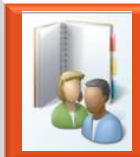


Phần I: Giới thiệu chung về CNPM

Chương 1: Bản chất phần mềm

1.1 Định nghĩa chung về phần mềm

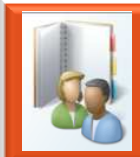
1.2 Kiến trúc phần mềm

1.3 Các khái niệm

1.4 Đặc tính chung của phần mềm

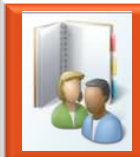
1.5 Thế nào là phần mềm tốt?

1.6 Các ứng dụng phần mềm



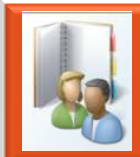
1.1. Định nghĩa chung về phần mềm

- **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 phần mềm ngày càng cao và nay cao hơn phần cứng



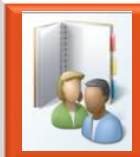
Các đặc tính của SW và HW

Hardware	Software
• 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



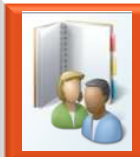
Định nghĩa 1: Phần mềm là

- 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

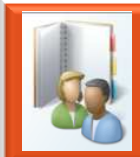
- Vai trò SW ngày càng thể hiện trội
- Máy tính là . . . chiếc hộp không có SW
- Ngày nay, SW quyết định chất lượng một hệ thống máy tính (HTMT), là chủ đề cốt lõi, trung tâm của HTMT
- Hệ thống máy tính gồm HW và SW



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

- 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

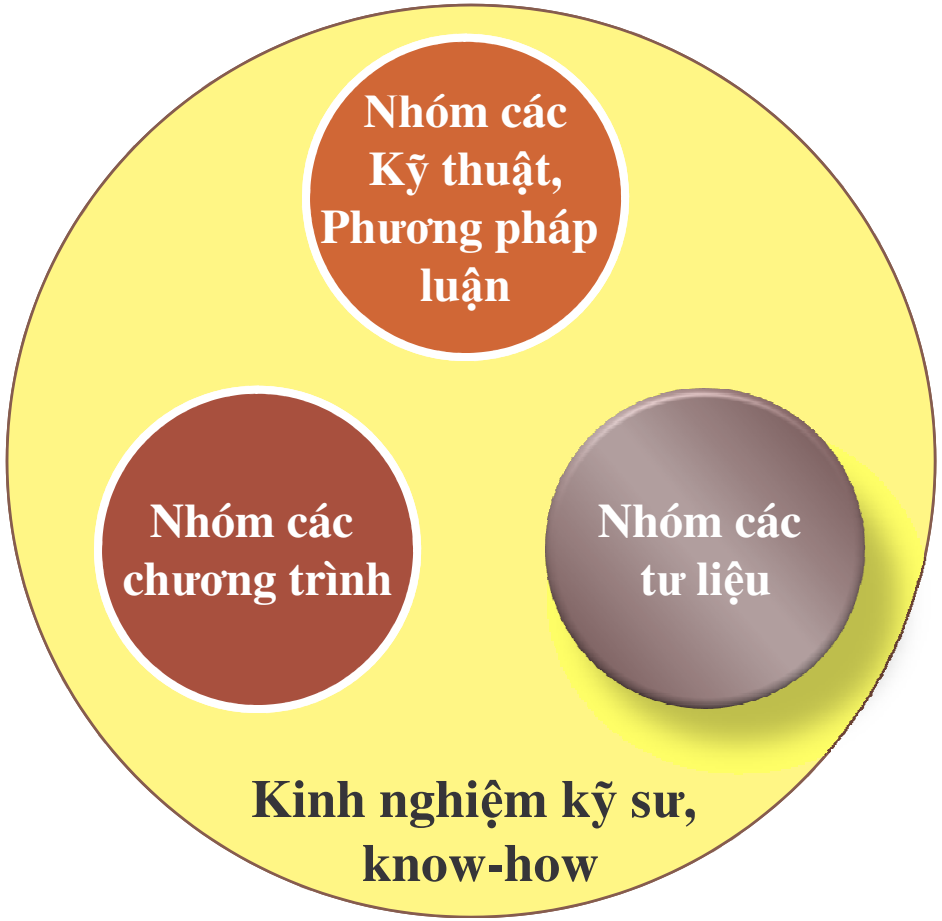


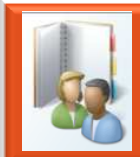
SW theo nghĩa rộng

- 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

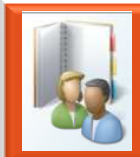
Phần mềm là gì ?





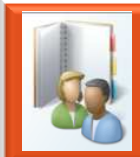
Nhóm các kỹ thuật, phương pháp luận

- 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



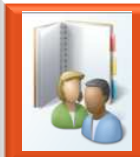
Nhóm các chương trình

- 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, . . .



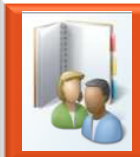
Nhóm các tư liệu

- 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

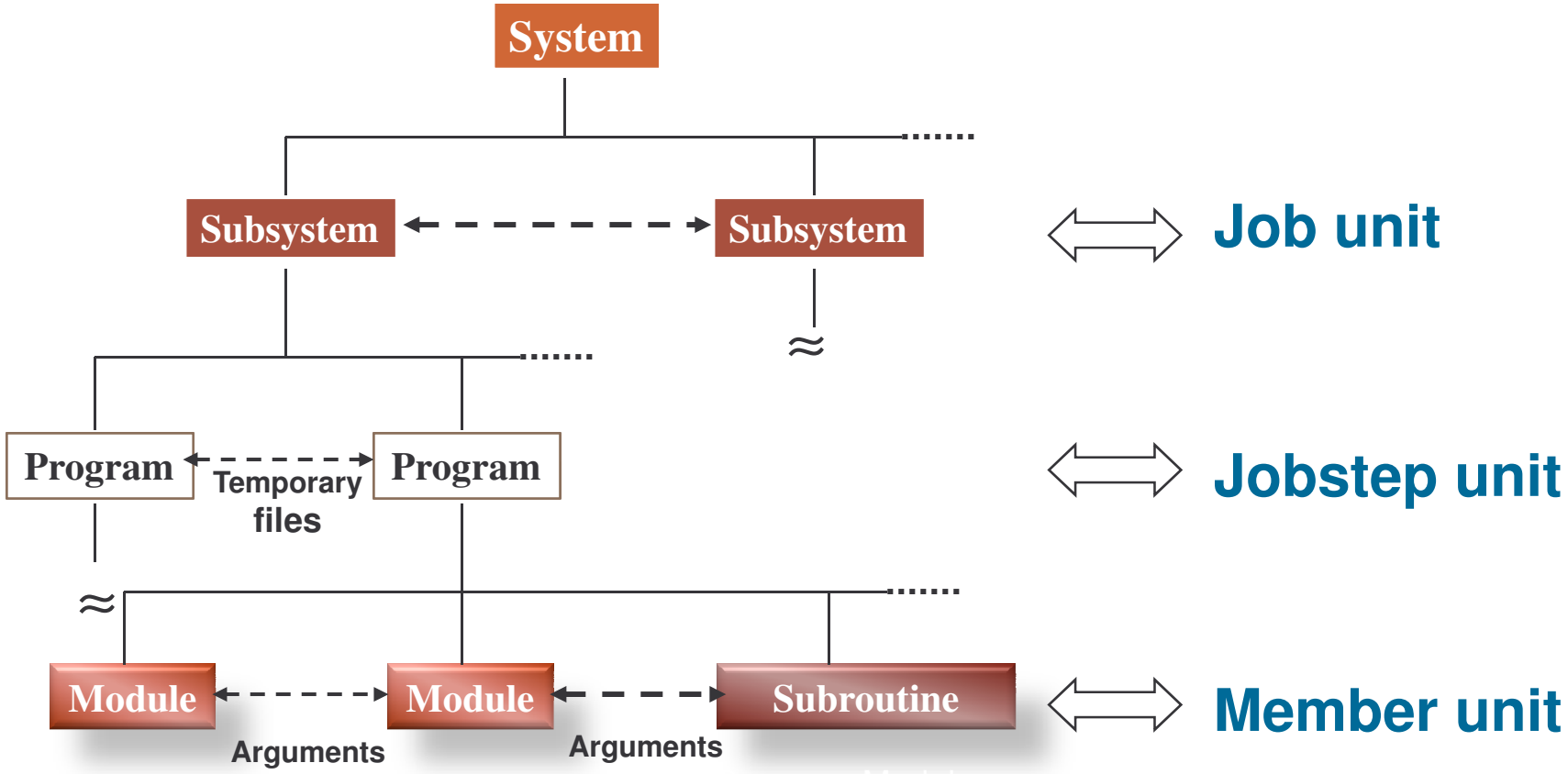


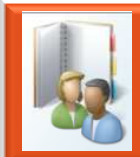
1.2 Kiến trúc phần mềm

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 (Program)
- Dưới chương trình là các Modules hoặc Subroutines với các đối số (Arguments)

Kiến trúc phần mềm

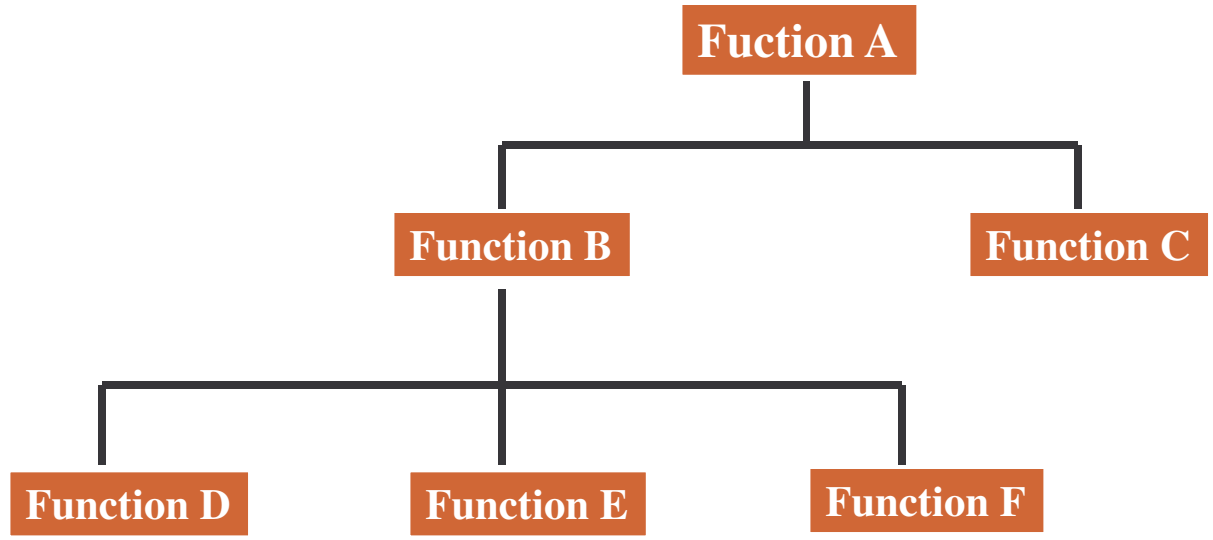




1.2.2 Phần mềm nhìn từ cấu trúc và thủ tục

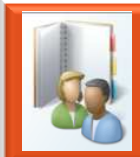
- Hai yếu tố cấu thành của phần mềm
  - Phương diện cấu trúc
  - Phương diện thủ tục
- **Cấu trúc phần mềm**: biểu thị kiến trúc các chức năng mà phần mềm đó có và điều kiện phân cấp các chức năng (thiết kế cấu trúc)
- **Thiết kế chức năng**: theo chiều đứng (càng sâu càng phức tạp) và chiều ngang (càng rộng càng nhiều chức năng, qui mô càng lớn)

Cấu trúc phần mềm



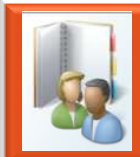
Cấu trúc chiều đứng
(Vertical structure)

Cấu trúc chiều ngang
(Horizontal structure)



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]

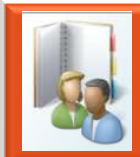


1.3 Các khái niệm

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

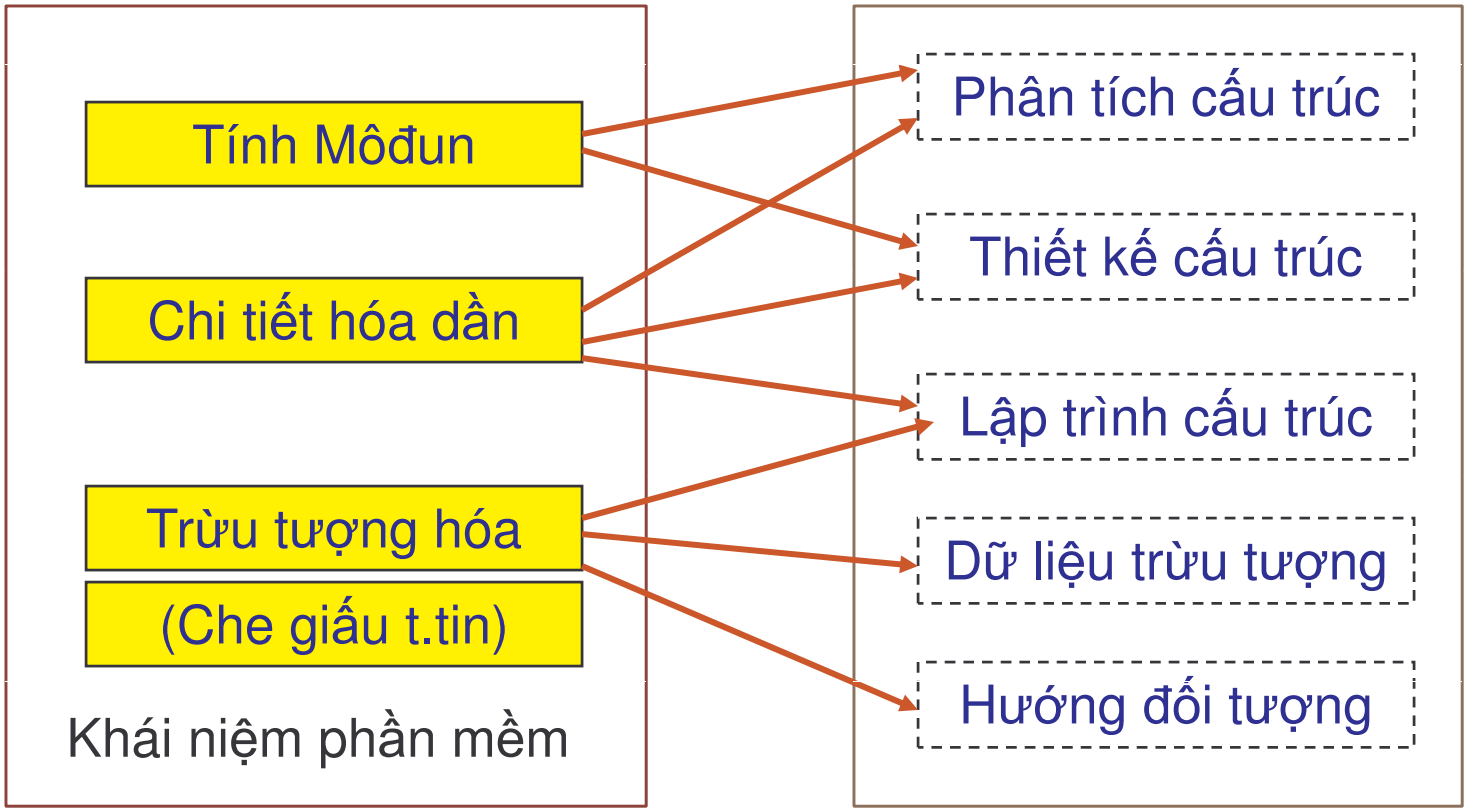
Từ phương pháp luận triển khai đến kỹ thuật

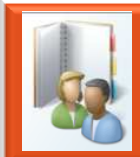


Các khái niệm (Software concepts)

- 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

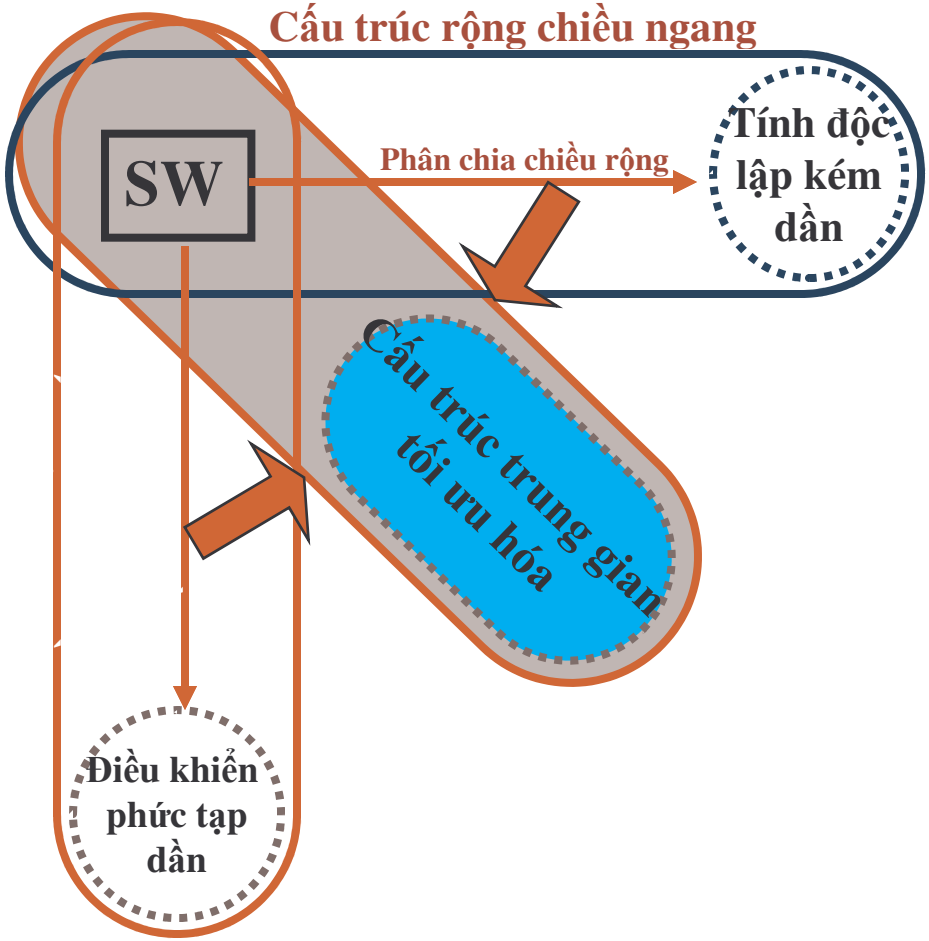




1.3.1 Tính môđun (Modularity)

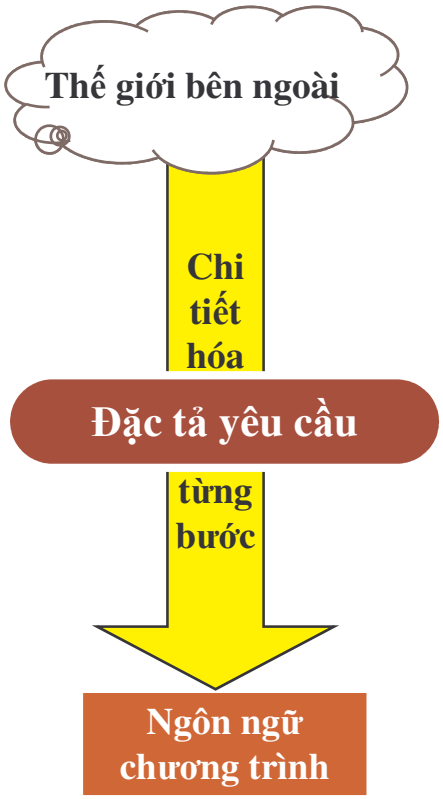
- Là khả năng phân chia phần mềm thành các môđun ứng với các chức năng, đồng thời cho phép quản lý tổng thể: khái niệm phân chia và trộn (partion and merge)
- Hai phương pháp phân chia môđun theo chiều
  - **Sâu (depth, thẳng đứng):** điều khiển phức tạp dần
  - **Rộng (width, nằm ngang):** môđun phụ thuộc dần
- Quan hệ giữa các môđun: qua các đối số (arguments)

Chuẩn phân chia môđun



1.3.2 Chi tiết hóa từng bước

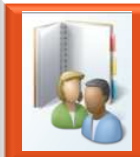
Cách tiếp cận từ trên xuống (top-down approach)



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

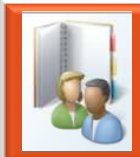


Ví dụ: Trình tự giải quyết vấn đề từ mức thiết kế chương trình đến mức lập trình

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 (phương pháp nhị phân)



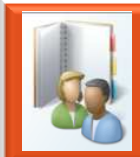
Chương 2:

Khủng hoảng phần mềm

(SOFTWARE CRISIS)

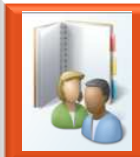
2.1 Khủng hoảng phần mềm là gì ?

2.2 Những vấn đề (khó khăn) trong sản xuất phần mềm



2.1 Khủng hoảng phần mềm là gì?

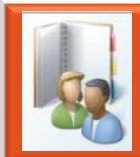
- 10/1968 tại Hội nghị của NATO các chuyên gia phần mềm đã đưa ra thuật ngữ "**Khủng hoảng phần mềm**" (Software crisis). Qua hàng chục năm, thuật ngữ này vẫn được dùng và ngày càng mang tính cấp bách
- **Khủng hoảng là gì ?** [Webster's Dict.]
  - Điểm ngoặt trong tiến trình của bất kỳ cái gì; thời 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)

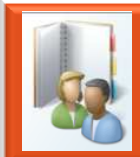
Là sự day dứt kinh niên (kéo dài theo thời gian hoặc thường tái diễn, liên tục không kết thúc) gặp phải trong phát triển phần mềm máy tính, như

- 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 p/mềm gây ra những vấn đề xã hội?



