Viện Điện tử - Viễn thông
Bộ Môn Điện tử - Kỹ thuật máy tính

Kỹ thuật phần mềm ứng dụng

Chương 1: Tổng quan môn học

Các nội dung chính

- Giới thiệu chung
- Các khái niệm cơ bản
- Các loại phần mềm
- Giới thiệu các mô hình tiến trình phổ biến

Giới thiệu chung

- **Kỹ thuật phần mềm** (hay kỹ nghệ phần mềm – software engineering) là một *chuyên ngành kỹ thuật* (engineering discipline) với trọng tâm nhằm phát triển các ***hệ thống phần mềm*** *chất lượng cao* một cách *hiệu quả*
- Phần mềm có đặc điểm là **trừu tượng** và **không chạm đến được** (intangible). Điều này làm cho phần mềm rất dễ trở nên phức tạp và khó hiểu

Giới thiệu chung

- Khái niệm "**Software Engineering**" xuất hiện lần đầu vào năm 1968 trong một cuộc họp bàn về một vấn đề được gọi là "**Cuộc khủng hoảng phần mềm**" (Software crisis)
- Chuyên ngành SE ra đời trong hoàn cảnh đó, với sứ mạng tìm ra các biện pháp giúp ngành công nghiệp phần mềm tránh được nguy cơ khủng hoảng. Và thực sự, nó đã hoàn thành sứ mạng này, và cái gọi là "cuộc khủng hoảng phần mềm" đã không thực sự xảy ra.

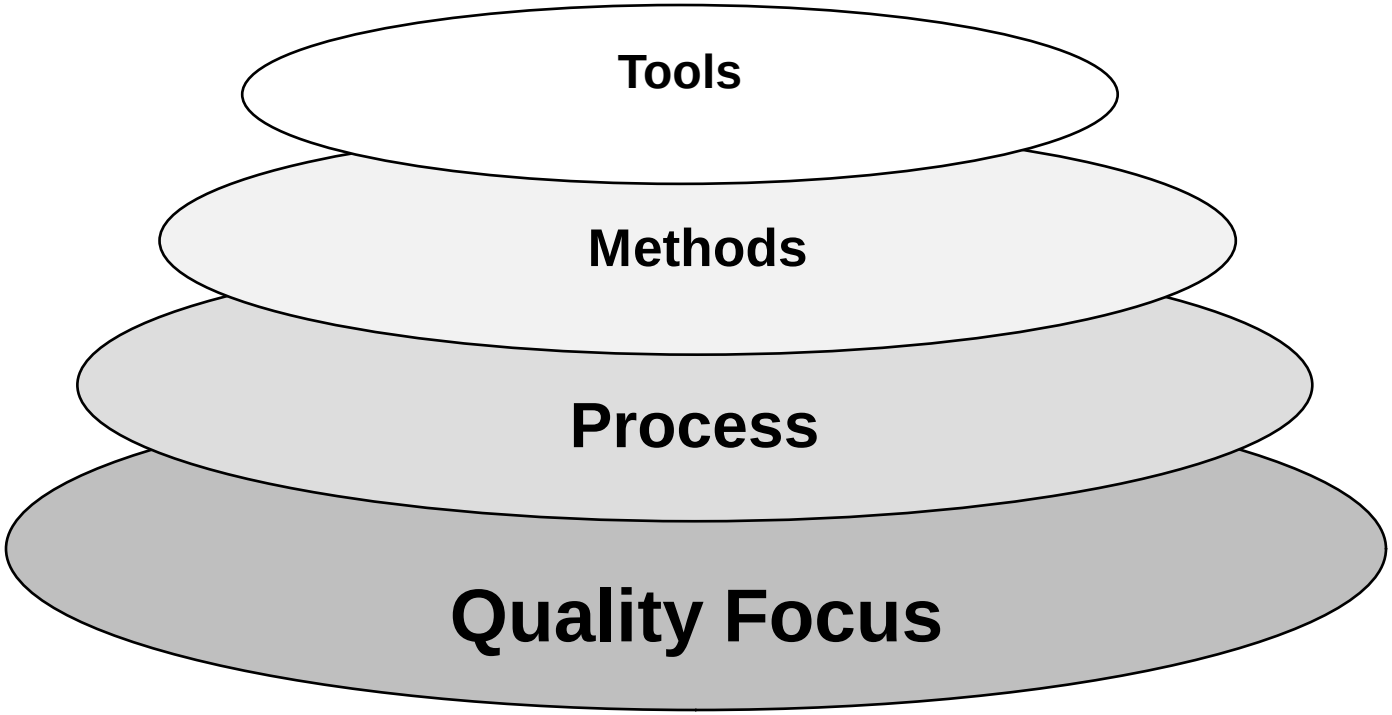
Các khái niệm cơ bản

- **Phần mềm** (sản phẩm phần mềm), bao gồm:
  - **Chương trình** (Program): là phần được thi hành trên máy tính
  - **Dữ liệu** (Data): gồm các cấu trúc dữ liệu, cơ sở dữ liệu lưu giữ các dữ liệu vào và ra của chương trình
  - **Tài liệu** (Documentation): tài liệu hệ thống, tài liệu người dùng

Các khái niệm cơ bản

- **Kỹ thuật phần mềm** (Software Engineering): Là một chuyên ngành kỹ thuật mà quan tâm đến tất cả các khía cạnh của việc sản xuất phần mềm, với mục tiên sản xuất ra các **sản phẩm phần mềm** *đa dạng*, *chất lượng cao*, một cách *hiệu quả nhất*.

Các tầng của SE



Các tầng của SE

- **Đảm bảo chất lượng** (quality focus) sản phẩm hay dịch vụ luôn là một nhiệm vụ sống còn của các công ty hay tổ chức. Do đó, mọi nền tảng công nghệ và kỹ thuật đều phải lấy việc đảm bảo chất lượng là mục tiêu hướng tới, và kỹ thuật phần mềm cũng không thể nằm ngoài mục tiêu này
- **Tầng Tiến trình** (process) có nhiệm vụ định nghĩa một khung các giai đoạn và các hoạt động cần thực hiện, cũng như các kết quả kèm theo chúng. Tầng này đóng vai trò nền tảng để kết nối các phương pháp, công cụ trong các bước thực hiện cụ thể, để có thể tạo ra các phần mềm có chất lượng và đúng thời hạn
- **Các phương pháp** (methods) kỹ thuật phần mềm cung cấp các chi tiết kỹ thuật là làm thế nào để xây dựng được phần mềm
- **Các công cụ** (tools) cung cấp các phương tiện hỗ trợ tự động hoặc bán tự động cho các giai đoạn hay các phương pháp. Các hệ thống phần mềm hỗ trợ trong công nghệ phần mềm được gọi là **CASE** (computer-aided software engineering)

Tiến trình phần mềm

- Là một dãy các giai đoạn và các hoạt động trong đó, cũng như các kết quả kèm theo. Kết quả cuối cùng chính là phần mềm cần phải xây dựng, đáp ứng được các yêu cầu của người dùng, và hoàn thành theo đúng kế hoạch về thời gian và ngân sách
- Có ba giai đoạn chính trong tiến trình phần mềm:
  - Giai đoạn định nghĩa (definition phase)
  - Giai đoạn phát triển (development phase)
  - Giai đoạn hỗ trợ (support phase)

Tiến trình phần mềm

- **Giai đoạn định nghĩa**: tập trung vào làm rõ ***Cái gì***, bao gồm:
  - Thông tin gì cần xử lý, bao gồm thông tin đầu vào và đầu ra.
  - Các chức năng gì cần thực hiện.
  - Hành vi nào của hệ thống sẽ được mong đợi.
  - Các tiêu chuẩn hợp lệ nào để đánh giá được sự đúng đắn và thành công của hệ thống.

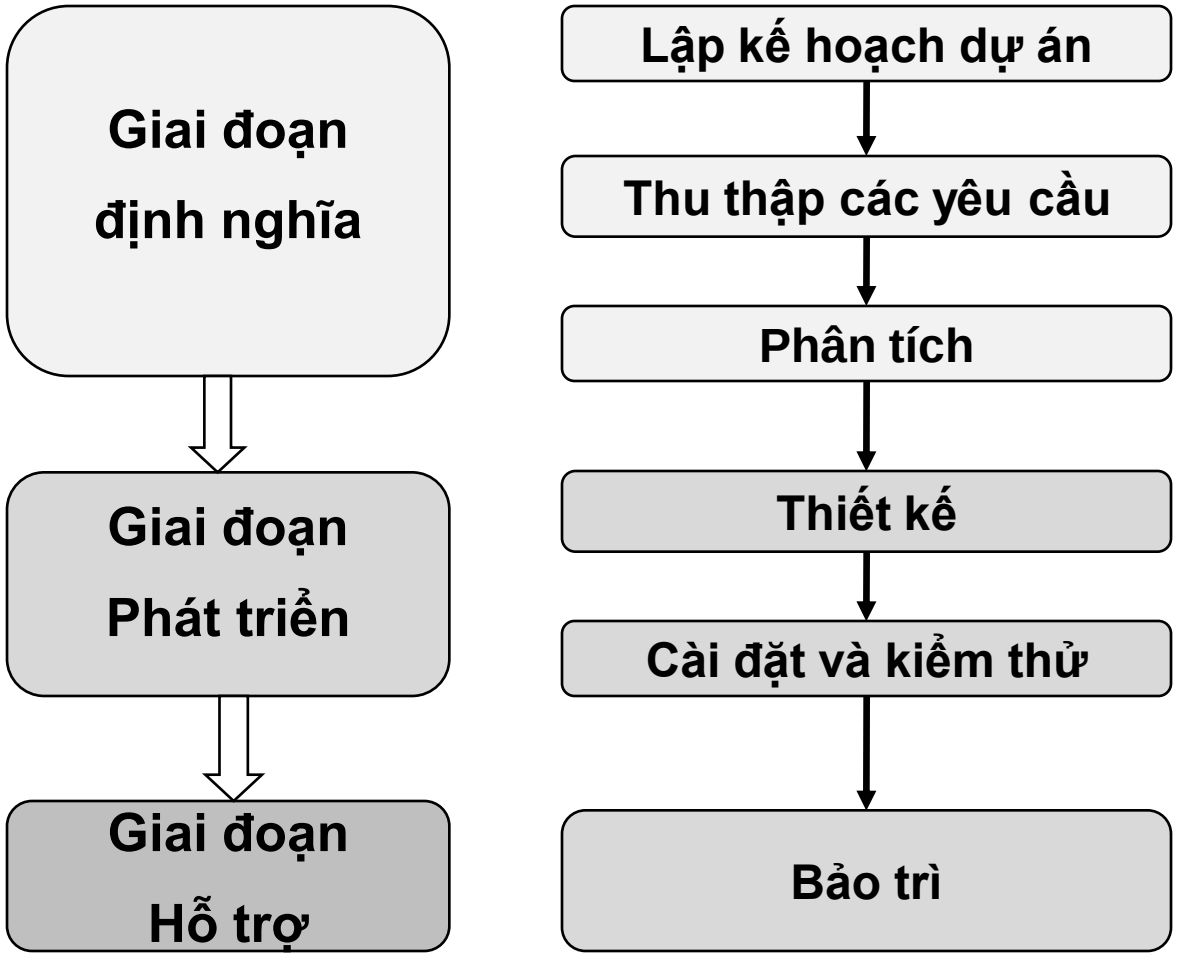
Tiến trình phần mềm

- **Giai đoạn phát triển**: tập trung vào ***Làm thế nào***, bao gồm:
  - Kiến trúc hệ thống (system architecture) được tổ chức thế nào.
  - Các chức năng được cài đặt và liên kết với nhau thế nào.
  - Tổ chức các cấu trúc dữ liệu, cơ sở dữ liệu thế nào.
  - Chuyển từ thiết kế sang cài đặt thế nào?
  - Việc kiểm thử sẽ được thực hiện thế nào?

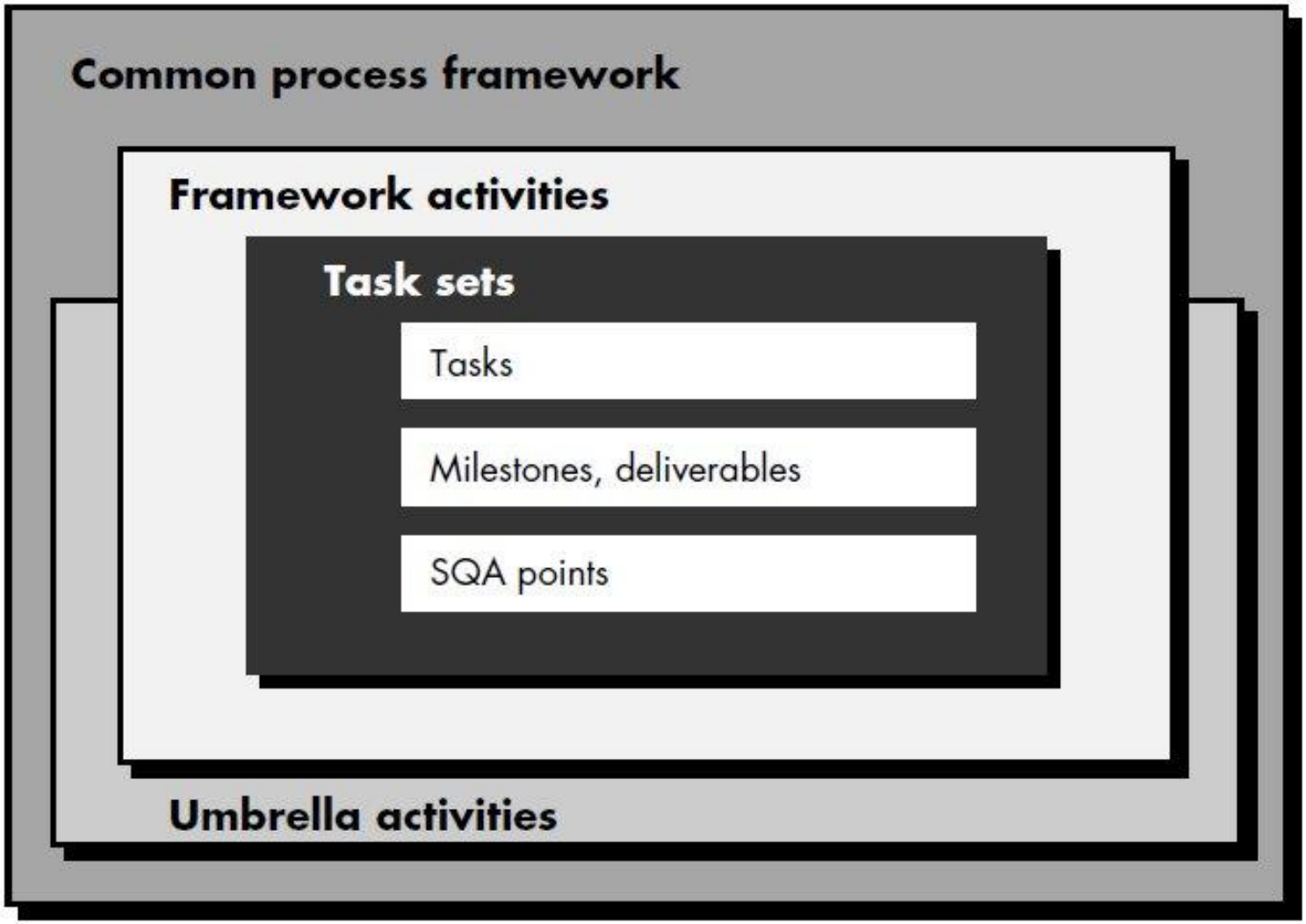
Tiến trình phần mềm

- **Giai đoạn hỗ trợ**: còn gọi là giai đoạn bảo trì, tập trung vào việc ứng phó với các thay đổi của hệ thống phần mềm, bao gồm:
  - Sửa lỗi (Correction)
  - Làm thích ứng (Adaptation)
  - Nâng cấp (Upgrade)
  - Phòng ngừa (Prevention), còn gọi là tái kỹ thuật phần mềm (software reengineering)

Tiến trình phần mềm



Tiến trình phần mềm



Tiến trình phần mềm

- Khung tiến trình chung (common process framework): là mô hình chung cho các dự án phần mềm khác nhau trong một tổ chức. Nó bao gồm:
  - Các công việc trong khung (Framework activities) gồm:
    - Các nhiệm vụ cụ thể (tasks)
    - Các mốc thời gian (milestones)
    - Các kết quả bàn giao (deliverables)
    - Các điểm kiểm tra chất lượng hệ thống (SQA points)
  - Các công việc bao trùm (Umbrella activities) gồm:
    - Quản lý chất lượng phần mềm
    - QL cấu hình phần mềm

Mô hình tiến trình phần mềm

- **Mô hình tiến trình** (process model) Là một chiến lược phát triển phần mềm , bao gồm các cách thức kết hợp, sử dụng tiến trình phần mềm, cách vận dụng các phương pháp và các công cụ trong mỗi giai đoạn phát triển.
- Mô hình tiến trình cũng còn được gọi là **mẫu tiến trình** (process paradigm), hay **mô hình phát triển phần mềm**.

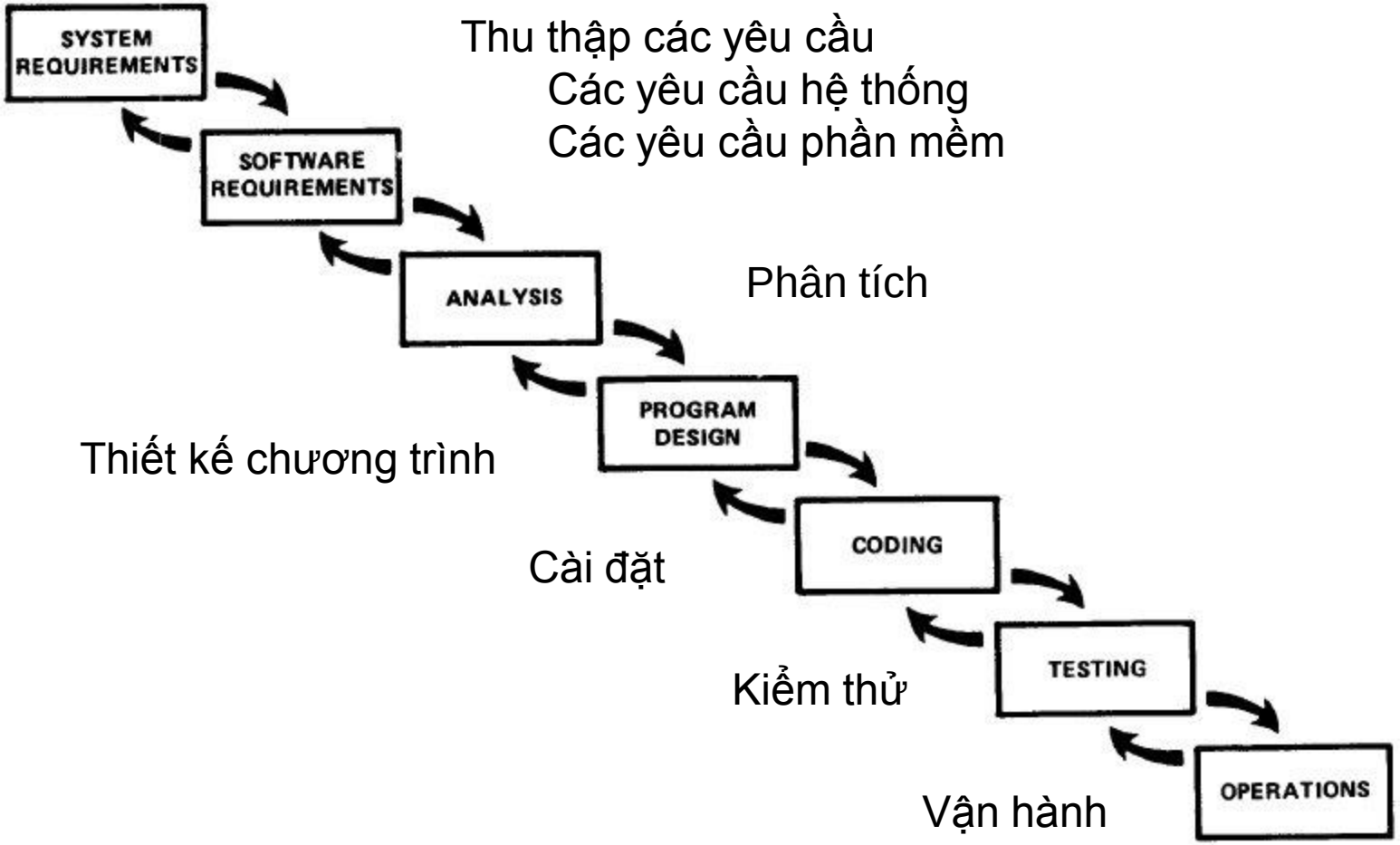
Các loại phần mềm

- **Phần mềm hệ thống** (system software)
- **Phần mềm thời gian thực** (real time sw)
- **Phần mềm quản lý** (business sw): cũng được gọi là ***hệ thông tin quản lý*** (management information system – MIS)
- **Phần mềm khoa học và công nghệ** (engineering and scientific sw)
- **Phần mềm nhúng** (embedded sw)
- **Phần mềm văn phòng** (office sw)
- **Phần mềm Web** (Web-based sw)
- **Phần mềm trí tuệ nhân tạo** (artificial intelligence sw)
- V.v.

Các mô hình tiến trình

- Mô hình tuyến tính cổ điển (mô hình thác nước – Waterfall model)
- Mô hình bản mẫu (Prototyping model)
- Mô hình RAD (Rapid Application Development model)
- Mô hình tăng trưởng (Incremental model)
- Mô hình xoáy ốc (Spiral model)

Mô hình tuyến tính cổ điển*



Thu thập các yêu cầu
Các yêu cầu hệ thống
Các yêu cầu phần mềm

Phân tích

Thiết kế chương trình

Cài đặt

Kiểm thử

Vận hành

Mô hình tuyến tính cổ điển

- Mô hình này có một số đặc điểm như sau:
  - Các bước được tiến hành tuần tự, kết thúc bước trước thì mới thực hiện đến bước sau
  - Thời gian thực hiện mỗi bước thường kéo dài do phải làm thật hoàn chỉnh
  - Thường chỉ tiếp xúc với người dùng vào giai đoạn đầu và giai đoạn cuối. Người dùng thường không tham gia vào các bước ở giữa, như từ thiết kế, cài đặt và đến tích hợp

Mô hình tuyến tính cổ điển

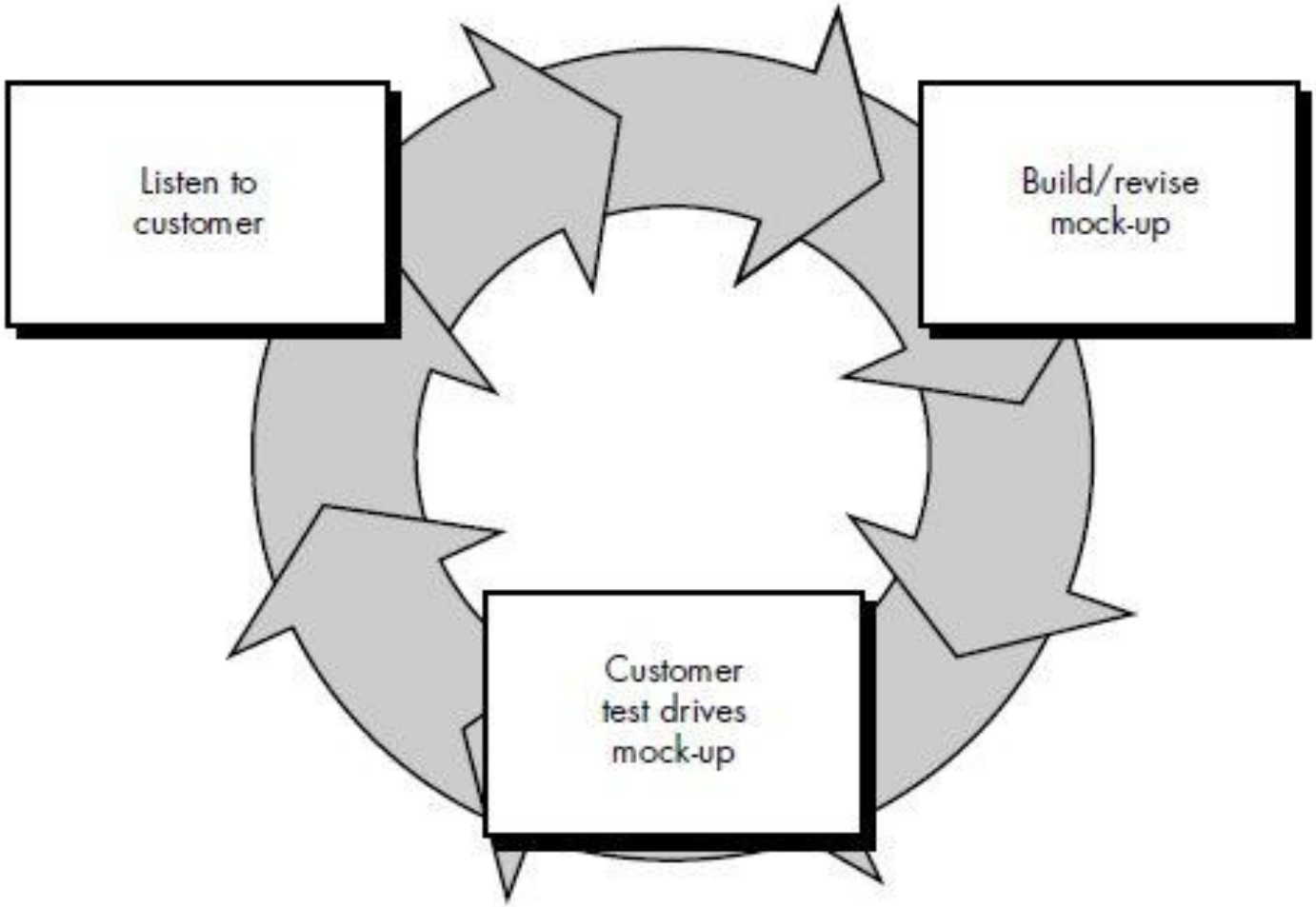
- **Ưu điểm**:
  - Đơn giản và rõ ràng
  - Đóng vai trò như một mẫu tham chiếu cho các mô hình khác
  - Vẫn còn được sử dụng rộng rãi cho đến nay
- **Nhược điểm**:
  - Không dễ dàng cho việc thu thập đầy đủ và tường minh tất cả các yêu cầu hệ thống ngay từ ban đầu
  - Người dùng phải chờ đến cuối cùng mới có được hệ thống để dùng, nên thời gian chờ đợi là khá lâu, có khi đến hàng năm. Khi đó có thể có các yêu cầu mới đã phát sinh, dễ dẫn khả năng hệ thống không còn đáp ứng được kỳ vọng của người dùng.
  - Dễ dẫn đến tình trạng "**blocking states**", tức là khi có một nhóm bị chậm tiến độ, thì các nhóm khác phải chờ, và thời gian chờ đợi thậm chí vượt quá thời gian làm việc.

Mô hình bản mẫu

- Thông thường trong thực tế, các yêu cầu của hệ thống khó có thể xác định rõ ràng và chi tiết ngay trong gia đoạn đầu của dự án phần mềm vì:
  - **Người dùng** cũng chỉ đưa ra các mục tiêu tổng quát của phần mềm, chứ cũng chưa định rõ được một cách chi tiết các chức năng cụ thể, hay các thông tin chi tiết đầu vào, đầu ra như thế nào.
  - **Nhà phát triển** cũng chưa xác định rõ ràng ngay các yêu cầu, cũng như chắc chắn về chất lượng phần mềm, cũng như khả năng thỏa mãn của khách hàng

  → **mô hình bản mẫu**

Mô hình bản mẫu



Mô hình bản mẫu

Gồm các giai đoạn:

- **Thu thập các yêu cầu** (requirements gathering): khách hàng và nhà phát triển sẽ gặp nhau để xác định ra các mục tiêu tổng thể của phần mềm. Sau đó họ sẽ định ra phần nào đã rõ, phần nào cần phải định nghĩa thêm.
- **Thiết kế nhanh (quick design):** thiết kế này tập trung vào những phần mà khách hàng có thể nhìn thấy được (giao diện, các dữ liệu vào, ra). Sau đó, từ thiết kế này, một bản mẫu sẽ được xây dựng.
- **Kiểm tra và đánh giá bản mẫu:** Bản mẫu này sẽ được dùng để cho phép người dùng đánh giá, nhằm làm rõ hơn các yêu cầu của họ. Đồng thời, thông qua bản mẫu, người phát triển hệ thống cũng hình dung cụ thể hơn về những yêu cầu của khách hàng, cũng như khả năng cài đặt và hiệu quả hoạt động của hệ thống.

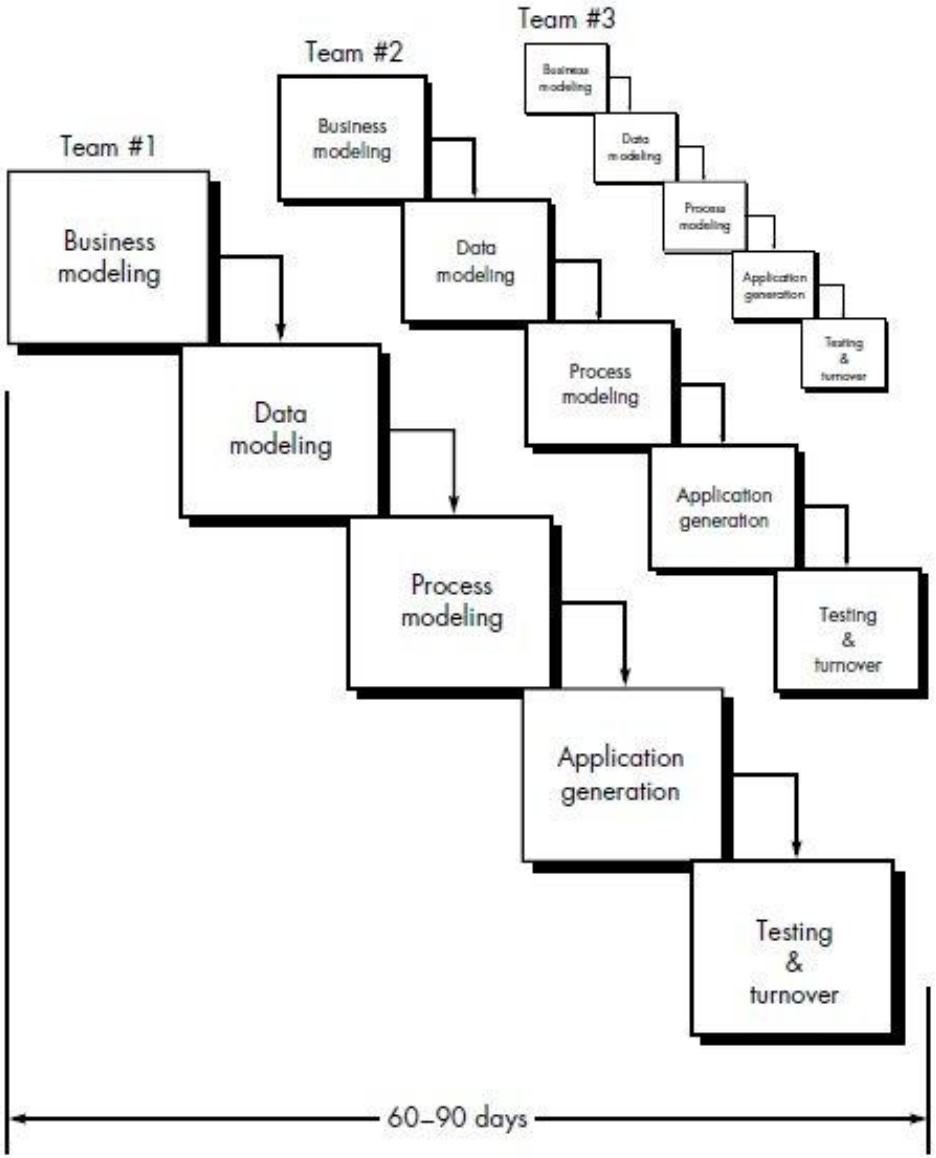
Mô hình bản mẫu

- **Ưu điểm**:
  - Cho phép người dùng xác định yêu cầu của mình rõ ràng và cụ thể hơn, đồng thời nhà phát triển cũng năm được chính xác hơn các yêu cầu đó.
  - Cả người dùng và nhà phát triển thường đều thích mô hình này, do người dùng luôn cảm nhận được hệ thống thực sẽ như thế nào, và nhà phát triển cũng luôn có cái để xây dựng và dần hoàn thiện.
- **Nhược điểm**:
  - Để có được bản mẫu nhanh, việc thiết kế cũng được làm nhanh, nên thường được làm không cẩn thận. Điều này dễ dẫn đến các thiết kế có tính chắp vá, không có cái nhìn tổng thể và dài hạn.
  - Việc làm bản mẫu nhanh cũng thường kéo theo việc lựa chọn các công cụ cài đặt vội vàng, không cẩn thận, (như ngôn ngữ lập trình, hệ quản trị cơ sở dữ liệu,v.v). Điều này sẽ ảnh hưởng đến các giai đoạn phát triển sau khi quy mô và yêu cầu của hệ thống ngày càng lớn lên

Mô hình RAD

- Là mô hình tiến trình phát triển phần mềm tăng trưởng, nhưng nhấn mạnh vào chu trình phát triển phần mềm có thời gian rất ngắn. Mô hình này gồm các giai đoạn:
  - **Mô hình hóa nghiệp vụ** (Business modeling): mô hình hóa các luồng thông tin nghiệp vụ giữa các chức năng nghiệp vụ
  - **Mô hình hóa dữ liệu** (Data modeling): từ các thông tin nghiệp vụ, các thực thể dữ liệu, các thuộc tính của chúng, và các liên kết giữa các thực thể này sẽ được xác định và được mô hình hóa.
  - **Mô hình hóa xử lý** (Process modeling): Mô tả các chức năng xử lý trên các đối tượng dữ liệu đã được xác định ở giai đoạn trên.
  - **Sản sinh ứng dụng** (Application generation): RAD sử dụng các kỹ thuật công nghệ phần mềm thế hệ thứ 4, cho phép dễ dàng sản sinh mã chương trình từ các đặc tả và thiết kế trừu tượng. Các kỹ thuật này cũng cho phép tái sử dụng các thành phần chương trình có sẵn (kết hợp mô hình Component-based development).
  - **Kiểm thử và bàn giao** (Testing and turnover): phần ứng dụng đã xây dựng sẽ được kiểm tra và bàn giao cho bên tích hợp hệ thống.

