CÔNG NGHỆ PHẦN MỀM

CHƯƠNG 3

Thiết kế và thực thi phần mềm

Nghệ An, 2021

Nội dung giảng dạy

3.1. Mô hình hóa

3.2. Nội dung thiết kế

3.3. Thiết kế kiến trúc

3.4. Thiết kế giao diện

3.5. Thiết kế dữ liệu

3.6. Thiết kế hướng đối tượng với UML

3.7. Các vấn đề của việc thực thi phần mềm

3.8. Lập trình hiệu quả

3.1. Mô hình hóa

- Mô hình hóa hệ thống (system modeling)
  - Là quy trình phát triển các mô hình trừu tượng của một hệ thống, trong đó mỗi mô hình biểu diễn một góc nhìn.
  - Có thể sử dụng các mô hình khác nhau để biểu diễn hệ thống từ nhiều khía cạnh khác nhau.
  - Các mô hình
    - Giúp cho người phân tích hiểu được chức năng của một hệ thống
    - Được sử dụng để giao tiếp với khách hàng

Sử dụng mô hình hệ thống

- Các mô hình của những hệ thống đã có sẵn
  - Được sử dụng trong suốt giai đoạn công nghệ yêu cầu.
  - Giúp làm rõ việc hệ thống đó làm được gì.
  - Là một cơ sở để thảo luận về độ mạnh yếu của hệ thống cũ → tìm ra những yêu cầu cho hệ thống mới.
- Các mô hình cho hệ thống mới
  - Được sử dụng trong suốt quá trình công nghệ yêu cầu.
  - Hỗ trợ việc giải thích các yêu cầu cho các stakeholder của hệ thống
  - Sử dụng để thảo luận về các thiết kế và viết tài liệu hệ thống cho phần cài đặt.
- Quy trình công nghệ hướng mô hình (model-driven engineering process) có thể phát sinh một phần hay toàn bộ cài đặt hệ thống từ mô hình hệ thống

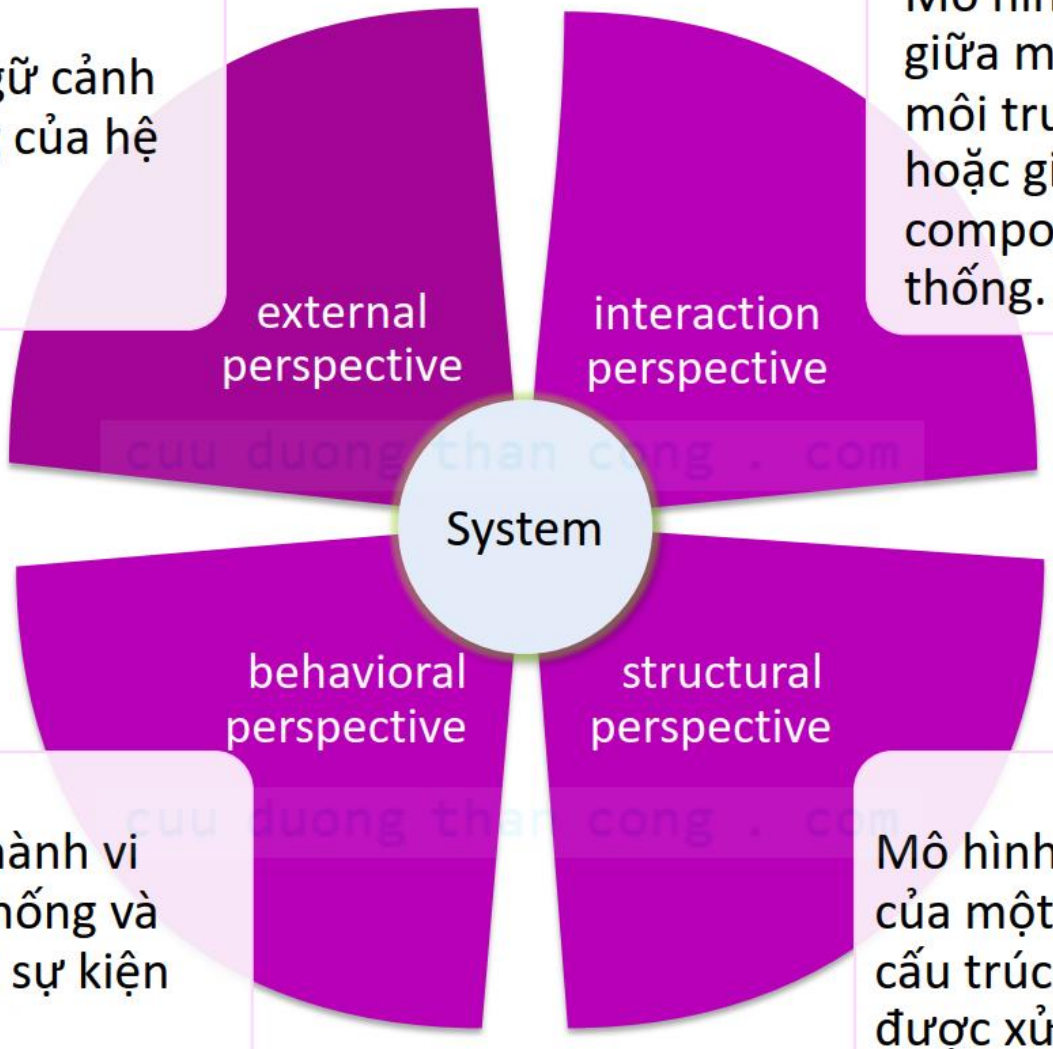
Sử dụng mô hình hệ thống

- Cách sử dụng các mô hình hệ thống:
  - Là phương tiện để thảo luận về hệ thống có sẵn hoặc hệ thống mới.
    - Các mô hình không cần đầy đủ và không chính xác.
  - Là một cách để viết tài liệu về hệ thống có sẵn
    - Cần chính xác nhưng không cần đầy đủ.
  - Là một mô tả chi tiết về hệ thống, có thể được sử dụng để phát sinh việc cài đặt hệ thống.
    - Các mô hình phải vừa đầy đủ và chính xác.

Các góc nhìn hệ thống

Mô hình hóa ngữ cảnh hay môi trường của hệ thống.

Mô hình hóa tương tác giữa một hệ thống và môi trường của nó, hoặc giữa các component của một hệ thống.



Mô hình hóa hành vi động của hệ thống và cách nó trả lời sự kiện như thế nào.

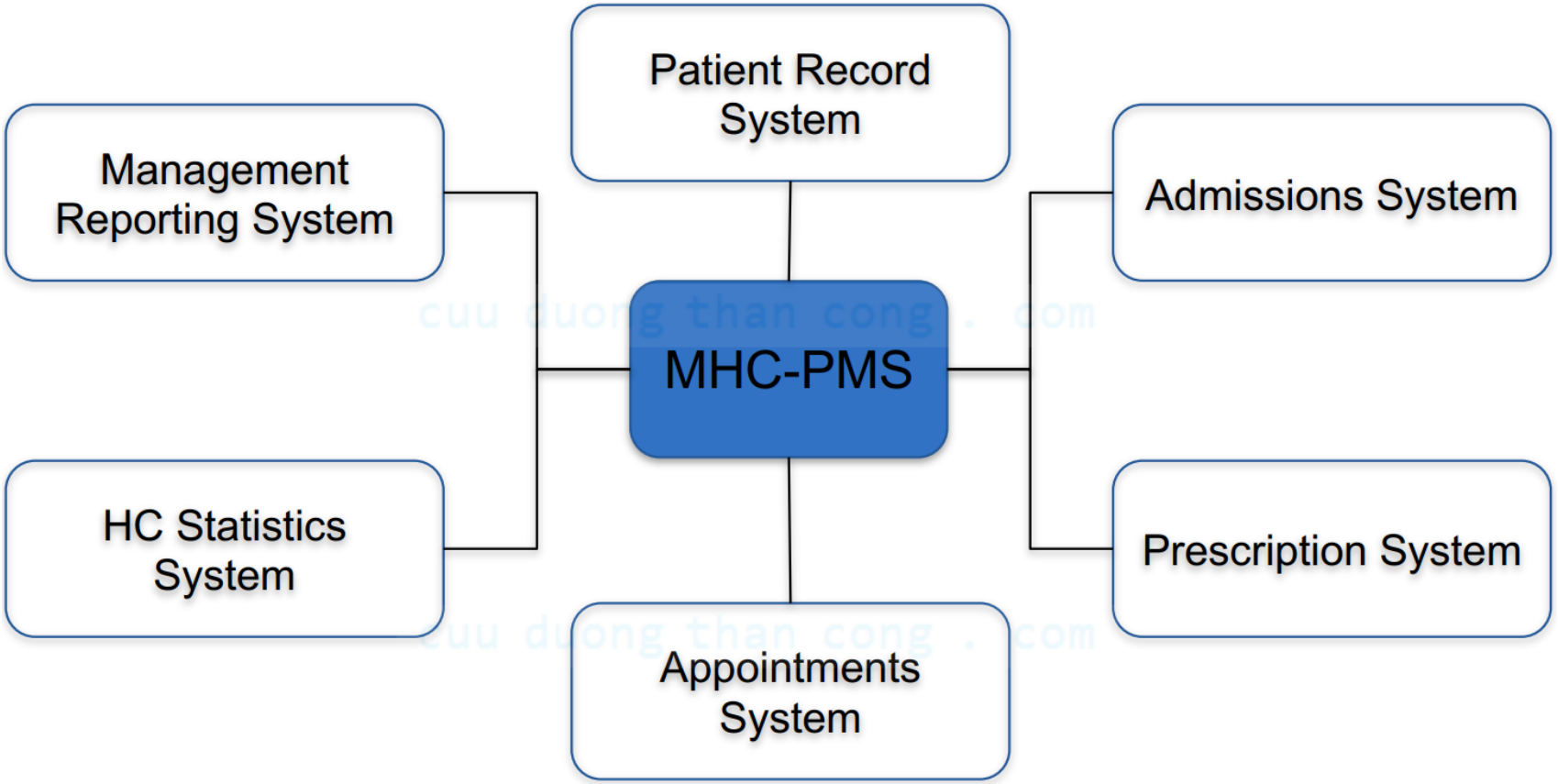
Mô hình hóa tổ chức của một hệ thống hay cấu trúc của dữ liệu được xử lý bởi hệ thống.

Mô hình ngữ cảnh

- Được dùng để minh họa cho ngữ cảnh vận hành của một hệ thống
  - Chỉ ra cái nào nằm bên trong hệ thống, bên ngoài hệ thống.
- Các yếu tố về xã hội và tổ chức có thể ảnh hưởng đến quyết định đưa ra vị trí đường ranh giới hệ thống.
- Các mô hình kiến trúc chỉ ra kiến trúc của một hệ thống và mối quan hệ với các hệ thống khác.
- Ranh giới hệ thống
  - Các ranh giới hệ thống được thiết lập để định nghĩa cái gì ở bên trong và cái gì ở bên ngoài hệ thống.
  - Vị trí của đường ranh giới hệ thống có ảnh hưởng sâu sắc đến yêu cầu hệ thống.

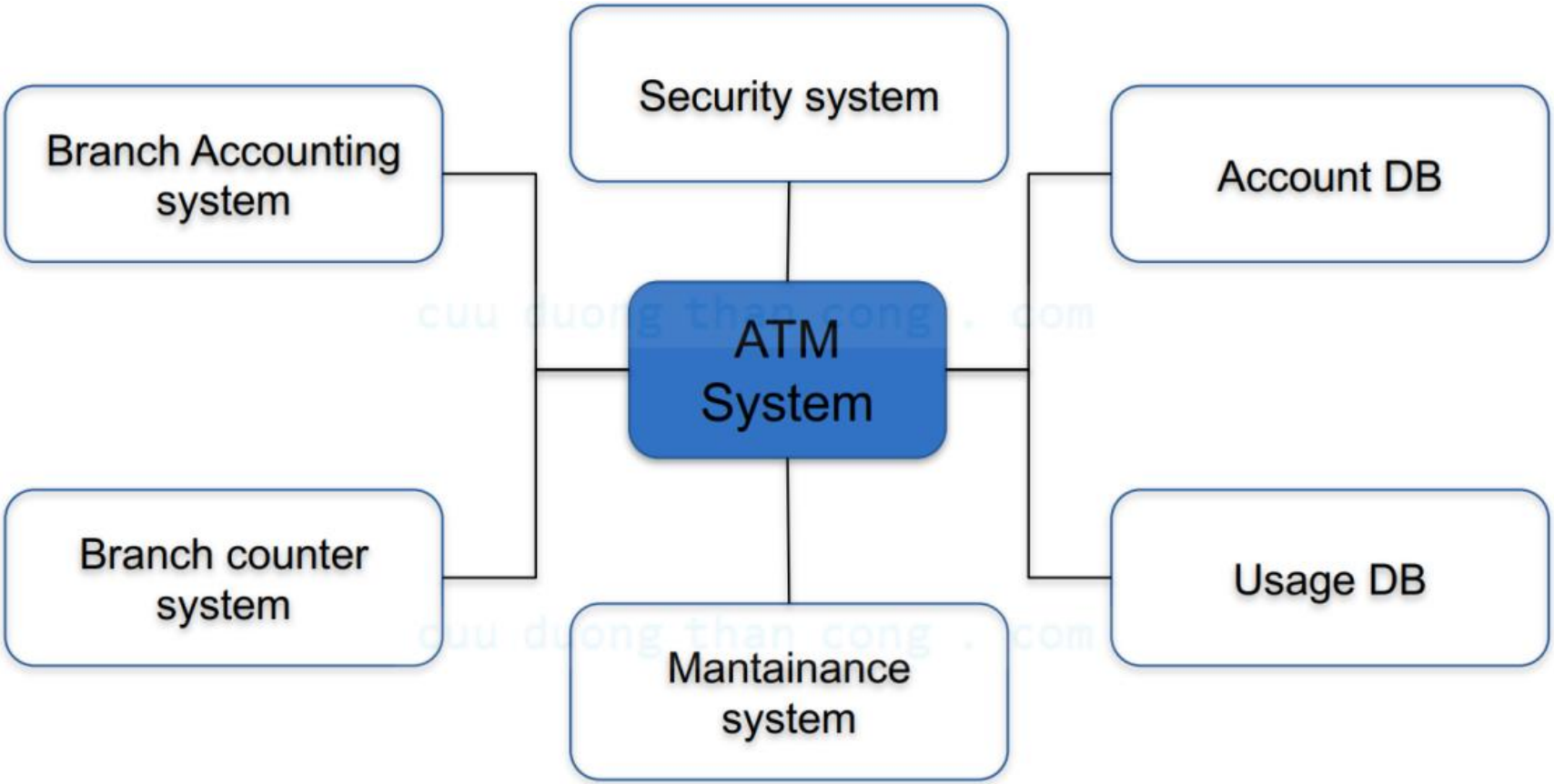
Mô hình ngữ cảnh

- Ngữ cảnh của hệ thống MHC-PMS (Mental Health Care-Patient Management System):



Mô hình ngữ cảnh

- Ngữ cảnh của hệ thống ATM:



Mô hình tương tác

- Mô hình tương tác người dùng hỗ trợ việc nhận diện các yêu cầu người dùng.
- Mô hình hóa tương tác của một hệ thống với hệ thống khác làm nổi rõ các vấn đề về mặt giao tiếp có thể phát sinh giữa hai hệ thống.
- Mô hình hóa tương tác component giúp ta hiểu rõ liệu một cấu trúc hệ thống đưa ra có đáp ứng được các yêu cầu về hiệu năng và độ tin cậy hay không.
- Có thể sử dụng biểu đồ use case, sequence diagram