Một số yếu tố

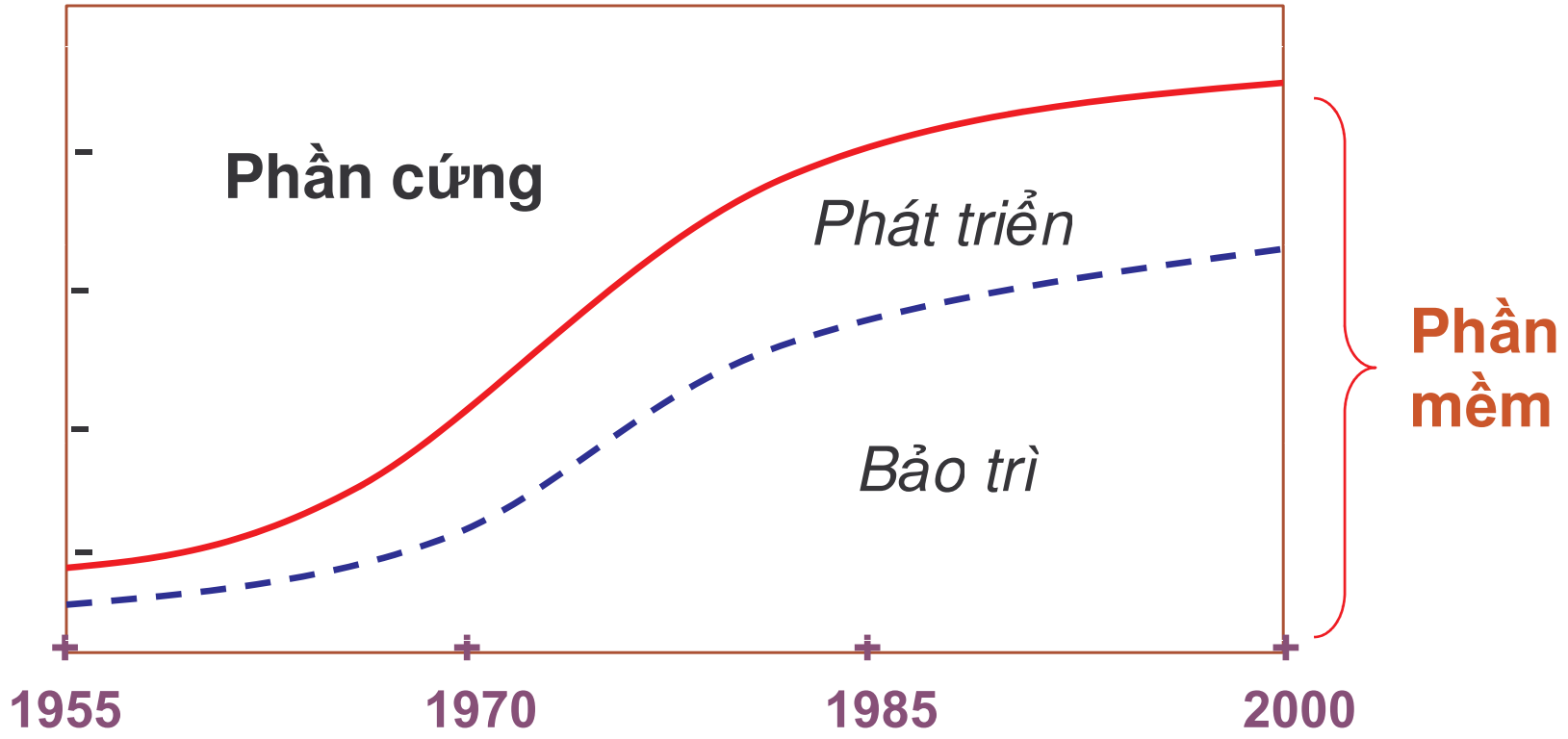
- Phần mềm càng lớn sẽ kéo theo phức tạp hóa và tăng chi phí phát triển
- Đổi vai trò giá thành SW vs. HW
- Công sức cho bảo trì càng tăng thì chi phí cho Backlog càng lớn
- Nhân lực chưa đáp ứng được nhu cầu phần mềm
- Những phiền hà của phần mềm gây ra những vấn đề xã hội



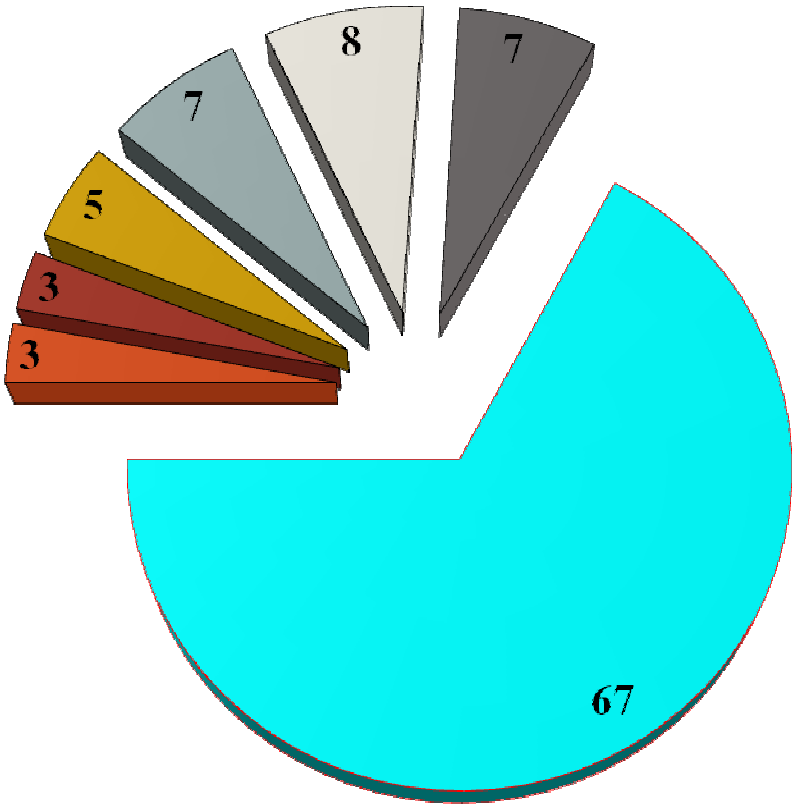
Những dự án lớn của NASA (National Aeronautics and Space Administration)

Tên dự án	Thời điểm phát triển	Tổng số bước (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

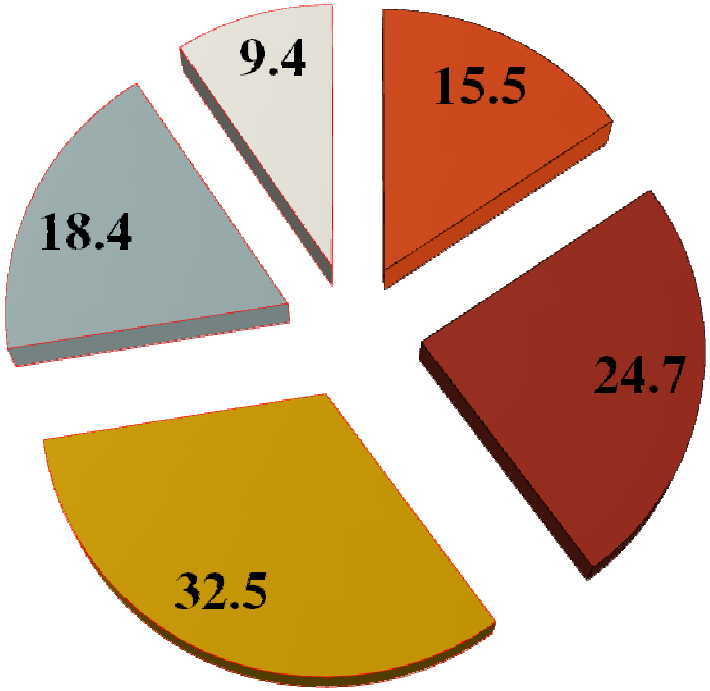


So sánh chi phí cho các pha

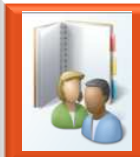


- Xác định yêu cầu 3%
- Đặc tả 3%
- Thiết kế 5%
- Lập trình 7%
- Kiểm thử môđun 8%
- Kiểm thử tích hợp 7%
- Bảo trì 67%

Backlog tại Nhật Bản năm 1985



- Dưới 6 tháng 15.5%
- 6 tháng đến 1 năm 24.7%
- Từ 1 đến 2 năm 32.5%
- Từ 2 đến 3 năm 18.4%
- Trên 3 năm 9.4%

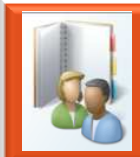


Những vấn đề (khó khăn) trong sản xuất PM

(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

(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)



Những vấn đề trong sản xuất phần mềm (tiếp)

(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

(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

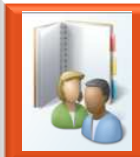
(7) Nếu coi thường việc tái sử dụng phần mềm (software reuse), thì năng suất lao động sẽ giảm

(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

(11) Khi đầu tư nhân lực lớn vào bảo trì sẽ làm giảm hiệu suất lao động của nhân viên



Những vấn đề trong sản xuất phần mềm (tiếp)

(12) Công việc bảo trì kéo dài làm giảm chất lượng của tư liệu và ảnh hưởng xấu đến những việc khác

(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!

CÔNG NGHỆ HỌC PHẦN MỀM

(Software Engineering)

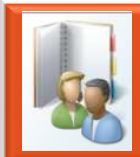
3.1 Lịch sử tiến triển Công nghệ học phần mềm

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



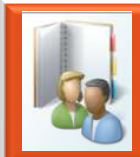
3.1 Lịch sử tiến triển của CNHPM

- **Nửa đầu 1960:**

  Ít quan tâm đến phần mềm, chủ yếu tập trung nâng cao tính năng và độ tin cậy của phần cứng

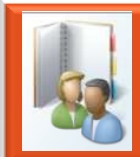
- **Giữa những năm 1960**:

  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



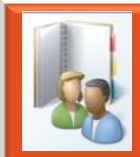
Lịch sử tiến triển của CNHPM (tiếp)

- **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



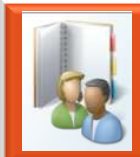
Lịch sử tiến triển của CNHPM (tiếp)

- **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)**



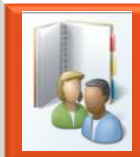
Lịch sử tiến triển của CNHPM (tiếp)

- **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



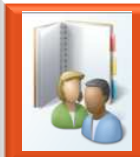
Lịch sử tiến triển của CNHPM (tiếp)

- **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)



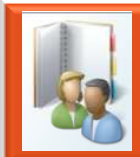
Lịch sử tiến triển của CNHPM (tiếp)

- **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



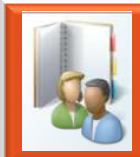
Hiện nay

- 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



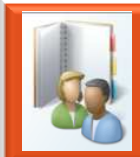
3.2 Sự tiến triển của các phương pháp thiết kế PM

- Phương pháp luận trong CNHPM: bắt đầu từ những năm 1970
- Trong phát triển phần mềm:
  - Nâng cao năng suất (productivity)
  - Độ tin cậy (Reliability)
  - Giá thành - tính năng (Cost-performance)
- Tiến triển phương pháp thiết kế:
  - Sơ khởi, Trưởng thành,
  - Phát triển và Biến đổi



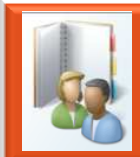
Sơ khởi: nửa đầu 1970

- Khái niệm về tính môđun, cụ thể hóa từng bước trong phương pháp luận thiết kế ra đời
- **N. Wirth**: Chi tiết hóa từng giai đoạn.
  - Thiết kế trên xuống.
  - Lập trình môđun



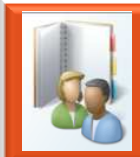
Trưởng thành: nửa cuối 1970

- 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)**



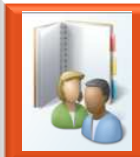
Phát triển: nửa đầu 1980

- 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

Đưa ra các kỹ thuật, phương pháp luận

↓

ứng dụng thực tế vào từng quy trình

↓

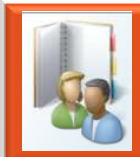
Cải biên, biến đổi vào từng sản phẩm và công cụ phần mềm (máy tính hóa từng phần)

↓

Tổng hợp, hệ thống hóa cho từng loại công cụ (Máy tính hóa toàn bộ quy trình sản xuất phần mềm)

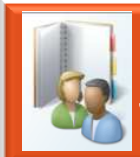
↓

Hướng tới sản xuất phần mềm tự động



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



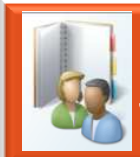
3.3 Định nghĩa Công nghệ học phần mềm (tiếp)

- **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;

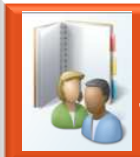
  (2) nghiên cứu các phương pháp tiếp cận được dùng trong (1)

- **Pressman [1995]:** CNHPM *là bộ môn tích hợp cả quy trình, các phương pháp, các công cụ để phát triển phần mềm máy tính*



3.3 Định nghĩa Công nghệ học phần mềm (tiếp)

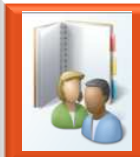
- **Sommerville [1995]:** CNHPM là lĩnh vực liên quan đến lý thuyết, phương pháp và công cụ dùng cho phát triển phần mềm
- **K. Kawamura [1995]:** CNHPM là lĩnh vực học vấn về các kỹ thuật, phương pháp luận công nghệ học (lý luận và kỹ thuật được hiện thực hóa trên những nguyên tắc, nguyên lý nào đó) trong toàn bộ quy trình phát triển phần mềm nhằm nâng cao cả chất và lượng của sản xuất phần mềm



3.3 Định nghĩa Công nghệ học phần mềm (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]

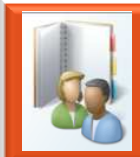


Công nghệ học trong CNHPM ?

(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

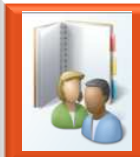


Công nghệ học trong CNHPM ? (tiếp)

(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

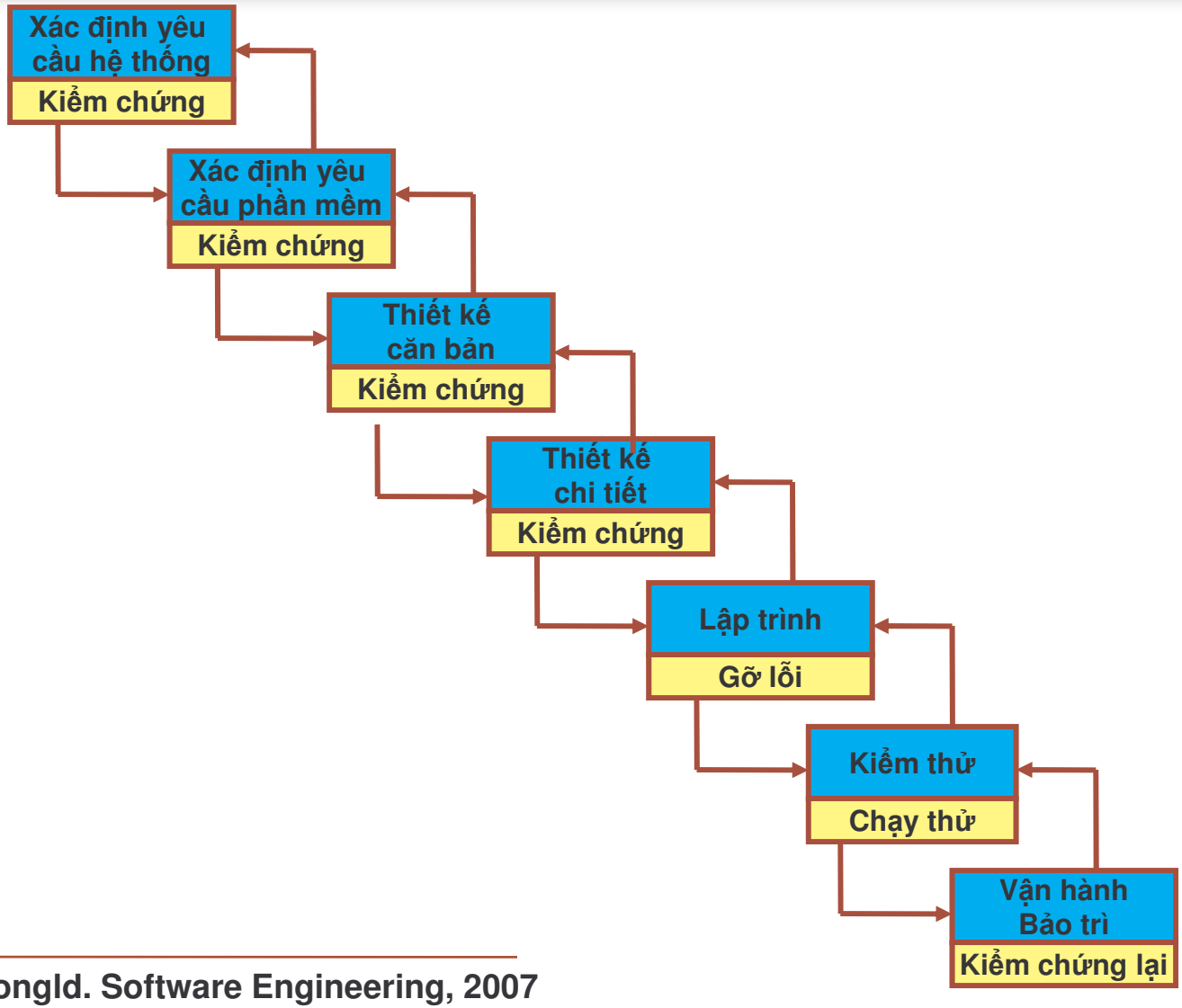
(6) Cách tiếp cận công nghệ học (khái niệm công nghiệp hóa) thể hiện ở chỗ nhằm nâng cao năng suất (tính năng suất) và độ tin cậy của phần mềm, đồng thời giảm chi phí giá thành

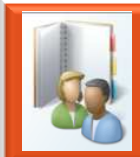


3.4 Vòng đời phần mềm (Software life-cycle)

- 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





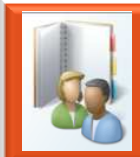
Suy nghĩ mới về vòng đời phần mềm

(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)

(4) Trước khi chuyển sang pha kế tiếp phải đảm bảo pha hiện nay đã được kiểm thử không còn lỗi



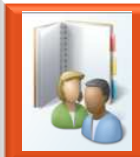
Suy nghĩ mới về vòng đời phần mềm

(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

(7) Cần chuẩn hóa mẫu biểu, cách ghi chép tạo tư liệu cho từng pha, nhằm đảm bảo chất lượng phần mềm

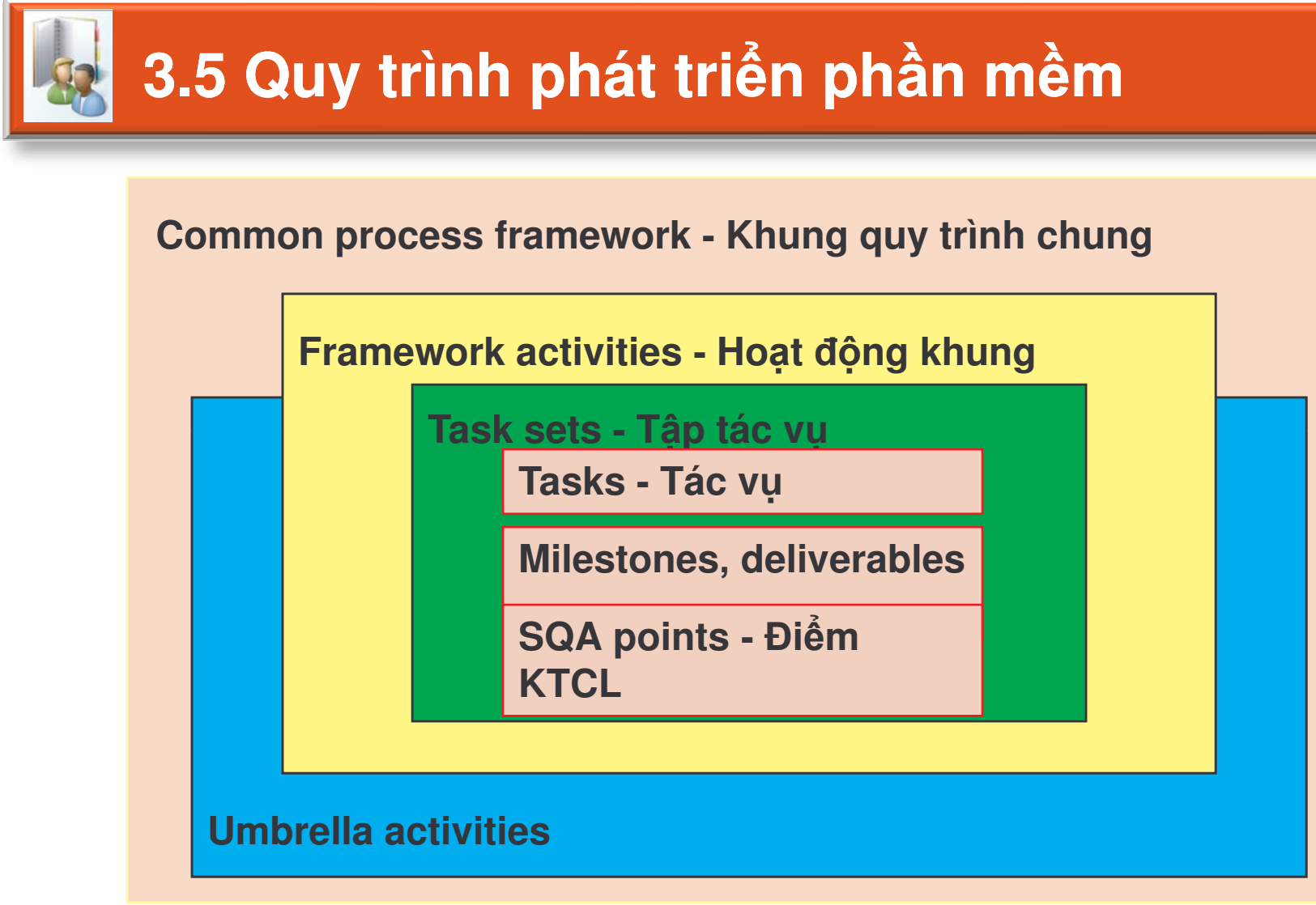
(8) Thao tác bảo trì phần mềm là việc xử lý quay vòng trở lại các pha trong vòng đời phần mềm nhằm biến đổi, sửa chữa, nâng cấp phần mềm

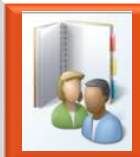


Các phương pháp luận và kỹ thuật cho từng pha

Tên pha	Nội dung nghiệp vụ	Phương pháp, kỹ thuật
Xác định yêu cầu	Đặc tả yêu cầu người 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 chương trình	Là thiết kế chi tiết: Thiết kế cấu trúc bên trong của phần mềm (đơn vị chương trình hoặc môđun)	Lập trình cấu trúc Phương pháp Jackson Phương pháp Warnier
Lập trình	Mã hóa bởi ngôn ngữ lập trình	Mã hóa cấu trúc hóa
Đảm bảo chất lượng	Kiểm tra chất lượng phần mềm đã phát triển	Phương pháp kiểm thử chương trình
Vận hành Bảo trì	Sử dụng, vận hành phần mềm đã phát triển. Biến đổi, điều chỉnh phần mềm	Chưa cụ thể

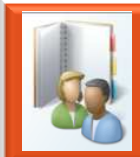
3.5 Quy trình phát triển phần mềm





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

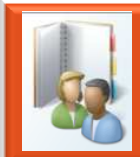


3.5.1 Capability Maturity Model (CMM) by SEI:

- **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



LEVEL 2: Repeatable

1. SW configuration management
2. SW quality assurance
3. SW subcontract management
4. SW project tracking and oversight
5. SW project planning
6. Requirements management

7. Peer reviews
8. Intergroup coordination
9. SW product engineering
10. IntegratedSW management
11. Training program
12. Organization process definition
13. Organization process focus

LEVEL 3: Defined

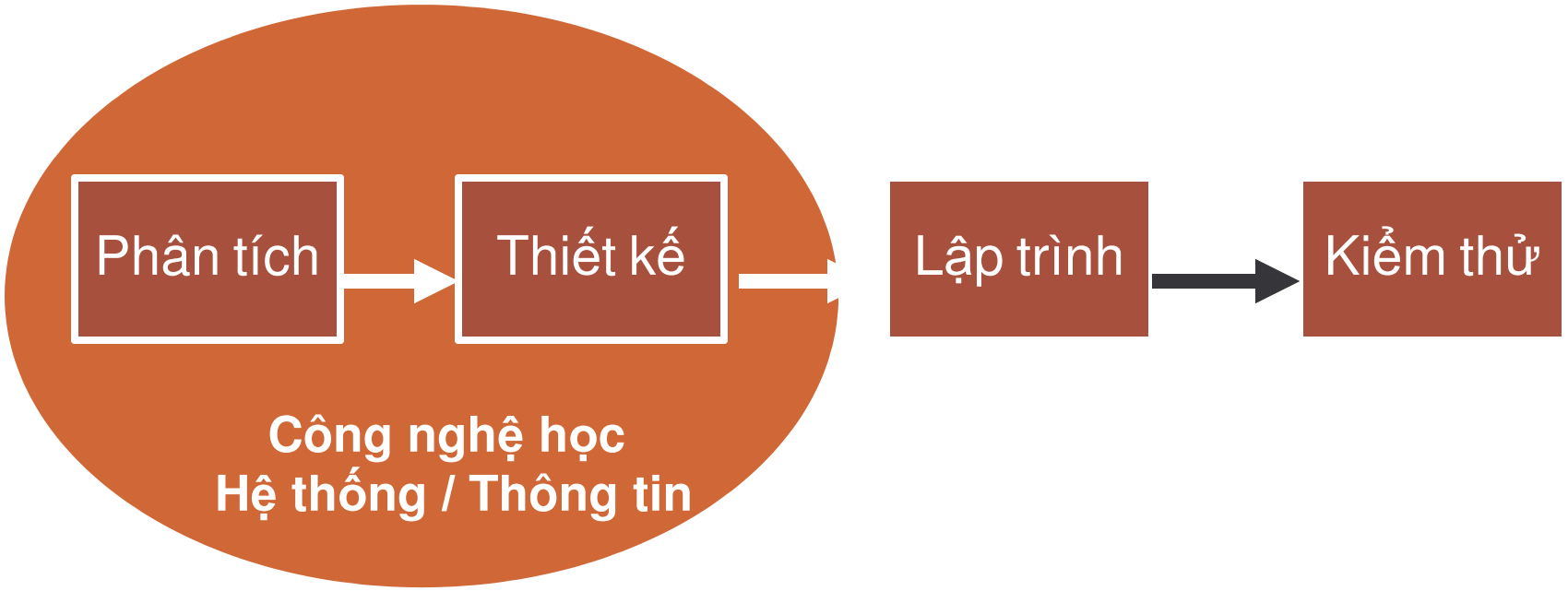
14. SW quality Management
15. Quantitative process management

LEVEL 4: Managed

16. Process change management
17. Technology change management
18. Defect prevention

LEVEL 5: Optimizing

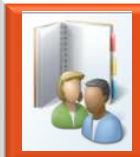
3.5.2 Mô hình tuyến tính



Điển hình là mô hình vòng đời cổ điển (mô hình thác nước) Classic life cycle / waterfall model: là mô hình hay được dùng nhất

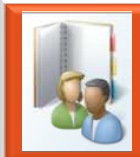
3.5.2 Mô hình tuyến tính (tiếp)

- **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



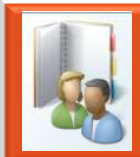
3.5.2 Mô hình tuyến tính (tiếp)

- **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



3.5.2 Mô hình tuyến tính (tiếp)

- **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!

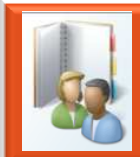
Department of Software Engineering.
Faculty of Mathematic & Informatic. Hai Phong University.