Mô hình RAD


Team #1
Business modeling
Data modeling
Process modeling
Application generation
Testing & turnover
Team #2
Business modeling
Data modeling
Process modeling
Application generation
Testing & turnover
Team #3
Business modeling
Data modeling
Process modeling
Application generation
Testing & turnover
60–90 days

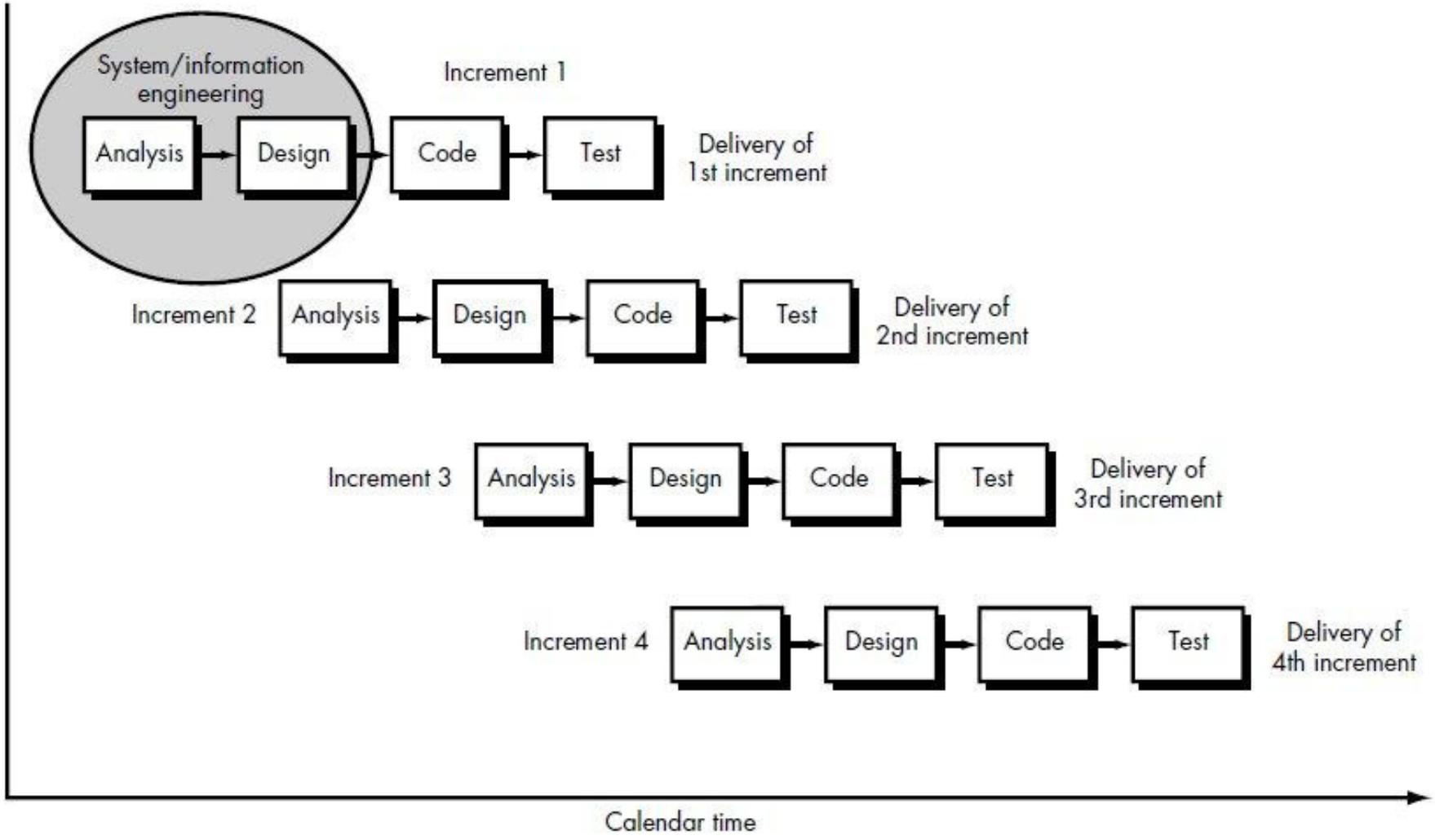
Mô hình RAD

- **Ưu điểm**:
  - Tận dụng các công nghệ mới trong phát triển hệ thống, cho phép hoàn thành hệ thống trong thời gian ngắn hơn đáng kể.
  - Khuyến khích việc tái sử dụng các thành phần của chương trình
- **Nhược điểm:**
  - Không phù hợp với các phần mềm mà không có sự phân chia modul rõ ràng,
  - Đòi hỏi tài nguyên và chi phí phát triển cao như số lượng nhân lực nhiều, công cụ CASE thế hệ 4 đắt tiền

Mô hình tăng trưởng

- Là sự kết hợp của mô hình tuyến tính và triết lý lặp lại của mô hình bản mẫu
- Phần mềm được chia thành các ***phần tăng trưởng*** (increment), trong đó mỗi phần là một sản phẩm hoàn chỉnh (đã chạy được và có thể bàn giao cho người dùng). Đồng thời *phần tăng trưởng* sau sẽ bổ sung thêm tính năng còn thiếu trong những phần trước

Mô hình tăng trưởng



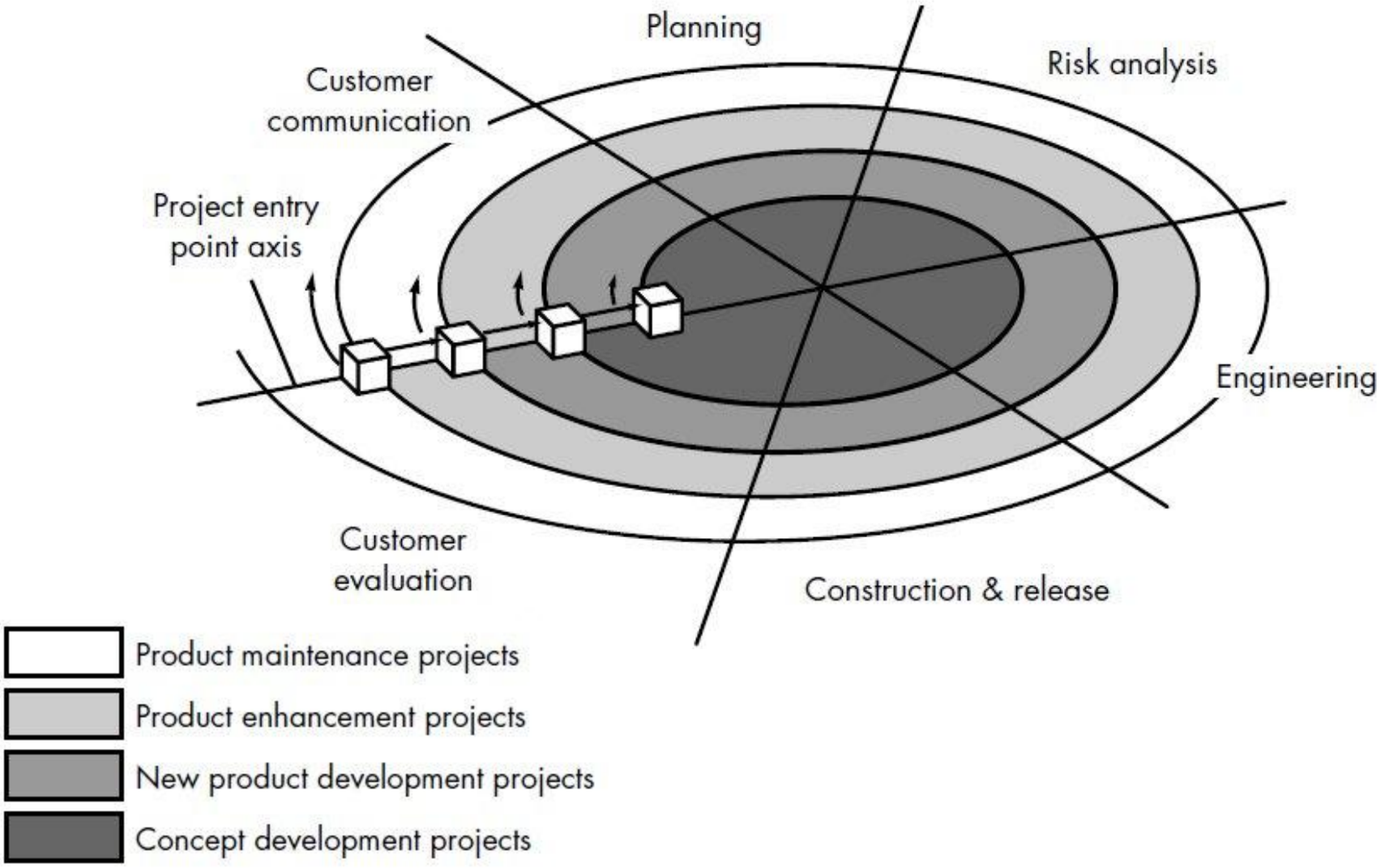
Mô hình tăng trưởng

- **Ưu điểm**
  - Kết hợp được các ưu điểm của các mô hình tuyến tính và làm bản mẫu
  - Rất phù hợp khi số lượng nhân viên hạn chế, và người dùng có đòi hỏi phải sớm có hệ thống thử nghiệm
- **Nhược điểm**
  - Việc gấp gáp đưa ra các thành phần tăng trưởng cũng có thể gây ra sự manh mún trong phân tích và thiết kế
  - Khó khăn trong việc đảm bảo tính tương thích (compatibility) giữa các thành phần tăng trưởng.

Mô hình xoáy ốc

- Cũng là một mô hình tiến hóa kết hợp đặc tính lặp lại của mô hình bản mẫu và tính hệ thống của mô hình thác nước cổ điển
- Mô hình này cũng cho phép tạo ra một dãy các phiên bản tăng trưởng (incremental release). Tuy nhiên khác với mô hình tăng trưởng, các phiên bản đầu tiên của mô hình xoáy ốc thường chỉ là các mô hình trên giấy hoặc bản mẫu (prototype). Đến các phiên bản sau thì mới là các bản chạy được và càng ngày càng hoàn chỉnh.

Mô hình xoáy ốc



Mô hình xoáy ốc

- Mô hình này phân chia thành các giai đoạn, được gọi là các ***vùng nhiệm vụ*** (task regions).
- Số lượng vùng nhiệm vụ có thể thay đổi, và thường có từ 3 cho đến 6 vùng.
- Mỗi vùng lại bao gồm ***một tập các nhiệm vụ*** (set of tasks), và số lượng cũng thay đổi tùy theo tính chất của dự án.

Mô hình xoáy ốc

- **Ưu điểm**:
  - Linh hoạt, dễ thích ứng với các loại phần mềm và các nhu cầu sử dụng khác nhau, nhất là các phần mềm quy mô lớn
  - Có khá đầy đủ các bước trong tiến trình phát triển, nhất là việc chú trọng phân tích tính rủi ro (risk) của phần mềm cả về mặt kỹ thuật và quản lý
- **Hạn chế**:
  - Phức tạp, cần khá nhiều thời gian để hiểu và vận dụng được một cách hiệu quả
  - Khó khăn trong việc quản lý nhiều chu trình phát triển

Tóm tắt

- Các khái niệm cơ bản
- Các loại phần mềm
- Các mô hình tiến trình phổ biến

Thank you!

Viện Điện tử - Viễn thông
Bộ Môn Điện tử - Kỹ thuật máy tính

Kỹ thuật phần mềm ứng dụng

Chương 2: Quản trị dự án phần mềm

Các nội dung chính

- Các khía cạnh cần quản lý gồm 4 P:
  - Con người (People)
  - Sản phẩm phần mềm (Product)
  - Tiến trình phần mềm (Process)
  - Dự án (Project)

Con người

- Những người tham gia trong một dự án:
  - **Nhà quản trị cao cấp** (senior managers): là người xác định vấn đề nghiệp vụ và có ảnh hưởng rất quan trọng đến dự án.
  - **Nhà quản trị (về kỹ thuật) dự án** (project managers): là người lên kế hoạch, tổ chức, khuyến khích và kiểm tra công việc của những nhân viên khác trong dự án.
  - **Nhân viên kỹ thuật** (practitioners): là những người có những kiến thức kỹ thuật cần thiết để tạo ra phần mềm
  - **Khách hàng** (customers): là người xác định các yêu cầu cho phần mềm và những cổ đông (stakeholders) có lợi ích liên quan
  - **Những người dùng cuối** (end-users)
- Tổ chức về nhân sự trong một dự án:
  - Thường tổ chức thành một hoặc nhiều team (nhóm), mỗi nhóm có 1 team leader (trưởng nhóm).

Team – Vấn đề tổ chức

- Các cách tổ chức team:
  - **Dân chủ phi tập trung** (Democratic decentralized – DD):
    - Không có team leader thường trực
    - Quyết định dựa trên sự thống nhất của nhóm
    - Sự trao đổi diễn ra theo chiều ngang (horizontal communication)
  - **Phi tập trung có kiểm soát** (Controlled decentralized - CD):
    - Có team leader thường trực
    - Quyết định cũng hoạt động theo nhóm
    - Sự trao đổi diễn ra theo cả hai chiều ngang và dọc
  - **Tập trung có kiểm soát** (Controlled centralized – CC)
    - Có leader thường trực
    - Ra quyết định và điều phối là trách nhiệm của leader
    - Sự trao đổi giữa leader và các thành viên khác là theo chiều dọc

Team – Vấn đề tổ chức

- Các tiêu chí quyết định cách tổ chức team:
  - Mức độ khó của vấn đề cần giải quyết
  - Kích thước của chương trình (size of the resultant program)
  - Thời gian tồn tại của nhóm
  - Mức độ modul hóa của vấn đề
  - Yêu cầu về chất lượng và độ tin cậy của hệ thống
  - V.v.

Team

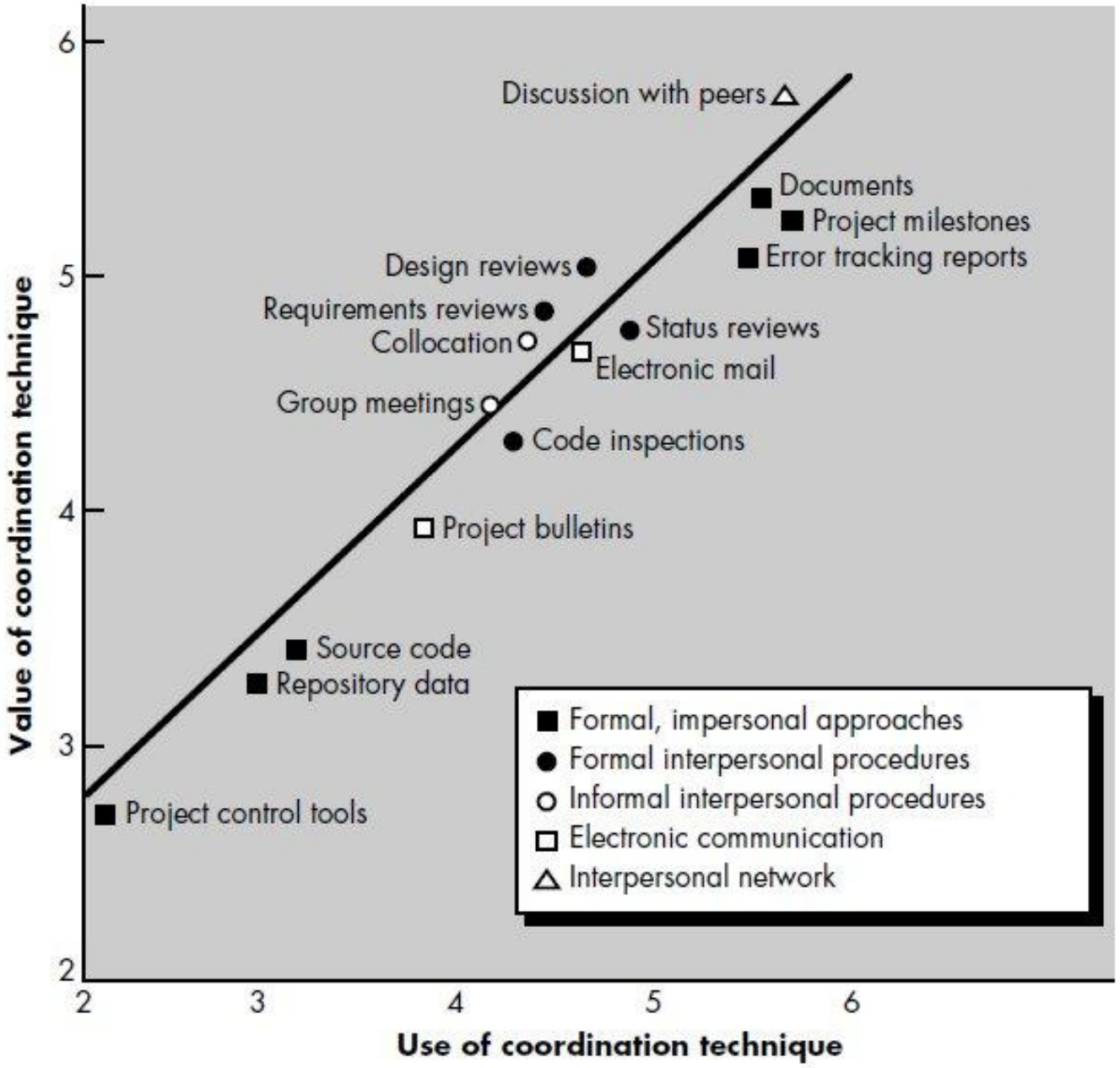
- Mục đích của việc tổ chức và quản lý team là để tạo ra một team có năng suất làm việc cao (high-performance) và gắn kết (cohesiveness) → tạo ra một ***Team gắn bó* (*Jelled Team*)**:
  - Cần làm:
    - Có tổ chức phù hợp
    - Tin tưởng lẫn nhau
    - Các thôn g tin luôn rõ ràng và cởi mở
    - Phân chia công việc phù hợp
  - Cần tránh:
    - Chủ nghĩa làm việc đơn độc
    - Thiếu tinh thần trách nhiệm
    - Thái độ bất mãn

Team - Vấn đề điều phối và trao đổi

- Điều phối (coordination) và trao đổi (communication) là rất cần thiết và quan trọng do một số nhân tố:
  - Mức độ của các nỗ lực phát triển (scale of development efforts) là rất lớn
  - Sự không chắc chắn thường xuyên xảy ra
  - Tính tương thông (interoperability) là đặc tính quan trọng của đa số các hệ thống

Team - Vấn đề điều phối và trao đổi

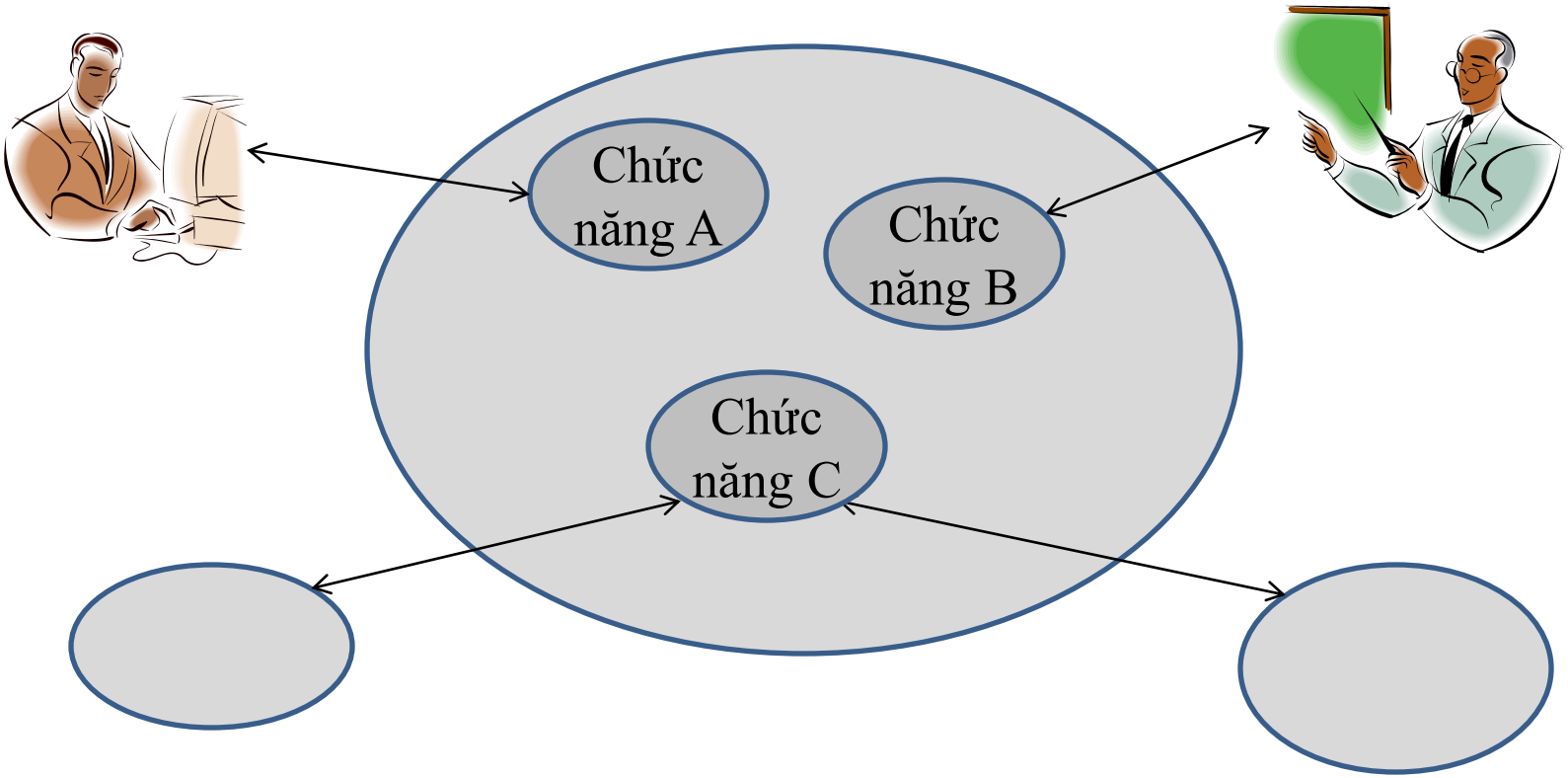
- Các kỹ thuật điều phối và trao đổi:
  - Hình thức và không hình thức (formal & informal)
  - Không liên quan và có liên quan đến cá nhân (impersonal & interpersonal)
  - Trao đổi điện tử
  - Mạng xã hội


Value of coordination technique
Use of coordination technique
6
5
4
3
2
2
3
4
5
6
Discussion with peers
Documents
Project milestones
Error tracking reports
Design reviews
Requirements reviews
Collocation
Status reviews
Electronic mail
Group meetings
Code inspections
Project bulletins
Source code
Repository data
Project control tools
Formal, impersonal approaches
Formal interpersonal procedures
Informal interpersonal procedures
Electronic communication
Interpersonal network

Sản phẩm

- Xác định phạm vi của sản phẩm phần mềm (software scope), liên quan đến các mặt:
  - **Khung cảnh** (context): là cái nhìn tổng quan về đường ranh giới giữa sản phẩm cần phát triển và các thành phần khác bên ngoài.
  - **Các mục đích về thông tin** (information objectives): là các yêu cầu cụ thể hơn về thông tin đầu vào và ra của người dùng
  - **Các chức năng và hiệu năng** (functions and performance): là các yêu cầu về các chức năng cần thực hiện, các ràng buộc về hiệu năng của chúng nếu có.

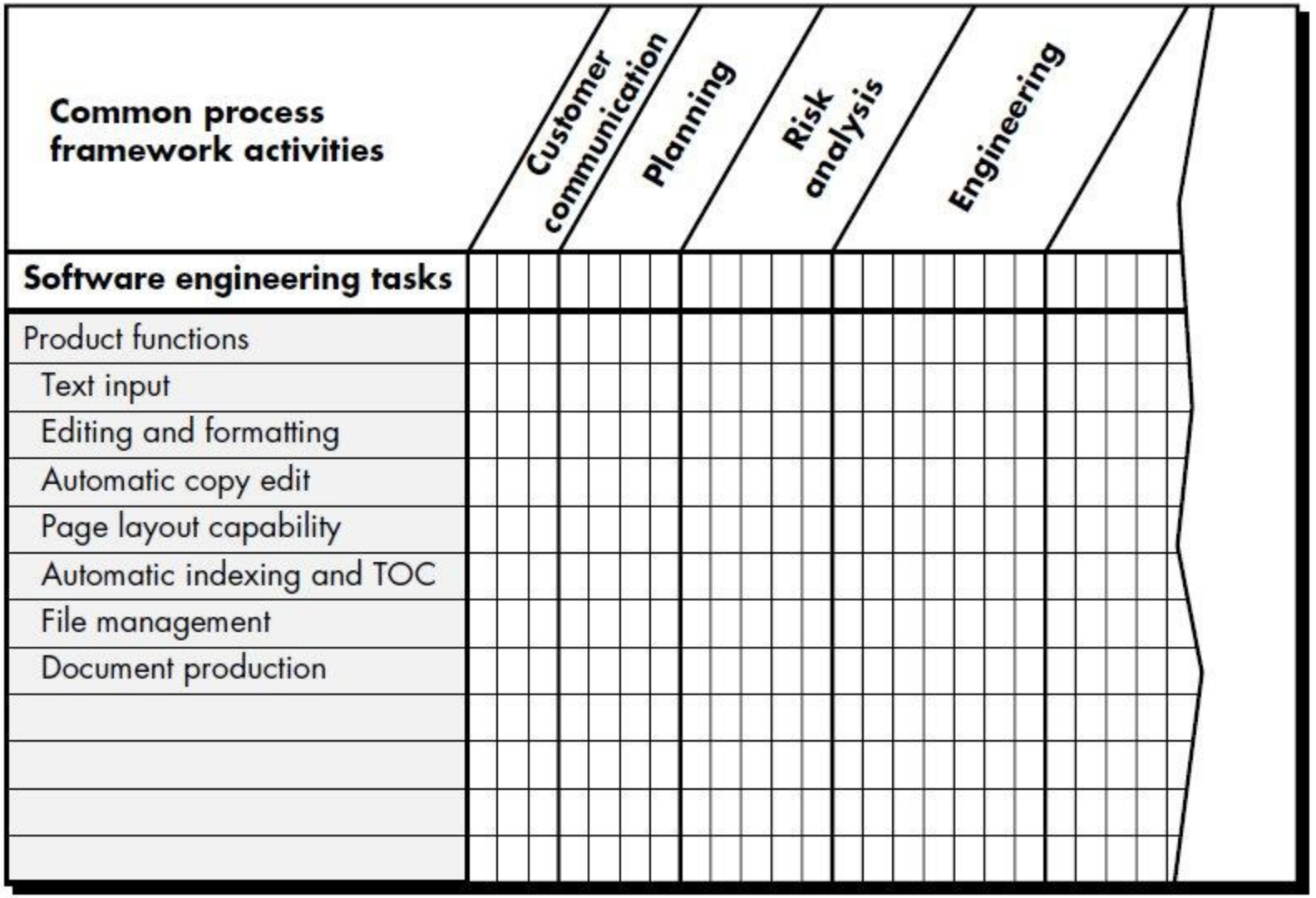
Sản phẩm – Xác định phạm vi



Tiến trình phần mềm - Chọn tiến trình phù hợp

- Chọn mô hình tiến trình phù hợp với phần mềm cần phát triển
  - Mô hình tuyến tính cổ điển
  - Mô hình bản mẫu
  - Mô hình RAD
  - Mô hình tăng trưởng
  - Mô hình xoáy ốc

Tiến trình phần mềm - Kết hợp tiến trình và phần mềm



Tiến trình phần mềm – Chia tiến trình thành các nhiệm vụ phù hợp

- Các bước trong các mô hình tiến trình thường tương đối khái quát và trừu tượng, nên chúng cần phải được xác định chi tiết hơn cho phù hợp với từng dự án cụ thể.
- Điều này dẫn đến việc, từng công ty, tùy theo loại hình phần mềm và dự án của mình, họ thường đưa ra các mô hình tiến trình đầy đủ và chi tiết phù hợp với hoàn cảnh thực tế của họ, và nó sẽ trở thành ***Phương pháp luận phát triển phần mềm*** của công ty đó.

Dự án (Project)

- Để quản lý thành công các dự án phần mềm, các nhà quản lý cần nắm được các vấn đề giúp dự án thành công, cũng như các vấn đề có thể dẫn đến thất bại.
- Có một số dấu hiệu giúp phát hiện việc quản lý dự án đang có vấn đề nguy hại:
  - Không hiểu rõ các yêu cầu của khách hàng
  - Phạm vi của hệ thống xác định không đầy đủ
  - Không có hay bất hợp lý trong việc ứng phó với các thay đổi
  - Thời gian thực hiện theo kế hoạch không hiện thực
  - Thiếu nhân viên có các kinh nghiệm phù hợp
  - V.v.

Tóm tắt

- Có 4 lĩnh vực mà nhà quản trị dự án cần quan tâm:
  - Con người
  - Sản phẩm phần mềm
  - Tiến trình phát triển
  - Bản thân dự án

Thank you!

