

NHẬP MÔN CÔNG NGHỆ PHẦN MỀM (INTRODUCTION TO SOFTWARE ENGINEERING)

Thiết kế phần mềm

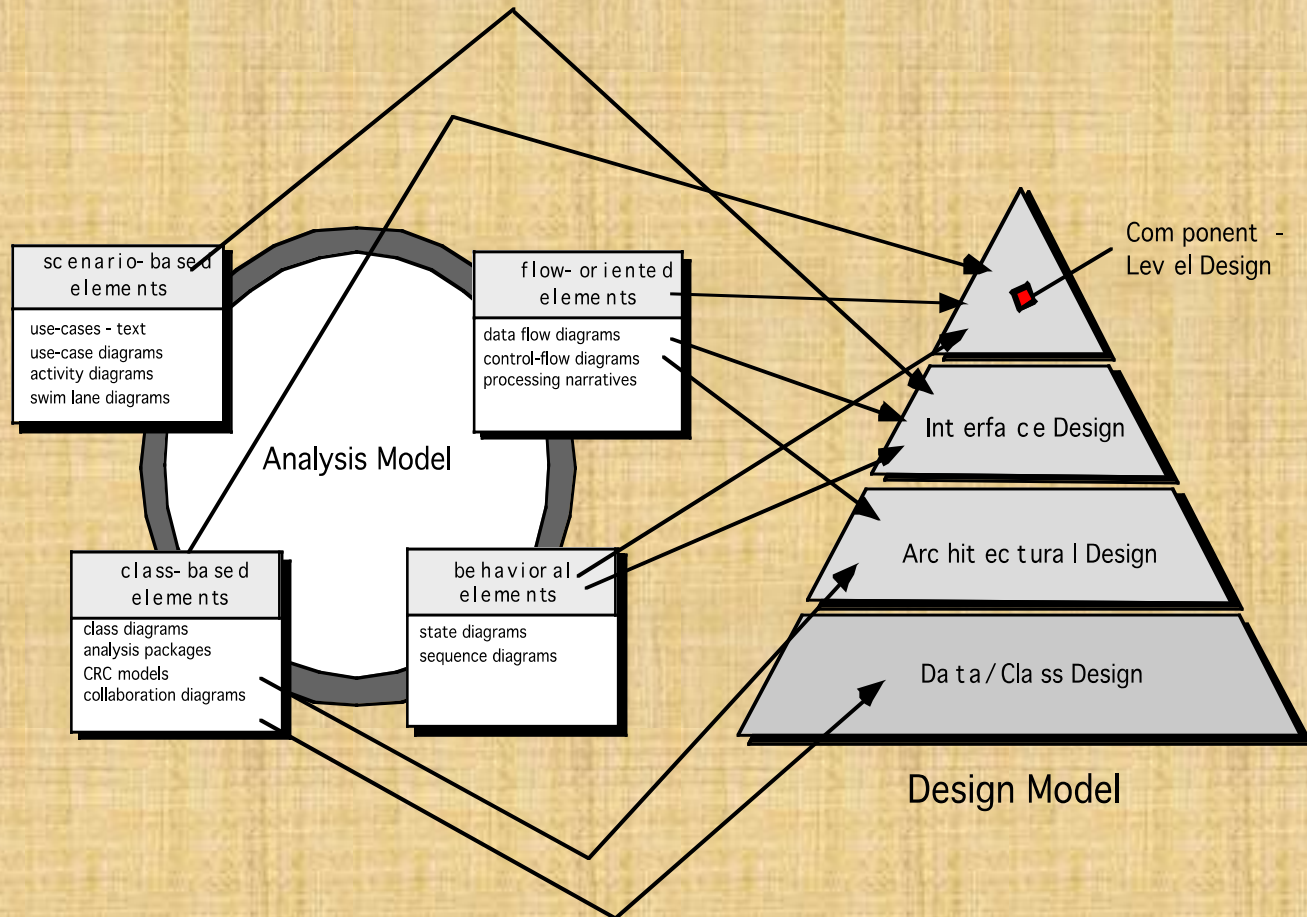
- Nội dung
 1. **Tổng quan thiết kế phần mềm**
 2. Thiết kế kiến trúc phần mềm
 3. Thiết kế chi tiết phần mềm
 4. Thiết kế giao diện người dùng
 5. Thiết kế mẫu
 6. Thiết kế giao diện cho ứng dụng WebApp

Design

- Theo Mitch Kapor, người đã tạo ra Lotus 1-2-3, giới thiệu trong “Tuyên ngôn về thiết kế phần mềm” trên Dr. Dobbs Journal. :
 - Thiết kế phần mềm tốt nên thể hiện
 - **Sự ổn định (Firmness)**: Một chương trình không nên có bất cứ lỗi nào làm hạn chế chức năng của nó.
 - **Tiện nghi(Commodity)**: Một chương trình nên phù hợp với mục đích đã định của nó .
 - **Sự hài lòng(Delight)**: Trải nghiệm sử dụng chương trình nên làm hài lòng người dùng.

Analysis Model -> Design Model

(Mô hình phân tích-> Mô hình thiết kế)



These slides are designed to accompany Software Engineering:
A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Thiết kế và chất lượng

- **thiết kế phải thực hiện tất cả các yêu cầu rõ ràng** chứa trong mô hình phân tích, và nó phải đáp ứng tất cả các yêu cầu tiềm ẩn khách hàng mong muốn .
- **thiết kế phải là một hướng dẫn dễ hiểu dễ đọc** cho những người tạo ra code và cho những người kiểm thử và sau đó hỗ trợ cho phần mềm.
- **thiết kế nên cung cấp một bức tranh hoàn chỉnh của phần mềm**, giải quyết vấn đề dữ liệu, chức năng, và hành vi từ một quan điểm thực thi.

Nguyên tắc Chất lượng

- Một thiết kế nên thể hiện một kiến trúc mà (1) đã được tạo ra bằng cách sử dụng phong cách kiến trúc hoặc các pattern được công nhận, (2) bao gồm các thành phần mang những đặc tính thiết kế tốt và (3) có thể được thực hiện một cách tiến hóa
 - Đối với các hệ thống nhỏ hơn, thiết kế đôi khi có thể được phát triển tuyến tính.
- Một thiết kế nên môđun hoá; đó là, phần mềm nên được phân chia thành các thành phần hợp lý hoặc hệ thống con
- Một thiết kế cần có biểu diễn riêng biệt của dữ liệu, kiến trúc, giao diện, và các thành phần.
- Một thiết kế nên dẫn đến các cấu trúc dữ liệu thích hợp cho các lớp sẽ được thực thi và được rút ra từ mô hình dữ liệu có thể nhận biết.
- Một thiết kế nên dẫn đến các thành phần mang những đặc tính chức năng độc lập.
- Một thiết kế nên dẫn đến giao diện mà giảm sự phức tạp của các kết nối giữa các thành phần và với môi trường bên ngoài
- Một thiết kế nên được chuyển hoá bằng cách sử dụng một phương pháp lặp lại được dẫn dắt bởi các thông tin thu được trong quá trình phân tích các yêu cầu phần mềm.
- Một thiết kế nên được đại diện bằng một ký hiệu truyền đạt hiệu quả ý nghĩa của nó.

Nguyên tắc thiết kế

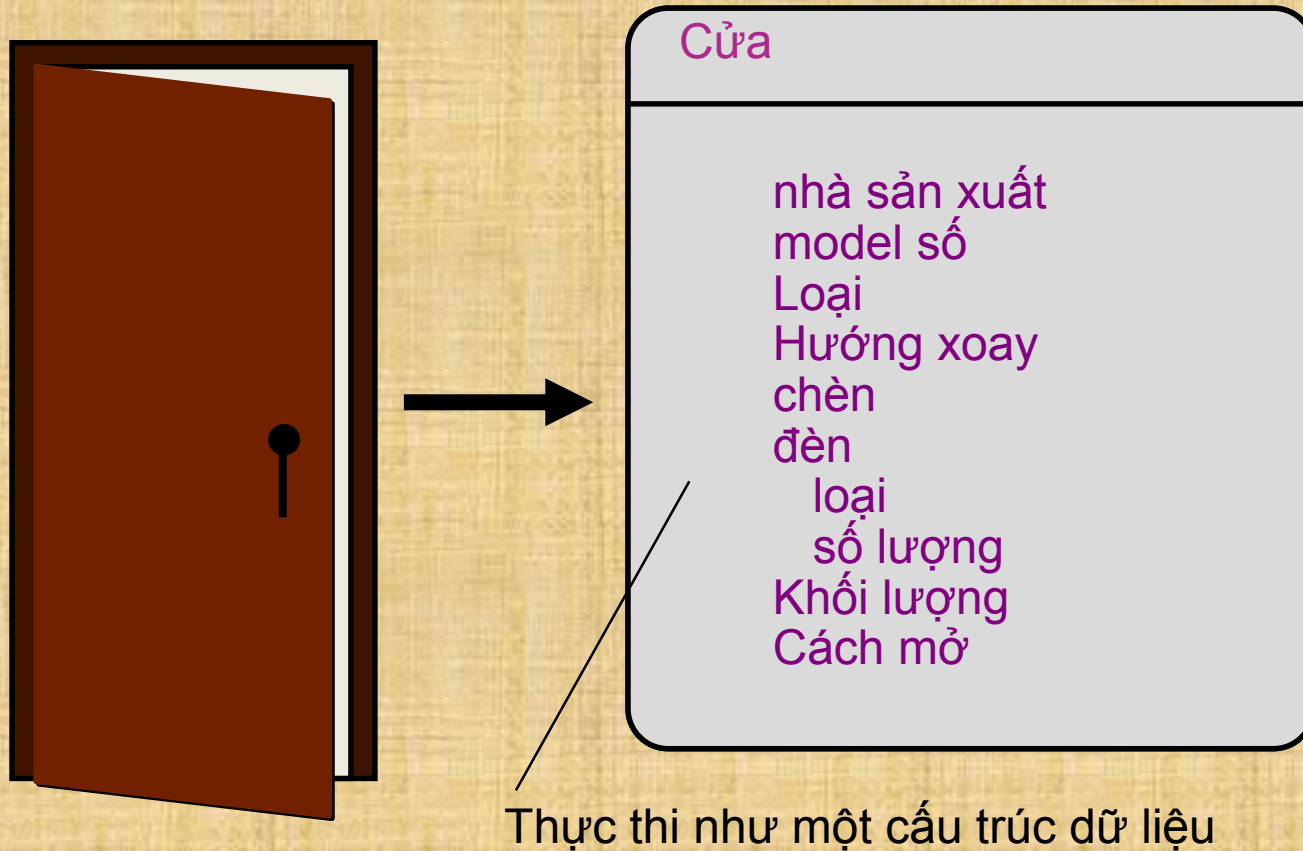
- Quá trình thiết kế không nên mắc phải 'tunnel vision.'
- Việc thiết kế nên có thể truy ngược về mô hình phân tích.
- Việc thiết kế không nên 'phát minh lại bánh xe'.
- Việc thiết kế nên "giảm thiểu khoảng cách trí tuệ" [DAV95] giữa phần mềm và bài toán như nó tồn tại trong thế giới thực.
- Việc thiết kế nên biểu lộ tính đồng nhất và tích hợp.
- Việc thiết kế nên được cấu trúc để thích ứng với thay đổi.
- Việc thiết kế nên được cấu trúc để làm suy thoái (degrade) nhẹ nhàng, ngay cả khi đang gặp phải dữ liệu bất thường, các sự kiện, hoặc điều kiện hoạt động .
- Thiết kế không phải là coding, coding không phải là thiết kế.
- Việc thiết kế nên được đánh giá về chất lượng khi nó được tạo ra, chứ không phải sau thực tế.
- Việc thiết kế cần được xem xét để giảm thiểu lỗi khái niệm (ngữ nghĩa).

From Davis [DAV95]

Các khái niệm cơ bản

- **Trừu tượng hoá**—dữ liệu, thủ tục, kiểm soát
- **Kiến trúc**—cấu trúc tổng thể của phần mềm
- **Patterns**—"chuyên tải những tình túy" của một giải pháp thiết kế đã được chứng minh
- **Separation of concerns**—bất kỳ vấn đề phức tạp có thể được xử lý dễ dàng hơn nếu nó được chia thành nhiều mảnh
- **Modularity**—mô đun hoá các dữ liệu và chức năng
- **Tính ẩn**—giao diện được điều khiển
- **Độc lập Chức năng**—Các hàm đơn mục đích và khớp nối thấp.
- **Sàng lọc**—xây dựng chi tiết cho tất cả các khái niệm trừu tượng
- **Các khía cạnh**—một cơ chế cho sự hiểu biết các yêu cầu tổng thể ảnh hưởng đến thiết kế như thế nào
- **Refactoring**— một kỹ thuật tái tổ chức nhằm đơn giản hóa thiết kế
- **OO design concepts**—Phụ lục II
- **Thiết kế Lớp**—cung cấp chi tiết thiết kế mà sẽ cho phép các lớp đã phân tích được thực thi

Trình tượng dữ liệu



Trình tượng thủ tục



Kiến trúc

- “Cấu trúc tổng thể của phần mềm và cách thức mà cấu trúc cung cấp tính toàn vẹn khái niệm cho một hệ thống.” [SHA95a]
 - **Tính cấu trúc.** Khía cạnh này của các đại diện thiết kế kiến trúc xác định các thành phần của một hệ thống (ví dụ, mô-đun, các đối tượng, các bộ lọc) và cách thức mà những thành phần này được đóng gói và tương tác với nhau. Ví dụ, đối tượng được đóng gói cả dữ liệu và việc xử lý các thao tác dữ liệu và tương tác thông qua việc gọi các phương pháp
 - **Tính thêm chức năng.** Các mô tả thiết kế kiến trúc nên giải quyết cách kiến trúc thiết kế đạt yêu cầu về hiệu suất, công suất, độ tin cậy, an toàn, khả năng thích ứng, và các đặc điểm khác của hệ thống.
 - **Họ các hệ thống liên quan.** Thiết kế kiến trúc sẽ dựa trên mô hình lặp lại mà thường gặp trong thiết kế của các họ của các hệ thống tương tự. Về bản chất, các thiết kế cần phải có khả năng sử dụng lại các khối xây dựng kiến trúc.

Patterns(mẫu)

Bản mẫu thiết kế pattern

Tên Pattern—mô tả bản chất của mô hình trong một tên ngắn nhưng ý nghĩa

Intent (ý định)—mô tả các mô hình và những gì nó làm

Also-known-as—liệt kê các từ đồng nghĩa cho các pattern

Motivation(Động lực)—cung cấp một ví dụ về vấn đề

Applicability (Khả năng áp dụng)—lưu ý tình huống thiết kế cụ thể, trong đó mô hình được áp dụng

Structure(cấu trúc)—mô tả các lớp được yêu cầu để thực hiện mô hình

Participants (thành phần tham gia)—mô tả trách nhiệm của các lớp được yêu cầu để thực hiện pattern

Collaborations (sự công tác)—mô tả cách những thành phần tham gia cộng tác để thực hiện trách nhiệm của mình

Consequences(hệ quả)—mô tả các "lực lượng thiết kế" có ảnh hưởng đến các mô hình và các đánh đổi tiềm năng phải được xem xét khi mô hình được thực hiện

Related patterns(patterns liên quan)—tham khảo chéo liên quan đến các mẫu thiết kế

Phân tách các Mối quan tâm

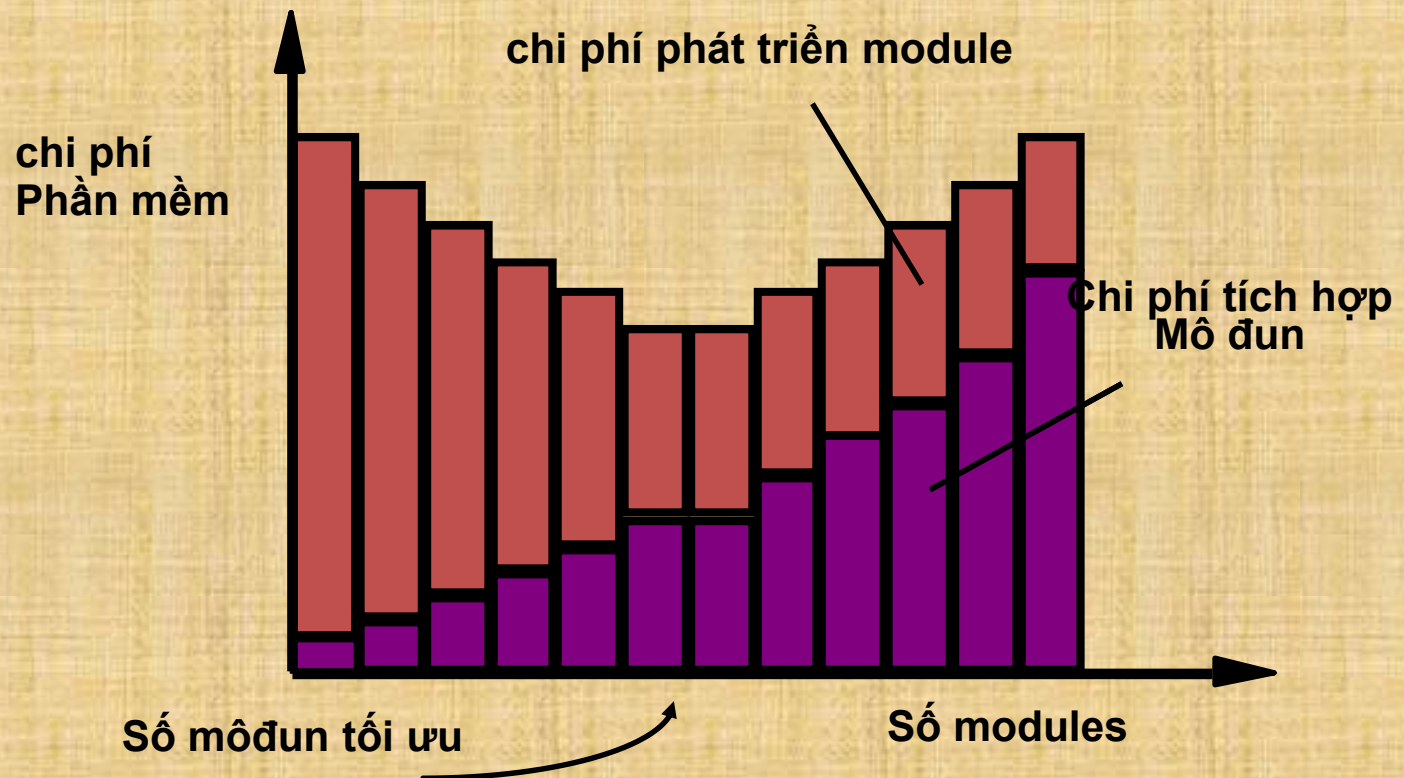
- Bất kỳ vấn đề phức tạp có thể được xử lý dễ dàng hơn nếu nó được chia thành từng mảnh mà mỗi mảnh có thể được giải quyết và / hoặc tối ưu hóa một cách độc lập
- Mối quan tâm là một tính năng hoặc hành vi được quy định như là một phần của mô hình yêu cầu cho các phần mềm
- Bằng cách tách mối quan tâm ra nhỏ hơn, và các mảnh dễ quản lý hơn, một vấn đề mất ít hơn công sức và thời gian để giải quyết.

Mô đun

- “Mô đun là thuộc tính duy nhất của phần mềm cho phép một chương trình có thể quản lý một cách thông minh” [Mye78].
- Phần mềm nguyên khối (ví dụ, một chương trình lớn gồm một mô-đun duy nhất) có thể không được dễ dàng nắm bắt được bởi một kỹ sư phần mềm.
 - Số lượng các đường dẫn điều khiển, khoảng thời gian tham khảo, số lượng các biến, và độ phức tạp tổng thể sẽ làm cho việc hiểu được gần như không thể.
- Trong hầu hết các trường hợp, bạn nên phá vỡ thiết kế thành nhiều module, hy vọng sẽ làm cho việc hiểu biết dễ dàng hơn và như một hệ quả, giảm chi phí cần thiết để xây dựng các phần mềm.

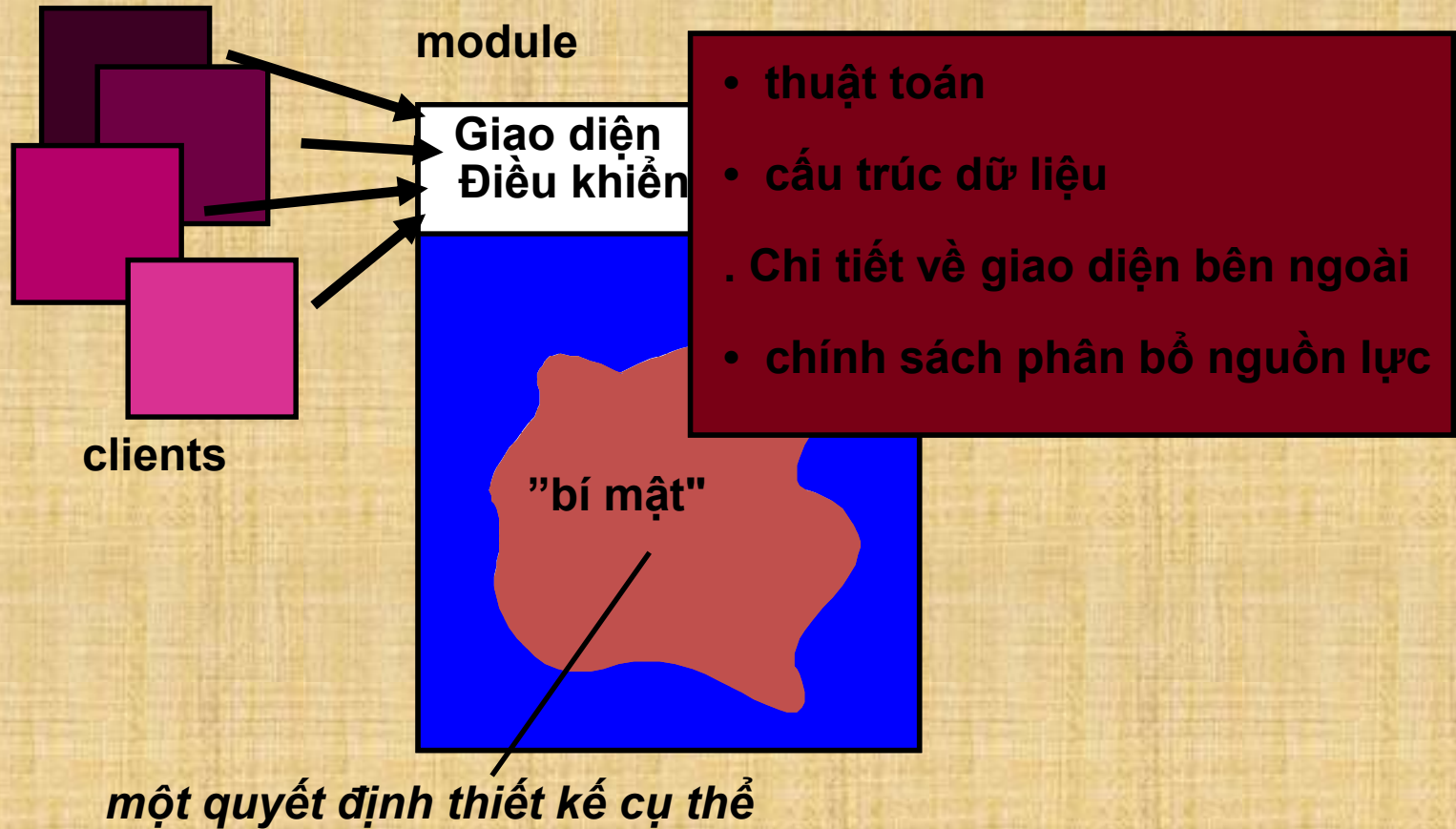
Modularity: sự đánh đổi

Số "chính xác" các mô đun
cho một thiết kế phần mềm cụ thể?



These slides are designed to accompany Software Engineering:
A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

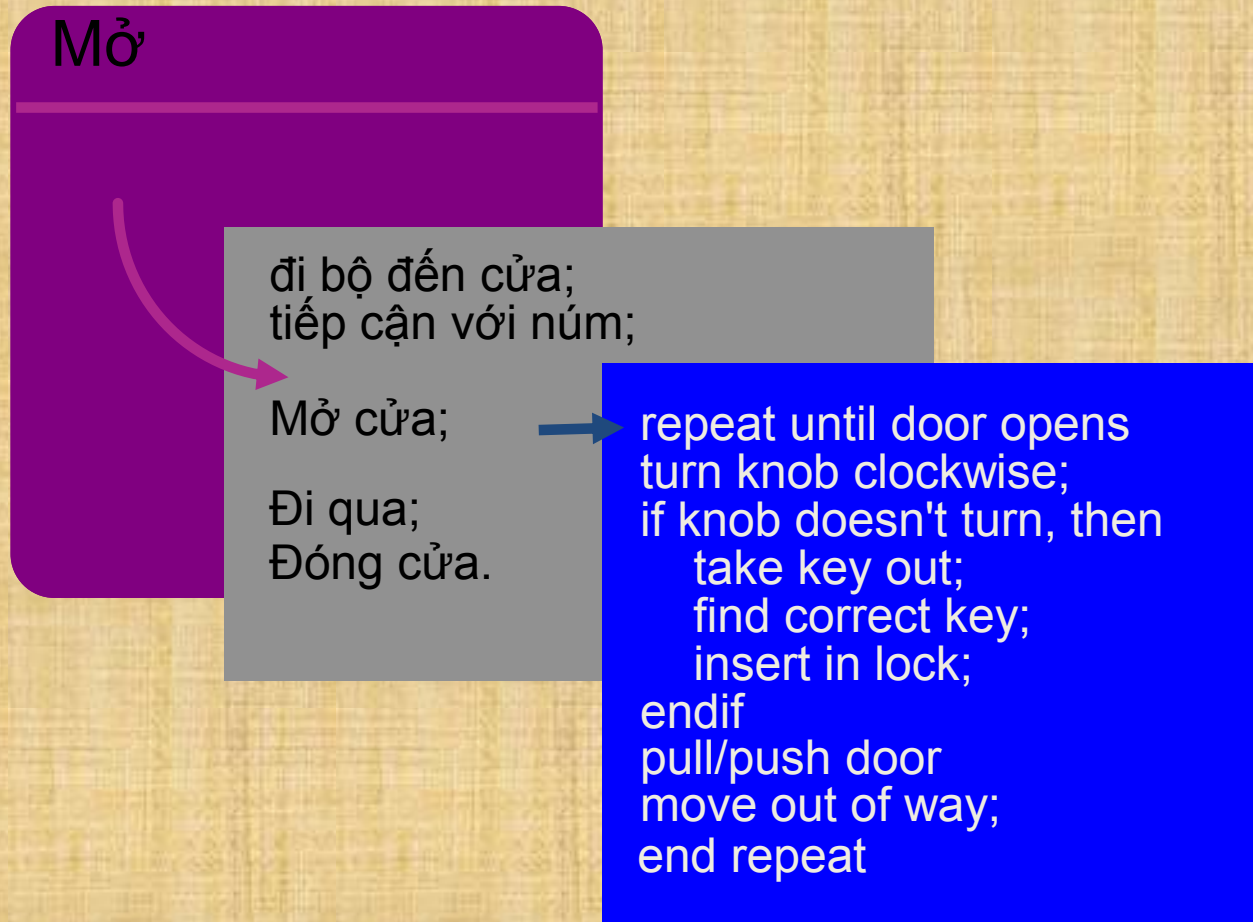
Ẩn thông tin



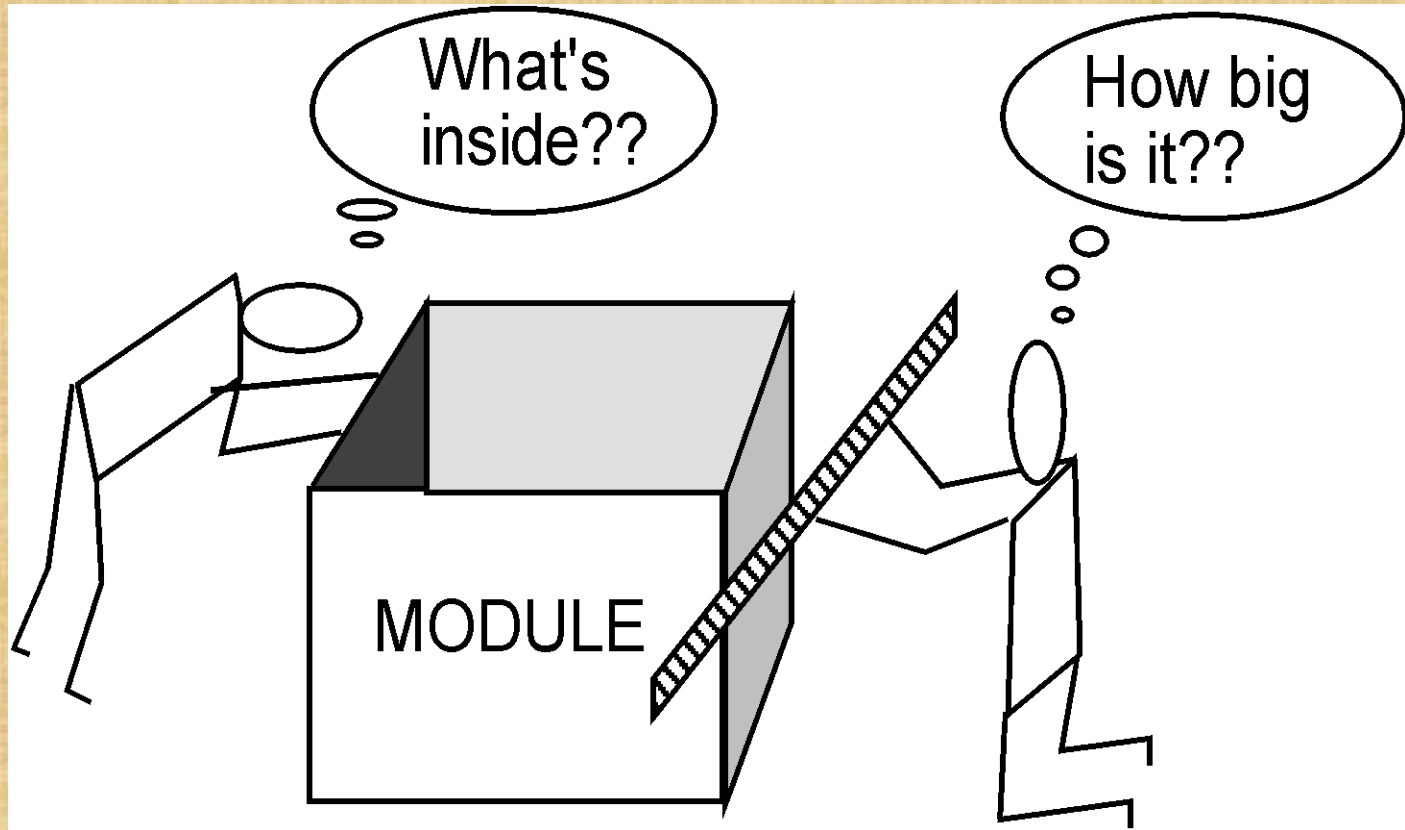
Tại sao lại Ẩn thông tin?