Công nghệ phần mềm

(Introduction to Software Engineering)

Phần II: Phương pháp quản lý dự án CNTT

Editor: **LÊ ĐẮC NHƯỜNG**
Email: **Nhuongld@yahoo.com**
Phone: 0987394900

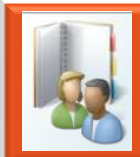


Chương 4

QUẢN LÝ DỰ ÁN CNTT

(IT Project Management)

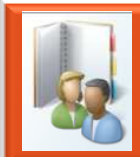
1. Tổng quan
2. Lập kế hoạch quản lý
3. Tổ chức dự án
4. Quản lý rủi ro
5. Phát triển nhóm
6. Quản lý chất lượng
7. Lập kế hoạch làm việc chi tiết
8. Kiểm soát và lập báo cáo dự án
9. Quản lý vấn đề và kiểm soát thay đổi
10. Quản lý cấu hình
11. Hoàn tất dự án



4.1. Tổng quan

Mục tiêu

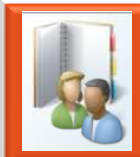
- Khái niệm về dự án và quản lý dự án
- Tại sao các dự án lại thất bại ?
- Các dự án CNTT có gì đặc biệt ?



Các định nghĩa về quản lý dự án

Một dự án:

- là riêng biệt, độc lập
- có điểm bắt đầu và điểm kết thúc
- có sản phẩm cụ thể cuối cùng
- là duy nhất, hoặc về sản phẩm hoặc về môi trường của nó



Mục đích của quản lý dự án

Quản lý dự án là để đưa ra một sản phẩm cuối cùng:

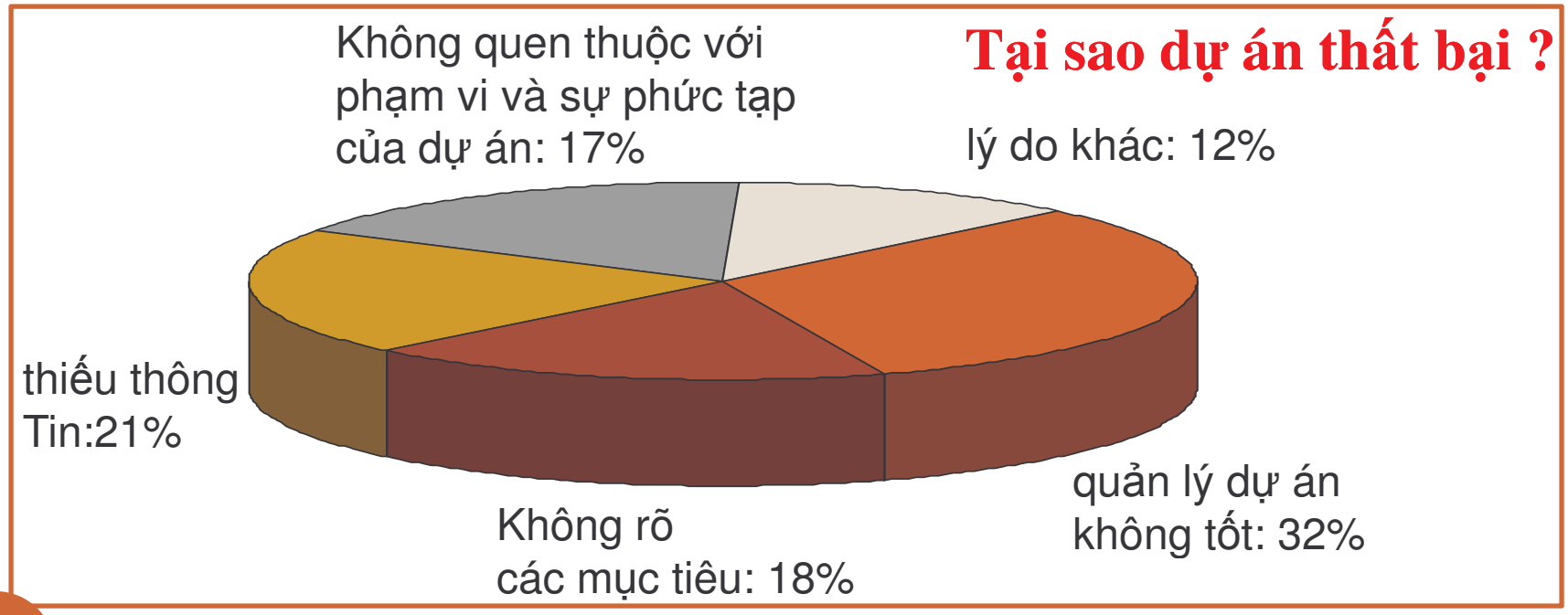
- đúng hạn
- trong phạm vi ngân sách hay nguồn tài chính cho phép
- phù hợp theo các đặc tả
- với một mức độ chất lượng để phục vụ các nhu cầu kinh doanh và đáp ứng các tiêu chuẩn chuyên môn và kỳ vọng của công tác quản lý

Tại sao các dự án lại thất bại?
hay điều gì khiến một dự án thành công?

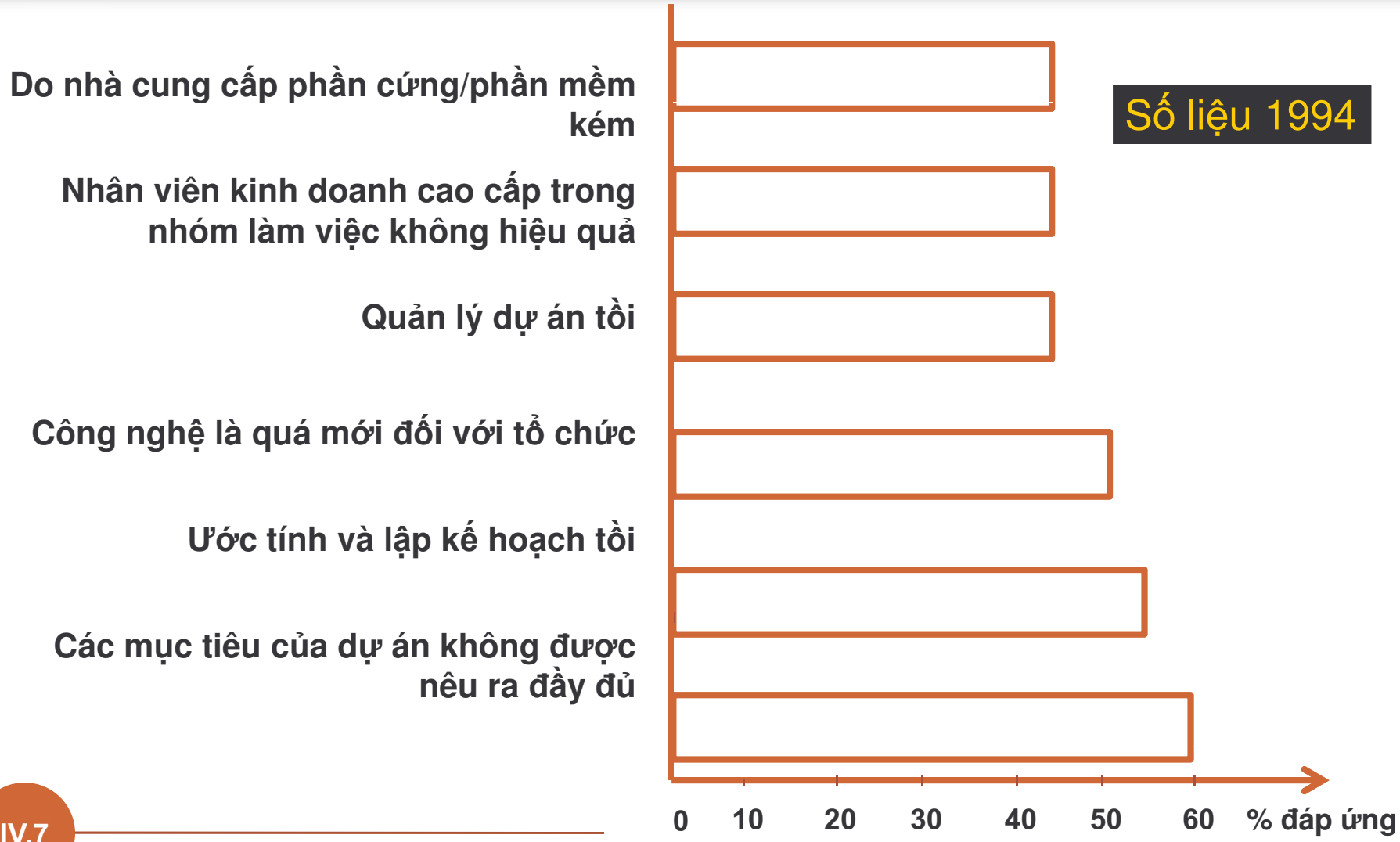
Định nghĩa về dự án bị thất bại

Một dự án mà:

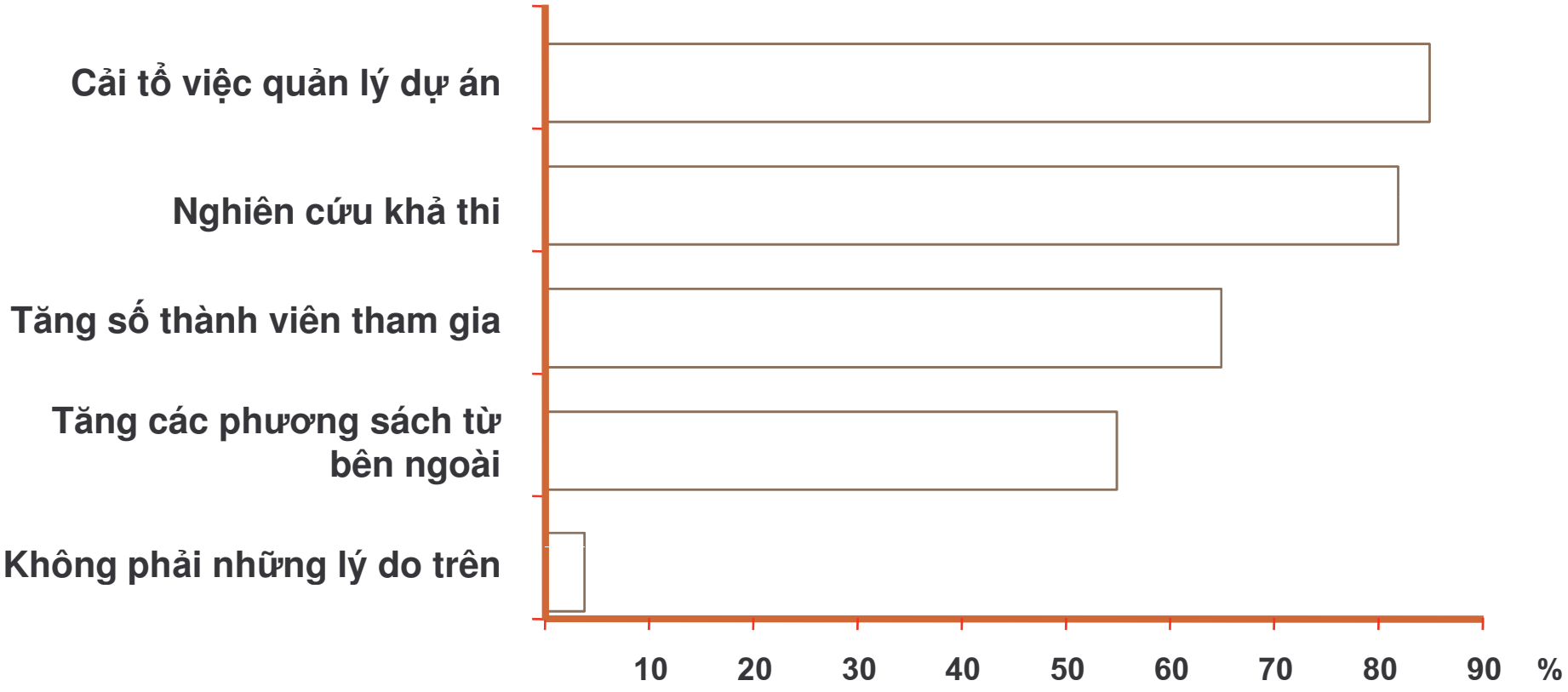
- Không đạt được các mục tiêu của dự án, và/hoặc
- Bị vượt quá ngân sách ít nhất 30%



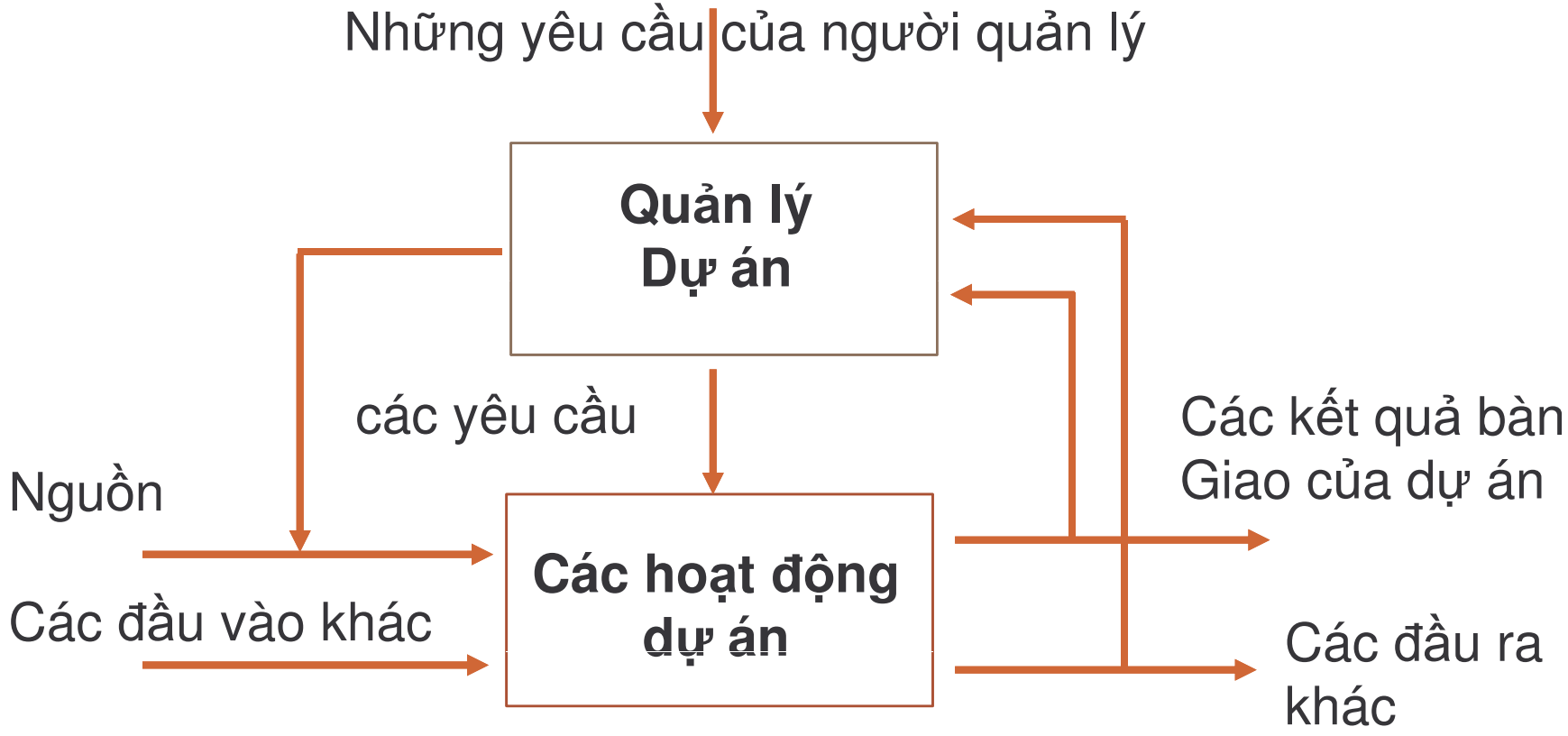
Những nguyên nhân thất bại



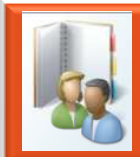
Để tránh thất bại



Thực hiện dự án không có nghĩa là Quản trị dự án!



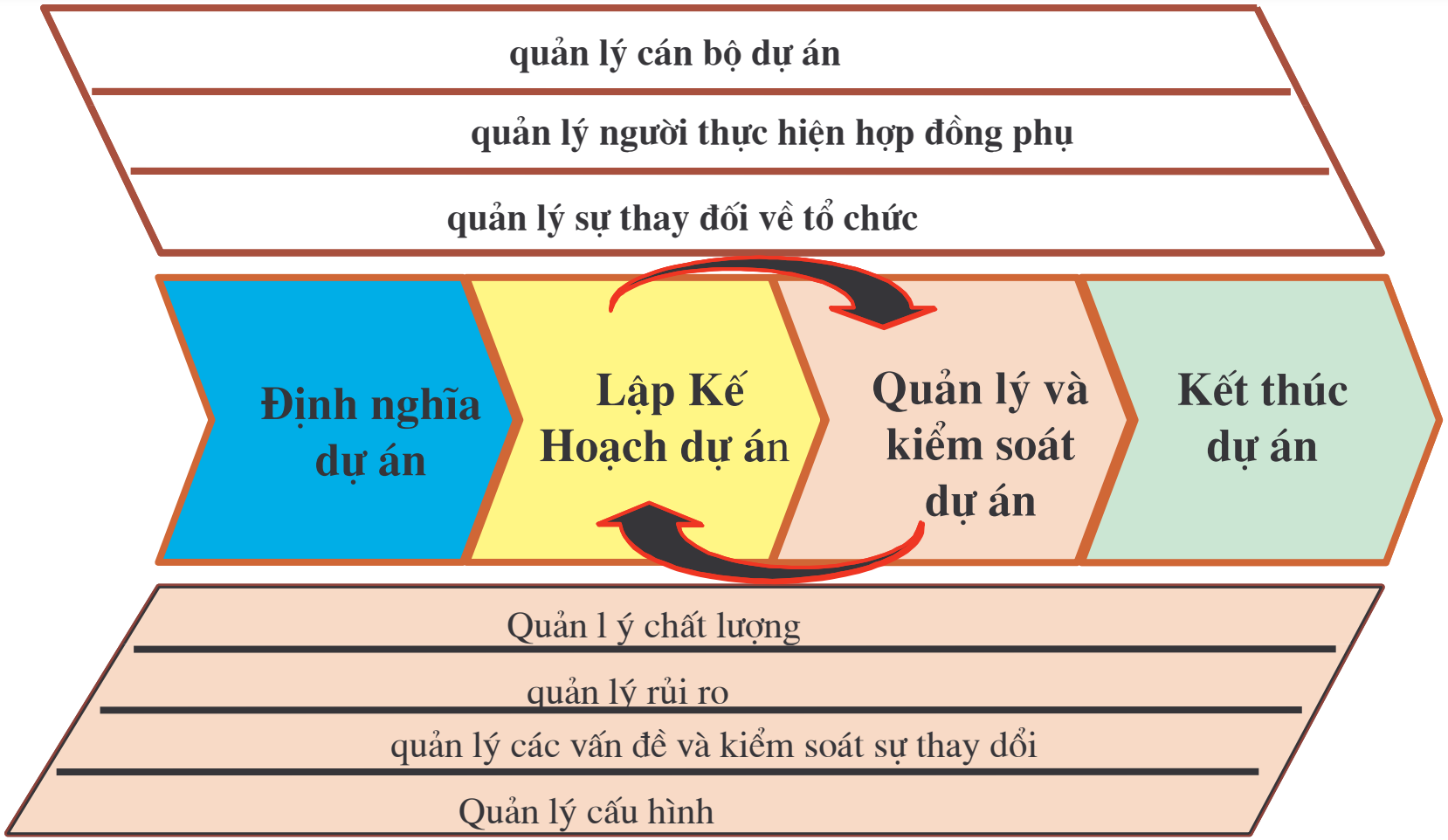
.....và quản lý dự án không phải là thực hiện dự án!

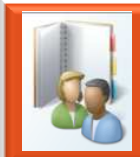


Các thuộc tính đặc trưng của dự án IT

- Các kết quả bàn giao có thể là ít hữu hình và ít quen thuộc hơn so với các loại dự án khác. Phạm vi có thể khó kiểm soát
- Đội dự án thường có những kỹ năng, kinh nghiệm, thái độ và kỳ vọng trái ngược nhau
- Dự án có thể bị căng thẳng để đạt được các mục tiêu kinh doanh. Dự án có thể được kết nối với những sự thay đổi quan trọng về tổ chức
- Các yêu cầu, phạm vi, và lợi nhuận chính xác có thể rất khó xác định. Sự thay đổi nhanh chóng về công nghệ có thể làm cho nền tảng của dự án trở nên lỗi thời

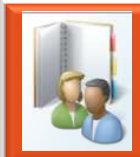
Cấu trúc Phương pháp QLDA





10 quy tắc vàng

- Quản lý dự án thành công chính là vấn đề về con người
  - ❖ nhưng không được quên quản trị
- Khám phá các nguồn hỗ trợ và chống đỡ
- Sự hiện diện có thể là dối trá - xem xét lịch trình ẩn đằng sau
- Phải hiểu rằng những con người khác nhau thì có những cách nhìn khác nhau và hãy đặt mình vào địa vị của họ
- Thiết lập kế hoạch của bạn sao cho có thể chỉnh sửa dễ dàng
- Đối mặt với từng sự kiện như là nó đã có từ trước
- Sử dụng quản trị để hỗ trợ cho các mục đích của dự án
- Thời gian mục tiêu đối với từng nhiệm vụ không được giống như đã nêu trong kế hoạch
- Đọc lại phạm vi và các mục tiêu của dự án mỗi tuần 1 lần
- Không ngạc nhiên!