Mô hình cấu trúc

- Hiển thị cấu trúc của một hệ thống về các component tạo nên hệ thống đó và mối quan hệ của chúng.
- Các mô hình cấu trúc có thể là
  - Mô hình tĩnh (static model): chỉ ra cấu trúc của thiết kế hệ thống.
  - Mô hình động (dynamic model): chỉ ra tổ chức của hệ thống khi nó được thực thi.
- Tạo ra các mô hình cấu trúc của một hệ thống khi thảo luận và thiết kế kiến trúc hệ thống.
- Có thể sử dụng biểu đồ lớp (class diagram)

🕮 Mô hình hành vi

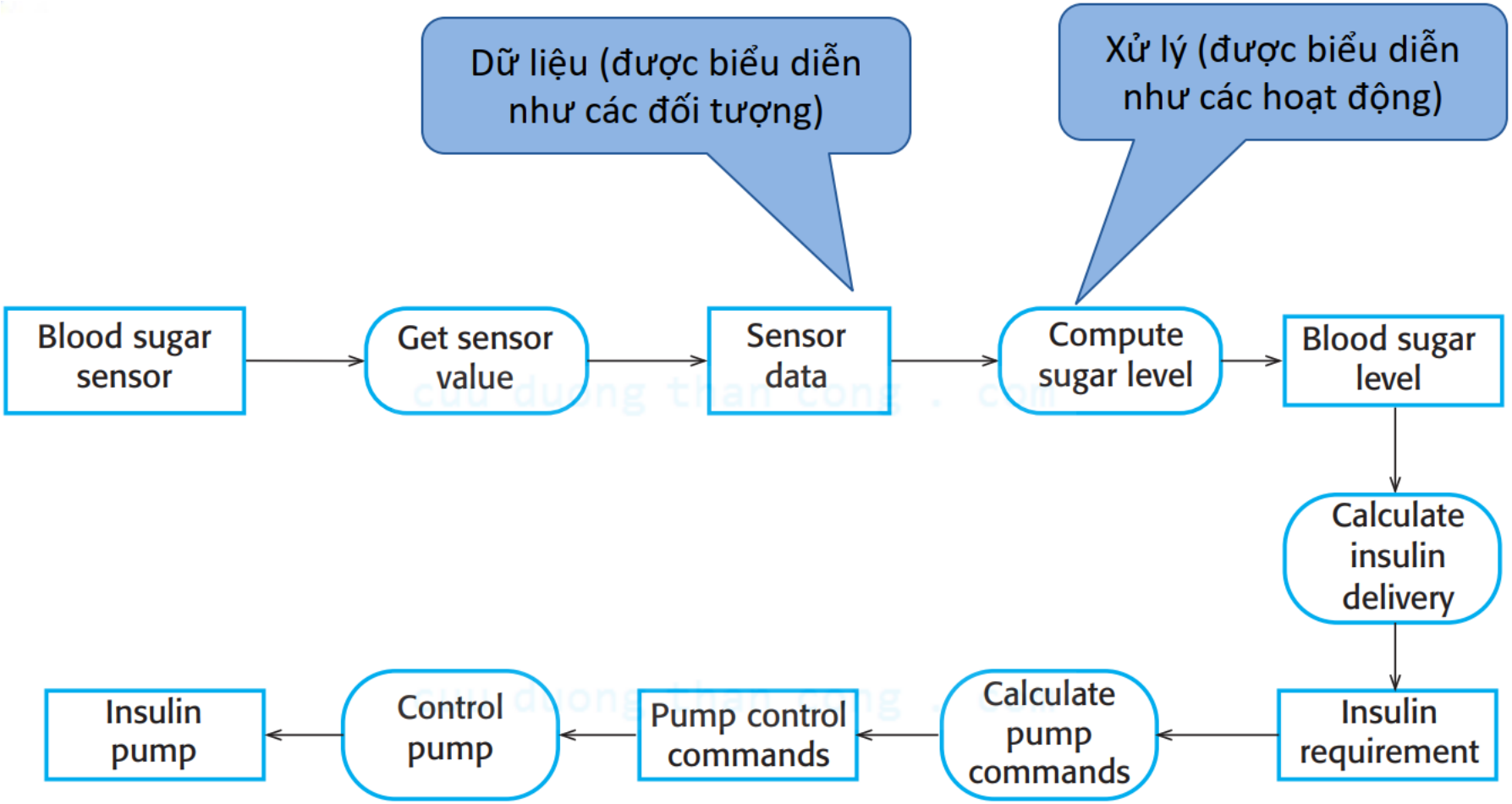
- Là các mô hình hành vi động (dynamic behavior) của một hệ thống khi nó đang thực thi.
- Chỉ ra cái gì xảy ra hoặc cái gì được giả định là xảy ra khi một hệ thống trả lời một tác động (stimuli) từ môi trường.
- Có hai loại tác động được đề cập đến:
  - Dữ liệu: Một số dữ liệu đến mà hệ thống phải xử lý.
  - Sự kiện: Một số sự kiện xảy ra làm kích hoạt việc xử lý của hệ thống

Mô hình hành vi

- Mô hình hướng dữ liệu:
  - Nhiều hệ thống doanh nghiệp là hệ thống xử lý dữ liệu.
    - Những hệ thống này được điều khiển bởi dữ liệu đầu vào của hệ thống, với việc xử lý khá ít các sự kiện bên ngoài.
  - Mô hình hướng dữ liệu chỉ ra một chuỗi tuần tự các hành động gồm việc xử lý dữ liệu đầu vào và phát sinh đầu ra tương ứng.
  - Những mô hình này đặc biệt hữu ích trong suốt quá trình phân tích yêu cầu vì chúng có thể chỉ ra được việc xử lý end-to-end trong một hệ thống.

Mô hình hành vi

- Mô hình hướng dữ liệu:



Mô hình hành vi

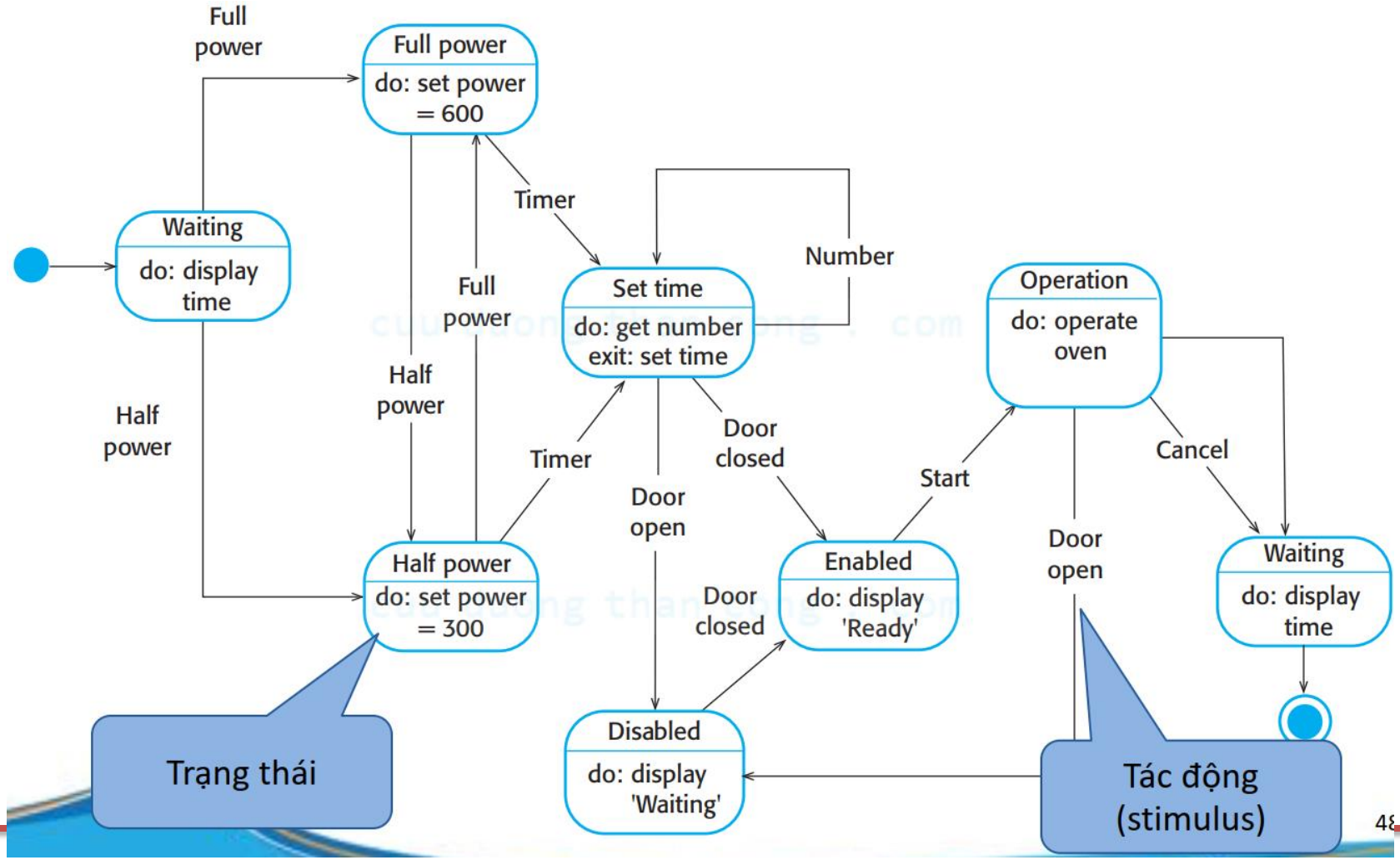
- Mô hình hướng sự kiện:
  - Các hệ thống thời gian thực thường là những hệ thống hướng sự kiện, với việc xử lý dữ liệu cực tiểu.
    - Ví dụ: hệ thống chuyển đổi đường điện thoại trả lời các sự kiện như ‘receiver off hook’ bằng việc phát sinh chuông.
  - Mô hình hướng sự kiện chỉ ra cách một hệ thống trả lời các sự kiện bên trong và bên ngoài.
  - Dựa vào giả thuyết rằng một hệ thống có một tập hữu hạn các trạng thái và các sự kiện đó có thể gây nên một chuyển đổi từ trạng thái này đến trạng thái khác.

Mô hình hành vi

- Mô hình máy trạng thái:
  - Mô hình hóa hành vi của hệ thống để trả lời các sự kiện bên trong và bên ngoài.
  - Chỉ ra các trả lời của hệ thống đối với sự kiện (thường được sử dụng để mô hình hóa các hệ thống thời gian thực)
  - Mô hình máy trạng thái (state machine model) chỉ ra các trạng thái của hệ thống (các nút) và sự kiện (cung) giữa các nút đó. Khi một sự kiện xảy ra, hệ thống chuyển từ trạng thái này sang trạng thái khác.
  - Biểu đồ trạng thái là một phần của UML và được dùng để biểu diễn các mô hình máy trạng thái.

Mô hình hành vi

- Biểu đồ trạng thái của một lò vi sóng:



Mô hình hành vi

- Trạng thái của lò vi sóng:

Trạng thái	Mô tả
Waiting	Lò vi ba đang đợi đầu vào. Màn hình hiển thị thời gian hiện tại.
Half power	Nguồn được thiết lập ở mức 300 watts. Màn hình hiển thị ‘Half power’.
Full power	Nguồn được thiết lập ở mức 600 watts. Màn hình hiển thị ‘Full power’.
Set time	Thời gian nấu được thiết lập bởi giá trị đầu vào của người dùng. Màn hình hiển thị thời gian nấu đã được chọn và cập nhật lại thời gian người dùng đã thiết lập.
Disabled	Hoạt động của lò vi ba bị dừng vì lý do an toàn. Đèn bên trong lò bật sáng. Màn hình hiển thị ‘Not ready’.
Enabled	Hoạt động của của lò vi ba ở trang thái sẵn sàng để nấu. Đèn trong lò tắt. Màn hình hiển thị ‘Ready to cook’.
Operation	Lò đang ở trạng thái hoạt động. Đèn trong lò bật sáng. Màn hình hiển thị bộ đếm lùi. Khi nấu xong, có một tiếng buzz trong 5s. Đèn trong lò bật sáng. Màn hình hiển thị ‘Cooking complete’ khi đang có tiếng buzz

Mô hình hành vi

- Các tác động vào lò vi sóng:

Tác động	Mô tả
Half power	Người sử dụng ấn nút half-power.
Full power	Người sử dụng ấn nút full-power.
Timer	Người sử dụng ấn một trong các nút định thời gian.
Number	Người sử dụng ấn nút số.
Door open	Cửa lò không được đóng.
Door closed	Cửa lò được đóng.
Start	Người sử dụng ấn nút Start.
Cancel	Người sử dụng ấn nút Cancel

3.2. Nội dung thiết kế

- Thế nào là thiết kế
  - Là một quá trình sáng tạo:
    - Nghĩ xem nên làm như thế nào (cách thức, phương án)
    - Biểu diễn cách thức, phương án
    - Xem xét lại, chi tiết hóa
    - → Đủ chi tiết để người lập trình biết phải làm như thế nào
  - Là phương tiện để trao đổi thông tin để đảm bảo chất lượng
    - Dễ hiểu, dễ sửa đổi hơn mã chương trình, cung cấp cái nhìn tổng thể đồng thời có nhiều mức chi tiết
  - Nếu không có thiết kế hoặc thiết kế tồi → Làm tăng công sức viết mã chương trình, tăng công sức bảo trì

Các nội dung của thiết kế

- Thiết kế kiến trúc: xác định tổng thể phần mềm gồm những thành phần nào, quan hệ giữa chúng
- Đặc tả trừu tượng: các dịch vụ của mỗi hệ con
- Thiết kế giao diện
- Thiết kế các thành phần
- Thiết kế cấu trúc dữ liệu
- Thiết kế thuật toán

Các mức thiết kế

- Thiết kế mức cao (high level design)
  - Mô hình tổng thể của hệ thống
  - Cách thức hệ thống được phân rã thành các module
  - Mối quan hệ giữa các module
  - Cách thức trao đổi thông tin giữa các module
- Thiết kế mức thấp (low level design) – thiết kế chi tiết: thiết kế nội dung của các thủ tục (sử dụng các biểu đồ, mã giả)

Tiếp cận hướng đối tượng

- Đặc trưng
  - Không có vùng dữ liệu dùng chung
  - Các đối tượng là thực thể độc lập
  - Các đối tượng có thể phân tán, hoạt động tuần tự or song song
- Cơ sở của thiết kế hướng đối tượng là các lớp (Class)
- Các bước thiết kế:
  - Xác định lớp đối tượng
  - Xác định thuộc tính + hành vi
  - Xác định tương tác giữa các đối tượng
  - Áp dụng tính kế thừa: xây dựng các lớp trừu tượng

3.3. Thiết kế kiến trúc

- Kiến trúc phần mềm không phải là mô hình hoạt động mà là mô hình phân hoạch giúp kĩ sư hệ thống:
  - Phân tích được tính hiệu quả của sự đáp ứng yêu cầu phần mềm
  - Nghiên cứu các giải pháp thay thế ở giai đoạn sớm khi các thay đổi còn tương đối dễ dàng
    - Thay đổi mô hình kiến trúc
  - Giảm các rủi ro gắn với phát triển phần mềm
    - Tính khả thi của mô hình
- Thiết kế kiến trúc là pha sớm nhất trong quy trình thiết kế hệ thống
- Kết quả của quy trình thiết kế kiến trúc là bản đặc tả về kiến trúc phần mềm.

Thiết kế kiến trúc

- Nếu có được bản thiết kế kiến trúc rõ ràng thì sẽ thấy được các ưu điểm của nó trong những hoạt động sau:
  - Giao tiếp giữa các stakeholder:
    - Kiến trúc hệ thống thường được sử dụng làm tâm điểm của các buổi thảo luận giữa các stakeholder.
  - Phân tích hệ thống:
    - Phân tích để xác định liệu hệ thống có thoả mãn các yêu cầu phi chức năng của nó hay không.
  - Tái sử dụng với quy mô lớn:
    - Kiến trúc có thể được tái sử dụng trong nhiều hệ thống.