- làm giảm khả năng "tác dụng phụ"
- hạn chế ảnh hưởng chung của quyết định thiết kế cục bộ
- nhấn mạnh truyền thông qua giao diện điều khiển
- không khuyến khích việc sử dụng các dữ liệu toàn cục
- dẫn đến đóng gói, một thuộc tính của thiết kế chất lượng cao
- kết quả trong phần mềm chất lượng cao

Sàng lọc theo từng bước



Định kích thước mô đun: Hai cách nhìn



These slides are designed to accompany Software Engineering:
A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Tính độc lập chức năng

- Tính độc lập chức đạt được bằng cách phát triển các module với chức năng đơn lẻ và một "ác cảm" với tương tác quá mức với các module khác.
- **Sự gắn kết** là một dấu hiệu của sức mạnh chức năng tương đối của một module.
- Một module gắn kết thực hiện một nhiệm vụ duy nhất, đòi hỏi ít tương tác với các thành phần khác trong các phần khác của chương trình. Nói đơn giản, một mô-đun gắn kết nên (lý tưởng) làm chỉ là một việc.
- **Ghép nối** là một dấu hiệu của sự phụ thuộc lẫn nhau tương đối giữa các mô-đun.
- Ghép nối phụ thuộc vào độ phức tạp giao diện giữa các module, các điểm mà tại đó entry hoặc tham chiếu được thực hiện cho một mô-đun, và những dữ liệu truyền qua giao diện.

Các khía cạnh

- Hãy xem xét hai yêu cầu, A và B. Yêu cầu A giao nhau với yêu cầu B "nếu một phân tách phần mềm [tính chế] được lựa chọn trong đó B không thể làm hài lòng mà không cần dùng A.[Ros04]
- Một **khía cạnh** là một đại diện của một mối quan tâm giao nhau.



Các khía cạnh — Ví dụ

- Hãy xem xét hai yêu cầu với SafeHomeAssured.com WebApp. Yêu cầu A được mô tả qua các trường hợp sử dụng camera giám sát truy cập thông qua Internet. Một tình hình thiết kế sẽ tập trung vào những mô-đun có thể cho phép một người sử dụng đăng ký để truy cập video từ camera đặt khắp không gian. Yêu cầu B là một yêu cầu an ninh chung chung mà nói rằng một người sử dụng đăng ký phải được xác nhận trước khi sử dụng SafeHomeAssured.com. Yêu cầu này được áp dụng cho tất cả các chức năng có sẵn cho người dùng SafeHome đăng ký. Vì tình hình thiết kế xảy ra, A * là một đại diện thiết kế cho yêu cầu A và B * là một đại diện thiết kế cho yêu cầu B. Vì vậy, A * và B * là đại diện của các mối quan tâm, và B * chéo cắt A *.
- Một khía cạnh là một đại diện của một mối quan tâm xuyên suốt. Do đó, các đại diện thiết kế, B *, các yêu cầu, một người sử dụng được đăng ký phải được xác nhận trước khi sử dụng SafeHomeAssured.com, là một khía cạnh của SafeHome WebApp.

Tái cấu trúc

- Fowler [FOW99] định nghĩa cấu trúc lại theo cách sau đây:
 - "Tái cấu trúc là quá trình thay đổi một hệ thống phần mềm trong một cách mà nó không làm thay đổi hành vi bên ngoài của mã [thiết kế] nhưng cải thiện cấu trúc bên trong của nó."
 - Khi phần mềm được refactored, thiết kế hiện có được kiểm tra về sự
 - » Dư thừa
 - » yếu tố thiết kế không sử dụng
 - » các thuật toán không hiệu quả hoặc không cần thiết
 - » cấu trúc dữ liệu xây dựng kém hoặc không phù hợp
 - » hoặc bất kỳ sự thất bại thiết kế khác có thể được điều chỉnh để mang lại một thiết kế tốt hơn.

These slides are designed to accompany Software Engineering:
A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

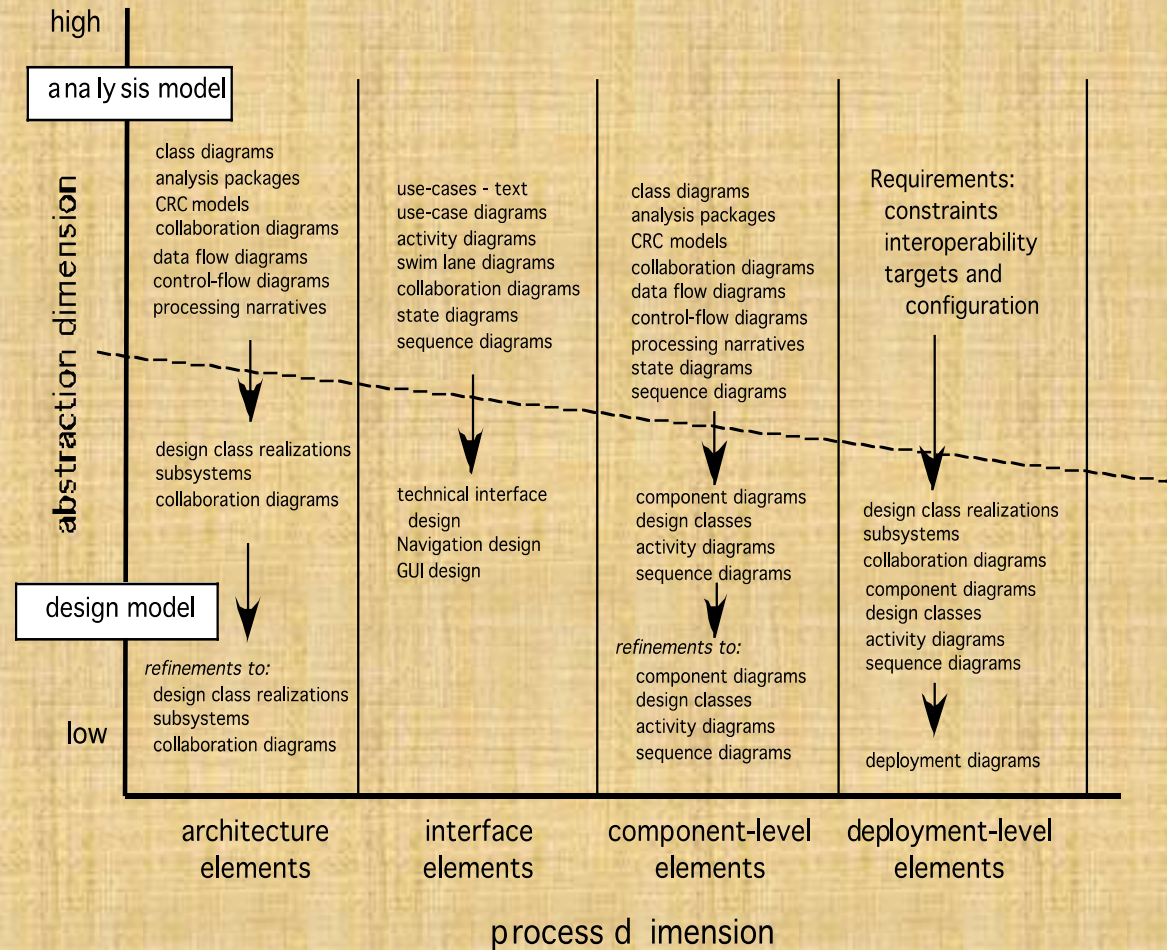
OO Các khái niệm thiết kế

- **Các lớp thiết kế**
 - Các lớp thực thể
 - Các lớp biên
 - các lớp điều khiển
- **sự thừa kế**—tất cả các trách nhiệm của một lớp cha được ngay lập tức được thừa kế bởi tất cả các lớp con
- **thông điệp**—khuyến khích một số hành vi xảy ra trong đối tượng nhận
- **Đa hình**—một đặc tính mà làm giảm đáng kể nỗ lực cần thiết để mở rộng thiết kế.

Thiết kế các lớp

- Các lớp phân tích được tinh chỉnh trong quá trình thiết kế để trở thành **các lớp thực thể**
- **Các lớp biên** phát triển trong thiết kế để tạo ra giao diện (ví dụ, màn hình tương tác hoặc báo cáo) mà người dùng thấy và tương tác với với phần mềm.
 - Các lớp biên được thiết kế với trách nhiệm quản lý các đối tượng cách thực thể được đại diện cho người sử dụng.
- **các lớp điều khiển** được thiết kế để quản lý
 - việc tạo ra hoặc cập nhật các đối tượng thực thể;
 - Sự tức thời của các đối tượng biên khi họ có được thông tin từ các đối tượng thực thể;
 - truyền thông phức tạp giữa các tập của các đối tượng;
 - xác nhận của dữ liệu trao đổi giữa các đối tượng hoặc giữa người sử dụng và với ứng dụng.

Mô hình thiết kế



These slides are designed to accompany Software Engineering:
A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Các thành phần Mô hình thiết kế

- Các thành phần dữ liệu
 - Mô hình dữ liệu -> cấu trúc dữ liệu
 - Mô hình dữ liệu -> kiến trúc cơ sở dữ liệu
- Các thành phần kiến trúc
 - miền ứng dụng
 - Các lớp phân tích, mối quan hệ, hợp tác và hành vi của chúng được chuyển thành chứng ngộ thiết kế
 - Patterns và “kiểu” (Chapters 9 and 12)
- Các thành phần giao diện
 - giao diện người dùng (UI)
 - giao diện bên ngoài đến các hệ thống khác, các thiết bị, mạng, hoặc các nhà sản xuất khác hoặc người dùng thông tin
 - giao diện nội bộ giữa các thành phần thiết kế khác nhau.
- Các yếu tố cấu thành
- các thành phần triển khai

Các thành phần kiến trúc

- Các mô hình kiến trúc [Sha96] có nguồn gốc từ ba nguồn:
 - thông tin về miền ứng dụng cho xây dựng phần mềm;
 - các thành phần mô hình yêu cầu cụ thể như sơ đồ luồng dữ liệu và phân tích các lớp, các mối quan hệ và sự hợp tác của chúng, và
 - sự sẵn có của mô hình kiến trúc (Chương 12) và các kiểu (Chương 9).

Các thành phần giao diện

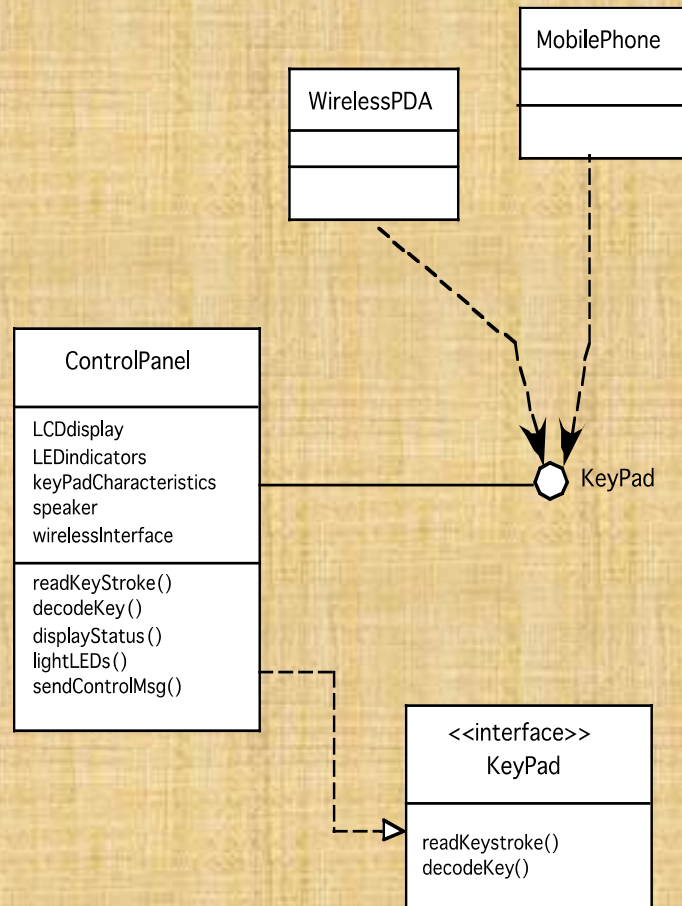
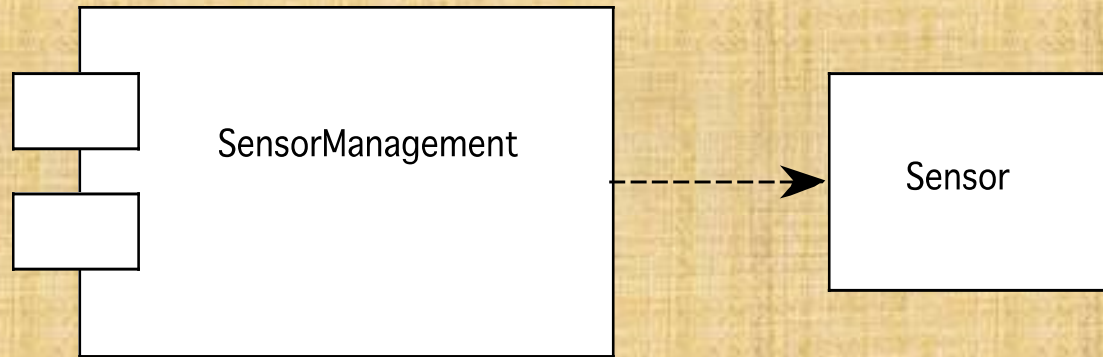


Figure 9.6 UML interface representation for ControlPanel

Component Elements



Các thành phần triển khai

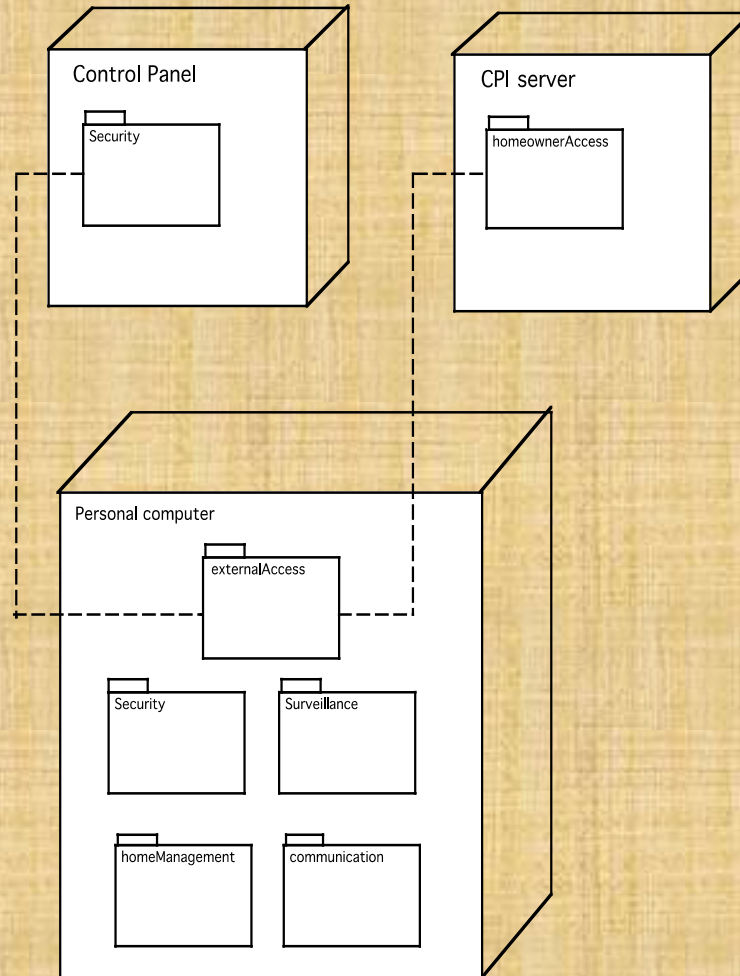


Figure 9.8 UML deployment diagram for *SafeHome*

Thiết kế phần mềm

- Nội dung
 1. Tổng quan thiết kế phần mềm
 - 2. Thiết kế kiến trúc phần mềm**
 3. Thiết kế chi tiết phần mềm
 4. Thiết kế giao diện người dùng
 5. Thiết kế mẫu
 6. Thiết kế giao diện cho ứng dụng WebApp

Why Architecture?

- Các kiến trúc không phải là phần mềm hoạt động. Thay vào đó, nó là một đại diện cho phép một kỹ sư phần mềm để:
 1. phân tích hiệu quả của thiết kế trong việc đáp ứng các yêu cầu đề ra,
 2. xem xét lựa chọn thay thế kiến trúc khi thay đổi thiết kế vẫn tương đối dễ dàng, và
 3. giảm thiểu rủi ro gắn liền với việc xây dựng các phần mềm.

Tại sao kiến trúc quan trọng?

- **Đại diện của kiến trúc phần mềm là một tạo khả năng** cho truyền thông giữa tất cả các bên (các bên liên quan) quan tâm đến sự phát triển của một hệ thống dựa trên máy tính.
- **Những kiến trúc làm nổi bật thiết kế quyết định ban đầu** mà sẽ có một tác động sâu sắc trên tất cả các công việc kỹ thuật phần mềm sau và, quan trọng hơn, vào sự thành công cuối cùng của hệ thống như là một thực thể hoạt động.
- **Kiến trúc "tạo thành một mode minh bạch tương đối nhỏ** về cách hệ thống được cấu trúc và cách các thành phần của nó làm việc cùng nhau" [BAS03].

Mô tả kiến trúc

- IEEE Computer Society đã đề nghị IEEE-Std-1471-2000, Recommended Practice for Architectural Description of Software-Intensive System, [IEE00]
 - đề thiết lập một khuôn khổ khái niệm và từ vựng cho việc sử dụng trong các thiết kế của kiến trúc phần mềm,
 - đề cung cấp hướng dẫn chi tiết cho đại diện cho một mô tả kiến trúc, and
 - khuyến khích thực hành thiết kế kiến trúc âm thanh.
- The IEEE Standard định nghĩa một mô tả kiến trúc (AD) là một "một tập hợp các sản phẩm để tài liệu hoá một kiến trúc."
 - Các mô tả chính nó được đại diện bằng cách sử dụng nhiều quan điểm, nơi từng xem là "một đại diện của cả một hệ thống từ quan điểm của một tập hợp có liên quan của [các bên liên quan] các mối quan tâm."

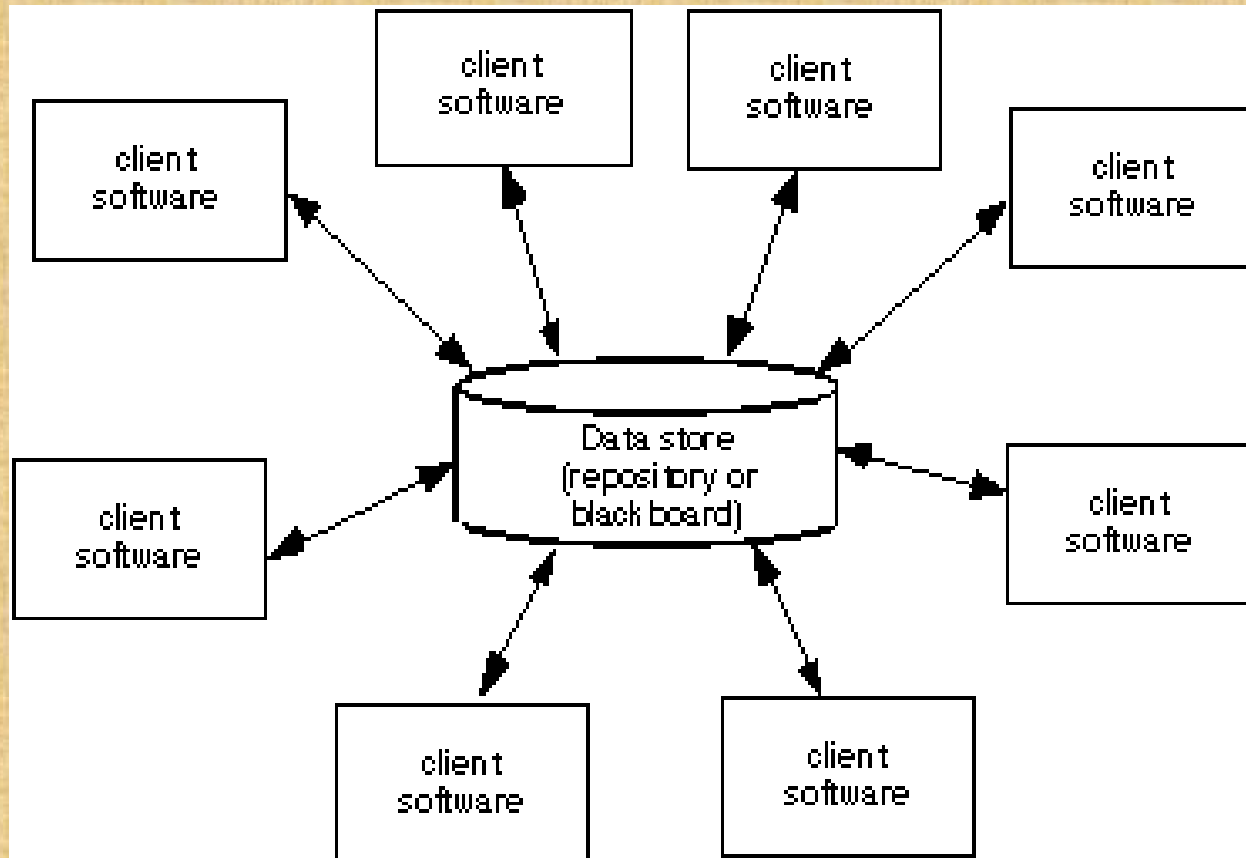
Thể loại kiến trúc

- **Thể loại** ngụ ý một phân loại cụ thể trong lĩnh vực phần mềm tổng thể.
- Trong mỗi thể loại, bạn gặp phải một số tiểu phân loại.
 - Ví dụ, trong các thể loại của các tòa nhà, bạn sẽ gặp phải những phong cách chung sau đây: nhà ở, căn hộ, chung cư, cao ốc văn phòng, tòa nhà công nghiệp, nhà kho, vv.
 - Trong mỗi phong cách chung, phong cách cụ thể hơn có thể được áp dụng. Mỗi phong cách sẽ có một cấu trúc có thể được mô tả bằng một tập các mô hình dự đoán được.

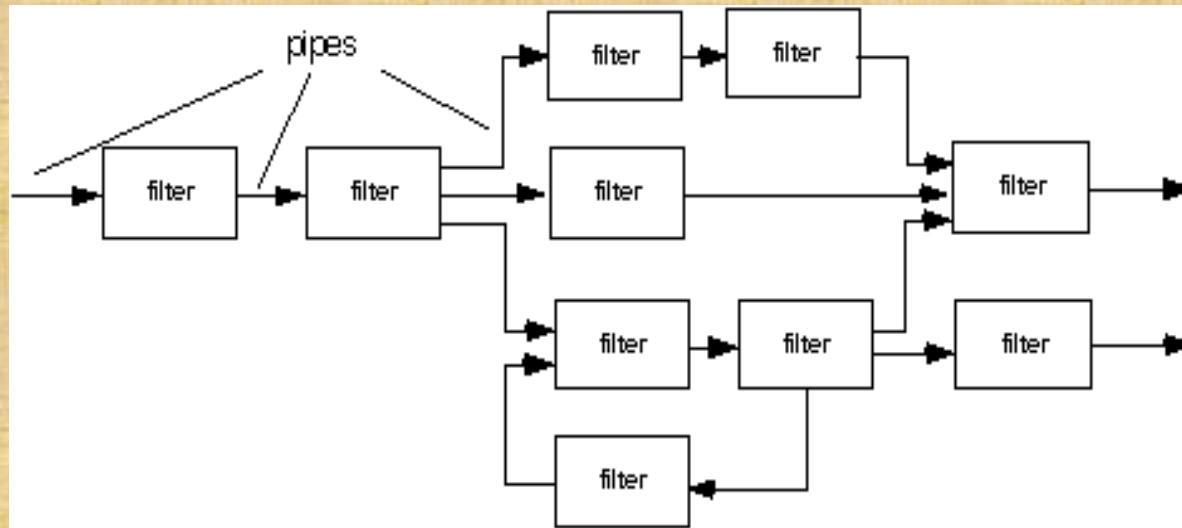
Kiểu kiến trúc

- Mỗi phong cách mô tả một loại hệ thống bao gồm: (1) một tập hợp các thành phần (ví dụ, một cơ sở dữ liệu, mô đun tính toán) thực hiện một chức năng cần thiết của một hệ thống, (2) một tập hợp các kết nối cho phép "truyền thông, phối hợp và hợp tác" giữa các thành phần, (3) khó khăn để xác định cách các thành phần có thể được tích hợp để tạo thành hệ thống, và (4) các mô hình ngữ nghĩa cho phép một nhà thiết kế phải hiểu được tính chất tổng thể của hệ thống bằng cách phân tích các đặc tính được biết đến của các bộ phận cấu thành của nó.
 - Kiến trúc lấy dữ liệu làm trung tâm
 - Kiến trúc luồng dữ liệu
 - kiến trúc gọi và trả về
 - Kiến trúc hướng đối tượng
 - kiến trúc lớp