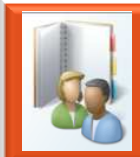


4.2. Lập kế hoạch quản lý

Các mục tiêu

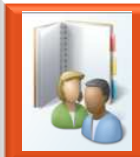
Sau khi kết thúc phần này bạn sẽ:

- Hiểu được sự cần thiết của việc lập kế hoạch và các bước của việc lập kế hoạch quản lý
- Có thể lập ra một kế hoạch quản lý toàn diện ở một mức độ chi tiết hợp lý đối với dự án và đây chính là bước mở đầu của dự án
- Có thể đưa ra cho khách hàng về sự cần thiết của việc lập kế hoạch quản lý



4.2 Lập kế hoạch quản lý

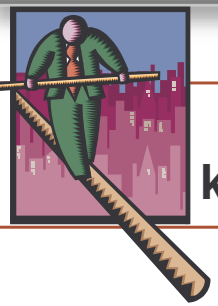
- Xác định ranh giới của dự án
  - đội lập kế hoạch, văn bản/thông tin hiện có
- Xây dựng các lựa chọn tiếp cận dự án
  - chiến lược thực hiện và các phương pháp luận tổ chức dự án
- Xây dựng các ước tính ban đầu
- Xây dựng cơ sở hạ tầng nguồn
  - môi trường làm việc
- Xây dựng cơ sở hạ tầng của dự án
  - quản lý cấu hình, chất lượng, rủi ro, sự kiện, sự thay đổi, kiểm soát dự án, lập báo cáo, và lập kế hoạch
- Lập thành văn bản về kế hoạch quản lý



Các vai trò và trách nhiệm của dự án

Vai trò	Trách nhiệm	Vai trò trong việc lập kế hoạch quản lý	thời gian thực hiện
Ban điều hành	Chiến lược kinh doanh	Không	Không
Ban chỉ đạo	điều hành dự án	phê chuẩn	từ lúc bắt đầu DA
Nhà tài trợ DA	luôn sẵn sàng hỗ trợ dự án	đầu vào về phạm vi, mục tiêu, lợi ích	từ lúc bắt đầu DA
Giám đốc dự án	quản lý chiến dự án	xem xét và phê chuẩn	từ lúc bắt đầu DA
Quản lý dự án	quản lý hoạt động dự án	chịu trách nhiệm về kết quả	Trong thời gian thực hiện dự án
Nhóm trưởng dự án	chịu trách nhiệm về nhiệm vụ dự án	hỗ trợ người quản lý dự án lập	trong suốt thời gian kế hoạch quản lý
Cán bộ dự án	hoàn thành nhiệm vụ	None	trong suốt thời gian hoạt động dự án

Xây dựng & Thông qua kế hoạch quản lý



Những rủi ro gặp phải khi không lập kế hoạch quản lý

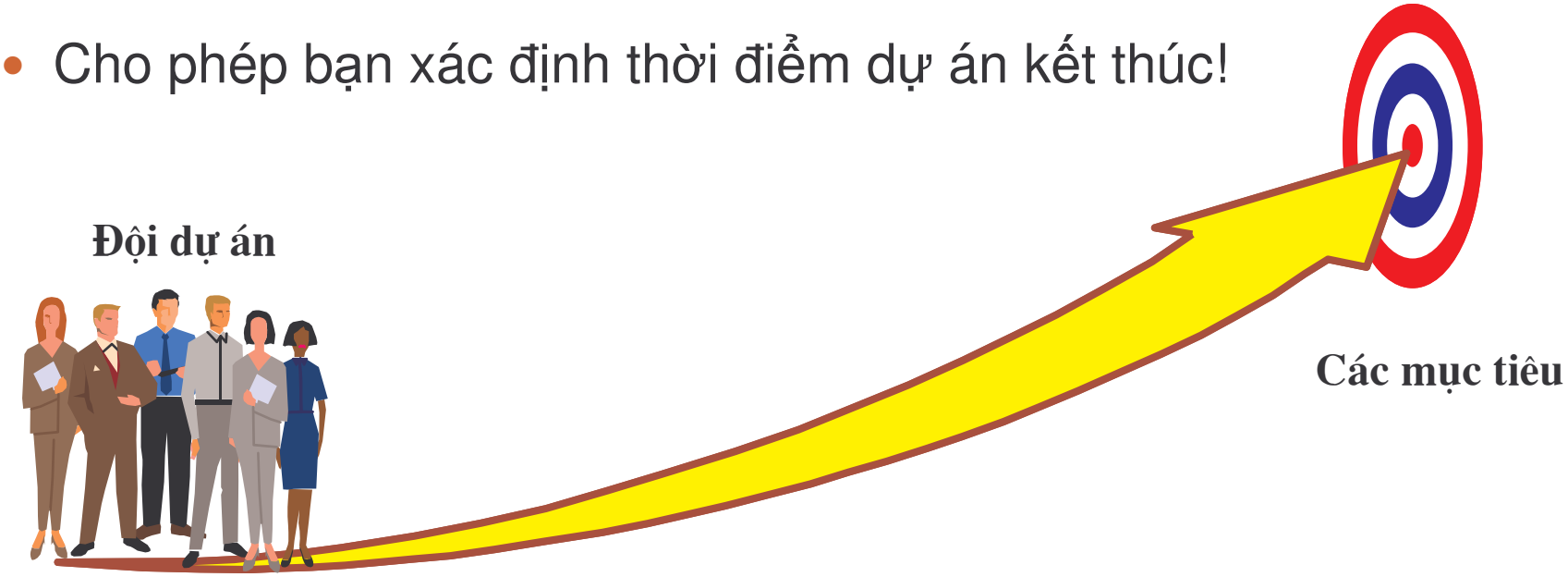
- Khởi đầu sai lệch
- Bị nhầm lẫn
- Không đáp ứng được sự mong đợi của nhà tài trợ và/hoặc các mục tiêu
- Thông tin nghèo nàn

Các lợi ích khi lập kế hoạch quản lý

- Đáp ứng các mục tiêu của nhà tài trợ
- Gây dựng lòng tin của đối tác
- Thiết lập hướng làm việc chung
- Bao quát được các thách thức
- Mở ra các kênh thông tin liên lạc
- Bắt đầu dự án với một phương thức
- có hệ thống

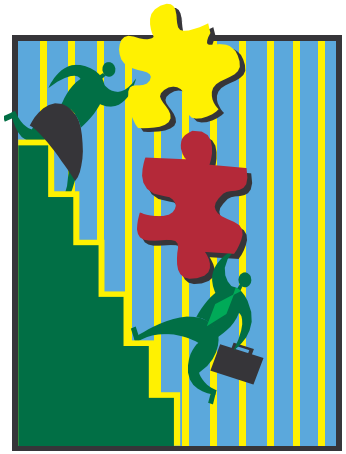
Giá trị của các mục tiêu rõ ràng

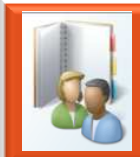
- Thiết lập sự mong đợi của nhà tài trợ dự án và các nhà đầu tư
- Đưa ra điểm mục tiêu để hướng dẫn đội dự án
- Cho phép bạn xác định thời điểm dự án kết thúc!



Các bước xác định phạm vi dự án

- Xem xét lại các văn bản hiện có
- Lập danh sách các văn bản/ thông tin chưa đầy đủ hay còn thiếu
- Tiến hành phỏng vấn, hội thảo để thu thập các thông tin còn thiếu
- Phân loại các thông tin cụ thể liên quan đến
- Các cam kết, lịch trình và các kết quả bàn giao
- Tiếp tục kết hợp chặt chẽ các chi tiết vào kế hoạch quảnlý
- Đạt được thoả thuận





Ích lợi của việc xác định phạm vi

- "Báo cáo phạm vi dự án" được xây dựng
- Các lợi ích của dự án được lập thành văn bản rõ ràng
- Xác định được các kết quả chính và các tiêu thức để hoàn thành dự án
- Xác định rõ các hạn chế, giả thuyết, điểm bên trong và bên ngoài

Department of Software Engineering.
Faculty of Mathematic & Informatic. Hai Phong University.

Công nghệ phần mềm

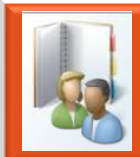
(Introduction to Software Engineering)

Phần III: Yêu cầu người dùng
User's Requirements

Editor: **LÊ ĐẮC NHƯỜNG**

Email: **Nhuongld@yahoo.com**

Phone: 0987394900



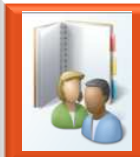
Chương 5

PHƯƠNG PHÁP XÁC ĐỊNH YÊU CẦU

5.1. Kỹ thuật xác định yêu cầu

5.2. Nội dung xác định yêu cầu

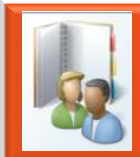
5.3. Các nguyên lý phân tích yêu cầu



5.1. Kỹ thuật xác định yêu cầu phần mềm
SW Requirements Engineering

Yêu cầu phần mềm: là tất cả các yêu cầu về phầm mềm do khách hàng - người sử dụng phần mềm - nêu ra, bao gồm:

- Các chức năng của phần mềm,
- Hiệu năng của phần mềm,
- Các yêu cầu về thiết kế và giao diện,
- Các yêu cầu đặc biệt khác

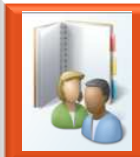


5.1. Kỹ thuật xác định yêu cầu phần mềm
SW Requirements Engineering

Thông thường các yêu cầu phần mềm được phân loại theo 4 thành phần của phần mềm:

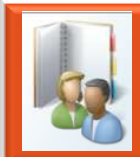
- **Các yêu cầu về phần mềm (Software)**
- **Các yêu cầu về phần cứng (Hardware)**
- **Các yêu cầu về dữ liệu (Data)**
- **Các yêu cầu về con người (People, Users)**

Mục đích: mục đích của yêu cầu phần mềm là xác định được phần mềm đáp ứng được các yêu cầu và mong muốn của khách hàng - người sử dụng phần mềm



Tại sao cần phải đặt ra yêu cầu phần mềm ?

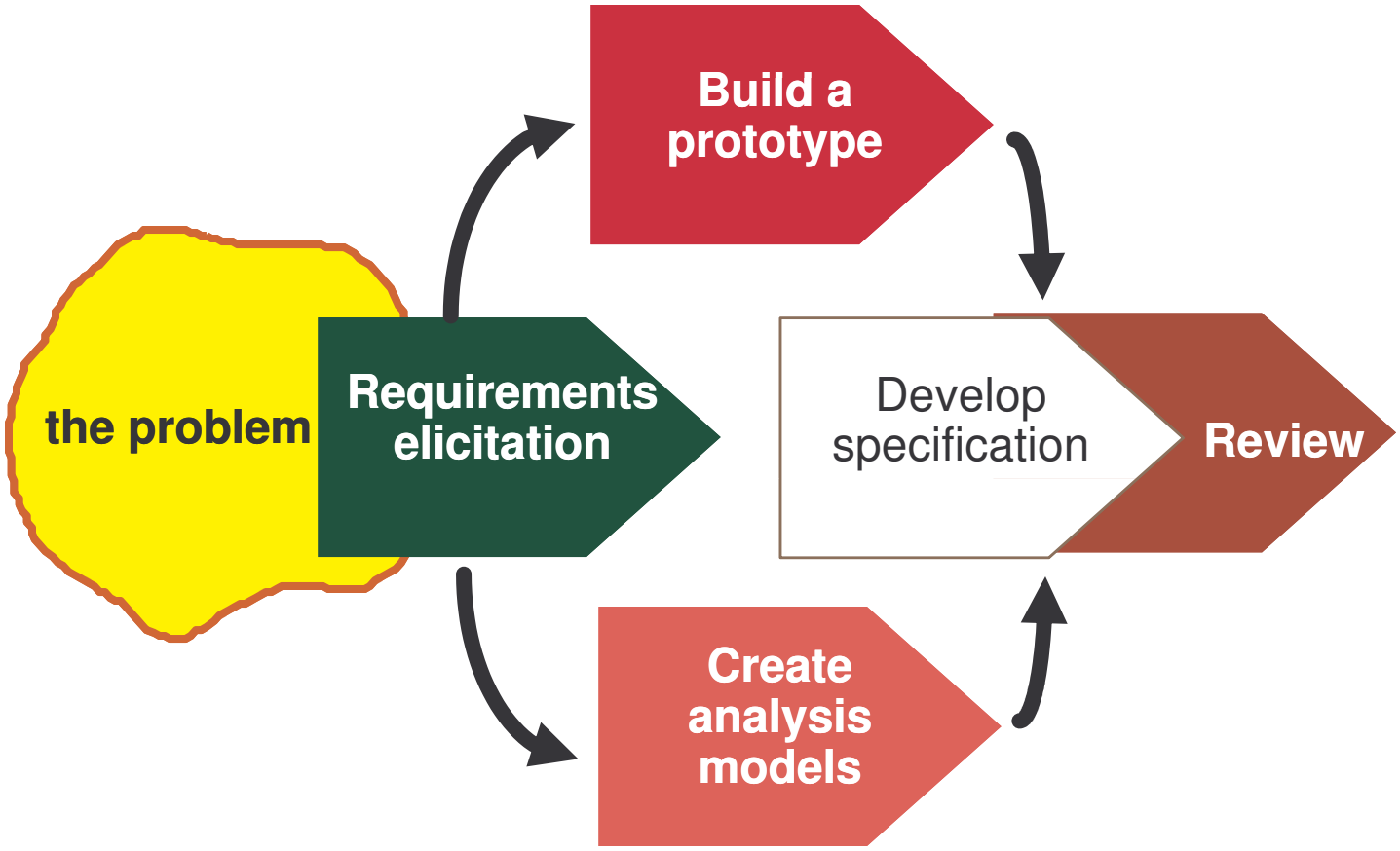
- Khách hàng chỉ có những ý tưởng còn mơ hồ về phần mềm cần phải xây dựng để phục vụ công việc của họ, chúng ta phải sẵn sàng, kiên trì theo đuổi để đi từ các ý tưởng mơ hồ đó đến "Phần mềm có đầy đủ các tính năng cần thiết"
- Khách hàng rất hay thay đổi các đòi hỏi của mình, chúng ta nắm bắt được các thay đổi đó và sửa đổi các mô tả một cách hợp lý



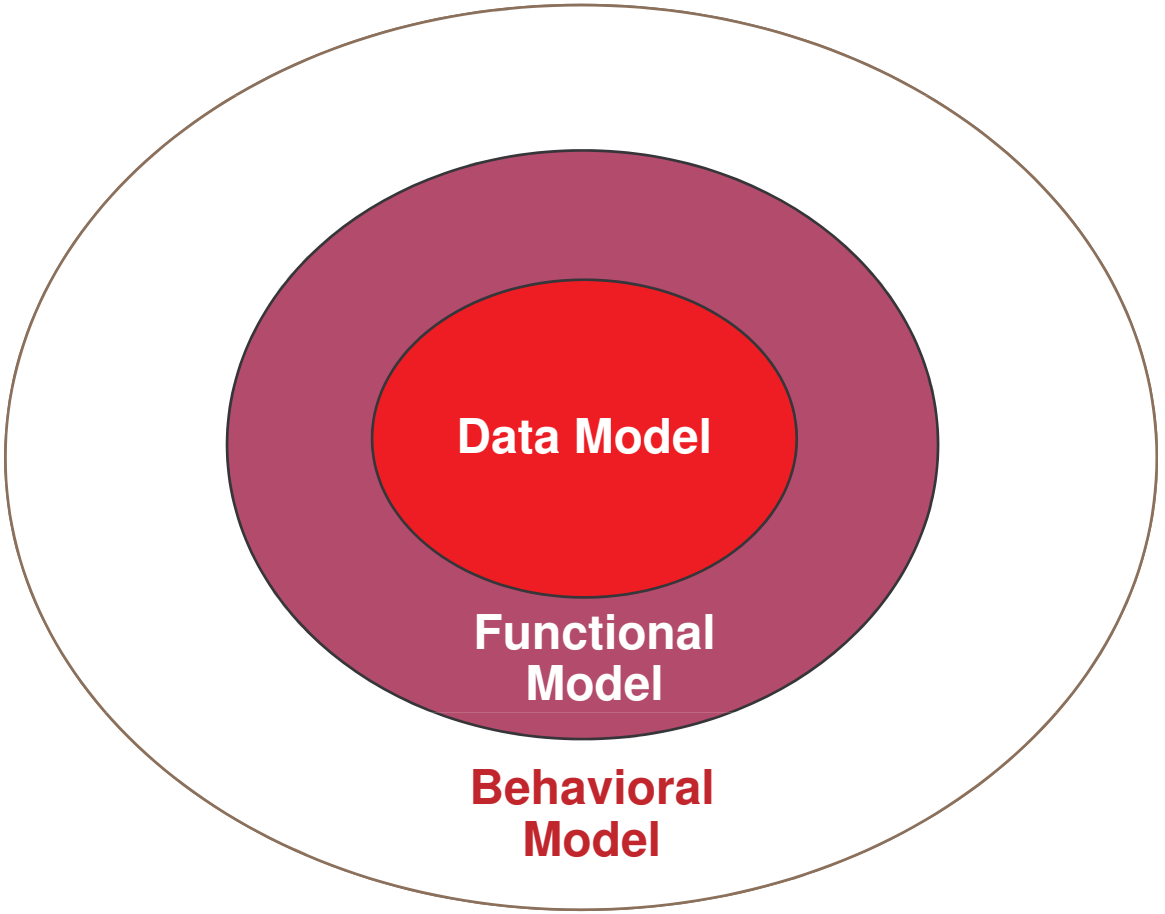
5.2. Nội dung xác định yêu cầu phần mềm
Contents of Requirements Engineering

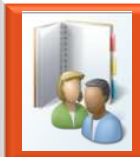
- Phát hiện các yêu cầu phần mềm (Requirements elicitation)
- Phân tích các yêu cầu phần mềm và thương lượng với khách hàng (Requirements analysis and negotiation)
- Mô tả các yêu cầu phần mềm (Requirements specification)
- Mô hình hóa hệ thống (System modeling)
- Kiểm tra tính hợp lý các yêu cầu phần mềm (Requirements validation)
- Quản trị các yêu cầu phần mềm (Requirements management)

Quy trình xác định yêu cầu phần mềm



The Analysis Model

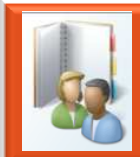




5.2.1. Phát hiện yêu cầu phần mềm (Requirements Elicitation)

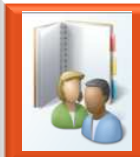
Các vấn đề của phát hiện yêu cầu phần mềm (Problems)

- Phạm vi của phần mềm (Scope)
- Hiểu rõ phần mềm (Understanding)
- Các thay đổi của hệ thống (Volatility)



Phương pháp phát hiện yêu cầu phần mềm
Requirements Elicitation Methodology

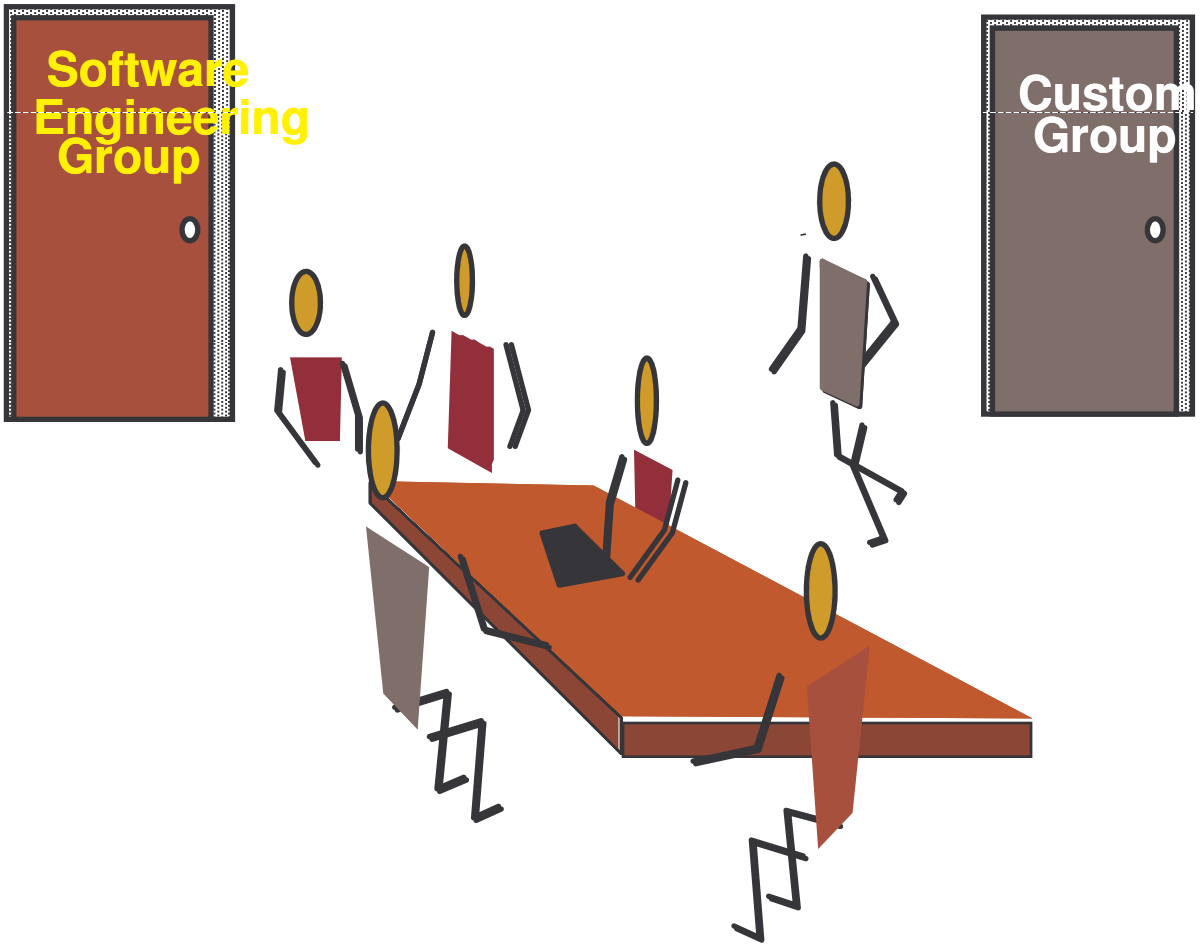
- Xác định các phương pháp sử dụng phát hiện các yêu cầu phần mềm: phỏng vấn, làm việc nhóm, các buổi họp, gặp gỡ đối tác, v.v.
- Tìm kiếm các nhân sự (chuyên gia, người sử dụng) có những hiểu biết sâu sắc nhất, chi tiết nhất về hệ thống giúp chúng ta xác định yêu cầu phần mềm
- Xác định "môi trường kỹ thuật - technical environment"
- Xác định các "ràng buộc lĩnh vực domain constraints"
- Thu hút sự tham gia của nhiều chuyên gia, khách hàng để chúng ta có được các quan điểm xem xét phần mềm khác nhau từ phía khách hàng
- Thiết kế các kịch bản sử dụng của phần mềm

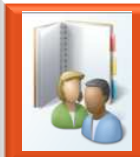


Sản phẩm (output) của "phát hiện yêu cầu phần mềm"

- Bảng kê (statement) các đòi hỏi và chức năng khả thi của phần mềm
- Bảng kê phạm vi ứng dụng của phần mềm
- Mô tả môi trường kỹ thuật của phần mềm
- Bảng kê tập hợp các kịch bản sử dụng của phần mềm
- Các nguyên mẫu xây dựng, phát triển hay sử dụng trong phần mềm (nếu có)
- Danh sách nhân sự tham gia vào quá trình phát hiện các yêu cầu phần mềm - kể cả các nhân sự từ phía công ty- khách hàng

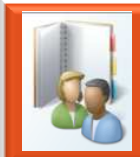
5.2.2. Phân tích các yêu cầu phần mềm và thương lượng với khách hàng





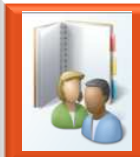
Requirements Analysis and Negotiation

- Phân loại các yêu cầu phần mềm và sắp xếp chúng theo các nhóm liên quan
- Khảo sát tỉ mỉ từng yêu cầu phần mềm trong mối quan hệ của nó với các yêu cầu phần mềm khác
- Thẩm định từng yêu cầu phần mềm theo các tính chất: phù hợp, đầy đủ, rõ ràng, không trùng lặp
- Phân cấp các yêu cầu phần mềm theo dựa trên nhu cầu và đòi hỏi khách hàng / người sử dụng



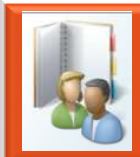
Requirements Analysis and Negotiation

- Thẩm định từng yêu cầu phầm mềm để xác định chúng có khả năng thực hiện được trong môi trường kỹ thuật hay không, có khả năng kiểm định các yêu cầu phần mềm hay không?
- Thẩm định các rủi ro có thể xảy ra với từng yêu cầu phần mềm
- Đánh giá thô (tương đối) về giá thành và thời gian thực hiện của từng yêu cầu phần mềm trong giá thành sản phẩm phần mềm và thời gian thực hiện phần mềm
- Giải quyết tất cả các bất đồng về yêu cầu phần mềm với khách hàng / người sử dụng trên cơ sở thảo luận và thương lượng các yêu cầu đề ra



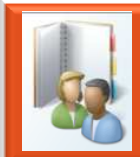
5.2.3. Đặc tả yêu cầu phần mềm

- Đặc tả các yêu cầu phần mềm là công việc xây dựng các tài liệu đặc tả, trong đó có thể sử dụng tới các công cụ như: mô hình hóa, mô hình toán học hình thức (a formal mathematical model), tập hợp các kịch bản sử dụng, các nguyên mẫu hoặc bất kỳ một tổ hợp các công cụ nói trên
- Chất lượng của hồ sơ đặc tả đánh giá qua các tiêu thức:
  - Tính rõ ràng, chính xác
  - Tính phù hợp
  - Tính đầy đủ, hoàn thiện



Requirements Specification

- **Các thành phần của hồ sơ đặc tả**
  - Đặc tả phi hình thức (Informal specifications) được viết bằng ngôn ngữ tự nhiên
  - Đặc tả hình thức (Formal specifications) được viết bằng tập các ký pháp có các quy định về *cú pháp (syntax) và ý nghĩa (sematic) rất chặt chẽ*
  - Đặc tả vận hành chức năng (Operational specifications) mô tả các hoạt động của hệ thống phần mềm sẽ xây dựng
  - Đặc tả mô tả (Descriptive specifications) – đặc tả các đặc tính đặc trưng của phần mềm



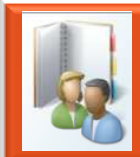
Requirements Specification

- **Đặc tả chức năng (Operational Specifications):**

  Thông thường khi đặc tả các chức năng của phần mềm người ta sử dụng các công cụ tiêu biểu sau

  - Biểu đồ luồng dữ liệu (Data Flow Diagrams)
  - Máy trạng thái hữu hạn (Finite State Machines)
  - Mạng Petri (Petri nets)

- **Đặc tả mô tả (Descriptive Specifications)**
  - Biểu đồ thực thể liên kết (Entity-Relationship Diagrams)
  - Đặc tả Logic (Logic Specifications)
  - Đặc tả đại số (Algebraic Specifications)

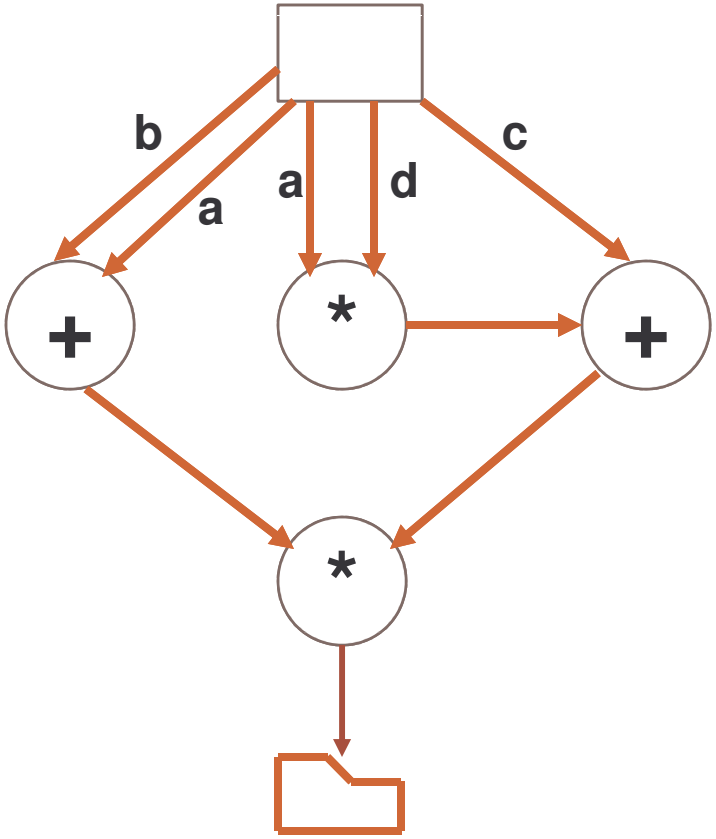


Biểu đồ luồng dữ liệu (DFD)