Các bước của thiết kế kiến trúc

- Bao gồm việc phát hiện các thành phần chính của hệ thống và giao tiếp giữa chúng.
- Cấu trúc hóa hệ thống
  - Phân chia hệ thống thành các hệ con (sub-system) độc lập và xác định trao đổi thông tin giữa các hệ con
- Mô hình hóa điều khiển
  - Xác lập mô hình điều khiển giữa các phần của hệ thống
- Phân rã module
  - Phân rã các hệ con thành các module

Phân rã hệ thống

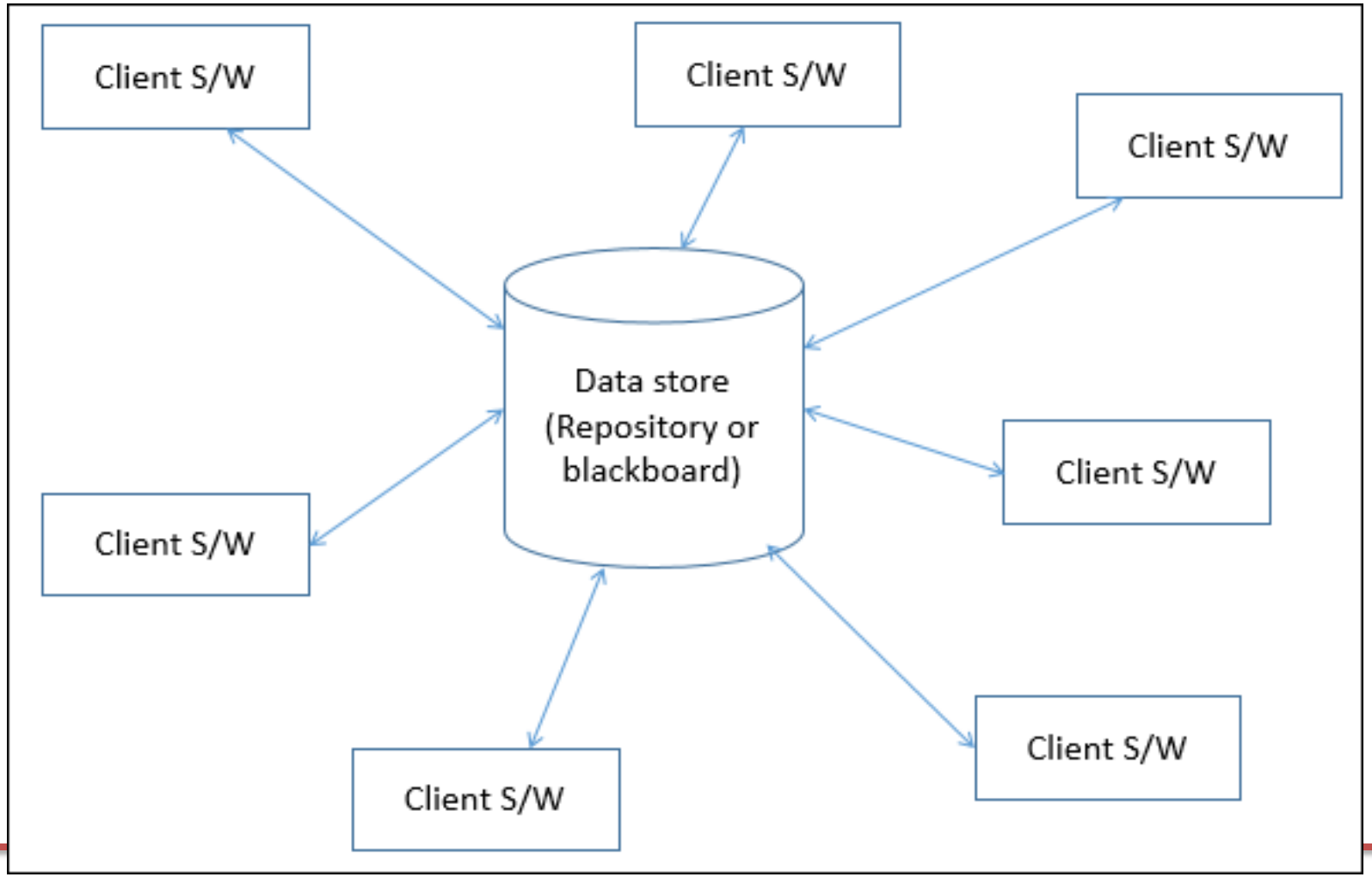
- Hệ thống con là một hệ thống có thể vận hành một cách độc lập, có thể sử dụng một số dịch vụ được cung cấp bởi các hệ thống con khác hoặc cung cấp dịch vụ cho các hệ thống con khác sử dụng.
- Module là một thành phần hệ thống cung cấp các dịch vụ cho các thành phần khác, nhưng nó thường không được coi như là một hệ thống riêng rẽ, độc lập.
- Có hai cách để phân rã các hệ thống con thành các module:
  - Phân rã hướng đối tượng: hệ thống được phân rã thành các đối tượng tương tác với nhau.
  - Phân rã hướng chức năng hoặc luồng dữ liệu: hệ thống được phân rã thành các module chức năng chịu trách nhiệm chuyển đổi thông tin đầu vào thành kết quả đầu ra.

Các mô hình kiến trúc

- Data-centered architectures
- Client-server architectures
- Layered architectures
- Call and return architectures
- Data flow architectures
- Object-oriented architectures

