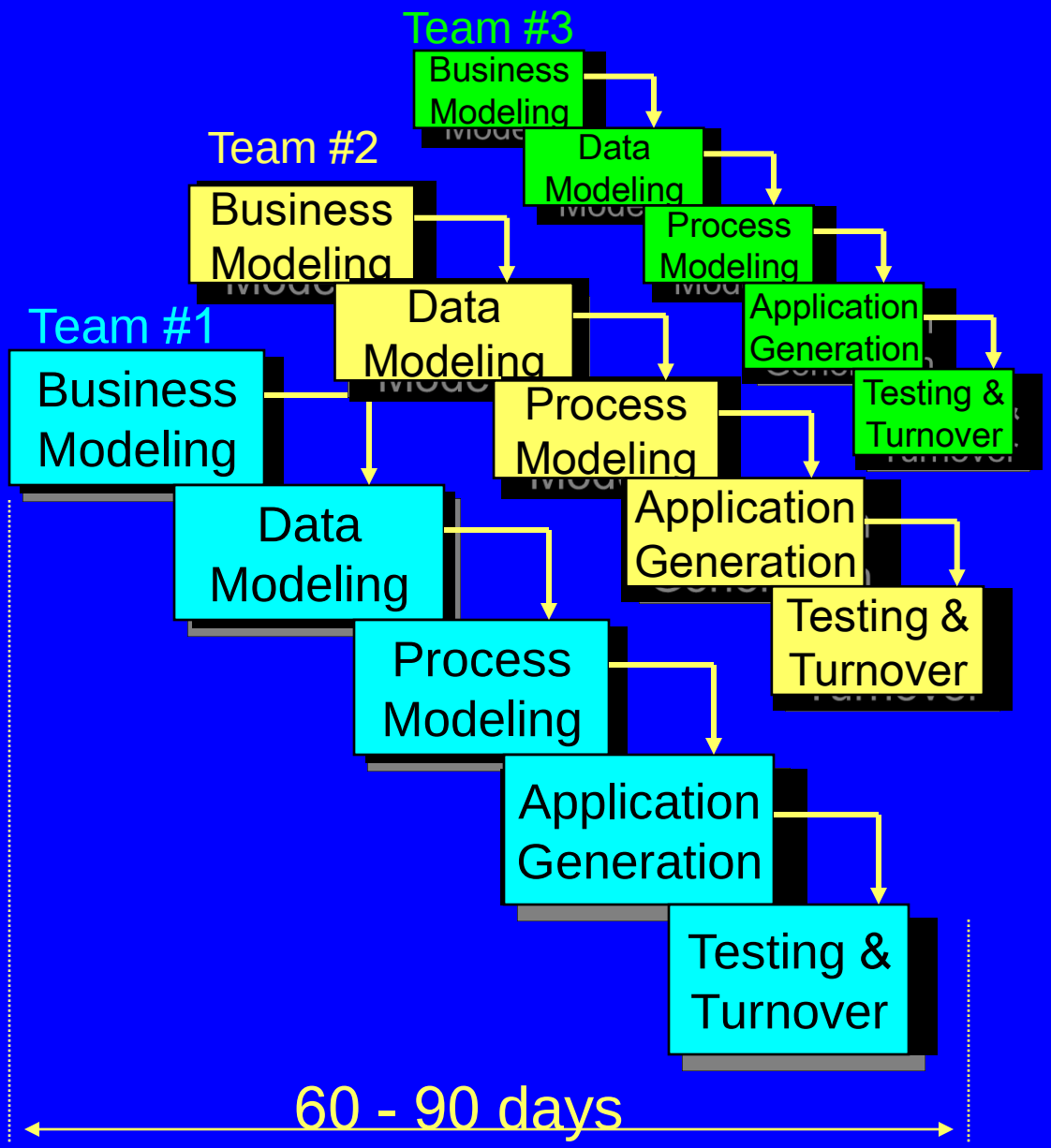
Mô hình chế thử: Khi nào ?