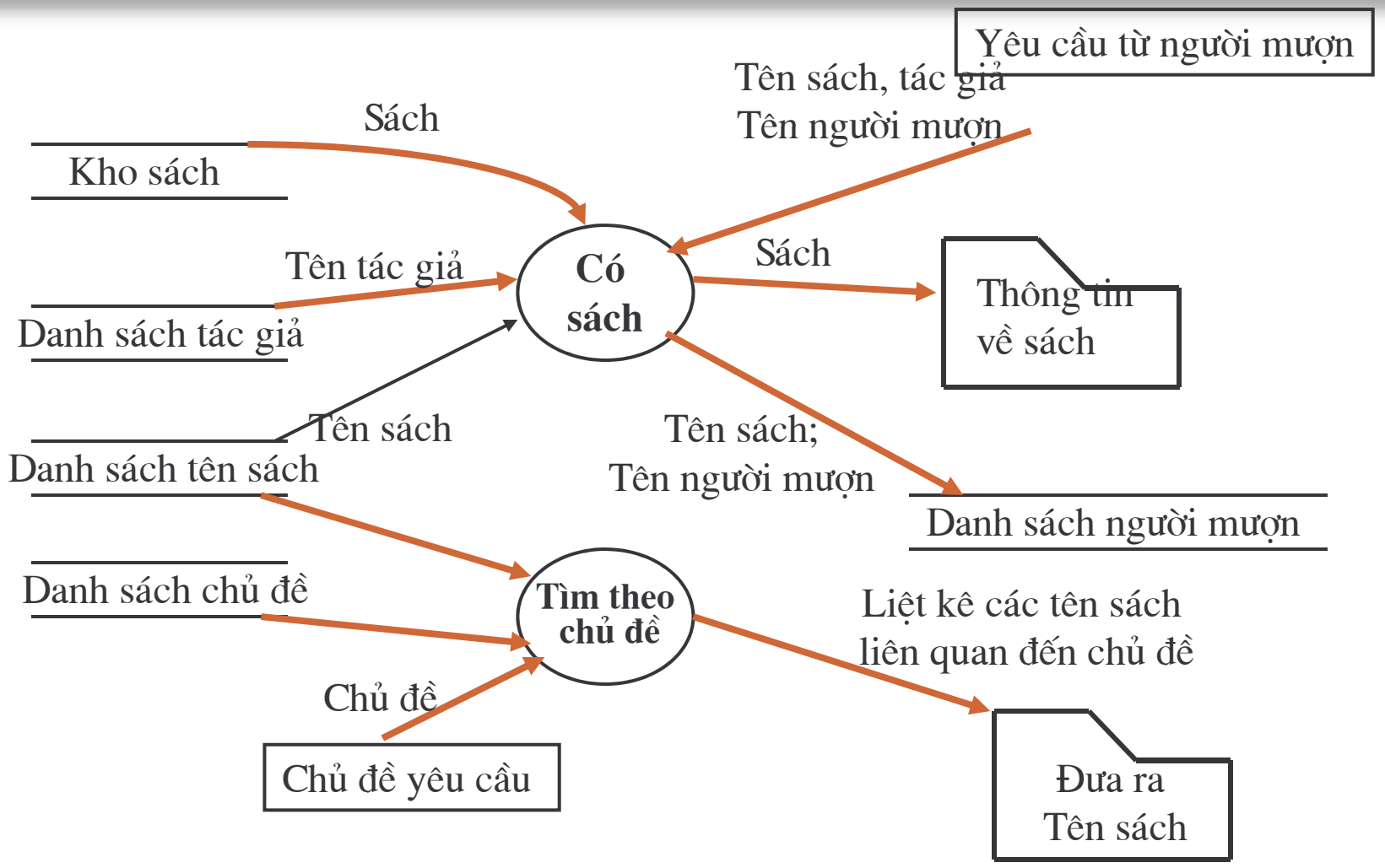
- **Hệ thống (System):** tập hợp các dữ liệu (data) được xử lý bằng các chức năng tương ứng (functions)
- **Các ký pháp sử dụng:**
  - Thể hiện các chức năng (functions)
  - Thể hiện luồng dữ liệu
  - Kho dữ liệu
  - Vào ra dữ liệu và tương tác giữa
  - hệ thống và người sử dụng

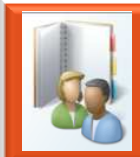
Ví dụ mô tả biểu thức toán học bằng DFD

(a+b)*(c+a*d)



Ví dụ đặc tả các chức năng của thư viện qua DFD





Các hạn chế của DFD

Ý nghĩa của các ký pháp sử dụng được xác định bởi các định danh lựa chọn của NSD. Ví dụ của chức năng tìm kiếm:

If NSD nhập vào cả tên tác giả và tiêu đề sách **Then**

tìm kiếm sách tương ứng, không có thì thông báo lỗi

Elseif chỉ nhập tên tác giả **Then**

hiển thị danh sách các sách tương ứng với

tên tác giả đã nhập và yêu cầu NSD lựa chọn sách

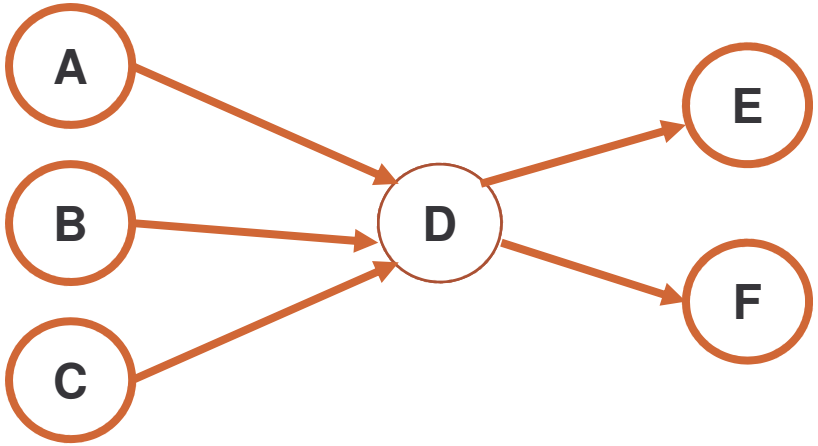
Elseif chỉ nhập tiêu đề sách **Then**

. . .

Endif

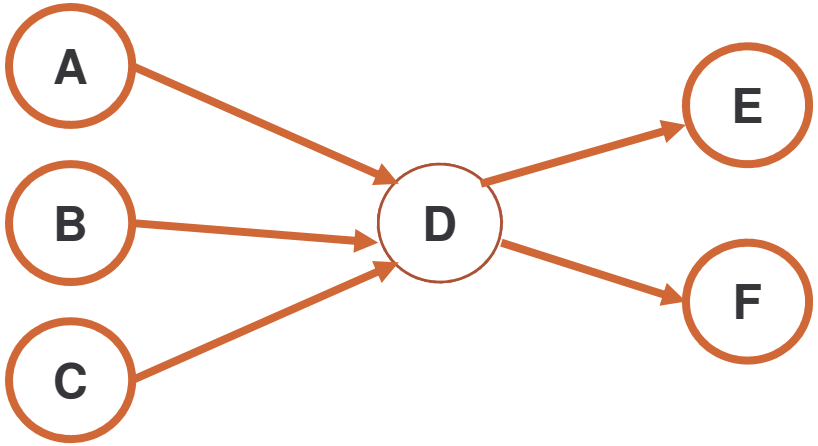
Các hạn chế của DFD

Trong DFD không xác định rõ các hướng thực hiện (control aspects)



Biểu đồ DFD này không chỉ rõ đầu vào là gì để thực hiện chức năng D và đầu ra là gì sau khi thựchiện chức năng D.

Các hạn chế của DFD



- Chức năng D có thể cần cả A, B và C
- Chức năng D có thể chỉ cần một trong A, B và C để thực hiện
- Chức năng D có thể kết xuất kết quả cho một trong E và F
- Chức năng D có thể kết xuất kết quả chung cho cả E và F
- Chức năng D có thể kết xuất kết quả riêng cho cả E và F

Các hạn chế của DFD

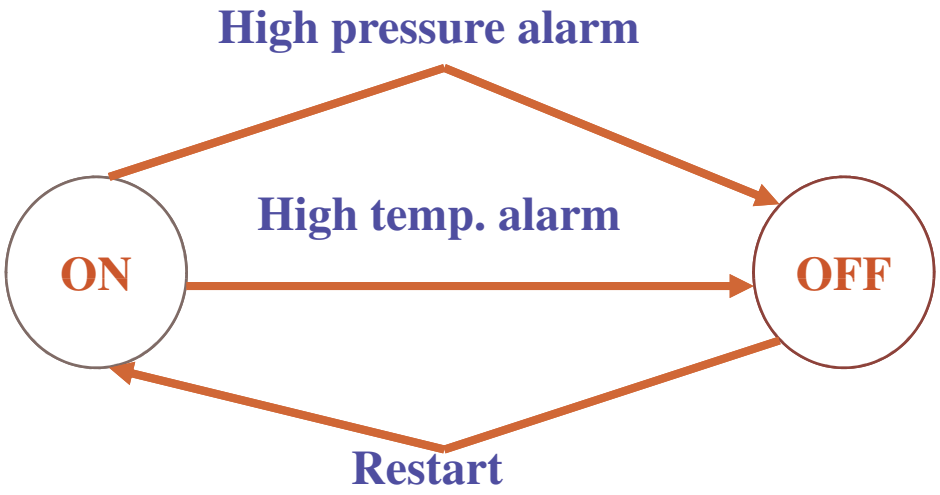
DFD không xác định sự đồng bộ giữa các chức năng / mô-đun

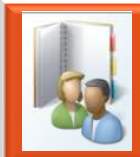
- A xử lý dữ liệu và B được hưởng (nhận) các kết quả được xử lý từ A
- A và B là các chức năng không đồng bộ (asynchronous activities) vì thế cần có buffer để ngăn chặn tình trang mất dữ liệu.

Finite State Machines (FSM)

FSM chứa

- Tập hữu hạn các trạng thái Q
- Tập hữu hạn các đầu vào I
- Các chức năng chuyển tiếp

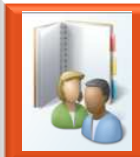




Đặc tả các yêu cầu phần mềm bằng FSM

Xem xét ví dụ về thư viện với các giao dịch như sau:

- Mượn sách / Trả sách
- Thêm đầu sách / Loại bỏ đầu sách
- Liệt kê danh sách các đầu sách theo tên tác giả hay theo chủ đề
- Tìm kiếm sách theo các yêu cầu của người mượn
- Tìm kiếm sách quá hạn trả, . . .



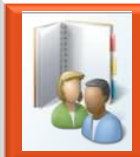
Đặc tả các yêu cầu phần mềm bằng FSM

Các yêu cầu đặc biệt của thư viện:

- Độc giả không được mượn quá một số lượng sách nhất định, trong một thời gian nhất định
- Một số sách không được mượn về
- Một số người không được mượn một số loại sách nào đó, . . .

Các đối tượng:

Tên sách
Mã quyển
Nhân viên phục vụ
Người mượn



Đặc tả các yêu cầu phần mềm bằng FSM

- Chúng ta cần có tập hợp (danh sách) các tiêu đề sách, danh sách các tác giả cho từng quyển sách, danh sách các chủ đề liên quan của các quyển sách
- Ta có tập hợp các sách (mỗi đầu sách có thể có nhiều quyển sách trong thư viện). Mỗi quyển sách có thể có 1 trong 5 trạng thái sau:
- (AV) - Available được phép mượn, (CO) - (BR) - đã mượn (Check Out; Borrow), (L): Last, (R): Remove

Department of Software Engineering.
Faculty of Mathematic & Informatic. Hai Phong University.

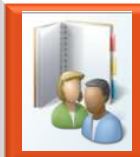
Công nghệ phần mềm

(Introduction to Software Engineering)

Phần IV: Thiết kế và lập trình

Design & Programming

Editor: **LÊ ĐẮC NHƯỜNG**
Email: **Nhuongld@yahoo.com**
Phone: 0987394900

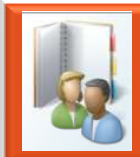


Chương 6

PHƯƠNG PHÁP THIẾT KẾ HỆ THỐNG

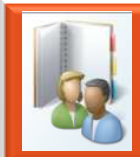
6.1. Thiết kế hệ thống là gì?

6.2. Phương pháp thiết kế hệ thống



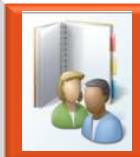
6.1. Thiết kế hệ thống là gì?

- Là thiết kế cấu hình phần cứng và cấu trúc phần mềm (gồm cả chức năng và dữ liệu) để có được hệ thống thỏa mãn các yêu cầu đề ra
- Có thể xem như Thiết kế cấu trúc (WHAT), chứ không phải là Thiết kế Logic (HOW)



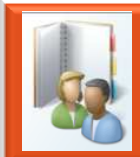
Quy trình thiết kế hệ thống

- Phân chia mô hình phân tích ra các hệ con
- Tìm ra sự tương tranh (concurrency) trong hệ thống
- Phân bố các hệ con cho các bộ xử lý hoặc các nhiệm vụ (tasks)
- Phát triển thiết kế giao diện
- Chọn chiến lược cài đặt quản trị dữ liệu



Quy trình thiết kế hệ thống (tiếp)

- Tìm ra nguồn tài nguyên chung và cơ chế điều khiển truy nhập chúng
- Thiết kế cơ chế điều khiển thích hợp cho hệ thống, kể cả quản lý nhiệm vụ
- Xem xét các điều kiện biên được xử lý như thế nào
- Xét duyệt và xem xét các thỏa hiệp (trade-offs)



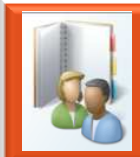
Các điểm lưu ý khi thiết kế hệ thống

(1) Có thể trích được luồng dữ liệu từ hệ thống: đó là phần nội dung đặc tả yêu cầu và giao diện

(2) Xem xét tối ưu tài nguyên kiến trúc lên hệ thống rồi quyết định kiến trúc

(3) Theo quá trình biến đổi dữ liệu, hãy xem những chức năng được kiến trúc như thế nào

(4) Từ kiến trúc các chức năng theo (3), hãy xem xét và chỉnh lại, từ đó chuyển sang kiến trúc chương trình và thiết kế chi tiết



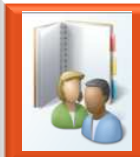
Các điểm lưu ý (tiếp)

(5) Quyết định các đơn vị chương trình theo các chức năng của hệ phần mềm có dựa theo luồng dữ liệu và phân chia ra các thành phần

(6) Khi cấu trúc chương trình lớn quá, phải phân chia nhỏ hơn thành các môđun

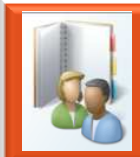
(7) Xem xét dữ liệu vào-ra và các tệp dùng chung của chương trình. Truy cập tệp tối ưu

(8) Hãy nghĩ xem để có được những thiết kế trên thì nên dùng phương pháp luận và những kỹ thuật gì ?



Thiết kế hệ thống

- Thiết kế hệ thống
  - Thiết kế hệ thống phần cứng [(1), (2)]
  - Thiết kế hệ thống phần mềm [(3)-(7)]
- Thiết kế hệ thống phần mềm
  - Thiết kế tệp (file design) [(7)]
  - Thiết kế chức năng hệ thống [(3)-(6)]



6.2 Phương pháp thiết kế hệ thống

- **Phương pháp thiết kế cấu trúc hóa**

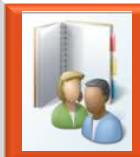
  Structured Design của Constantine

- Ngoài ra còn các phương pháp khác, như

  Phương pháp thiết kế tổng hợp

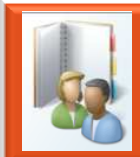
  Composite Design của Myers

Nhuongld. Software Engineering, 2007



Thiết kế cấu trúc hóa

- Bắt nguồn từ modularity, top-down design, structured programming
- Còn xem như Phương pháp thiết kế hướng luồng dữ liệu (Data flow-oriented design)
- Quy trình 6 bước: (1) tạo kiểu luồng thông tin; (2) chỉ ra biên của luồng; (3) ánh xạ DFD sang cấu trúc chương trình; (4) xác định phân cấp điều khiển; (5) tinh lọc cấu trúc; (6) chọn mô tả kiến trúc



Thiết kế cấu trúc hóa

(1) Môđun và tham số

(2) Lưu đồ bong bóng và cấu trúc phân cấp

- Lưu đồ bong bóng (Bubble chart)
- Cấu trúc phân cấp (Hierarchical structured chart)

(3) Phương pháp phân chia STS (Source/Transform/Sink) và TR (Transaction)

(4) Phân tích cấu trúc hóa

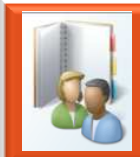
(5) Chuẩn phân chia môđun

(1) Môđun

- Dãy các lệnh nhằm thực hiện chức năng (function) nào đó
- Có thể được biên dịch độc lập
- Môđun đã được dịch có thể được môđun khác gọi tới
- Giao diện giữa các môđun thông qua các biến tham số

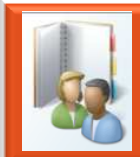
(2a) Lưu đồ bong bóng (Bubble chart)

- Biểu thị luồng xử lý dữ liệu
- Ký pháp



(2b) Cấu trúc phân cấp
(Hierarchical structured chart)

- Là phân cấp biểu thị quan hệ phụ thuộc giữa các môđun và giao diện (interface) giữa chúng
- Các quy ước:
  - Không liên quan đến trình tự gọi các môđun, nhưng ngầm định là từ trái qua phải
  - Mỗi môđun xuất hiện trong cấu trúc 1 lần, có thể được gọi nhiều lần
  - Quan hệ trên dưới: không cần nêu số lần gọi

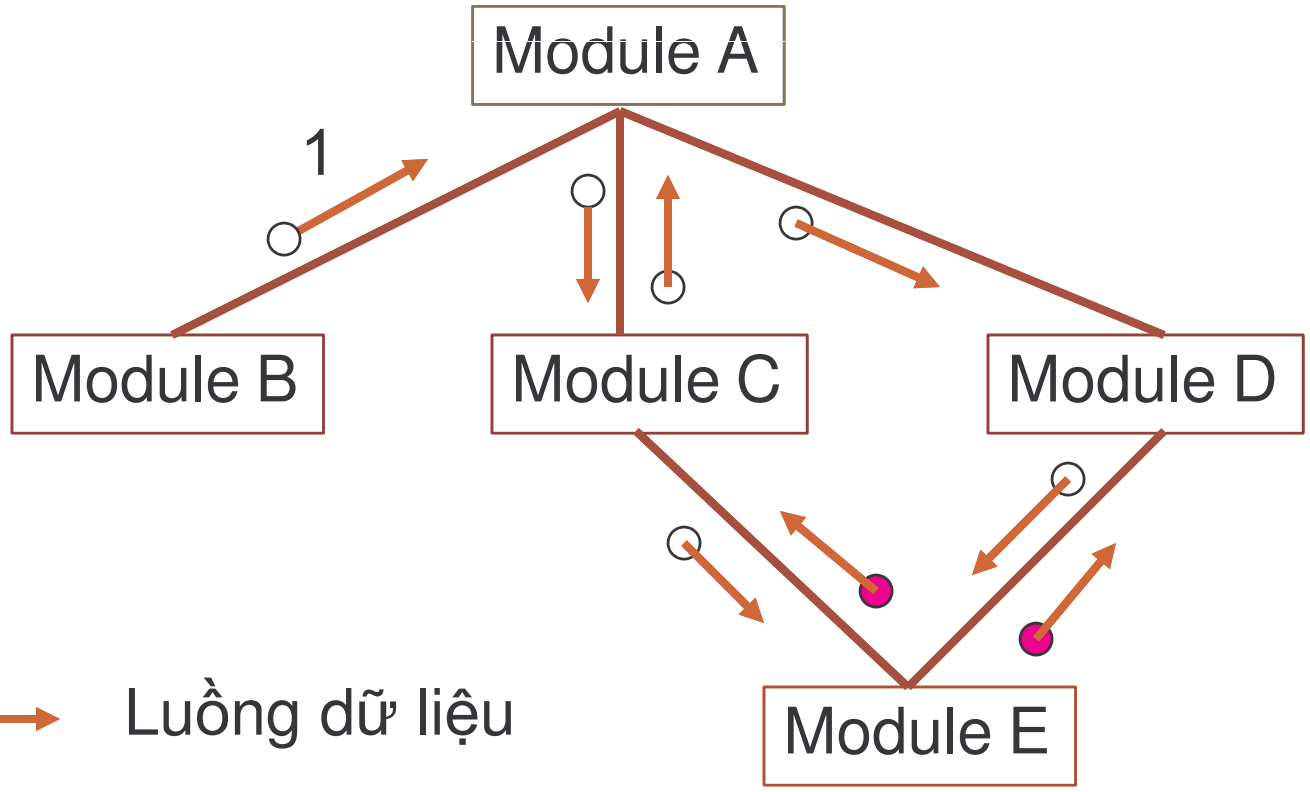


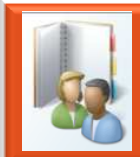
Hierarchical structured chart

Các quy ước (tiếp):

- Tên môđun biểu thị chức năng ("làm gì"), đặt tên sao cho các môđun ở phía dưới tổng hợp lại sẽ biểu thị đủ chức năng của môđun tương ứng phía trên
- Biến số (arguments) biểu thị giao diện giữa các môđun, biến số ở các môđun gọi/bị gọi có thể khác nhau
- Mũi tên với đuôi tròn trắng biểu thị dữ liệu, đuôi tròn đen (hồng) biểu thị flag
- Chiều của mũi tên là hướng truyền tham số

Hierarchical structured chart



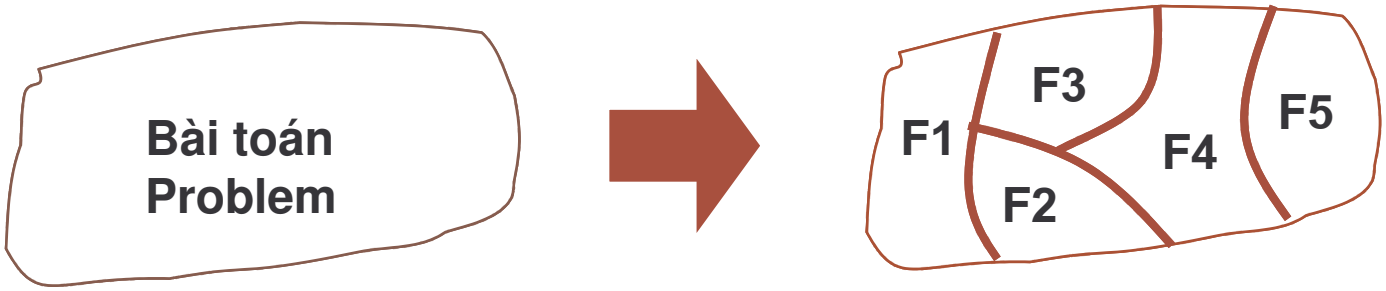


(3) Phương pháp phân chia STS, TR

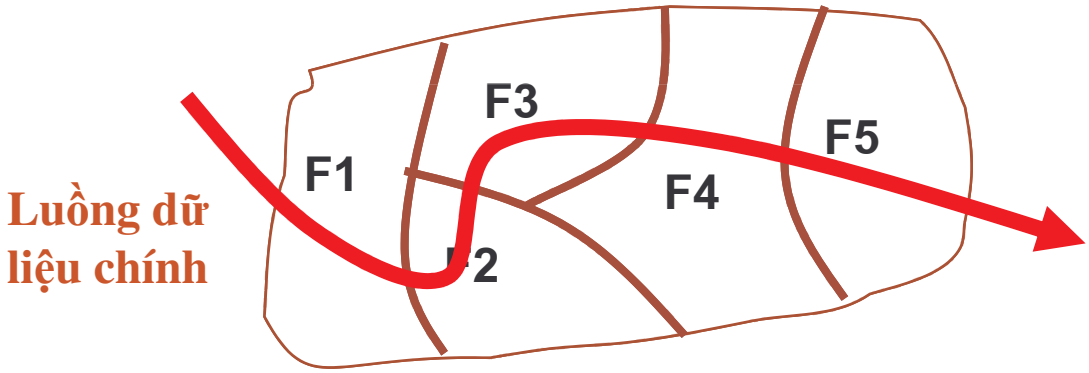
- **Thiết kế cấu trúc:**
  - Phương pháp phân chia STS (Source/Transform/Sink: Nguồn/Biếnđổi/Hấpthụ)
  - Phương pháp phân chia TR (Transaction)
- Minh họa phân chia chức năng theo bong bóng của DFD (biểu đồ luồng dữ liệu)