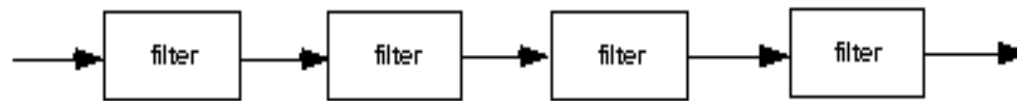
Kiến trúc lấy dữ liệu làm trung tâm



Kiến trúc luồng dữ liệu

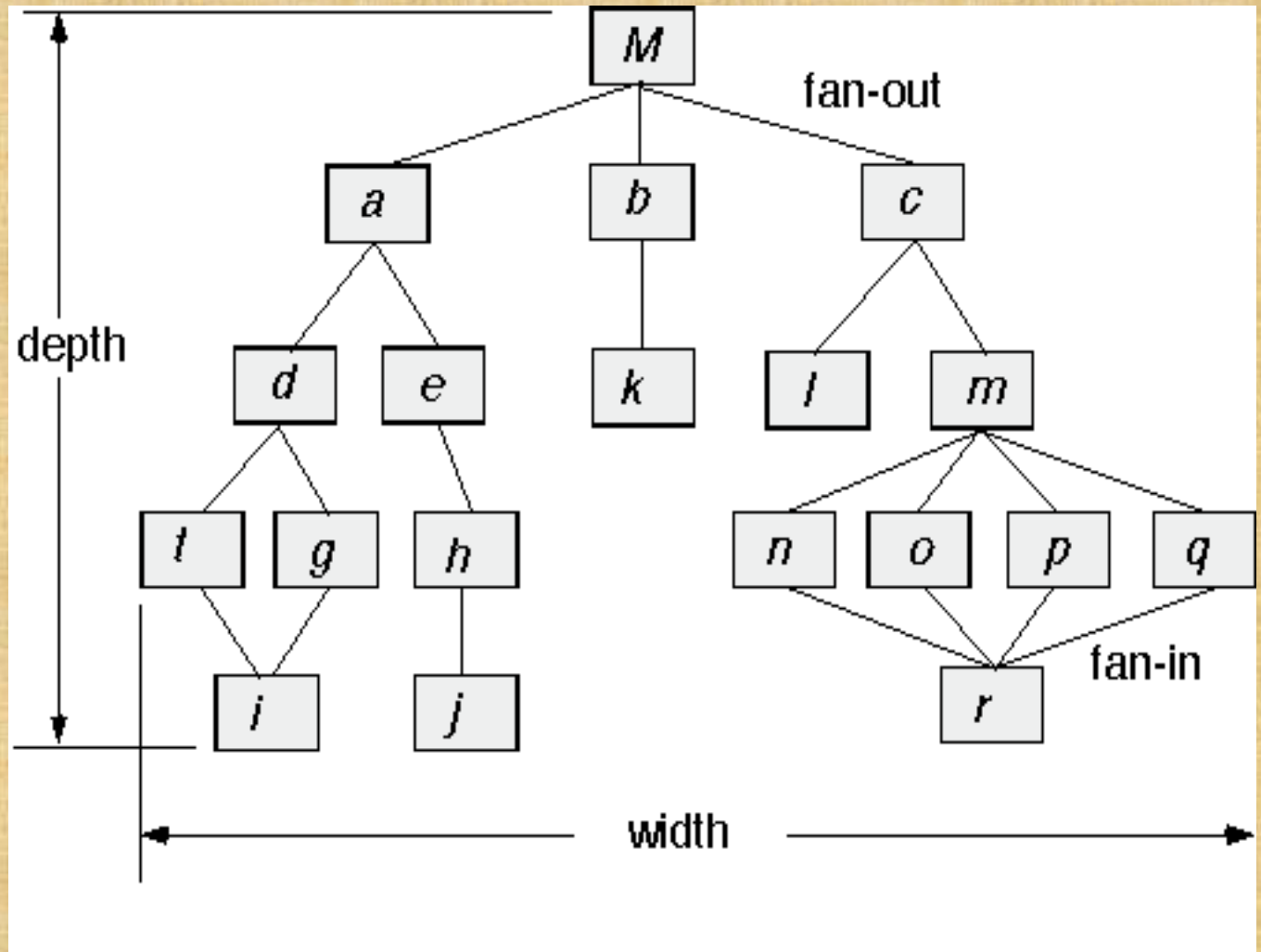


(a) pipes and filters

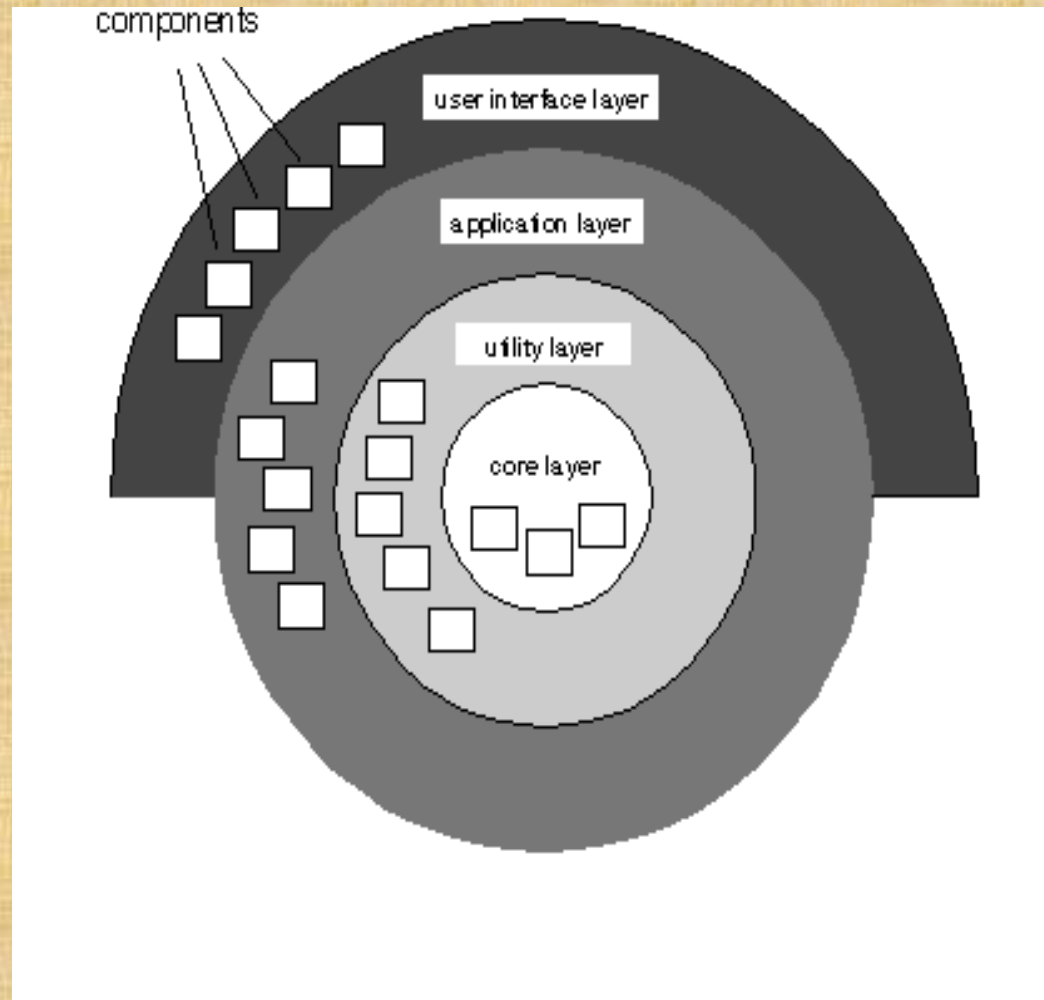


(b) batch sequential

Kiến trúc gọi và trả về



Kiến trúc phân lớp



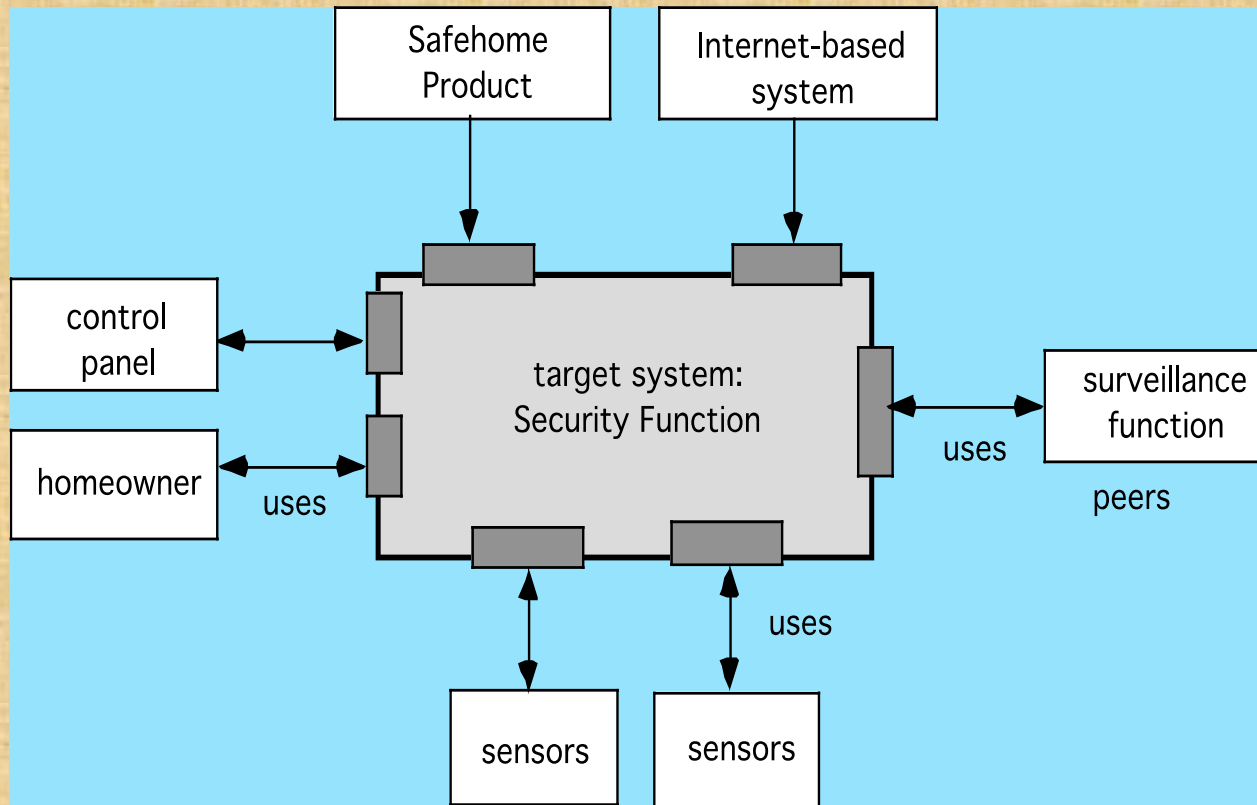
Mô hình kiến trúc

- **Đồng thời**—các ứng dụng phải xử lý nhiều nhiệm vụ một cách mô phỏng song song
 - mô hình quản lý điều hành quy trình hệ thống
 - bảng kế hoạch pattern
- **Persistence**—Dữ liệu vẫn tồn tại nếu nó tồn tại qua việc thực hiện các quá trình tạo ra nó. Hai mô hình phổ biến:
 - một mô hình cơ sở dữ liệu hệ thống quản lý áp dụng lưu trữ và truy xuất khả năng của một DBMS với các kiến trúc ứng dụng
 - một mô hình bền bỉ mức độ ứng dụng mà xây dựng tính năng bền bỉ vào các kiến trúc ứng dụng
- **Phân phối**— cách thức mà hệ thống hoặc các thành phần trong hệ thống giao tiếp với nhau trong một môi trường phân phối
 - Một nhà môi giới hoạt động như một "người trung gian" giữa các thành phần client và một thành phần máy chủ.

Thiết kế kiến trúc

- Phần mềm này phải được đặt trong bối cảnh
 - thiết kế cần xác định các thực thể bên ngoài (các hệ thống khác, thiết bị, con người) mà phần mềm tương tác với và bản chất của sự tương tác
- Một tập hợp các nguyên mẫu kiến trúc cần được xác định
 - Một nguyên mẫu là một trừu tượng (tương tự như một lớp) đại diện cho một phần tử của hệ thống hành vi
- Các nhà thiết kế xác định cấu trúc của hệ thống bằng cách xác định và tinh chỉnh các thành phần phần mềm thực hiện từng nguyên mẫu

Bối cảnh kiến trúc



Nguyên mẫu

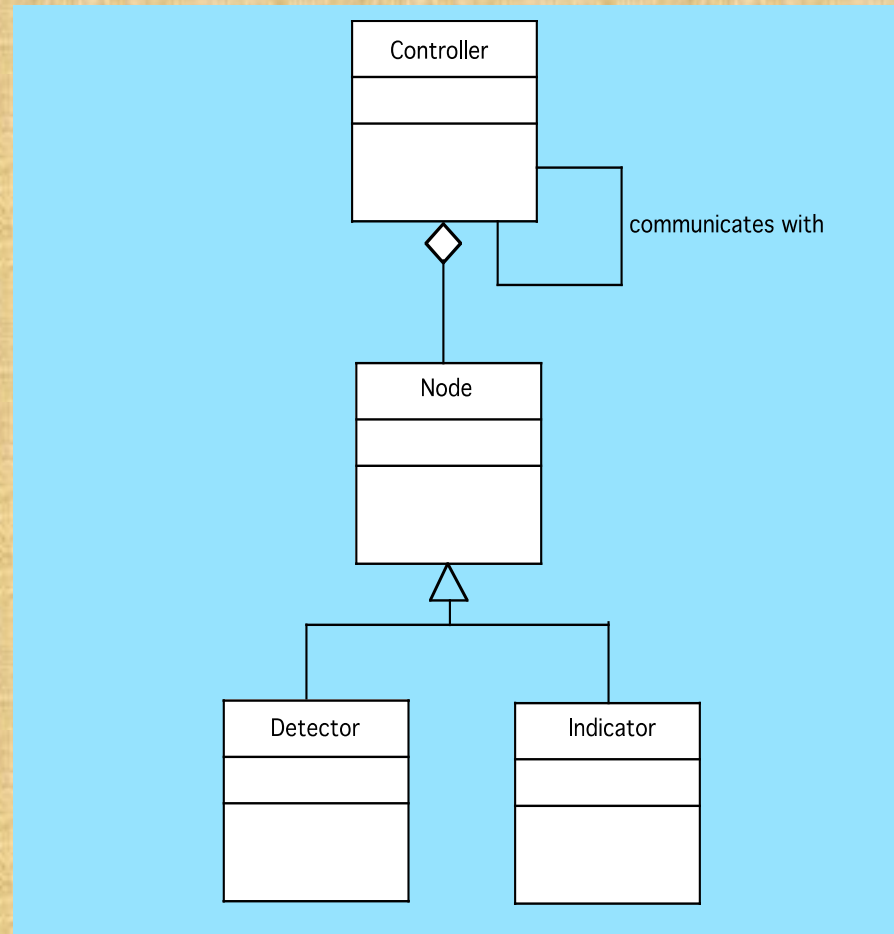
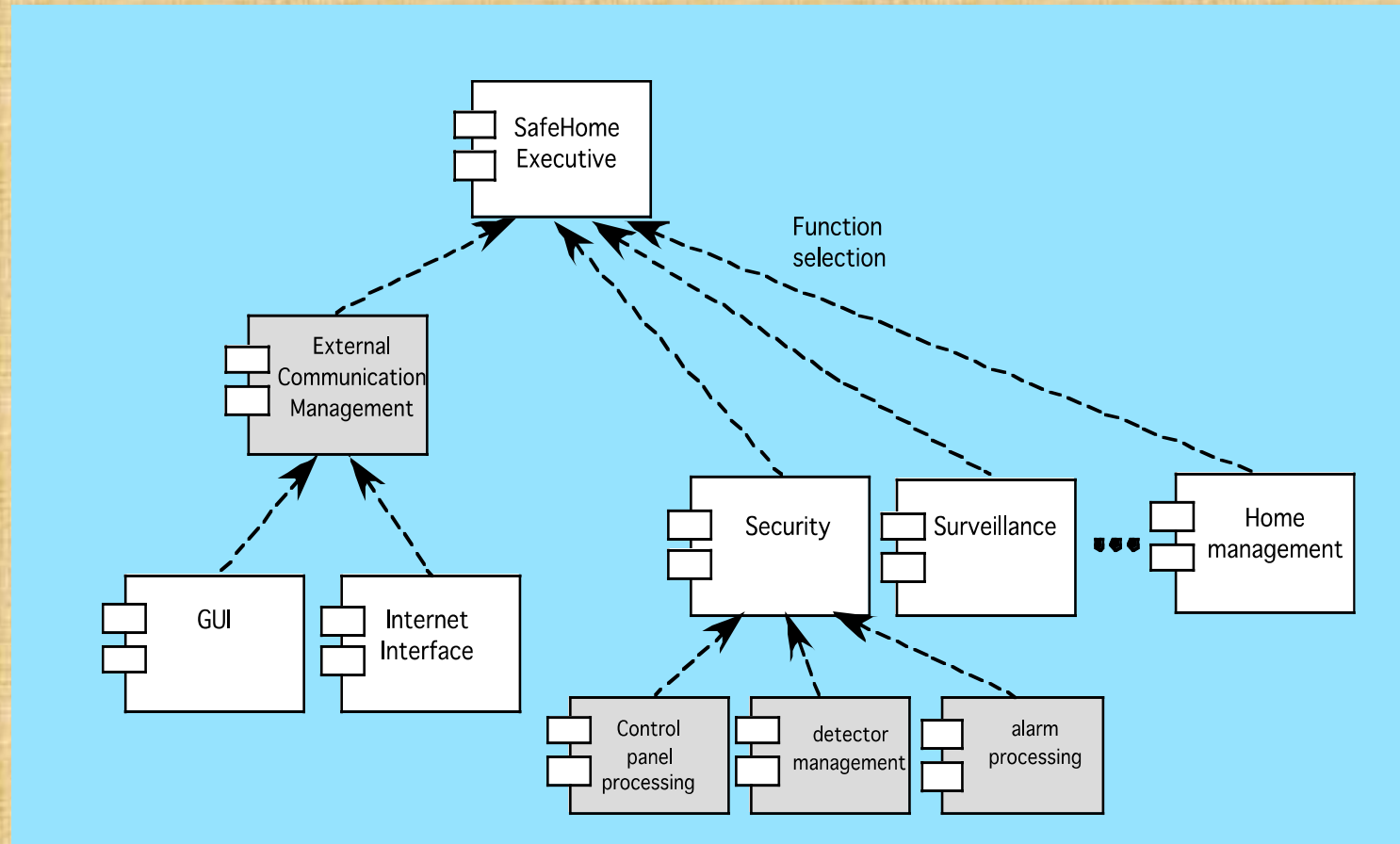
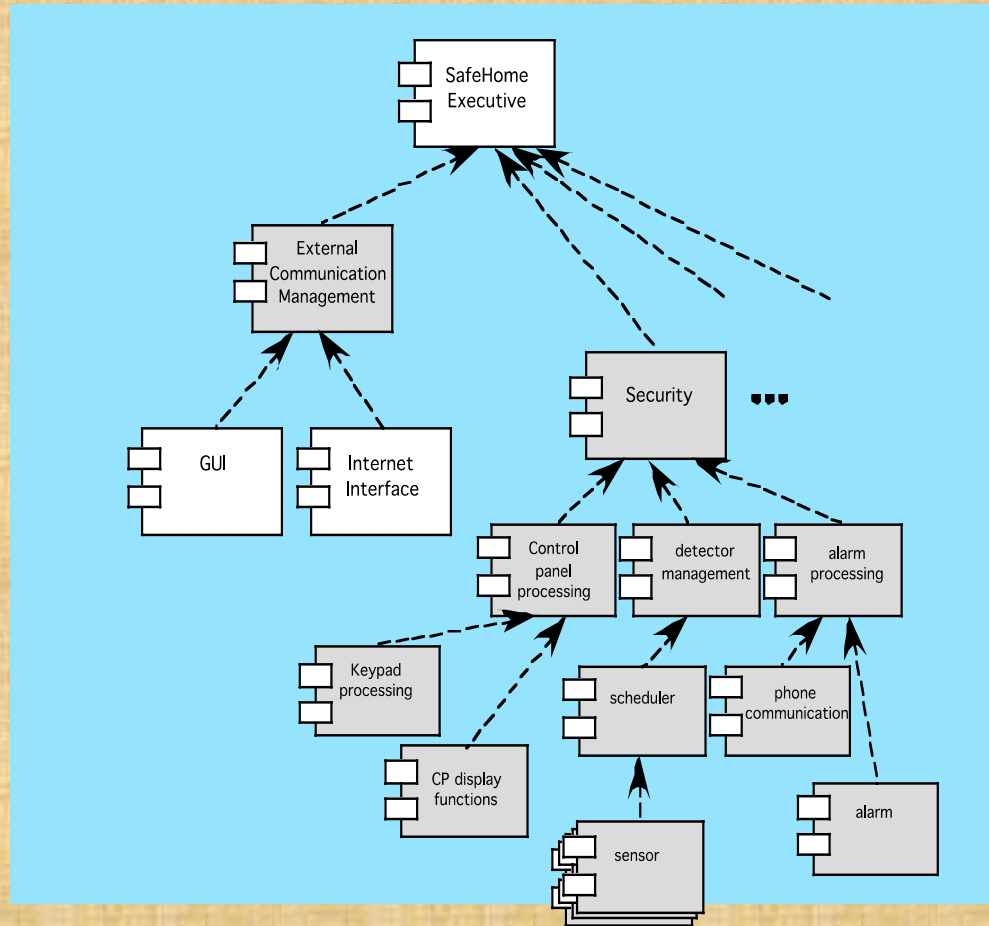


Figure 10.7 UML relationships for SafeHome security function archetypes (adapted from [BOS00])

Cơ cấu thành phần



Cơ cấu thành phần tinh chỉnh



Phân tích thiết kế kiến trúc

1. Thu thập các kịch bản.
2. Gợi ý các yêu cầu, ràng buộc, và đặc tả môi trường.
3. Mô tả các phong cách / mô hình kiến trúc đã được lựa chọn để giải quyết các tình huống và yêu cầu:
 - quan điểm mô-đun
 - góc nhìn quá trình
 - quan điểm dòng dữ liệu
4. Đánh giá chất lượng thuộc tính bằng cách coi mỗi thuộc tính trong sự cô lập.
5. Xác định mức độ nhạy cảm của các thuộc tính chất lượng với các thuộc tính kiến trúc khác nhau cho một phong cách kiến trúc cụ thể.
6. Kiến trúc ứng cử viên phê bình (được phát triển trong bước 3) bằng cách sử dụng phân tích độ nhạy được thực hiện ở bước 5.

Độ phức tạp kiến trúc

- Độ phức tạp tổng thể của một kiến trúc đề xuất được đánh giá bằng cách xem xét sự phụ thuộc giữa các thành phần trong kiến trúc[Zha98]
 - **Chia sẻ phụ thuộc** đại diện cho mối quan hệ phụ thuộc giữa các khách hàng sử dụng cùng một tài nguyên hoặc các nhà sản xuất đã sản xuất ra cho cùng những người tiêu dùng .
 - **Phụ thuộc luồng** đại diện cho mối quan hệ phụ thuộc giữa sản xuất và tiêu dùng các nguồn lực.
 - **Phụ thuộc ràng buộc** đại diện cho các ràng buộc vào các luồng liên quan của kiểm soát trong một tập hợp các hoạt động.

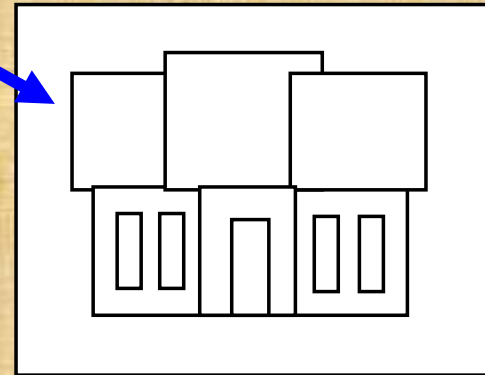
ADL

- Architectural description language (Ngôn ngữ mô tả kiến trúc-ADL) cung cấp các ngữ nghĩa và cú pháp để mô tả một kiến trúc phần mềm
- Cung cấp cho nhà thiết kế với khả năng:
 - phân giải các thành phần kiến trúc
 - kết hợp thành phần riêng lẻ thành các khối kiến trúc lớn hơn và
 - đại diện cho giao diện (cơ chế kết nối) giữa các thành phần.

Một phương pháp thiết kế kiến trúc

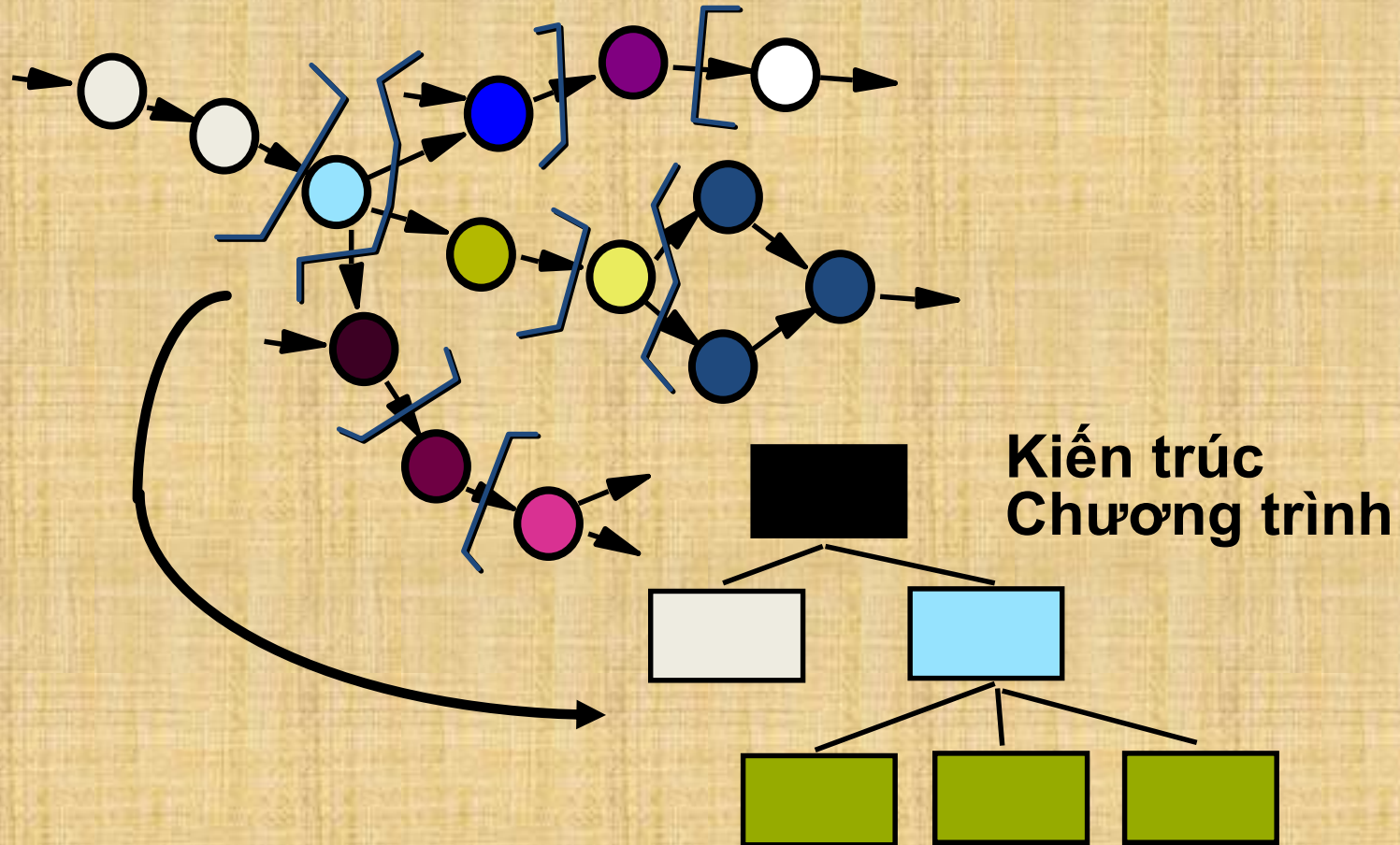
yêu cầu khách hàng

"bốn phòng ngủ, ba phòng tắm,
rất nhiều kính..."



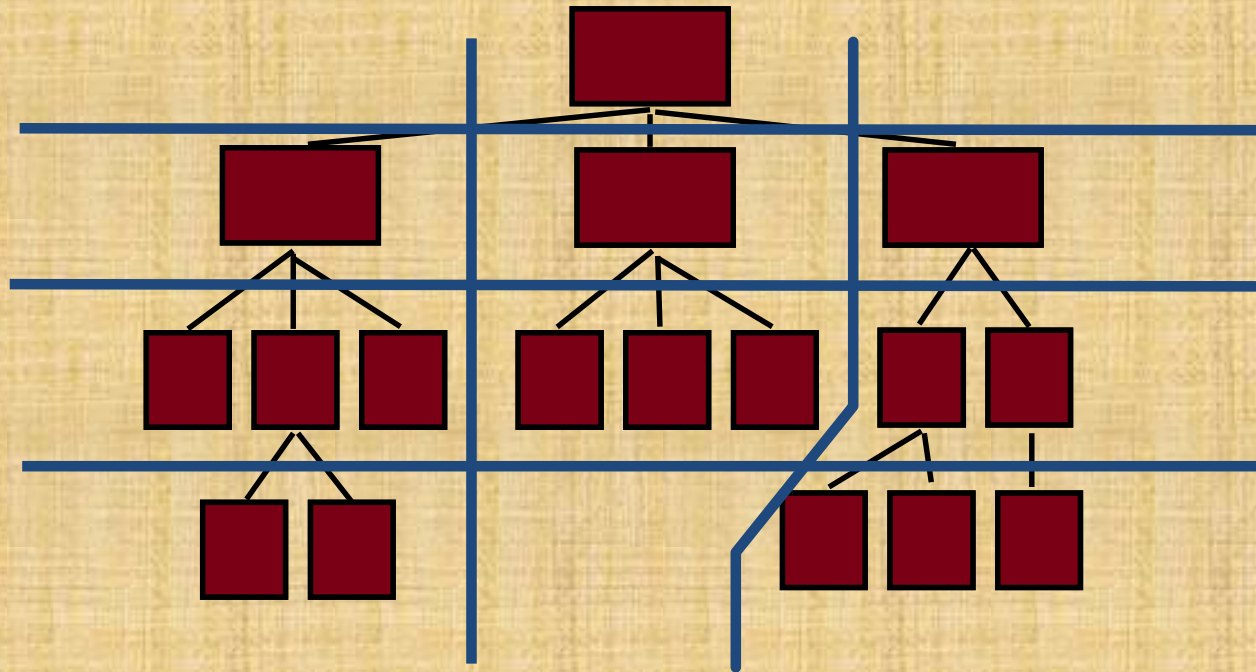
thiết kế kiến trúc

Phát sinh Kiến trúc Chương trình



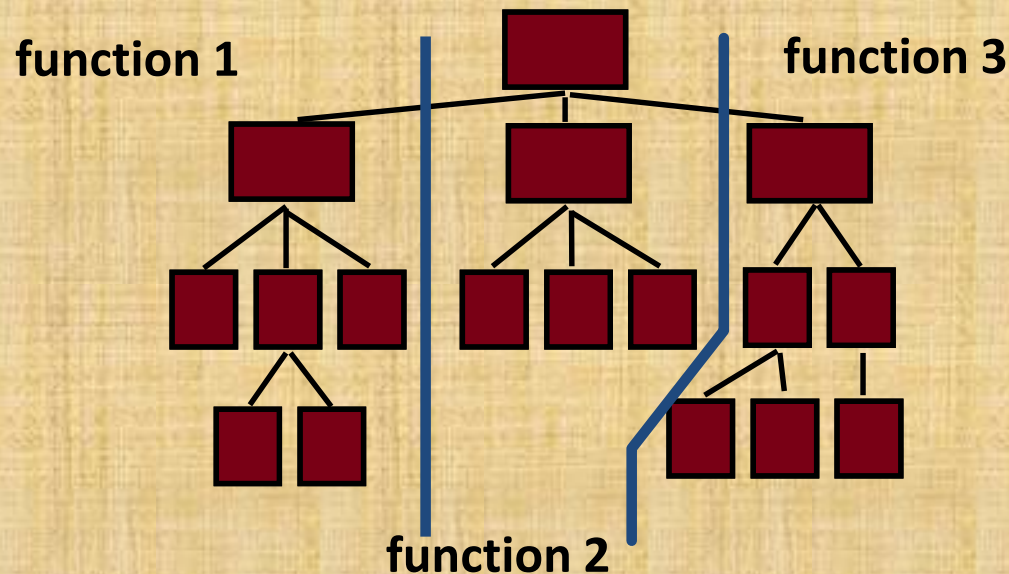
Phân vùng Kiến trúc

- Phân vùng "ngang" và "dọc" là cần thiết



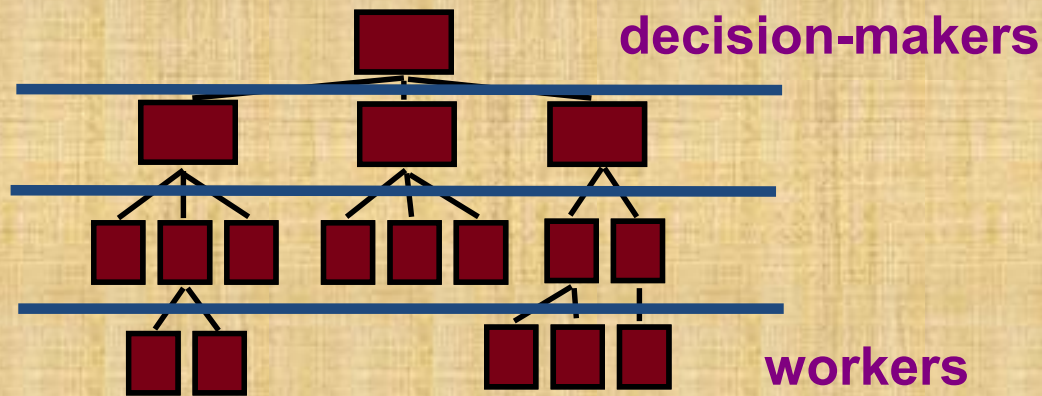
Phân vùng ngang

- xác định các nhánh riêng biệt của hệ thống phân cấp module cho từng chức năng chính
- sử dụng mô-đun điều khiển để phối hợp truyền thông giữa các chức năng



Phân vùng dọc: Factoring

- thiết kế để việc ra quyết định và làm việc được phân tầng
- module ra quyết định nên nằm ở trên cùng của kiến trúc



Tại sao phân vùng Kiến trúc?