Viện Điện tử - Viễn thông
Bộ Môn Điện tử - Kỹ thuật máy tính

Kỹ thuật phần mềm ứng dụng

Chương 3: Kỹ thuật hệ thống (System Engineering)

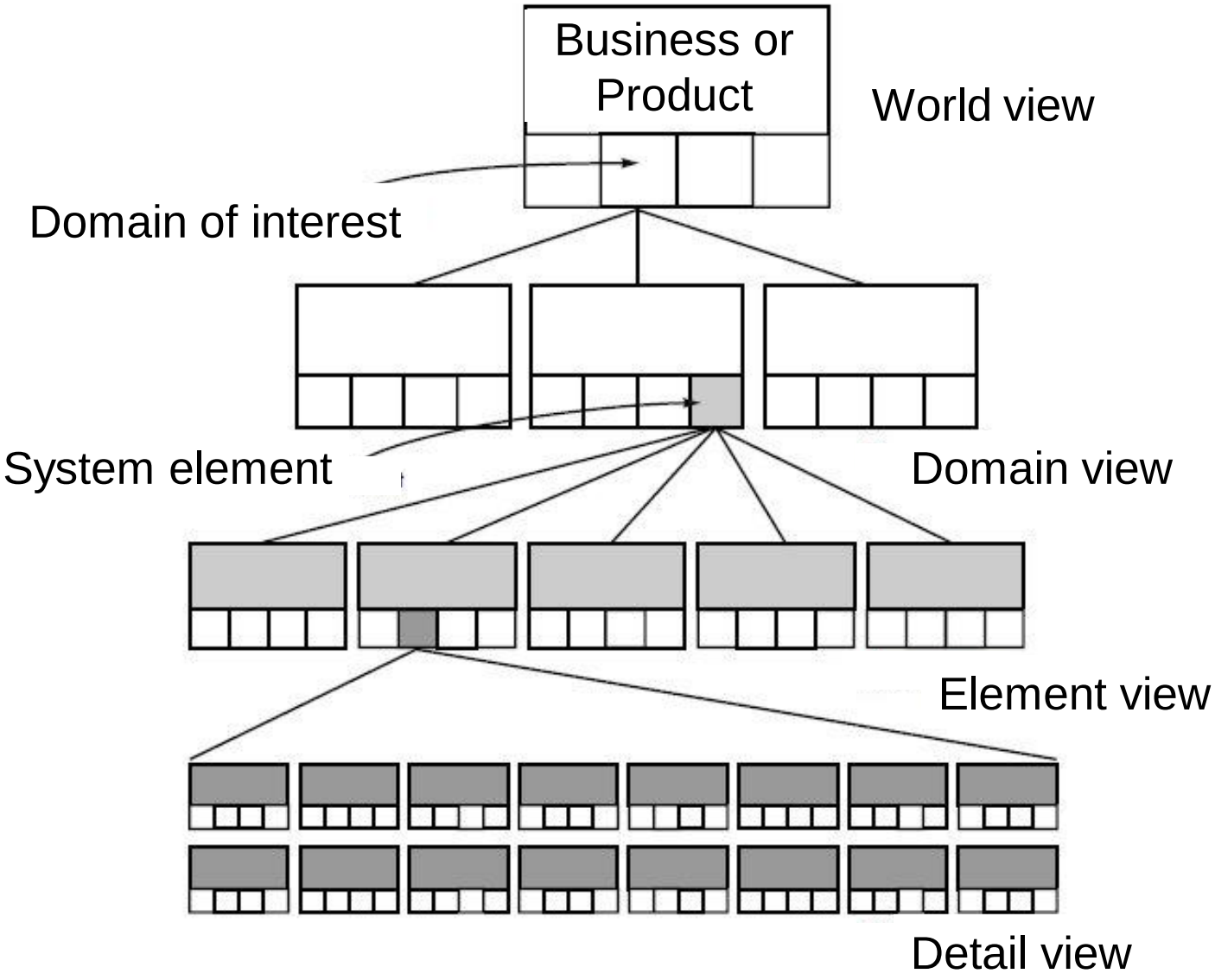
Các nội dung chính

- Các khái niệm cơ bản
- Sự phân cấp của kỹ thuật hệ thống
- Kỹ thuật tiến trình nghiệp vụ
- Kỹ thuật sản phẩm phần mềm
- Kỹ thuật thu thập và xử lý yêu cầu (requirements engineering)

Các khái niệm cơ bản

- Hệ thống máy tính (computer-based system):
  - **Định nghĩa**: Là một tập hợp hay bố trí các phần tử mà được tổ chức sao cho hoàn thành một mục tiêu xác định nào đó qua việc xử lý thông tin [Pressman, p246]
  - Các thành phần của hệ thống máy tính:
    - Phần mềm
    - Phần cứng
    - Con người
    - Cơ sở dữ liệu
    - Tài liệu
    - Thủ tục

Kỹ thuật hệ thống – Tính phân cấp

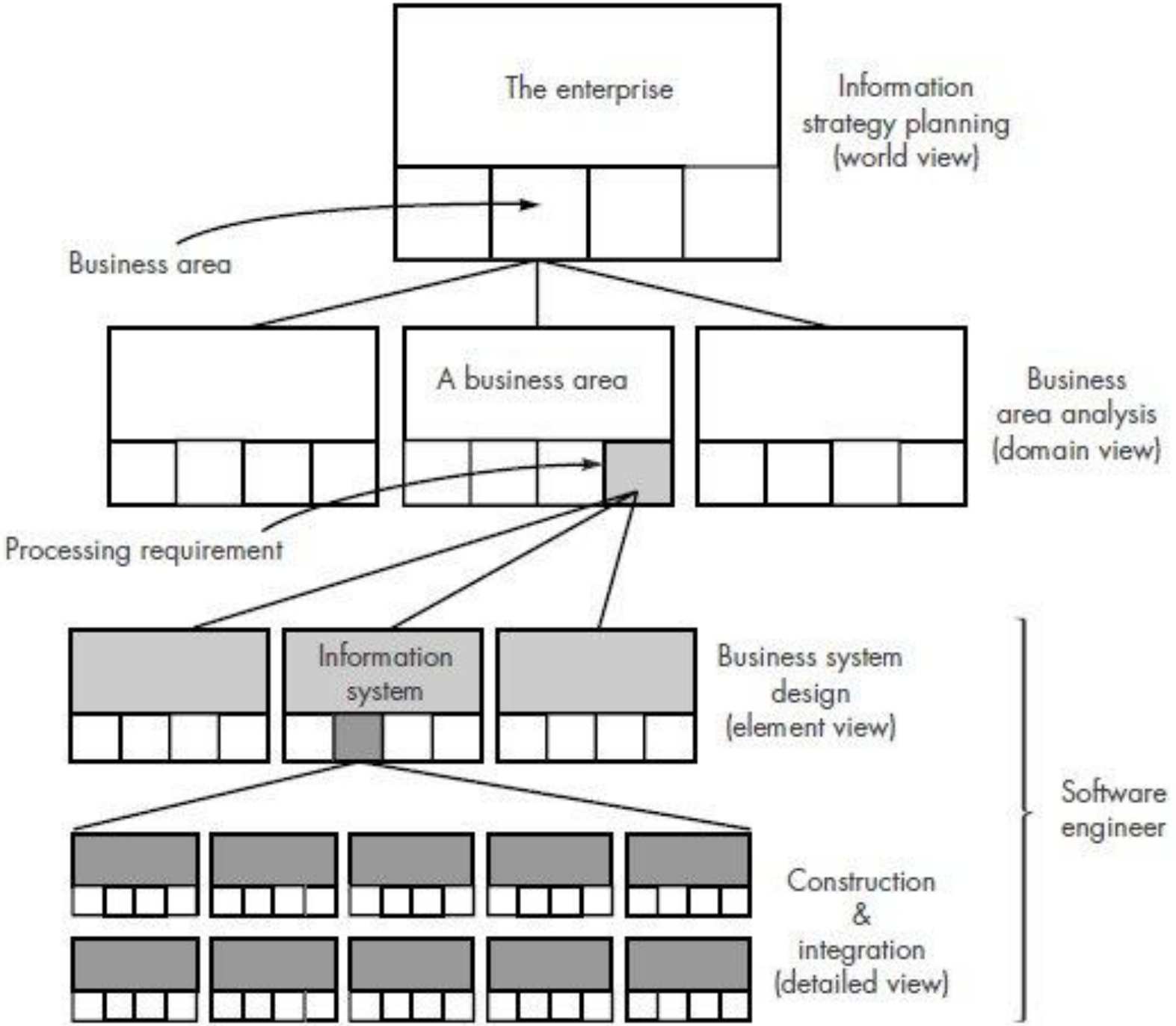


Kỹ thuật hệ thống – Phân loại

- **Kỹ thuật tiến trình nghiệp vụ** (Business Process Engineering)
  - Là kỹ thuật tập trung vào mặt nghiệp vụ của một tổ chức
  - Mỗi nghiệp vụ có thể tạo ra nhiều sản phẩm phần mềm
- **Kỹ thuật sản phẩm phần mềm** (Product Engineering)
  - Là kỹ thuật tập trung vào việc sản xuất ra 1 sản phẩm phần mềm cho một nghiệp vụ nào đó

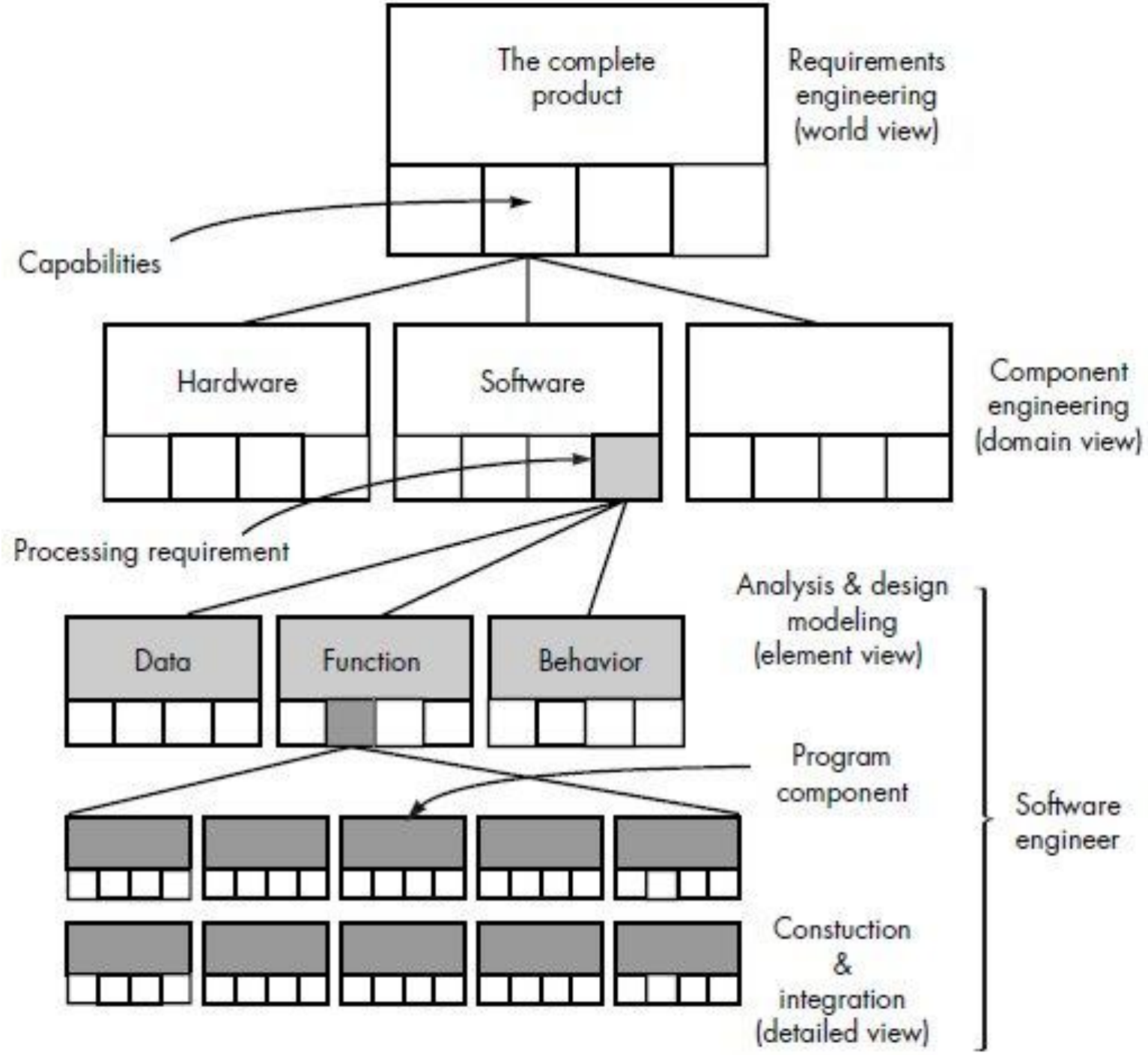
Kỹ thuật tiến trình nghiệp vụ

- **Mục đích**: Là quá trình xác định các kiến trúc cho phép một nghiệp vụ sử dụng thông tin một cách hiệu quả.
- Các kiến trúc cần xác định:
  - **Kiến trúc dữ liệu** (data architecture)
  - **Kiến trúc ứng dụng** (application architecture)
  - **Hạ tầng thông tin** (information infrastructure))


The enterprise
Information strategy planning (world view)
Business area
A business area
Business area analysis (domain view)
Processing requirement
Information system
Business system design (element view)
Software engineer
Construction & integration (detailed view)

Kỹ thuật sản phẩm phần mềm

- **Mục đích**: là chuyển các yêu cầu của khách hàng thành tập các tính năng (capabilities) trong sản phẩm phần mềm.
- **Tính chất**:
  - Nó cũng có tính phân cấp tương tự như kỹ thuật tiến trình nghiệp vụ và kỹ thuật hệ thống


The complete product
Requirements engineering (world view)
Capabilities
Hardware
Software
Component engineering (domain view)
Processing requirement
Analysis & design modeling (element view)
Data
Function
Behavior
Program component
Software engineer
Constuction & integration (detailed view)

Kỹ thuật thu thập và xử lý yêu cầu

- **Mục đích**: là cơ chế phù hợp để giúp hiểu rõ khách hàng cần gì, phân tích các yêu cầu, đánh giá tính khả thi, đàm phán để đưa ra giải pháp hợp lý.
- Kỹ thuật này bao gồm 4 bước:
  - Thu thập các yêu cầu
  - Phân tích và đàm phán
  - Kiểm tra tính hợp lệ của các yêu cầu
  - Quản lý các yêu cầu

Requirements Engineering: Thu thập các yêu cầu

- **Mục đích**: thu thập đầy đủ các loại yêu cầu của hệ thống cần xây dựng
- **Stakeholders**: là bất kỳ cá nhân hay nhóm người bị ảnh hưởng bởi hệ thống một cách trực tiếp hay gián tiếp. Đây là những nguồn cung cấp các yêu cầu cho hệ thống.

Requirements Engineering: Thu thập các yêu cầu

- Phân loại các yêu cầu:
  - **Yêu cầu về chức năng** (functional requirements): mô tả các dịch vụ mà hệ thống có thể thực hiện
  - **Yêu cầu phi chức năng** (non-functional requirements): là các y/c liên quan đến các ràng buộc như độ tin cậy, thời gian đáp ứng, độ an toàn, tuân theo các tiêu chuẩn, v.v

Requirements Engineering: Thu thập các yêu cầu

- Các khó khăn của việc thu thập y/c:
  - Vấn đề xác định không rõ phạm vi của hệ thống:
    - Không xác định rõ biên của hệ thống
  - Vấn đề thấu hiểu hệ thống không đầy đủ:
    - Không rõ hệ thống cần làm gì
    - Không rõ vấn đề thực sự của hệ thống là gì
    - Mức độ hiểu khác nhau, dễ dẫn đến hiểu lầm, hiểu sai
    - Thường số lượng và chủng loại y/c khá nhiều, thậm chí có thể mâu thuẫn với nhau
  - Các yêu cầu lại luôn thay đổi:
    - Do nhu cầu của người dùng
    - Do sự thay đổi trong môi trường

Requirements Engineering: Thu thập các yêu cầu

- Một số chỉ dẫn:
  - Xác định rõ những người dùng có thể giúp mô tả chi tiết các yêu cầu, cũng như các vấn đề của hệ thống
  - Xác định rõ môi trường kỹ thuật mà hệ thống sẽ hoạt động trong đó (như kiến trúc tính toán, hệ điều hành,v.v.)
  - Tạo ra các kịch bản sử dụng (usage scenarios hay use cases) nhằm giúp mô tả các y/c rõ ràng và chi tiết hơn

Requirements Engineering: Phân tích và đàm phán

- Phân tích y/c gồm:
  - Phân loại các y/c: y/c chức năng, y/c dữ liệu, mức độ ưu tiên của y/c
  - Mô hình hóa các y/c: mô hình phân cấp chức năng, biểu đồ luồng dữ liệu, mô hình thực thể-liên kết, biểu đồ chuyển trạng thái
  - Đặc tả các y/c
  - Kiểm tra sự nhất quán (consistency), sự rõ ràng (không nhập nhằng) của các y/c

Requirements Engineering: Phân tích và đàm phán

- Đàm phán nhằm:
  - Dung hòa các xung đột về y/c lợi ích giữa các khách hàng với nhau cũng như với và nhà phát triển
  - Đánh giá lại các y/c, nhằm chọn giải pháp phù hợp đáp ứng các y/c để giảm thiểu các rủi ro

Requirements Engineering: Kiểm tra tính hợp lệ của các y/c

- Giai đoạn này nhằm kiểm tra:
  - Tính rõ ràng, không nhập nhằng của các y/c
  - Các y/c là nhất quán
  - Các y/c tuân thủ các quy định của tổ chức, của các tiêu chuẩn mà tổ chức đang tuân theo hoặc hướng tới.

Requirements Engineering: Quản lý các y/c

- Giai đoạn này nhằm xác định và kiểm soát hiệu quả các thay đổi của các y/c. Nó gồm các công việc:
  - Phân loại và đánh số các y/c
  - Xây dựng các bảng theo dõi (traceability tables), có khả năng theo dõi các thay đổi của các y/c và ảnh hưởng của chúng
  - Cập nhật thường xuyên các bảng theo dõi khi có thay đổi trong các y/c

Tóm tắt

- Tính phân cấp của kỹ thuật hệ thống cho phép nhìn hệ thống ở nhiều mức khác nhau
- Mối liên hệ giữa *Kỹ thuật tiến trình nghiệp vụ* và *Kỹ thuật sản phẩm phần mềm*
- Các bước cơ bản trong *Kỹ thuật thu thập và xử lý yêu cầu*

Thank you!

Viện Điện tử - Viễn thông
Bộ Môn Điện tử - Kỹ thuật máy tính

Kỹ thuật phần mềm ứng dụng

Chương 4: Các khái niệm và các nguyên tắc phân tích

Các nội dung chính

- Giới thiệu về giai đoạn phân tích
- Các bước trong giai đoạn phân tích
- Một số kỹ thuật phân tích
- Các nguyên tắc phân tích

Giới thiệu về giai đoạn phân tích

- Mục đích:
  - Xác định rõ vấn đề của hệ thống hiện tại
    - Thời gian thực hiện kéo dài quá mức
    - Thường xuyên có sai sót do thực hiện thủ công
  - Từ đó đề xuất các giải pháp khả thi cho vấn đề
    - Nhà phân tích cần chỉ rõ cho khách hàng tính khả thi của giải pháp phần mềm đối với các vấn đề trên

Các bước phân tích

- Xác định vấn đề
- Đánh giá và tổng hợp các giải pháp
- Mô hình hóa
- Đặc tả

Các kỹ thuật phân tích

- Đặt câu hỏi:
  - Chọn các câu hỏi từ khái quát cho đến chi tiết
    - **Câu hỏi khái quát**: là những câu hỏi không trực tiếp đến hệ thống đang làm cụ thể những gì, mà khái quát hệ thống làm gì, ai đang dùng hệ thống, và những vấn đề nổi cộm là gì
    - **Câu hỏi chi tiết**: sẽ làm rõ hơn từng chức năng, từng thông tin vào/ra hệ thống, từng hành vi của hệ thống, rồi từng nghiệp vụ của từng người dùng

Các kỹ thuật phân tích

- FAST (Facilitated Application Specification Techniques): kỹ thuật này có một số đặc điểm:
  - **Tạo nhóm liên kết** (joint team): gồm các thành viên là khách hàng và nhà phát triển
  - **Tổ chức các cuộc họp của nhóm liên kết**: để các thành viên có thể gặp nhau và thảo luận để cùng phân tích các vấn đề và cùng tìm ra các giải pháp

Các kỹ thuật phân tích

- Use-case: còn gọi là kịch bản
  - Là bản mô tả việc sử dụng hệ thống của một người sử dụng
- Các bước xây dựng một UC:
  - Xác định những người dùng (user) và vai trò của mỗi người (actor). Mỗi người dùng có thể có nhiều vai trò khác nhau
  - Với mỗi vai trò của người dùng, xây dựng chi tiết kịch bản sử dụng của người đó với hệ thống

Các nguyên tắc phân tích

- Thông tin của bài toán cần được biểu diễn và được hiểu thấu
- Các chức năng của phần mềm phải được xác định
- Hành vi của phần mềm (dãy các sự kiện bên ngoài) phải được biểu diễn
- Các mô hình mô tả thông tin, các chức năng và hành vi phải được phân chia phân mức để có thể làm rõ các thông tin chi tiết
- Tiến trình phân tích nên chuyển dần từ các thông tin thiết yếu sang các chi tiết cài đặt

Thank you!

Viện Điện tử - Viễn thông
Bộ Môn Điện tử - Kỹ thuật máy tính

Kỹ thuật phần mềm ứng dụng

Chương 7: Phân tích hệ thống

Phần 1: Giới thiệu chung

Các nội dung chính

- Xác định lại vấn đề của hệ thống hiện tại
- Đề xuất giải pháp phù hợp
- Các thành phần cần phân tích:
  - Phân tích chức năng
  - Phân tích dữ liệu
  - Phân tích hành vi

Xác định lại vấn đề của hệ thống hiện tại

- Sau giai đoạn khảo sát kỹ lưỡng các y/c và hoạt động của hệ thống hiện tại, thì bắt đầu giai đoạn phân tích là giai đoạn thích hợp để làm rõ hơn vấn đề của hệ thống hiện tại, để từ đó xác định chi tiết và đầy đủ hơn giải pháp phần mềm để giải quyết các vấn đề đó.
- Các vấn đề của HT hiện tại chủ yếu nảy sinh do phương thức làm việc cũ, cũng như theo các cơ chế QL cũ không hiệu quả.

Xác định lại vấn đề của hệ thống hiện tại

- Các vấn đề cần phát hiện:
  - Chu trình làm việc cũ có chỗ không hợp lý
  - Công việc chồng chéo
  - Tốc độ thực hiện thường xuyên chậm chạp, không đáp ứng được y/c của người dùng và khách hàng
  - Thường xảy ra sai sót do phải thực hiện thủ công
  - Việc kiểm tra, kiểm soát gặp khó khăn

Đề xuất giải pháp phù hợp

- Đề xuất giải pháp phần mềm cho hệ thống mới nhằm giải quyết các vấn đề đã xác định ở trên:
  - Đưa ra chu trình làm việc hợp lý hơn nếu cần
  - Phân công nhiệm vụ cụ thể và rõ ràng hơn để tránh chồng chéo
  - Phần mềm nên tập trung vào các khâu mà đang có vấn đề chậm chạp, hoặc thường xuyên có sai sót
  - Hệ thống mới cũng nên bổ sung các tính năng hỗ trợ việc kiểm tra, giám sát của các nhà quản lý

Đề xuất giải pháp phù hợp

- Sau khi đề xuất giải pháp cho HT mới, cần có các cuộc gặp chính thức với khách hàng để thống nhất giải pháp:
  - Trình bầy giải pháp mới cho khách hàng
  - Kiến nghị với KH về các điều chỉnh cần thiết đối với chu trình cũ nếu có
  - Lắng nghe ý kiến phản hồi, để có các điều chỉnh cần thiết, nhằm làm tăng tính khả thi của giải pháp đó.
  - Cùng nhau đi đến thống nhất giải pháp mới

Phân tích chức năng

- **Mục đích**:
  - Làm rõ các thành phần chức năng của hệ thống mới và các thành phần liên quan như đối tượng sử dụng, dữ liệu trao đổi, trong đó chức năng đóng vai trò trung tâm
  - Làm rõ mối quan hệ giữa các chức năng với nhau và với các thành phần khác

Phân tích chức năng

- **Các phương pháp**:
  - Theo mức độ trừu tượng:
    - Phân tích đại thể
    - Phân tích chi tiết
  - Theo góc nhìn:
    - **Phân tích tĩnh**: chỉ tập trung xác định các chức năng của hệ thống, không quan tâm việc thực hiện các chức năng đó như thế nào
    - **Phân tích động**: ngoài chức năng, còn xác định các thành phần khác như đối tượng sử dụng, dữ liệu, và mối quan hệ giữa chúng khi thực hiện chức năng như thế nào

Phân tích chức năng

- **Công cụ sử dụng**:
  - Phân tích tĩnh và đại thể: biểu đồ phân cấp chức năng (BPC)
  - Phân tích động: biểu đồ luồng dữ liệu (BLD)
  - Phân tích chi tiết: các đặc tả chức năng:
    - Đặc tả chức năng (PSpec)
    - Các bảng quyết định

Phân tích chức năng – Biểu đồ phân cấp chức năng

- Là mô hình phân tích đại thể và tĩnh về các chức năng của hệ thống
- Xác định mối quan hệ bao hàm giữa các chức năng: chức năng đại thể bao hàm các chức năng chi tiết hơn ➔ tạo ra cây phân cấp các chức năng
- Việc phân cấp chức năng này thường được dùng để xác định menu chính của phần mềm sau này

Phân tích chức năng – Biểu đồ luồng dữ liệu (BLD)

- Là mô hình phân tích động hệ thống
- Xác định rõ những đối tượng mà hệ thống mới sẽ phục vụ (người dùng, tác nhân ngoài)
- Làm rõ các thành phần chức năng của hệ thống mới
- Xác định rõ mối quan hệ giữa các đối tượng và các chức năng: đối tượng nào dùng chức năng nào và dùng như thế nào → các luồng dữ liệu vào/ra hệ thống
- Xác định mối quan hệ giữa các chức năng như: thứ tự thực hiện, đồng bộ, thông tin trao đổi → các luồng thông tin nội bộ

Phân tích chức năng – Các đặc tả chức năng

- Đặc tả chức năng (PSpec):

  Mô tả chi tiết hoạt động bên trong của mỗi chức năng, có vai trò như giải thuật thực hiện chức năng đó

- Các bảng quyết định:

  Dùng để mô tả chi tiết các tình huống có nhiều lựa chọn trong các đặc tả chức năng

Phân tích dữ liệu

- Mục đích:
  - Làm rõ các thành phần dữ liệu của hệ thống và mối quan hệ giữa chúng
- Phương pháp:
  - Phân tích hướng dữ liệu: chỉ tập trung làm rõ phần dữ liệu và các ràng buộc nghiệp vụ trong đó
- Công cụ:
  - Từ điển dữ liệu
  - Mô hình thực thể liên kết

Phân tích hành vi

- Mục đích:
  - Xác định các trạng thái của hệ thống, và các sự kiện gây ra sự thay đổi các trạng thái đó
  - Xác định các hành động cần làm khi có một sự kiện nào đó xảy ra
  - Thường được dùng để mô tả các hệ thống mà thời điểm, thứ tự xuất hiện các sự kiện là không xác định (ngẫu nhiên bất kỳ), như các hệ thống điều khiển thời gian thực
- Công cụ:
  - Biểu đồ chuyển trạng thái

Tóm tắt

- Xác định chi tiết và cụ thể hơn các vấn đề của hệ thống hiện tại
- Đưa ra giải pháp phần mềm hợp lý giải quyết thỏa đáng các vđ trên
- Đi vào phân tích các thành phần của hệ thống:
  - Phân tích chức năng
  - Phân tích dữ liệu
  - Phân tích hành vi

Xin cảm ơn!