(3a) Phương pháp phân chia STS

1) Chia đối tượng "bài toán" thành các chức năng thành phần



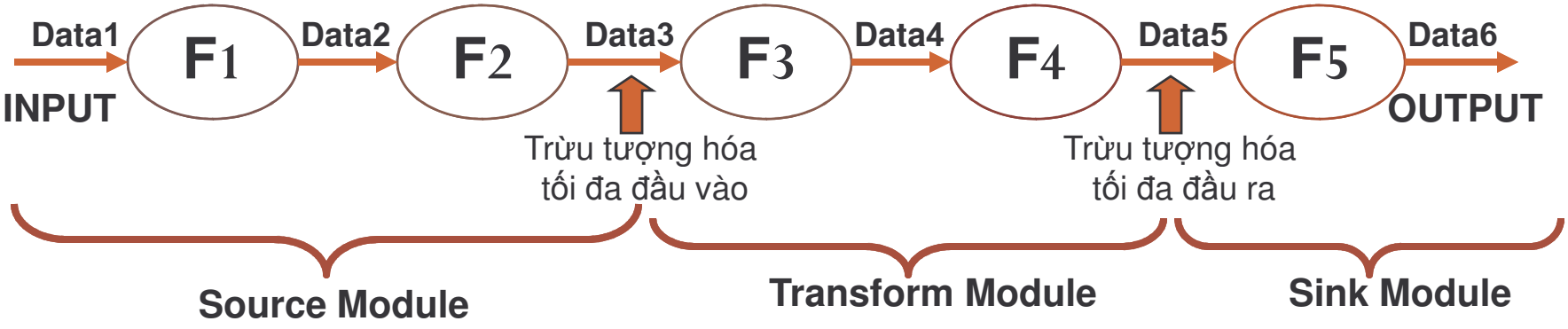
2) Tìm ra luồng dữ liệu chính đi qua các chức năng: từ đầu vào (Input) tới đầu ra (Output)



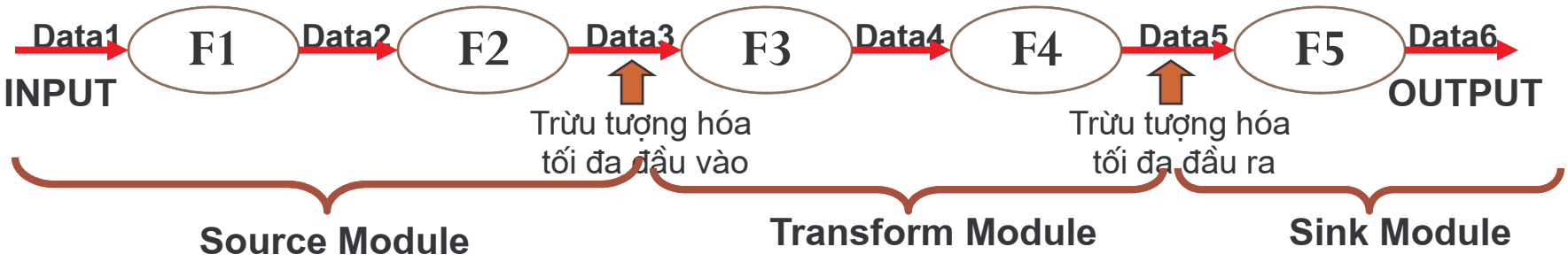
Quyết định bong bóng và dữ liệu

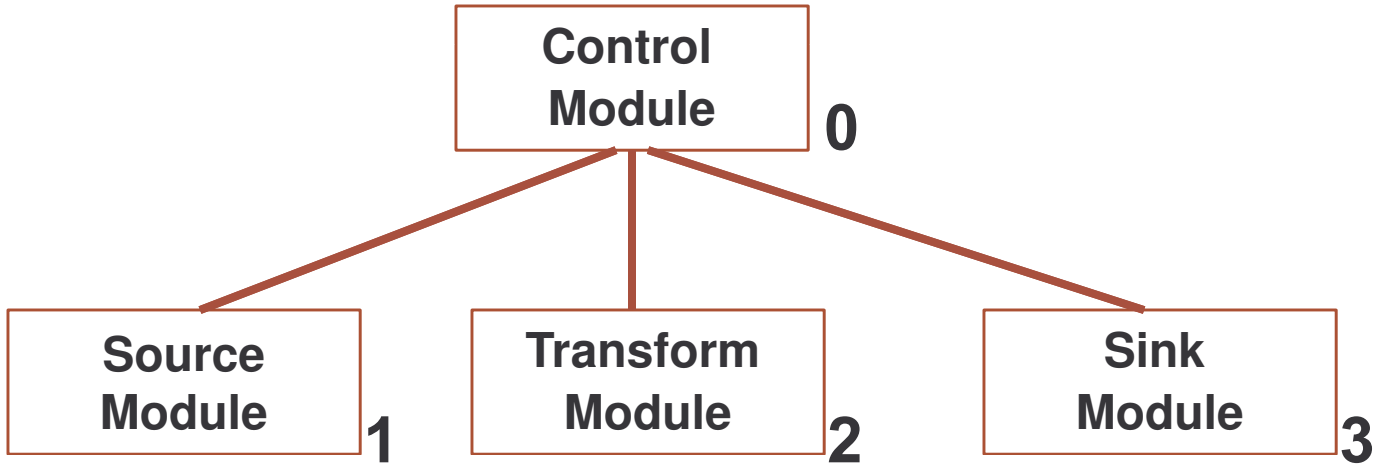
3) Theo luồng dữ liệu chính: thay từng chức năng bởi bong bóng và làm rõ dữ liệu giữa các bong bóng

4) Xác định vị trí trừu tượng hóa tối đa đầu vào và đầu ra



5) Chuyển sang sơ đồ phân cấp





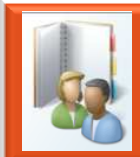
KỸ THUẬT THIẾT KẾ CHƯƠNG TRÌNH

7.1 Thiết kế chương trình là gì ?

7.2 Phương pháp thiết kế chương trình

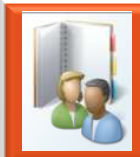
7.3 Công cụ thiết kế

- Là thiết kế chi tiết cấu trúc bên trong của phần mềm: thiết kế tính năng từng môđun và giao diện tương ứng
- Cấu trúc ngoài của phần mềm: thiết kế hệ thống
- Trình tự xử lý bên trong: Thuật toán (giải thuật, Algorithm); Logic



7.2 Phương pháp thiết kế chương trình

- Không có trạng thái mờ (fuzzy), để đảm bảo thiết kế cấu trúc trong đúng đắn
- Ngôn ngữ lập trình phù hợp
- Triển khai đúng đắn đặc tả chức năng các môđun và chương trình nhờ phương pháp luận thiết kế chi tiết
- Dùng quy trình thiết kế dễ chuẩn hóa từng bước



Kỹ thuật thiết kế chương trình

Kỹ thuật thiết kế mô hình hệ phần mềm

- **Hướng tiến trình** (process) :

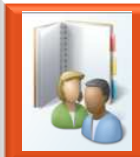
  Kỹ thuật thiết kế cấu trúc điều khiển

- **Hướng cấu trúc dữ liệu** (data):

  Kỹ thuật thiết kế cấu trúc dữ liệu

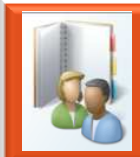
- **Hướng sự vật / đối tượng** (object):

  Kỹ thuật thiết kế hướng đối tượng



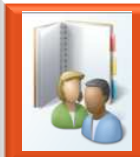
7.2.1 Lập trình cấu trúc hóa

- Khái niệm cơ bản: tuần tự, nhánh (chọn), lặp; cấu trúc mở rộng, tiền xử lý, hậu xử lý
- Những điểm lợi khi thiết kế thuật toán
  - Tính độc lập của môđun: chỉ quan tâm vào-ra
  - Làm cho chương trình dễ hiểu
  - Dễ theo dõi chương trình thực hiện
  - Hệ phức tạp sẽ dễ hiểu nhờ tiếp cận phân cấp



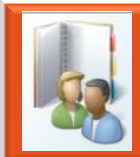
Loại bỏ GOTO

- GOTO dùng để làm gì?
  - Cho phép thực hiện các bước nhảy đến một nhãn nhất định
- Tại sao cần loại bỏ GOTO ?
  - Phá vỡ tính cấu trúc của lập trình cấu trúc hóa
- Phương pháp loại bỏ GOTO
- Có thể loại bỏ GOTO trong mọi trường hợp?
- Thế nào là “kỹ năng lập trình cấu trúc”



Lưu ý khi thiết kế chương trình

- Phụ thuộc vào kỹ năng và kinh nghiệm của người thiết kế
- Cần chuẩn hóa tài liệu đặc tả thiết kế chi tiết
- Khi thiết kế cấu trúc điều khiển của giải thuật, vì theo các quy ước cấu trúc hóa nên đôi khi tính sáng tạo của người thiết kế bị hạn chế, bó buộc theo khuôn mẫu đã có

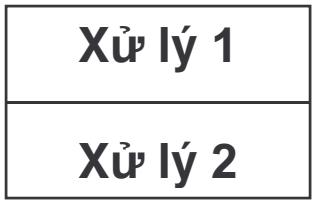


7.2.2 Lưu đồ cấu trúc hóa

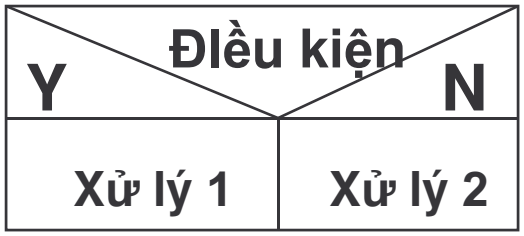
- Tác dụng của lưu đồ (flow chart)
- Quy phạm (discipline)
- Trừu tượng hóa thủ tục
- Lưu đồ cấu trúc hóa
  - Cấu trúc điều khiển cơ bản
  - Chi tiết hóa từng bước giải thuật
  - Thể hiện được trình tự điều khiển thực hiện

Lưu đồ Nassi-Shneiderman (NS chart by IBM)

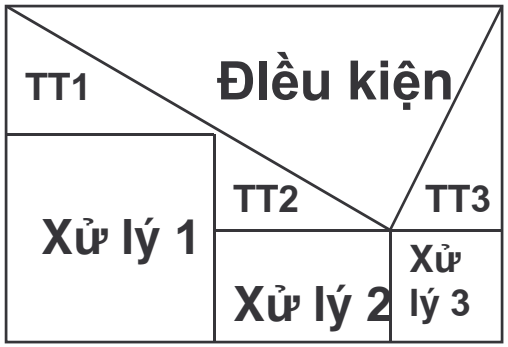
a- Nối (concatination)



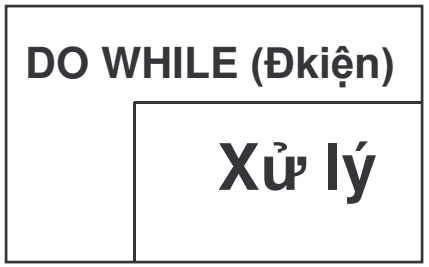
b- Chọn (selection)

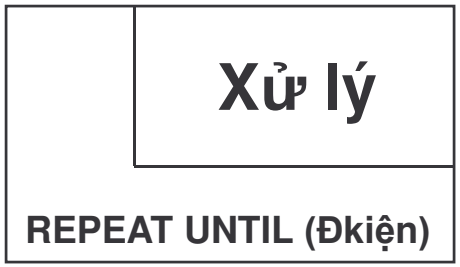


c- Đa nhánh (CASE)



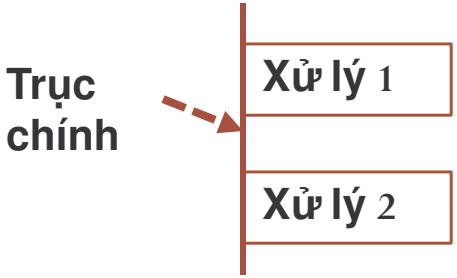
d- Lặp (repetition)



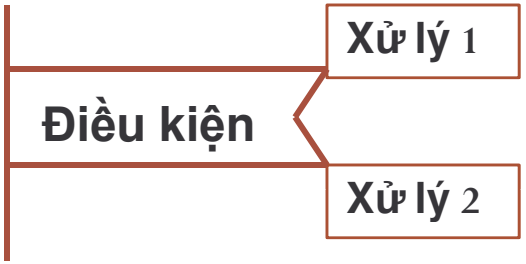


Lưu đồ Phân tích bài toán (PAD chart by Hitachi)

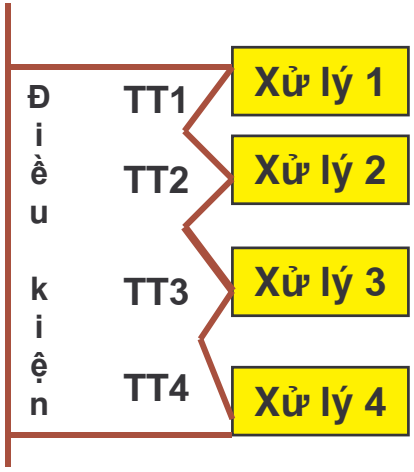
a- Nối (concatination)



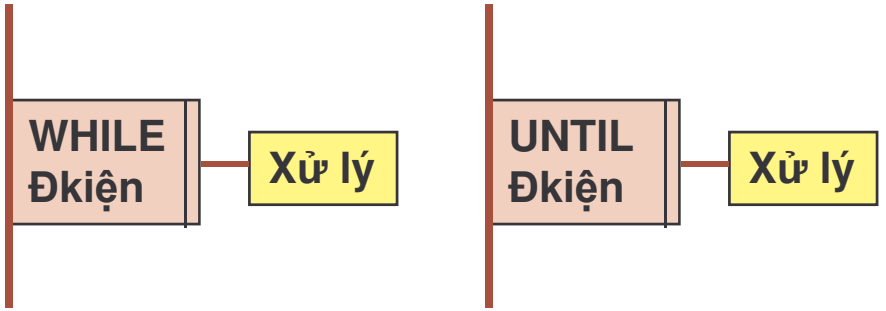
b- Chọn (selection)

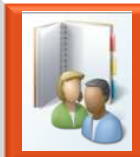


c- Đa nhánh (CASE)



d- Lặp (repetition)



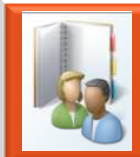


7.2.3 Về Phương pháp Giắc-sơn (Jackson's method)

- JSP: Jackson Structured Programming
- Các ký pháp:
  - Cơ sở (elementary)
  - Tuần tự (sequence)
  - Lặp
  - Rẽ nhánh

Trình tự thiết kế chung

- Thiết kế cấu trúc dữ liệu (Data step)
- Thiết kế cấu trúc chương trình (Program step)
- Thiết kế thủ tục (Operation step)
- Thiết kế đặc tả chương trình (Text step)



7.2.4 Về Phương pháp Wa-ny (Warnier's method)

- Khái niệm chung
- Trình tự thiết kế
  - Thiết kế dữ liệu ra
  - Thiết kế dữ liệu vào
  - Thiết kế cấu trúc chương trình
  - Thiết kế lưu đồ
  - Thiết kế lệnh thủ tục
  - Thiết kế đặc tả chi tiết

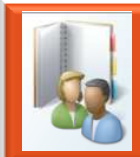
KỸ THUẬT LẬP TRÌNH

8.1 Lịch sử phát triển của ngôn ngữ lập trình

8.2 Cấu trúc chương trình

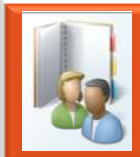
- Cấu trúc dữ liệu dễ hiểu
- Cấu trúc thuật toán dễ hiểu

8.3 Các công cụ lập trình



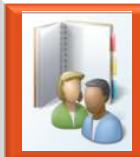
8.1 Lịch sử ngôn ngữ lập trình

- Các ngôn ngữ thế hệ thứ nhất: (1GL)
  - Ngôn ngữ lập trình mã máy (machine code)
  - Ngôn ngữ lập trình assembly
- Các ngôn ngữ thế thế thứ hai (2GL)
  - FOTRAN, COBOL, ALGOL, BASIC
  - Phát triển 1950-1970
- Các ngôn ngữ thế hệ thứ ba (3GL)
  - Ngôn ngữ lập trình cấp cao vạn năng (cấu trúc)
  - Lập trình hướng đối tượng
  - Lập trình hướng suy diễn – logic
- Các ngôn ngữ thế hệ thứ tư (4GL)
  - Truy vấn
  - Các ngôn ngữ hỗ trợ quyết định UML, Rational Rose…



8.2 Cấu trúc dữ liệu dễ hiểu

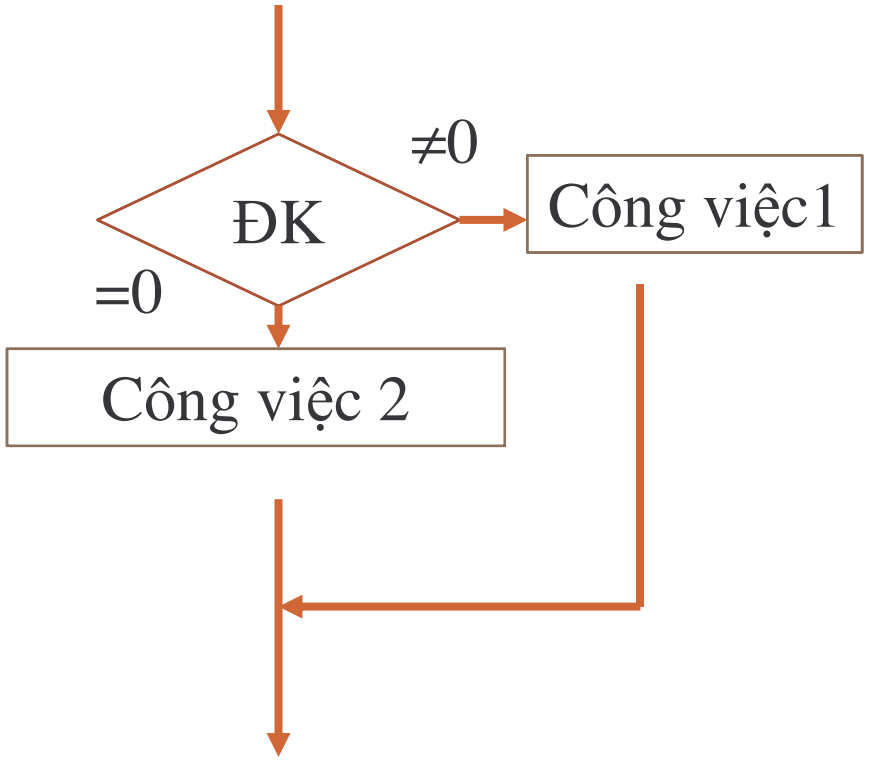
- Nên xác định tất cả các cấu trúc dữ liệu và các thao tác cần thực hiện trên từng cấu trúc dữ liệu
- Việc biểu diễn/khai báo các cấu trúc dữ liệu chỉ nên thực hiện ở những mô đun sử dụng trực tiếp dữ liệu
- Nên thiết lập và sử dụng từ điển dữ liệu khi thiết dữ liệu

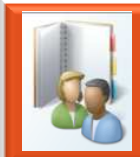


Cấu trúc thuật toán dễ hiểu

- Algorithm
- Structured coding và 9 điểm lưu ý:
  - Tuân theo quy cách lập trình
  - Một đầu vào, một đầu ra
  - Tránh GOTO, trừ khi phải ra khỏi lặp và dừng
  - Dùng comments hợp lý
  - Dùng tên biến có nghĩa, gợi nhớ
  - Cấu trúc lồng rõ ràng
  - Tránh dùng CASE / switch nhiều hoặc lồng nhau
  - Mã nguồn 1 chương trình / môđun nên viết trên 1 trang
  - Tránh viết nhiều lệnh trên 1 dòng

IF THEN / IF THEN ELSE

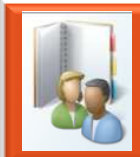




IF THEN / IF THEN ELSE

PASCAL	Ngôn ngữ C

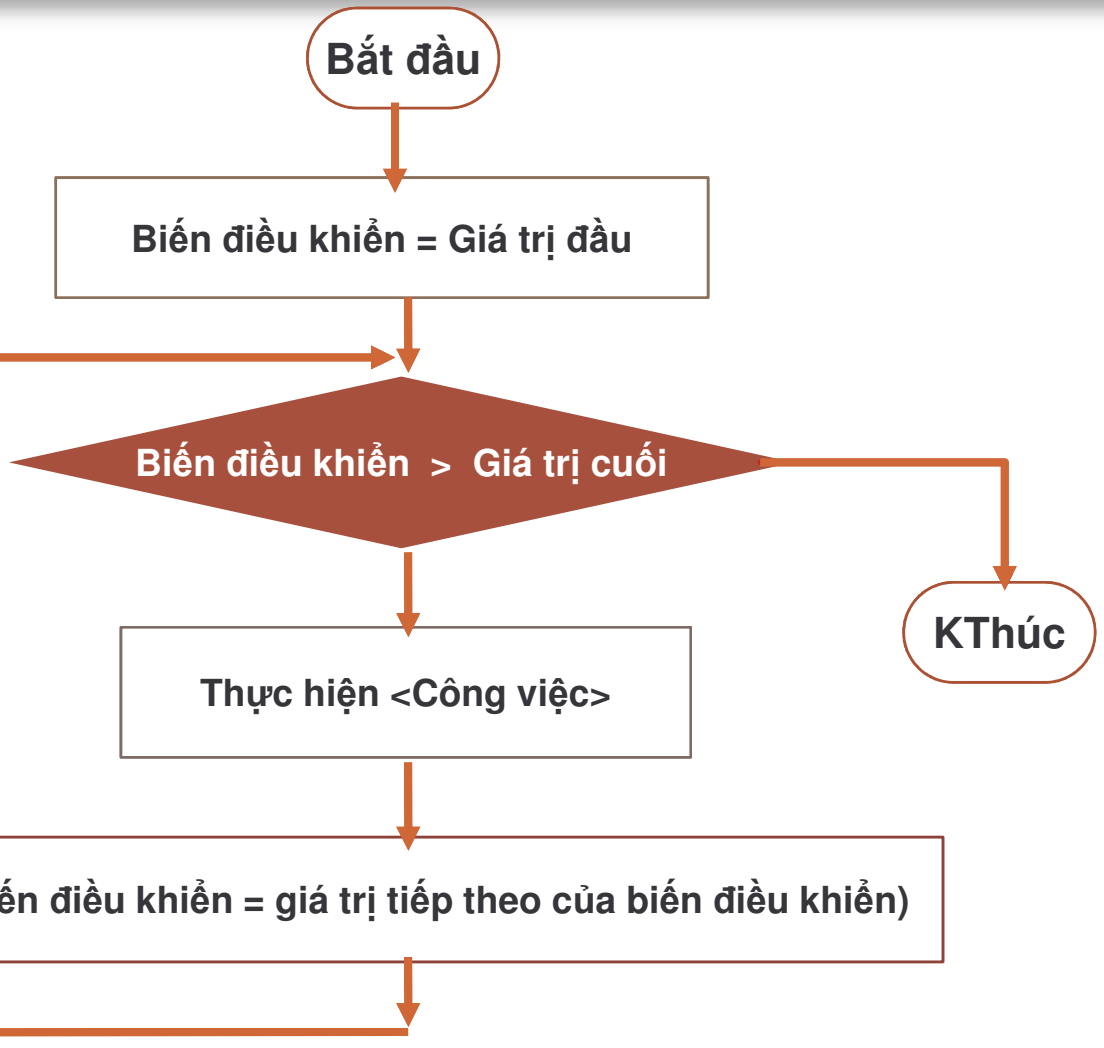
if điều kiện **then**
begin
công việc 1
end;
else
begin
công việc 2
end
	if (điều kiện)
	{ công việc 1}
	else
	{công việc 2}

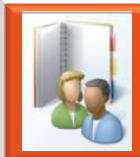


CASE / switch

PASCAL	Ngôn ngữ C
CASE <biểu thức> **OF**	**Switch** (<bthức>)
gtrị1: <việc 1>;	{
gtrị2: <việc 2>;	**case** <gtrị1>: <việc1>;[break;]
.....	**case** <gtrị2>: <việc2>; [break;]
gtrịN: <việc N>;	**case** <gtrịN>: <việcN>; [break;]
ELSE <việc N+1>;	[**default** : <việcN+1>; [break;]]
END;	}