- Khi mới rõ mục đích chung chung của phần mềm, chưa rõ chi tiết đầu vào hay xử lý ra sao hoặc chưa rõ yêu cầu đầu ra
- Dùng như "Hệ sơ khai" để thu thập yêu cầu người dùng qua các thiết kế nhanh
- Các giải thuật, kỹ thuật dùng làm bản mẫu có thể chưa nhanh, chưa tốt, miễn là có mẫu để thảo luận gợi yêu cầu của người dùng

3.5.4 Mô hình phát triển ứng dụng nhanh (Rapid Application Development: RAD)

- Là quy trình phát triển phần mềm gia tăng, tăng dần từng bước (Incrimental software development) với mỗi chu trình phát triển rất ngắn (60-90 ngày)
- Xây dựng dựa trên hướng thành phần (Component-based construction) với khả năng tái sử dụng (reuse)
- Gồm một số nhóm (teams), mỗi nhóm làm 1 RAD theo các pha: Mô hình nghiệp vụ, Mô hình dữ liệu, Mô hình xử lý, Tạo ứng dụng, Kiểm thử và đánh giá (Business, Data, Process, Appl. Generation, Test)

Mô hình phát triển ứng dụng nhanh



RAD: Business modeling

Luồng thông tin được mô hình hóa để trả lời các câu hỏi:

- **Thông tin nào điều khiển xử lý nghiệp vụ ?**
- **Thông tin gì được sinh ra?**
- **Ai sinh ra nó ?**
- **Thông tin đi đến đâu ?**
- **Ai xử lý chúng ?**

RAD: Data and Process modeling

- **Data modeling:** các đối tượng dữ liệu cần để hỗ trợ nghiệp vụ (business). Định nghĩa các thuộc tính của từng đối tượng và xác lập quan hệ giữa các đối tượng
- **Process modeling:** Các đối tượng dữ liệu được chuyển sang luồng thông tin thực hiện chức năng nghiệp vụ. Tạo mô tả xử lý để cập nhật (thêm, sửa, xóa, khôi phục) từng đối tượng dữ liệu

RAD: Appl. Generation and Testing

- **Application Generation:** Dùng các kỹ thuật thế hệ 4 để tạo phần mềm từ các thành phần có sẵn hoặc tạo ra các thành phần có thể tái dụng lại sau này. Dùng các công cụ tự động để xây dựng phần mềm
- **Testing and Turnover:** Kiểm thử các thành phần mới và kiểm chứng mọi giao diện (các thành phần cũ đã được kiểm thử và dùng lại)

RAD: Hạn chế ?

- Cần nguồn nhân lực dồi dào để tạo các nhóm cho các chức năng chính
- Yêu cầu hai bên giao kèo trong thời gian ngắn phải có phần mềm hoàn chỉnh, thiếu trách nhiệm của một bên dễ làm dự án đổ vỡ
- RAD không phải tốt cho mọi ứng dụng, nhất là với ứng dụng không thể môđun hóa hoặc đòi hỏi tính năng cao
- Mạo hiểm kỹ thuật cao thì không nên dùng RAD