Viện Điện tử - Viễn thông
Bộ Môn Điện tử - Kỹ thuật máy tính

Kỹ thuật phần mềm ứng dụng

Chương 7: Phân tích hệ thống

Phần 2: Phân tích về chức năng

Các nội dung chính

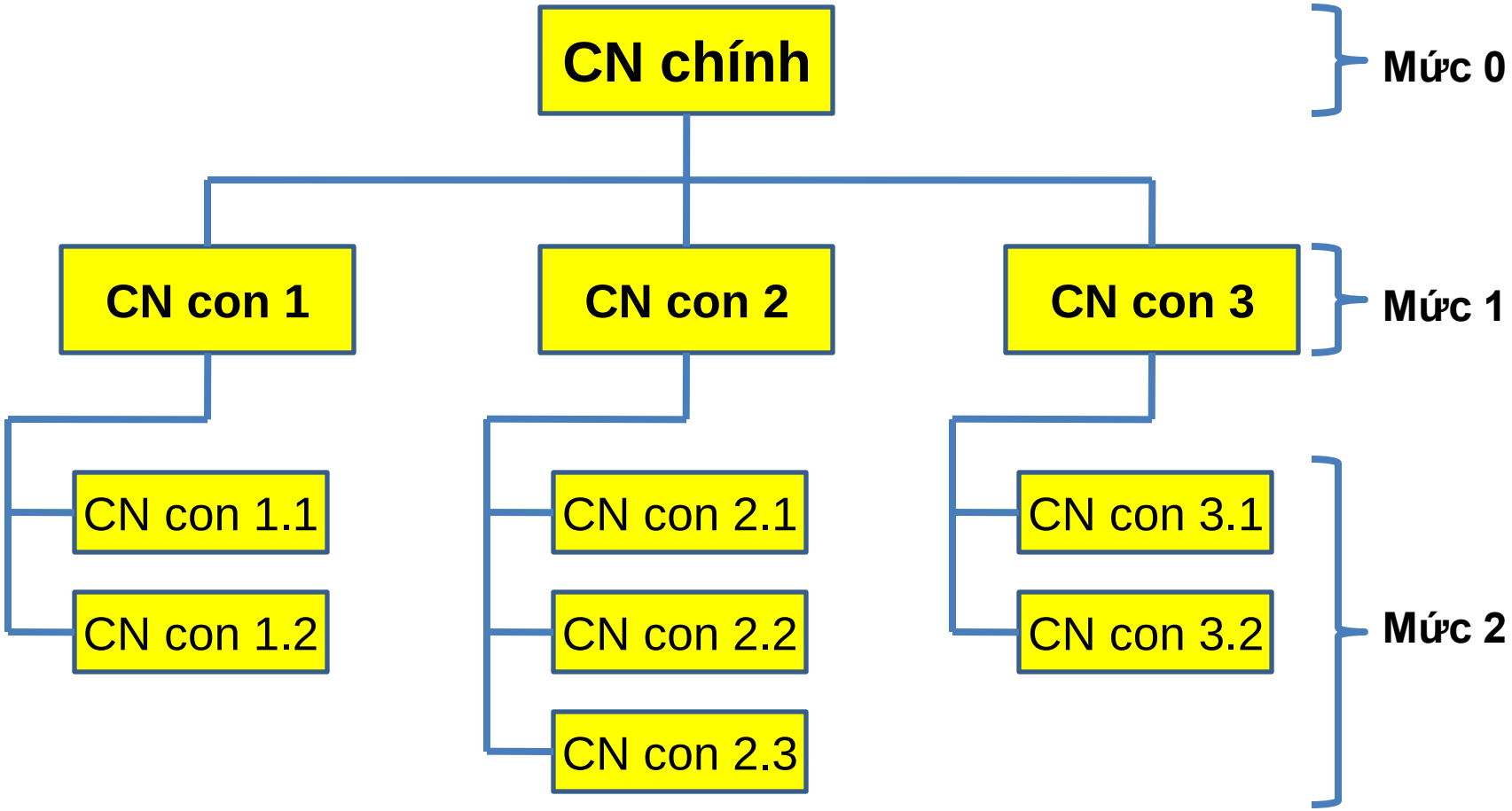
- Phân tích tĩnh thuần túy chức năng: Biểu đồ phân cấp chức năng
- Phân tích động và tổng thể: các biểu đồ luồng dữ liệu
- Phân tích chi tiết: đặc tả tiến trình
- Phân tích hành vi: biểu đồ chuyển trạng thái

Biểu đồ phân cấp chức năng

- Mục đích:
  - Xác định mối quan hệ bao hàm giữa các chức năng: chức năng đại thể bao hàm các chức năng chi tiết hơn ➔ tạo ra cây phân cấp các chức năng
  - Việc phân cấp chức năng này thường được dùng để xác định menu chính của phần mềm sau này

Biểu đồ phân cấp chức năng

- Cấu trúc:



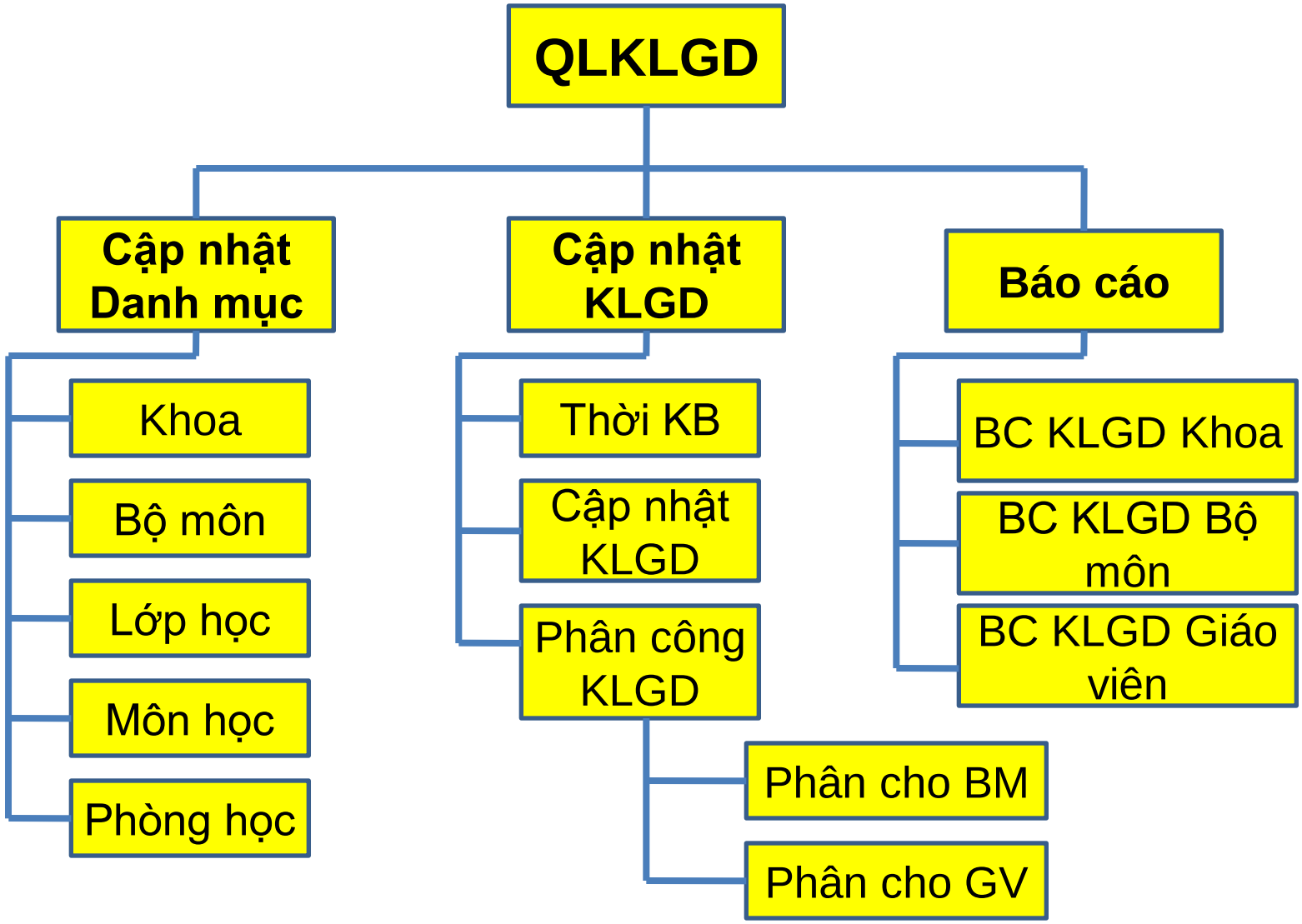
Biểu đồ phân cấp chức năng

- Phương pháp xây dựng:
  - **Đầu vào**:
    - các kịch bản sử dụng;
    - phần mô tả khái quát các chức năng của hệ thống;
  - **Đầu ra**: **BPC**
  - **Nội dung**:
    - Xác định mối quan hệ bao hàm giữa các chức năng, từ đó xác định số mức của biểu đồ
    - Xác định các chức năng của từng mức

Biểu đồ phân cấp chức năng

- Bản chất mối quan hệ bao hàm: chức năng A bao hàm chức năng B:
  - *Ý nghĩa 1*: chức năng B là một bộ phận cần thiết của chức năng A
  - *Ý nghĩa 2*: chức năng A là trừu tượng để gộp nhóm các chức năng, trong đó có chức năng B
- Các phương pháp gộp nhóm:
  - Gộp theo người dùng
  - Gộp theo loại chức năng: đầu vào (nhập DL), đầu ra (báo cáo), xử lý

Ví dụ: BPC cho hệ thống QL KLGD

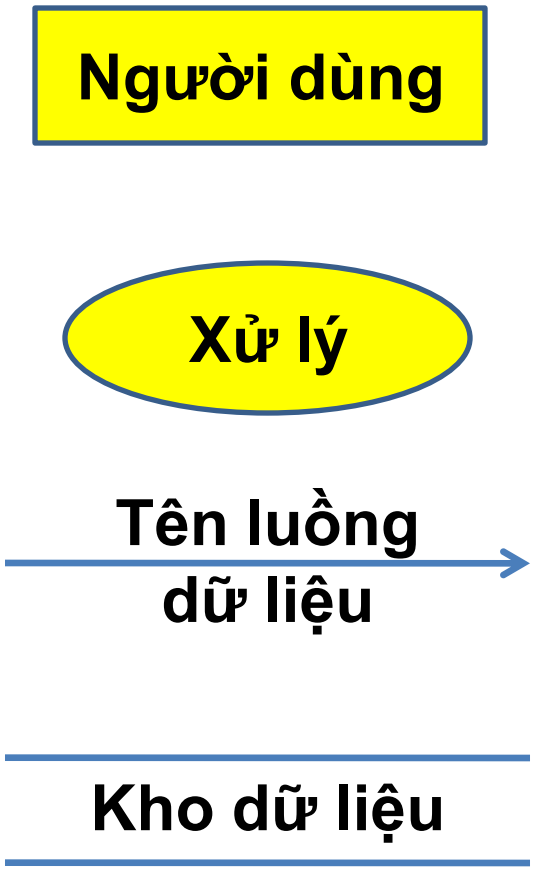


Biểu đồ luồng dữ liệu

- Mục đích:
  - Là mô hình phân tích động hệ thống
  - Xác định rõ những đối tượng mà hệ thống mới sẽ phục vụ (người dùng, tác nhân ngoài)
  - Làm rõ các thành phần chức năng của hệ thống mới
  - Xác định rõ mối quan hệ giữa các đối tượng và các chức năng: đối tượng nào dùng chức năng nào và dùng như thế nào → các luồng dữ liệu vào/ra hệ thống
  - Xác định mối quan hệ giữa các chức năng như: thứ tự thực hiện, đồng bộ, thông tin trao đổi → các luồng thông tin nội bộ

Biểu đồ luồng dữ liệu – Cấu tạo

- Tác nhân ngoài/trong
- Chức năng/tiến trình
- Luồng dữ liệu
- Kho dữ liệu



Biểu đồ luồng dữ liệu - Cấu tạo

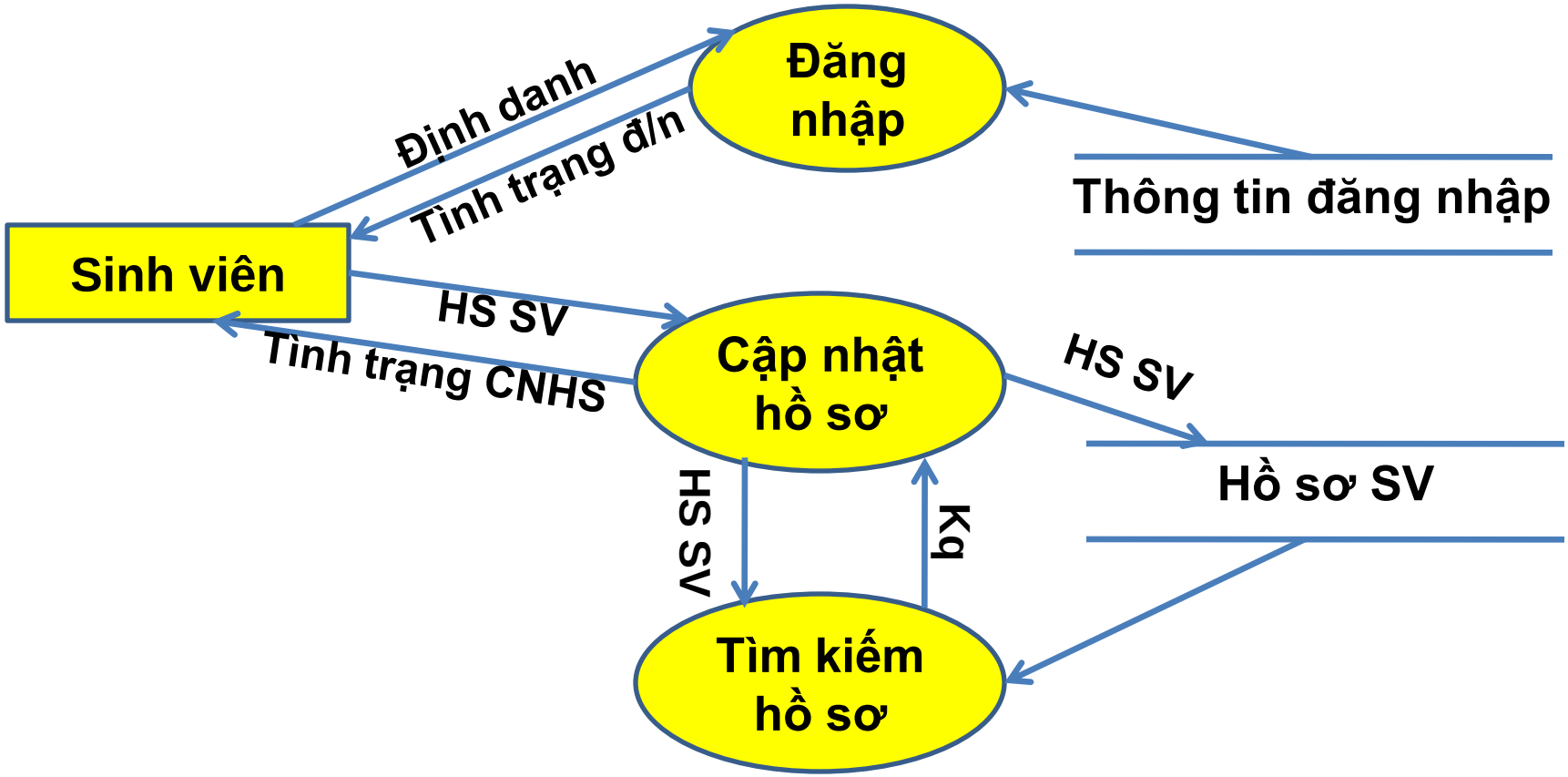
- **Tác nhân ngoài** (đối tác – actor): là đối tượng ở bên ngoài hệ thống, nhưng có nhu cầu trao đổi thông tin với hệ thống. Đây thường là những người dùng cuối của hệ thống

- **Tác nhân trong**: là một chức năng hoặc hệ con đã được mô tả ở trang khác của mô hình, nhưng lại có trao đổi thông tin với các thành phần trong trang hiện tại

Biểu đồ luồng dữ liệu - Cấu tạo

- Chức năng/tiến trình:
  - Có nhiệm vụ tiếp nhận và biến đổi luồng dữ liệu vào theo một quy tắc nào đó, rồi đưa thông tin ra
- Luồng dữ liệu:
  - Biểu diễn luồng thông tin trao đổi từ nguồn thông tin đến đích thông tin. Trong đó nguồn và đích có thể là các thành phần của biểu đồ
- Kho dữ liệu:
  - Biểu diễn nơi lưu trữ thông tin trong một thời gian nào đó, và cũng đồng thời giúp đồng bộ hoạt động của các tiến trình

Biểu đồ luồng dữ liệu – Ví dụ



Biểu đồ luồng dữ liệu – Các mức

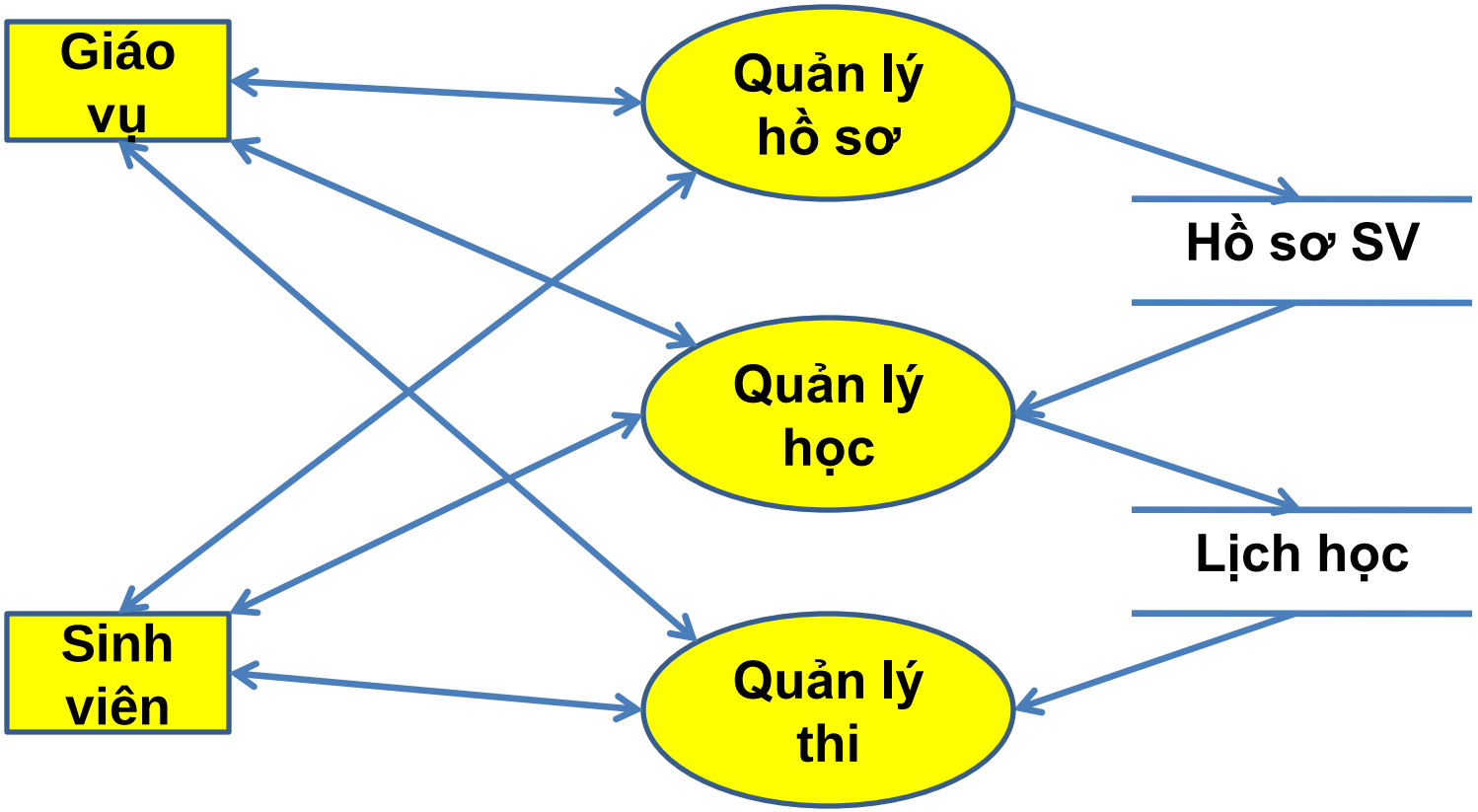
- **Biểu đồ mức 0** (mức khung cảnh): là biểu đồ mức cao nhất. Ở mức này chỉ có 1 tiến trình duy nhất là hệ thống cần xây dựng, được nối với các tác nhân bằng các luồng dữ liệu ngoài
- **Biểu đồ mức 1** (mức đỉnh): là kết quả của việc phân rã chức năng duy nhất trong mức 0, thành các tiến trình con. Ở mức này có thể bắt đầu có nhu cầu xuất hiện các Kho dữ liệu
- **Các biểu đồ mức 2,3,v.v**: làm tương tự như cho biểu đồ mức đỉnh

Biểu đồ luồng dữ liệu – Mức 0

- (1): là dữ liệu mà SV có thể cập nhật như hồ sơ SV; cũng có các thông tin mà SV muốn tìm kiếm như Lịch học, Lịch thi, Kết quả thi,v.v
- (2): là dữ liệu về các kết quả tìm kiếm trên

- (3): là dữ liệu mà GVụ có thể cập nhật và tìm kiếm như hồ sơ SV, hồ sơ GV, Kết quả kiểm tra, thi của SV, Lịch giảng dạy, Lớp học,v,v
- (4): là dữ liệu về các kết quả tìm kiếm trên

Biểu đồ luồng dữ liệu – Mức 1



Đặc tả tiến trình

- Xem tài liệu tham khảo

Biểu đồ chuyển trạng thái

- Xem tài liệu tham khảo

Cảm ơn!

Viện Điện tử - Viễn thông
Bộ Môn Điện tử - Kỹ thuật máy tính

Kỹ thuật phần mềm ứng dụng

Chương 8: Thiết kế hệ thống

Phần 1: Giới thiệu chung

Nội dung chính

- Mục đích của thiết kế
- Các nguyên tắc thiết kế
- Các phần cần thiết kế
  - Thiết kế CSDL ➔ CSDL ít nhất ở dạng chuẩn 3
  - Thiết kế kiến trúc ➔ Lược đồ cấu trúc chương trình
  - Thiết kế giao diện ➔ các menu, form nhập, mẫu báo cáo, thông báo

Mục đích của giai đoạn Thiết kế

- Là quá trình chuyển các y/c của phần mềm sang dạng biểu diễn của phần mềm mà nó có thể được đánh giá về chất lượng trước khi cài đặt.
- Thiếu thiết kế, việc cài đặt có thể gặp các vấn đề:
  - **Thiếu kế hoạch cài đặt**: không biết rõ thứ tự cài đặt các thành phần, do đó gây ra sự lộn xộn và khó khăn trong việc ước lượng và phân công công việc
  - **Không rõ ràng**: chưa hiểu rõ các y/c sẽ được cài đặt thế nào
  - **Khó nâng cấp và bảo trì**: khi có lỗi, rất khó xác định nó nằm ở phần nào. Khi muốn nâng cấp cũng không biết cần nâng cấp ở đâu, ảnh hưởng của nó đến hệ thống hiện tại thế nào

➔ **Ảnh hưởng xấu đến chất lượng và tiến độ làm phần mềm**

Các nguyên tắc thiết kế

- Sự trừu tượng (abstraction)
- Làm mịn (tinh chỉnh từng bước - refinement)
- Modul hóa (modularity)

Các nguyên tắc thiết kế

- Sự trừu tượng:
  - Là sự tập trung vào một vấn đề ở một mức khái quát nào đó, và bỏ qua các chi tiết không liên quan
  - Quá trình thiết kế hệ thống đòi hỏi nhiều mức trừu tượng khác nhau
  - Với phần mềm thì có 3 loại trừu tượng
    - Trừu tượng thủ tục
    - Trừu tượng dữ liệu
    - Trừu tượng điều khiển

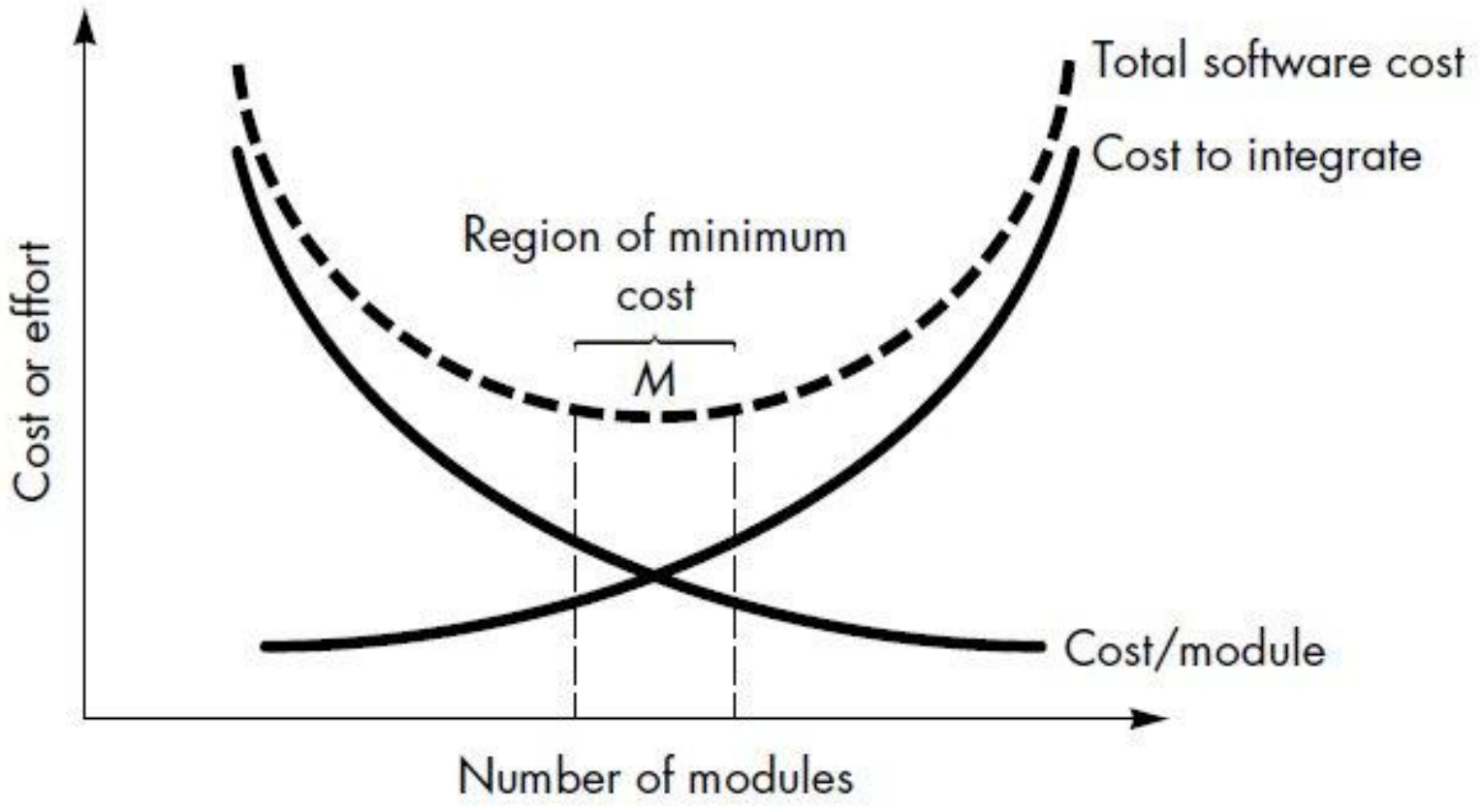
Các nguyên tắc thiết kế

- Làm mịn (tinh chỉnh từng bước - refinement)
  - Là quá trình làm chi tiết hóa từng thành phần của một biểu diễn nào đó, để dần đưa nó sang biểu diễn ở dạng chi tiết hơn (giảm mức độ trừu tượng)
  - Việc làm mịn giúp cho việc chuyển đổi này diễn ra một cách không đột ngột và dễ dàng quản lý.

Các nguyên tắc thiết kế

- Modul hóa (modularity):
  - Là quá trình phân chia hệ thống/phần mềm thành các thành phần riêng rẽ có tên và tương đối độc lập
  - Là một kỹ thuật cơ bản nhất để quản lý một cách hiệu quả độ phức tạp của hệ thống
  - Modul hóa tốt có thể giúp giảm thiểu thời gian và chi phí phát triển hệ thống

Modul hóa



Quan hệ giữa modul hóa và chi phí phần mềm

Module hóa hiệu quả

- Che dấu thông tin
  - Là cách thiết kế làm sao để thông tin trong một modul (cả chức năng và dữ liệu) là không nhìn thấy và không truy nhập được từ các thành phần bên ngoài mà không có nhu cầu về thông tin đó
- Độc lập chức năng (functional independence)
  - Là tính chất phản ánh mức độ đơn nhất về chức năng và đơn giản về giao diện của một modul. Nó được đo lường theo 2 tiêu chuẩn:
    - Mức độ cố kết (cohesion)
    - Mức độ tương liên (coupling)

Mức độ cố kết

- Khái niệm:

  Mức độ cố kết của một modul là một đơn vị đo về sức mạnh chức năng của modul đó. Mức độ này càng cao thì tính độc lập chức năng cũng càng cao.

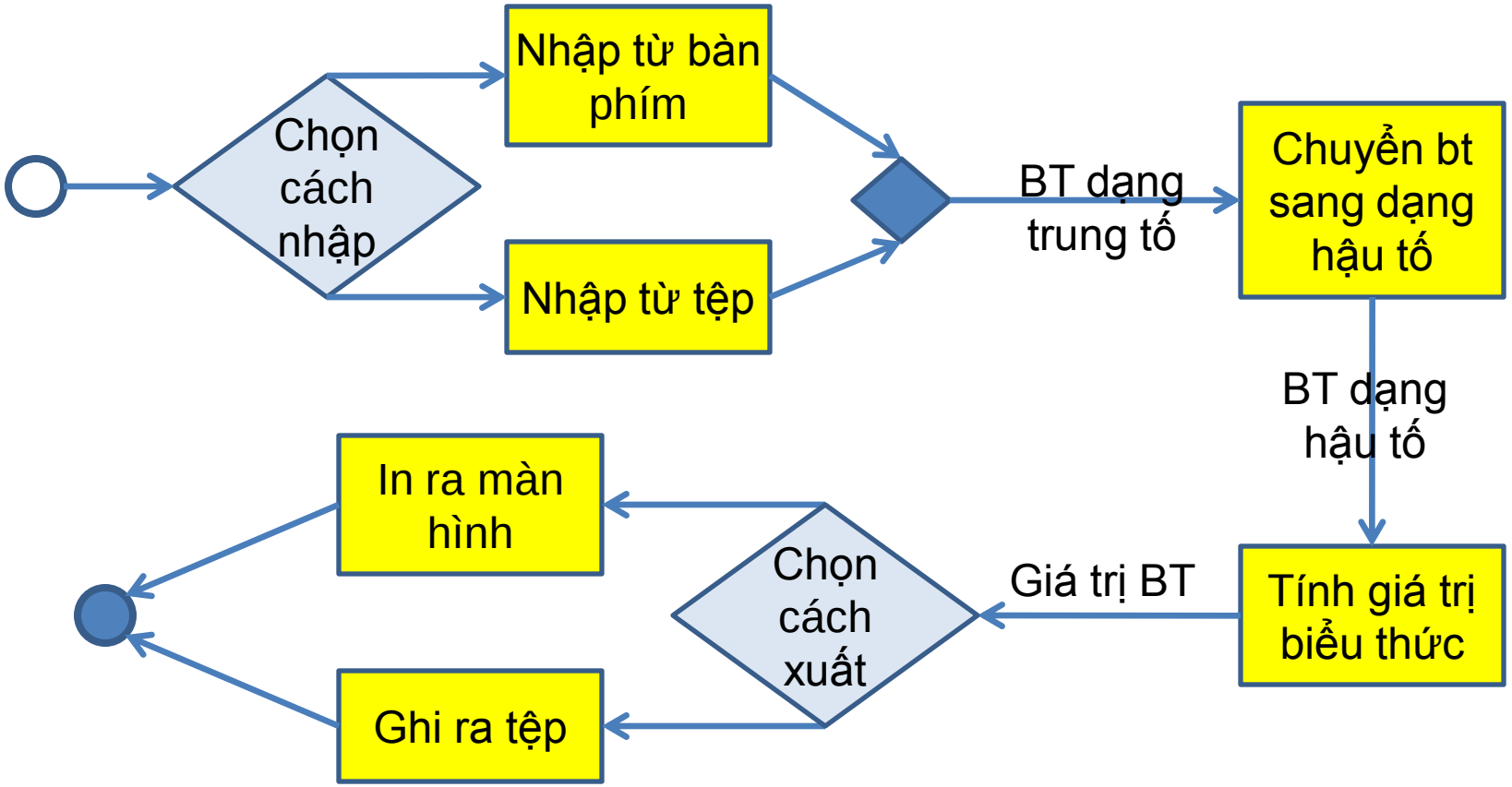
Các loại cố kết và mức độ của chúng

Mức độ	Loại cố kết	Ý nghĩa
Thấp	Cố kết trùng khớp	Modul bao gồm một dãy các công việc mà liên quan rất ít đến nhau
	Cố kết logic	Modul bao gồm một dãy các công việc mà có liên quan đến nhau một cách logic
	Cố kết thời gian	Modul bao gồm một dãy các công việc mà phải hoàn thành trong cùng một khoảng tg.
Vừa	Cố kết thủ tục	Các công việc trong modul đó liên quan đến nhau và phải được thực hiện theo một trật tự nhất định
	Cố kết truyền thông	Khi các công việc trong một modul cùng sử dụng một phần nào đó của một cấu trúc dữ liệu
Cao	Cố kết thủ tục rõ ràng	Khi modul đó chỉ thực hiện một công việc