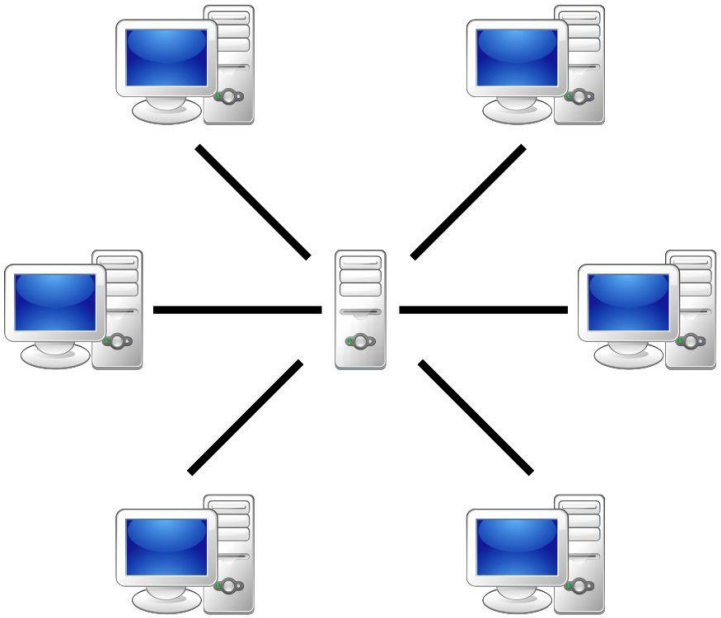
Các mô hình kiến trúc

- Kiến trúc dữ liệu tập trung

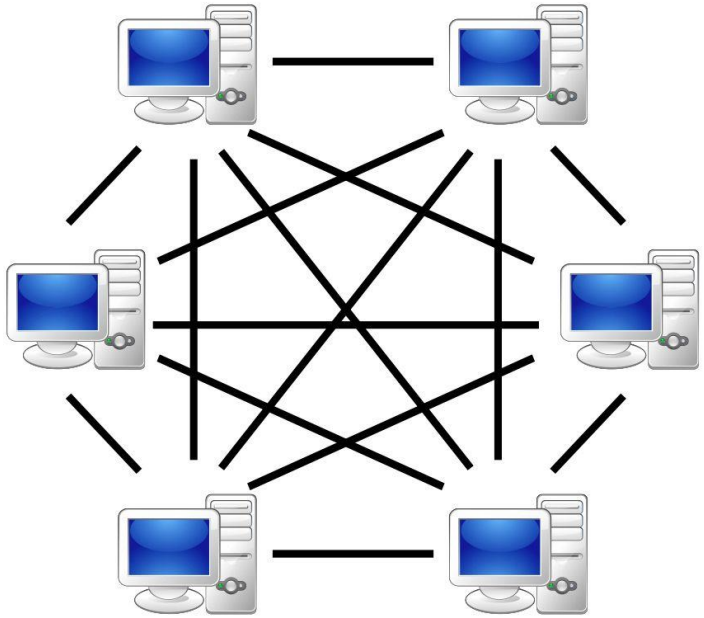


Các mô hình kiến trúc

- Kiến trúc Client – Server



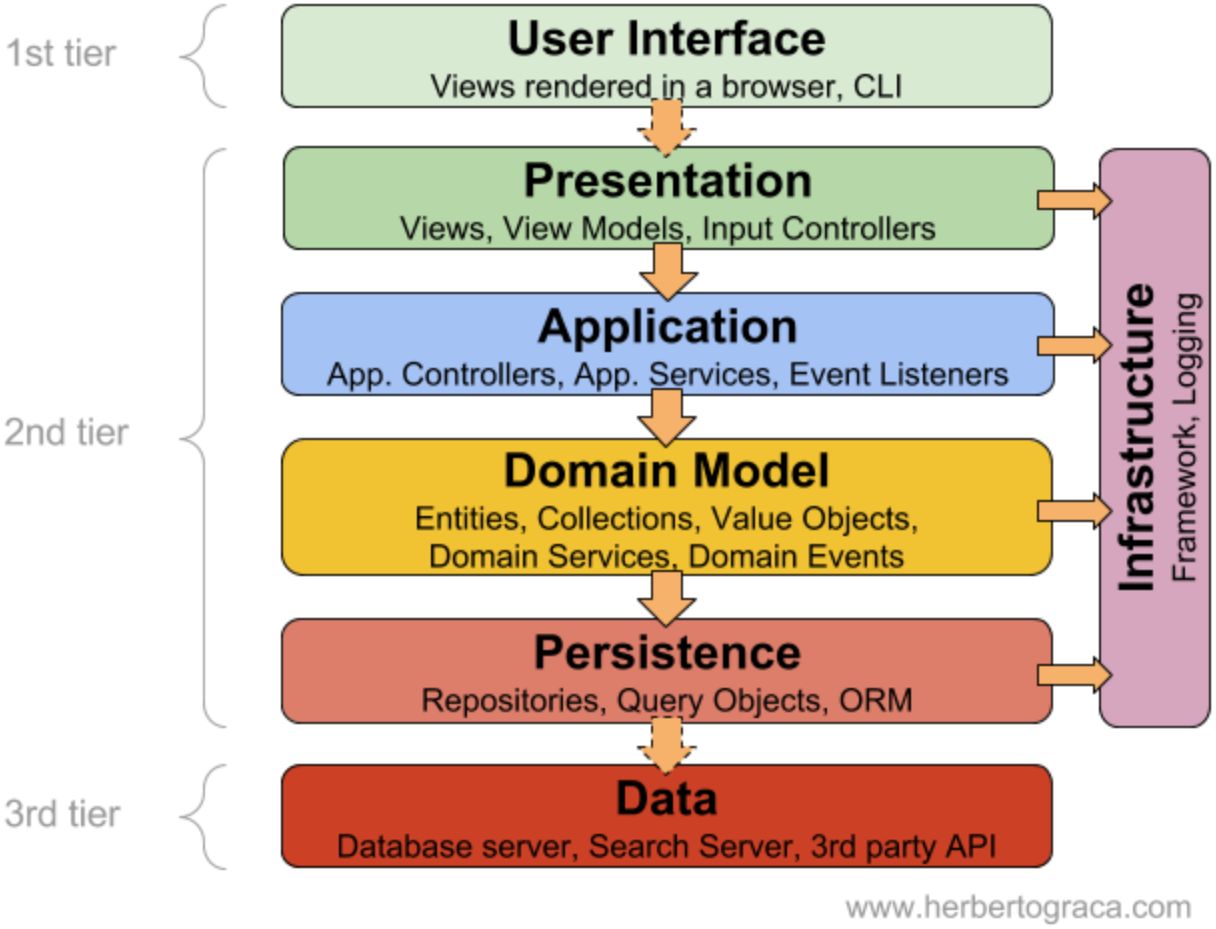
Server-based



P2P-network

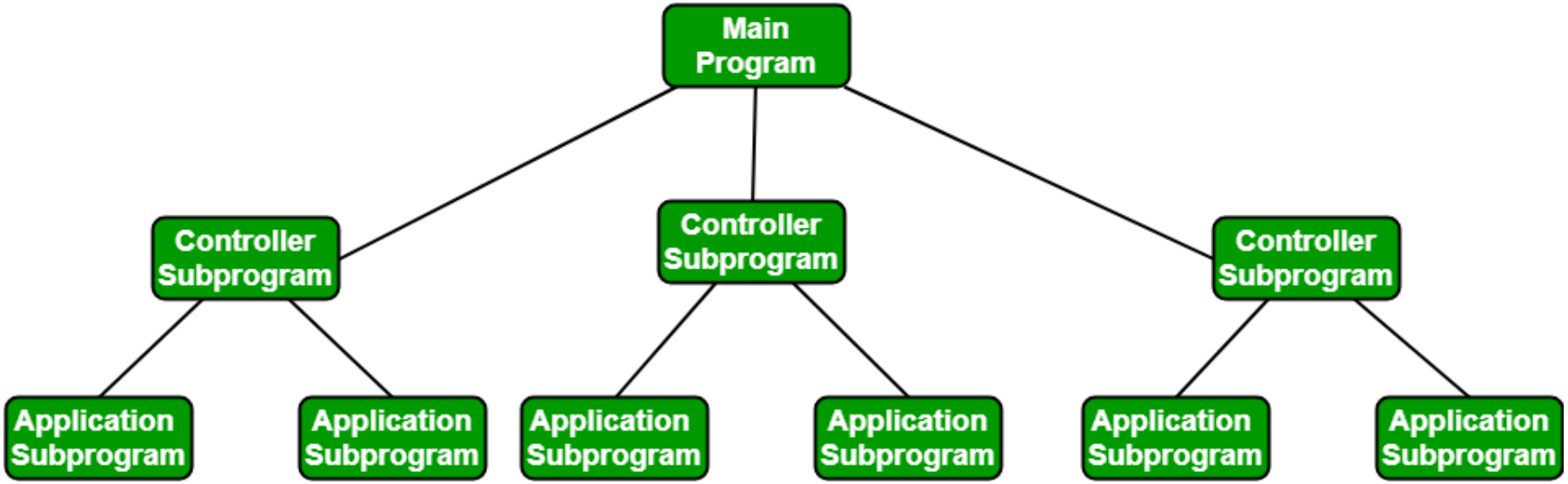
Các mô hình kiến trúc

- Kiến trúc phân tầng



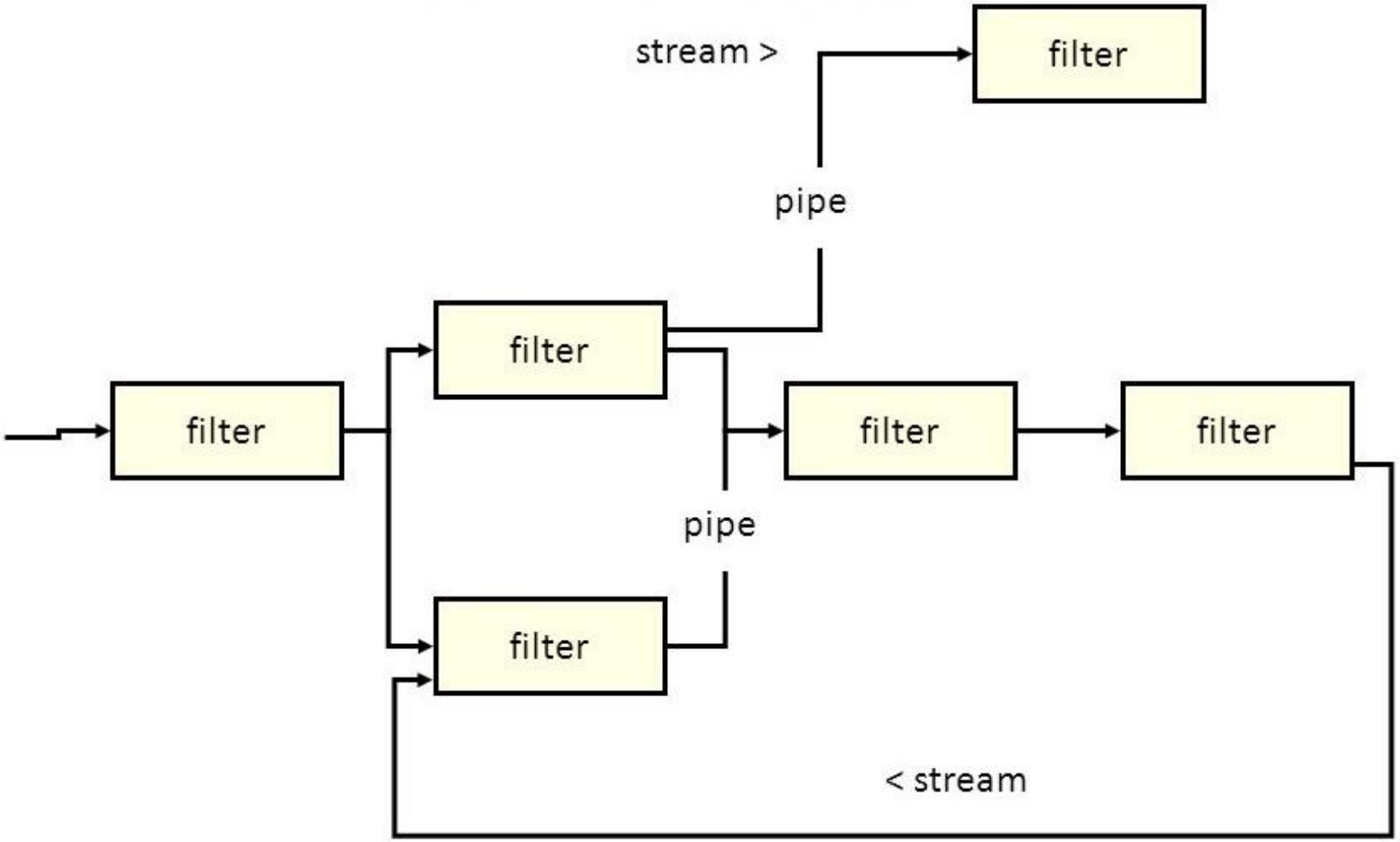
Các mô hình kiến trúc

- Kiến trúc gọi và chạy



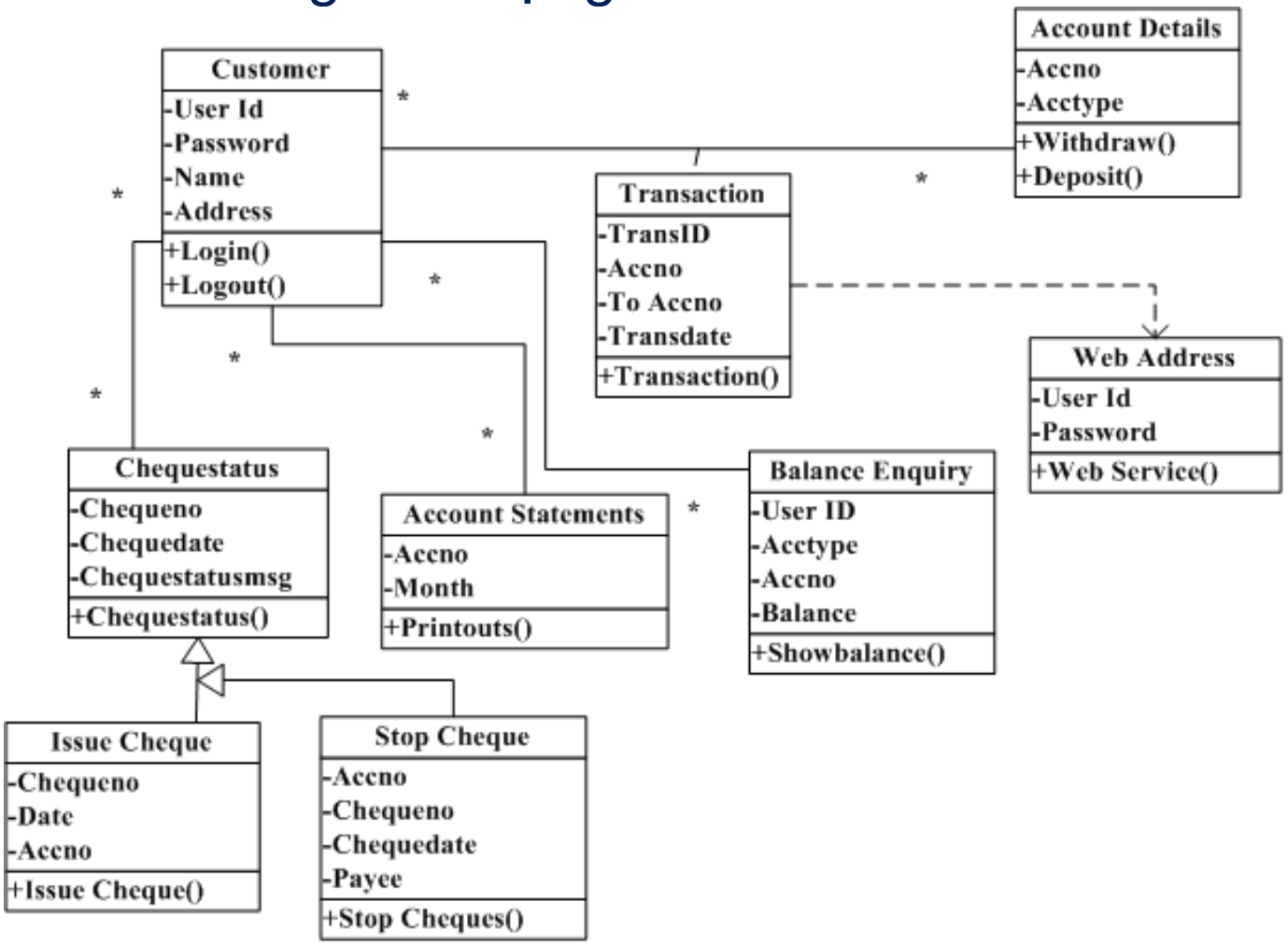
Các mô hình kiến trúc

- Kiến trúc luồng dữ liệu



Các mô hình kiến trúc

- Kiến trúc hướng đối tượng



3.4. Thiết kế giao diện

- Mô tả chi tiết cách thức giao tiếp giữa người sử dụng và phần mềm
- Màn hình giao diện
  - Nội dung
  - Hình thức trình bày
  - Biến cố phải xử lý

Thiết kế giao diện

- Ví dụ:

Màn hình tra cứu học sinh theo lớp

Tên lớp

Danh sách học sinh

STT	Họ tên	Giới tính	Ngày sinh	Địa chỉ

Giao diện người dùng

- Nên được thiết kế cho phù hợp với kĩ năng, kinh nghiệm và sự trông đợi của người dùng tương lai của hệ thống.
- Người dùng hệ thống ban đầu thường đánh giá một hệ thống theo giao diện thay vì chức năng.
- Một giao diện thiết kế tồi
  - Có thể dẫn đến việc một người dùng phạm những lỗi nghiêm trọng.
- Người thiết kế không nên chỉ thiết kế cho năng lực của chính mình.
- Mỗi người hợp với những kiểu tương tác khác nhau

Các nguyên tắc thiết kế UI

- Thiết kế UI phải xét đến nhu cầu, kinh nghiệm và năng lực của người dùng hệ thống.
- Người thiết kế cần
  - Nhận thức được các hạn chế về vật lý và tâm lý của con người (ví dụ: giới hạn của trí nhớ ngắn hạn)
  - Nhận thức rằng con người ai cũng có thể nhầm lẫn.
- Các nguyên tắc thiết kế UI đóng vai trò nền tảng cho các thiết kế giao diện
  - Tuy rằng không phải tất cả các nguyên tắc đều áp dụng được cho tất cả các thiết kế.

Các nguyên tắc thiết kế UI

Nguyên tắc	Mô tả
Quen thuộc với người dùng	Giao diện nên dùng các thuật ngữ và khái niệm rút ra từ kinh nghiệm của những người sẽ dùng hệ thống nhiều nhất.
Nhất quán	giao diện cần nhất quán sao cho các thao tác gần giống nhau có thể được kích hoạt theo cùng kiểu.
Ngạc nhiên tối thiểu	Người dùng không bao giờ bị bất ngờ về hành vi của hệ thống.
Khôi phục được	Giao diện nên có các cơ chế cho phép người dùng khôi phục lại tình trạng hoạt động bình thường sau khi gặp lỗi.
Hướng dẫn người dùng	Giao diện nên có phản hồi có nghĩa khi xảy ra lỗi và cung cấp các tiện ích trợ giúp theo ngữ cảnh.
Người dùng đa dạng	Giao diện nên cung cấp các tiện ích tương tác thích hợp cho các loại người dùng hệ thống khác nhau.

Các nguyên tắc thiết kế UI

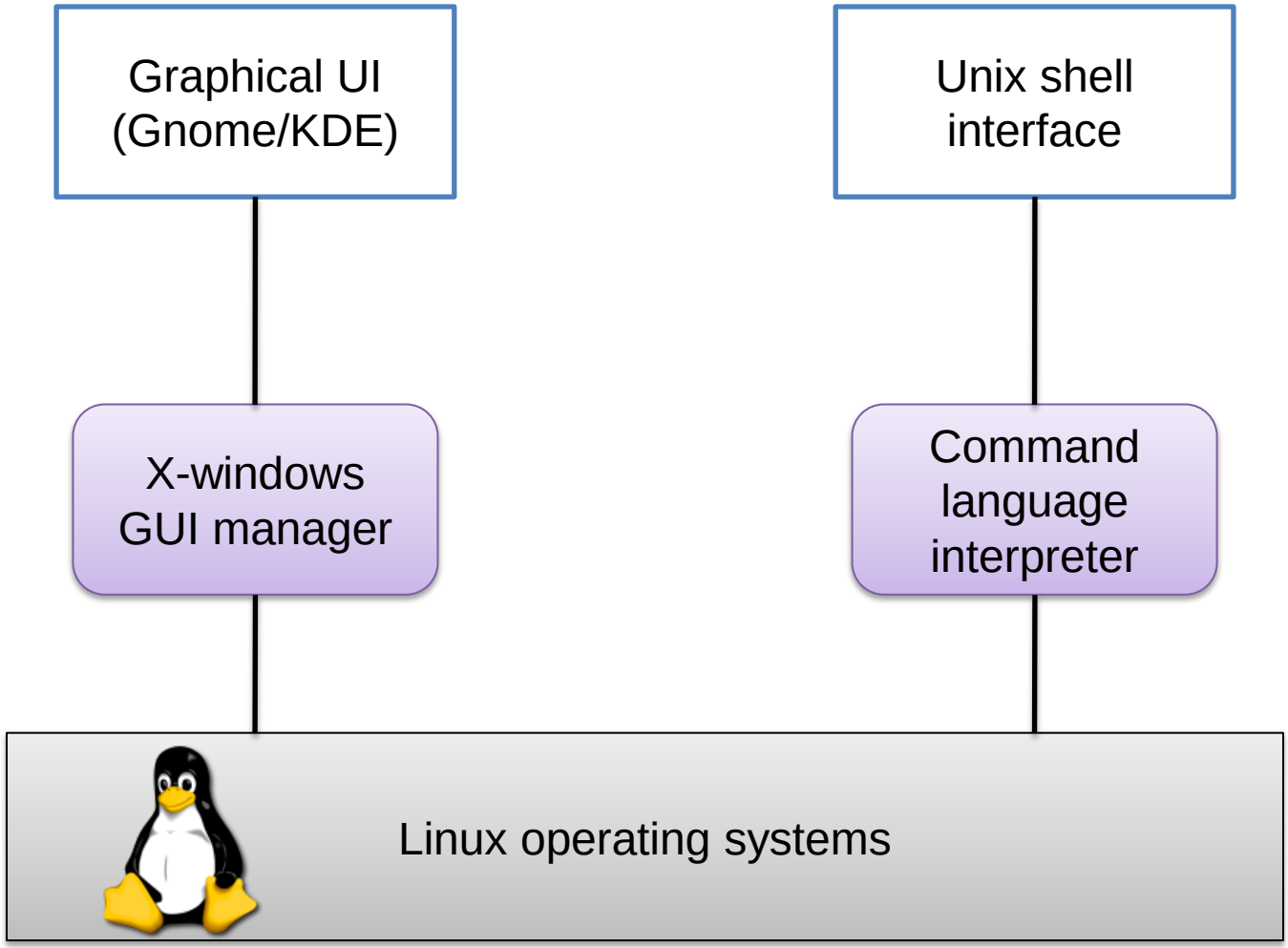
- Các vấn đề khi thiết kế UI
  - Người dùng cung cấp thông tin cho hệ thống bằng cách nào?
  - Hệ thống nên trình bày thông tin (output) cho người dùng như thế nào?
- Các kiểu tương tác
  - Thao tác trực tiếp (Direct manipulation)
  - Chọn lựa bằng menu (Menu selection)
  - Điền form (Form fill-in)
  - Dòng lệnh (Command language)
  - Ngôn ngữ tự nhiên (Natural language)

Các nguyên tắc thiết kế UI

– Các kiểu tương tác

Kiểu T.Tác	Ưu điểm	Nhược điểm	Ví dụ
Direct manipulation	Tương tác nhanh và trực quan Dễ học	Có thể khó cài đặt. Chỉ thích hợp khi có ẩn dụ hình ản cho các tác vụ và đối tượng.	Trò chơi điện tử Các hệ thống dùng drag-n-drop
Menu selection	Tránh lỗi người dùng Không phải gõ nhiều	Chậm chạp đối với người dùng nhiều kinh nghiệm. Có thể phức tạp nếu có nhiều lựa chọn menu.	Đa số các hệ thống thông dụng
Form fill-in	Nhập dữ liệu đơn giản Dễ học Kiểm tra được	Tốn không gian màn hình. Gây rắc rối khi các lựa chọn của người dùng không khớp với các trường của form.	Khai thuế, xử lý nợ cá nhân
Command language	Mạnh và linh động	Khó học xử lý lỗi kém	hệ điều hành
Natural language	Đáp ứng được người dùng không chuyên Dễ mở rộng	Cần gõ nhiều. Các hệ thống hiểu ngôn ngữ tự nhiên không đáng tin cậy.	Information retrieval systems

Các dạng UI

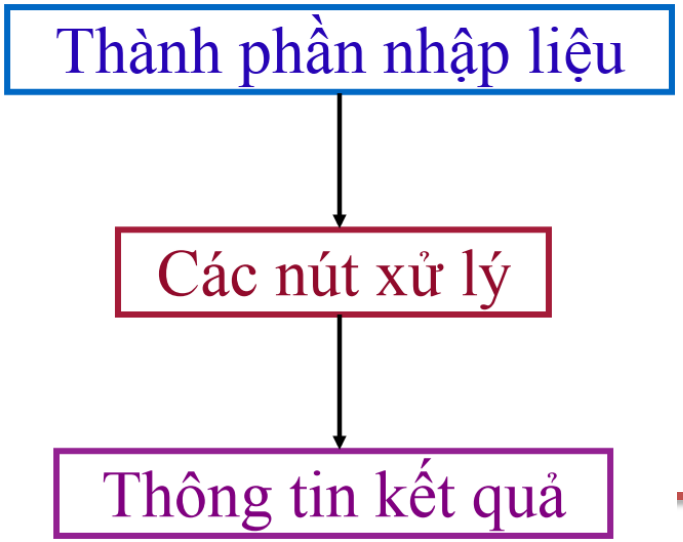

Graphical UI
(Gnome/KDE)
Unix shell
interface
X-windows
GUI manager
Command
language
interpreter
Linux operating systems

Phân loại màn hình

- Màn hình chính: cho phép người sử dụng chọn các công việc sẽ thực hiện với phần mềm.
- Màn hình nhập liệu: cho phép người sử dụng nhập vào các thông tin để lưu trữ hoặc tính toán.
- Màn hình tra cứu: cho phép tìm kiếm thông tin đã đượcb lưu trữ với các tiêu chuẩn tìm kiếm.
- Màn hình thông báo: hiển thị các thông báo, nhắc nhở.
- Báo cáo thống kê: các báo cáo thống kê theo một mốc thời gian định sẵn.