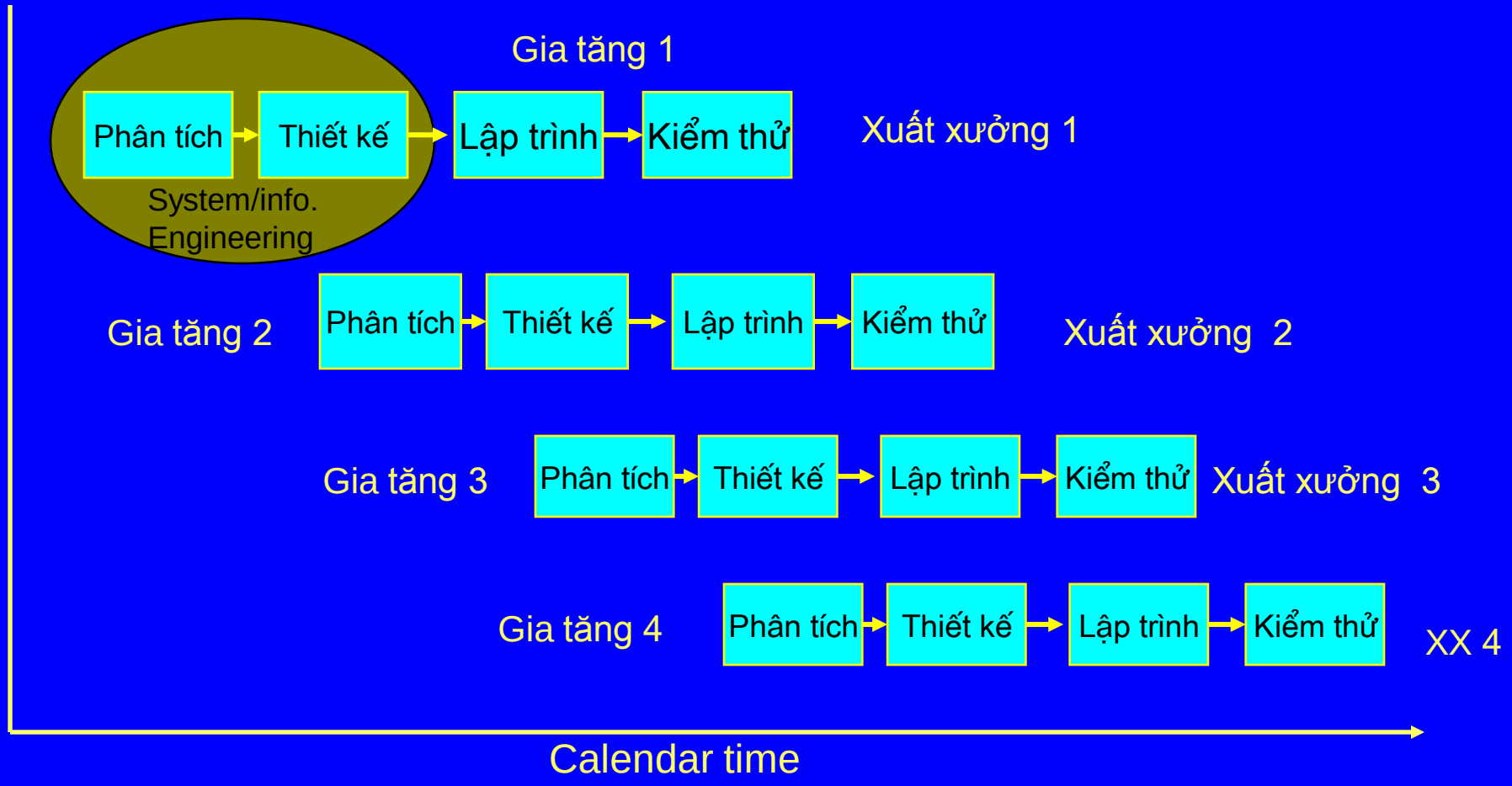
3.5.5 Các mô hình tiến hóa: gia tăng, xoắn ốc, xoắn WINWIN, ...

- Phần lớn các hệ phần mềm phức tạp đều tiến hóa theo thời gian: môi trường thay đổi, yêu cầu phát sinh thêm, hoàn thiện thêm chức năng, tính năng
- Các mô hình tiến hóa (evolutionary models) có tính lặp lại. Kỹ sư phần mềm tạo ra các phiên bản (versions) ngày càng hoàn thiện hơn, phức tạp hơn
- Các mô hình: incremental, spiral, WINWIN spiral, concurrent development model