FOR TO / DOWNTO





FOR TO / DOWNTO

PASCAL

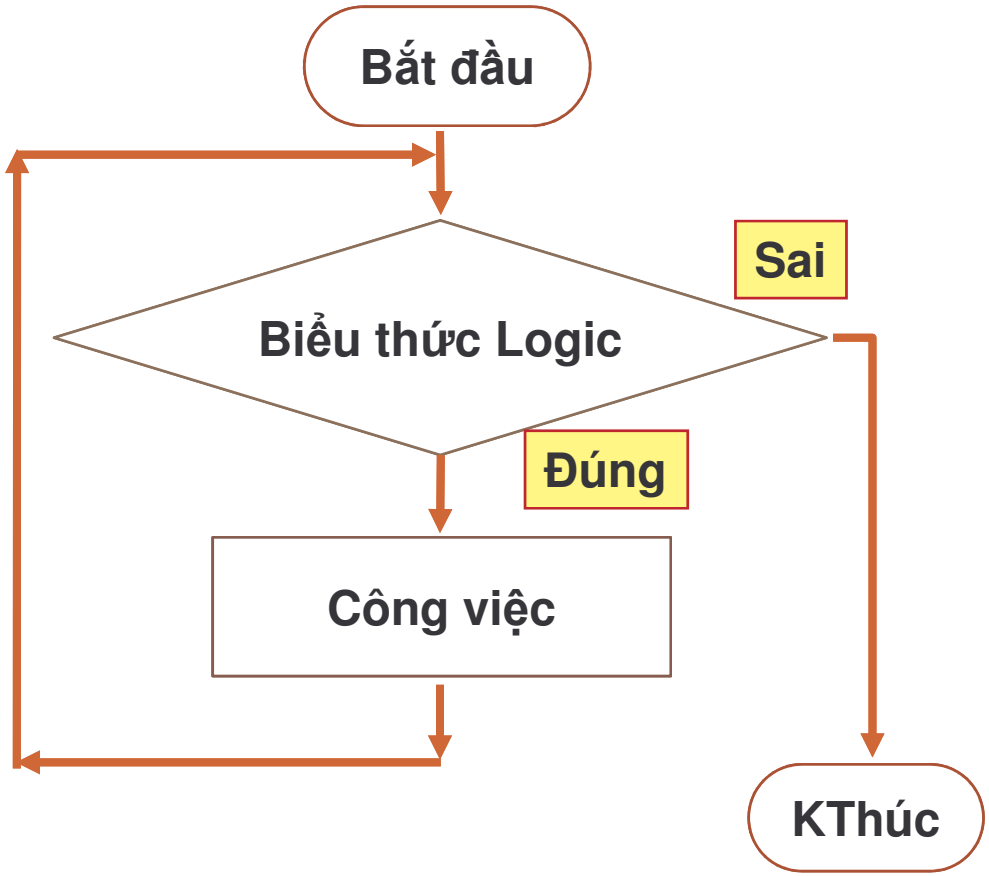
```
FOR  biếnđkhiển := GTđầu   TO   GTCuối   DO
        begin
                <việc>
        end;
```

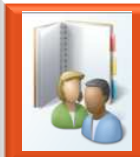
Ngôn ngữ C

```
for ( [biểuthức1] ; [biểuthứcĐK]; [biểuthức2] )
        { <việc>; }
```

Đặc biệt: có các lệnh thoát: Break; Continue; Exit

DO WHILE



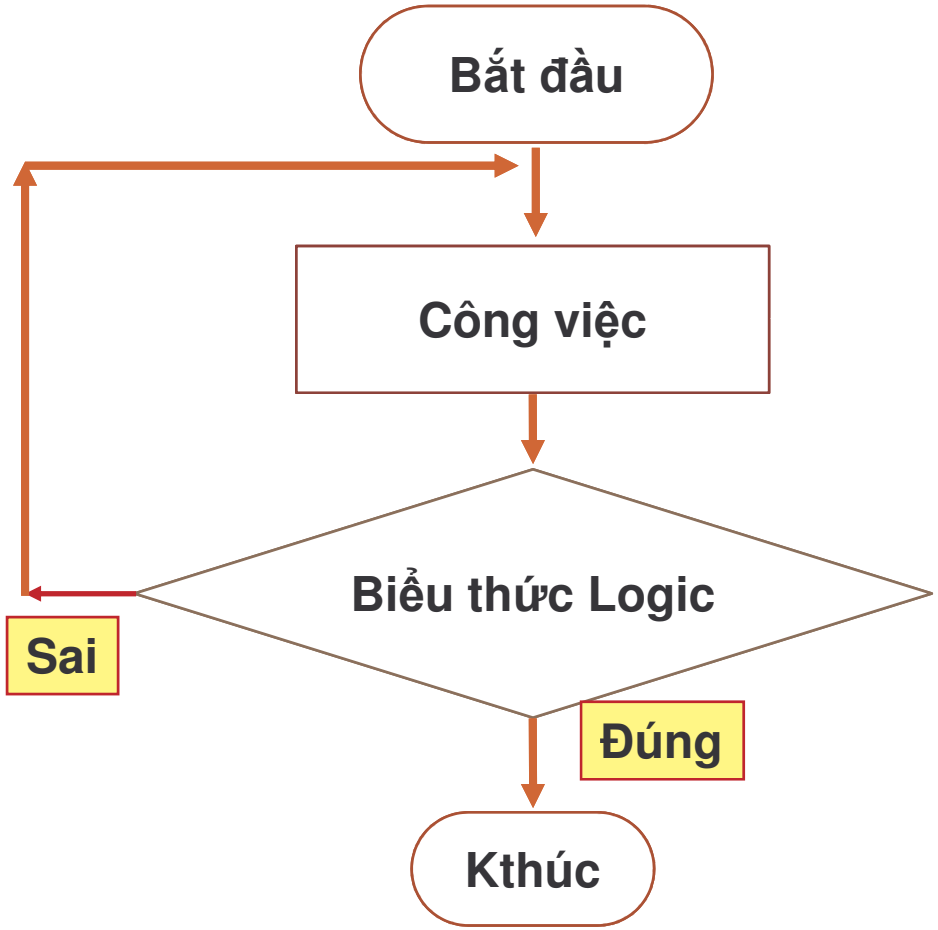


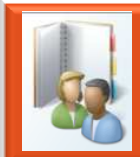
PASCAL	Ngôn ngữ C

While BiểuthứcBoolean **DO**
begin
<Công việc>
end;
	while (<biểuthứcĐK>)
	{
	<Công việc>;
	}

- Kiểm tra điều kiện trước khi thực hiện
- Lỗi thường gặp: Lặp vô hạn

REPEAT UNTIL

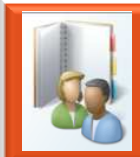




PASCAL	Ngôn ngữ C

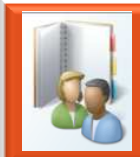
Repeat
<Công việc>
until Biểu_thức_Boolean;
	do {
	<Công việc>;
	}
	while (<biểuthứcĐK>);

- Có sự khác nhau giữa hai ngôn ngữ?



Chú thích trong chương trình

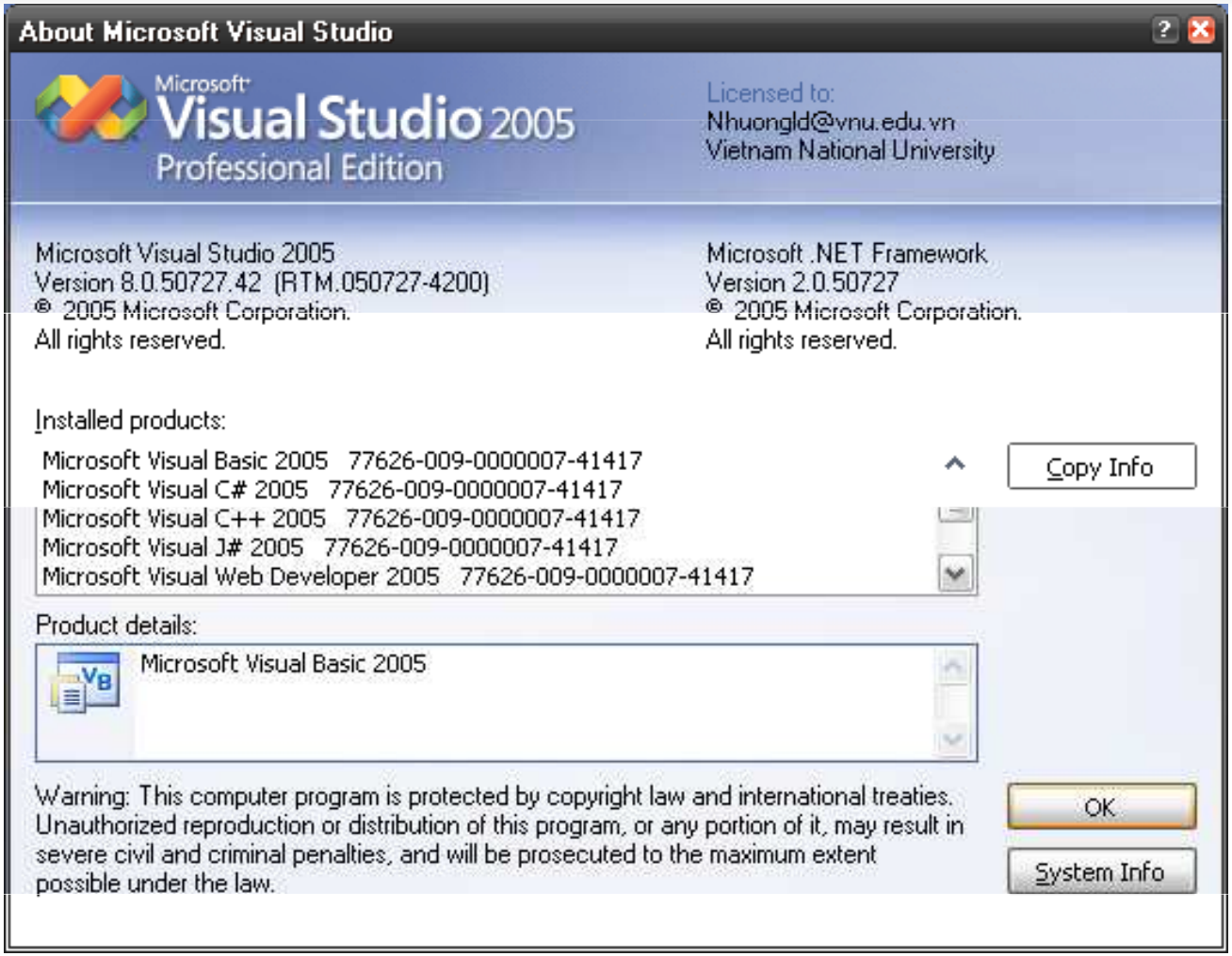
- Tại sao cần đặt các chú thích trong chương trình ?
- Vị trí đặt các chú thích trong chương trình
  - Thành phần/ Module
  - Lớp
  - Hàm/thủ tục
  - Các vị trí đặc biệt khác
- Một số quy định khi đặt chú thích:
  - Ngắn gọn
  - Gợi nhớ



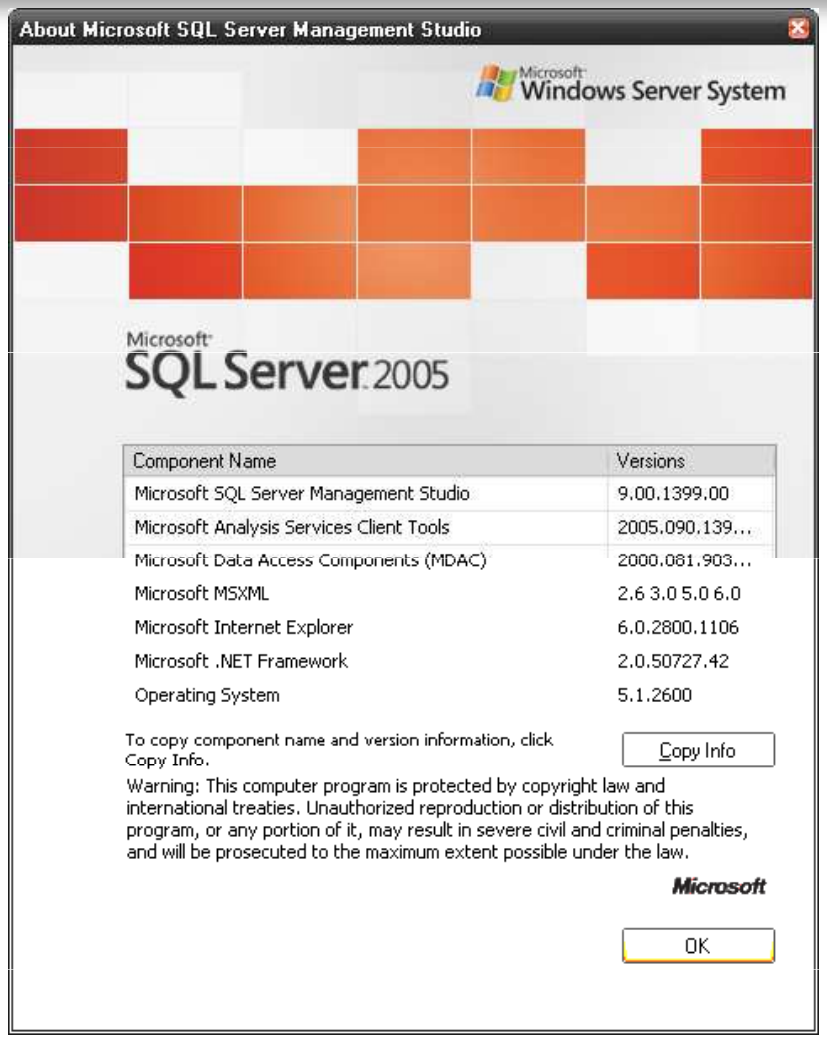
8.3 Các công cụ lập trình

- Environments: DOS, WINDOWS, UNIX/LINUX
- Editors, Compilers, Linkers, Debuggers
- TURBO C, Turbo C++, PASCAL
- MS C, Visual Basic, Visual C++, ASP
- UNIX/LINUX: C/C++, gcc (Gnu C Compiler)
- JAVA, CGI, Perl
- C#, VB.NET, J#, ASP.NET,
- .NET Framework

8.3 Các công cụ lập trình



8.3 Các công cụ lập trình



8.3 Các công cụ lập trình

Department of Software Engineering.
Faculty of Mathematic & Informatic. Hai Phong University.

Công nghệ phần mềm

(Introduction to Software Engineering)

Phần V: Kiểm thử và bảo trì

Test & Maintenance

Editor: **LÊ ĐẮC NHƯỜNG**
Email: **Nhuongld@yahoo.com**
Phone: 0987394900

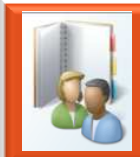
KIỂM THỬ PHẦN MỀM

9.1 Khái niệm kiểm thử

9.2 Phương pháp thử

9.3 Kỹ thuật thiết kế trường hợp thử

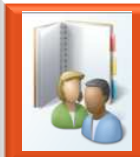
9.4 Phương pháp thử các môđun



9.1 Khái niệm kiểm thử

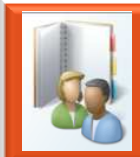
Định nghĩa kiểm thử:

- Là mấu chốt của đảm bảo chất lượng phần mềm
- Là tiến trình (và là nghệ thuật) nhằm phát hiện lỗi bằng việc xem xét lại đặc tả, thiết kế và mã hoá.
- Kiểm thử thành công là phát hiện ra lỗi; kiểm thử không phát hiện ra lỗi là kiểm thử dở (Sue A.Conger- The New SE)



Những khó khăn khi kiểm thử

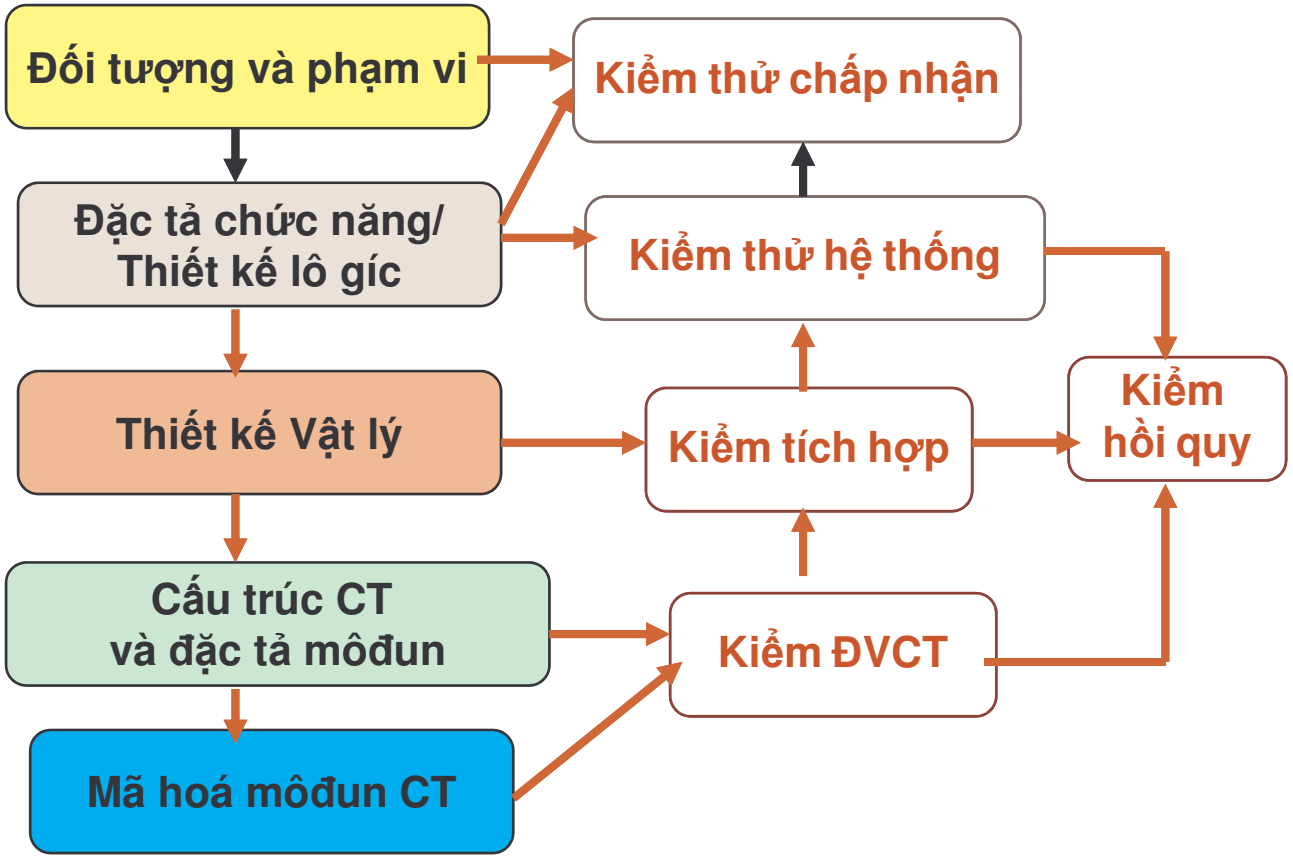
- Nâng cao chất lượng phần mềm nhưng không vượt quá chất lượng khi thiết kế: chỉ phát hiện các lỗi tiềm tàng và sửa chúng
- Phát hiện lỗi bị hạn chế do thực hiện thủ công là chính
- Dễ bị ảnh hưởng của tâm lý trong khi kiểm thử
- Khó đảm bảo tính đầy đủ của kiểm thử

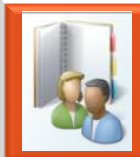


6 điểm lưu ý khi kiểm thử

1) Chất lượng phần mềm do khâu thiết kế quyết định là chủ yếu, chứ không phải khâu kiểm thử
2) Tính dễ kiểm thử phụ thuộc vào cấu trúc chương trình
3) Người kiểm thử và người phát triển nên khác nhau
4) Dữ liệu thử cho kết quả bình thường thì không có ý nghĩa nhiều, cần có những dữ liệu kiểm thử mà phát hiện ra lỗi
5) Khi thiết kế trường hợp thử, không chỉ dữ liệu kiểm thử nhập vào, mà phải thiết kế trước cả dữ liệu kết quả sẽ có
6) Khi phát sinh thêm trường hợp thử thì nên thử lại những trường hợp thử trướcđó để tránh ảnh hưởng lan truyền sóng

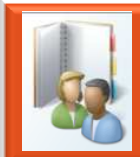
Tương ứng giữa vòng đời dự án và kiểm thử





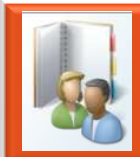
9.2 Phương pháp thử: thử tĩnh

- Kiểm thử trên bàn hay Kiểm thử tĩnh: giấy và bút trên bàn, kiểm tra logic, lần từng chi tiết ngay sau khi lập trình xong
- Đi xuyên suốt (walk through)
- Thanh tra (inspection)



Kiểm thử trên máy

- Gỡ lỗi bằng máy (machine debug) hay kiểm thử động: Dùng máy chạy chương trình để điều tra trạng thái từng động tác của chương trình
- 9 bước của trình tự kiểm thử bằng máy



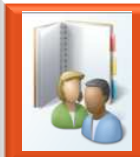
Trình tự kiểm thử bằng máy

(1) Thiết kế trường hợp thử theo thử trên bàn

(2) Trường hợp thử phải có cả kết quả kỳ vọng sẽ thu được

(3) Dịch chương trình nguồn và tạo môđun tải để thực hiện

(4) Khi trường hợp thử có xử lý tệp vào-ra, phải làm trước trên bàn việc xác định miền của các tệp



Trình tự kiểm thử bằng máy (tiếp)

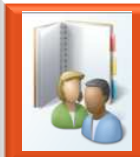
(5) Nhập dữ liệu đã thiết kế cho trường hợp kiểm thử

(6) Điều chỉnh môi trường thực hiện môđun tải (tạo thủ tục đưa các tệp truy cập tệp vào chương trình)

(7) Thực hiện môđun tải và ghi nhận kết quả

(8) Xác nhận kết quả với kết quả kỳ vọng

(9) Lặp lại thao tác (5)-(8)



9.3 Kỹ thuật thiết kế trường hợp thử

- Kỹ thuật thiết kế trường hợp thử dựa trên đặc tả bề ngoài của chương trình: **Kiểm thử hộp đen** (Black box test):

  WHAT ?

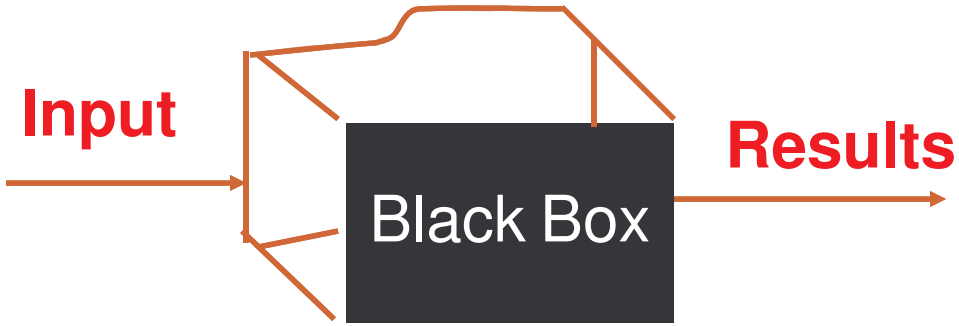
- Kỹ thuật thiết kế trường hợp thử dựa trên đặc tả bên trong của chương trình: **Kiểm thử hộp trắng** (white box test):

  HOW ?