- kết quả trong phần mềm dễ dàng kiểm tra hơn
- dẫn đến phần mềm dễ dàng để duy trì hơn
- kết quả trong lan truyền ít tác dụng phụ hơn
- kết quả trong phần mềm dễ dàng để mở rộng hơn

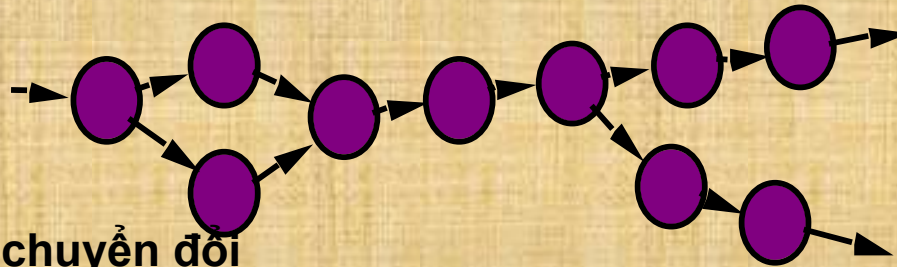


These slides are designed to accompany Software Engineering:
A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Thiết kế cấu trúc

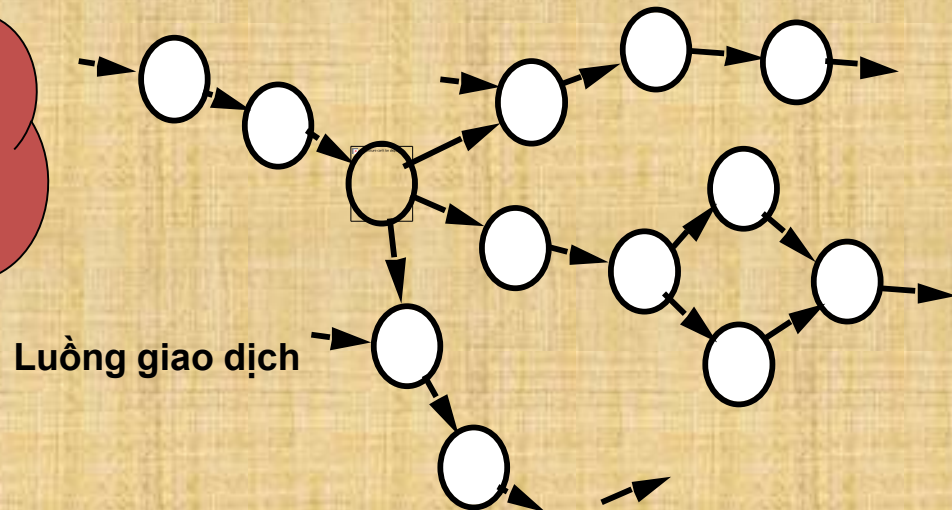
- Mục tiêu: để đưa ra một kiến trúc chương trình được phân chia
- phương pháp tiếp cận:
 - một DFD được ánh xạ vào một kiến trúc chương trình
 - các PSPEC và STD được sử dụng để chỉ ra các nội dung của mỗi mô-đun
- ký pháp: biểu đồ cấu trúc

Các đặc tính luồng



Luồng chuyển đổi

This edition of SEPA does not cover transaction mapping. For a detailed discussion see the SEPA website



Luồng giao dịch

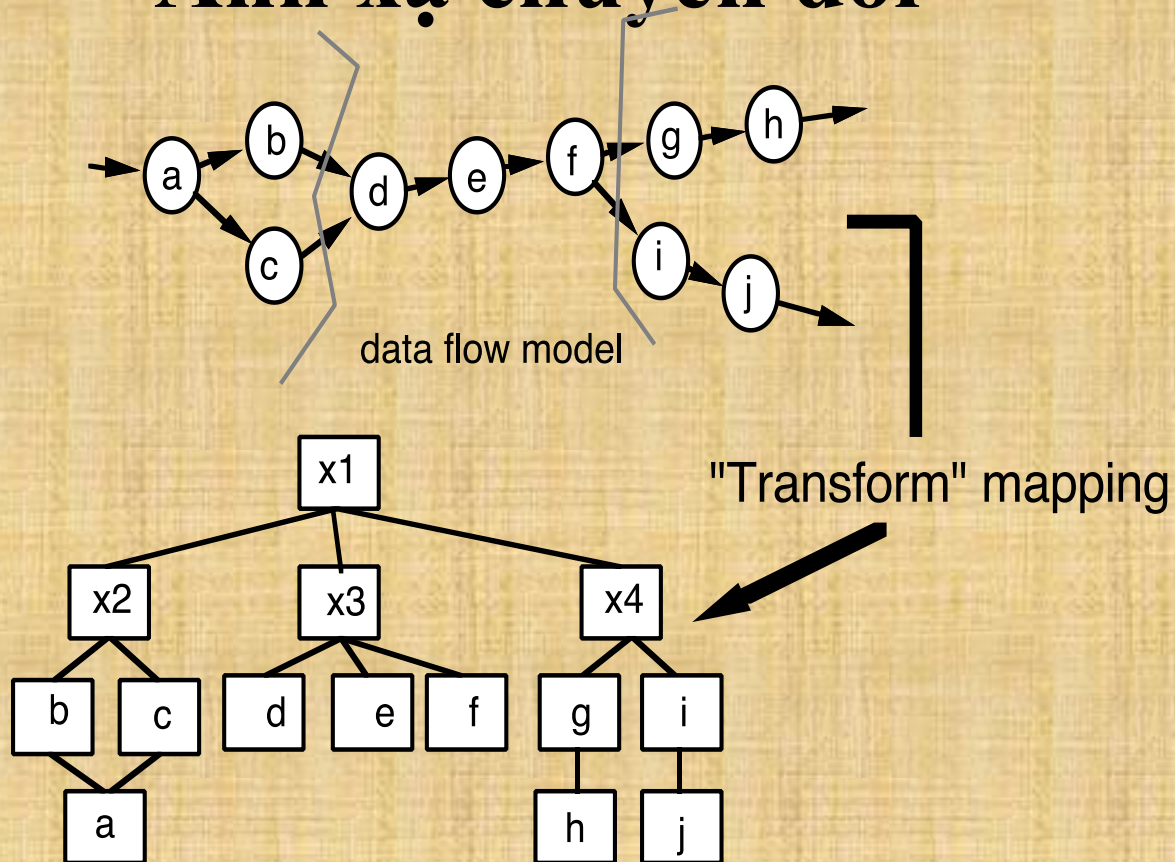
Phương pháp tiếp cận ánh xạ chung

- cô lập ranh giới luồng vào và ra; cho các luồng giao dịch, cô lập các trung tâm giao dịch
- làm việc kể từ ranh giới bên ngoài, ánh xạ DFD biến đổi thành các module tương ứng
- thêm module điều khiển theo yêu cầu
- tinh chỉnh cấu trúc chương trình kết quả bằng cách sử dụng các khái niệm mô đun hiệu quả

Phương pháp tiếp cận ánh xạ chung

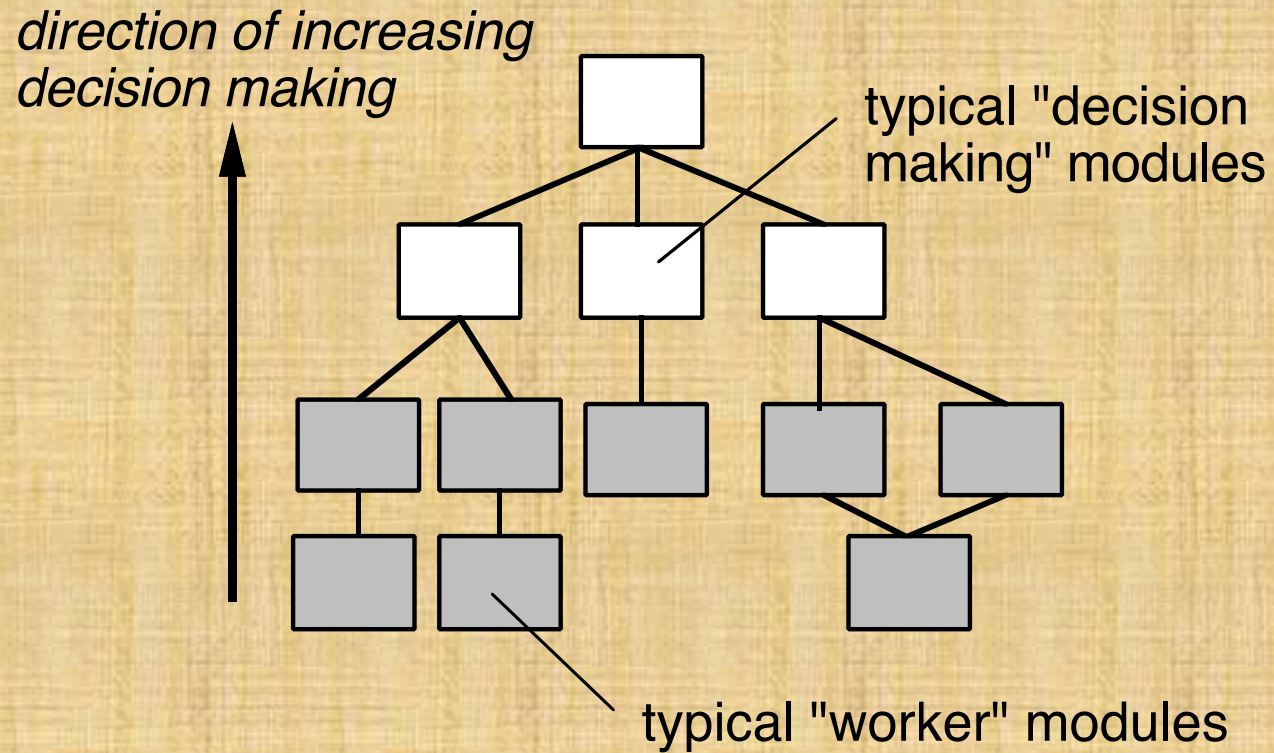
- Cô lập các trung tâm chuyển đổi bằng cách xác định ranh giới luồng vào và ra
- Thực hiện "factoring.cấp độ đầu tiên"
 - Các kiến trúc chương trình có nguồn gốc sử dụng kết quả ánh xạ này trong một phân bố trên-xuống của điều khiển.
 - Factoring dẫn đến một cấu trúc chương trình trong đó các thành phần cấp cao thực hiện việc ra quyết định và thành phần ở mức độ thấp thực hiện hầu hết công việc đầu vào, tính toán, và đầu ra.
 - Thành phần cấp trung thực hiện một số kiểm soát và làm một lượng vừa phải công việc.
- Thực hiện "factoring.cấp độ hai."

Ảnh xạ chuyển đổi

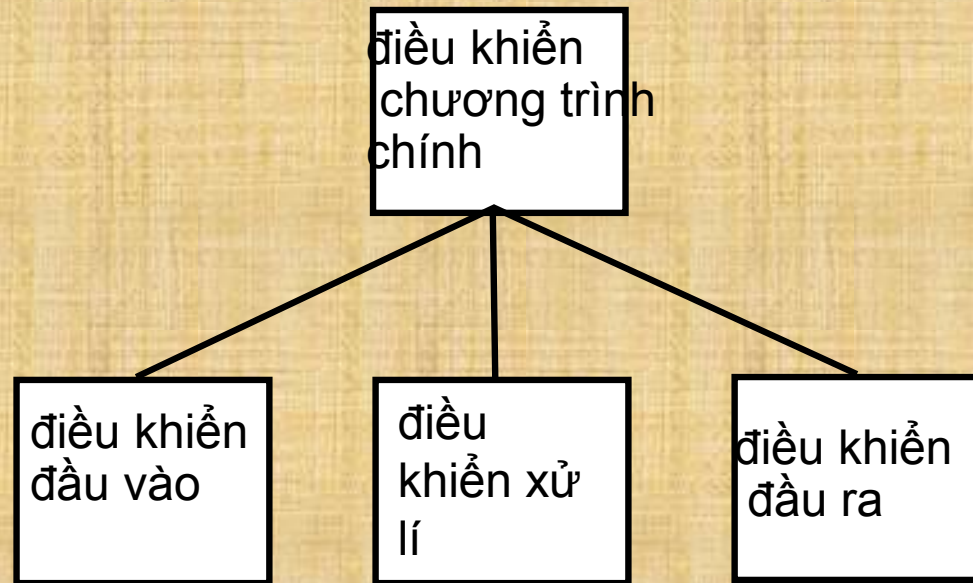


These slides are designed to accompany Software Engineering:
A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

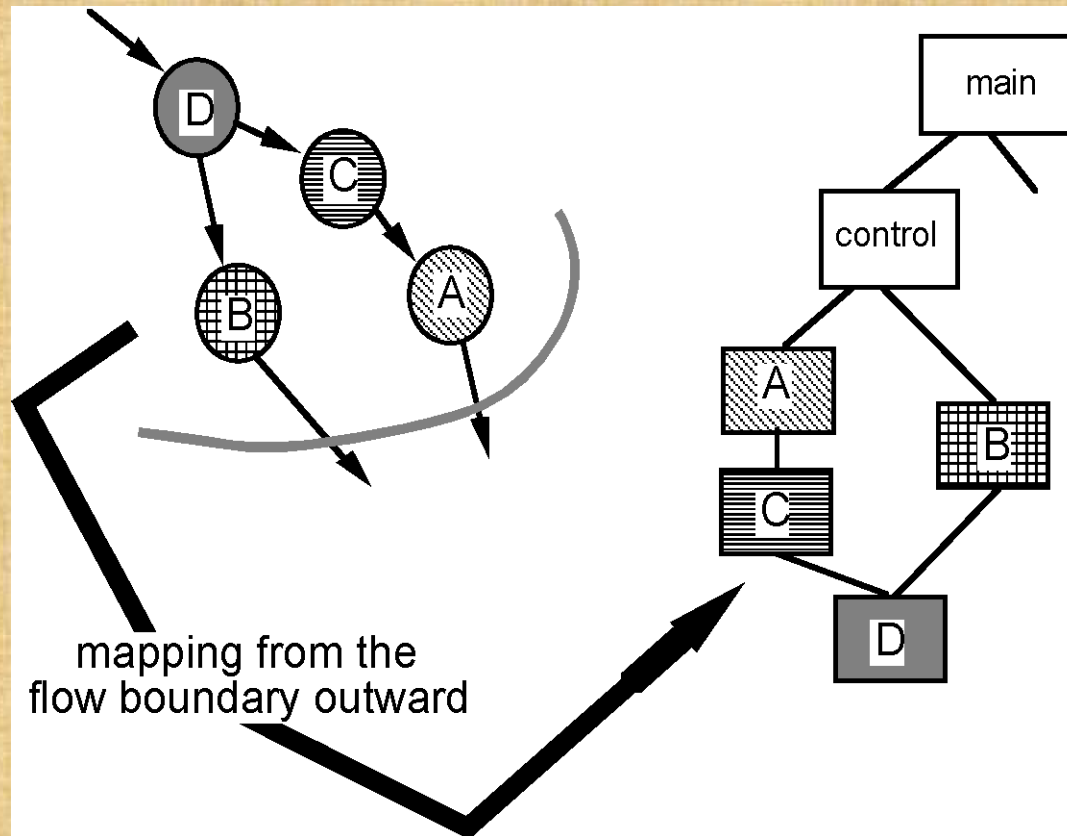
Factoring



Factoring cấp độ một



Ảnh xạ cấp độ hai



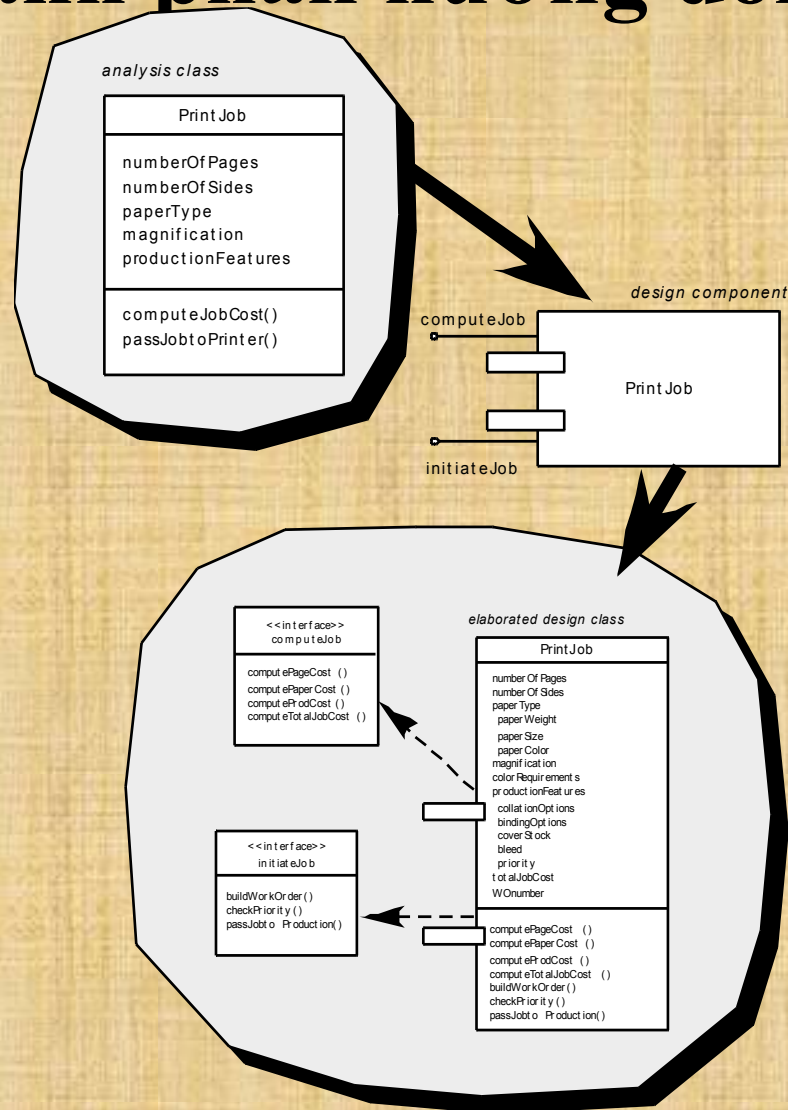
Thiết kế phần mềm

- Nội dung
 1. Tổng quan thiết kế phần mềm
 2. Thiết kế kiến trúc phần mềm
 - 3. Thiết kế chi tiết phần mềm**
 4. Thiết kế giao diện người dùng
 5. Thiết kế mẫu
 6. Thiết kế giao diện cho ứng dụng WebApp

Thế nào là một thành phần?

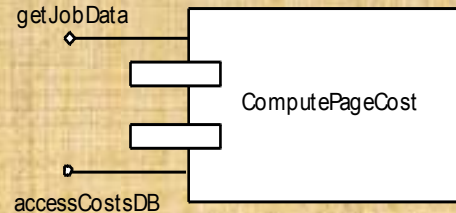
- *OMG Unified Modeling Language Specification* [OMG01] định nghĩa một thành phần (component) như là:
 - Một phần có tính mô-đun, có thể triển khai và thay thế của một hệ thống mà đóng gói việc thực thi và đưa ra một tập các giao diện.
- **Góc nhìn hướng đối tượng:** Một thành phần bao gồm một tập các lớp cộng tác.
- Góc nhìn quy ước: Một thành phần bao gồm logic xử lý, cấu trúc dữ liệu bên trong cần thiết để triển khai logic đó và một giao diện cho phép thành phần được gọi và dữ liệu được truyền đến nó.

Thành phần hướng đối tượng

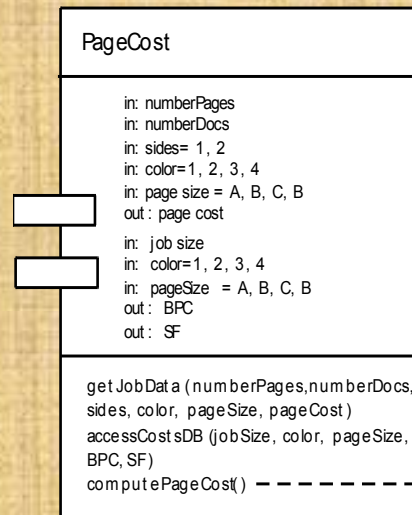


Thành phần quy ước

design component



elaborated module



job size (JS) =
 numberPages * numberDocs;
 lookup base page cost (BPC) -->
 accessCostsDB (JS, color);
 lookup size factor (SF) -->
 accessCostDB (JS, color, size)
 job complexity factor (JCF) =
 1 + [(sides-1) * sideCost + SF]
 pagecost = BPC * JCF

Nguyên lý thiết kế cơ bản

- **Nguyên lý mở-đóng (OCP):** Một mô-đun (thành phần) nên được mở cho việc mở rộng nhưng nên đóng cho việc hiệu chỉnh.
- **Nguyên lý thay thế Liskov (LSP):** Các lớp con nên thay thế được các lớp gốc.
- **Nguyên lý trao đổi phụ thuộc (DIP):** Phụ thuộc vào trừu tượng hóa, không phụ thuộc vào kết cấu.
- **Nguyên lý tách biệt giao diện (ISP):** “Nhiều giao diện khách hàng cụ thể thì tốt hơn một giao diện mục đích chung.”
- **Nguyên lý phát hành và tái sử dụng tương đương (DEP):** “Hạt nhỏ của việc tái sử dụng cũng là hạt nhỏ của việc phát hành”
- **Nguyên lý đóng chung (CCP).** “Các lớp thay đổi cùng nhau thì thuộc về nhau”
- **Nguyên lý tái sử dụng chung (CRP).** “Các lớp không được tái sử dụng cùng nhau thì không nên nhóm lại với nhau”

Source: Martin, R., “Design Principles and Design Patterns,” downloaded from <http://www.objectmentor.com>, 2000.

Hướng dẫn thiết kế

- **Thành phần:**
 - Quy ước đặt tên nên được thiết lập cho các thành phần được xác định như là một phần của mô hình kiến trúc rồi sau đó tinh chỉnh và xây dựng như là một phần của mô hình mức thành phần
- **Giao diện:**
 - Giao diện cung cấp thông tin quan trọng về sự giao tiếp và cộng tác (đồng thời cũng giúp chúng ta đạt được OPC)
- **Phụ thuộc và kế thừa:**
 - Việc mô hình hóa sự phụ thuộc từ trái sang phải và sự kế thừa từ dưới lên trên.

Sự gắn kết (cohesion)

- Góc nhìn quy ước:
 - Một “tư duy đơn lẻ” của một mô-đun.
- Góc nhìn hướng đối tượng:
 - Sự gắn kết nghĩa là một thành phần hoặc một lớp chỉ đóng gói các thuộc tính và hoạt động liên quan chặt chẽ với nhau và với bản thân thành phần hoặc lớp đó.



Sự gắn kết (cohesion)

- Các mức của sự gắn kết:
 - Chức năng
 - Tầng
 - Truyền thông
 - Liên tục
 - Thủ tục
 - Phụ thuộc thời gian
 - Lợi ích



Ghép nối (coupling)

- Góc nhìn quy ước:
 - Là mức độ mà một thành phần kết nối với các thành phần khác và với thế giới bên ngoài.
- Góc nhìn hướng đối tượng:
 - Một thước đo chất lượng mức độ kết nối của các lớp với lớp khác.
- Các mức của sự ghép nối:
 - Nội dung
 - Chung
 - Điều khiển
 - Đánh dấu
 - Dữ liệu
 - Lời gọi công thức
 - Loại sử dụng
 - Sự lồng ghép và bao hàm.
 - Bên ngoài

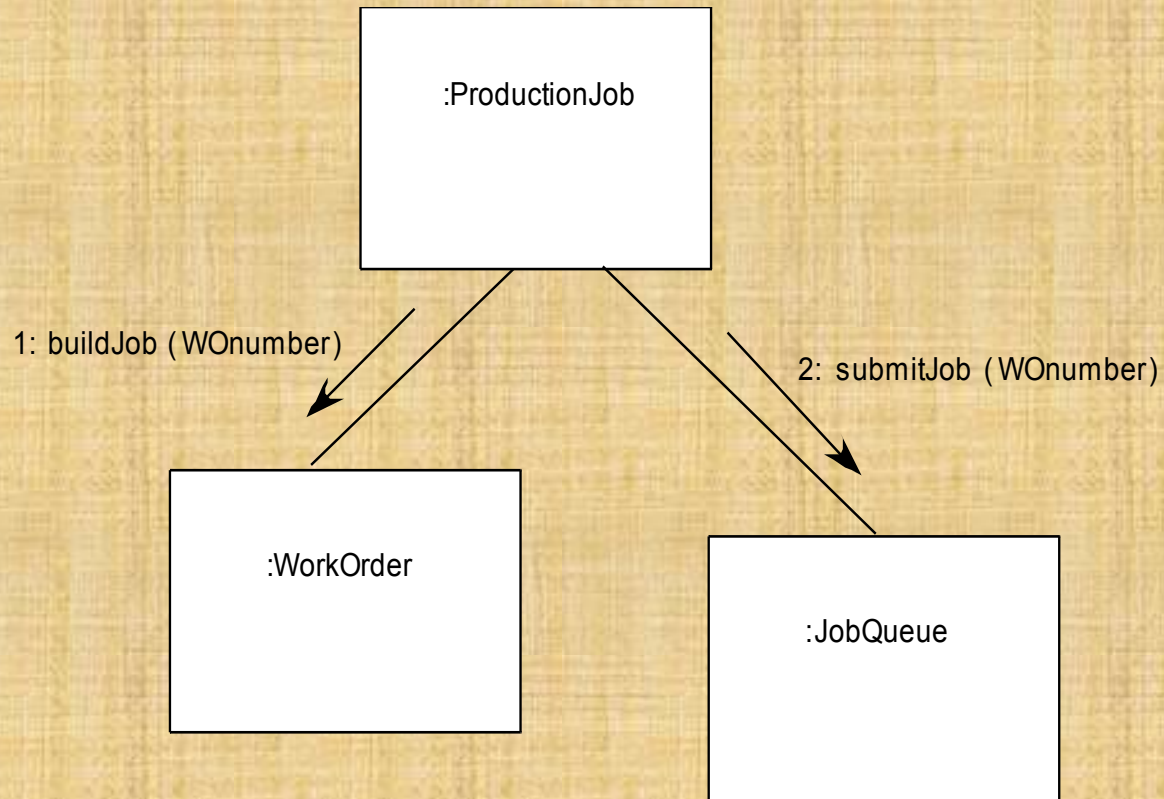
Thiết kế mức thành phần - I

- Bước 1. Xác định tất cả các lớp thiết kế tương ứng với miền vấn đề.
- Bước 2. Xác định tất cả các lớp thiết kế tương ứng với kết cấu hạ tầng.
- Bước 3. Xây dựng tất cả các lớp không có được như là thành phần tái sử dụng.
- Bước 3a. Xác định chi tiết thông điệp khi các lớp hoặc các thành phần cộng tác.
- Bước 3b. Xác định các giao diện thích hợp cho mỗi thành phần.