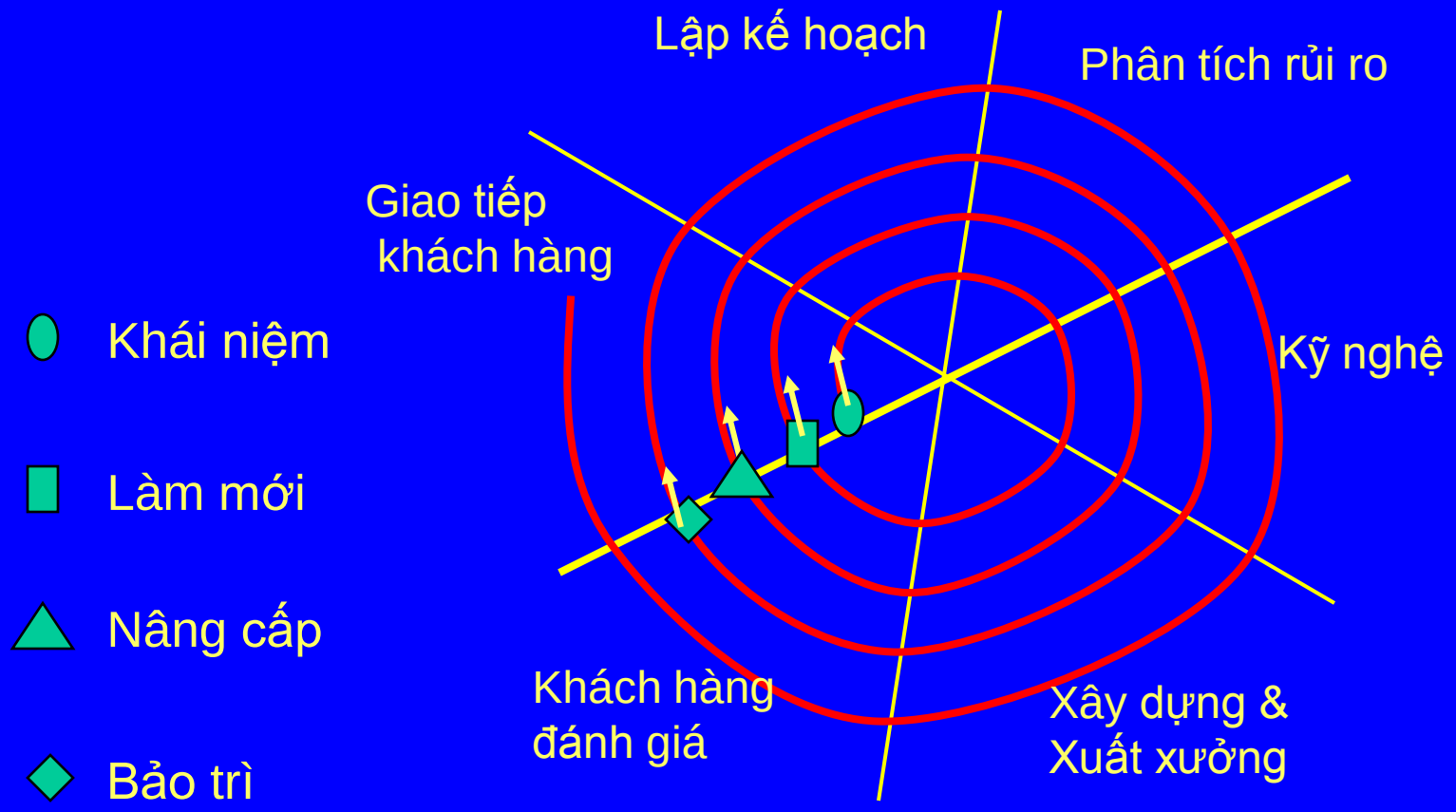
Mô hình gia tăng (The incremental model)

- Kết hợp mô hình tuần tự và ý tưởng lặp lại của chế bản mẫu
- Sản phẩm lõi với những yêu cầu cơ bản nhất của hệ thống được phát triển
- Các chức năng với những yêu cầu khác được phát triển thêm sau (gia tăng)
- Lặp lại quy trình để hoàn thiện dần

Mô hình gia tăng



Mô hình xoắn ốc (spiral)



Mô hình xoắn ốc (tiếp)

- Giao tiếp khách hàng: giữa người phát triển và khách hàng để tìm hiểu yêu cầu, ý kiến
- Lập kế hoạch: Xác lập tài nguyên, thời hạn và những thông tin khác
- Phân tích rủi ro: Xem xét mạo hiểm kỹ thuật và mạo hiểm quản lý
- Kỹ nghệ: Xây dựng một hay một số biểu diễn của ứng dụng

Mô hình xoắn ốc (tiếp)

- Xây dựng và xuất xưởng: xây dựng, kiểm thử, cài đặt và cung cấp hỗ trợ người dùng (tư liệu, huấn luyện, . . .)
- Đánh giá của khách hàng: Nhận các phản hồi của người sử dụng về biểu diễn phần mềm trong giai đoạn kỹ nghệ và cài đặt

Mô hình xoắn ốc: Mạnh và yếu?