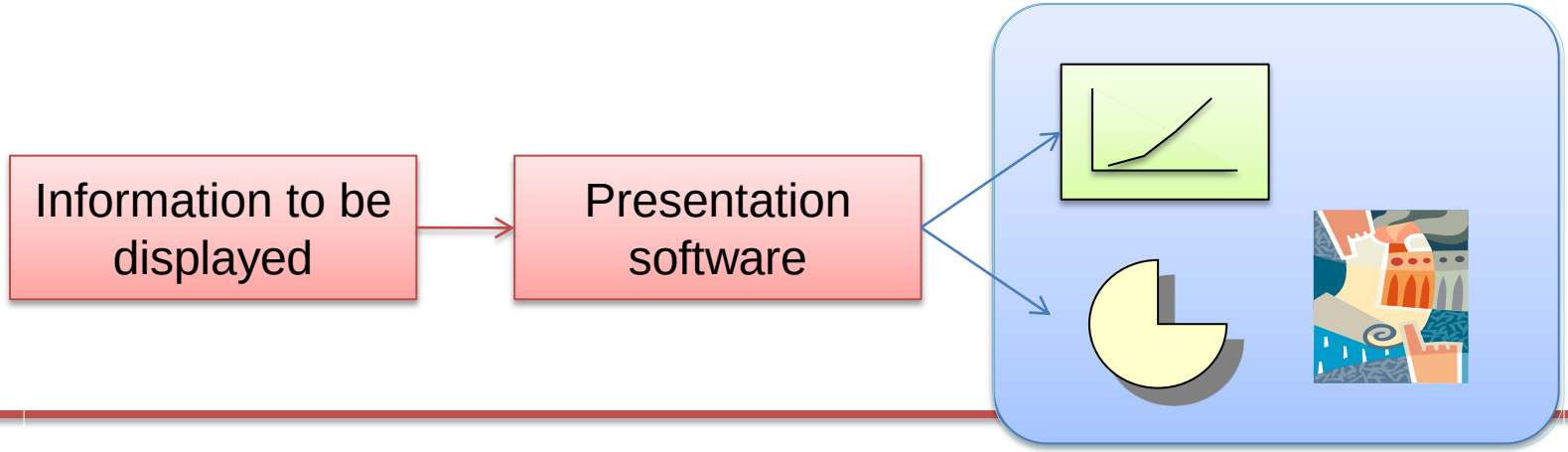
Kiến trúc màn hình

- Thành phần nhập liệu: cho phép người sử dụng nhập dữ liệu dưới nhiều hình thức khác nhau.
- Các nút xử lý: cho phép người sử dụng yêu cầu phần mềm thực hiện một xử lý nào đó.
- Thông tin kết quả: cho phép người sử dụng xem thông tin kết quả dưới nhiều hình thức khác nhau.



Biểu diễn thông tin

- Trình bày thông tin hệ thống như thế nào cho người dùng.
- Thông tin có thể được trình bày trực tiếp (ví dụ text trong một trình soạn thảo) hoặc được biến đổi thành một dạng biểu diễn khác (ví dụ dạng đồ họa)
- Thông tin tĩnh: Khởi tạo ở đầu session, không thay đổi trong suốt session.
- Thông tin động: Thay đổi trong session



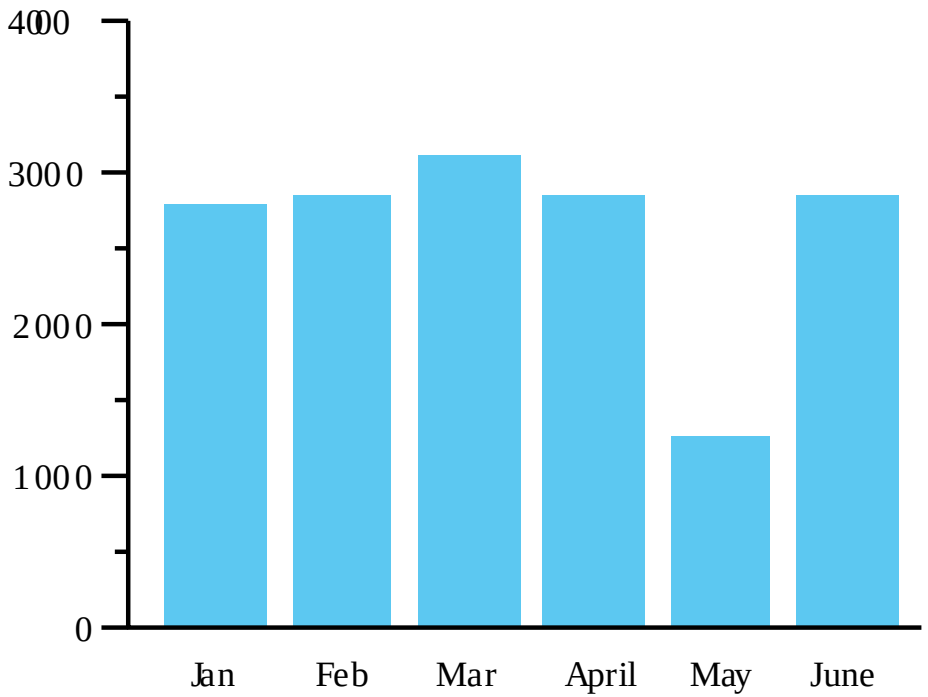
Các nhân tố khi hiển thị thông tin

- Người dùng quan tâm đến con số chính xác hay mối tương quan giữa các số liệu?
- Các giá trị của thông tin thay đổi nhanh chậm ra sao? Có phải lập tức thông báo khi có thay đổi?
- Người dùng có phải hành động để phản ứng với một thay đổi?
- Có giao diện thao tác trực tiếp (direct manipulation) không?
- Thông tin dạng số hay text? Các giá trị tương đối có quan trọng không?

Các nhân tố khi hiển thị thông tin

- Biểu diễn số (Digital presentation)
  - Ngắn gọn, chiếm ít không gian màn hình
  - Cho biết giá trị chính xác
- Biểu diễn tương tự (Analogue presentation)
  - Nhanh chóng lấy được ấn tượng về một giá trị
  - Có thể biểu diễn các giá trị tương đối
  - Dễ thấy các giá trị dữ liệu đặc biệt

Jan	Feb	Mar	April	May	June
2842	2851	3164	2789	1273	2835

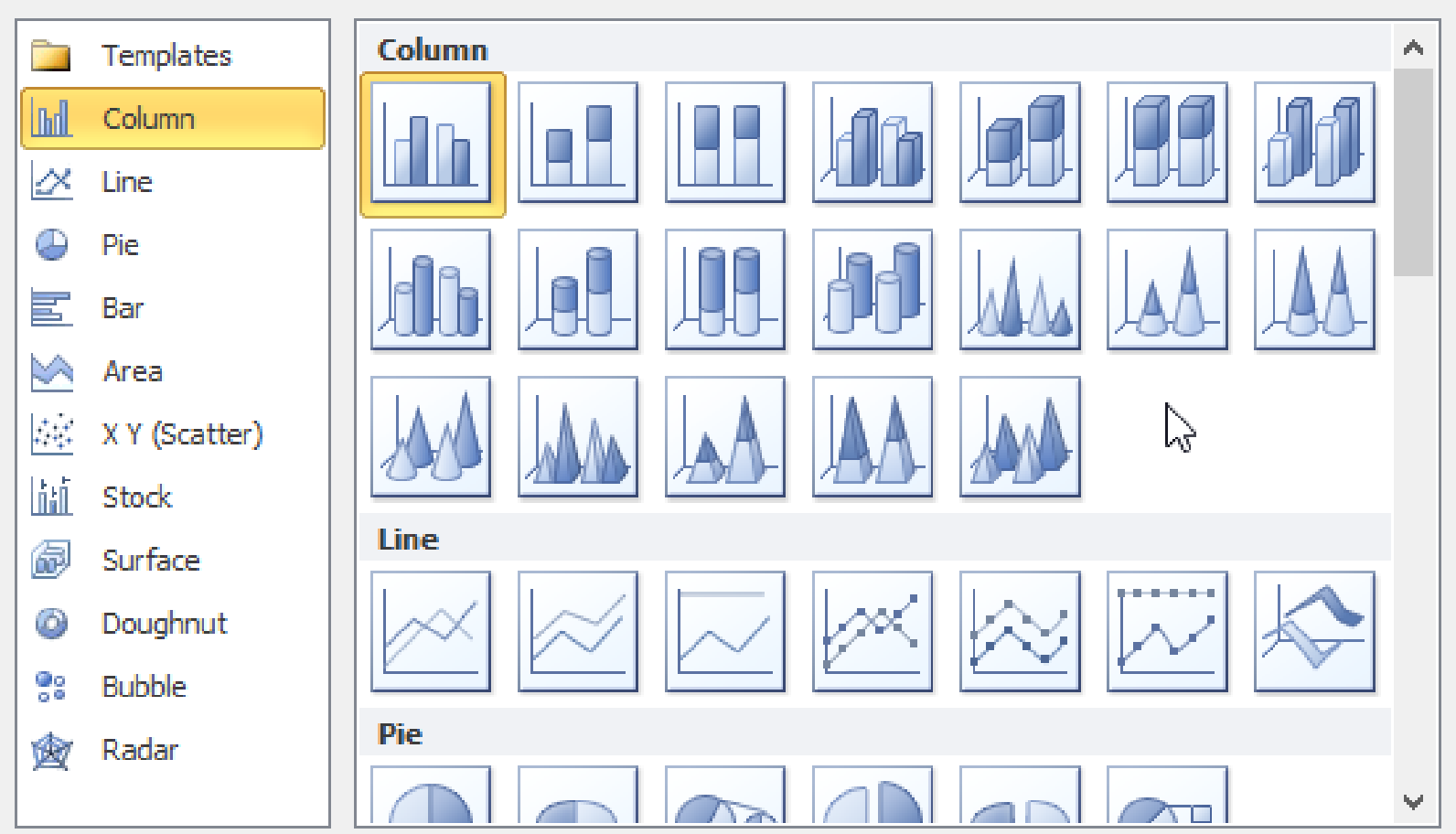


Các phương pháp trình bày

- Data visualisation
  - Các kĩ thuật hiển thị lượng lớn thông tin.
  - Hình ảnh có thể cho thấy quan hệ giữa các thực thể và các xu hướng của dữ liệu.
  - Các ví dụ:
    - Thông tin thời tiết thu thập từ nhiều nguồn;
    - Trạng thái của một mạng điện thoại dưới dạng một tập các nút được kết nối với nhau;
    - Một mô hình phân tử hiển thị dạng ba chiều;
    - ...

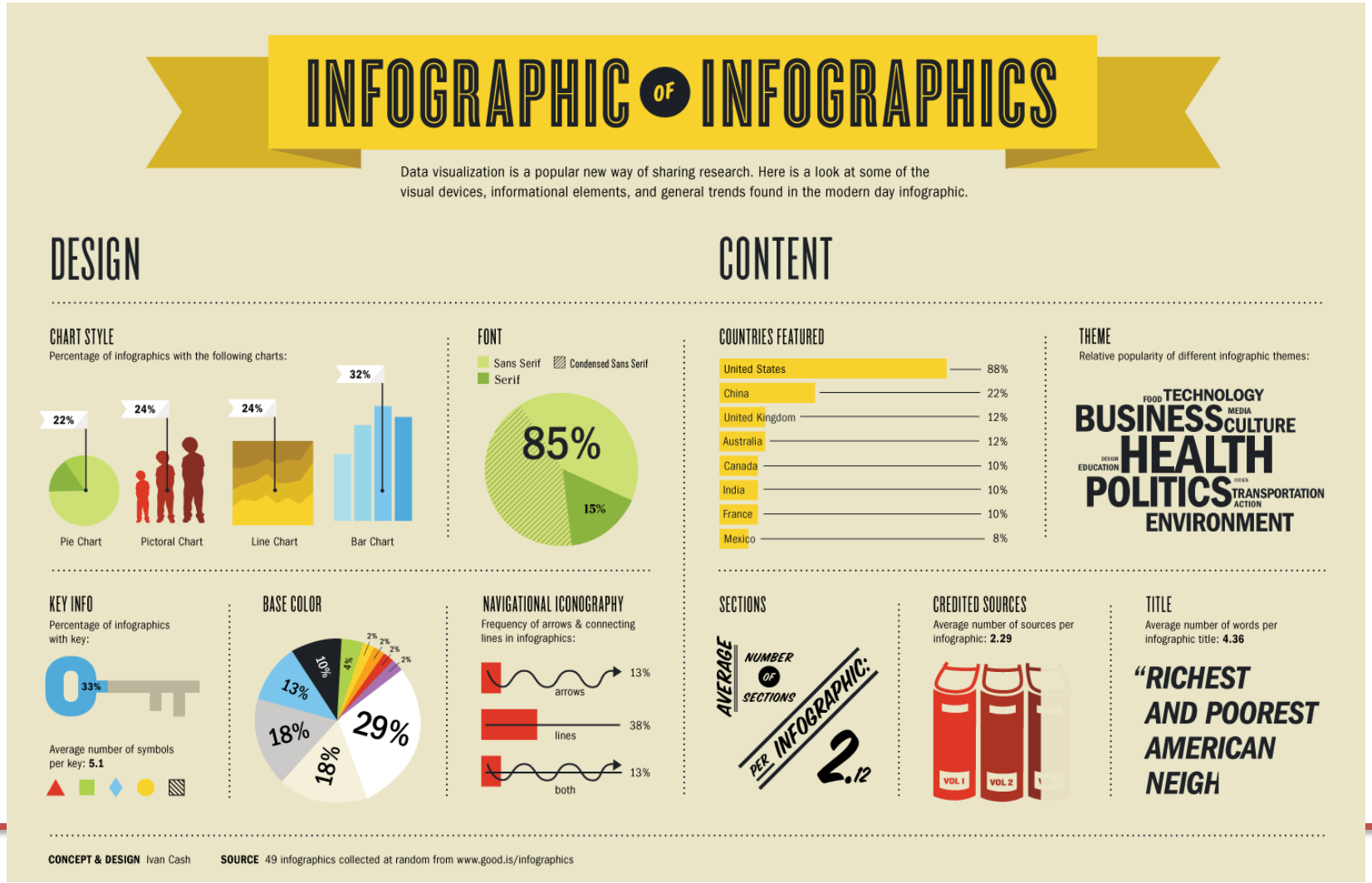
Các phương pháp trình bày

- Data visualisation



Các phương pháp trình bày

- Data visualisation



Các phương pháp trình bày

- Data visualisation

Hiển thị màu

- Màu sắc bổ sung thêm một chiều cho một giao diện và giúp người dùng hiểu các cấu trúc thông tin phức tạp.
- Có thể dùng màu để highlight các thông tin đặc biệt.
- Các lỗi thường gặp trong việc dùng màu sắc trong thiết kế UI:
  - Dùng màu sắc để diễn đạt ý nghĩa.
  - Lạm dụng màu sắc trong trình bày.