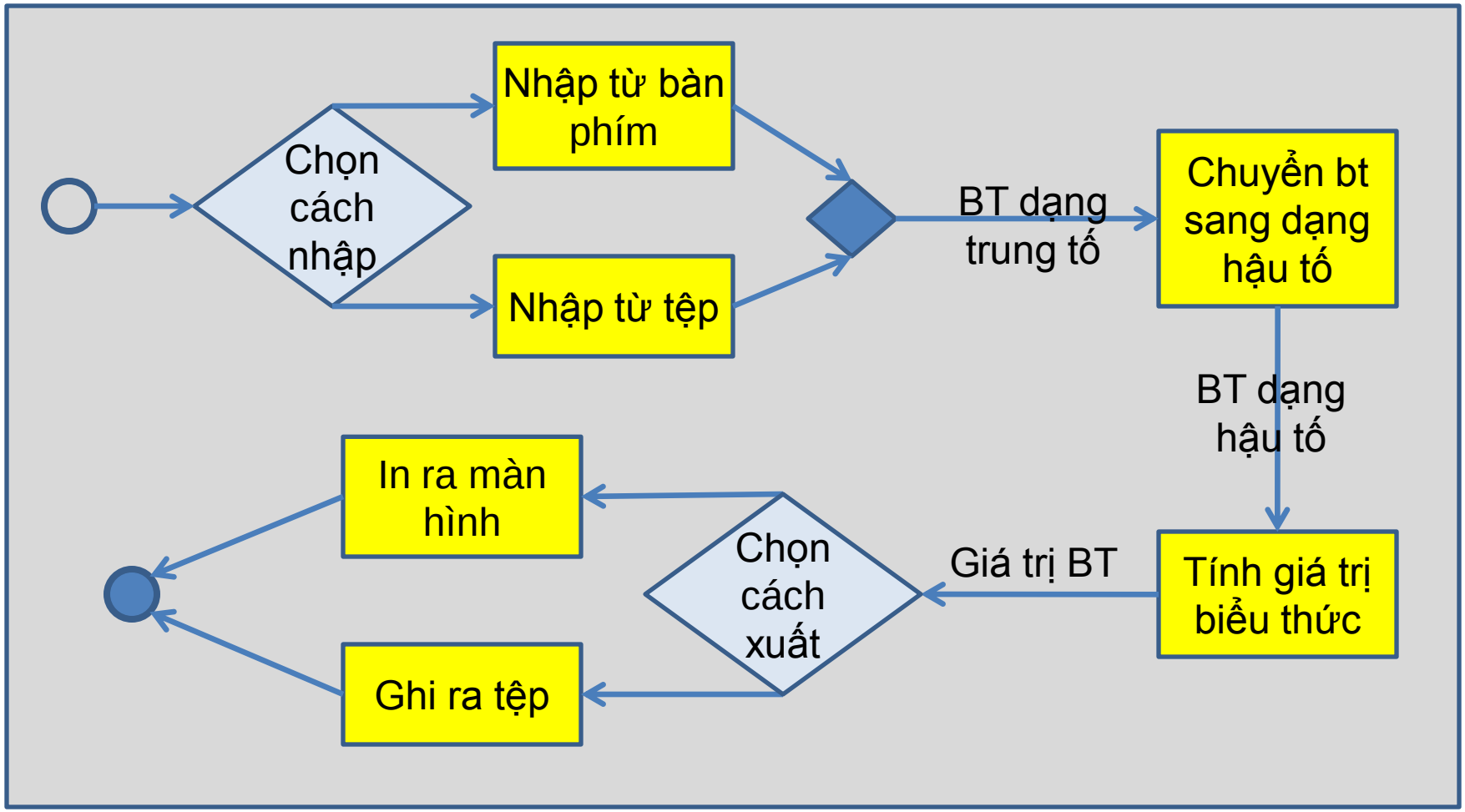
Ví dụ về mức độ cố kết

- Bài toán: viết một chương trình tính giá trị một biểu thức số học mà có thể được nhập từ bàn phím hay từ một tệp văn bản. Kết quả đưa ra cũng có thể đưa ra màn hình hoặc ghi vào tệp văn bản.
- Sơ đồ cho giải thuật của bài toán trên được cho ở hình sau:

Ví dụ về mức độ cố kết

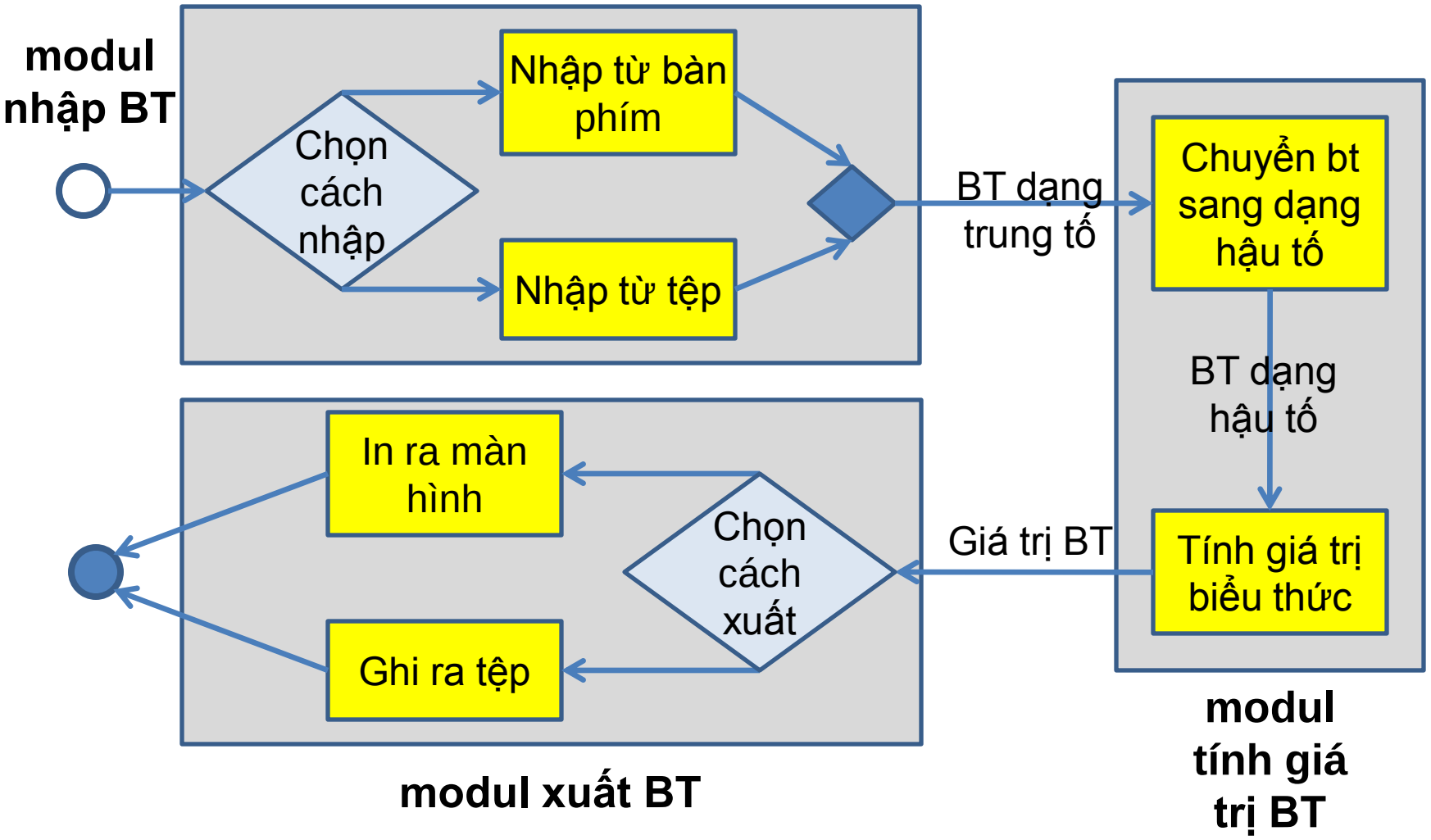


Mức độ cố kết thấp: trùng khớp

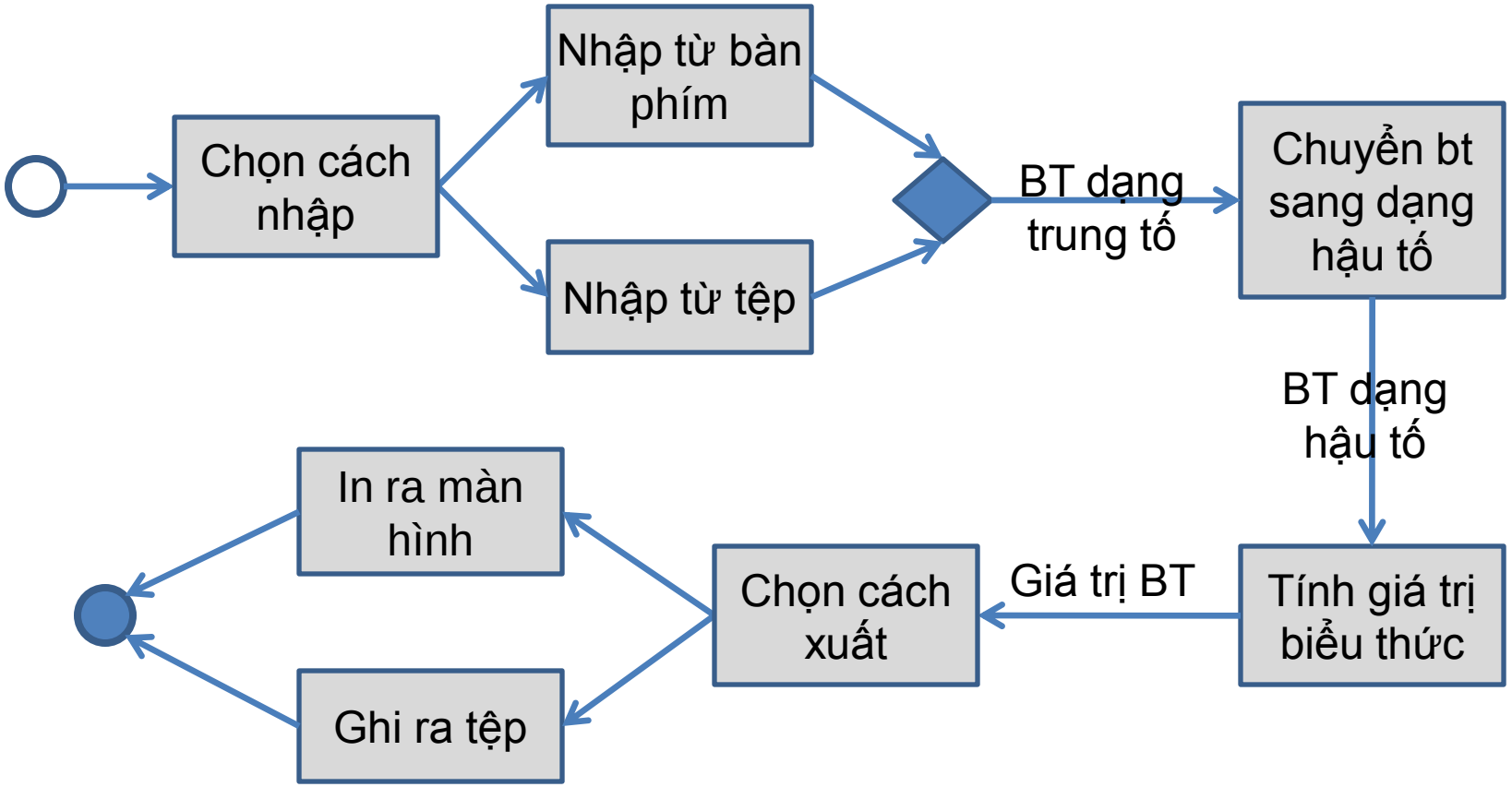


Một modul làm toàn bộ các công việc

Mức độ cố kết vừa: thủ tục



Mức độ cố kết cao: thủ tục rõ ràng



Mức độ tương liên

- Khái niệm:

  Mức độ tương liên của một modul đơn vị đo lường mức độ kết nối của modul đó với các modul khác. Điều này phụ thuộc vào độ phức tạp của giao diện, điểm truy nhập hay tham chiếu của modul.

Mức độ tương liên

Mức độ	Loại	Ý nghĩa
Thấp	Tương liên dữ liệu	Là khi một modul truyền tham số đến modul khác
Vừa	Tương liên điều khiển	Là khi một modul truyền một thông tin điều khiển (cờ điều khiển) đến modul khác
	Tương liên ngoài	Là khi một modul phụ thuộc vào một thiết bị bên ngoài (như các t/b nhập/xuất)
Cao	Tương liên chung dữ liệu	Là khi một số modul tham chiếu/chia sẻ đến cùng một đối tượng dữ liệu toàn cục
	Tương liên nội dung	Là khi một modul sử dụng dữ liệu hay điều khiển thông tin trong phạm vi của một modul khác; Nó cũng xuất hiện khi có tồn tại lệnh rẽ nhánh trong modul

Cảm ơn!

Viện Điện tử - Viễn thông
Bộ Môn Điện tử - Kỹ thuật máy tính

Kỹ thuật phần mềm ứng dụng

Chương 8: Thiết kế hệ thống

Phần 2: TK cơ sở dữ liệu

Các nội dung chính

- Các bước thiết kế một CSDL
- Ví dụ minh họa

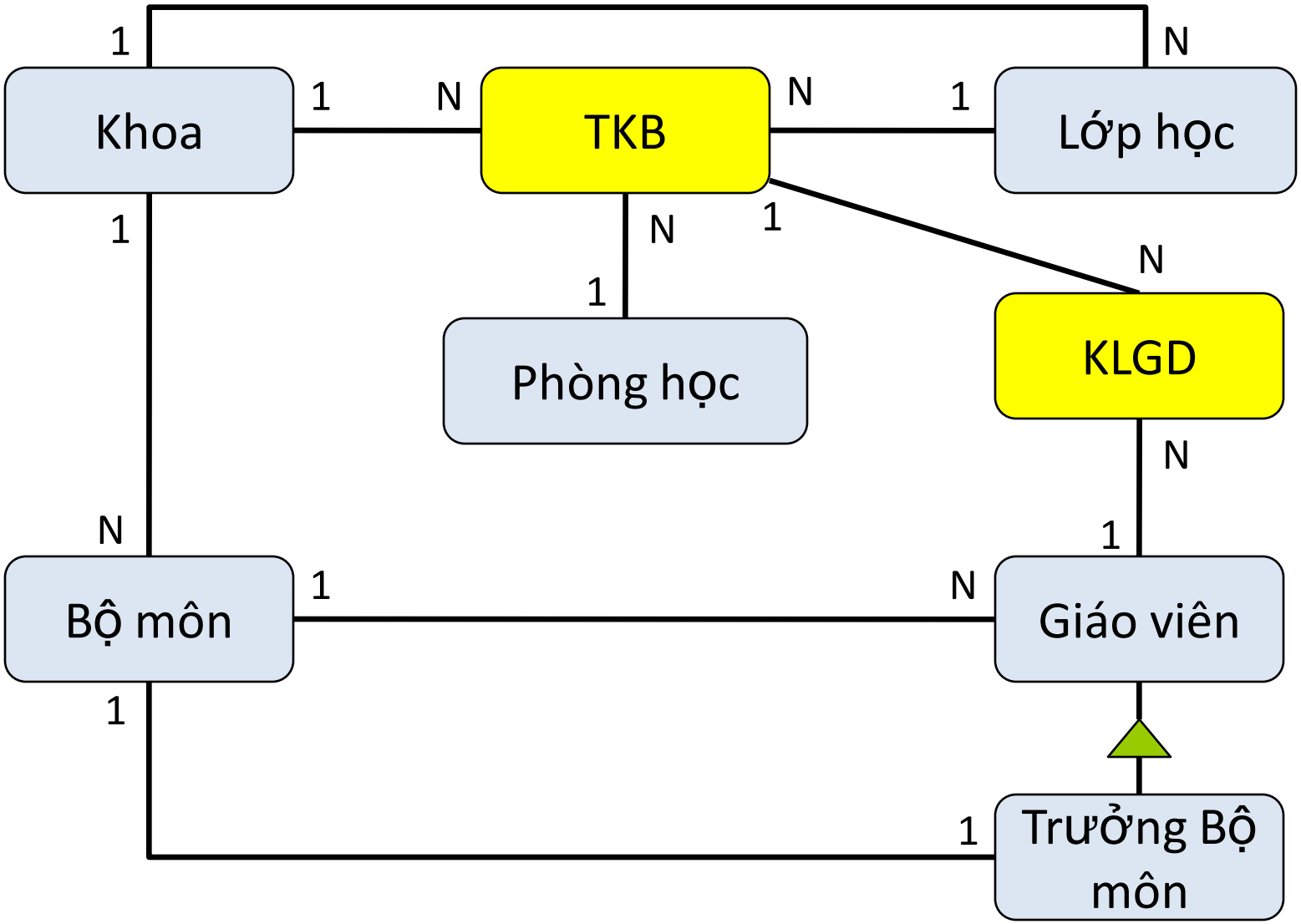
Thiết kế Cơ sở dữ liệu

1. Chuyển từ mô hình thực thể liên kết sang mô hình quan hệ.
2. Xác định các phụ thuộc hàm từ các ràng buộc dữ liệu và các quy tắc nghiệp vụ.
3. Chuẩn hóa các lược đồ quan hệ, đưa chúng về các lược đồ ở dạng chuẩn 3.
4. Bổ sung thêm các thuộc tính khóa nếu cần, nhất là khi quan hệ có nhiều thuộc tính khóa.
5. Xác định chi tiết các miền giá trị cho các thuộc tính, từ đó xác định kiểu dữ liệu cho chúng. Lập bảng mô tả chi tiết các kiểu dữ liệu của từng thuộc tính cho từng quan hệ (bảng)

Ví dụ về TK CSDL

Tên thực thể	Tên sử dụng	Các thuộc tính
Khoa	TKhoa	Tên khoa, Văn phòng, Điện thoại, Fax
Bộ môn	TBoMon	Tên BM, Văn phòng
Giáo viên	TGiaoVien	TênGV, Ngày sinh, Địa chỉ, Chức danh, Chức vụ, Ngày chức danh, Ngày chức vụ
Lớp học	TLopHoc	Tên lớp, Khóa học, Số Lượng SV
Phòng học	TPhong	Tên phòng, SL chỗ ngồi
Trưởng BM	TTrBM	Gồm các thuộc tính của Giáo viên, Ngày nhậm chức, Ngày thôi chức
Thời khóa biểu	TKB	Năm học, Học kỳ, Khoa, Lớp học, Phòng học, Môn học, Tiết học
Khối lượng giảng dạy	KLGD	TKB, Giáo viên

Ví dụ về TK CSDL



Các bảng được suy ra

Bảng Khoa	
Thuộc tính	Tên khoa, Văn phòng, Điện thoại, Fax
Ràng buộc & Quy tắc nghiệp vụ	**Phụ thuộc hàm**
Mỗi khoa có 1 văn phòng Và thường mỗi VP thuộc về một khoa, vì việc chuyển địa điểm làm việc của khoa rất hiếm khi xảy ra	Tên khoa → Văn phòng;
Mỗi văn phòng có một số fax và có thể có nhiều số điện thoại	Văn phòng → Fax
Chuẩn hóa: Khóa: K = (Tên khoa, Điện thoại); Vi phạm chuẩn 2 và 3 ➔Tách thành 3 quan hệ: **Khoa** (Tên khoa, Văn phòng); **VănPhòng** (Văn phòng, Fax); **ĐiệnThoạiKhoa** (Tên khoa, Điên thoai);	**Ghi chú**: có thể cân nhắc việc ghép 2 bảng Khoa và Văn phòng do mối qhệ giữa 2 bảng này.

Các bảng được suy ra

Bảng Giáo viên	
Thuộc tính	TênGV, Ngày sinh, Địa chỉ, Chức danh, Ngày chức danh, Chức vụ, Ngày chức vụ
Ràng buộc & Quy tắc nghiệp vụ	**Phụ thuộc hàm**
Mỗi GV có 1 ngày sinh	TênGV → Ngày sinh;
Mỗi GV có thể có nhiều địa chỉ, nhưng ở mỗi thời điểm thì chỉ có 1 chức danh và nhiều nhất là 1 chức vụ (có thể không có).	TênGV, Ngày chức danh → Chức danh; TênGV, Ngày chức vụ → Chức vụ
Chuẩn hóa: Khóa duy nhất: K = (TênGV, Địa chỉ, Ngày chức danh, Ngày chức vụ) Vi phạm chuẩn 2; ➔ Tách thành 4 quan hệ: **GiáoViên** (TênGV, Ngày sinh); **GV-Địa chỉ** (TênGV, Địa chỉ, Ngày chức danh, Ngày chức vụ); **GV-Chức danh** (TênGV, Ngày chức danh, chức danh); **GV-Chức vụ** (TênGV, Ngày chức vu, chức vụ);	

Ghép 2 bảng Bộ môn và Trưởng BM

Bảng Bộ môn	
Thuộc tính	TênBM, Văn phòng, Trưởng BM, Ngày nhậm chức, Ngày thôi chức
Ràng buộc & Quy tắc nghiệp vụ	**Phụ thuộc hàm**
Mỗi Bộ môn có 1 Văn phòng	TênBM→ Văn phòng;
Mỗi trưởng BM có 1 ngày nhậm chức và 1 ngày thôi chức Ở mỗi thời điểm thì 1 BM chỉ có 1 trưởng BM	Trưởng BM → Ngày nhậm chức, ngày thôi chức Tên BM, Ngày nhậm chức → Trưởng BM
Chuẩn hóa: Khóa: K1 = (TênBM, Trưởng BM); K2 = (Tên BM, Ngày nhậm chức) Vi phạm chuẩn 2; ➔ Tách thành các quan hệ: **Bộ môn** (Tên BM, Văn phòng); **TrưởngBM** (TrưởngBM, Ngày nhậm chức, Ngày thôi chức); **BM-TrưởngBM** (Tên BM, Ngày nhâm chức, Trưởng môn);	

Các bảng được suy ra

Bảng TKB	
Thuộc tính	Năm học, Học kỳ, Tên Khoa , Tên lớp, Tên phòng, Môn học, Tiết học
Ràng buộc & Quy tắc nghiệp vụ	**Phụ thuộc hàm**
Mỗi lớp học thuộc một khoa	Tên lớp → Tên Khoa;
Mỗi môn học của một lớp trong một năm học và 1 học kỳ thì học ở 1 tiết học phải học ở 1 phòng	Năm học, Học kỳ, Tên lớp, Môn học, Tiết học → Tên phòng;
Chuẩn hóa: Khóa duy nhất: K = (Năm học, Học kỳ, Tên lớp, Tên phòng, Tiết học) Vi phạm chuẩn 2; ➔ Tách thành các quan hệ: **Lớp học** (Tên lớp, Tên khoa); **TKB** (Năm hoc, Hoc kỳ, Tên lớp, Tên phòng, Tiết hoc, Tên phòng);	

Bổ sung các thuộc tính khóa và xác định miền giá trị cho các thuộc tính

Bảng Khoa		
Thuộc tính	**Kiểu dữ liệu**	**Ràng buộc**
ID	**int**	**PK**
TênKhoa	Varchar(200)	Not NULL
ID_VP	int	FK to VănPhòng(ID)

Bảng Văn phòng		
Thuộc tính	**Kiểu dữ liệu**	**Ràng buộc**
ID	**int**	**PK**
Tên VP	Varchar(200)	Not NULL
Fax	Varchar(20)	Dãy các chữ số liên tiếp

Bảng ĐiệnThoạiKhoa		
Thuộc tính	**Kiểu dữ liệu**	**Ràng buộc**
ID	**int**	**PK**
ID_Khoa	Int	FK to Khoa(ID)
ĐiệnThoại	Varchar(20)	Dãy các chữ số liên tiếp

Cảm ơn!

Viện Điện tử - Viễn thông
Bộ Môn Điện tử - Kỹ thuật máy tính

Kỹ thuật phần mềm

Chương 8: Thiết kế phần mềm

Phần 3: Thiết kế kiến trúc

Thiết kế kiến trúc phần mềm

- Khái niệm về kiến trúc phần mềm:
- Các phong cách kiến trúc
- Các phương pháp thiết kế

Kiến trúc phần mềm

- Khái niệm:

  "*Là một cấu trúc bao gồm các thành phần phần mềm, các tính chất có thể thấy được từ bên ngoài của các thành phần này, và các liên kết giữa chúng*" *
- Các thành phần phần mềm có thể gồm:
  - Các module
  - Các cấu trúc dữ liệu, cơ sở dữ liệu

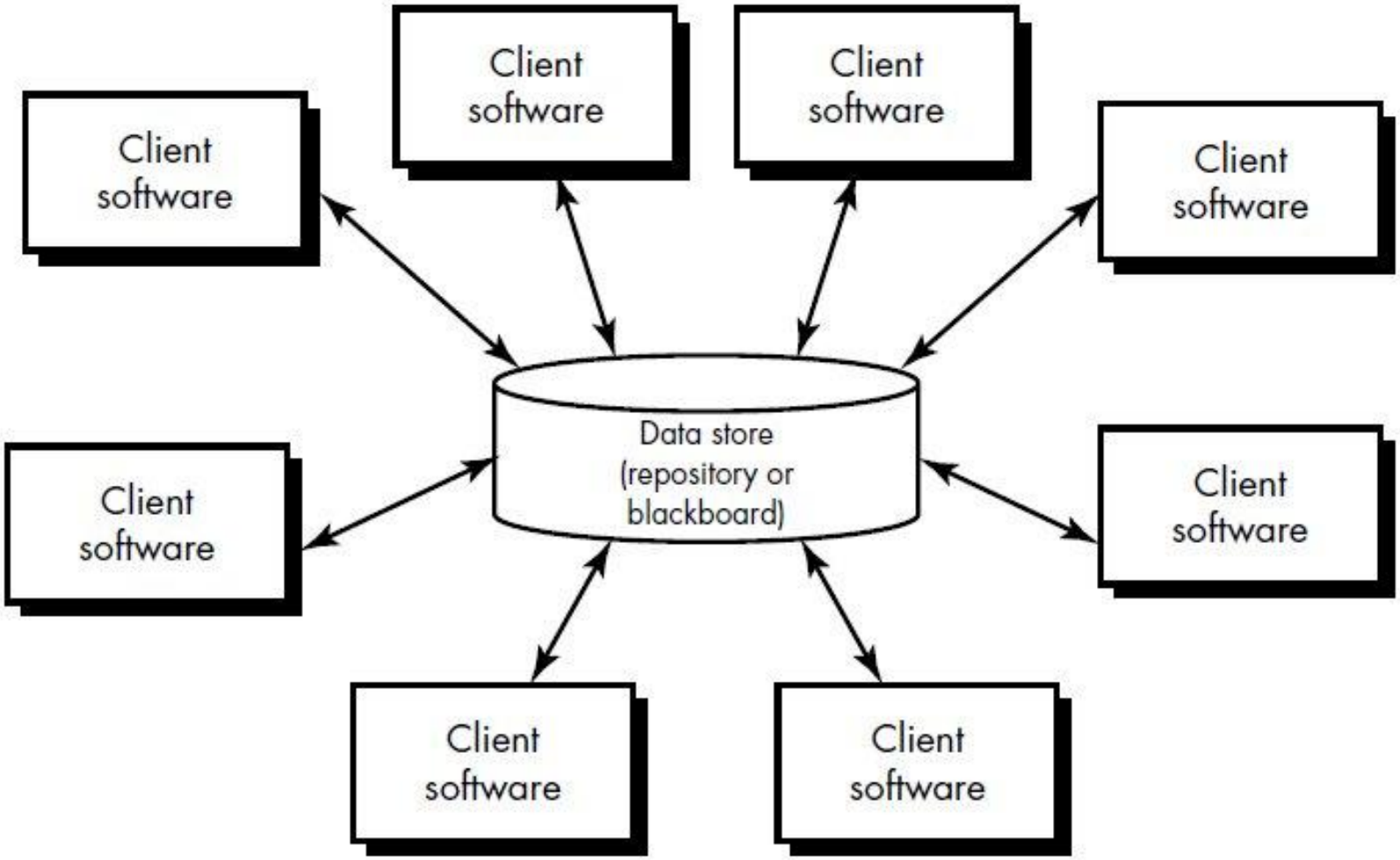
Kiến trúc phần mềm

- Mục đích sử dụng:
  - Để đánh giá tính hiệu quả của phần mềm trong việc đáp ứng các y/c của hệ thống
  - Cân nhắc để chọn ra kiến trúc phù hợp nhất giữa các kiến trúc khác nhau
  - Giúp dự trù sớm và tương đối chính xác các tài nguyên cần chuẩn bị cho giai đoạn cài đặt phần mềm
  - Đóng vai trò như thiết kế tổng thể, làm nền tảng cho các thiết kế chi tiết sau đó
  - Giúp giảm thiểu các rủi ro trong quá trình xây dựng phần mềm sau này

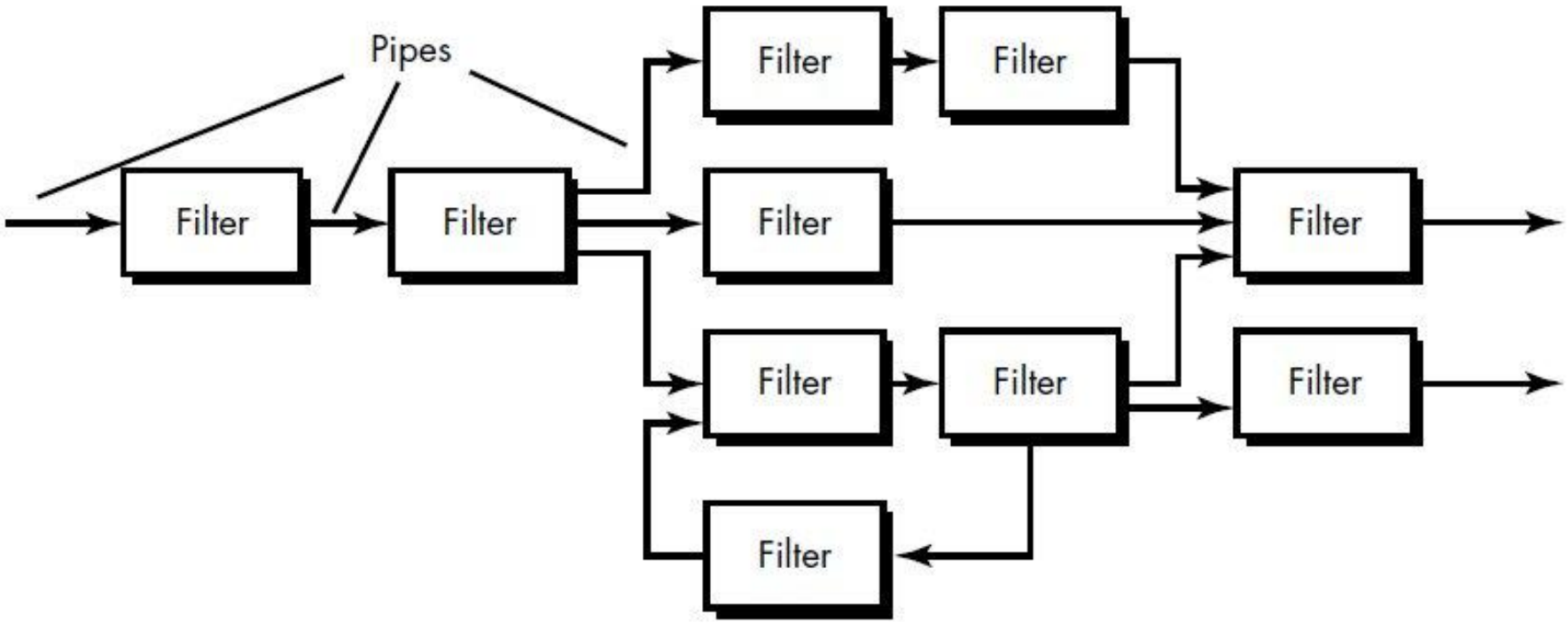
Kiến trúc phần mềm

- Các phong cách kiến trúc
  - Kiến trúc lấy dữ liệu làm trung tâm (data-centered architectures)
  - Kiến trúc luồng dữ liệu (data flow architectures)
  - Kiến trúc gọi và trả về (call and return architectures)
    - Main module/sub module
    - Remote procedure call