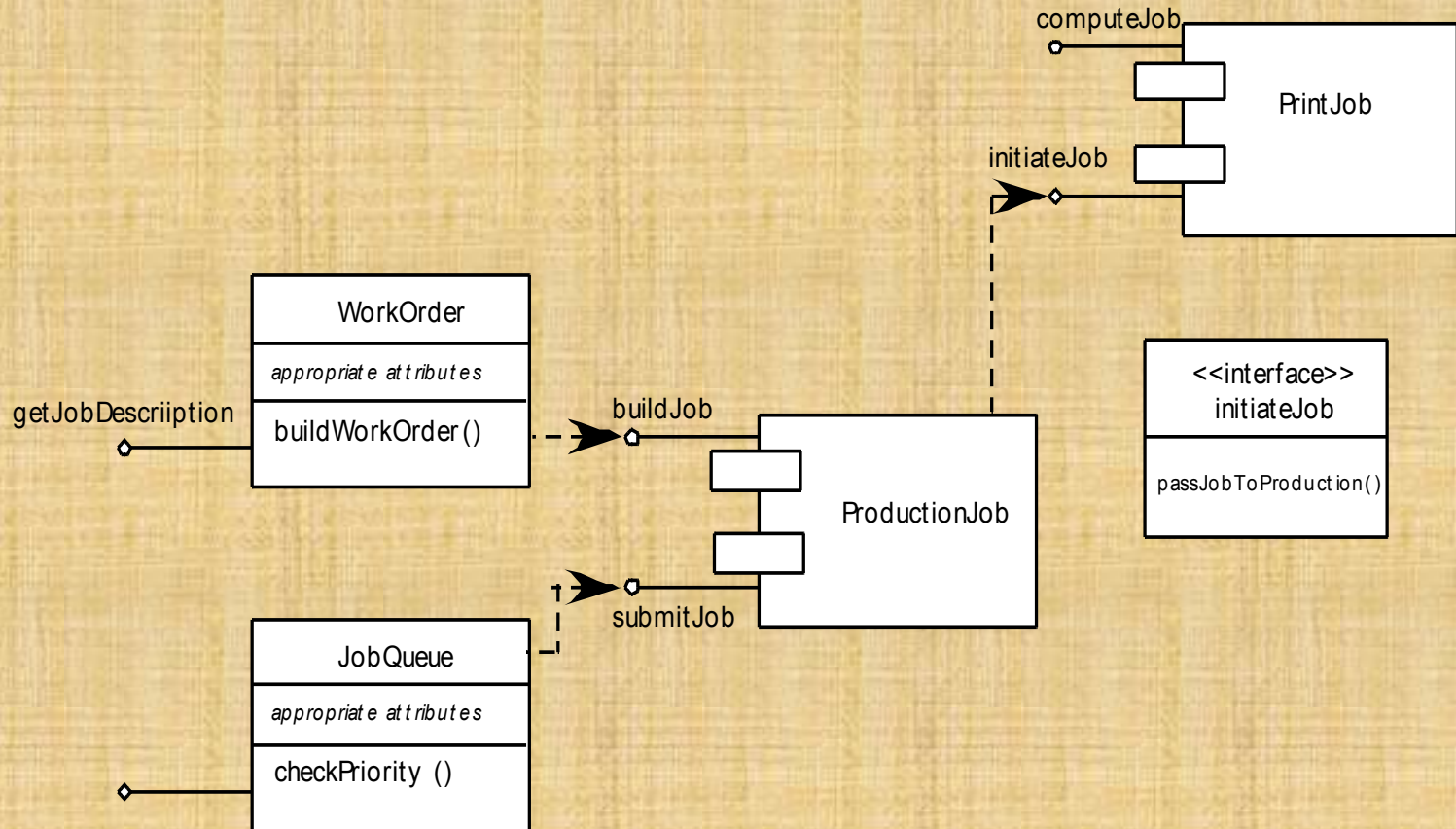
Thiết kế mức thành phần - II

- Step 3c. Xây dựng các thuộc tính và xác định kiểu dữ liệu và cấu trúc dữ liệu cần thiết để thực hiện chúng.
- Step 3d. Mô tả luồng xử lý trong từng hoạt động cụ thể.
- Step 4. Mô tả các nguồn dữ liệu bền vững (cơ sở dữ liệu và các tập tin) và xác định các lớp cần thiết để quản lý chúng.
- Step 5. Phát triển và xây dựng biểu diễn hành vi cho một lớp hoặc một thành phần.
- Step 6. Xây dựng sơ đồ triển khai để cung cấp thêm thông tin về chi tiết thực hiện.
- Step 7. Nhân tố hóa mỗi thể hiện thiết kế mức thành phần và luôn luôn xem xét phương án thay thế.

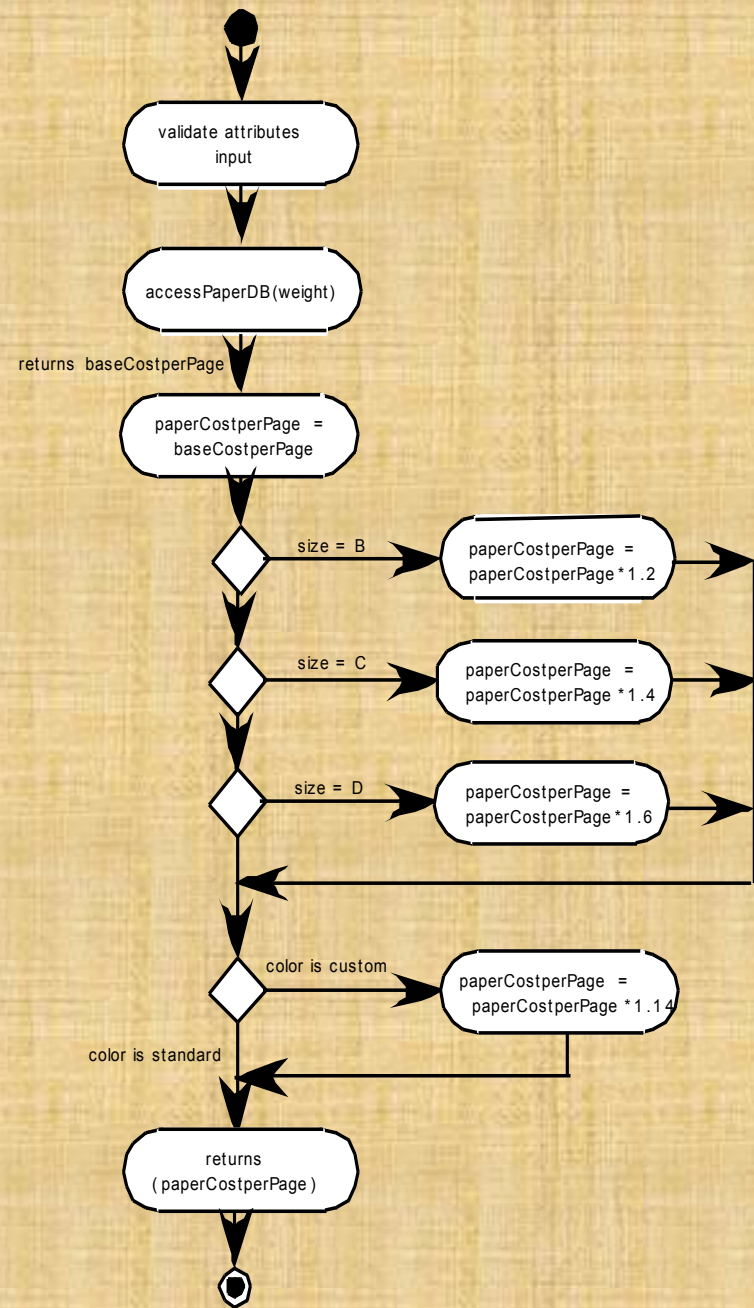
Sơ đồ cộng tác



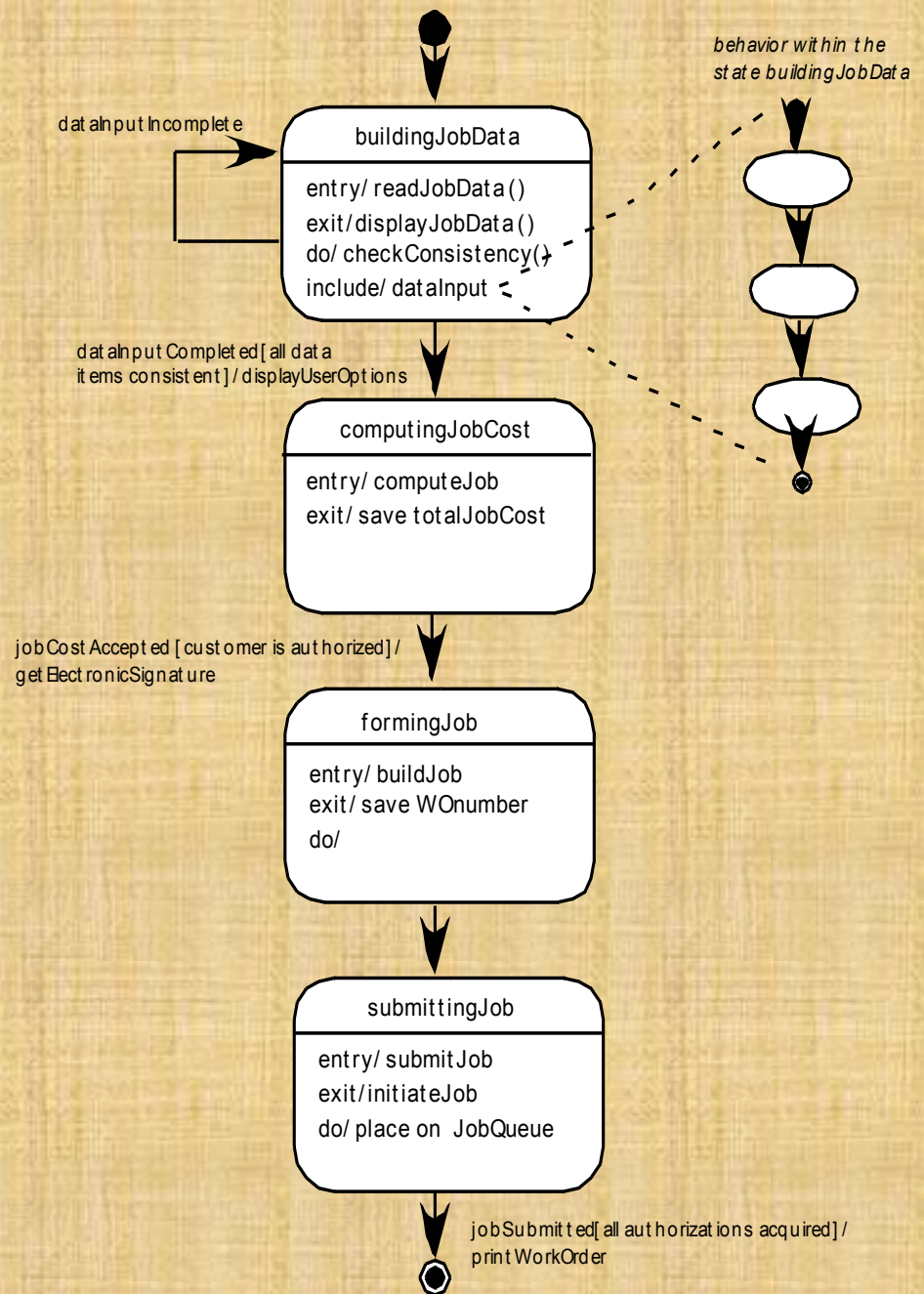
Tái cấu trúc



Sơ đồ hoạt động



Sơ đồ trạng thái



Thiết kế thành phần cho WebApps

- Thành phần WebApp là:
 - (1) Một chức năng được xác định rõ và có tính kết hợp, thao tác nội dung hoặc cung cấp quá trình tính toán hoặc xử lý dữ liệu cho người dùng cuối hoặc:
 - (2) Một gói có tính kết hợp của nội dung và chức năng, cung cấp cho người dùng cuối các khả năng được yêu cầu.
- Vì vậy, thiết kế mức thành phần cho WebApps thường kết hợp các yếu tố của thiết kế nội dung và thiết kế chức năng.

Thiết kế nội dung cho WebApps

- Tập trung vào các đối tượng nội dung và cách thức mà chúng có thể được đóng gói để trình bày cho một người dùng cuối.
- Hãy xem xét một khả năng giám sát video trên nền web tại SafeHomeAssured.com
 - Các thành phần nội dung tiềm năng có thể được xác định cho khả năng giám sát video:
 - » (1) các đối tượng nội dung biểu diễn cách bố trí không gian (kế hoạch sàn) với các biểu tượng thêm vào để đại diện cho vị trí của cảm biến và máy quay video;
 - » (2) tập hợp các hình ảnh thu nhỏ và đoạn video (cho mỗi đối tượng dữ liệu riêng biệt) và
 - » (3) cửa sổ quay video cho từng camera riêng biệt.
 - Mỗi một thành phần trên có thể được đặt tên và xử lý một cách riêng biệt như là một gói.

Thiết kế chức năng cho WebApps

- Các ứng dụng Web hiện đại cung cấp các chức năng xử lý ngày càng phức tạp:
 - (1) thực hiện xử lý địa phương để tạo ra nội dung và khả năng chuyển hướng theo kiểu động;
 - (2) cung cấp khả năng tính toán hoặc xử lý dữ liệu thích hợp cho miền nghiệp vụ của WebApps;
 - (3) cung cấp truy vấn hoặc truy cập CSDL phức tạp, hoặc
 - (4) thiết lập giao diện dữ liệu với hệ thống hợp tác bên ngoài.
- Để đạt được những khả năng này (và nhiều khả năng khác), bạn sẽ thiết kế và xây dựng thành phần chức năng của WebApp giống hệt về dạng với các thành phần phần mềm trong phần mềm quy ước.

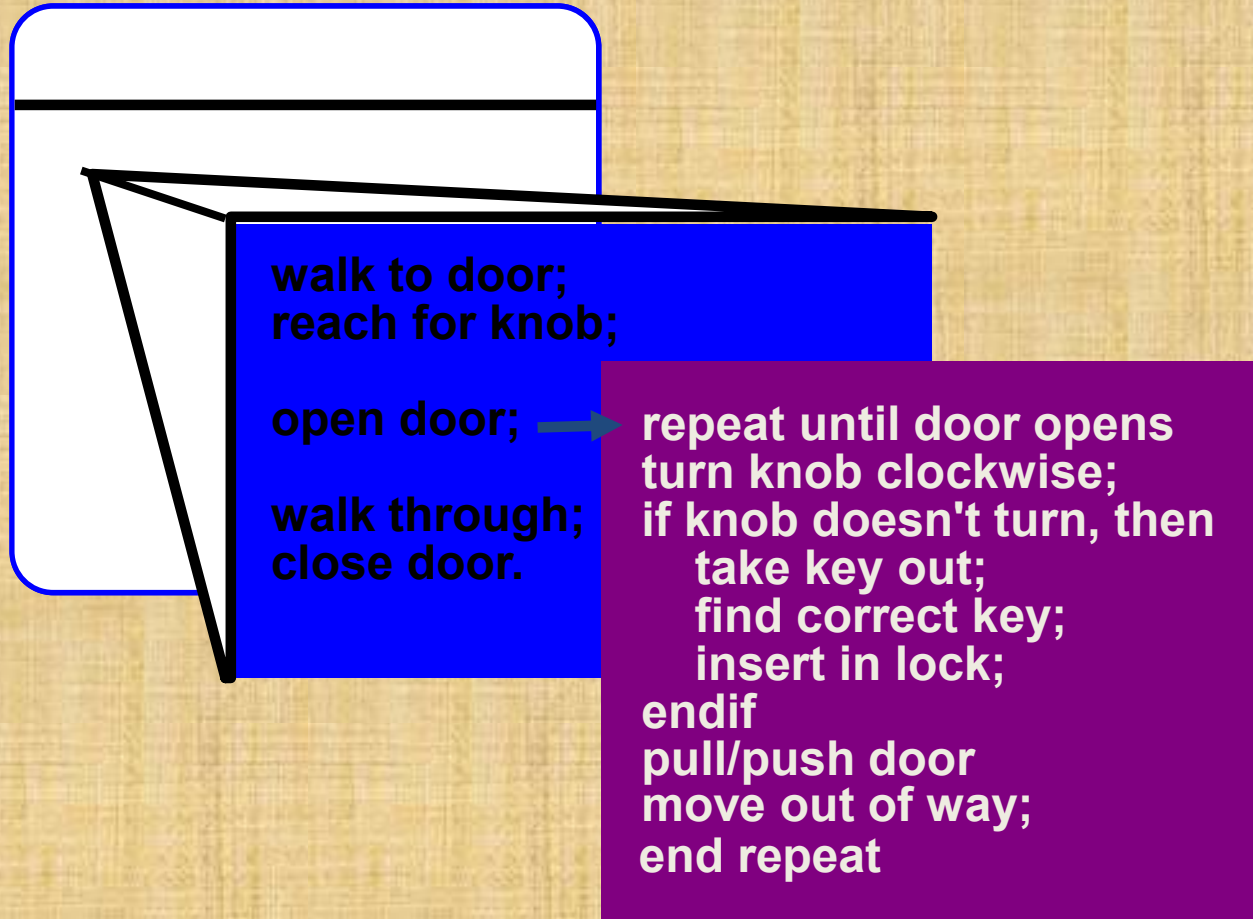
Thiết kế thành phần quy ước

- Việc thiết kế các xử lý logic được quy định bởi các nguyên tắc cơ bản của thiết kế thuật toán và lập trình có cấu trúc.
- Việc thiết kế các cấu trúc dữ liệu được định nghĩa bởi các mô hình dữ liệu được phát triển cho hệ thống.
- Việc thiết kế các giao diện được quy định bởi sự hợp tác mà một thành phần phải có ảnh hưởng.

Thiết kế thuật toán

- Là hoạt động thiết kế sát nhất với việc coding.
- Cách tiếp cận:
 - xem xét mô tả thiết kế cho các thành phần
 - sử dụng tinh chỉnh từng bước để phát triển thuật toán
 - sử dụng lập trình có cấu trúc để thực hiện logic có tính thủ tục
 - sử dụng ‘phương pháp chính thống’ để chứng minh logic.

Tinh chỉnh từng bước



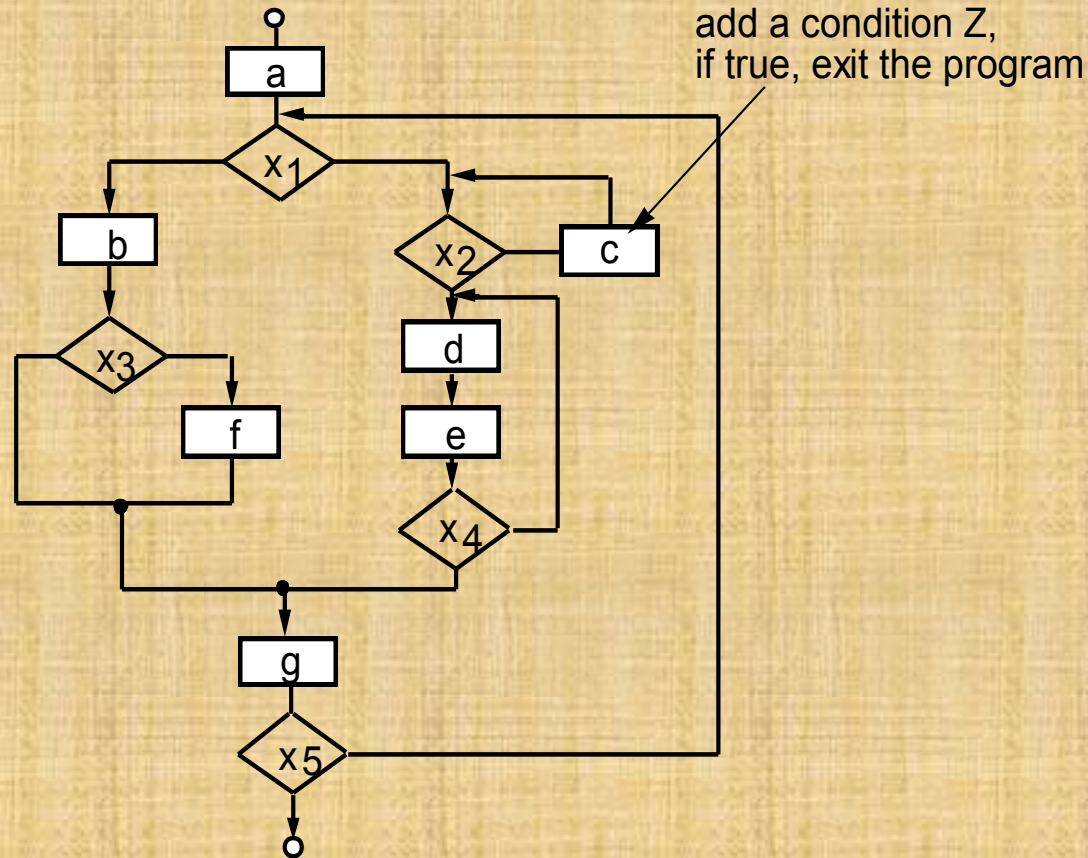
Mô hình thiết kế thuật toán

- Trình bày các thuật toán ở mức độ chi tiết mà có thể được xem xét lại chất lượng.
- Tùy chọn:
 - Đồ họa (e.g. flowchart, box diagram)
 - Mã giả (e.g., PDL)
 - Ngôn ngữ lập trình
 - Bảng quyết định

Lập trình cấu trúc

- Sử dụng một tập hạn chế các cấu trúc logic:
 - ▣ *Chuỗi*
 - ▣ *Điều kiện* — if-then-else, select-case
 - ▣ *Vòng lặp* — do-while, repeat until
- Dẫn đến những đoạn mã dễ đọc, dễ kiểm thử.
- Có thể sử dụng kết hợp với “chứng minh tính đúng đắn”
- Rất quan trọng để có thể đạt được chất lượng tốt,
- nhưng chưa đủ.

Một thiết kế thủ tục có cấu trúc

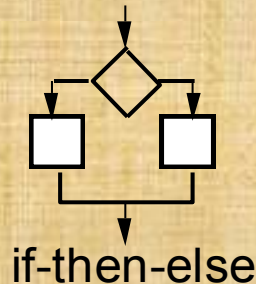


Bảng quyết định

Rules

Conditions	1	2	3	4	5	6
regular customer	T	T				
silver customer			T	T		
gold customer					T	T
special discount	F	T	F	T	F	T
Rules						
no discount	✓					
apply 8 percent discount			✓	✓		
apply 15 percent discount					✓	✓
apply additional x percent discount		✓		✓		✓

Ngôn ngữ lập trình thiết kế (PDL)



```
if condition x  
  then process a;  
  else process b;  
endif
```

PDL

- ☐ easy to combine with source code
- ☐ machine readable, no need for graphics input
- ☐ graphics can be generated from PDL
- ☐ enables declaration of data as well as procedure
- ☐ easier to maintain

Vì sao chọn ngôn ngữ thiết kế?

- Có thể được bắt nguồn từ HOL của lựa chọn
- Máy móc có thể hiểu và xử lý được.
- Có thể được nhúng với mã nguồn, do đó dễ bảo trì hơn.
- Có thể được trình bày rất chi tiết, nếu người thiết kế và người lập trình là khác nhau
- Dễ dàng xem xét lại

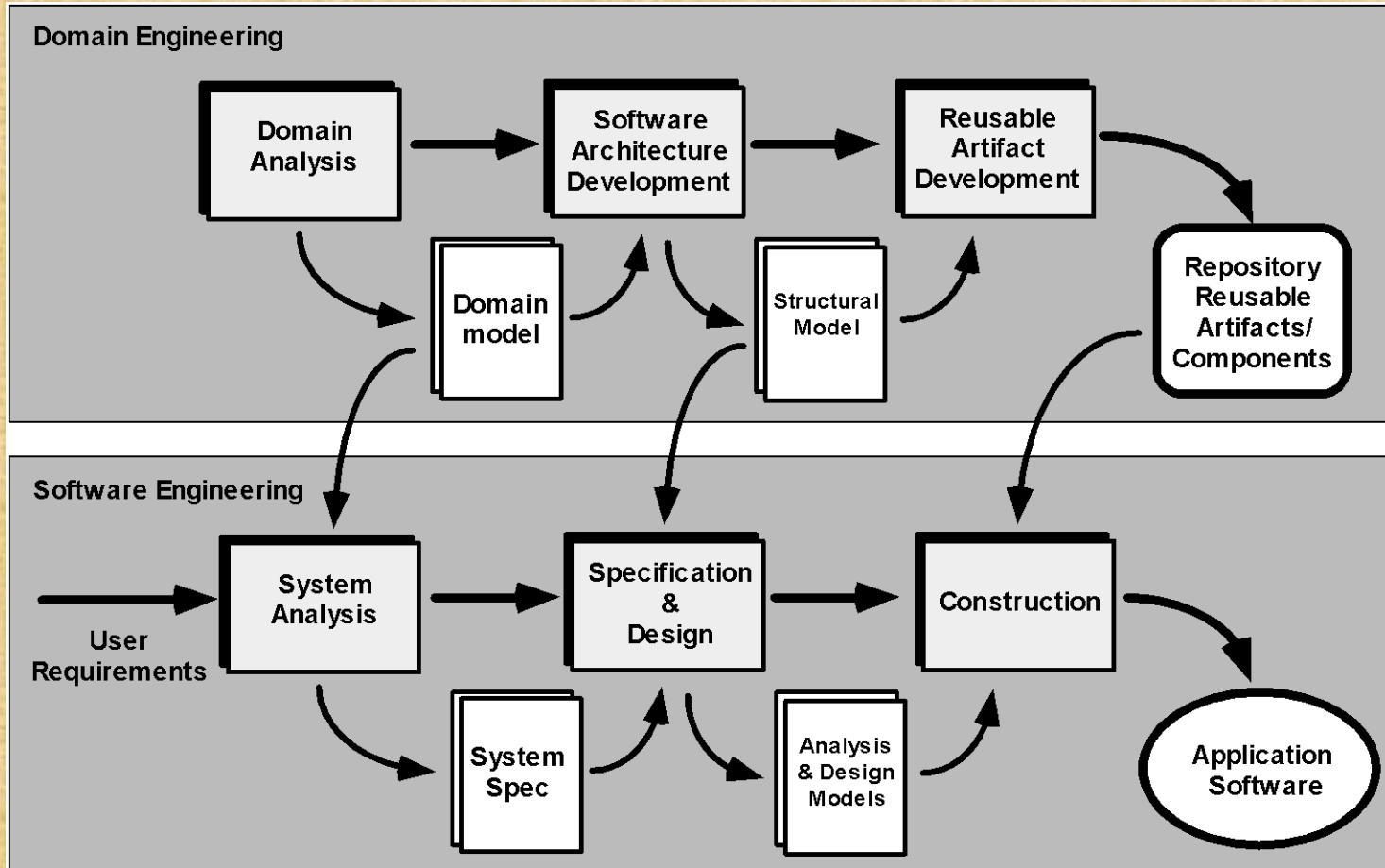
Sự phát triển dựa trên thành phần

- Khi phải đối mặt với khả năng tái sử dụng, nhóm phần mềm xem xét:
 - Liệu các thành phần thương mại off-the-shelf có sẵn để triển khai các yêu cầu?
 - Liệu thành phần tái sử dụng phát triển bên trong có sẵn để thực hiện các yêu cầu?
 - Liệu các giao diện cho các thành phần có sẵn có tương thích trong kiến trúc của hệ thống được xây dựng?
- Đồng thời, họ cũng đang phải đối mặt với những trở ngại sau đây để tái sử dụng ...

Trở ngại để tái sử dụng

- Rất ít công ty và các tổ chức có một kế hoạch tái sử dụng phần mềm toàn diện dù chỉ ở mức khá nghiêm túc.
- Mặc dù ngày càng nhiều nhà cung cấp phần mềm hiện nay bán các công cụ hoặc thành phần cung cấp sự hỗ trợ trực tiếp cho việc tái sử dụng phần mềm, phần lớn nhà phát triển không sử dụng chúng.
- Tương đối ít sự huấn luyện có sẵn để giúp các kỹ sư và các nhà quản lý phần mềm hiểu và áp dụng tái sử dụng.
- Nhiều học viên phần mềm tiếp tục tin rằng tái sử dụng là "rắc rối hơn là giá trị."
- Nhiều công ty tiếp tục khuyến khích các phương pháp phát triển phần mềm không áp dụng tái sử dụng
- Rất ít công ty cung cấp ưu đãi để sản xuất các chương trình tái sử dụng.

Quá trình CBSE



These slides are designed to accompany Software Engineering:
A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Kỹ thuật miền

1. Xác định các miền được nghiên cứu..
2. Phân loại các mặt hàng được xuất từ miền.
3. Thu thập một mẫu đại diện của các ứng dụng trong miền.
4. Phân tích mỗi ứng dụng trong mẫu.
5. Xây dựng một mô hình phân tích cho các đối tượng.