Hiển thị màu

- Hướng dẫn về dùng màu
  - Hạn chế số màu và mức độ sặc sỡ.
  - Dùng sự thay đổi màu để báo hiệu thay đổi trạng thái hệ thống.
  - Dùng kí hiệu màu (color coding) để hỗ trợ công việc người dùng đang cố làm.
    - Highlight những điểm người dùng cần chú ý.
  - Dùng kí hiệu màu một cách cẩn trọng và nhất quán.
    - Nếu màu đỏ được dùng cho các thông báo lỗi không nên dùng màu đỏ cho các thông báo dạng khác để tránh người dùng nhầm lẫn các thông báo màu đỏ là thông báo lỗi.
  - Cẩn thận về hiệu ứng cặp đôi của màu sắc.
    - Một số tổ hợp màu gây khó đọc (ví dụ: không thể cùng lúc chú ý cả màu đỏ và màu xanh lam).

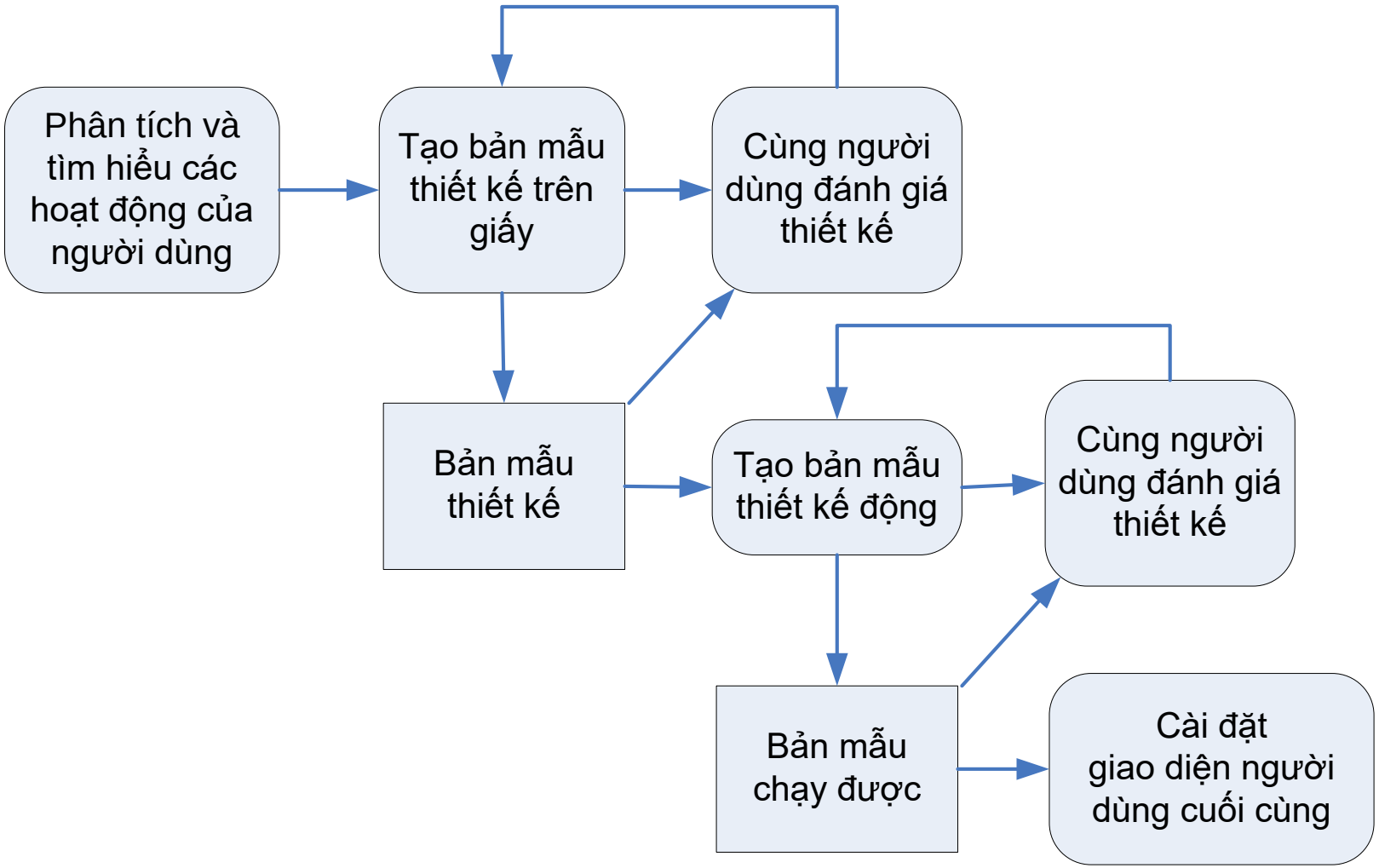
Các thông báo lỗi

- Thiết kế thông báo lỗi là việc rất quan trọng
  - Các thông báo lỗi kém có thể dẫn đến việc người dùng không chấp nhận sử dụng một hệ thống.
- Các thông báo nên lịch sự, ngắn gọn, nhất quán và mang tính xây dựng.
- Nên xét đến background và kinh nghiệm của người dùng như là nhân tố quyết định khi thiết kế các thông báo lỗi.

Quy trình thiết kế UI

- Thiết kế UI là một quy trình lặp đi lặp lại với sự liên lạc chặt chẽ giữa người dùng và người thiết kế.
- Ba hoạt động chính trong quy trình:
  - User analysis: tìm hiểu người dùng sẽ làm gì với hệ thống;
  - System prototyping: phát triển một loạt các bản mẫu để thử nghiệm;
  - Interface evaluation: thử nghiệm các bản mẫu cùng với người dùng.

Quy trình thiết kế UI



📖 Quy trình thiết kế UI

- Bản mẫu UI
  - Mục tiêu của bản mẫu là cho phép người dùng có được trải nghiệm trực tiếp đối với giao diện.
  - Nếu không có trải nghiệm trực tiếp, không thể đánh giá khả năng sử dụng một giao diện.
  - Quy trình làm bản mẫu có thể có 2 bước:
    - Lúc đầu, có thể dùng bản mẫu trên giấy.
    - Sau đó tinh chỉnh thiết kế, và phát triển các bản mẫu tự động hóa với độ phức tạp ngày càng tăng.

Các thuộc tính usability

Thuộc tính	Miêu tả
Khả năng học	Người dùng mới cần bao lâu để có thể hoạt động hiệu quả với hệ thống?
Tốc độ vận hành	Tốc độ phản ứng của hệ thống có đáp ứng tốt công việc của người dùng?
Chịu lỗi	Mức độ dung thứ lỗi của hệ thống đối với lỗi người dùng.
Khả năng khôi phục	Khả năng hệ thống khôi phục từ lỗi của người dùng.
Tương thích	hệ thống gắn bó chặt chẽ với một kiểu làm việc đến đâu?

Các kĩ thuật đánh giá

- Câu hỏi điều tra để lấy phản hồi của người dùng.
- Quay video về việc sử dụng hệ thống rồi sau đó đánh giá nội dung.
- Cài các đoạn mã thu thập thông tin về các tiện ích được sử dụng và lỗi của người dùng.
- Phần mềm có chức năng thu thập phản hồi trực tuyến của người dùng.

Một số nguyên tắc trong thiết kế giao diện

- Tất cả màn hình phải có tên.
- Thiết kế phù hợp với đối tượng sử dụng.
- Dễ học, dễ nhớ, phù hợp với người mới sử dụng, đồng thời hỗ trợ các cách làm nhanh, làm tắt cho người sử dụng có kinh nghiệm.
- Thứ tự trình bày trên màn hình phải phù hợp với văn hóa, thói quen của người sử dụng
- Chỉ trình bày những nội dung thật sự cần thiết, không trình bày quá nhiều thông tin trên một màn hình.
- Chọn font chữ rõ ràng, cỡ chữ phù hợp.
- Kết hợp màu nền và màu chữ hợp lý, không dùng quá nhiều màu sắc trên một màn hình.

3.5. Thiết kế dữ liệu

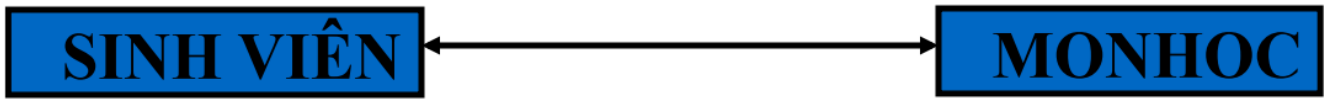
- Mục tiêu của việc thiết kế dữ liệu là nhằm mô tả cách thức tổ chức lưu trữ dữ liệu của phần mềm bên trong máy tính.
- Khi thiết kế dữ liệu, cần quan tâm đến ba vấn đề sau:
  - Thiết kế dữ liệu với tính đúng đắn
  - Thiết kế dữ liệu với tính tiến hóa
  - Thiết kế dữ liệu với yêu cầu hiệu quả về mặt truy xuất và lưu trữ
- Tổ chức lưu trữ
  - Bảng/Tập tin
  - Thuộc tính/Cấu trúc
  - Liên kết giữa các bảng/Tập tin

Sơ đồ Logic

- Một phần tử của bảng A xác định duy nhất một phần tử của bảng B
- Ngược lại, một phần tử của bảng B có thể tương ứng với một hoặc nhiều phần tử của bảng A
- Bảng A chứa thuộc tính khóa của bảng B (là khóa ngoại của bảng A và là khóa chính của bảng B).

Sơ đồ Logic

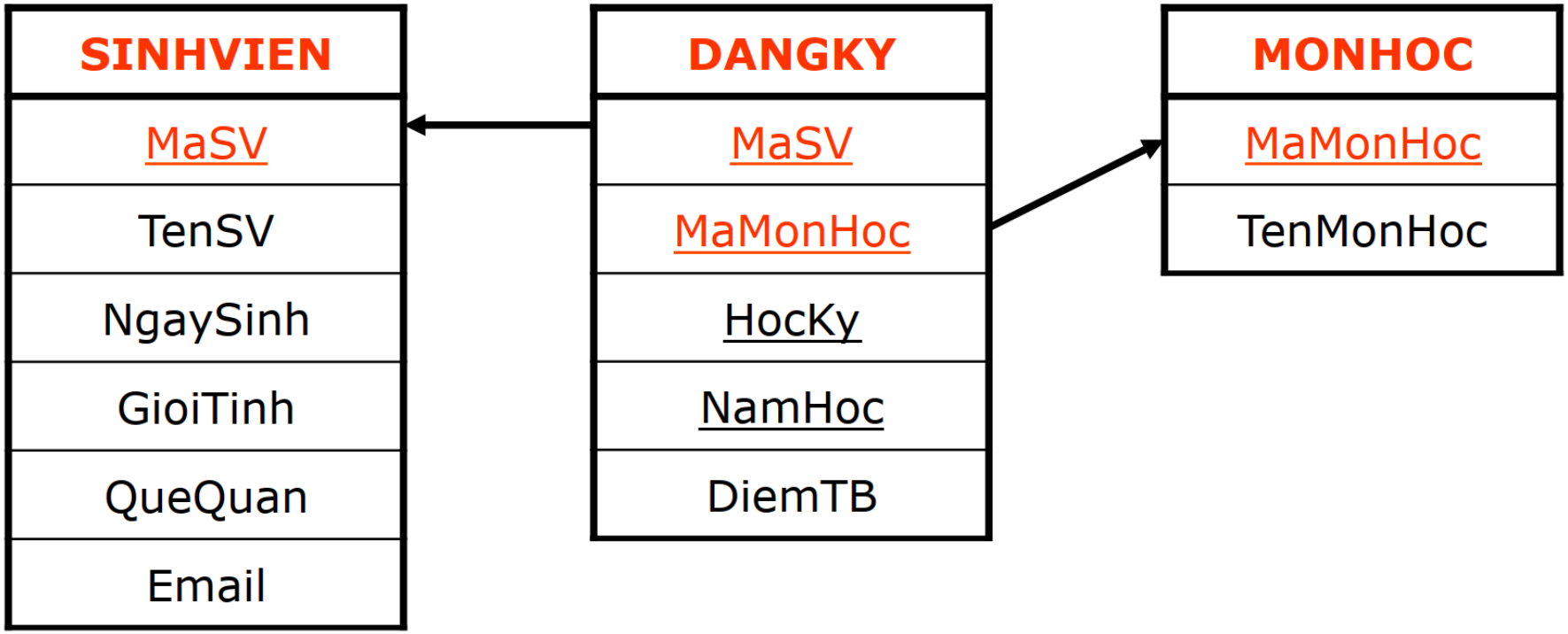
- Nếu quan hệ giữa A và B là quan hệ 1-1 thì có thể gộp hai table A và B lại thành 1 table duy nhất chứa tất cả thuộc tính của A và B.
- Nếu quan hệ giữa A và B là quan hệ n-n:
  - Tách quan hệ này thành 2 quan hệ 1-n bằng cách thêm vào 1 table trung gian chứa khóa chính của A và B.
  - Ví dụ:



Chuyển thành:

Sơ đồ Logic

- Ví dụ quan hệ n-n



Xác định khóa chính

- 3 Tính chất của khoá chính:
  - Tối thiểu.
  - Không trùng lặp (bao gồm NOT NULL).
  - Không thay đổi theo thời gian.
- Thuộc tính trừu tượng:
  - Là thuộc tính không xuất hiện trong thế giới thực, chỉ có trong phần mềm.
  - Sử dụng thuộc tính trừu tượng để làm khoá chính cho table.
  - Ví dụ: MaDaiLy, MaLoaiDaiLy…
- Xác định kiểu dữ liệu cho thuộc tính khóa:
  - Cân nhắc lựa chọn giữa kiểu số và kiểu chuỗi.
  - Sử dụng tối ưu chiều dài của mã đồng thời phải xem xét khả năng mở rộng.

Lưu ý khi đặt tên

- Tên Table: viết bằng chữ IN HOA, không dấu, không có khoảng cách giữa các từ.
  - Ví dụ: NHANVIEN, KHACHHANG…
- Tên thuộc tính: viết hoa các ký tự đầu của mỗi từ, không dấu, không có khoảng cách giữa các từ.
  - Ví dụ: HoTen, NgaySinh, DiaChi…
- Đặt tên table, tên thuộc tính của table súc tích, cô đọng và nhất quán trong toàn bộ CSDL.

Thiết kế dữ liệu

- Danh sách bảng

STT	Bảng	Ý nghĩa	Ghi chú
1			
...			

- Mô tả bảng

STT	Tên thuộc tính	Kiểu	Ràng buộc	Giá trị Khởi động	Ghi chú
1					
...					