- Tốt cho các hệ phần mềm quy mô lớn
- Dễ kiểm soát các mạo hiểm ở từng mức tiến hóa
- Khó thuyết phục khách hàng là phương pháp tiến hóa xoắn ốc có thể kiểm soát được
- Chưa được dùng rộng rãi như các mô hình tuyến tính hoặc chế thử

Mô hình xoắn ốc WINWIN

- Nhằm thỏa hiệp giữa người phát triển và khách hàng, cả hai cùng "Thắng" (win-win)
  - Khách thì có phần mềm thỏa mãn yêu cầu chính
  - Người phát triển thì có kinh phí thỏa đáng và thời gian hợp lý
- Các hoạt động chính trong xác định hệ thống:
  - Xác định cổ đông (stakeholders)
  - Xác định điều kiện thắng của cổ đông
  - Thỏa hiệp điều kiện thắng của các bên liên quan

Mô hình xoắn ốc WINWIN

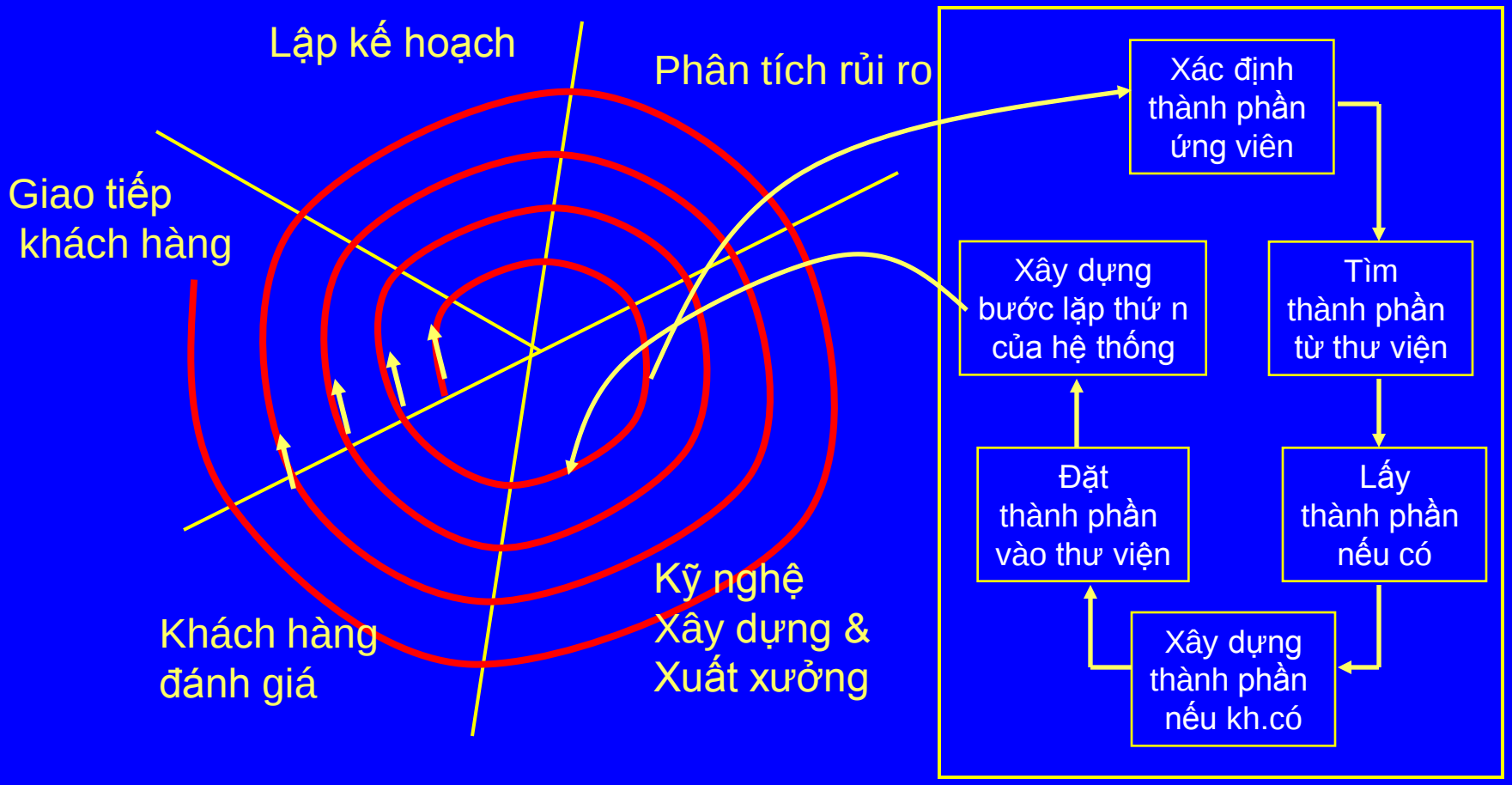
Mô hình phát triển đồng thời (The concurrent development model)