Xác định các thành phần tái sử dụng

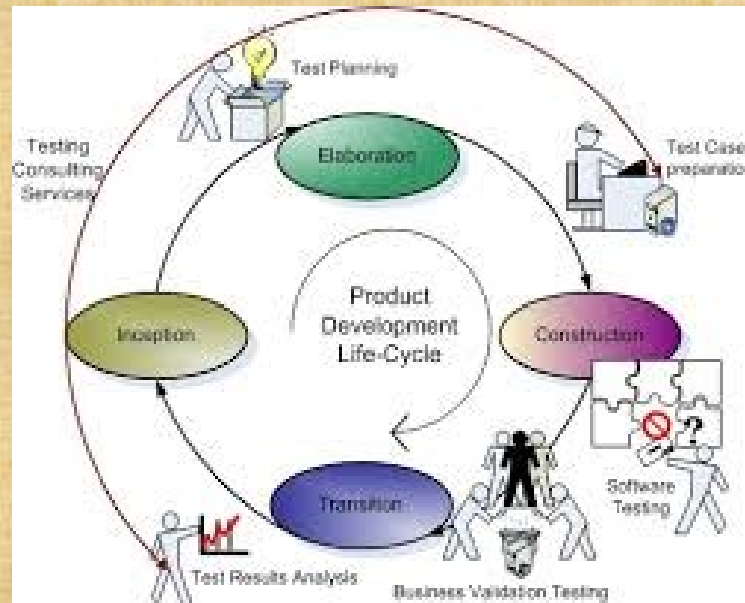
- Chức năng thành phần có cần thiết cho việc triển khai trong tương lai?
- Mức độ phổ biến của chức năng của các thành phần trong các tên miền là như thế nào?
- Có trùng lặp chức năng của các thành phần trong các tên miền không?
- Thành phần có phụ thuộc phần cứng không?
- Liệu các phần cứng có giữ nguyên trạng trong quá trình triển khai?
- Chi tiết cụ thể phần cứng có thể được loại bỏ cho thành phần khác?
- Liệu thiết kế có đủ tối ưu hóa cho bước triển khai tiếp theo?
- Liệu chúng ta có thể tham số hóa một thành phần không thể tái sử dụng để nó trở nên tái sử dụng?
- Liệu có thể tái sử dụng các thành phần trong nhiều lần triển khai với chỉ những thay đổi nhỏ?
- Việc tái sử dụng thông qua sửa đổi có khả thi?
- Một thành phần không thể tái sử dụng có thể được phân tách để tạo ra phần tái sử dụng?
- Việc phân tách thành phần cho tái sử dụng hợp lệ như thế nào?

Kỹ thuật phần mềm dựa trên thành phần (CBSE)

- Một thư viện các thành phần phải sẵn có
- Các thành phần cần có một cấu trúc nhất quán
- Nên tồn tại một tiêu chuẩn, ví dụ,
 - OMG / CORBA
 - Microsoft COM
 - Sun JavaBeans

Các hoạt động CBSE

- Tiêu chuẩn thành phần
- Thích ứng thành phần
- Ghép nối thành phần
- Cập nhật thành phần



These slides are designed to accompany Software Engineering:
A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Tiêu chuẩn

Trước khi một thành phần có thể sử dụng, bạn phải xem xét:

- Giao diện lập trình ứng dụng (API)
- Các công cụ phát triển và tích hợp theo yêu cầu của các thành phần
- Các yêu cầu tại thời điểm chạy bao gồm cả sử dụng tài nguyên (ví dụ, bộ nhớ hoặc lưu trữ), thời gian hay tốc độ, và giao thức mạng
- Các yêu cầu dịch vụ bao gồm cả giao diện hệ điều hành và hỗ trợ từ các thành phần khác.
- Các tính năng bảo mật bao gồm kiểm soát truy cập và giao thức xác thực
- giả định thiết kế nhúng bao gồm cả việc sử dụng các thuật toán số học hoặc phi số học cụ thể
- Xử lý ngoại lệ

Sự thích ứng

Ý nghĩa của 'đề dằn tích hợp' là:

1. các phương pháp nhất quán trong quản trị tài nguyên đã được triển khai cho tất cả các thành phần trong thư viện;
2. có hoạt động chung như quản lý dữ liệu tồn tại cho tất cả các thành phần, và
3. có giao diện trong kiến trúc và với môi trường bên ngoài đã được triển khai một cách nhất quán.

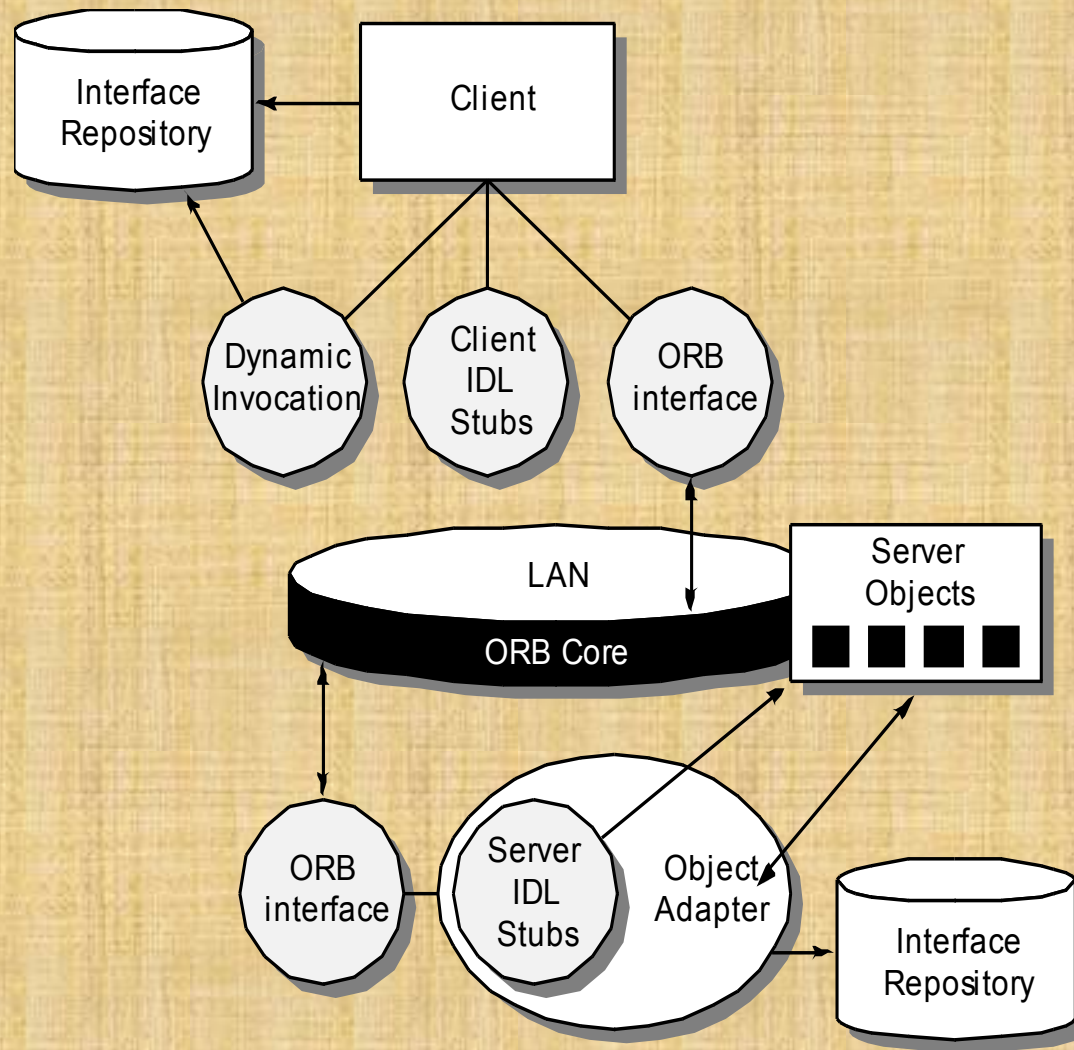
Ghép nối (composition)

- Một cơ sở hạ tầng phải được thiết lập để ràng buộc các thành phần với nhau
- Thành phần kiến trúc cho các thành phần bao gồm:
 - Mô hình trao đổi dữ liệu
 - Tự động hóa
 - Lưu trữ có cấu trúc
 - Mô hình đối tượng tiềm ẩn

OMG/ CORBA

- Nhóm quản lý đối tượng (Object Management Group) đã xuất bản một kiến trúc môi giới yêu cầu đối tượng chung (*common object request broker architecture*) (OMG/CORBA).
- Một sự môi giới yêu cầu đối tượng (ORB) cung cấp dịch vụ cho phép các thành phần (đối tượng) giao tiếp với các thành phần khác, bất kể vị trí của chúng trong hệ thống.
- Sự tích hợp các thành phần CORBA (mà không sửa đổi) trong một hệ thống được đảm bảo nếu một ngôn ngữ định nghĩa giao diện (IDL) được tạo ra cho mỗi thành phần.
- Các đối tượng trong các ứng dụng khách yêu cầu một hoặc nhiều dịch vụ từ máy chủ ORB. Yêu cầu được thực hiện thông qua một IDL hoặc tự động tại thời điểm chạy.
- Một kho giao diện chứa tất cả các thông tin cần thiết về định dạng của yêu cầu và hồi đáp của dịch vụ.

Kiến trúc ORB



Microsoft COM

- Mô hình đối tượng thành phần (COM) cung cấp một đặc tả cho việc sử dụng các thành phần được sản xuất bởi các nhà cung cấp khác nhau trong một ứng dụng duy nhất chạy dưới hệ điều hành Windows.
- COM bao gồm 2 thành phần:
 - Giao diện COM (được triển khai như đối tượng COM)
 - một tập hợp các cơ chế đăng ký và truyền các thông điệp giữa các giao diện COM.

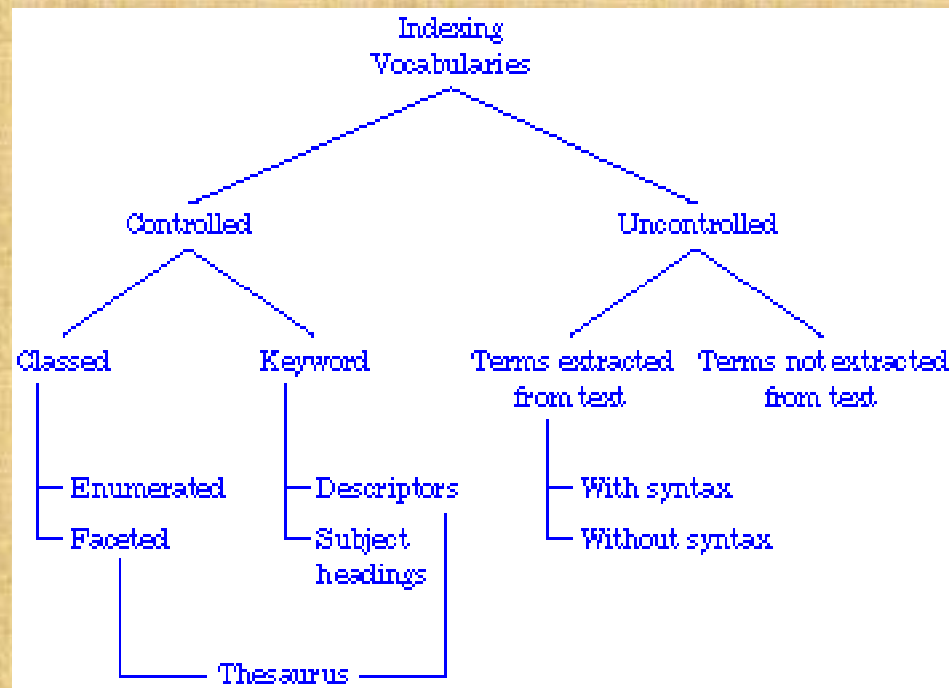
Sun JavaBeans

- Hệ thống thành phần JavaBeans là một cơ sở hạ tầng CBSE di động, độc lập nền tảng được phát triển sử dụng ngôn ngữ lập trình Java.
- Hệ thống thành phần JavaBeans bao gồm một tập các công cụ, được gọi là *Bean Development Kit* (BDK) cho phép nhà phát triển:
 - phân tích sự hoạt động của một Beans (thành phần).
 - tùy chỉnh hành vi và sự hình thức của chúng
 - thiết lập cơ chế phối hợp và truyền thông
 - phát triển các Beans tùy chỉnh để sử dụng trong một ứng dụng cụ thể
 - kiểm tra và đánh giá hành vi của Beans

Phân loại

- **Phân loại liệt kê:** các linh kiện được mô tả bằng cách định nghĩa một cấu trúc phân cấp trong đó các lớp và mức độ khác nhau của các lớp con của các thành phần phần mềm được xác định
- **Phân loại nhiều mặt:** một khu vực miền được phân tích và một tập hợp các tính năng mô tả cơ bản được xác định
- **Phân loại thuộc tính-giá trị:** một tập các thuộc tính được định nghĩa cho tất cả các thành phần trong một khu vực miền.

Đánh chỉ mục



These slides are designed to accompany Software Engineering:
A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Môi trường tái sử dụng

- Một cơ sở dữ liệu thành phần có khả năng lưu trữ các thành phần phần mềm và thông tin phân loại cần thiết để lấy chúng.
- Một hệ thống quản lý thư viện cung cấp quyền truy cập vào cơ sở dữ liệu.
- Một hệ thống truy xuất thành phần phần mềm cho phép một ứng dụng khách có thể lấy các thành phần và dịch vụ từ các máy chủ thư viện.
- Công cụ CBSE có hỗ trợ việc tích hợp các thành phần tái sử dụng trong một thiết kế hoặc thực hiện mới.

Thiết kế phần mềm

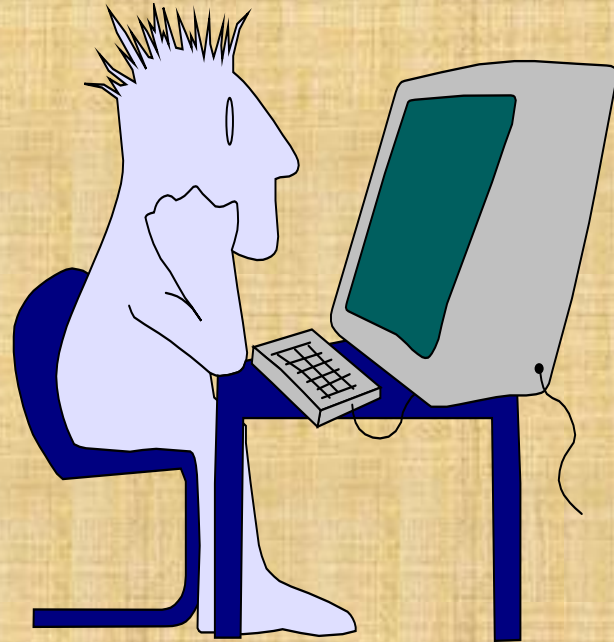
- Nội dung
 1. Tổng quan thiết kế phần mềm
 2. Thiết kế kiến trúc phần mềm
 3. Thiết kế chi tiết phần mềm
 - 4. Thiết kế giao diện người dùng**
 5. Thiết kế mẫu
 6. Thiết kế giao diện cho ứng dụng WebApp

Thiết kế giao diện

Dễ học?

Dễ sử dụng?

Dễ hiểu?

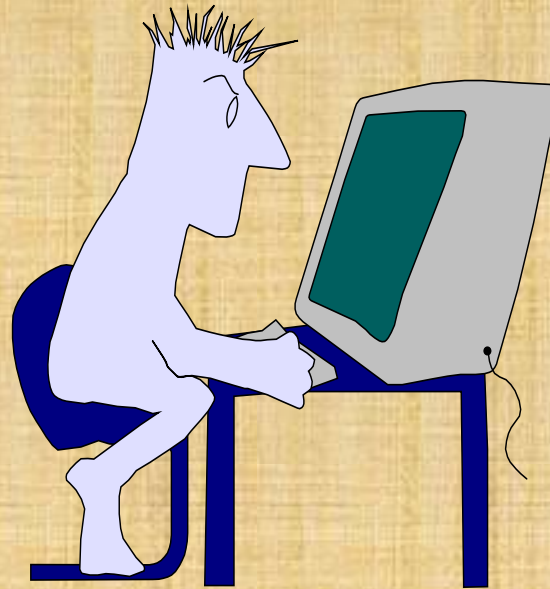


These slides are designed to accompany Software Engineering:
A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Thiết kế giao diện

Lỗi thiết kế thông thường

thiếu nhất quán
quá nhiều ghi nhớ
không có hướng dẫn / giúp đỡ
không nhạy cảm với ngữ cảnh
đáp ứng kém
Phức tạp / không thân thiện



Quy tắc vàng

- Đặt người dùng trong sự kiểm soát
- Giảm tải bộ nhớ cho người dùng
- Làm cho giao diện nhất quán



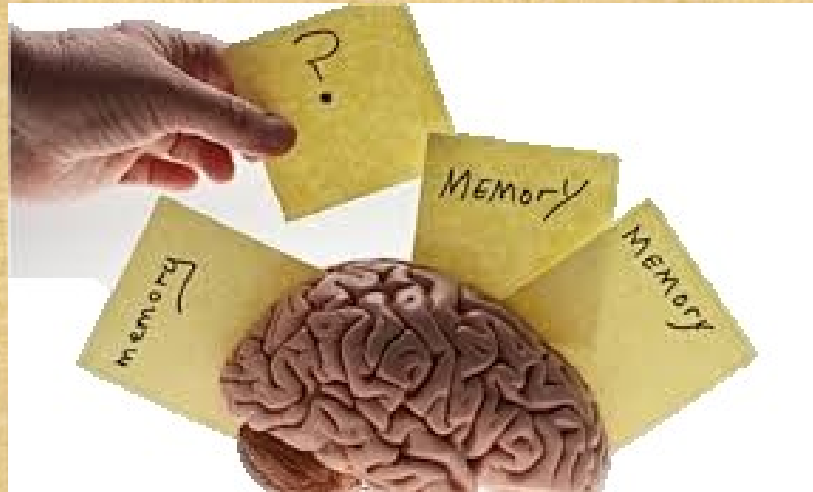
Source: Martin, R., "Design Principles and Design Patterns," downloaded from <http://www.objectmentor.com>, 2000.

Đặt người dùng trong sự kiểm soát

- Xác định phương thức tương tác theo một cách mà không ép người dùng tới những hành động không cần thiết hoặc không mong muốn.
- Cung cấp sự tương tác linh hoạt.
- Cho phép tương tác người dùng được ngắt và hoàn tác .
- Hợp lý hóa tương tác như trình độ kỹ năng cao và cho phép tùy chỉnh tương tác.
- Ẩn kỹ thuật bên trong với người sử dụng bình thường.
- Thiết kế cho tương tác trực tiếp với các đối tượng xuất hiện trên màn hình.

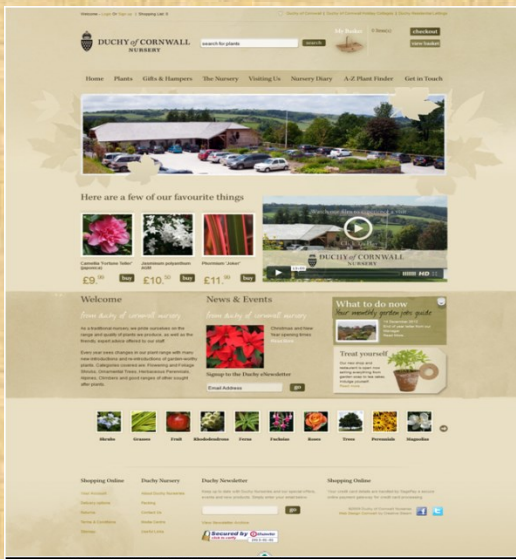
Giảm tải bộ nhớ cho người dùng

- Giảm nhu cầu về bộ nhớ ngắn hạn.
- Thiết lập các mặc định có ý nghĩa.
- Xác định các phím tắt trực quan.
- Các thiết kế trực quan của giao diện phải được dựa trên một phép ẩn dụ thế giới thực.
- Tiết lộ thông tin theo kiểu lũy tiến.



Làm cho giao diện nhất quán

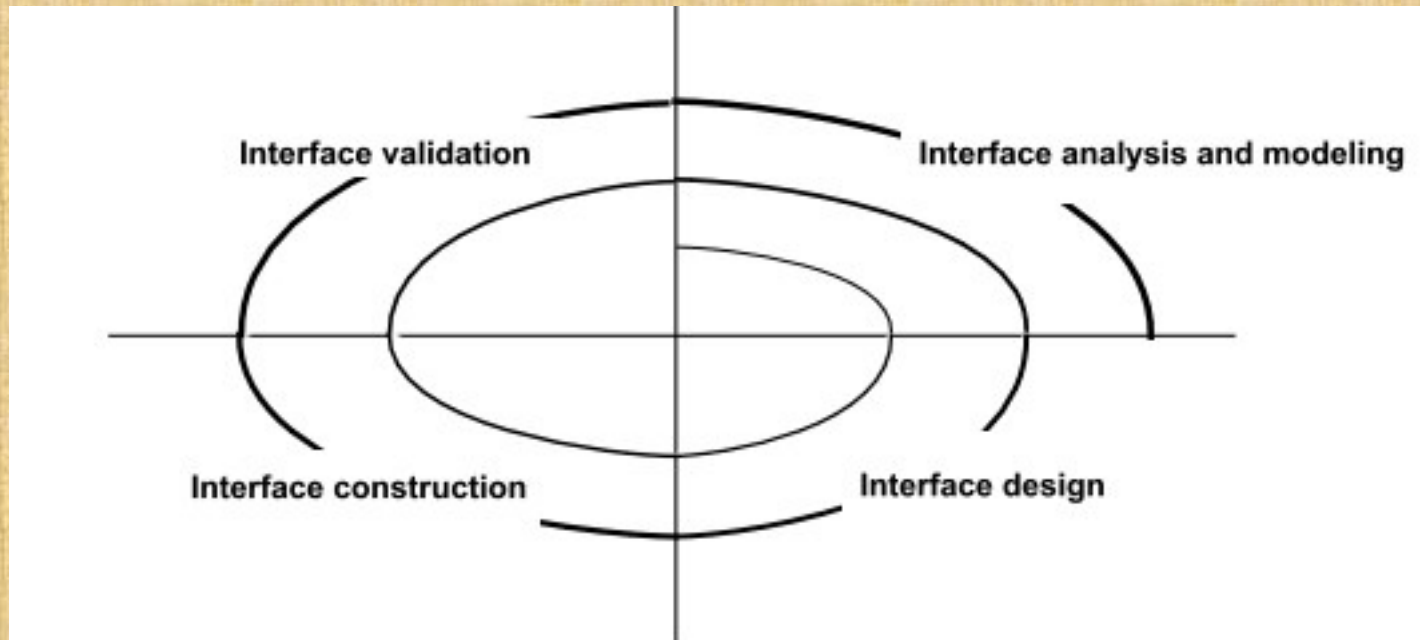
- Cho phép người sử dụng đưa các tác vụ hiện hành vào một ngữ cảnh có ý nghĩa.
- Duy trì tính nhất quán giữa một họ các ứng dụng.
- Nếu mô hình tương tác cũ đã tạo ra những kỳ vọng của người dùng, không làm thay đổi trừ khi có một lý do thuyết phục để làm như vậy.



Mô hình thiết kế giao diện người dùng

- **Mô hình người dùng** — một hồ sơ của tất cả người dùng cuối của hệ thống
- Mô hình thiết kế — một nhận thức thiết kế của các mô hình sử dụng
- **Mô hình về tinh thần (nhận thức hệ thống)** — hình ảnh nhận thức của người dùng về những gì giao diện là
- **Mô hình triển khai** — giao diện "nhìn và cảm nhận" cùng với thông tin hỗ trợ mô tả cú pháp và ngữ nghĩa giao diện

Quá trình thiết kế giao diện người dùng



Phân tích giao diện

- Phân tích giao diện nghĩa là hiểu được:
 - (1) những người (người dùng cuối), người sẽ tương tác với các hệ thống thông qua giao diện;
 - (2) các tác vụ mà người dùng phải thực hiện để làm công việc của họ,
 - (3) các nội dung được trình bày như là một phần của giao diện
 - (4) môi trường trong đó những công việc này sẽ được tiến hành.

Phân tích người dùng

- Liệu người dùng là các chuyên gia đã qua đào tạo, kỹ thuật viên, văn thư hay là công nhân sản xuất?
- Mức độ giáo dục trung bình của người dùng là gì?
- Liệu người dùng có thể học từ những tài liệu viết hay họ bày tỏ mong muốn một chương trình luyện tập kiểu lớp học?
- Người dùng là những chuyên gia đánh máy hay họ sợ bàn phím?
- Độ tuổi của cộng đồng người dùng là bao nhiêu?
- Liệu người dùng sẽ được đại diện chủ yếu bởi một giới tính?
- Người dùng sẽ trả cho công việc họ thực hiện như thế nào?
- Liệu người dùng chỉ làm việc trong giờ công sở hay họ sẽ làm việc đến khi công việc hoàn thành?
- Liệu phần mềm sẽ trở thành một phần quan trọng trong công việc của người dùng hay nó sẽ chỉ thỉnh thoảng được sử dụng?
- Ngôn ngữ chính giữa các người dùng là gì?
- Sẽ có những hệ quả nào nếu người dùng gây ra lỗi khi sử dụng hệ thống?
- Liệu người dùng có phải là chuyên gia trong lĩnh vực liên quan được xử lý bởi hệ thống?
- Liệu người dùng có muốn biết về công nghệ phía sau giao diện?

Phân tích tác vụ và mô hình hóa

- Trả lời những câu hỏi sau...
 - Công việc gì mà người dùng sẽ thực hiện trong những trường hợp cụ thể?
 - Tác vụ hay tác vụ con nào sẽ được thực hiện khi người dùng làm việc?
 - Những vấn đề miền đối tượng cụ thể nào mà người dùng sẽ thao tác khi công việc được thực hiện?
 - Chuỗi các nhiệm vụ-quy trình là gì?
 - Hệ thống cấp bậc của tác vụ là gì?
- Use-cases xác định tương tác cơ bản.
- Xây dựng tác vụ điều chỉnh các nhiệm vụ tương tác.
- Xây dựng đối tượng xác định đối tượng giao diện (lớp)
- Phân tích quy trình làm việc xác định cách một quá trình làm việc được hoàn thành khi một số người (và vai trò) đều tham gia

Sơ đồ Swimlane

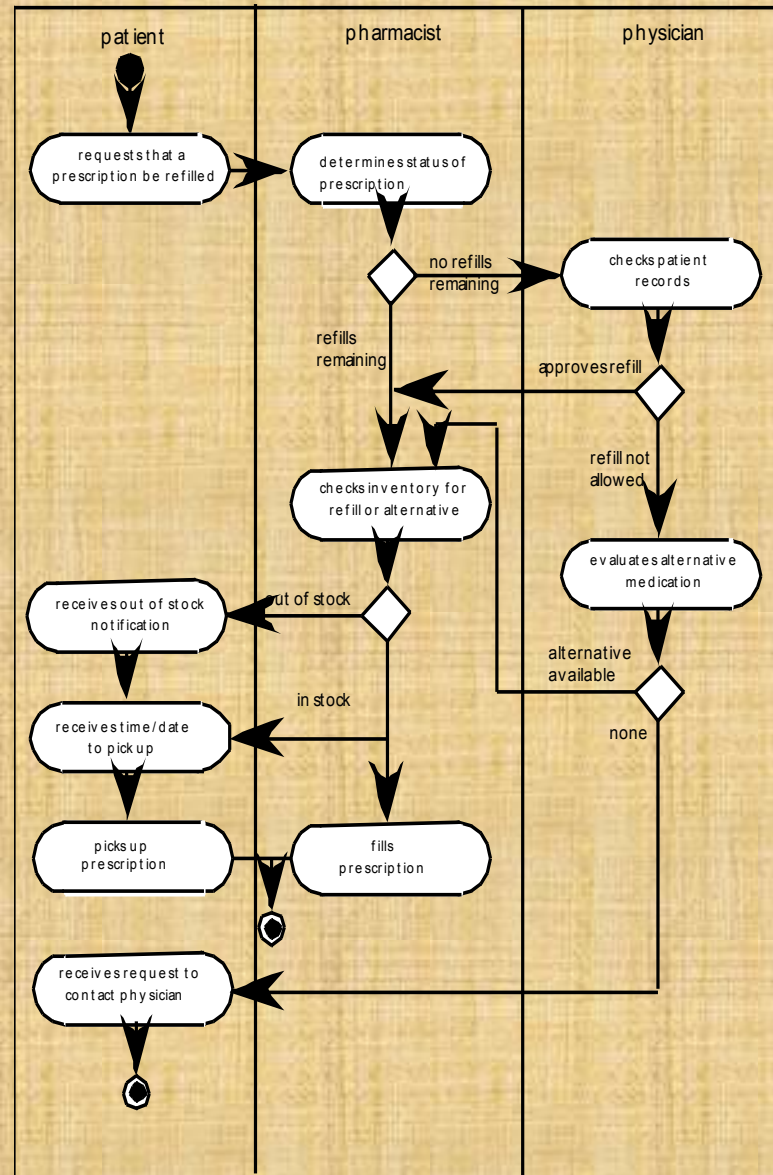


Figure 12.2 Swimlane diagram for prescription refill function

Phân tích nội dung hiển thị

- Liệu các loại dữ liệu khác nhau có được gán cho vị trí địa lý nhất định trên màn hình (ví dụ, hình ảnh luôn luôn xuất hiện ở góc trên bên phải)?
- Liệu người dùng có thể tùy chỉnh vị trí màn hình cho nội dung?
- Liệu các nhận dạng phù hợp có được gán cho tất cả nội dung?
- Nếu một báo cáo lớn được trình bày, nó sẽ được phân chia như thế nào cho dễ hiểu?
- Liệu cơ chế có sẵn sàng để di chuyển trực tiếp tới thông tin tóm tắt cho lượng dữ liệu lớn?
- Liệu các đầu ra đồ họa có được căn chỉnh để vừa vặn với các giới hạn của thiết bị hiển thị được sử dụng?
- Màu sắc sẽ được sử dụng như thế nào để tăng tính dễ hiểu?
- Thông báo lỗi và cảnh báo sẽ được trình bày tới người dùng như thế nào?