3.5. Thiết kế hướng đối tượng với UML

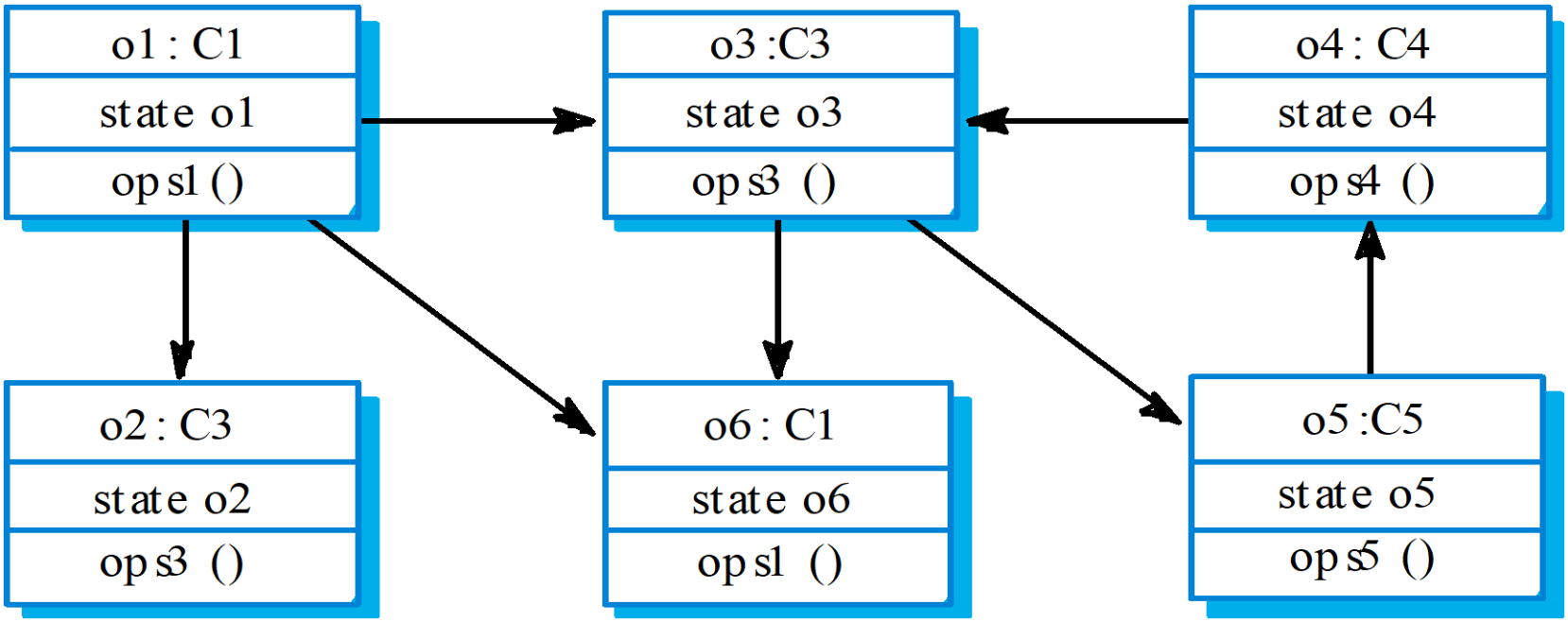
- Phân tích, thiết kế, lập trình hướng đối tượng có sự liên quan đến nhau nhưng không giống nhau
  - Phân tích hướng đối tượng: phát triển một mô hình đối tượng của miền ứng dụng
  - Thiết kế hướng đối tượng: phát triển mô hình hệ thống hướng đối tượng → cài đặt OOP
  - Lập trình hướng đối tượng: sử dụng ngôn ngữ OOP (Java, C++...) để xây dựng chương trình theo thiết kế hướng đối tượng

Phát triển hướng đối tượng

- Đặc điểm của OOD
  - Đối tượng là sự khái quát hóa các thực thể của thế giới thực và các thông tin của chúng.
  - Đối tượng bao gồm các thuộc tính và các hành vi
  - Không có sự chia sẻ dữ liệu chung, các đối tượng trao đổi dữ liệu qua truyền message.
  - Các đối tượng có thể phân tán, hoạt động tuần tự hoặc song song.

Phát triển hướng đối tượng

- Tương tác giữa các đối tượng

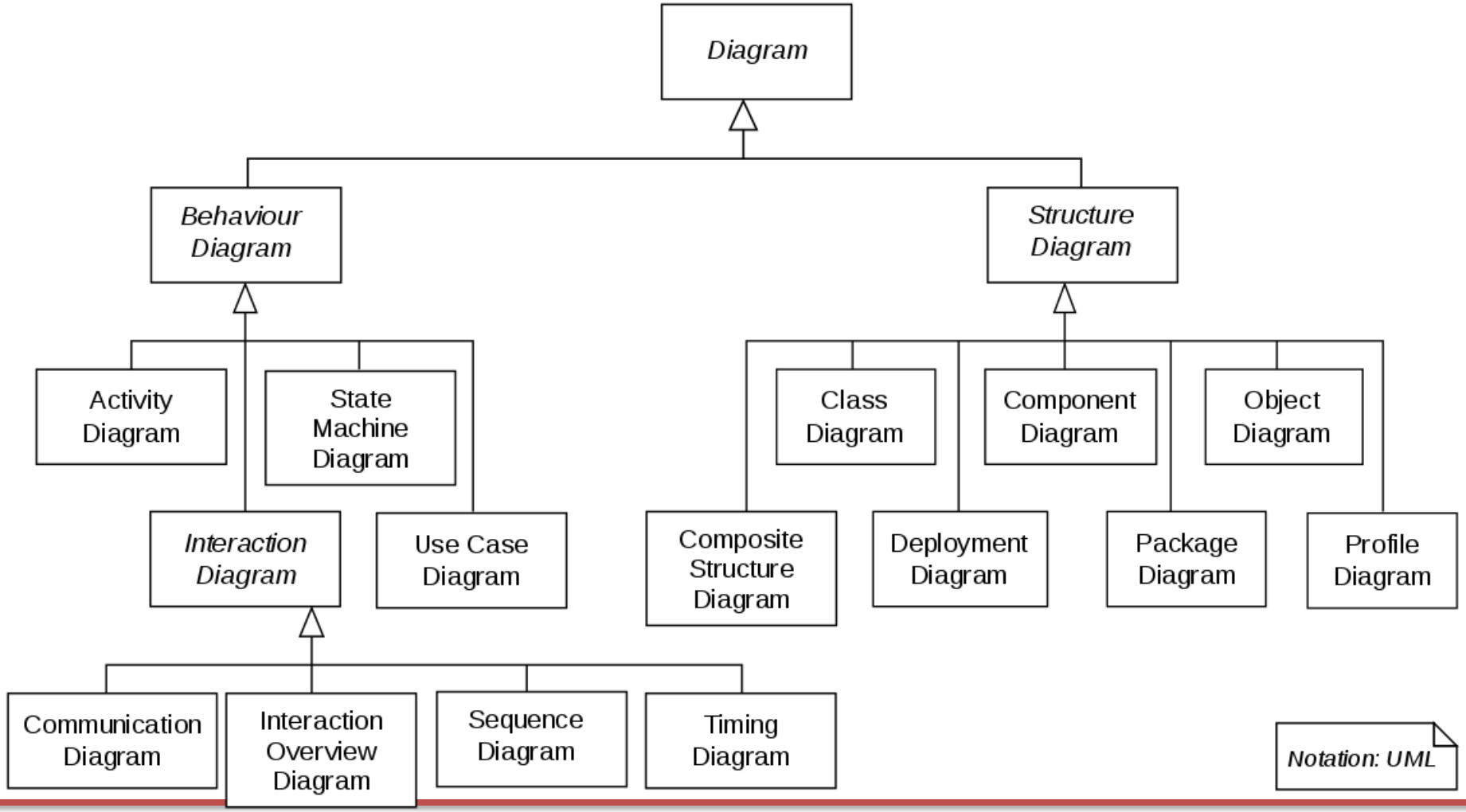


UML (Unified Modeling Language)

- Là một ngôn ngữ mô hình hoá, giúp xây dựng các mô hình chính xác, đầy đủ và không nhập nhằng.
- Tất cả các công đoạn từ phân tích, thiết kế cho đến triển khai đều có các biểu đồ UML biểu diễn.
- Mục tiêu: cung cấp cho người thiết kế, kỹ sư phần mềm, người phát triển hệ thống một công cụ để phân tích, thiết kế và cài đặt các hệ thống phần mềm cũng như để mô hình hoá quy trình nghiệp vụ.
- Một số Case tool (công cụ) hỗ trợ UML: Rational Rose, Visual Paradiagm, Power designer, Star UML…

Biểu đồ UML

- Có 15 loại biểu đồ khác nhau

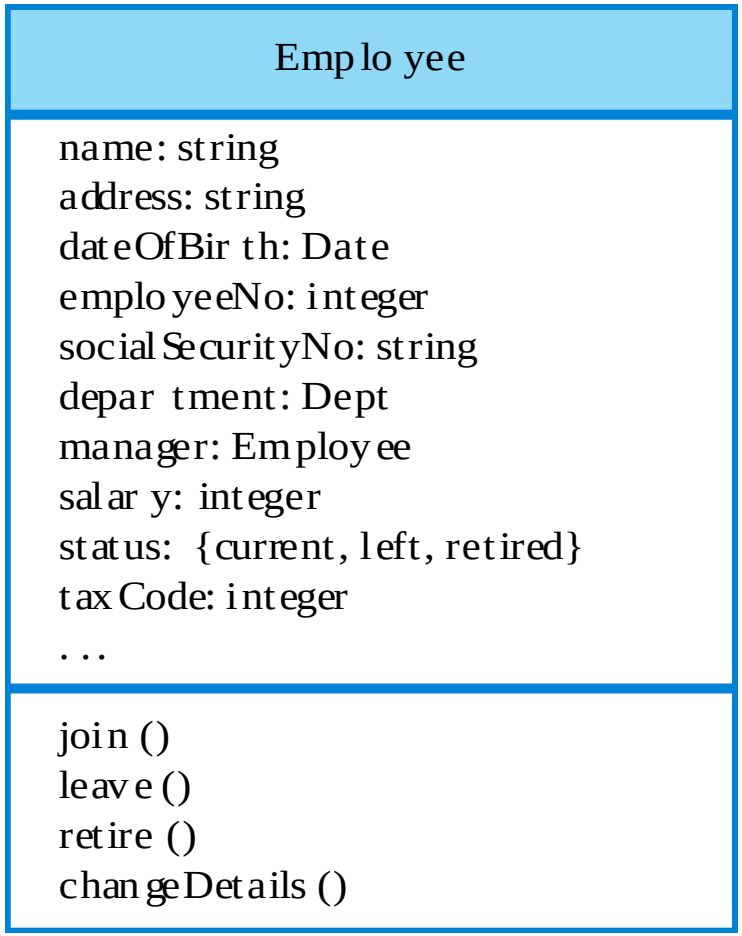


Biểu đồ UML

- Các biểu đồ UML thường dùng:
  - Biểu đồ hoạt động (activity diagram): Chỉ ra các hoạt động trong một quy trình hay trong việc xử lý dữ liệu.
  - Biểu đồ use case (use case diagram): Chỉ ra các tương tác giữa một hệ thống và môi trường của nó.
  - Biểu đồ tuần tự (sequence diagram): Chỉ ra các tương tác giữa các actor và hệ thống, và giữa các component của hệ thống với nhau.
  - Biểu đồ lớp (class diagram): Chỉ ra các lớp đối tượng trong hệ thống và các quan hệ giữa các lớp này.
  - Biểu đồ trạng thái (state diagram): Chỉ ra hệ thống tương tác với các sự kiện bên trong và bên ngoài như thế nào.

Biểu đồ UML

- Employee object class (UML)

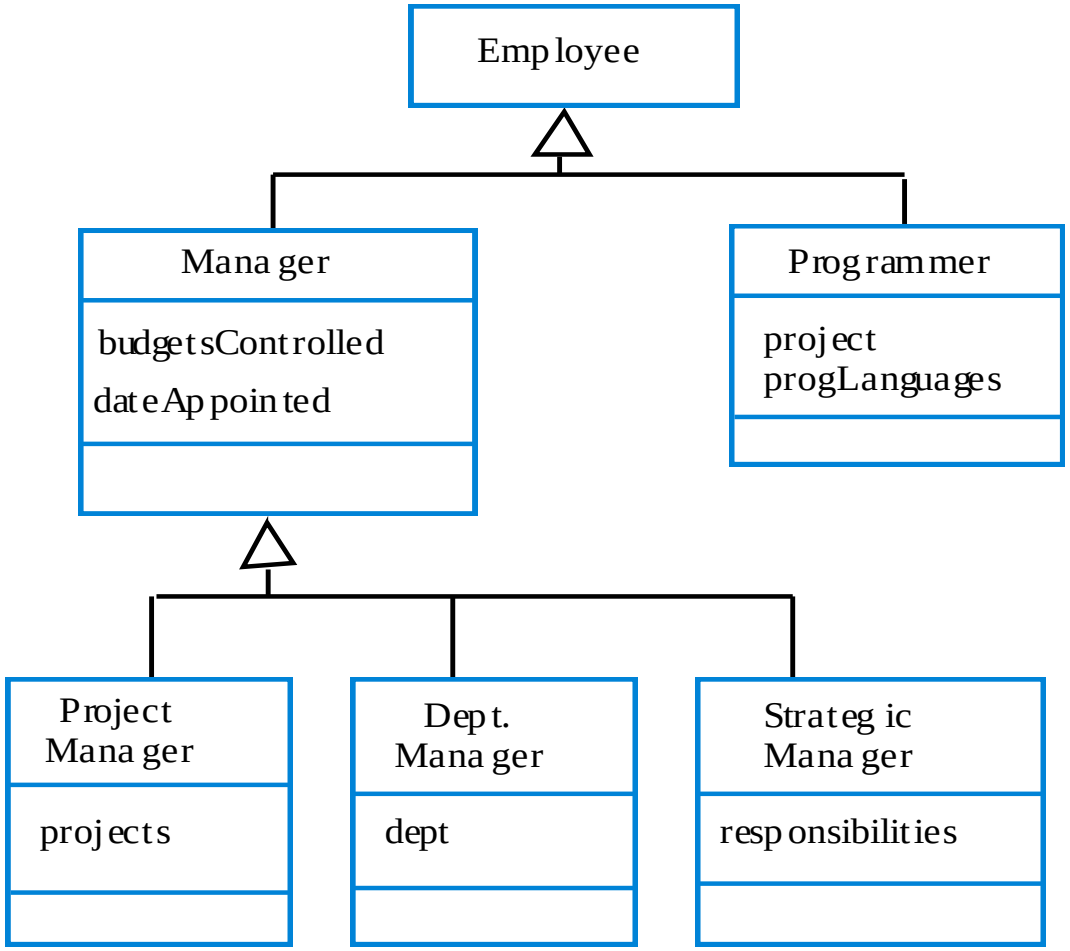


Generalisation vs inheritance

- Đối tượng là các thể hiện của Class (thuộc tính + phương thức)
- Classes thường được sắp trong cây kế thừa với các quan hệ kế thừa (super-class and sub-class)
- Generalisation trong UML được thể hiện như kế thừa trong lập trình hướng đối tượng