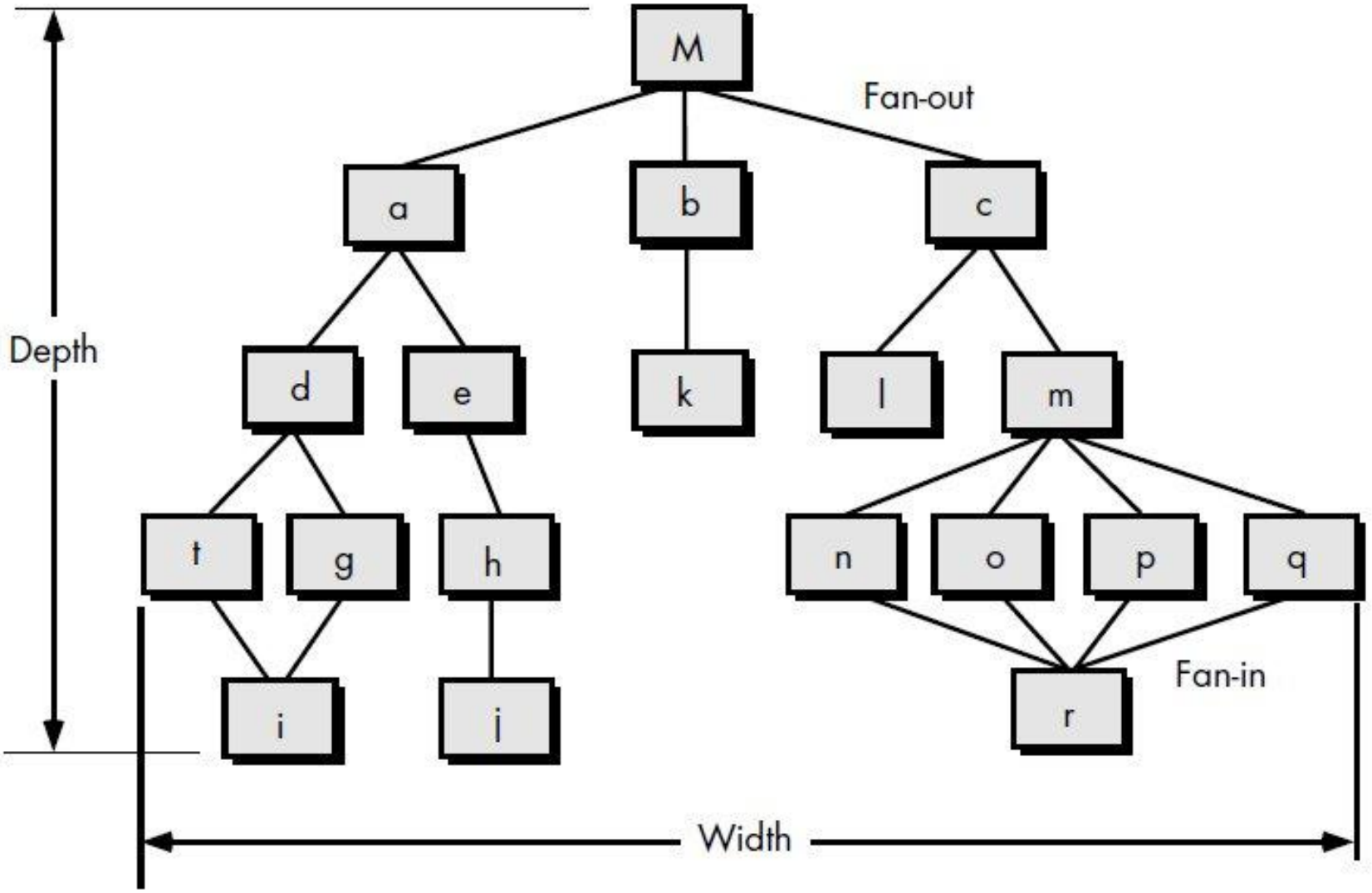
Kiến trúc lấy dữ liệu làm trung tâm



Kiến trúc luồng dữ liệu



Kiến trúc gọi và trả về



Kiến trúc gọi và trả về

- Kiến trúc này còn được gọi là "*cấu trúc chương trình*", hay "*phân cấp điều khiển*"
- Một số khái niệm liên quan:
  - **Fan-in**: của một module là độ đo số lượng module khác mà điều khiển/gọi module đó
  - **Fan-out**: của một module là độ đo số lượng module mà module đó điều khiển/gọi
  - **Chiều sâu** (depth): xác định số mức điều khiển/gọi
  - **Độ rộng** (width): xác định phạm vi điều khiển/gọi

Thiết kế kiến trúc phần mềm

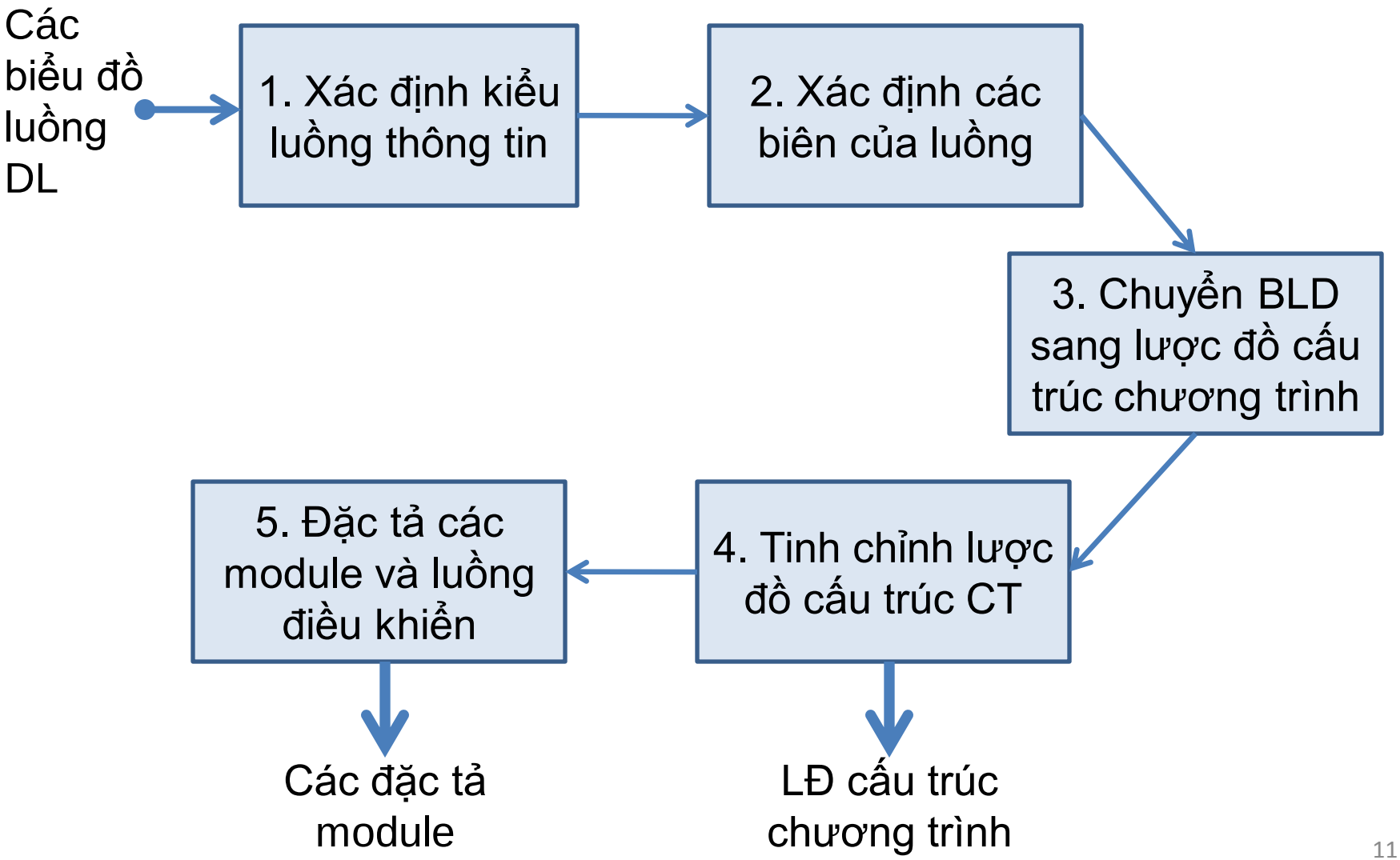
- Phương pháp được sử dụng:

  "***Thiết kế có cấu trúc***" (structured design)

- Đặc điểm của phương pháp:

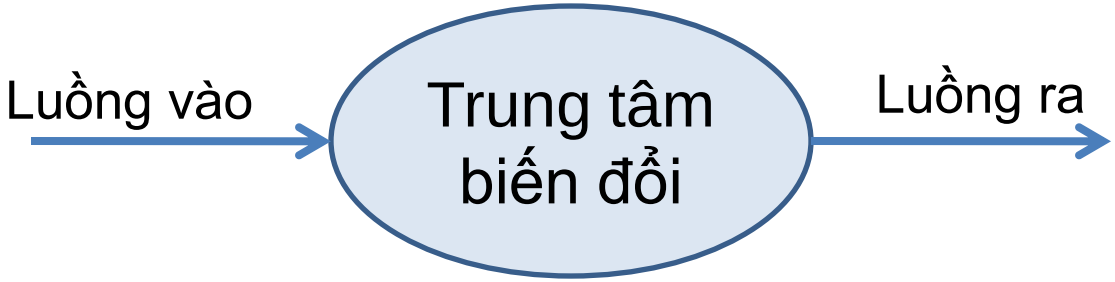
  Có hướng luồng dữ liệu, cung cấp cách thuận tiện để chuyển từ các biểu đồ luồng dữ liệu sang mô hình kiến trúc phần mềm

Thiết kế có cấu trúc



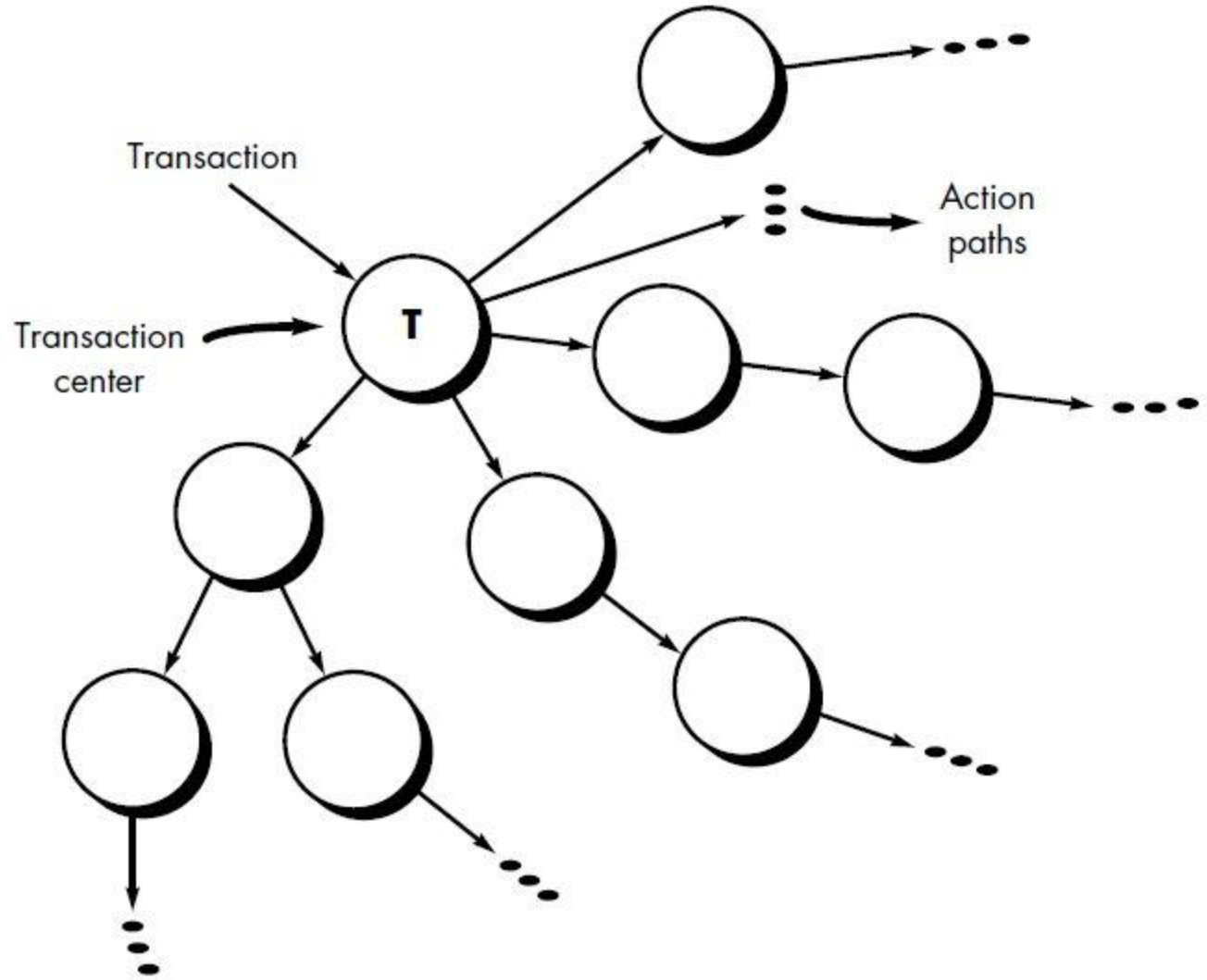
Các kiểu luồng thông tin

- Luồng biến đổi (transform flow)



- Luồng giao tác (transaction flow)
  - Là loại luồng biến đổi đặc biệt, trong đó có một luồng vào và có nhiều luồng ra

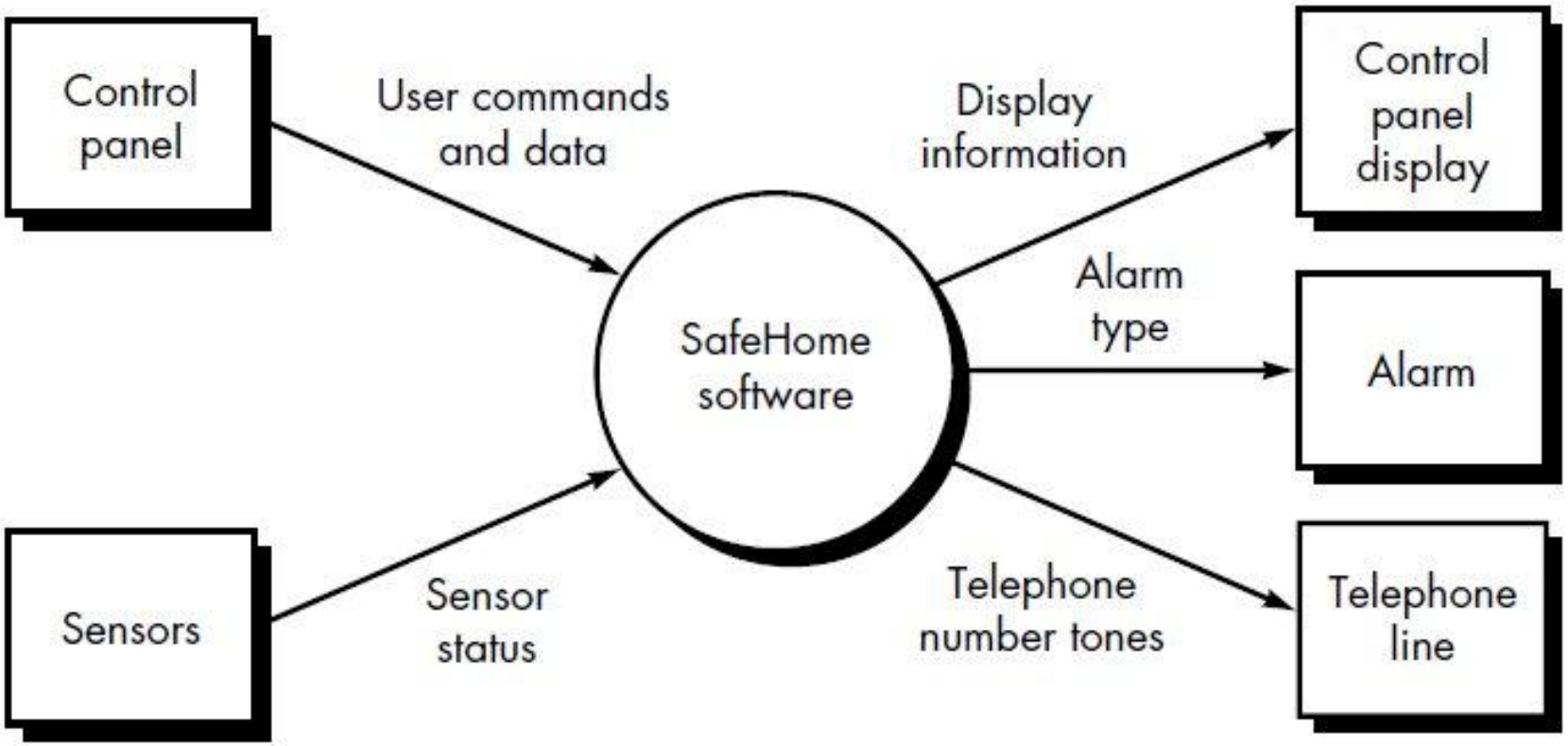
Luồng giao tác



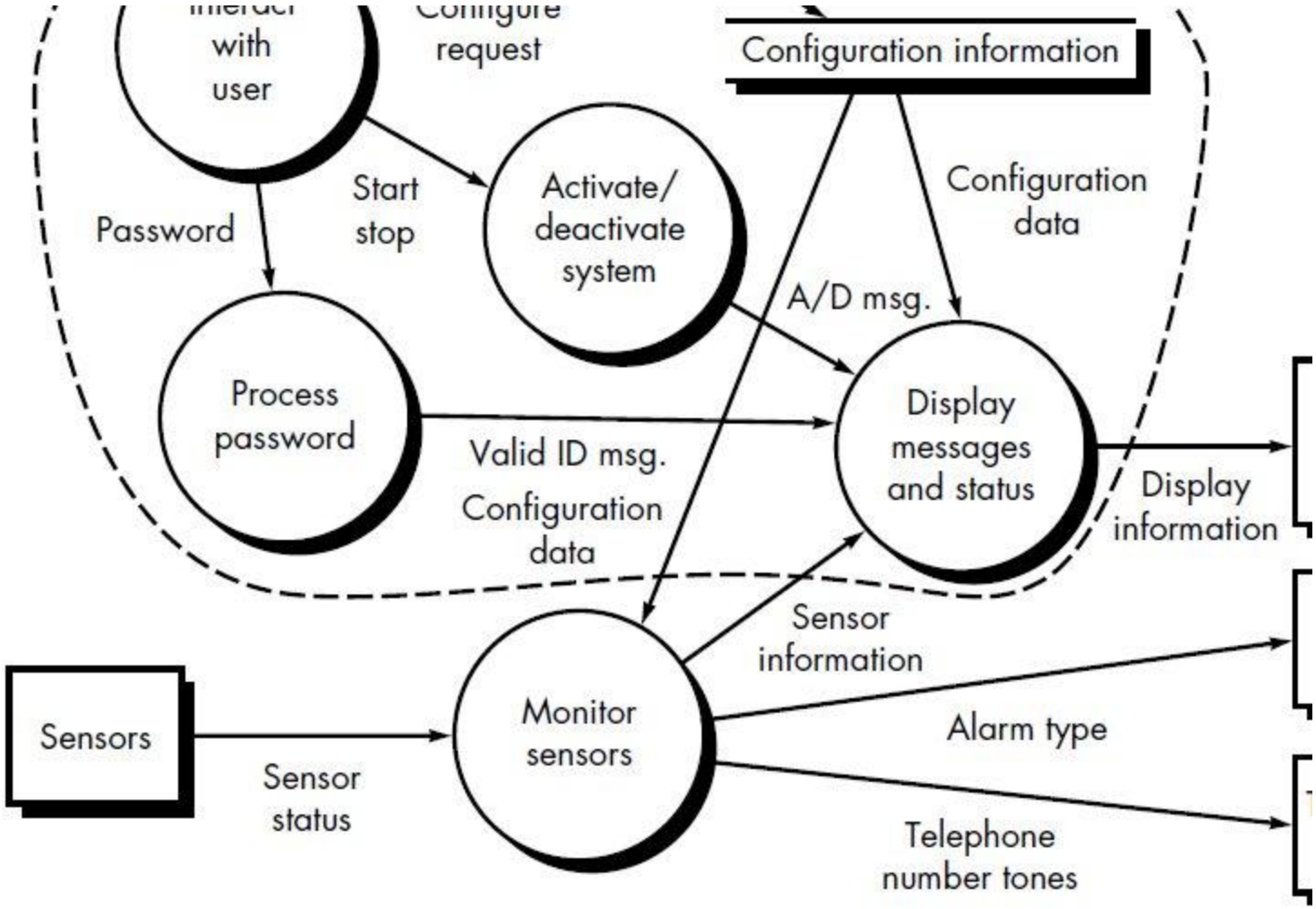
Chuyển luồng biến đổi

- Bước 1: Làm mịn các biểu đồ luồng dữ liệu đến mức cần thiết
- Bước 2: Xác định các biên của luồng dữ liệu để xác định trung tâm biến đổi và các luồng vào và ra
- Bước 3: Chuyển đổi BLD sang lược đồ cấu trúc
- Bước 4: Tinh chỉnh LĐCT