Các bước thiết kế giao diện

- Sử dụng thông tin được phát triển trong quá trình phân tích giao diện, **xác định đối tượng giao diện và hành động (hoạt động).**
- **Xác định các sự kiện (các hành động người dùng)** sẽ gây ra sự thay đổi trạng thái của giao diện người dùng. Mô hình hóa hành vi này.
- **Miêu tả mỗi trạng thái giao diện** như thể nó sẽ thực sự tìm đến người dùng cuối.
- **Xác định cách người dùng diễn giải các trạng thái của hệ thống** từ thông tin được cung cấp thông qua giao diện.

Các vấn đề thiết kế

- Thời gian trả lời
- Trợ giúp các tiện nghi
- Xử lý lỗi
- Gắn nhãn menu và câu lệnh
- Khả năng tiếp cận ứng dụng
- Quốc tế hóa



Thiết kế giao diện WebApp

- **Tôi đang ở đâu?** Giao diện nên
 - cung cấp một dấu hiệu rằng các WebApp đó đã được tiếp cận
 - thông báo cho người sử dụng vị trí của họ trong hệ thống phân cấp nội dung.
- **Tôi có thể làm gì bây giờ?** Giao diện phải luôn luôn giúp người dùng hiểu các tùy chọn hiện tại của mình
 - Những chức năng nào đang có sẵn?
 - Những liên kết nào còn hoạt động?
 - Những nội dung nào có liên quan?
- **Tôi đã từng ở đâu, tôi sẽ đi đâu?** Giao diện phải tạo điều kiện cho việc điều hướng
 - Cung cấp một "bản đồ" (được thực hiện một cách dễ hiểu) về nơi mà người sử dụng đã đi qua và những con đường có thể được thực hiện để chuyển đến nơi khác trong phạm vi WebApp.

Giao diện WebApp hiệu quả

- Bruce Tognozzi [TOG01] gợi ý rằng...
 - Giao diện hiệu quả rất trực quan rõ ràng và rộng rãi, mang lại cho người dùng cảm giác kiểm soát. Người dùng nhanh chóng nhận ra sự rộng rãi của các tùy chọn, nắm bắt cách để đạt được mục tiêu và làm công việc của họ.
 - Giao diện hiệu quả khiến người dùng không phải quan tâm đến hoạt động bên trong hệ thống. Công việc được lưu trữ cẩn thận và liên tục, với đầy đủ tùy chọn cho người dùng được hoàn tác bất kì hoạt động nào tại thời điểm bất kì.
 - Ứng dụng và dịch vụ hiệu quả thực hiện công việc lượng công việc tối đa, trong khi yêu cầu lượng thông tin tối thiểu.

Nguyên tắc thiết kế giao diện-I

- **Dự đoán**—Một WebApp nên được thiết kế sao cho nó dự đoán được động thái tiếp theo của việc sử dụng.
- **Truyền thông**—Giao diện nên truyền thông tình trạng của bất kỳ hoạt động khởi xướng bởi người sử dụng
- **Nhất quán**—việc sử dụng điều hướng, menu, biểu tượng, và thẩm mỹ (ví dụ: màu sắc, hình dạng, bố cục)
- **Tự điều khiển**—Giao diện điều khiển nên tạo điều kiện cho chuyển động sử dụng trong suốt WebApp, nhưng nó phải làm như vậy bằng cách thực hiện quy ước đã được thiết lập cho các ứng dụng.
- **Hiệu quả**—Thiết kế của WebApp và giao diện của nó nên tối ưu hóa hiệu quả công việc của người sử dụng, không phải của các kỹ sư thiết kế và xây dựng nó hoặc của các môi trường client-server thực hiện điều đó.

Nguyên tắc thiết kế giao diện-II

- **Tập trung** -Giao diện WebApp (và nội dung nó trình bày) nên tiếp tục tập trung vào các nhiệm vụ người dùng.
- **Luật Fitt** - "Thời gian để có được một mục tiêu là một hàm của khoảng cách đến và kích thước của các mục tiêu."
- **Đối tượng giao diện con người** - một thư viện tái sử dụng rộng lớn của các đối tượng giao diện con người đã được phát triển cho các ứng dụng web.
- **Giảm độ trễ**- WebApp nên sử dụng đa tác vụ bằng cách cho phép người sử dụng tiến hành với công việc nếu các hoạt động đã được hoàn thành.
- **Tính học được**- WebApp nên được thiết kế để giảm thiểu thời gian học tập, và một khi đã học, phải giảm thiểu sự học lại cần thiết khi WebApp được sử dụng lại.

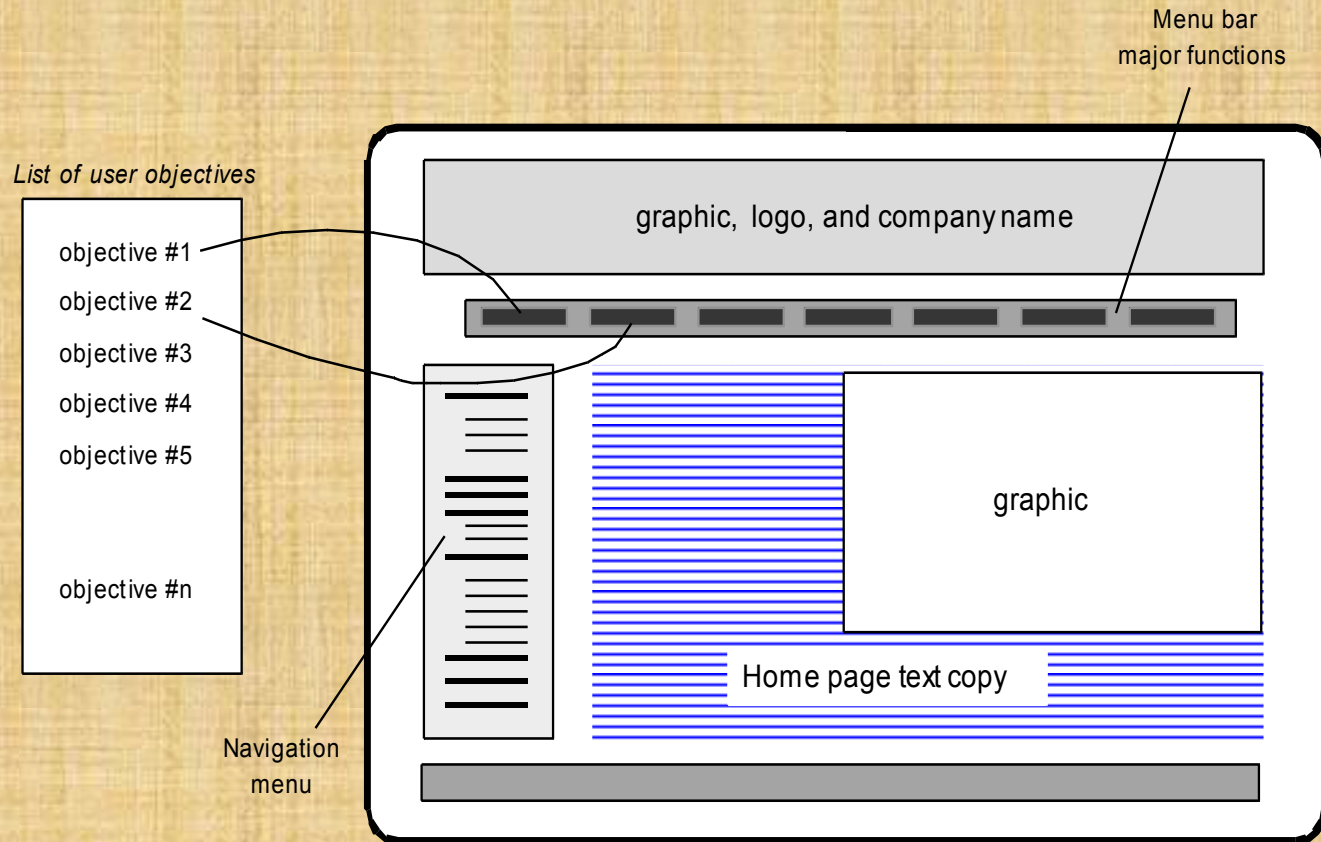
Nguyên tắc thiết kế giao diện-III

- **Duy trì tính toàn vẹn của sản phẩm công việc**—Một sản phẩm công việc (ví dụ: một biểu mẫu hoàn thành bởi người dùng, một danh sách chi tiết người dùng) cần được lưu trữ tự động để không bị mất khi có lỗi xảy ra.
- **Tính đọc được**—Tất cả thông tin trình bày qua giao diện cần phải đọc được bởi tất cả mọi người
- **Theo dõi trạng thái**—Lúc thích hợp, trạng thái của tương tác người dùng nên được theo dõi và lưu trữ để một người dùng có thể đăng xuất và sau đó trở lại đúng giao diện mà người đó đã rời đi.
- **Hiện thị điều hướng**—A giao diện WebApp được thiết kế tốt cung cấp "sự ảo tưởng rằng người sử dụng ở cùng một vị trí, với các công việc mang lại cho họ."

Quy trình thiết kế giao diện-I

- Xem lại thông tin chứa trong các mô hình phân tích và tinh chỉnh khi cần thiết.
- Xây dựng một phác thảo sơ bộ giao diện bố cục WebApp.
- Map các mục tiêu sử dụng vào các hoạt động giao diện cụ thể.
- Xác định một tập hợp các nhiệm vụ người dùng có liên quan đến từng hành động.
- Lên kịch bản hình ảnh màn hình cho mỗi hành động giao diện.
- Tinh chỉnh giao diện bố cục và kịch bản sử dụng đầu vào từ thiết kế thẩm mỹ.

Xác định mục tiêu người dùng



These slides are designed to accompany Software Engineering:
A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

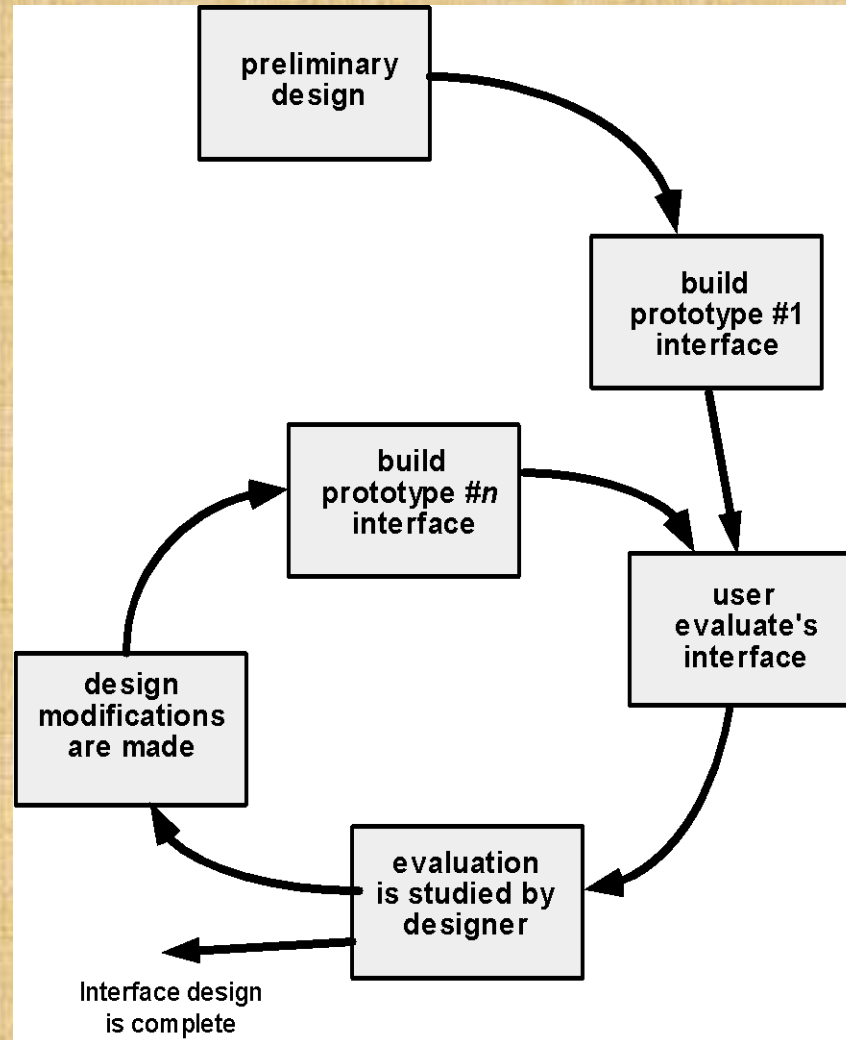
Quy trình thiết kế giao diện-II

- Xác định đối tượng giao diện người dùng được yêu cầu để thực hiện các giao diện. Xây dựng biểu diễn thủ tục của sự tương tác của người dùng với giao diện.
- Xây dựng một biểu diễn hành vi của giao diện.
- Mô tả bố cục giao diện cho mỗi trạng thái.
- Tinh chỉnh và xem xét các mô hình thiết kế giao diện.

Thiết kế thẩm mỹ

- Không ngại khoảng trắng
- Nhấn mạnh nội dung.
- Tổ chức các yếu tố định dạng từ góc trái trên đến phải dưới.
- Nhóm điều hướng, nội dung và chức năng địa lý trong trang.
- Không mở rộng thành phần bất động của bạn với thanh cuộn.
- Xem xét kích thước cửa sổ trình duyệt và độ phân giải khi thiết kế.

Chu kỳ đánh giá thiết kế



Thiết kế phần mềm

- Nội dung
 1. Tổng quan thiết kế phần mềm
 2. Thiết kế kiến trúc phần mềm
 3. Thiết kế chi tiết phần mềm
 4. Thiết kế giao diện người dùng
 - 5. Thiết kế mẫu**
 6. Thiết kế giao diện cho ứng dụng WebApp

Khuôn mẫu thiết kế (Design Patterns)

- Mỗi chúng ta đều đã trải qua vấn đề thiết kế và từng nghĩ:
Liệu đã có ai phát triển một lời giải cho vấn đề này chưa?
 - Điều gì sẽ xảy ra nếu đã có một cách tiêu chuẩn để mô tả một vấn đề (để bạn có thể nhìn nhận nó), và một phương pháp có tổ chức để trình bày lời giải cho vấn đề đó?
- **Khuôn mẫu thiết kế** là một phương pháp có hệ thống để mô tả các vấn đề và giải pháp, cho phép các cộng đồng công nghệ phần mềm có thể nắm bắt kiến thức thiết kế theo cách có thể tái sử dụng

Khuôn mẫu thiết kế (Design Patterns)

- *Mỗi khuôn mẫu mô tả một vấn đề xảy ra hết lần này đến lần khác trong môi trường của chúng ta và sau đó mô tả cốt lõi của giải pháp cho vấn đề đó theo một cách mà bạn có thể sử dụng nó hàng triệu lần mà không bao giờ phải làm lại một việc lần thứ hai.*

Christopher Alexander, 1977

- “một quy tắc ba phần diễn tả một mối quan hệ giữa một ngữ cảnh, một vấn đề, và một giải pháp.”

Khái niệm cơ bản

- *Ngữ cảnh* cho phép người đọc hiểu được môi trường trong đó các vấn đề phát sinh và giải pháp nào có thể thích hợp trong môi trường đó.
- Một tập hợp các yêu cầu, bao gồm cả những hạn chế và khó khăn, hoạt động như một hệ thống các lực lượng có ảnh hưởng tới cách:
 - các vấn đề có thể được diễn giải trong ngữ cảnh của nó và
 - giải pháp có thể được áp dụng hiệu quả như thế nào

Khuôn mẫu hiệu quả

- Coplien [Cop05] nêu đặc điểm cho một khuôn mẫu thiết kế hiệu quả theo các cách sau đây:
 - **Nó giải quyết một vấn đề:** Khuôn mẫu đưa ra giải pháp, không chỉ là quy tắc hay chiến lược trừu tượng.
 - **Nó là một khái niệm đã được chứng minh:** Khuôn mẫu đưa ra giải pháp dựa trên dữ liệu theo dõi, không phải là lý thuyết hay suy đoán.
 - **Giải pháp không hiển nhiên:** Rất nhiều kỹ thuật giải quyết vấn đề (chẳng hạn như mô hình hoặc các phương pháp thiết kế phần mềm) cố gắng đạt được giải pháp từ quy tắc đầu tiên. Khuôn mẫu tốt nhất tạo ra một giải pháp cho một vấn đề một cách gián tiếp - một cách tiếp cận cần thiết cho những vấn đề khó nhất của thiết kế.
 - **Nó mô tả một mối quan hệ:** Khuôn mẫu không chỉ mô tả mô-đun mà mô tả kiến trúc và cơ chế sâu hơn của hệ thống.
 - **Khuôn mẫu có một thành phần con người quan trọng (giảm thiểu ảnh hưởng của con người).** Tất cả phần mềm cung cấp cho con người sự thoải mái hoặc chất lượng cuộc sống; khuôn mẫu tốt nhất hấp dẫn rõ ràng về cả thẩm mỹ và tiện ích.

Khuôn mẫu khả sinh (Generative patterns)

- **Khuôn mẫu khả sinh** mô tả một khía cạnh quan trọng và có thể lặp lại của một hệ thống và sau đó cung cấp cho chúng ta một cách xây dựng khía cạnh đó trong một hệ thống của các nguồn lực riêng biệt đối với từng ngữ cảnh.
- Một tập hợp các khuôn mẫu thiết kế khả sinh có thể được dùng để sinh một ứng dụng hoặc một hệ thống dựa trên máy tính có kiến trúc cho phép nó thích nghi với sự thay đổi.

Các loại khuôn mẫu

- **Architectural patterns** Mẫu kiến trúc mô tả các vấn đề thiết kế trên diện rộng được giải quyết bằng cách tiếp cận cấu trúc.
- **Data patterns** Mẫu dữ liệu mô tả vấn đề dữ liệu và các giải pháp mô hình dữ liệu có thể được dùng để giải quyết vấn đề trên
- **Component patterns** Mẫu thành phần (còn gọi là các mẫu thiết kế) nhằm đến các vấn đề liên quan với việc phát triển các hệ thống con và các thành phần, cách thức chúng giao tiếp với nhau, và vị trí của chúng trong một kiến trúc lớn hơn
- **Interface design patterns.** Mẫu thiết kế giao diện mô tả các vấn đề giao diện người dùng thông thường và giải pháp bao gồm các đặc trưng cụ thể của người dùng cuối.
- **WebApp patterns.** Mẫu WebApp giải quyết tập hợp vấn đề có thể gặp phải khi xây dựng Ứng dụng web và thường kết hợp nhiều mô hình khác cập đến ở trên.

Các loại khuôn mẫu

- **Khuôn mẫu sáng tạo (Creational patterns)** tập trung vào việc khởi tạo, sáng tác và biểu diễn của các đối tượng, ví dụ:
 - [Abstract factory pattern](#)
 - [Factory method pattern](#)
- **Khuôn mẫu cấu trúc** tập trung vào các vấn đề và các giải pháp liên quan đến cách các lớp và các đối tượng được tổ chức và tích hợp để xây dựng một cấu trúc lớn hơn, ví dụ như:
 - [Adapter pattern](#)
 - [Aggregate pattern](#)
- **Khuôn mẫu hành vi** giải quyết các vấn đề liên quan đến việc phân công trách nhiệm giữa các đối tượng và cách thức mà giao tiếp được thực hiện giữa các đối tượng, ví dụ như:
 - [Chain of responsibility pattern](#):
 - [Command pattern](#):

Khung (frameworks)

- Bản thân khuôn mẫu có thể không đủ để phát triển một thiết kế đầy đủ
 - Trong một số trường hợp, cần thiết phải cung cấp một cơ sở hạ tầng thực hiện cụ thể, được gọi là một khung (**framework**) cho công việc thiết kế.
 - Có nghĩa là, bạn có thể chọn một “một kiến trúc nhỏ có thể tái sử dụng cung cấp cấu trúc chung và hành vi cho một họ các sự trừu tượng hóa phần mềm, cùng với một ngữ cảnh... xác định sự cộng tác và sử dụng của chúng trong một miền nhất định” [Amb98]
- **Một framework không phải là một khuôn mẫu kiến trúc**, mà là một bộ khung với một tập hợp các “**plug points**” (còn được gọi là hooks hay slots) cho phép nó thích ứng với một miền vấn đề xác định.
 - Những “plug points” cho phép bạn tích hợp các lớp vấn đề cụ thể hoặc chức năng trong các bộ khung.

Mô tả một khuôn mẫu

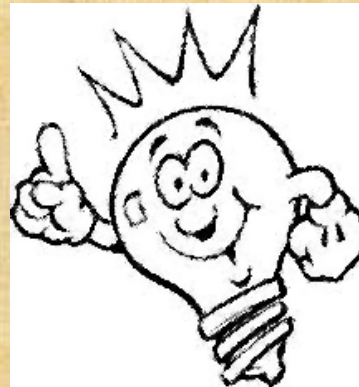
- **Tên khuôn mẫu**—mô tả bản chất của khuôn mẫu với một cái tên ngắn nhưng biểu cảm.
- **Vấn đề**—Mô tả vấn đề mà khuôn mẫu giải quyết.
- **Động lực**—cung cấp một ví dụ cho vấn đề
- **Ngữ cảnh**—Mô tả môi trường mà vấn đề phát sinh, bao gồm miền ứng dụng
- **Nguồn lực**—liệt kê các hệ thống của các lực lượng có ảnh hưởng đến cách thức mà các vấn đề được giải quyết; bao gồm một cuộc thảo luận về các giới hạn và ràng buộc phải được xem xét
- **Giải pháp**—cung cấp một mô tả chi tiết về các giải pháp đề xuất cho vấn đề
- **Mục đích**—mô tả các khuôn mẫu và những gì nó làm
- **Sự cộng tác**—mô tả cách các khuôn mẫu khác đóng góp vào giải pháp
- **Hệ quả**—mô tả những sự đánh đổi tiềm năng phải được xem xét khi khuôn mẫu được thực hiện và những hệ quả của việc sử dụng khuôn mẫu.
- **Sự triển khai**—xác định các vấn đề đặc biệt cần được xem xét khi thực hiện khuôn mẫu
- **Ứng dụng đã biết**—cung cấp các ví dụ về cách sử dụng của các khuôn mẫu thiết kế trong ứng dụng thực tế.
- **Khuôn mẫu liên quan**—tham khảo chéo liên quan đến các khuôn mẫu thiết kế

Ngôn ngữ khuôn mẫu

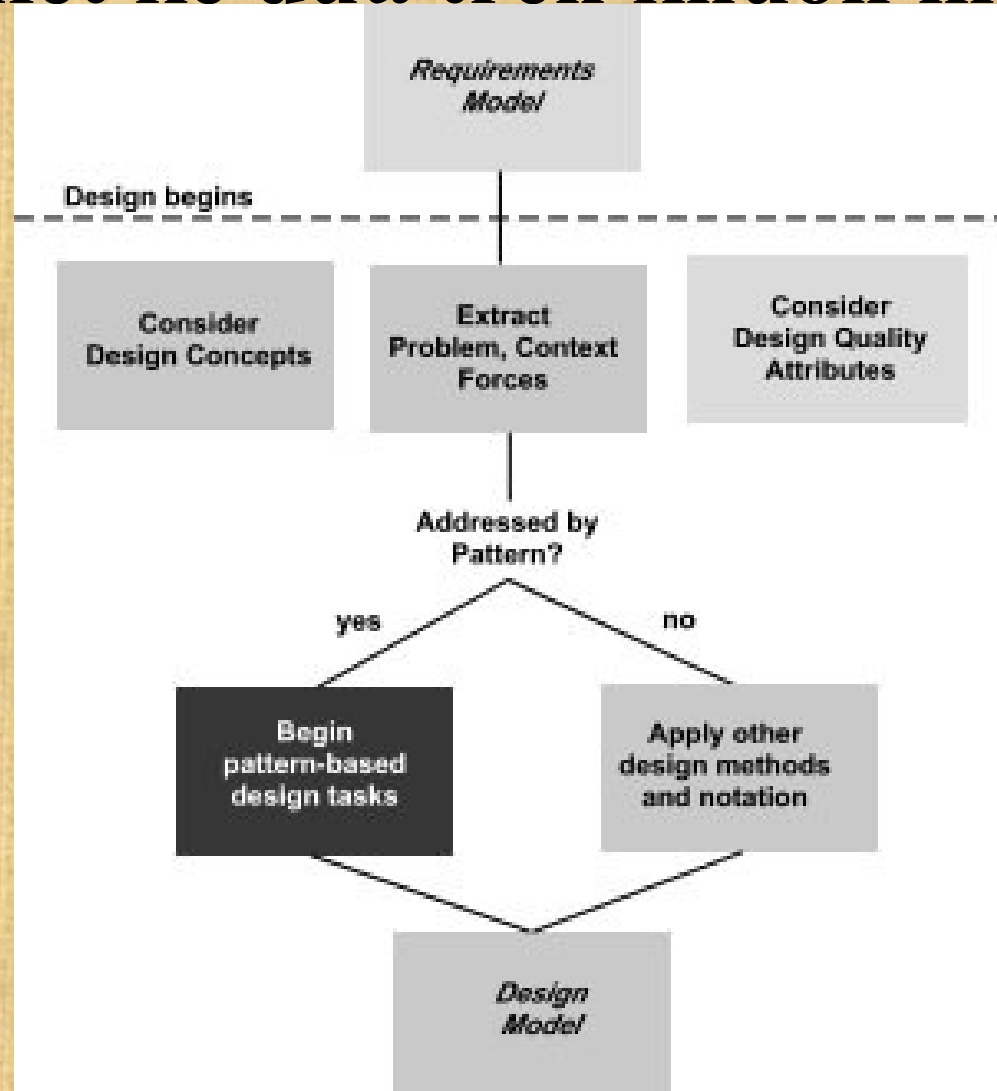
- Một ngôn ngữ khuôn mẫu bao gồm tập hợp các khuôn mẫu:
 - được mô tả sử dụng một bản mẫu tiêu chuẩn (phần 12.1.3) và
 - quan hệ với nhau để cho thấy cách các khuôn mẫu cộng tác để giải quyết vấn đề trên một miền ứng dụng.
- một ngôn ngữ khuôn mẫu tương tự như một sách hướng dẫn siêu văn bản để giải quyết vấn đề trong một miền ứng dụng cụ thể.
 - Các miền vấn đề được xem xét đầu tiên được mô tả theo thứ bậc, bắt đầu với các vấn đề thiết kế rộng liên quan đến miền và sau đó tinh chỉnh từng vấn đề rộng vào mức độ trừu tượng thấp hơn.

Thiết kế dựa trên khuôn mẫu

- Một nhà thiết kế phần mềm bắt đầu với một mô hình yêu cầu (hoặc là rõ ràng hay ngụ ý) trình bày một thể hiện trừu tượng của hệ thống.
- Các mô hình yêu cầu mô tả các tập vấn đề, thiết lập ngữ cảnh, và xác định các hệ thống của các nguồn lực.
- Sau đó ...



Thiết kế dựa trên khuôn mẫu



Suy nghĩ trong khuôn mẫu

- Shalloway và Trott [Sha05] đề xuất các phương pháp sau đây cho phép một nhà thiết kế suy nghĩ trong mô hình:
 1. Hãy chắc chắn rằng bạn hiểu được bức tranh lớn - ngữ cảnh trong đó phần mềm được xây dựng. Các yêu cầu mô hình phải truyền thông điệp này cho bạn.
 2. Xem xét các bức tranh lớn, trích xuất các khuôn mẫu có mặt ở mức độ trừu tượng đó.
 3. Bắt đầu thiết kế của bạn với khuôn mẫu "bức tranh lớn" mà thiết lập một ngữ cảnh hoặc bộ khung cho công việc thiết kế sau này.
 4. "Làm việc hướng nội từ ngữ cảnh" [Sha05] tìm kiếm các khuôn mẫu ở các mức trừu tượng thấp hơn góp phần vào các giải pháp thiết kế.
 5. Lặp lại các bước 1-4 cho đến khi thiết kế hoàn chỉnh được tạo ra.
 6. Hoàn thiện thiết kế bằng cách thích ứng từng mô hình vào các chi tiết cụ thể của phần mềm bạn đang cố gắng xây dựng.