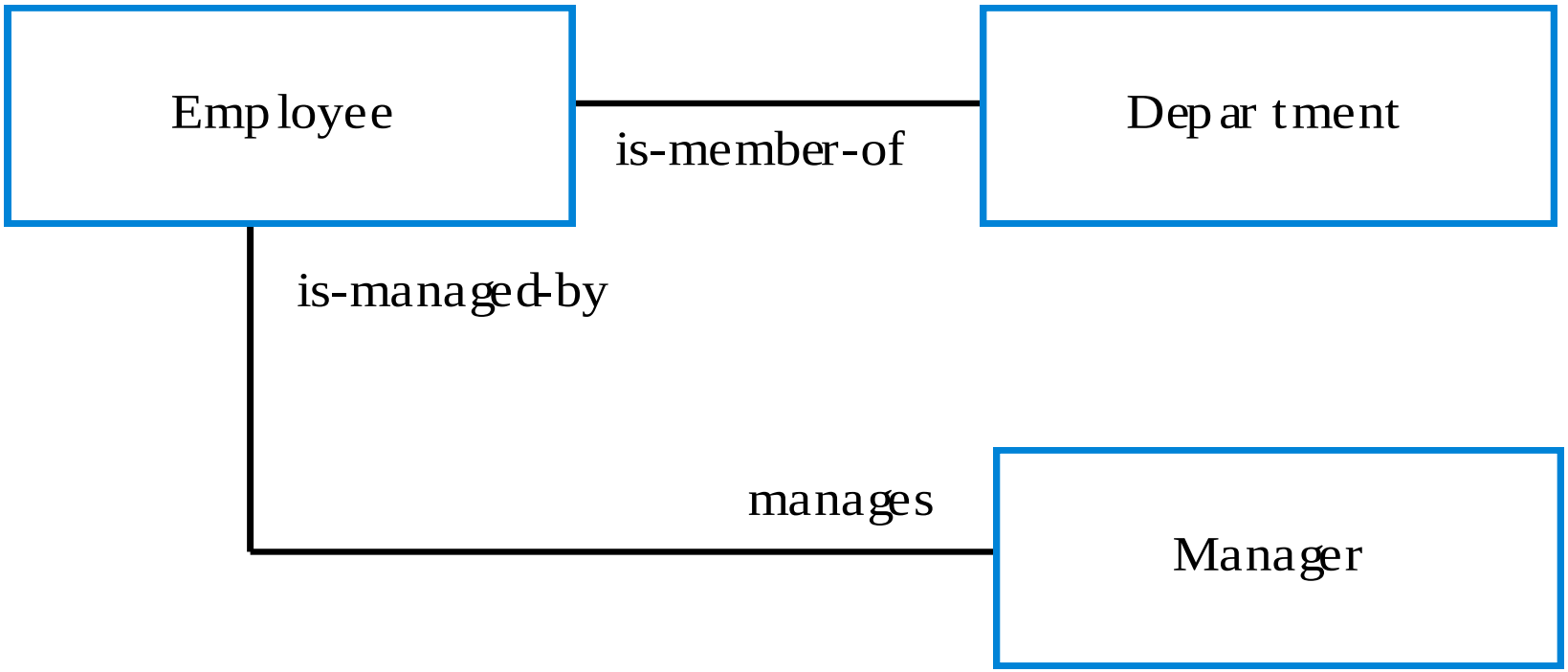
Generalisation vs inheritance

- Ví dụ về Generalisation



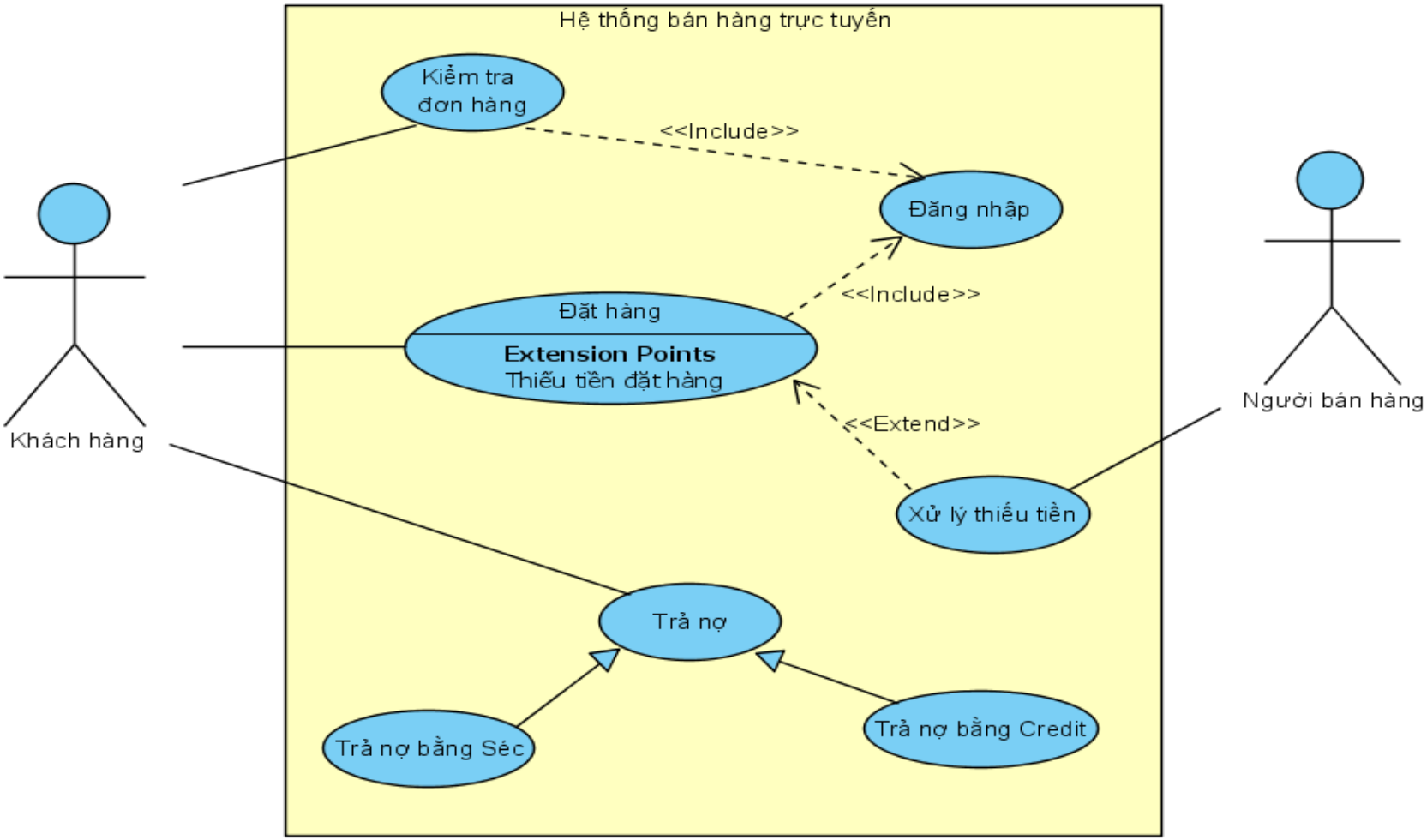
Generalisation vs inheritance

- Quan hệ giữa các lớp



Use case

- Biểu diễn các tương tác trong hệ thống



📖 Phát hiện lớp

- Là phần khó nhất trong thiết kế OOD
- Không có quy tắc chuẩn mực nào cho việc phát hiện lớp, chủ yếu dựa vào kỹ năng và miền hiểu biết của nhà thiết kế.
- Đó là một tiến trình lặp: tìm class → xác định liên kết giữa các class → thuộc tính → xây dựng lớp kế thừa → làm mịn → phân tách module.
- Phương pháp:
  - Sử dụng các quy tắc ngữ pháp trong mô tả ứng dụng bằng ngôn ngữ tự nhiên (danh từ → obj, hành vi → method...)
  - Dựa vào những thứ hữu hình để xác định classes
  - Dựa vào kịch bản → objects, attributes, methods trong mỗi kịch bản được xác định.

3.6. Các vấn đề của việc thực thi phần mềm

- Cài đặt?
  - Là quá trình chuyển đổi từ thiết kế chi tiết sang mã lệnh.
- Lựa chọn ngôn ngữ lập trình:
  - Phụ thuộc vào cấu hình máy
  - Phụ thuộc vào số lượng ngôn ngữ lập trình sẵn có
  - Phụ thuộc vào thói quen sử dụng ngôn ngữ lập trình
  - Phụ thuộc vào khách hàng
  - …
- Đánh giá rủi ro khi chọn ngôn ngữ lập trình

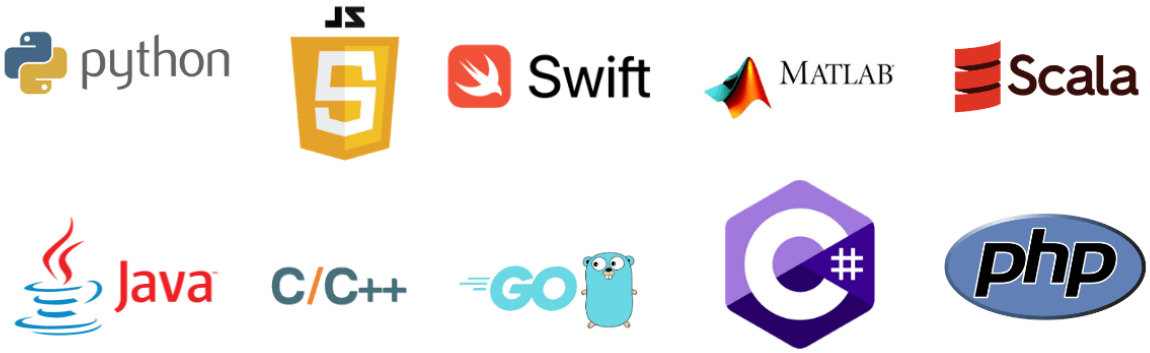
Ngôn ngữ lập trình

- Top Programming Languages 2020 - IEEE Spectrum

Rank	Language	Type	Score
1	Python		100.0
2	Java		95.3
3	C		94.6
4	C++		87.0
5	JavaScript		79.5
6	R		78.6
7	Arduino		73.2
8	Go		73.1
9	Swift		70.5
10	Matlab		68.4

Ngôn ngữ lập trình

Top 10 Programming Languages 2020



Top Paying and Most Popular Programming Languages in 2020

Rank by Average Salary	
1. Python	$119,000
2. JavaScript	$117,000
3. Java	$104,000
4. C	$103,000
5. C++	$102,000
6. C#	$97,000
7. PHP	$94,000
8. SQL	$92,000

Rank by Volume of Job Openings	
1. Python	50,000
2. SQL	50,000
3. Java	45,000
4. JavaScript	38,000
5. C++	29,000
6. C#	21,000
7. PHP	13,000
8. C	9,000

3.7. Lập trình hiệu quả

- Phong cách lập trình
  - Là một triết lý về lập trình, nhấn mạnh vào yếu tố dễ hiểu của source code.
  - Các yếu tố gồm: tài liệu trong chương trình, phương pháp khai báo dữ liệu, câu lệnh, input/output.
    - Tài liệu: quy tắc đặt tên, lời chú thích đầu mỗi module (mục đích, mô tả interface, dữ liệu thích hợp, tên người viết, người sửa đổi, ngày tháng…)
    - Khai báo dữ liệu: dựa theo tài liệu coding convention quy định về quy tắc đặt tên kiểu dữ liệu, chú thích ý nghĩa…

Một số nguyên tắc lập trình cơ bản

- Công nghệ là cách tìm ra giải pháp, chứ không phải giải pháp
- Thông minh là kẻ thù của sự rõ ràng
- Đừng viết code thừa, hãy viết đúng những gì cần viết
- Lạm dụng Comment trong code đa phần là không tốt
- Luôn biết code phải làm gì trước khi bắt đầu viết nó
- Kiểm tra lại code trước khi bàn giao nó
- Học kiến thức mới mỗi ngày
- Viết code là chuyện rất thú vị
- Chấp nhận rằng không phải thứ gì mình cũng biết
- Cách giải quyết tốt nhất còn tùy vào từng tình huống
- Luôn hướng đến sự đơn giản

📖 Phong cách lập trình

- Bao gồm các yếu tố
  - Cách đặt tên hàm và biến
  - Cách xây dựng câu lệnh, cấu trúc chương trình
  - Cách viết chú thích
- Hướng tới phong cách làm cho mã nguồn
  - Dễ hiểu, dễ sửa đổi
  - An toàn (ít lỗi)
- Người khác có thể hiểu được, bảo trì được

📖 Phong cách lập trình

- Câu lệnh: nên đơn giản, rõ ràng
  - Tránh các phép kiểm tra điều kiện phức tạp
  - Tránh lồng nhiều các kiểm tra điều kiện, chu trình lặp
  - Sử dụng dấu ngoặc để sáng tỏ biểu thức toán/logic
  - Chỉ dùng tính năng chuẩn của ngôn ngữ, không dùng câu lệnh phức tạp
- Vào/ra:
  - Đầu vào hợp lệ
  - Kiểm tra sự tin cậy của input
  - Giữ định dạng input đơn giản
  - …

📖 Phong cách lập trình

- Lý do cần dễ hiểu:
  - Phần mềm luôn cần sửa đổi
    - Sửa lỗi
    - Nâng cấp
  - Kéo dài tuổi thọ, nâng cao hiệu quả kinh tế
  - Nếu không dễ hiểu
    - Bảo trì tốn thời gian, chi phí cao
    - Tác giả phải bảo trì suốt vòng đời của phần mềm
    - Bản thân tác giả cũng không hiểu

Phong cách lập trình

- Chú thích
  - Mục đích sử dụng của các biến
  - Chức năng của khối lệnh, câu lệnh
    - Các lệnh điều khiển
    - Các lệnh phức tạp
  - Chú thích các module
    - Mục đích, chức năng của module
    - Tham số, giá trị trả lại (giao diện)
    - Các module cấp con
    - Cấu trúc, thuật toán
    - Nhiệm vụ của các biến cục bộ
    - Tác giả, người kiểm tra, thời gian

📖 Phong cách lập trình

- Đặt tên
  - Đặt tên biến, tên hàm có nghĩa, gợi nhớ
  - Sử dụng các ký hiệu, từ tiếng Anh có nghĩa
  - Làm cho dễ đọc
    - Dùng DateOfBirth hoặc date_of_birth
    - Không viết dateofbirth
  - Tránh đặt tên quá dài
    - Không đặt tên dài với các biến cục bộ
  - Thống nhất cách dùng
    - Tên lớp bắt đầu bằng chữ hoa, tên hằng toàn chữ hoa
    - Tên biến bắt đầu bằng chữ thường
    - i cho vòng lặp, tmp cho các giá trị tạm thời...

📖 Phong cách lập trình

- Câu lệnh:
  - Các câu lệnh phải mô tả cấu trúc
    - Tụt lề, dễ đọc, dễ hiểu
  - Làm đơn giản các lệnh
    - Mỗi lệnh trên một dòng
    - Triển khai các biểu thức phức tạp
    - Hạn chế truyền tham số là kết quả của hàm, biểu thức (printf("%s", strcpy(des, src));)
  - Tránh các cấu trúc phức tạp:
    - Các lệnh If lồng nhau
    - Điều kiện phủ định If not

📖 Phong cách lập trình

- Hàm và biến cục bộ
  - Chương trình cần được chia thành nhiều module (hàm)
  - Không viết hàm quá dài
    - Không quá 2 trang màn hình
    - Tạo ra các hàm thứ cấp để giảm độ dài từng hàm
  - Không dùng quá nhiều biến cục bộ
    - Không thể theo dõi đồng thời hoạt động của nhiều biến

📖 Phong cách lập trình

- Lỗi phần mềm
  - Có thể phát hiện lỗi trong khi thực hiện
    - Lỗi chia 0
    - Lỗi input/output
    - …
  - Xử lý lỗi
    - Nhất quán trong xử lý: phân loại lỗi, thống nhất định dạng thông báo…
    - Phân biệt output và thông báo lỗi
    - Các hàm thư viện nên tránh việc tự xử lý, tự đưa ra thông báo lỗi

Lập trình tránh lỗi

- Tránh lỗi dựa trên các yếu tố:
  - Sản phẩm của một đặc tả hệ thống chính xác
  - Thiết kế phần mềm dựa trên đóng gói dữ liệu và che giấu thông tin
  - Tăng cường duyệt lại, thẩm định trong quá trình phát triển
  - Chấp nhận triết lý: chất lượng là bánh lái của quá trình SE
  - Lập kế hoạch cẩn thận cho việc thử nghiệm hệ thống → tìm ra lỗi.

Lập trình tránh lỗi

- Lập trình có cấu trúc: không sử dụng goto, phát triển top-down → tăng tính tuần tự, dễ kiểm soát.
- Một số yếu tố dễ gây lỗi:
  - Số thực dấu chấm động → dễ sai khi làm tròn…
  - Con trỏ, bộ nhớ động
  - Song song
  - Đệ quy
  - Ngắt
- Phân quyền truy cập dữ liệu: che giấu cấu trúc thông tin…

Lập trình thử lỗi

- Một số hệ thống tin cậy đòi hỏi hoạt động tốt ngay khi có những thành phần bị lỗi → khả năng dung thứ lỗi.
- Hệ thống thứ lỗi cần:
  - Phát hiện lỗi
  - Định mức thiệt hại
  - Phục hồi sau khi gặp lỗi: quay lui về trạng thái an toàn trước đó
  - Chữa lỗi: cải tiến hệ thống không gặp lỗi đó nữa

Lập trình hướng hiệu quả

- Hiệu quả về chương trình: thuật toán nhanh, thiết kế tốt
  - Đơn giản hóa các biểu thức toán học
  - Tránh việc dùng con trỏ và danh sách phức tạp
  - ….
- Hiệu quả về bộ nhớ
- Hiệu quả vào/ra:
  - Số yêu cầu vào/ra nên tối thiểu
  - Vào/ra nên thông qua bộ đệm

📖 Câu hỏi thảo luận