- Xác định mạng lưới những hoạt động đồng thời (Network of concurrent activities)
- Các sự kiện (events) xuất hiện theo điều kiện vận động trạng thái trong từng hoạt động
- Dùng cho mọi loại ứng dụng và cho hình ảnh khá chính xác về trạng thái hiện trạng của dự án
- Thường dùng trong phát triển các ứng dụng khách/chủ (client/server applications): system and componets are developed concurrently

3.5.6 Mô hình theo thành phần (Component-based model)

- Gắn với những công nghệ hướng đối tượng (Object-oriented technologies) qua việc tạo các lớp (classes) có chứa cả dữ liệu và giải thuật xử lý dữ liệu
- Có nhiều tương đồng với mô hình xoắn ốc
- Với ưu điểm tái sử dụng các thành phần qua Thư viện / kho các lớp: tiết kiệm 70% thời gian, 80% giá thành, chỉ số sản xuất 26.2/16.9
- Với UML như chuẩn công nghiệp đang triển khai

Mô hình theo thành phần



3.5.7 Mô hình hình thức (Formal model)

- Còn gọi là CNHPM phòng sạch (Cleanroom SE)
- Tập hợp các công cụ nhằm đặc tả toán học phần mềm máy tính từ khâu định nghĩa, phát triển đến kiểm chứng
- Giúp kỹ sư phần mềm phát hiện và sửa các lỗi khó
- Thường dùng trong phát triển SW cần độ an toàn rất cao (y tế, hàng không, . . .)

Mô hình hình thức: Điểm yếu ?

- Cần nhiều thời gian và công sức để phát triển
- Phí đào tạo cao vì ít người có nền căn bản cho áp dụng mô hình hình thức
- Khó sử dụng rộng rãi vì cần kiến thức toán và kỹ năng của khách hàng

3.5.8 Các kỹ thuật thế hệ 4 (Fourth generation techniques)

- Tập hợp các công cụ cho phép xác định đặc tính phần mềm ở mức cao, sau đó sinh tự động mã nguồn dựa theo đặc tả đó
- Các công cụ 4GT điển hình: ngôn ngữ phi thủ tục cho truy vấn CSDL; tạo báo cáo; xử lý dữ liệu; tương tác màn hình; tạo mã nguồn; khả năng đồ họa bậc cao; khả năng bảng tính; khả năng giao diện Web; vv

4GT: How ?

- Từ thu thập yêu cầu cho đến sản phẩm: đối thoại giữa khách và người phát triển là quan trọng
- Không nên bỏ qua khâu thiết kế. 4GT chỉ áp dụng để triển khai thiết kế qua 4GL
- Mạnh: giảm thời gian phát triển và tăng năng suất
- Yếu: 4GT khó dùng hơn ngôn ngữ lập trình, mã khó tối ưu và khó bảo trì cho hệ thống lớn [redacted]n kỹ năng của kỹ sư phần mềm
- Tương lai: 4GT với mô hình theo thành phần

3.5.9 Sản phẩm và quy trình (Product and process)

- Quy trình yếu thì sản phẩm khó mà tốt, song không nên coi trọng quá mức vào quy trình hoặc quá mức vào sản phẩm
- Sản phẩm và quy trình cần được coi trọng như nhau

Bài tập Phần I và Đồ án I

- Xem lại các khái niệm, mô hình của phần mềm và CNHPM
- Đồ án môn học I (cho 13 nhóm, nạp báo cáo, tư liệu tìm được trên Web và thư viện):
  - Tìm hiểu và viết báo cáo, trình bày về mô hình phát triển phần mềm (10 mô hình / 10 nhóm)
  - Chuẩn ISO 9001 cho SE
  - Chuẩn CMM (www.sei.com)
  - Các kỹ thuật lập trình (cấu trúc, mô đun, . . .)