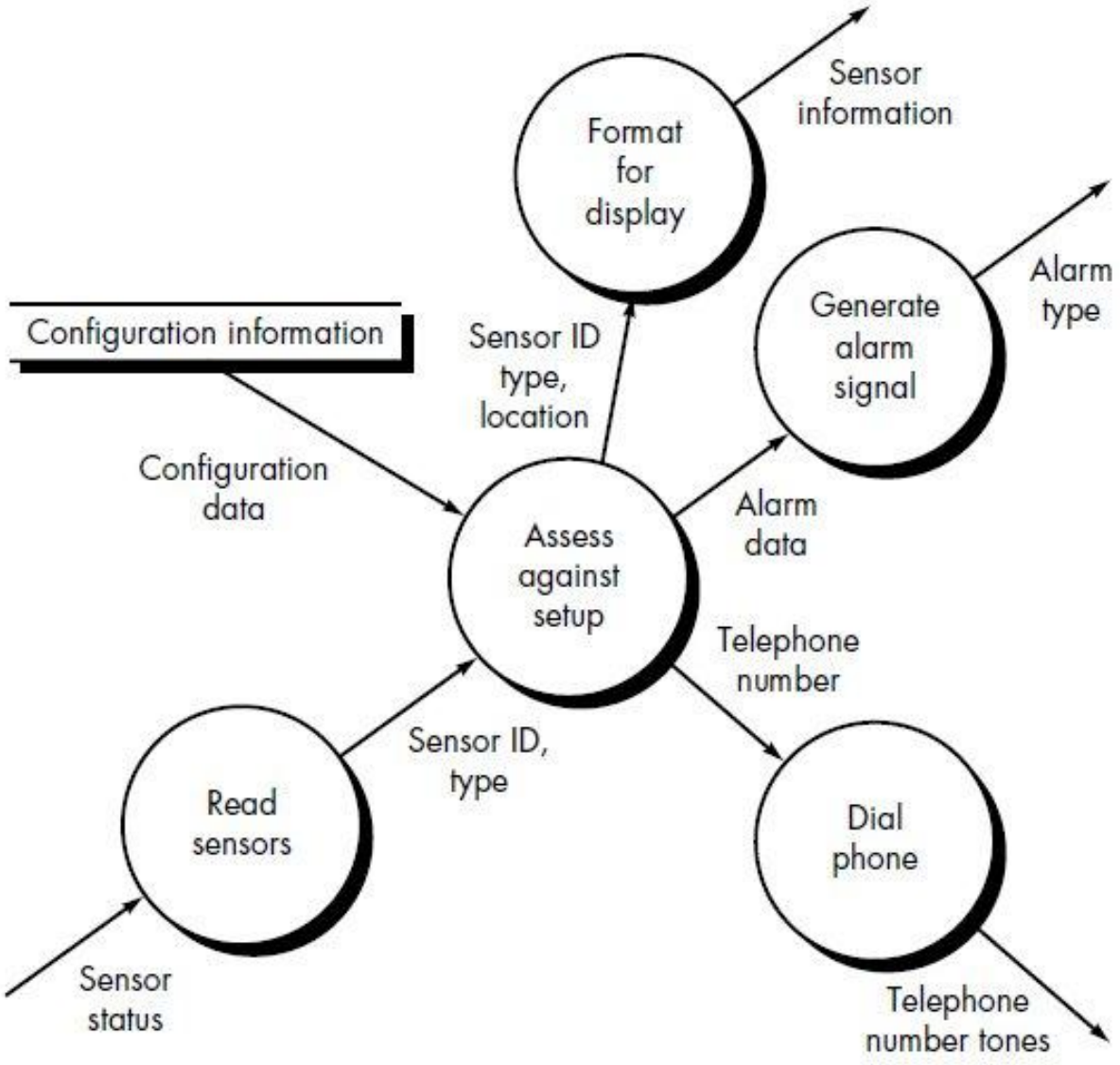
Chuyển luồng biến đổi – Bước 1



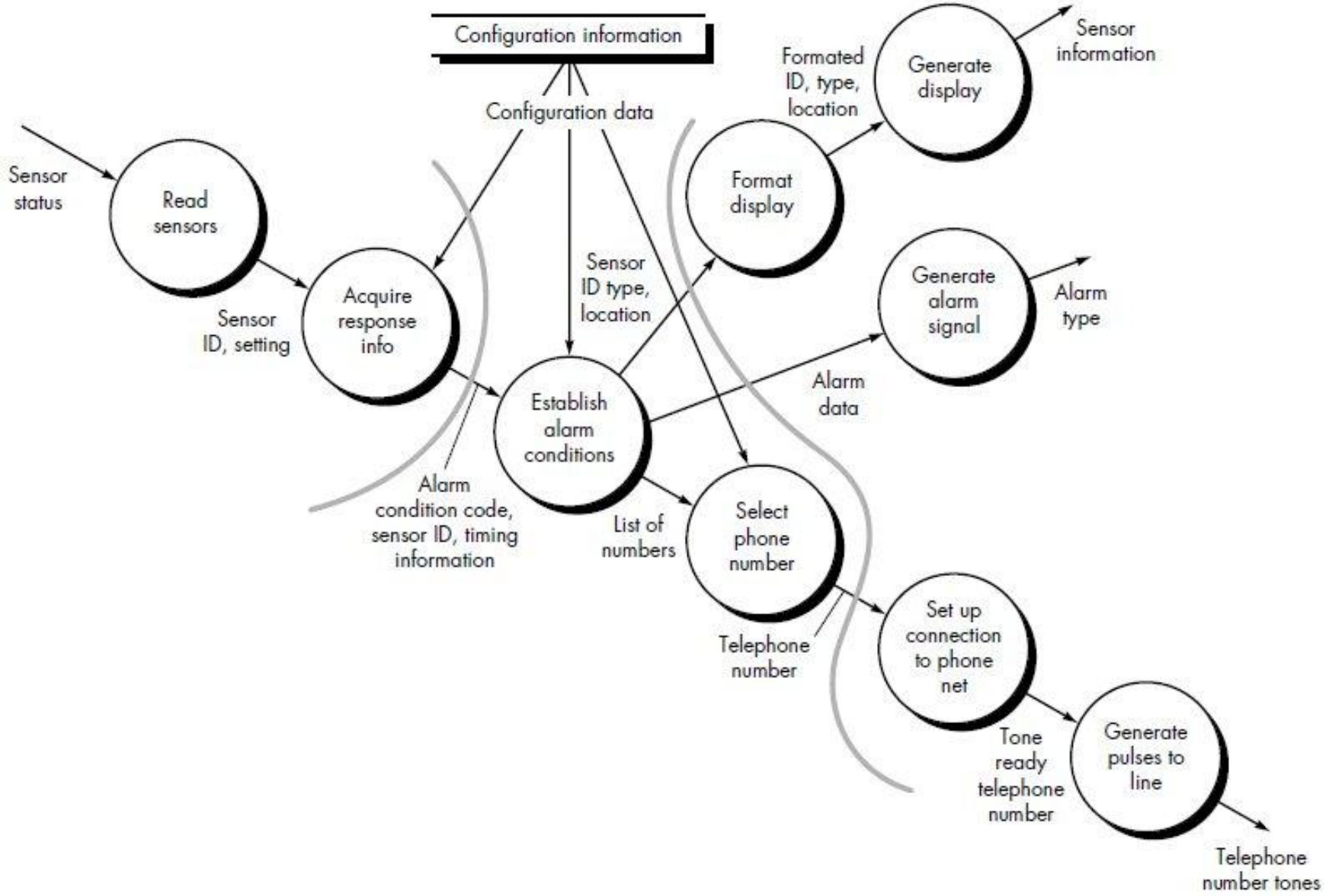
Chuyển luồng biến đổi – Bước 1



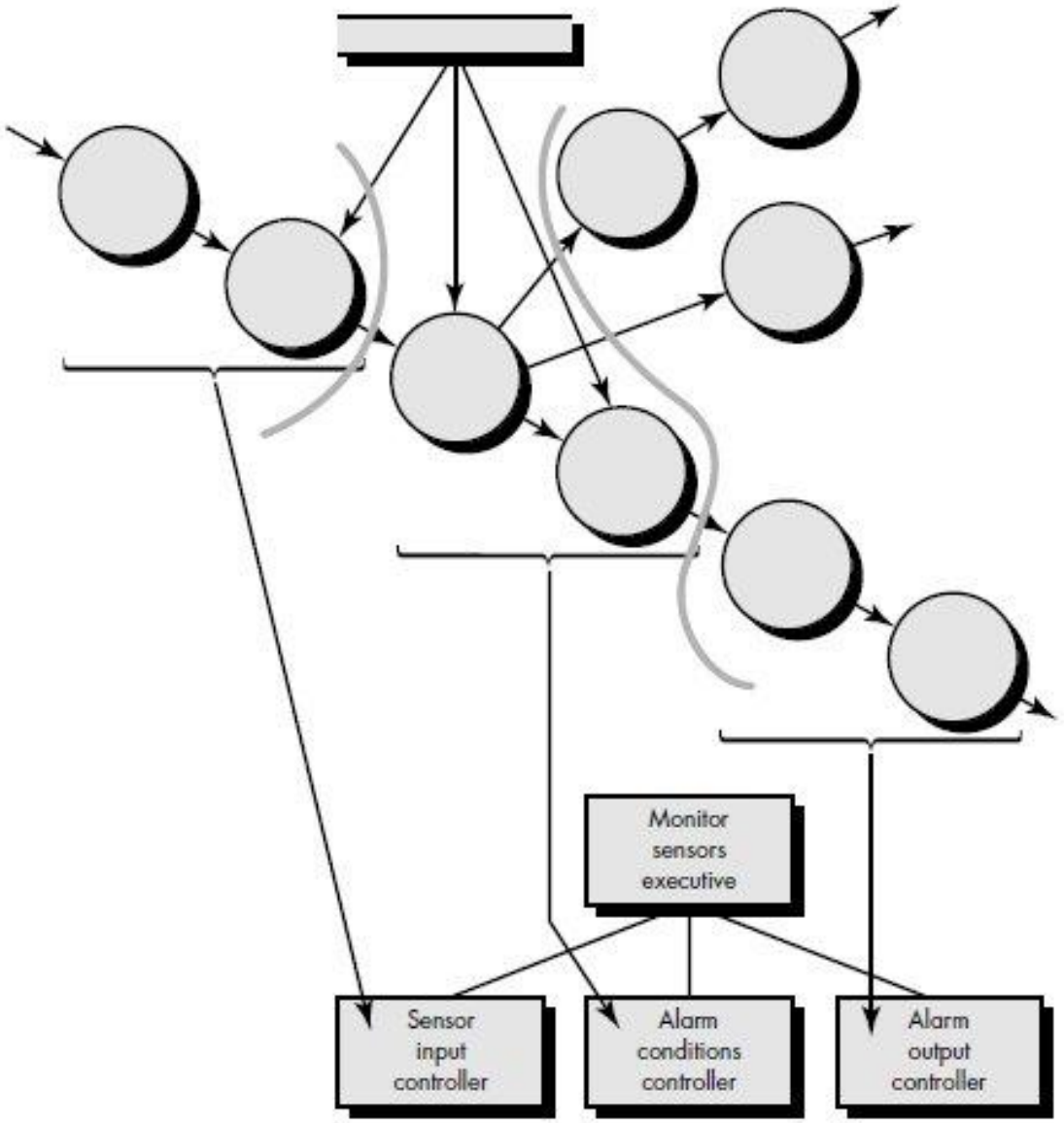
Chuyển luồng biến đổi – Bước 1

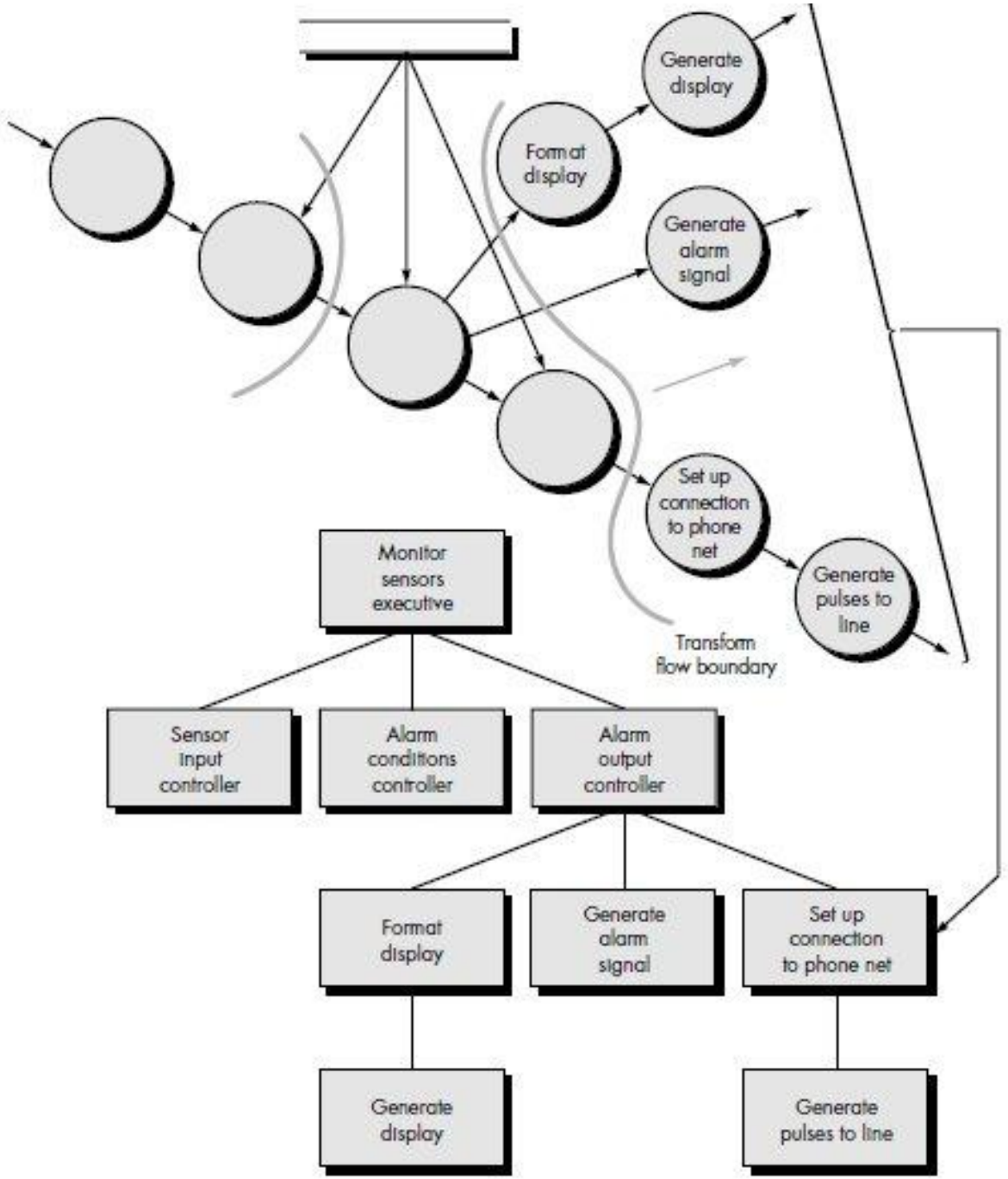


Chuyển luồng biến đổi – Bước 2

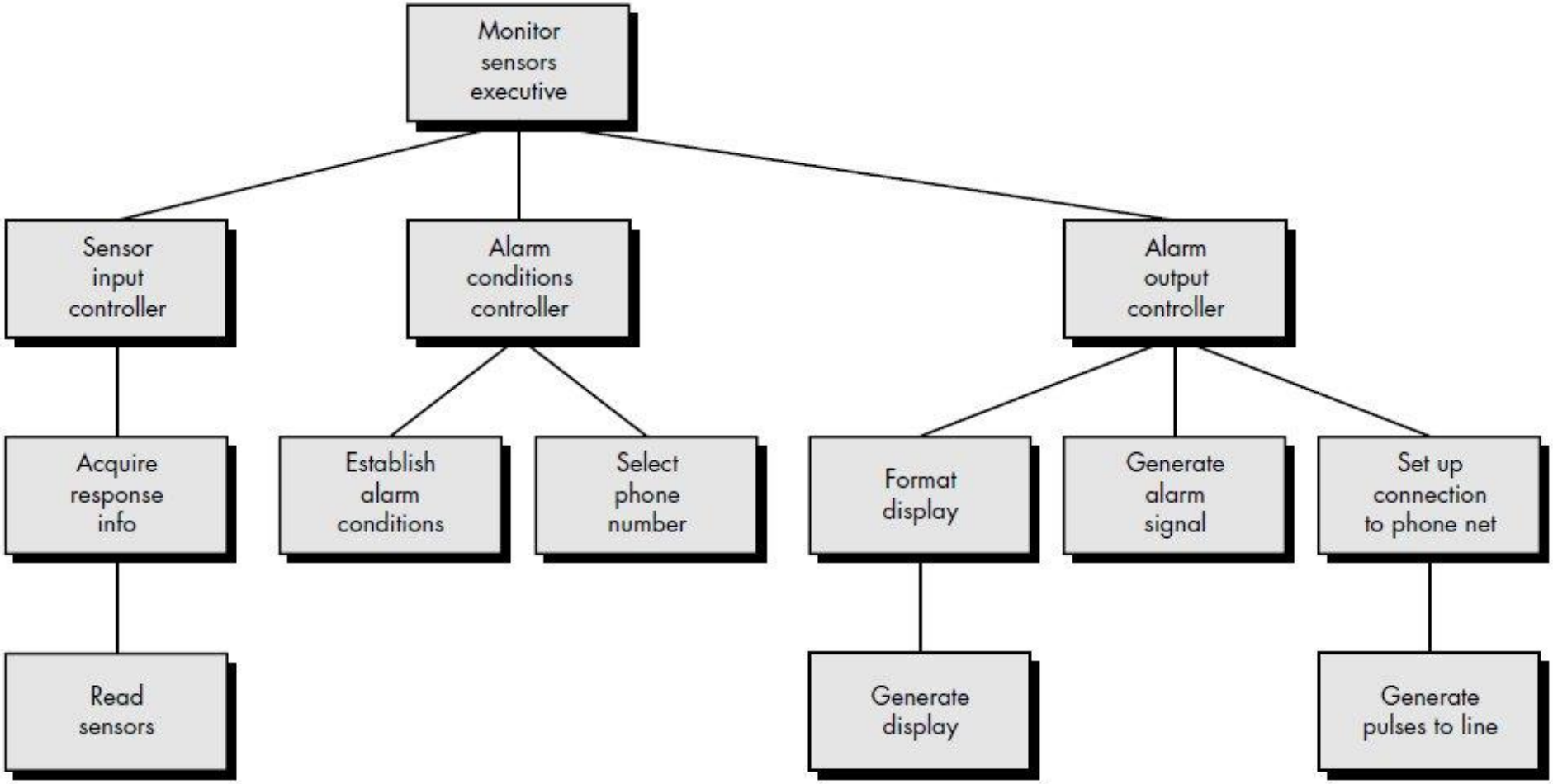


Chuyển luồng biến đổi – Bước 3

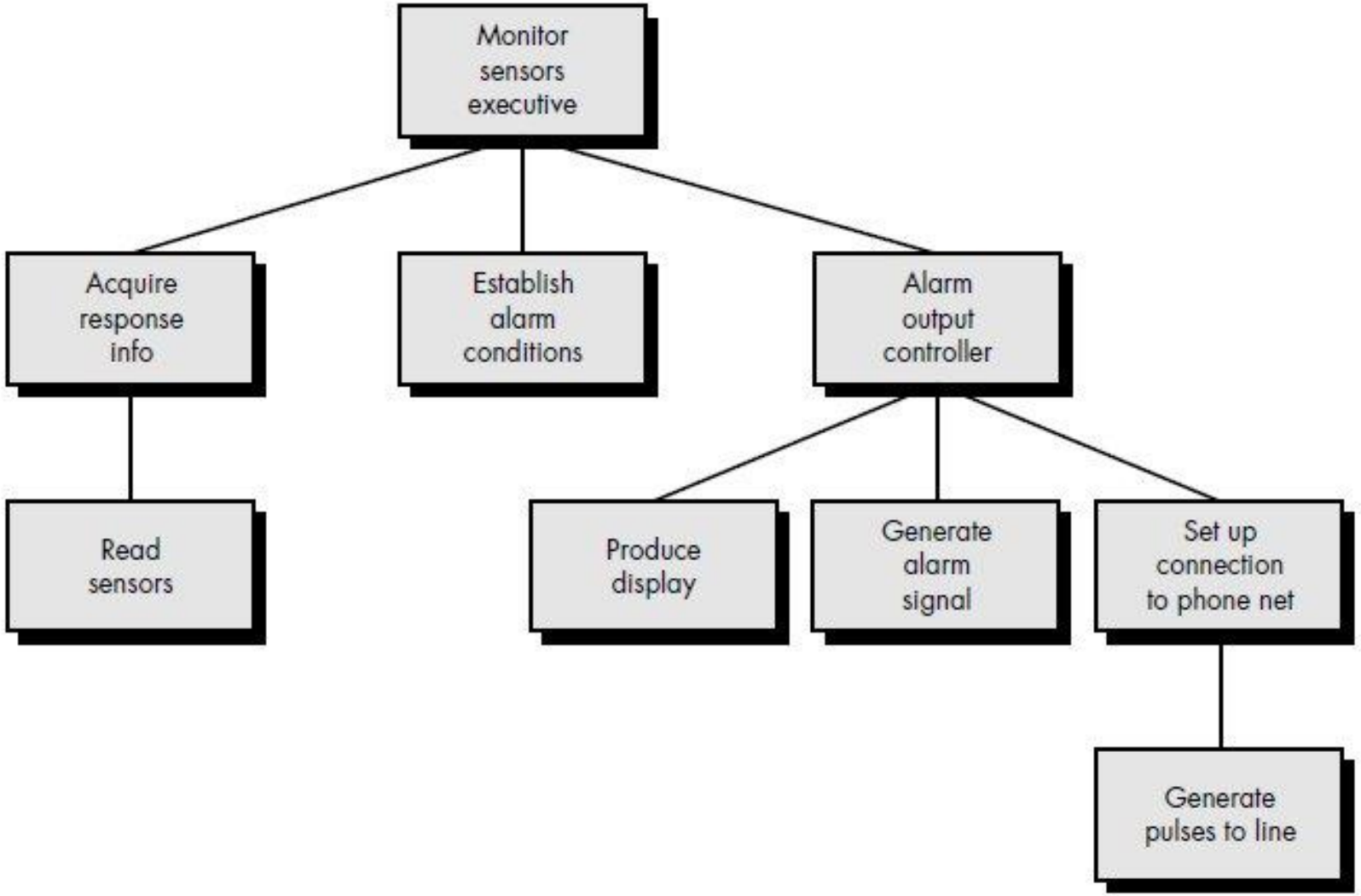



Generate display
Format display
Generate alarm signal
Set up connection to phone net
Generate pulses to line
Transform flow boundary
Monitor sensors executive
Sensor input controller
Alarm conditions controller
Alarm output controller
Format display
Generate alarm signal
Set up connection to phone net
Generate display
Generate pulses to line

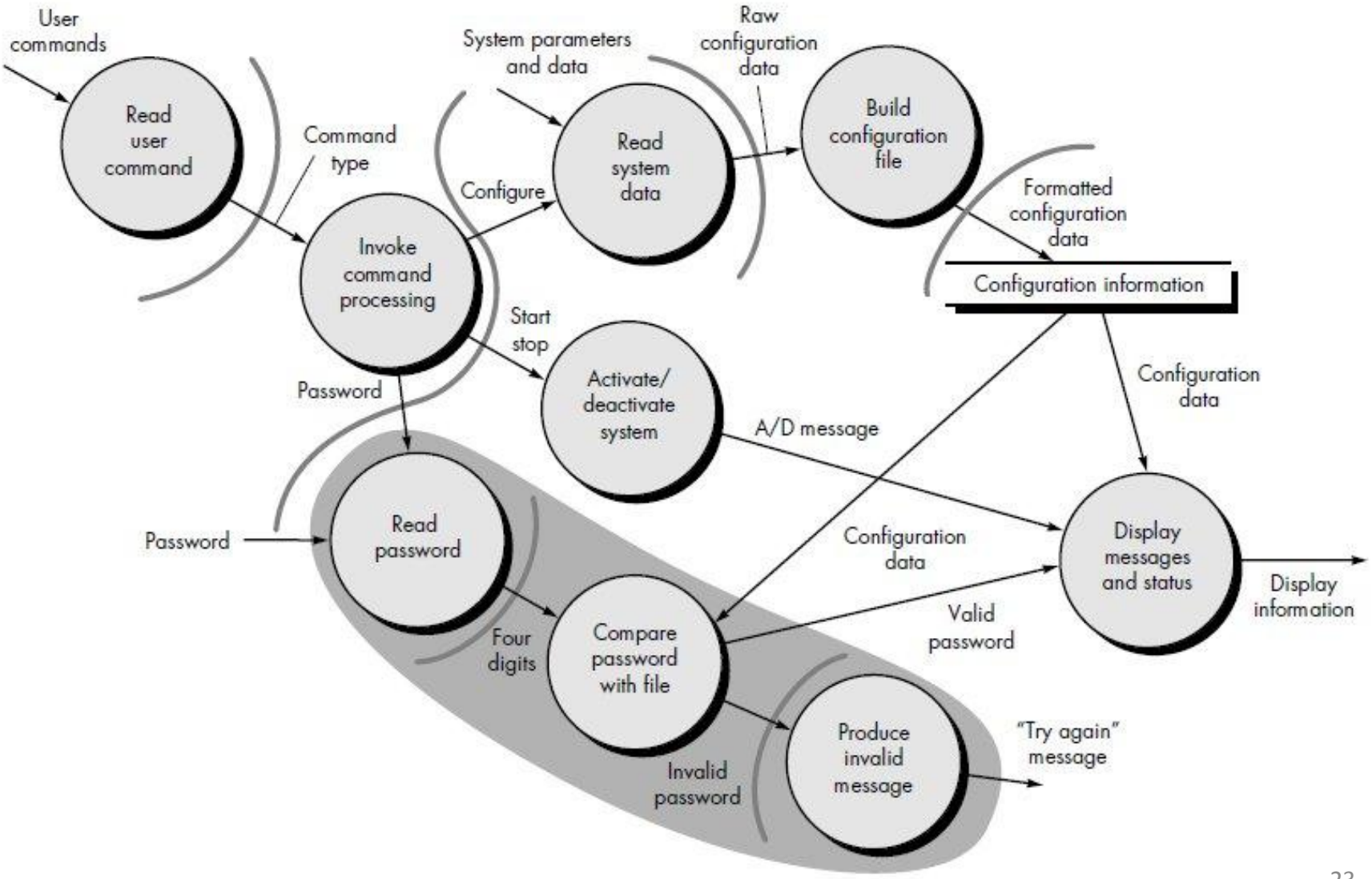
Chuyển luồng biến đổi – Bước 4



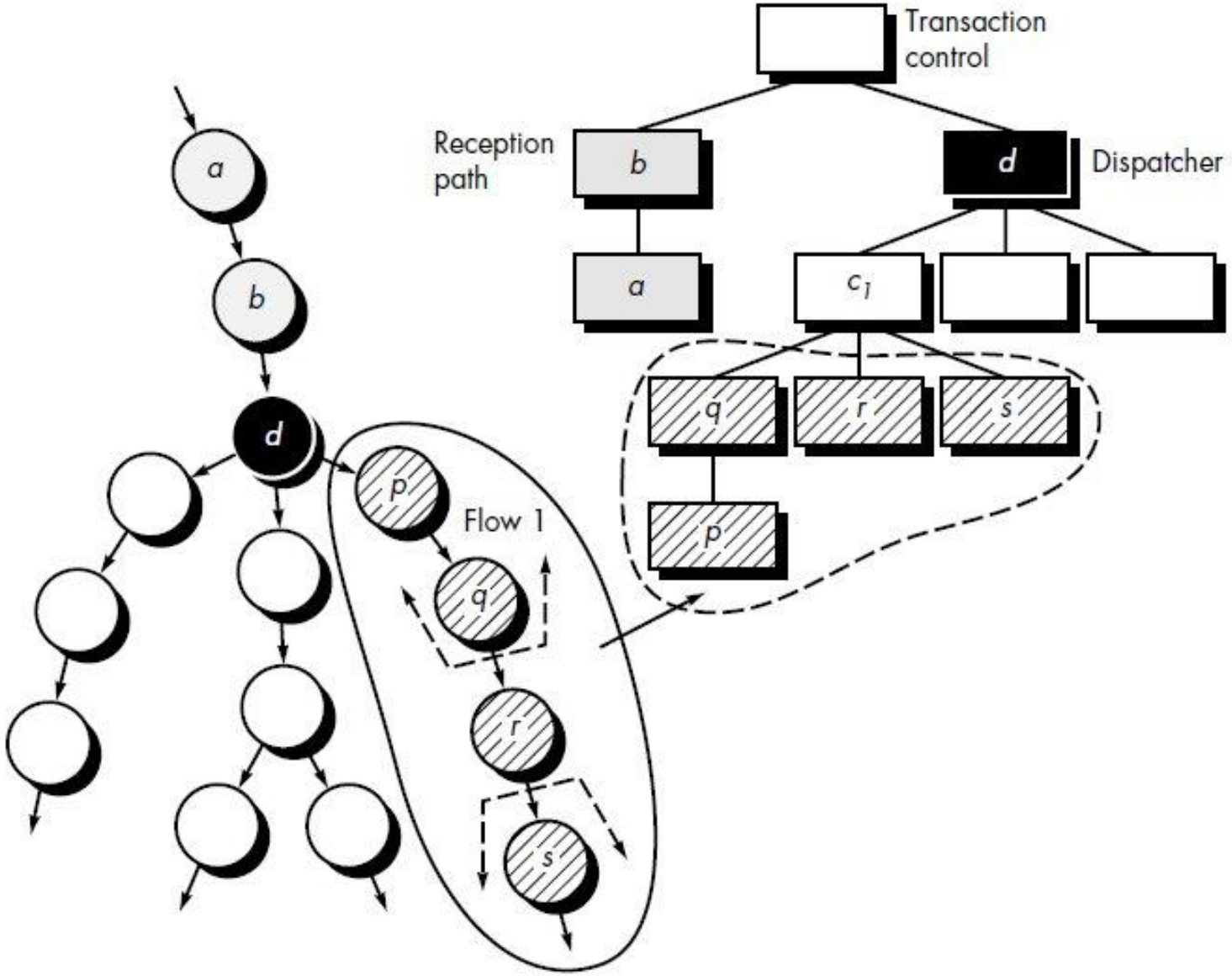
Chuyển luồng biến đổi – Bước 4



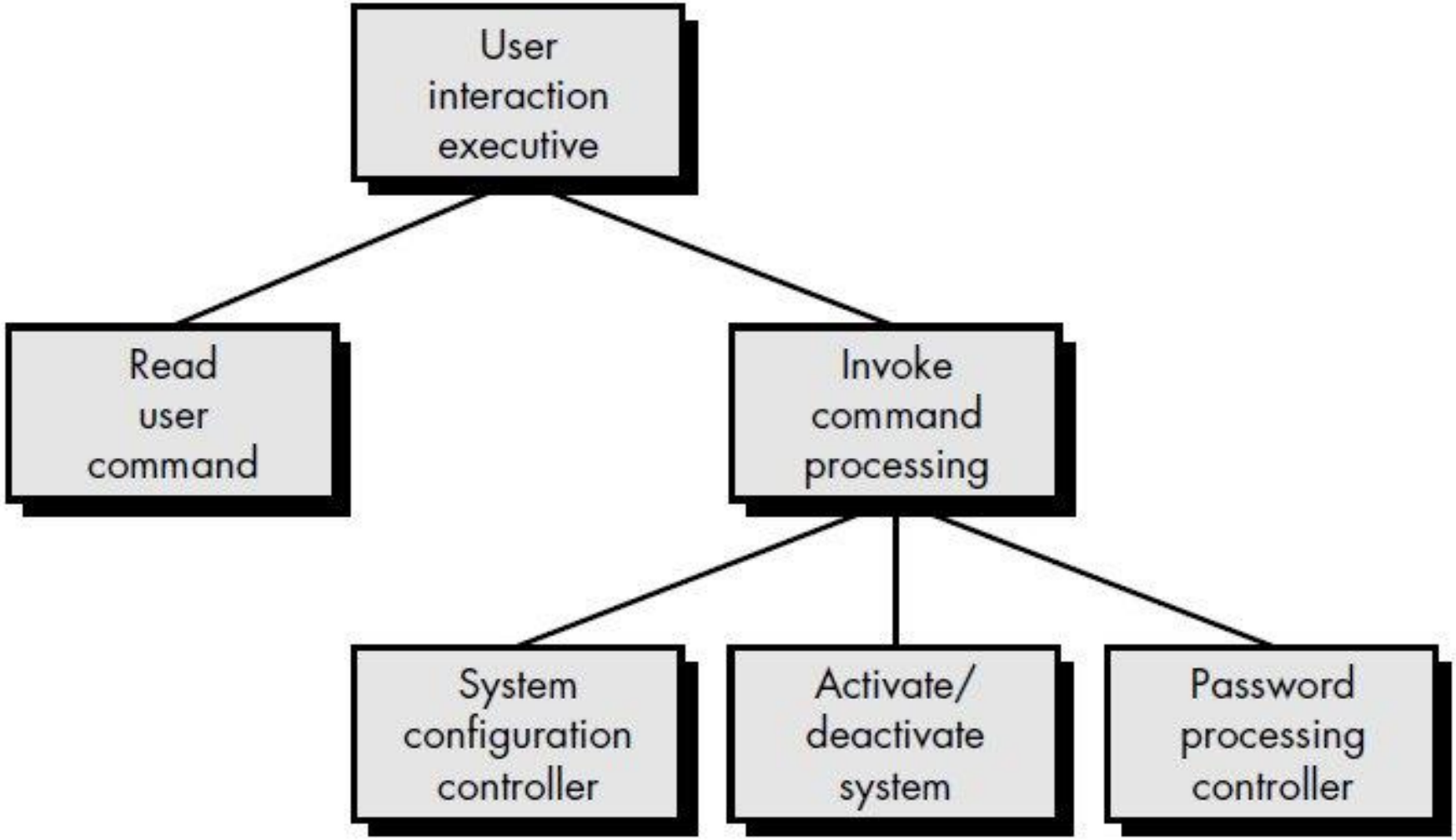
Chuyển luồng giao tác – Bước 2



Chuyển luồng giao tác – Bước 3



Chuyển luồng giao tác – Bước 3



Tóm tắt

- Kiến trúc phần mềm và vai trò của việc thiết kế KTPM
- Các phong cách kiến trúc
- Phương pháp thiết kế có cấu trúc
- Các loại luồng dữ liệu
  - Luồng biến đổi
  - Luồng giao tác
- Cách chuyển đổi từ BLD sang lược đồ cấu trúc chương trình

Cảm ơn!

Viện Điện tử - Viễn thông
Bộ Môn Điện tử - Kỹ thuật máy tính

Kỹ thuật phần mềm

Chương 8: Thiết kế phần mềm

Phần 4: Thiết kế giao diện

Các nội dung chính

- Các loại giao diện
- Tầm quan trọng của giao diện
- Các quy tắc thiết kế giao diện
- Các bước thiết kế

Các loại giao diện

- Giao diện giữa các module chương trình
- Giao diện giữa các modul và các thiết bị/hệ thống bên ngoài
- **Giao diện người máy**: giao diện giữa người sử dụng và phần mềm

Tầm quan trọng của giao diện

- **Là bộ mặt của phần mềm**: cần rõ ràng, sáng sủa, thân thiện.
- **Là nơi cung cấp các chức năng cho người dùng**: cần đầy đủ các chức năng, dễ sử dụng và sử dụng an toàn.
- **Là nơi cung cấp các trợ giúp**: cần hiểu được các nhu cầu trợ giúp và giúp đỡ kịp thời và hiệu quả.
- **Là công cụ sử dụng hàng ngày**: nên cần khả năng tùy biến và linh hoạt để tránh nhàm chán và sử dụng ngày càng hiệu quả.

Các quy tắc thiết kế

- **Các quy tắc vàng**:
  1. Luôn đặt người dùng vào vị trí điều khiển
  2. Giảm thiểu gánh nặng ghi nhớ của người dùng
  3. Tạo giao diện nhất quán

Quy tắc 1: Luôn đặt người dùng vào vị trí điều khiển

- Ý nghĩa:
  - Giao diện cần phải giúp người dùng luôn duy trì quyền điều khiển chương trình, chứ không phải bị điều khiển bởi chương trình
  - Việc thiết kế giao diện cần đứng từ góc độ người dùng và vì người dùng, chứ không chỉ ở góc độ của người phát triển hệ thống dùng chương trình.

Quy tắc 1: Luôn đặt người dùng vào vị trí điều khiển

- Các nguyên tắc cho phép cụ thể hóa quy tắc 1:
  - Xác định các chế độ tương tác (interaction modes) phù hợp, sao cho chúng không bắt người dùng phải thực hiện các hành động không mong muốn hay không cần thiết
  - Cung cấp nhiều loại tương tác linh hoạt: cho phép người dùng có nhiều lựa chọn loại hình tương tác, như bàn phím, chuột, cảm ứng, giọng nói,v.v.
  - Cho phép tương tác có thể bị ngắt và làm lại (undo)
  - Cho phép giao diện có thể tùy biến và tiến hóa theo sở thích và kinh nghiệm của người dùng
  - Che dấu các chi tiết kỹ thuật bên trong không cần thiết khỏi người dùng thông thường

Quy tắc 2: Giảm thiểu gánh nặng ghi nhớ của người dùng

- Ý nghĩa:
  - Quy tắc này giúp người dùng sử dụng hệ thống cảm thấy thoải mái hơn, chính xác hơn, giảm thiểu công sức phải nhớ, cũng như các lỗi do việc nhớ không chính xác

Quy tắc 2: Giảm thiểu gánh nặng ghi nhớ của người dùng

- Các nguyên tắc cụ thể:
  - Giảm thiểu yêu cầu ghi nhớ ngắn hạn: như các hành động đã thực hiện, các dữ liệu đã nhập, các cửa sổ trước,v.v
  - Thiết lập các giá trị mặc định có ý nghĩa
  - Xác định các shortcut trực quan và dễ nhớ
  - Hé mở thông tin theo một cách dần dần

Quy tắc 3:
Tạo giao diện nhất quán

- Ý nghĩa:
  - Các giao diện của một hệ thống cần phải theo một phong cách thống nhất nào đó, như các form nhập liệu theo một cách bố trí nhất định, các báo cáo có cùng định dạng nhất định, các cửa sổ có cùng phong cách nhất định
  - Tính nhất quán của giao diện sẽ làm người dùng cảm thấy thân thiện hơn, dễ sử dụng hơn, thích nghi nhanh hơn. Đồng thời nó cũng giảm thiểu các sai sót, thời gian làm quen của người dùng với hệ thống

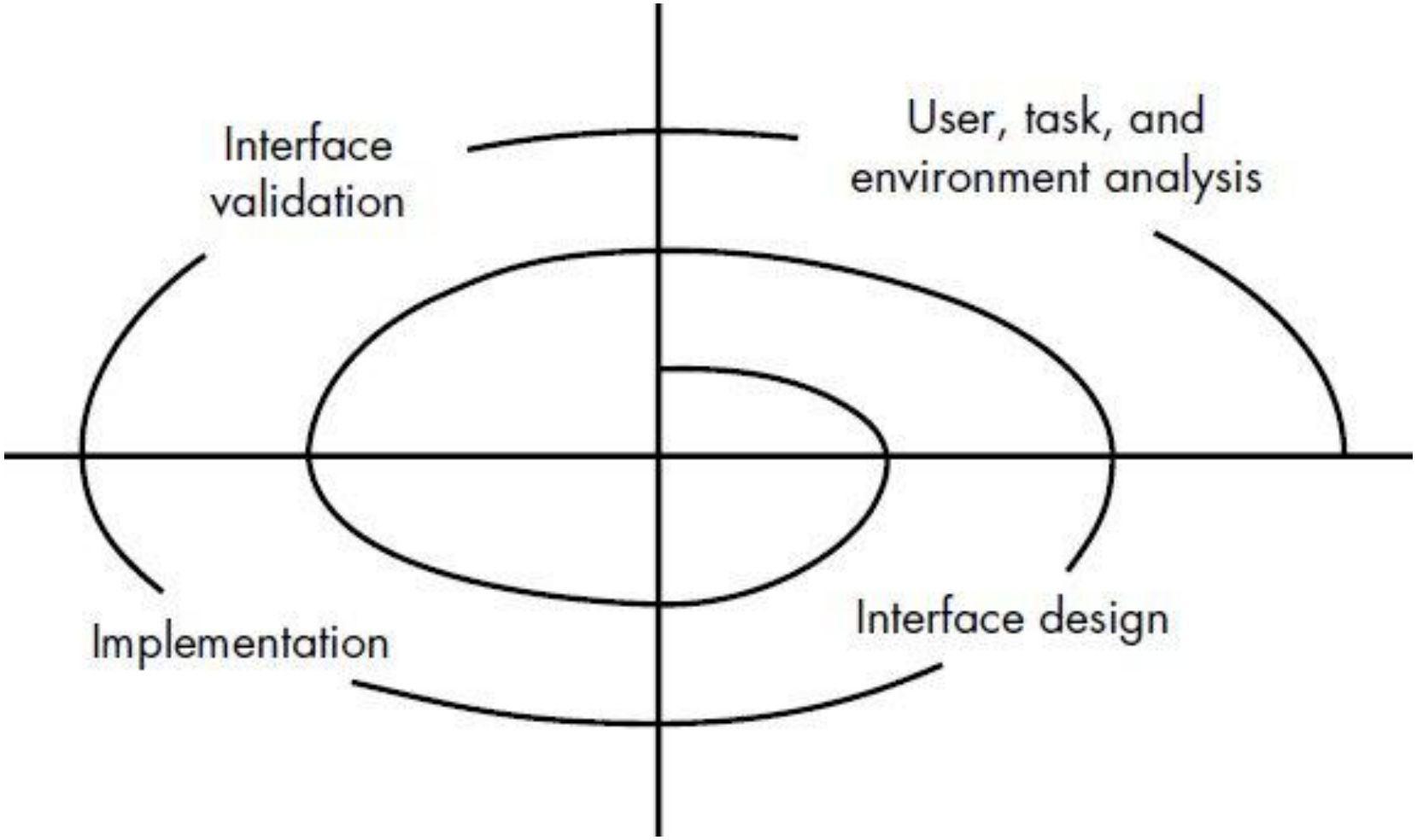
Quy tắc 3:
Tạo giao diện nhất quán

- Các nguyên tắc cụ thể:
  - Cho phép người dùng đặt công việc hiện tại trong khung cảnh có ý nghĩa: giao diện luôn cho phép NSD biết họ đang ở đâu và đang làm gì, cái gì đã làm, cái gì sẽ làm tiếp theo
  - Duy trì sự nhất quán trong một họ các ứng dụng
  - Hạn chế thay đổi các phong cách giao diện mà đã tạo được ấn tượng tốt với người dùng. Chỉ nên thay đổi khi có lý do chính đáng

Các bước thiết kế giao diện

- Phân tích môi trường, người dùng, các công việc
- Thiết kế giao diện
- Cài đặt giao diện
- Kiểm tra tính hợp lệ của giao diện

Các bước thiết kế giao diện



Các bước thiết kế giao diện

- Tham khảo chi tiết trong sách "***Software Engineering – A practitioner's approach***"

Cảm ơn!