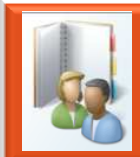
- Kiểm thử Top-Down hay Bottom-Up

9.3.1 Kiểm thử hộp đen

- Phân đoạn tương đương
- Phân tích giá trị biên
- Đoán lỗi

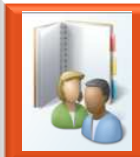


Black box Data Testing Strategy



9.3.2 Phương pháp phân đoạn tương đương (Equivalence Partition)

- **Mục đích**: giảm số lượng test bằng cách chọn các tập dữ liệu đại diện
- **Thực hiện**: Chia dữ kiệu vào thành các đoạn, mỗi đoạn đại diện cho một số dữ liệu → việc kiểm thử chỉ thực hiện trên đại diện đó
- **ưu điểm**: Test theo mức trừu tượng hơn là trường. áp dụng: màn hình, menu hay mức quá trình



9.3.3 Phương pháp phân tích giá trị biên (Boundary value analysis)

- Là 1 trường hợp riêng của phân đoạn
- Thí dụ: nếu miền dữ liệu là tháng thì giá trị 0 hay >12 là không hợp lệ
- Thường sử dụng trong kiểm thử môđun

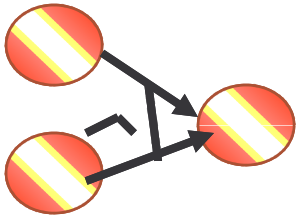
Phương pháp đoán lỗi (Error Guessing)

- Dựa vào trực giác và kinh nghiệm
- Thí dụ lỗi chia cho 0. Nếu môđun có phép chia thì phải kiểm thử lỗi này
- Nhược điểm: không phát hiện hết lỗi

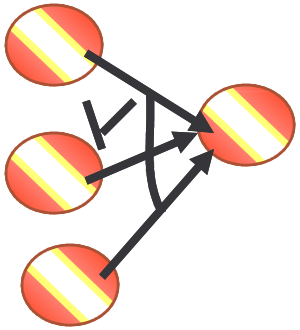
9.3.4 Phương pháp đồ thị nguyên nhân - kết quả (Cause-effect Graphing)

Mã tuần tự

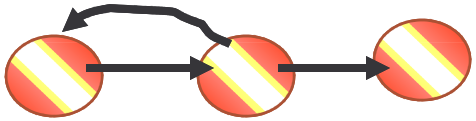
Phủ định



and



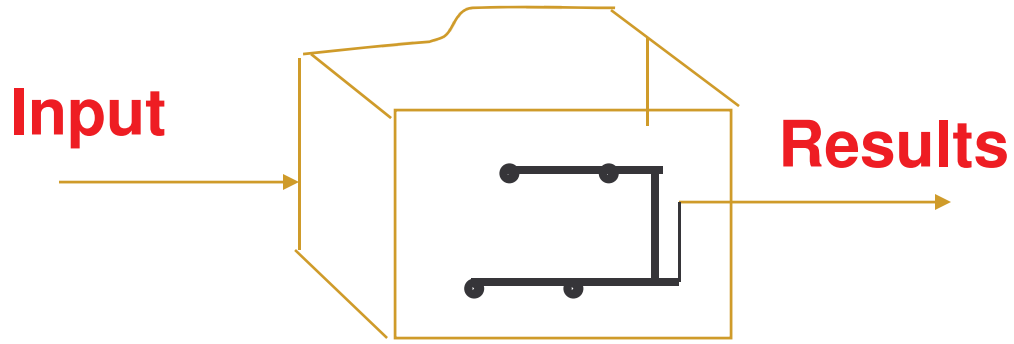
Or



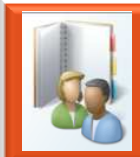
Do Until

9.3.5 Kiểm thử hộp trắng

- Bó các lệnh
- Bó các rẽ nhánh
- Bó các điều kiện
- Bó các điều kiện - rẽ nhánh

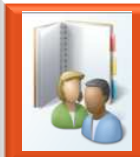


White Box Data Testing Strategy



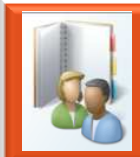
9.3.6 Trình tự thiết kế

- Kiểm thử môđun
- Kiểm thử tích hợp
  - Kiểm thử tích hợp trên xuống
  - Kiểm thử tích hợp dưới lên
  - Kiểm thử hồi qui



9.4 Kỹ thuật kiểm thử môđun

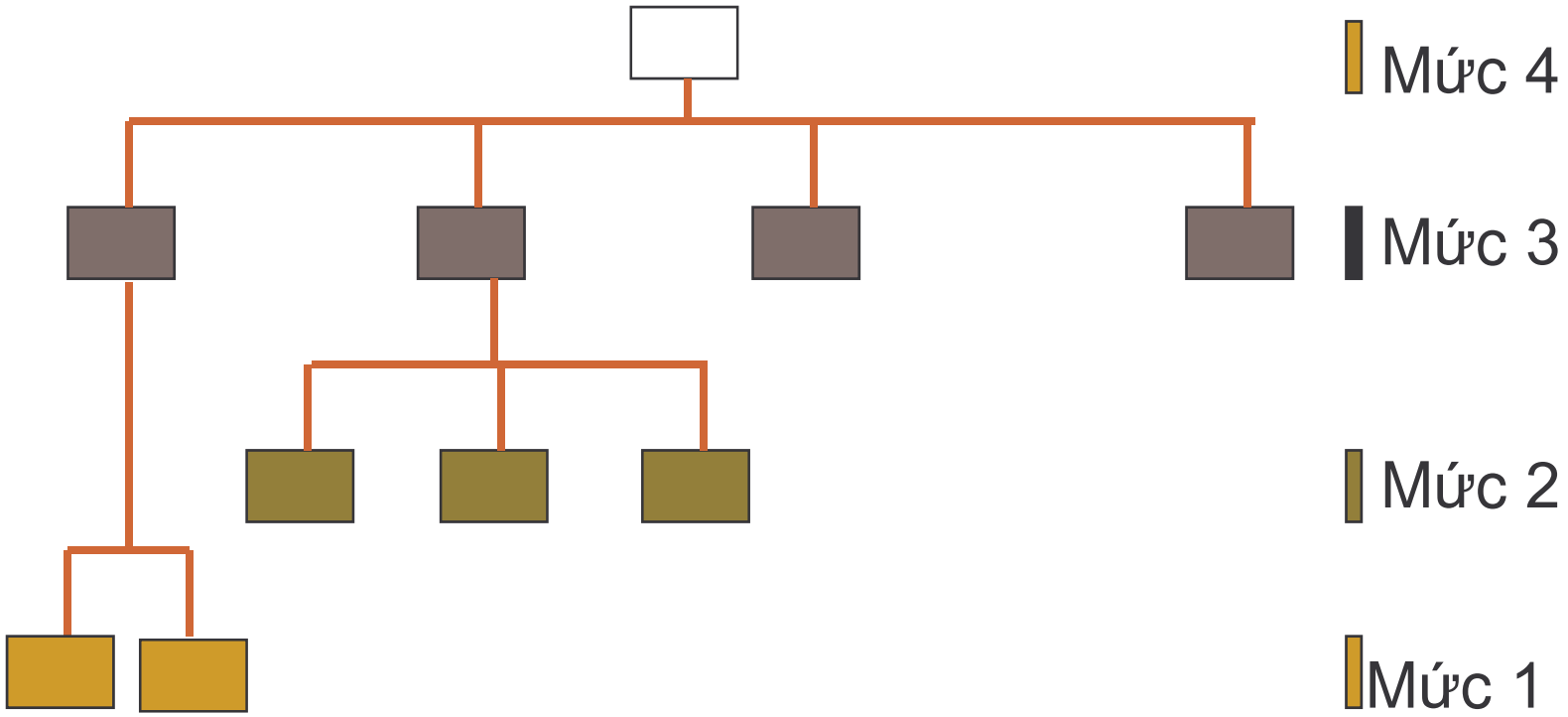
- **Kiểm thử tích hợp môđun**
  - Kiểm thử dưới lên (Bottom-up Test)
  - Kiểm thử trên xuống (Top-down Test)
  - Kiểm thử cột trụ (Big bung Test)
  - Kiểm thử kẹp (Sandwich Test)

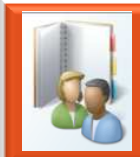


9.4.1 Kiểm thử dưới lên (Bottom-up Test)

- Các môđun mức thấp được tổ hợp vào các chùm thực hiện một chức năng con
- Viết trình điều khiển phối hợp vào/ ra và kiểm thử
- Kiểm thử chùm/bó
- Loại bỏ trình điều khiển và chuyển lên mức trên

9.4.1 Kiểm thử dưới lên Bottom-up Test (Tiếp)

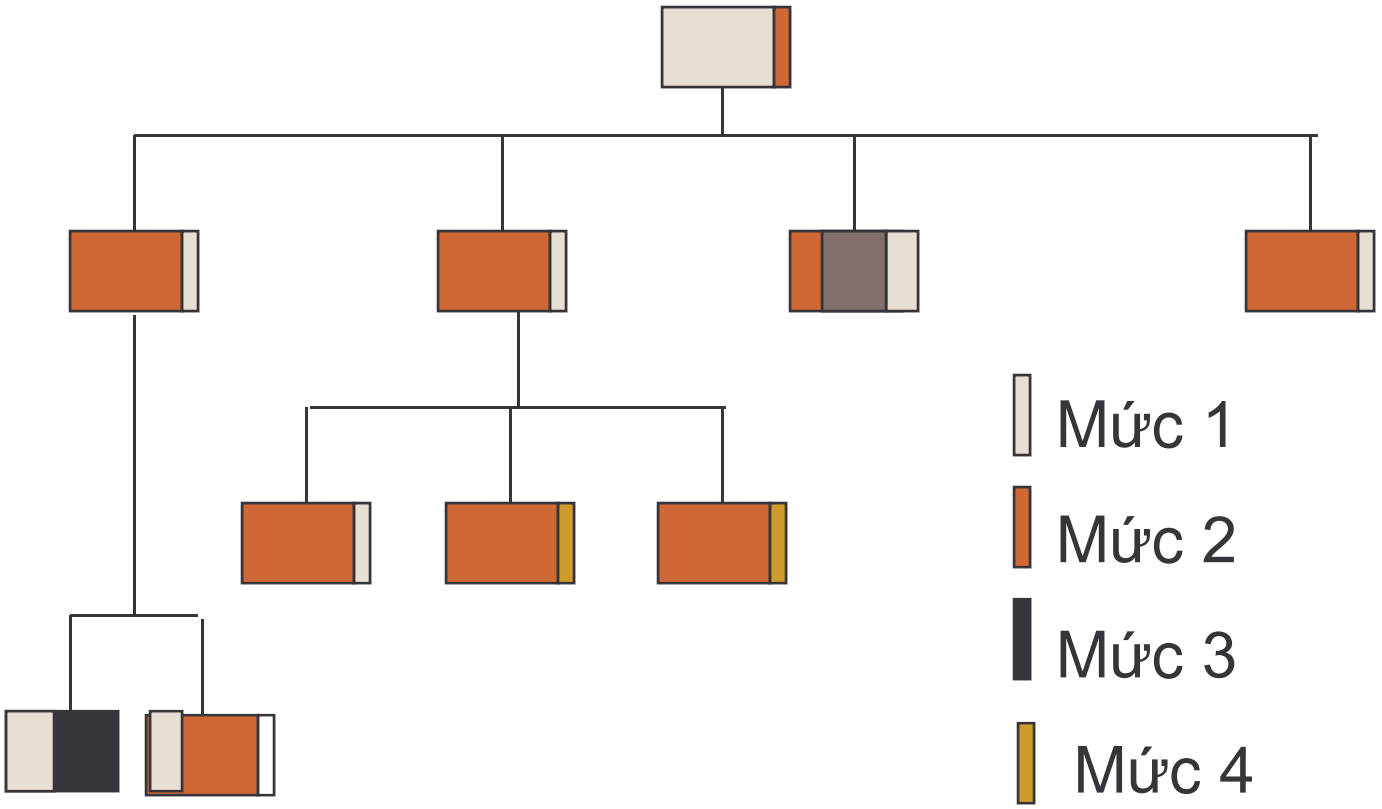


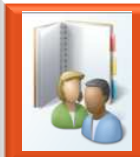


9.4.2 Kiểm thử trên xuống (Top-down Test)

- Môđun điều khiển chính được dùng như trình điều khiển kiểm thử, gắn các nút con trực tiếp vào nó
- Thay các nút con bằng các môđun thực tại (theo chiều sâu / ngang)
- Kiểm thử từng môđun được gắn vào
- Các 1 nút thử xong được thử tiếp nút khác
- Kiểm thử hồi quy

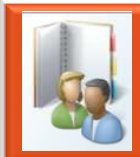
9.4.2 Kiểm thử trên xuống Top-down Test (tiếp)





9.5 Kiểm thử cột trụ (Big bung Test)

- Tích hợp không tăng dần
- Tất các các môđun đều được tổ hợp trước
- Toàn bộ chương trình được kiểm thử tổng thể
- Khó khăn: khó cô lập lỗi, khi chữa xong lỗi này có thể lỗi mới lại phát sinh



9.6 Sandwich Test

- Tích hợp trên xuống cho các mức trên cấu trúc chương trình
- Tích hợp dưới lên cho các mức phụ thuộc

9.7 Kiểm thử hệ thống

- Kiểm thử phục hồi: bắt buộc phần mềm hỏng nhiều cách để kiểm chứng phục hồi
- Kiểm thử an toàn: kiểm chứng cơ chế bảo vệ
- Kiểm thử gay cấn
- Kiểm thử hiệu năng

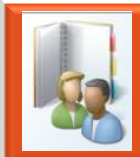
PHƯƠNG PHÁP BẢO TRÌ

(Maintenance Method)

10.1 Bảo trì là gì?

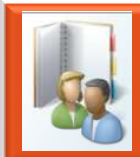
10.2 Trình tự nghiệp vụ bảo trì

10.3 Những vấn đề về bảo trì hiện nay



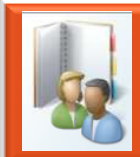
10.1 Bảo trì là gì?

- **Định nghĩa:** Bảo trì là công việc tu sửa, thay đổi phần mềm đã được phát triển (chương trình, dữ liệu, JCL, các loại tư liệu đặc tả,...) theo những lý do nào đó.
- Các hình thái bảo trì: bảo trì để
  - Tu chỉnh
  - Thích hợp
  - Cải tiến
  - Phòng ngừa



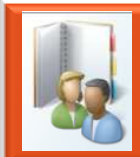
Bảo trì để tu sửa

- Là bảo trì khắc phục những khiếm khuyết có trong phần mềm
- Một số nguyên nhân điển hình:
  - Kỹ sư phần mềm và khách hiểu nhầm nhau
  - Lỗi tiềm ẩn của phần mềm do sơ ý của lập trình hoặc khi kiểm thử chưa bao quát hết
  - Vấn đề tính năng của phần mềm: không đáp ứng được yêu cầu về bộ nhớ, tệp, . . . Thiết kế sai, biên tập sai . . .
  - Thiếu chuẩn hóa trong phát triển phần mềm (trước đó)



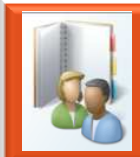
Bảo trì để tu sửa (tiếp)

- Kỹ nghệ ngược (Reverse Engineering):dò lại thiết kế để tu sửa
- Những lưu ý:
  - Mức trừu tượng
  - Tính đầy đủ
  - Tính tương tác
  - Tính định hướng



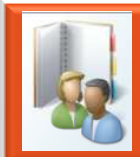
Bảo trì để thích hợp

- Là tu chỉnh phần mềm theo thay đổi của môi trường bên ngoài nhằm duy trì và quản lý phần mềm theo vòng đời của nó
- Thay đổi phần mềm thích nghi với môi trường: công nghệ phần cứng, môi trường phần mềm
- Những nguyên nhân chính:
  - Thay đổi về phần cứng (ngoại vi, máy chủ,. . .)
  - Thay đổi về phần mềm (môi trường): đổi OS
  - Thay đổi cấu trúc tệp hoặc mở rộng CSDL



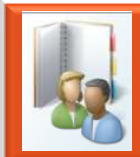
Bảo trì để cải tiến

- Là việc tu chỉnh hệ phần mềm theo các yêu cầu ngày càng hoàn thiện hơn, đầy đủ hơn, hợp lý hơn
- Những nguyên nhân chính:
  - Do muốn nâng cao hiệu suất nên thường hay cải tiến phương thức truy cập tệp
  - Mở rộng thêm chức năng mới cho hệ thống
  - Cải tiến quản lý kéo theo cải tiến tư liệu vận hành và trình tự công việc
  - Thay đổi người dùng hoặc thay đổi thao tác



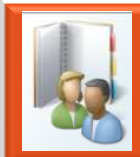
Bảo trì để cải tiến (tiếp)

- Còn gọi là tái kỹ nghệ (re-engineering)
- Mục đích: đưa ra một thiết kế cùng chức năng nhưng có chất lượng cao hơn
- Các bước thực hiện:
  - Xây dựng lưu đồ phần mềm
  - Suy dẫn ra biểu thức Bun cho từng dãy xử lý
  - Biên dịch bảng chân lí
  - Tái cấu trúc phần mềm



Bảo trì để phòng ngừa

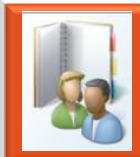
- Là công việc tu chỉnh chương trình có tính đến tương lai của phần mềm đó sẽ mở rộng và thay đổi như thế nào
- Thực ra trong khi thiết kế phần mềm đã phải tính đến tính mở rộng của nó, nên thực tế ít khi ta gặp bảo trì phòng ngừa nếu như phần mềm được thiết kế tốt



Bảo trì để phòng ngừa (tiếp)

Mục đích:

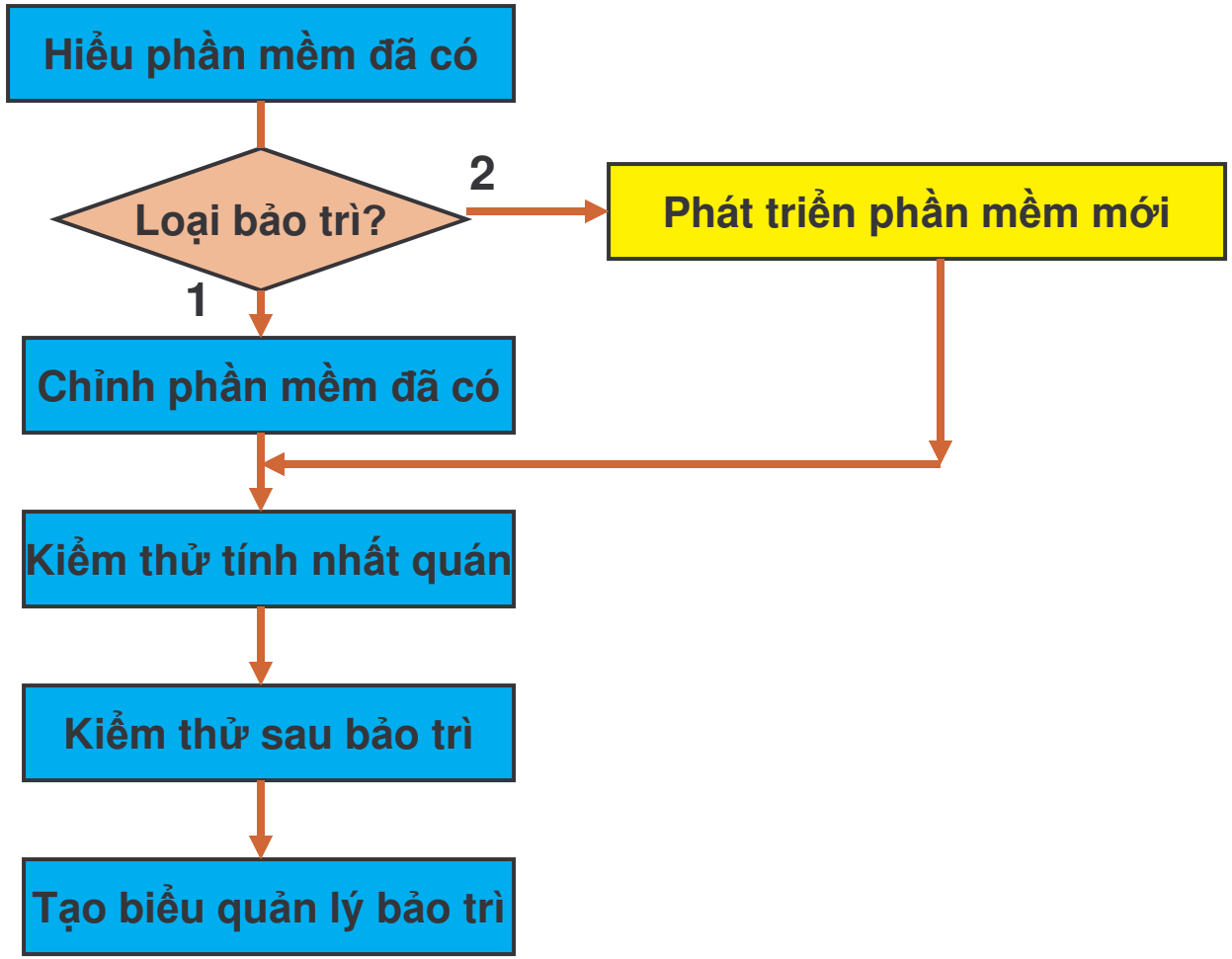
- sửa đổi để thích hợp với yêu cầu thay đổi sẽ có của người dùng
- Thực hiện những thay đổi trên thiết kế không tường minh
- Hiểu hoạt động bên trong chương trình
- Thiết kế / lập trình lại
- Sử dụng công cụ CASE

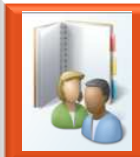


10.2 Trình tự nghiệp vụ bảo trì

- **Quy trình bảo trì là gì**? Đó là quá trình trong vòng đời của phần mềm, cũng tuân theo các pha phân tích, thiết kế, phát triển và kiểm thử từ khi phát sinh vấn đề cho đến khi giải quyết xong
- Thao tác bảo trì: Gồm 2 loại
  - Tu chỉnh cải đã có (loại 1)
  - Thêm cái mới (loại 2)

Sơ đồ bảo trì

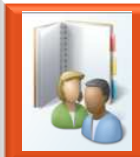

Hiểu phần mềm đã có
Loại bảo trì?
2
Phát triển phần mềm mới
1
Chỉnh phần mềm đã có
Kiểm thử tính nhất quán
Kiểm thử sau bảo trì
Tạo biểu quản lý bảo trì



Hiểu phần mềm đã có

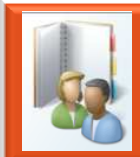
- Theo tài liệu nắm chắc các chức năng
- Theo tài liệu chi tiết hãy nắm vững đặc tả chi tiết, điều kiện kiểm thử, . . .
- Dò đọc chương trình nguồn, hiểu trình tự xử lý chi tiết của hệ thống

3 việc trên đều là công việc thực thi trên bàn



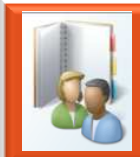
Tu sửa phần mềm đã có

- Bảo trì chương trình nguồn, tạo các môđun mới và dịch lại
- Thực hiện kiểm thử unit và tu chỉnh những mục liên quan có trong tư liệu đặc tả
- Chú ý theo sát tác động của môđun được sửa đến các thành phần khác trong hệ thống



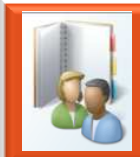
Phát triển phần mềm mới

- Khi thêm chức năng mới phải phát triển chương trình cho phù hợp với yêu cầu đã có
- Cần tiến hành từ thiết kế, lập trình, gỡ lỗi và kiểm thử unit với các chức năng mới được thêm vào
- Phản ảnh vào giao diện của phần mềm (thông báo, phiên bản, trợ giúp. . .) liên quan đến những thay đổi trên phần mềm hiện tại.



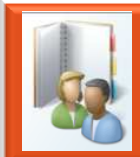
Kiểm chứng tính nhất quán bằng kiểm thử kết hợp

- Đưa đơn vị (unit) đã dược kiểm thử vào hoạt động trong hệ thống hiện tại.
- Điều chỉnh sự tương tích giữa các môđun
- Dùng các dữ liệu trước đây khi kiểm thử để kiểm thử lại tính nhất quán trong toàn bộ hệ thống
- Chú ý hiệu ứng **làn sóng** trong chỉnh sửa (hiệu chỉnh một đơn vị này nhưng lại tạo ra sự hiệu chỉnh trên nhiều đơn vị khác)



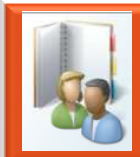
Kiểm tra khi hoàn thành bảo trì

- Kiểm tra nội dung mô tả có trong tư liệu đặc tả chưa?
- Cách ghi tư liệu có phù hợp với mô tả môi trường phần mềm mới hay không ?
- Những thay đổi đã được phản ánh đầy đủ trong hồ sơ phát triển hay chưa?



Lập biểu quản lý bảo trì

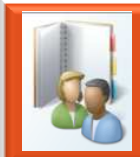
- Cần quản lý tình trạng bảo trì
- Lập biểu quản lý tình trạng bảo trì
  - Ngày tháng, giờ
  - Nguyên nhân
  - Tóm tắt cách khắc phục
  - Chi tiết khắc phục, hiệu ứng làn sóng
  - Người làm bảo trì
  - Số công



10.3 Những vấn đề lưu ý để bảo trì

Phương pháp cải tiến thao tác bảo trì:

- Sáng kiến trong quy trình phát triển phần mềm
- Sáng kiến trong quy trình bảo trì phần mềm
- Phát triển những kỹ thuật mới cho bảo trì

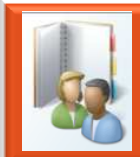


Sáng kiến trong quy trình phát triển phần mềm

(1) Chuẩn hóa mọi khâu trong phát triển phần mềm

(2) Người bảo trì chủ chốt tham gia vào giai đoạn phân tích và thiết kế để có thể hiểu được phần mềm và có thể dễ dàng xác định được nhưng sai xót khi kiểm tra và bảo trì phần mềm.

(3) Thiết kế để dễ bảo trì



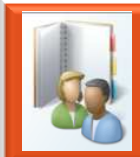
Sáng kiến trong quy trình bảo trì phần mềm

(1) Sử dụng các công cụ hỗ trợ phát triển phần mềm

(2) Chuẩn hóa thao tác bảo trì và thiết bị môi trường bảo trì

(3) Lưu lại những thông tin sử bảo trì

(4) Dự án nên cử một người chủ chốt của mình làm công việc bảo trì sau khi dự án kết thưc giai đoạn phát triển



Phát triển những kỹ thuật mới cho bảo trì

- Công cụ phần mềm hỗ trợ bảo trì
- Cơ sở dữ liệu cho bảo trì
- Quản lý tài liệu, quản lý dữ liệu, quản lý chương trình nguồn, quản lý dữ liệu thử, quản lý sử bảo trì
- Trạm bảo trì tính năng cao trong hệ thống mạng lưới bảo trì với máy chủ thông minh