Đại học Bách khoa Hà Nội
Viện Điện tử - Viễn thông

Kỹ thuật phần mềm ứng dụng

Chương 9: Ngôn ngữ SQL

Phần 1: Câu truy vấn đơn

Nội dung chính

- Tổng quan về SQL
- Transact SQL (T-SQL) của Microsoft

Tổng quan về SQL

- **SQL** (viết tắt của "Structured Query Language" – Ngôn ngữ truy vấn có cấu trúc) là tập các lệnh cho phép người dùng và cả các chương trình thực hiện các truy vấn dữ liệu trong cơ sở dữ liệu.

- Về mặt lịch sử, ban đầu nó có tên gọi là ***SEQUEL***, (Structured English Query Language) do ***Donald D. Chamberlin*** và ***Raymond F. Boyce*** tại hãng IBM phát triển vào đầu những năm 70 của thế kỷ trước. Sau này nó mới được đổi tên thành SQL (và vẫn được phát âm là "sequel").

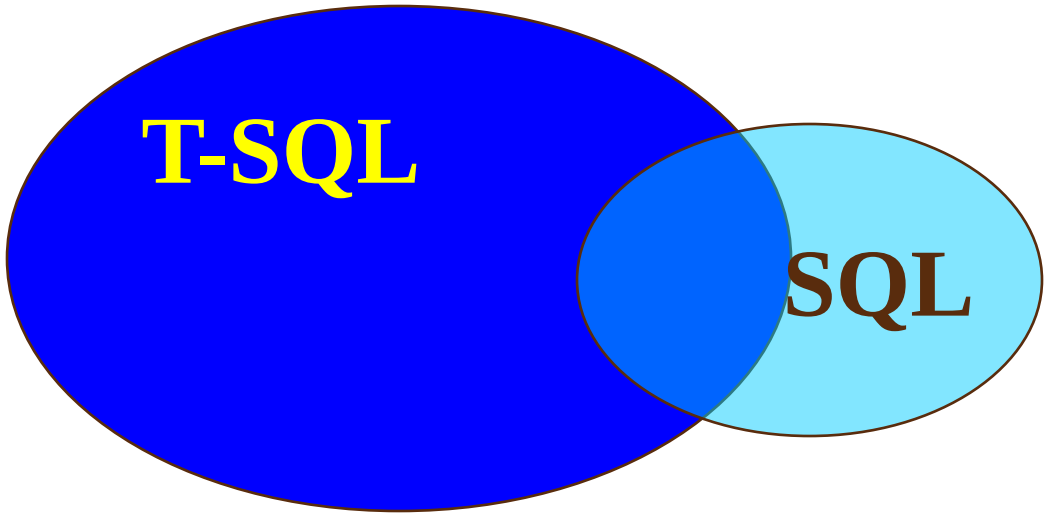
- Ngày nay, nó là ngôn ngữ chuẩn hóa của các hệ quản trị cơ sở dữ liệu quan hệ.

Các phiên bản SQL*

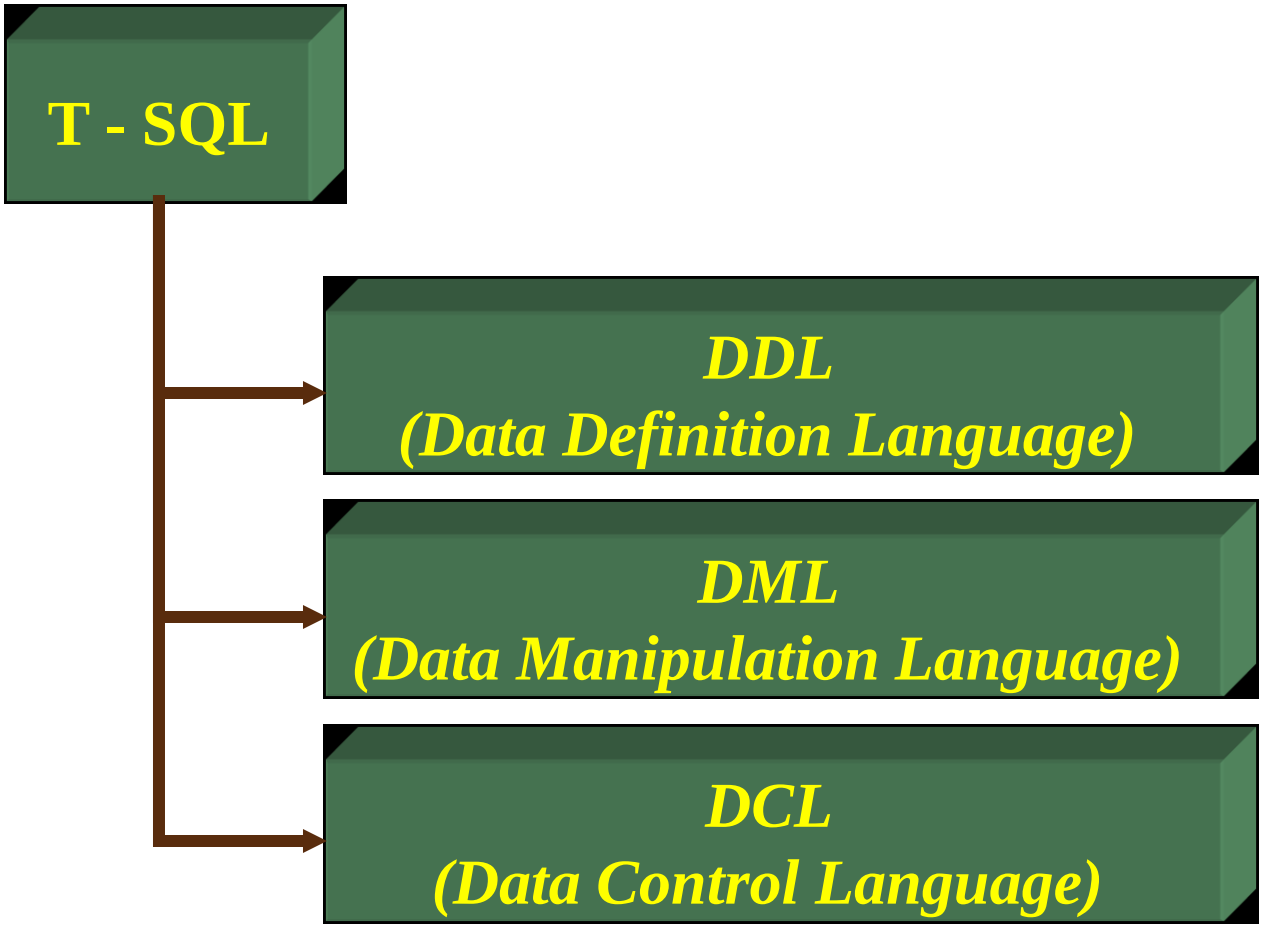
Year	Name	Alias	Comments
1986	**SQL-86**	SQL-87	First published by ANSI. Ratified by ISO in 1987.
1989	**SQL-89**	FIPS 127-1	Minor revision, adopted as FIPS 127-1.
1992	**SQL-92**	SQL2, FIPS 127-2	Major revision (ISO 9075), *Entry Level* SQL-92 adopted as FIPS 127-2.
1999	**SQL:1999**	SQL3	Added regular expression matching, recursive queries, triggers, support for procedural and control-of-flow statements, non-scalar types, and some object-oriented features.
2003	**SQL:2003**		Introduced XML-related features, *window functions*, standardized sequences, and columns with auto-generated values (including identity-columns).
2006	**SQL:2006**		ISO/IEC 9075-14:2006 defines ways in which SQL can be used in conjunction with XML. It defines ways of importing and storing XML data in an SQL database, manipulating it within the database and publishing both XML and conventional SQL-data in XML form. In addition, it provides facilities that permit applications to integrate into their SQL code the use of XQuery, the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents.
2008	**SQL:2008**		Defines more flexible windowing functions, clarifies SQL 2003 items that were still unclear [1]

Transact SQL (T-SQL)

- **Transact-SQL** (**T-SQL**) *: là mở rộng của ngôn ngữ SQL do Microsoft và Sybase phát triển, được sử dụng trong các hệ quản trị CSDL như SQL Server



Các thành phần ngôn ngữ của T-SQL



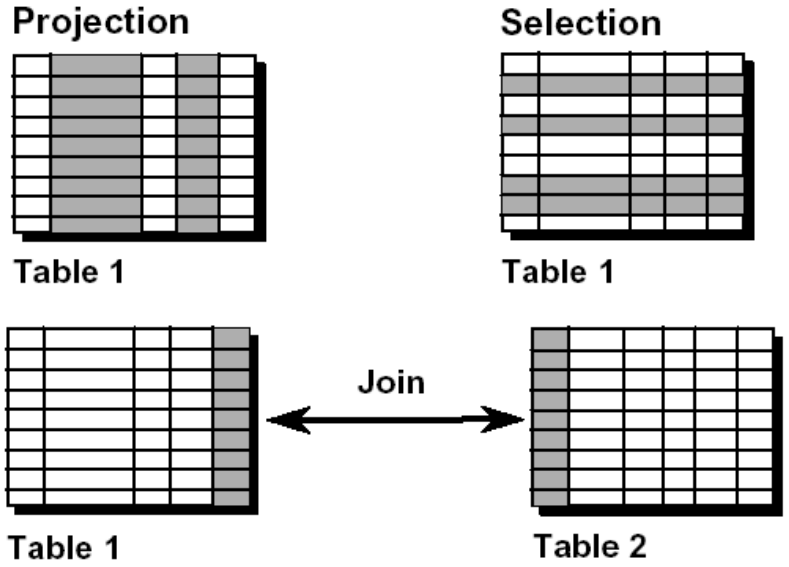
Các thành phần ngôn ngữ của T-SQL

SELECT	Data retrieval
INSERT UPDATE DELETE	Data manipulation language (DML)
CREATE ALTER DROP TRUNCATE	Data definition language (DDL)
COMMIT ROLLBACK	Transaction control
GRANT REVOKE	Data control language (DCL)

Lệnh SELECT

Lệnh SELECT là một lệnh đa năng để truy vấn dữ liệu trong CSDL. Nó cho phép thực hiện tất cả các thao tác cơ bản trong đại số quan hệ như:

- **Chiếu (Projection)**
- **Chọn (Selection)**
- **Nối (Joining)**
- **Hợp (Union)**
- **Trừ (Except)**



Lệnh SQL đơn giản nhất: in ra toàn bộ nội dung của 1 bảng:

```
SELECT *
FROM  table-name;
```

Lệnh SELECT

- Phân loại:
  - **Lệnh đơn**: là câu lệnh mà chỉ truy vấn thông tin từ 1 bảng
  - **Lệnh phức**: là câu lệnh truy vấn thông tin từ nhiều bảng
  - **Lệnh truy vấn con** (sub-query, hay còn gọi là lệnh SELECT lồng nhau): là câu lệnh SELECT mà bên trong nó cũng lại chứa 1 hay nhiều câu lệnh SELECT khác

Ghi chú

- Mỗi một lệnh (***statement***) trong SQL bao gồm một số mệnh đề (clause)
  Ví dụ:

  SELECT *

  FROM employees

  là một câu lệnh gồm có 2 mệnh đề
- Cú pháp trong SQL KHÔNG phân biệt chữ hoa với chữ thường
- Với các từ khóa nên viết hoa để dễ phân biệt với các từ khác

Lệnh đơn

1. Phép chiếu trong SQL
2. Phép chọn trong SQL
3. Đối sánh mẫu (Pattern matching) trong SQL
4. Giá trị NULL và ‘Unknown’
5. Sắp xếp đầu ra

Cú pháp lệnh SELECT

SELECT [ALL | DISTINCT]
    * | {column_name | expression [alias],…}
FROM table

- SELECT xác định các thuộc tính (cột) cần xuất ra
  - ALL: là lựa chọn mặc định, cho phép các hàng có giá trị trùng nhau cũng được xuất ra
  - DISTINCT: các hàng có giá trị trùng nhau chỉ được xuất ra 1 lần
- FROM xác định một hay nhiều bảng chứa các thông tin cần tìm

Ví dụ: SELECT tất cả các cột

```
SELECT  *
FROM    PC
```

model	speed	hdd	screen	price
1003	2	250	14	1000
1004	2	500	18	900
1005	2.5	500	18	800
1006	2.5	500	18	800

Phép chiếu trong SQL

- Trong mệnh đề SELECT, thay vì sử dụng "*" để liệt kê toàn bộ các thuộc tính, ta có thể liệt kê từng thuộc tính mà muốn xuất ra.
- VD:

```
SELECT  model, speed, price
FROM    PC
```

model	speed	price
1003	2	1000
1004	2	900
1005	2.5	800
1006	2.5	800

Có thể mở rộng phép chiếu sử dụng bí danh và biểu thức

```
SELECT  model, price [price in USD],
        price*20000 [price in VND]
FROM    PC
```

model	price in USD	price in VND
1003	1000	20000000
1004	900	18000000
1005	800	16000000
1006	800	16000000

Bí danh (Alias)

- Là biện pháp cho phép đổi tên các thuộc tính (cột), hay tên các bảng trong câu lệnh SELECT
- Nó có thể đi kèm với từ khóa AS (không bắt buộc)
- Trong trường hợp bí danh có khoảng trắng thì cần đặt nó trong cặp **“bí danh”** hoặc **[bí danh]**

Loại bỏ các bộ trùng lặp với từ khóa DISTINCT

Bảng PC

model	speed	hdd	screen	price
1003	2	250	14	1000
1004	2	500	18	900
1005	2.5	500	18	800
1006	2.5	500	18	800

```
SELECT DISTINCT speed
FROM PC
```

speed
2
2.5

```
SELECT DISTINCT speed,hdd
FROM PC
```

speed	hdd
2	250
2	500
2.5	500

Phép chọn trong SQL

SELECT [ALL | DISTINCT]
* | {column_name | expression [alias],…}
FROM table
WHERE condition

Trong đó:

[condition]: biểu thức logic biểu diễn điều kiện chọn.

Các ví dụ cho lệnh chọn

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90;
```

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'Goyal';
```

```
SELECT last_name, salary
FROM   employees
WHERE  salary <= 3000;
```

```
SELECT last_name, salary
FROM   employees
WHERE  salary BETWEEN  2500 AND 3500;
```

Các ví dụ cho lệnh chọn

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201);
```

```
 SELECT   first_name
 FROM     employees
 WHERE    first_name LIKE 'S%';
```

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL;
```

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >=10000
AND    job_id LIKE '%MAN%';
```

Các phép toán trong SQL

- Các phép toán số học: +,-,*,/,
- Các phép toán so sánh: <,>,<=,>=,=,<> hoặc !=, BETWEEN .. AND
- Phép đối sánh mẫu: LIKE
- Các phép toán logic: AND, OR, NOT
- Các phép toán tập hợp: IN, UNION, INTERSECTION, EXCEPT (MINUS)

Đối sánh mẫu

- Khi so sánh các chuỗi, ngoài các phép toán quan hệ thông thường (<,>,=,v.v), SQL còn cung cấp khả năng so sánh theo mẫu (pattern), nó được gọi là "đối sánh mẫu" (pattern matching).

- Cú pháp như sau: s LIKE p
  trong đó:
  - s: là chuỗi ta muốn đem so sánh
  - p: là một mẫu cần so sánh. Nó có thể chứa các ký bất kỳ và 2 loại ký tự mẫu đại diện đặc biệt:
    - **"%"**: đại diện cho một chuỗi bất kỳ, kể cả chuỗi rỗng
    - **"_"**: đại diện cho đúng 1 ký tự bất kỳ

Một số ví dụ

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy
1	Davolio	Nancy	Sales Representative	Ms.
2	Fuller	Andrew	Vice President, Sales	Dr.
3	Leverling	Janet	Sales Representative	Ms.
4	Peacock	Margaret	Sales Representative	Mrs.
5	Buchanan	Steven	Sales Manager	Mr.
6	Suyama	Michael	Sales Representative	Mr.
7	King	Robert	Sales Representative	Mr.
8	Callahan	Laura	Inside Sales Coordinator	Ms.
9	Dodsworth	Anne	Sales Representative	Ms.

Bảng Employees trong CSDL NorthWind

Một số ví dụ

Tìm các nhân viên có FirstName bắt đầu là ký tự ‘A’?

```
SELECT  EmployeeID, LastName, FirstName,Title
FROM    Employees
WHERE   FirstName LIKE 'A%'
```

EmployeeID	LastName	FirstName	Title
2	Fuller	Andrew	Vice President, Sales
9	Dodsworth	Anne	Sales Representative

Một số ví dụ

Tìm các nhân viên có LastName có độ dài ít nhất 2 ký tự và có ký tự cuối cùng là ‘n’ ?

```
SELECT  EmployeeID, LastName, FirstName,Title
FROM    Employees
WHERE   LastName LIKE '%_n'
```

EmployeeID	LastName	FirstName	Title
5	Buchanan	Steven	Sales Manager
8	Callahan	Laura	Inside Sales Coordinator

Giá trị NULL và logic UNKNOWN

- **NULL** là giá trị đặc biệt được đưa vào để biểu diễn giá trị cho các thuộc tính mà ***không có giá trị***.
- **NULL** không thuộc miền giá trị của bất kỳ kiểu dữ liệu nào, nên thực ra nó không được coi như một "giá trị" thực sự cho một thuộc tính, mà chỉ có ý nghĩa đánh dấu là thuộc tính này chưa có giá trị (chưa được khởi tạo, cũng như chưa được cập nhật giá trị).
- Do đó, việc so sánh một giá trị với **NULL** có thể không trả về giá trị logic ***TRUE*** hay ***FALSE*** như các giá trị thông thường. Chính vì vậy, các hệ QTCSDL đưa thêm vào một giá trị logic thứ ba để biểu diễn tình huống này và gọi nó là ***Unknown***

Bảng giá trị logic với ‘Unknown’

p	*q*	*p* OR *q*	*p* AND *q*	*p* = *q*
True	True	True	True	True
True	False	True	False	False
True	Unknown	True	Unknown	Unknown
False	True	True	False	False
False	False	False	False	True
False	Unknown	Unknown	False	Unknown
Unknown	True	True	Unknown	Unknown
Unknown	False	Unknown	False	Unknown
Unknown	Unknown	Unknown	Unknown	Unknown

p	NOT *p*
True	False
False	True
Unknown	Unknown

Sắp xếp kết quả tìm kiếm

SELECT [ALL | DISTINCT]
    * | {column_name | expression [alias],…}
FROM table
[WHERE conditions]
[ORDER BY {expression [ASC | DESC] ,…}]

- Sử dụng mệnh đề ORDER BY, nó phải là mệnh đề cuối cùng trong lệnh SELECT.
- **Expression**: Xác định một hoặc nhiều thuộc tính trong số các thuộc tính mà ta muốn sắp xếp. Khi có nhiều thuộc tính, thì việc sắp xếp sẽ lần lượt theo thứ tự xuất hiện của các thuộc tính.

Ví dụ

In d/s nhân viên có sắp xếp theo ***LastName***:

```
SELECT  EmployeeID, LastName, FirstName, Title, TitleOfCourtesy
FROM    Employees
ORDER BY LastName
```

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy
10	Binh	Nguyen	NULL	NULL
5	Buchanan	Steven	Sales Manager	Mr.
8	Callahan	Laura	Inside Sales Coordinator	Ms.
1	Davolio	Nancy	Sales Representative	Ms.
9	Dodsworth	Anne	Sales Representative	Ms.
2	Fuller	Andrew	Vice President, Sales	Dr.
7	King	Robert	Sales Representative	Mr.
3	Leverling	Janet	Sales Representative	Ms.
4	Peacock	Margaret	Sales Representative	Mrs.
6	Suyama	Michael	Sales Representative	Mr.

Ví dụ

In d/s nhân viên có sắp xếp theo ***TitleOfCourtesy*** (theo thứ tự giảm dần) và ***FirstName***:

```
SELECT  EmployeeID, LastName, FirstName, Title, TitleOfCourtesy
FROM    Employees
ORDER BY TitleOfCourtesy DESC, 3
```

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy
9	Dodsworth	Anne	Sales Representative	Ms.
3	Leverling	Janet	Sales Representative	Ms.
8	Callahan	Laura	Inside Sales Coordinator	Ms.
1	Davolio	Nancy	Sales Representative	Ms.
4	Peacock	Margaret	Sales Representative	Mrs.
6	Suyama	Michael	Sales Representative	Mr.
7	King	Robert	Sales Representative	Mr.
5	Buchanan	Steven	Sales Manager	Mr.
2	Fuller	Andrew	Vice President, Sales	Dr.
10	Binh	Nguyen	NULL	NULL

Tóm tắt

Thank You !

Viện Điện tử - Viễn thông
Bộ môn Điện tử - Kỹ thuật máy tính

Kỹ thuật phần mềm ứng dụng

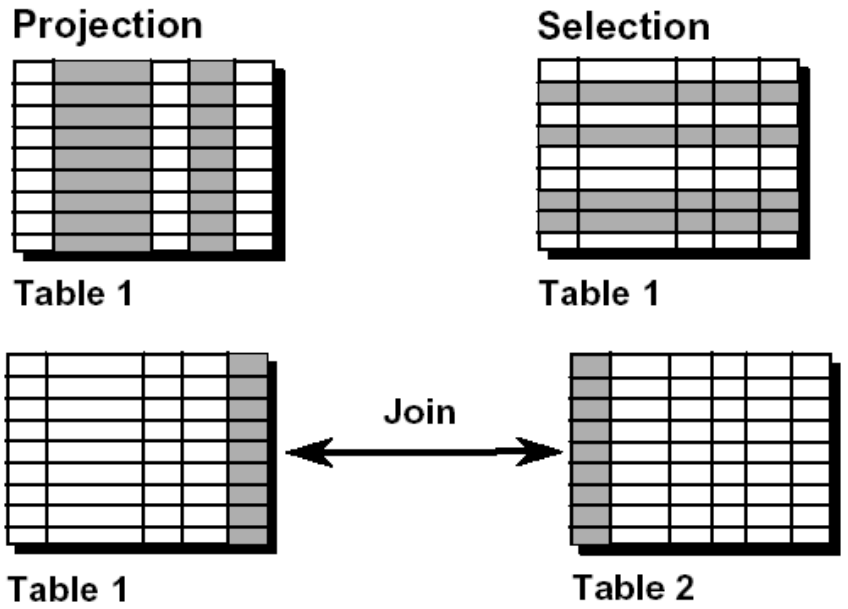
Chương 9: Ngôn ngữ SQL

Phần 2: Câu truy vấn trên nhiều bảng

Lệnh SELECT

Lệnh SELECT là một lệnh đa năng để truy vấn dữ liệu trong CSDL. Nó cho phép thực hiện tất cả các thao tác cơ bản trong đại số quan hệ như:

- Chiếu (Projection)
- Chọn (Selection)
- **Nối (Joining)**
- **Các phép toán tập hợp (Hợp, giao, trừ)**



Phép nối (join)

Oracle Proprietary Joins (8*i* and prior):

- Equijoin
- Non-equijoin
- Outer join
- Self join

SQL: 1999 Compliant Joins:

- Cross joins
- Natural joins
- Using clause
- Full or two sided outer joins
- Arbitrary join conditions for outer joins

Phép nối (join)

- Phân loại:
  - Tích Đề các: cross join
  - Nối bằng: equi-join
    - Nối tự nhiên: natural join
  - Nối không bằng: theta-join
  - Nối trong: inner join
  - Nối ngoài: outer join

Tích Đề các

Bảng PC

model	speed	hdd	screen	price
1003	2	250	14	1000
1004	2	500	18	900
1005	2.5	500	18	800
1006	2.5	500	18	800

Bảng Product

maker	model	type
A	1002	Laptop
A	1003	PC
B	1004	PC
B	1005	PC
B	1006	PC
B	1007	Laptop

Tính PC x Product

```
SELECT  *
FROM    PC, Product
```

Tích Đề các: PC x Product

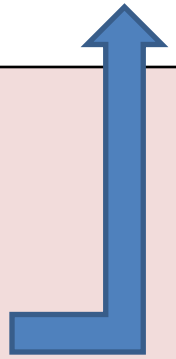
model	speed	hdd	screen	price	maker	model	type
1003	2	250	14	1000	A	1002	Laptop
1003	2	250	14	1000	A	1003	PC
1003	2	250	14	1000	B	1004	PC
1003	2	250	14	1000	B	1005	PC
1003	2	250	14	1000	B	1006	PC
1003	2	250	14	1000	B	1007	Laptop
1004	2	500	18	900	A	1002	Laptop
1004	2	500	18	900	A	1003	PC
1004	2	500	18	900	B	1004	PC
1004	2	500	18	900	B	1005	PC
1004	2	500	18	900	B	1006	PC

Tránh trùng tên bảng và thuộc tính - đặt bí danh

Bảng PC

model	speed	hdd	screen	price
1003	2	250	14	1000
1004	2	500	18	900
1005	2.5	500	18	800
1006	2.5	500	18	800

model 1	model 2
1004	1005
1004	1006
1005	1006



Tìm các cặp PC có ổ cứng bằng nhau:

```
SELECT  p1.model [model 1], p2.model [model 2]
FROM    PC p1, PC p2
WHERE (p1.model < P2.model ) AND
          (p1.hdd = P2.hdd)
```

Nối bằng

Bảng PC

model	speed	hdd	screen	price
1003	2	250	14	1000
1004	2	500	18	900
1005	2.5	500	18	800
1006	2.5	500	18	800

Bảng Product

maker	model	type
A	1002	Laptop
A	1003	PC
B	1004	PC
B	1005	PC
B	1006	PC
B	1007	Laptop

Liệt kê chi tiết thông tin về các sản phẩm của các nhà sản xuất

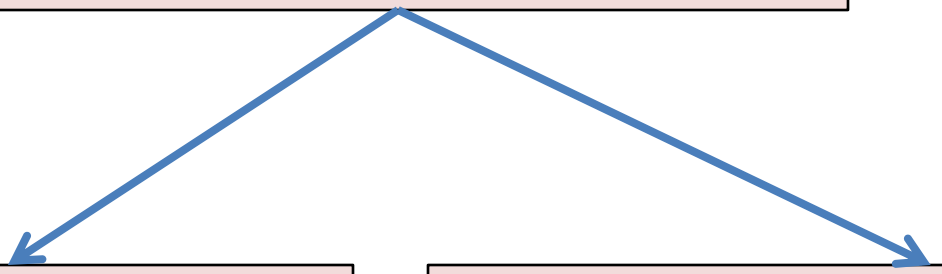
Product ⋈ PC

Product.model = PC.model

Nối bằng

Product ⋈ PC

Product.model = PC.model



```
SELECT  *
FROM    PC, Product
WHERE PC.model =
          Product.model
```

Cách 1: dùng logic chọn

```
SELECT  *
FROM    PC JOIN Product
          ON PC.model =
           Product.model
```

Cách 2: dùng lệnh JOIN

Kết quả

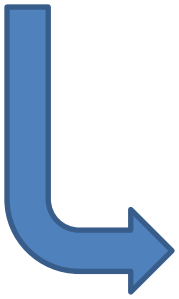
```
SELECT  *
FROM    PC, Product
WHERE PC.model =
          Product.model
```

```
SELECT  *
FROM    PC JOIN Product
          ON PC.model =
           Product.model
```

model	speed	hdd	screen	price	maker	model	type
1003	2	250	14	1000	A	1003	PC
1004	2	500	18	900	B	1004	PC
1005	2.5	500	18	800	B	1005	PC
1006	2.5	500	18	800	B	1006	PC

Sử dụng bí danh trong lệnh JOIN

```
SELECT  Pr.maker, PC.model, speed, hdd, screen, price
FROM    PC JOIN Product  Pr
                ON PC.model = Pr.model
```



maker	model	speed	hdd	screen	price
A	1003	2	250	14	1000
B	1004	2	500	18	900
B	1005	2.5	500	18	800
B	1006	2.5	500	18	800

Nối không bằng

- Trong t/h nối không bằng, thì tương tự như t/h nối bằng, ta cũng có thể sử dụng 1 trong 2 cách:
  - Sử dụng điều kiện nối trong mệnh đề WHERE,
  - Hoặc sử dụng lệnh JOIN với điều kiện nối (ON) không bằng

Nối trong và nối ngoài

- Nối trong (INNER JOIN):
  - kết quả chỉ ghép các bộ khớp nhau (matching tuples) trong 2 bảng thành phần (phép JOIN ở trên mặc định chính là INNER JOIN)
- Nối ngoài (OUTER JOIN): Kết quả chứa 2 thành phần:
  - Thành phần 1 như INNER JOIN
  - Thành phần 2 chứa cả các bộ không khớp nhau trong 2 bảng thành phần

Nối ngoài

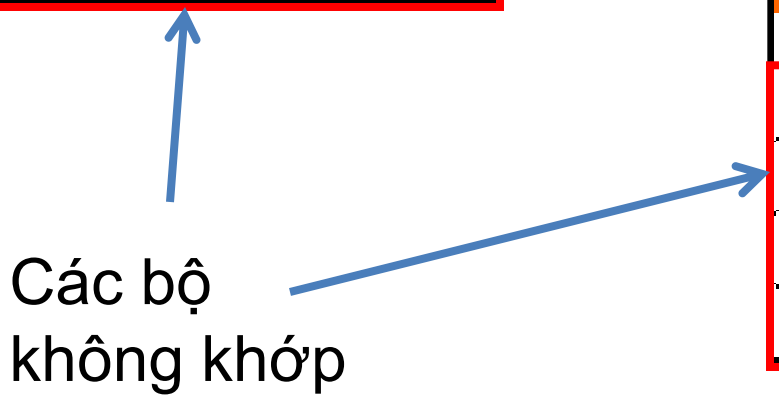
```
SELECT  *
FROM    A  OUTER  JOIN  B
            ON  A.X = B.Y;
```

- Phân loại:
  - **Left Outer Join**: chứa các bộ không khớp của bảng bên trái A
  - **Right Outer Join**: chứa các bộ không khớp của bảng bên phải B
  - **Full Outer Join**: chứa các bộ không khớp của cả hai bảng A và B

Ví dụ

CLASS	
CLASS_ID	**CLASS_NAME**
106	Lop 106
107	Lop 107
201	Lop 201
202	Lop 202

STUDENT		
CLASS_ID	**ID**	**NAME**
106	1	A
106	2	B
107	3	C
107	4	D
	5	E
	6	F
	7	G
	8	H



Left Outer Join

```
SELECT *
FROM   Class LEFT OUTER JOIN Student
       ON Class.Class_ID = Student.Class_ID;
```

CLASS		STUDENT		
CLASS_ID	**CLASS_NAME**	**CLASS_ID**	**ID**	**NAME**
106	Lop 106	106	1	A
106	Lop 106	106	2	B
107	Lop 107	107	3	C
107	Lop 107	107	4	D
201	Lop 201			
202	Lop 202			

RIGHT OUTER JOIN

```
SELECT *
FROM   Class RIGHT OUTER JOIN Student
       ON Class.Class_ID = Student.Class_ID;
```

CLASS		STUDENT		
CLASS_ID	CLASS_NAME	CLASS_ID	ID	NAME
106	Lop 106	106	1	A
106	Lop 106	106	2	B
107	Lop 107	107	3	C
107	Lop 107	107	4	D
			5	E
			6	F
			7	G
			8	H

FULL OUTER JOIN

```
SELECT *
FROM   Class FULL OUTER JOIN Student
       ON Class.Class_ID = Student.Class_ID;
```

CLASS		STUDENT		
CLASS_ID	CLASS_NAME	CLASS_ID	ID	NAME
106	Lop 106	106	1	A
106	Lop 106	106	2	B
107	Lop 107	107	3	C
107	Lop 107	107	4	D
			5	E
			6	F
			7	G
			8	H
201	Lop 201			
202	Lop 202			

Tóm tắt

- Các phép nối
  - Nối trong
  - Nối ngoài
  - Nối bằng
  - Nối không bằng
  - Tích Đề Các