Nhiệm vụ thiết kế-I

- Kiểm tra các mô hình yêu cầu và phát triển một hệ thống phân cấp vấn đề.
- Xác định nếu một ngôn ngữ khuôn mẫu đáng tin cậy đã được phát triển cho miền vấn đề.
- Bắt đầu với một vấn đề lớn, xác định một hoặc nhiều khuôn mẫu kiến trúc có sẵn cho nó.
- Sử dụng sự cộng tác được cung cấp cho các khuôn mẫu kiến trúc, kiểm tra hệ thống phụ hoặc các vấn đề mức thành phần và tìm kiếm các khuôn mẫu thích hợp để giải quyết chúng.
- Lặp lại các bước từ 2 đến 5 cho đến khi tất cả các vấn đề lớn đã được giải quyết.

Nhiệm vụ thiết kế-II

- Nếu vấn đề thiết kế giao diện người dùng đã được phân lập (điều này là hầu như luôn luôn như vậy), hãy tìm kiếm các giao diện người dùng trong kho mẫu thiết kế để tìm ra khuôn mẫu phù hợp.
- Bất kể mức độ trừu tượng của nó, nếu một ngôn ngữ khuôn mẫu và / hoặc kho khuôn mẫu hoặc khuôn mẫu riêng cho thấy sự hứa hẹn, hãy so sánh các vấn đề cần giải quyết đối với các khuôn mẫu hiện có.
- Hãy chắc chắn để tinh chỉnh các thiết kế như là nó có nguồn gốc từ các khuôn mẫu sử dụng các tiêu chí chất lượng thiết kế như một hướng dẫn.

Bảng sắp xếp khuôn mẫu

	Database	Application	Implementation	Infrastructure
Data/Content				
<i>Problem statement ...</i>	PatternName (a)		PatternName (a)	
<i>Problem statement ...</i>		PatternName (a)		PatternName (a)
<i>Problem statement ...</i>	PatternName (a)			PatternName (a)
Architecture				
<i>Problem statement ...</i>		PatternName (a)		
<i>Problem statement ...</i>		PatternName (a)		PatternName (a)
<i>Problem statement ...</i>				
Component-level				
<i>Problem statement ...</i>		PatternName (a)	PatternName (a)	
<i>Problem statement ...</i>				PatternName (a)
<i>Problem statement ...</i>		PatternName (a)	PatternName (a)	
User interface				
<i>Problem statement ...</i>		PatternName (a)	PatternName (a)	
<i>Problem statement ...</i>		PatternName (a)	PatternName (a)	
<i>Problem statement ...</i>		PatternName (a)	PatternName (a)	

Lỗi thiết kế thông thường

- Không đủ thời gian được sử dụng để hiểu được vấn đề cơ bản, ngữ cảnh và các nguồn lực của mình, và như một hệ quả, bạn chọn một khuôn mẫu có vẻ đúng, nhưng không phù hợp với giải pháp được yêu cầu.
- Một khi chọn sai khuôn mẫu, bạn không chịu nhìn ra vấn đề và sử dụng khuôn mẫu gượng ép.
- Trong các trường hợp khác, các vấn đề có nguồn lực không được xem xét bởi các khuôn mẫu bạn đã chọn, dẫn đến kết quả nghèo nàn.
- Đôi khi một khuôn mẫu được áp dụng quá thô thiển và sự thích nghi cần thiết cho không gian vấn đề của bạn không được thực hiện.

Khuôn mẫu kiến trúc

- Ví dụ: mỗi nhà (và mọi phong cách kiến trúc đối với nhà ở) sử dụng một khuôn mẫu nhà bếp.
- Khuôn mẫu nhà bếp và các khuôn mẫu mà nó cộng tác giải quyết các vấn đề liên quan tới việc lưu trữ và chuẩn bị thức ăn, các công cụ cần thiết để thực hiện những nhiệm vụ, và các quy tắc cho vị trí của các công cụ liên quan đến công việc trong phòng. Ngoài ra, các mô hình có thể giải quyết các vấn đề liên quan đến mặt bàn, chiếu sáng, công tắc tường, sàn nhà...
- Rõ ràng, có nhiều hơn một thiết kế cho một nhà bếp, thường quyết định bởi bối cảnh và hệ thống của lực lượng. Tuy nhiên, mỗi thiết kế có thể được hình thành trong bối cảnh của "giải pháp" được đề xuất bởi các khuôn mẫu nhà bếp.

Kho khuôn mẫu

- Có rất nhiều nguồn cho các khuôn mẫu thiết kế có sẵn trên web. Một số khuôn mẫu có thể được lấy từ các ngôn ngữ khuôn mẫu xuất bản riêng, trong khi những cái khác đang có sẵn như là một phần của một công cụ hoặc kho khuôn mẫu.
- Một danh sách các kho khuôn mẫu được thể hiện trong mục 12.3

Khuôn mẫu mức thành phần

- Các khuôn mẫu thiết kế mức thành phần cung cấp một giải pháp đã được chứng minh để giải quyết một hoặc nhiều vấn đề phụ phát sinh từ các mô hình yêu cầu. Trong nhiều trường hợp, các khuôn mẫu thiết kế của loại hình này tập trung vào một số yếu tố chức năng của một hệ thống.
- Ví dụ, ứng dụng SafeHomeAssured.com phải giải quyết các vấn đề thiết kế phụ sau đây: Làm thế nào chúng ta có thể có được thông số kỹ thuật sản phẩm và các thông tin liên quan cho bất kỳ thiết bị SafeHome nào?

Khuôn mẫu mức thành phần

- Sau khi đề ra các vấn đề phụ cần giải quyết, xem xét ngữ cảnh và hệ thống các lực lượng có ảnh hưởng đến các giải pháp.
- Kiểm tra các trường hợp sử dụng các yêu cầu mô hình thích hợp, các đặc điểm kỹ thuật cho một thiết bị SafeHome (ví dụ, một bộ cảm biến an ninh hoặc camera) được sử dụng cho mục đích thông tin của người tiêu dùng.
 - Tuy nhiên, các thông tin khác có liên quan đến các đặc điểm kỹ thuật (ví dụ, giá cả) có thể được sử dụng khi chức năng thương mại điện tử được chọn.
- Các giải pháp cho vấn đề phụ bao gồm một sự tìm kiếm. Kể từ khi tìm kiếm trở thành vấn đề rất phổ biến, không có gì ngạc nhiên khi có rất nhiều các khuôn mẫu liên quan đến tìm kiếm.
- Xem Phần 12.4

Khuôn mẫu giao diện người dùng (UI Patterns)

- **Giao diện người dùng tổng thể (Whole UI).** Cung cấp hướng dẫn thiết kế cho các cấu trúc cấp cao nhất và điều hướng trong suốt toàn bộ giao diện.
- **Bố cục trang.** Giải quyết các tổ chức chung của trang (cho các trang web) hay hiển thị màn hình riêng biệt (cho các ứng dụng tương tác)
- **Biểu mẫu và đầu vào.** Xem xét nhiều kỹ thuật thiết kế cho việc hoàn thành đầu vào dạng biểu mẫu.
- **Bảng.** Cung cấp hướng dẫn thiết kế cho việc tạo và xử lý dữ liệu dạng bảng dưới mọi dạng.
- **Xử lý dữ liệu trực tiếp.** Address data editing, modification, and transformation.
- **Điều hướng.** Hỗ trợ người dùng định hướng xuyên suốt menu phân cấp, các trang web và màn hình hiển thị tương tác.
- **Tìm kiếm.** Cho phép tính năng tìm kiếm nội dung cụ thể thông qua các thông tin được duy trì trong một trang Web hoặc được lưu trữ trong cơ sở dữ liệu có thể truy cập thông qua một ứng dụng tương tác..
- **Các phần tử trang.** Thực hiện các phần tử cụ thể của một trang web hoặc màn hình hiển thị.
- **Thương mại điện tử.** Tùy vào các trang web, các mô hình triển khai các yếu tố định kỳ của các ứng dụng thương mại điện tử.

Khuôn mẫu WebApp

- **Khuôn mẫu kiến trúc thông tin** liên quan đến cấu trúc tổng thể của không gian thông tin, và những cách mà người dùng sẽ tương tác với thông tin.
- **Khuôn mẫu điều hướng** xác định cấu trúc liên kết chuyển hướng, chẳng hạn như hệ thống phân cấp, vòng, tours,...
- **Khuôn mẫu tương tác** đóng góp vào việc thiết kế giao diện người dùng. Các khuôn mẫu loại này chỉ ra cách giao diện người dùng thông báo về hệ quả của một hành động cụ thể; làm thế nào một người dùng mở rộng nội dung dựa trên việc sử dụng ngữ cảnh và mong muốn của người dùng; làm thế nào để mô tả tốt nhất đích đến đằng sau một liên kết; làm thế nào để thông báo cho người dùng về tình trạng của một tương tác đang thực hiện, và các vấn đề liên quan đến giao diện.
- **Khuôn mẫu trình diễn** hỗ trợ trong việc trình bày các nội dung cho người sử dụng thông qua giao diện. Các khuôn mẫu loại này giải quyết vấn đề tổ chức các chức năng điều khiển giao diện người dùng cho khả năng sử dụng tốt hơn; vấn đề hiển thị các mối quan hệ giữa một hành động giao diện và các đối tượng nội dung nó ảnh hưởng, và vấn đề thiết lập hệ thống phân cấp nội dung hiệu quả.
- **Khuôn mẫu chức năng** xác định các quy trình công việc, hành vi, xử lý, truyền thông, và các yếu tố khác trong một thuật toán WebApp.

Thiết kế mức độ chi tiết

- Khi một vấn đề liên quan đến vấn đề "bức tranh lớn", cố gắng phát triển các giải pháp (và sử dụng các khuôn mẫu có liên quan) tập trung vào bức tranh lớn.
- Ngược lại, khi sự tập trung là rất hẹp (ví dụ, duy nhất chọn một mục từ một tập hợp nhỏ của năm mặt hàng hoặc ít hơn), các giải pháp (và các khuôn mẫu tương ứng) có mục tiêu khá hẹp.
- Xét về mức độ chi tiết, khuôn mẫu có thể được mô tả theo các mức sau:

Thiết kế mức độ chi tiết

- **Khuôn mẫu kiến trúc.** Mức độ trừu tượng này thường sẽ liên quan đến các khuôn mẫu xác định cấu trúc tổng thể của WebApp, chỉ ra các mối quan hệ giữa các thành phần khác nhau, và xác định các quy tắc để xác định mối quan hệ giữa các yếu tố (các trang, gói, thành phần, hệ thống con) của kiến trúc.
- **Khuôn mẫu thiết kế.** Những khuôn mẫu này chỉ ra một yếu tố cụ thể của thiết kế như một tập hợp của các thành phần để giải quyết một số vấn đề thiết kế, mối quan hệ giữa các yếu tố trên một trang, hoặc các cơ chế ảnh hưởng đến giao tiếp giữa các thành phần. Một ví dụ có thể là khuôn mẫu Broadsheet cho cách bố trí của một trang chủ WebApp.
- **Khuôn mẫu thành phần.** Mức độ trừu tượng này liên quan đến các yếu tố quy mô nhỏ lẻ của WebApp. Các ví dụ bao gồm các yếu tố tương tác cá nhân, các mục chuyên hướng, hoặc các yếu tố chức năng.

Thiết kế phần mềm

- Nội dung
 1. Tổng quan thiết kế phần mềm
 2. Thiết kế kiến trúc phần mềm
 3. Thiết kế chi tiết phần mềm
 4. Thiết kế giao diện người dùng
 5. Thiết kế mẫu
 6. **Thiết kế giao diện cho ứng dụng WebApp**

Thiết kế ứng dụng WebApp

“Có hai phương pháp thiết kế cơ bản: ý tưởng nghệ thuật để thể hiện bản thân và ý tưởng kỹ thuật để giải quyết vấn đề cho khách hàng.”

Jakob Nielsen

- Khi nào cần nhấn mạnh thiết kế WebApp
 - Khi các nội dung và chức năng rất phức tạp
 - Khi kích thước của WebApp bao gồm hàng trăm đối tượng, hàm và lớp phân tích
 - Khi sự thành công của WebApp sẽ tác động trực tiếp tới sự thành công của doanh nghiệp

Thiết kế và chất lượng WebApp

- An ninh
 - Ngăn ngừa tấn công từ bên ngoài
 - Loại trừ truy cập trái phép
 - Đảm bảo sự riêng tư của người dùng/khách hàng
- Khả dụng
 - Ước lượng được tỉ lệ thời gian WebApp khả dụng
- Khả năng mở rộng
 - WebApp có thể xử lý sự thay đổi đáng kể trong khối lượng người dùng/giao dịch
- Thời gian ra thị trường

Chất lượng với người dùng

- **Thời gian**
 - Trang Web thay đổi bao nhiêu sau khi nâng cấp
 - Các phần thay đổi có được highlight?
- **Cấu trúc**
 - Các bộ phận của website kết nối với nhau như thế nào?
 - Mọi liên kết trong/ngoài trang web có hoạt động?
 - Tất cả các hình ảnh có được hiển thị?
 - Có phần nào của website không được kết nối?
- **Nội dung**
 - Nội dung các trang quan trọng có khớp với thông tin tại đó?
 - Các từ khóa có tồn tại liên tục với các trang thay đổi nhanh?
 - Các trang quan trọng có duy trì được chất lượng nội dung?
 - Có thể tự động tạo các trang HTML?

Chất lượng với người dùng

- **Độ chính xác, nhất quán**
 - Bản sao của ngày hôm qua có giống hôm nay?
 - Dữ liệu được trình bày chính xác, đầy đủ?
- **Thời gian đáp ứng và độ trễ**
 - Server web có đáp ứng yêu cầu trình duyệt trong thời gian nhất định?
 - Trong thương mại điện tử, làm sao đảm bảo thời gian đáp ứng sau khi gửi một lệnh
 - Có phần nào của web quá chậm, khiến người dùng không tiếp tục sử dụng?
- **Hiệu năng**
 - Kết nối có đủ nhanh?
 - Hiệu suất thay đổi như thế nào theo thời gian trong ngày?
 - Hiệu suất có đáp ứng được các ứng dụng thương mại điện tử

Mục tiêu thiết kế

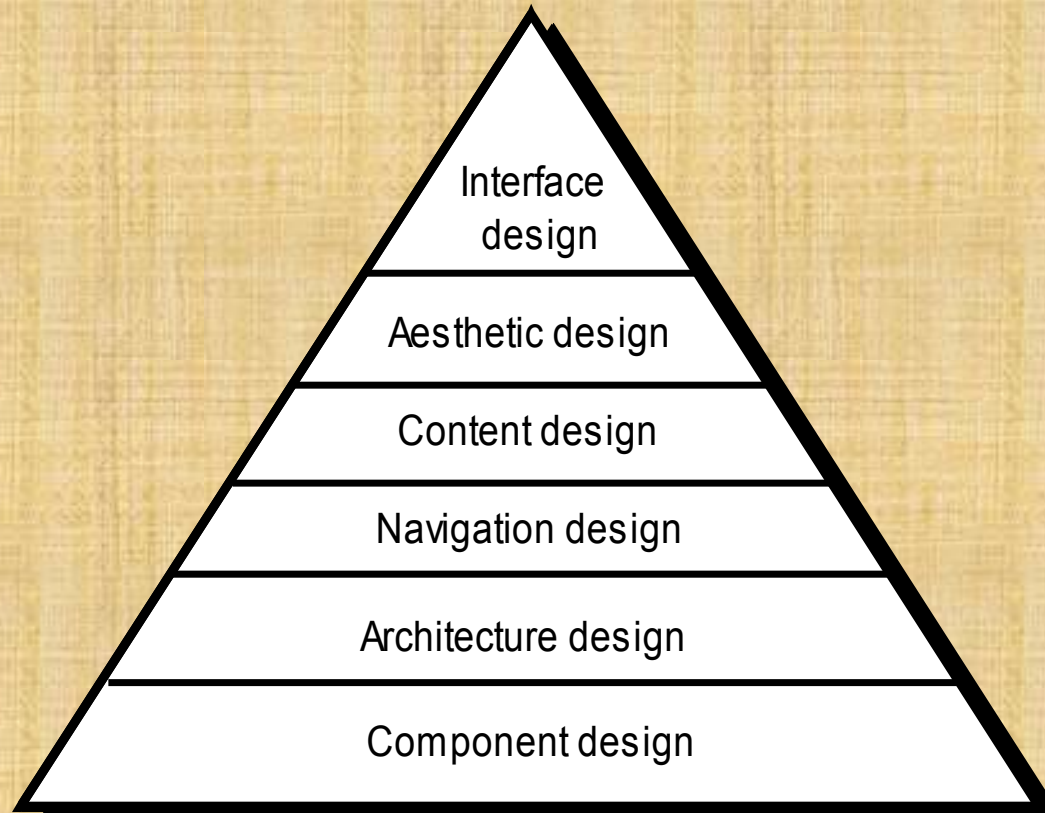
- Nhất quán
 - Nội dung cần được xây dựng nhất quán
 - Thiết kế đồ họa nên có một cách nhìn nhất quán trên các phần của ứng dụng web
 - Thiết kế kiến trúc cần đưa ra mẫu dẫn đến một cấu trúc siêu phương tiện phù hợp
 - Thiết kế giao diện nên xác định các tương tác, chuyển hướng và hiển thị phù hợp
 - Cơ chế điều hướng nên sử dụng nhất quán trên tất cả các yếu tố của ứng dụng

Mục tiêu thiết kế

- **Identity**
 - Thiết lập một “Identity-bản sắc” phù hợp với mục đích nghiệp vụ
- **Robustness**
 - Người dùng kỳ vọng nội dung và chức năng mạnh mẽ có liên quan đến nhu cầu của người sử dụng
- **Navigability**
 - Được thiết kế một cách trực quan và dễ dự đoán
- **Visual appeal**
 - Nhìn và cảm nhận về nội dung, bố trí giao diện, phối hợp màu sắc, sự cân bằng của văn bản, đồ họa và các phương tiện truyền thông khác, cơ chế điều hướng đến người dùng cuối cùng
- **Compatibility**
 - Với tất cả các môi trường và cấu hình phù hợp

Tháp thiết kế

user



technology

Thiết kế giao diện

- **Giao diện cần**
 - Cung cấp dấu hiệu truy cập
 - Thông báo cho người sử dụng vị trí của họ trong nội dung phân cấp
- **Giao diện luôn giúp người dùng hiểu tùy chọn hiện tại của họ**
 - Những chức năng nào khả dụng
 - Những liên kết nào còn sử dụng được
 - Những nội dung nào có liên quan
- **Giao diện cần tạo điều kiện chuyển hướng**
 - Cung cấp một “bản đồ” dễ hiểu để người dùng có thể chuyển hướng trong ứng dụng

Giao diện WebApp hiệu quả

- Bruce Tognozzi [TOG01] đề nghị
 - **Giao diện hiệu quả cần trực quan**, người sử dụng có thể nhanh chóng xem các lựa chọn, nắm bắt xem làm thế nào để đạt mục tiêu và làm việc
 - **Người sử dụng không liên quan đến hoạt động bên trong của hệ thống**. Công việc thực hiện xuyên suốt và thường xuyên được lưu, với các tùy chọn đầy đủ để có thể hoàn tác.
 - **Thực hiện công việc một cách tối đa**, trong khi nhận thông tin ít nhất từ người dùng

Nguyên lý thiết kế giao diện I

- **Tiên đoán** — WebApp nên thiết kế để dự kiến được hành động tiếp theo của người dùng
- **Truyền thông** — Giao diện cần truyền thông các trạng thái của các hoạt động của người dùng
- **Nhất quán** — Sử dụng công cụ điều hướng, menu, biểu tượng
- **Kiểm soát quyền tự trị** — Giao diện tạo điều kiện cho người dùng di chuyển trong ứng dụng, theo các quy ước điều hướng được quy định trong ứng dụng
- **Hiệu quả** — Thiết kế của WebApp và giao diện nên được tối ưu hóa hiệu quả làm việc của người dùng, chứ không nên tối ưu hóa theo công việc của kỹ sư hay môi trường thực thi

Nguyên lý thiết kế giao diện II

- **Tập trung** — Giao diện và nội dung nên tập trung vào các tác vụ người dùng đang sử dụng
- **Luật Fitt** — “Thời gian đạt được mục tiêu là hàm của khoảng cách và kích thước mục tiêu”
- **Đối tượng giao diện người dùng** — Một thư viện lớn của các đối tượng giao diện tái sử dụng, phát triển cho WebApp
- **Giảm độ trễ** — Ứng dụng thực hiện đa nhiệm tuy nhiên vẫn cho phép người dùng thực hiện công việc liên tục
- **Thời gian học** — Giao diện nên thiết kế giảm thiểu thời gian học, và sau khi học để giảm thiểu thời gian học lại khi được truy cập lại.

Nguyên lý thiết kế giao diện III

- **Duy trì kết quả công việc** — Công việc của người dùng phải được lưu lại tự động, tránh mất mát nếu có lỗi xảy ra
- **Dễ đọc** — Mọi thông tin được trình bày phải dễ đọc với mọi người
- **Theo dõi trạng thái** — Khi thích hợp, các trạng thái tương tác của người dùng nên được theo dõi và ghi lại, để người dùng có thể đăng xuất và sau đó quay lại trạng thái làm việc cuối của họ.
- **Điều hướng** — Một giao diện tốt cung cấp các điều hướng rõ ràng, bố trí hợp lý

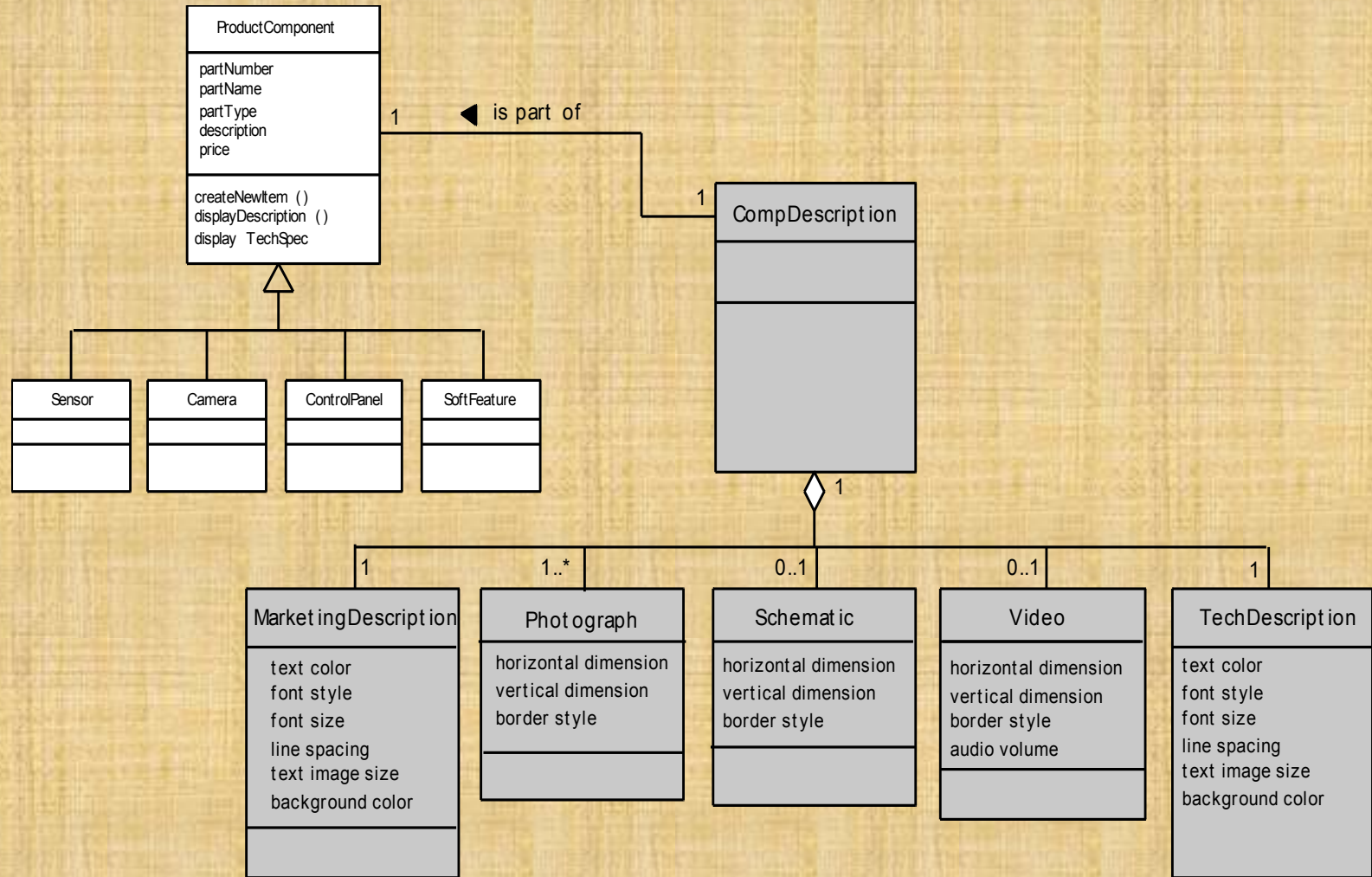
Tính thẩm mỹ

- Không ngại các khoảng trắng
- Nhấn mạnh nội dung
- Tổ chức các thành phần từ góc trên bên trái xuống góc dưới bên phải
- Nhóm các điều hướng, nội dung và các chức năng di chuyển trong trang
- Không mở rộng khung trang với thanh cuộn
- Xem xét độ phân giải và kích thước cửa sổ khi thiết kế

Nội dung

- Phát triển một thể hiện cho đối tượng nội dung
 - Với WebApp, đối tượng nội dung liên kết chặt chẽ với đối tượng dữ liệu hơn phần mềm thông thường
- Biểu diễn các cơ chế cần thiết để tạo các mối quan hệ giữa các đối tượng
 - Tương tự mối quan hệ giữa lớp phân tích và thiết kế thành phần trong chapter 11
- Một đối tượng nội dung gồm thông tin dành riêng cho nội dung và các thuộc tính hiển thị cụ thể được quy định như một phần của thiết kế

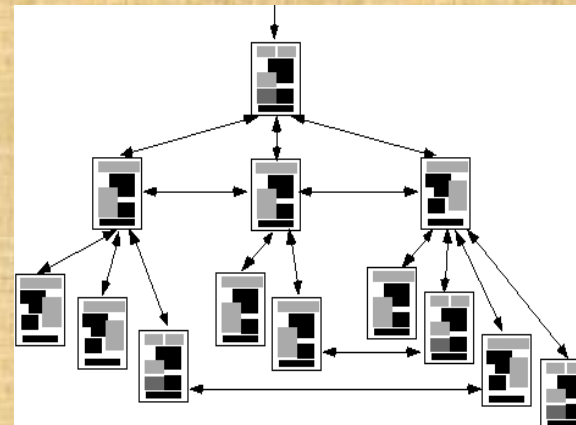
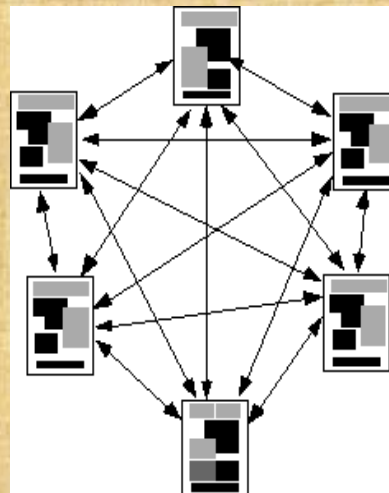
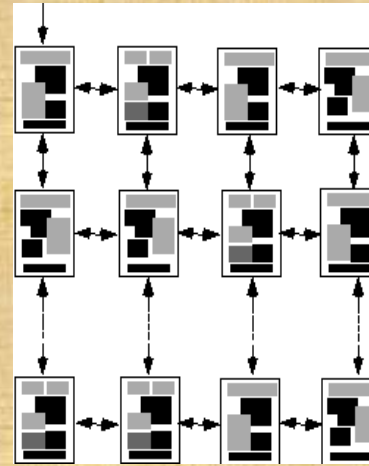
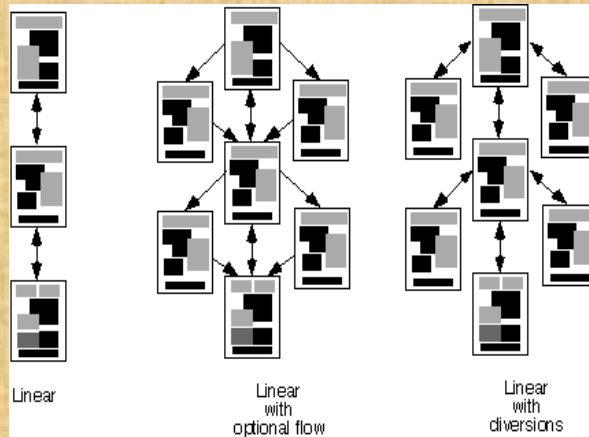
Thiết kế đối tượng nội dung



Thiết kế kiến trúc

- **Kiến trúc nội dung** tập trung vào cách các đối tượng nội dung được tổ chức có cấu trúc để chuyển hướng/hiển thị
 - Khái niệm kiến trúc thông tin cũng được sử dụng cho các cấu trúc giúp tổ chức, đánh chỉ mục, chuyển hướng, ... tốt hơn
- **Kiến trúc WebApp** địa chỉ hóa theo cách các ứng dụng được xây dựng để quản lý tương tác người dùng, xử lý công việc nội bộ, chuyển hướng hiệu quả và hiển thị nội dung
- Thiết kế kiến trúc được tiến hành song song với thiết kế giao diện, thẩm mỹ và nội dung

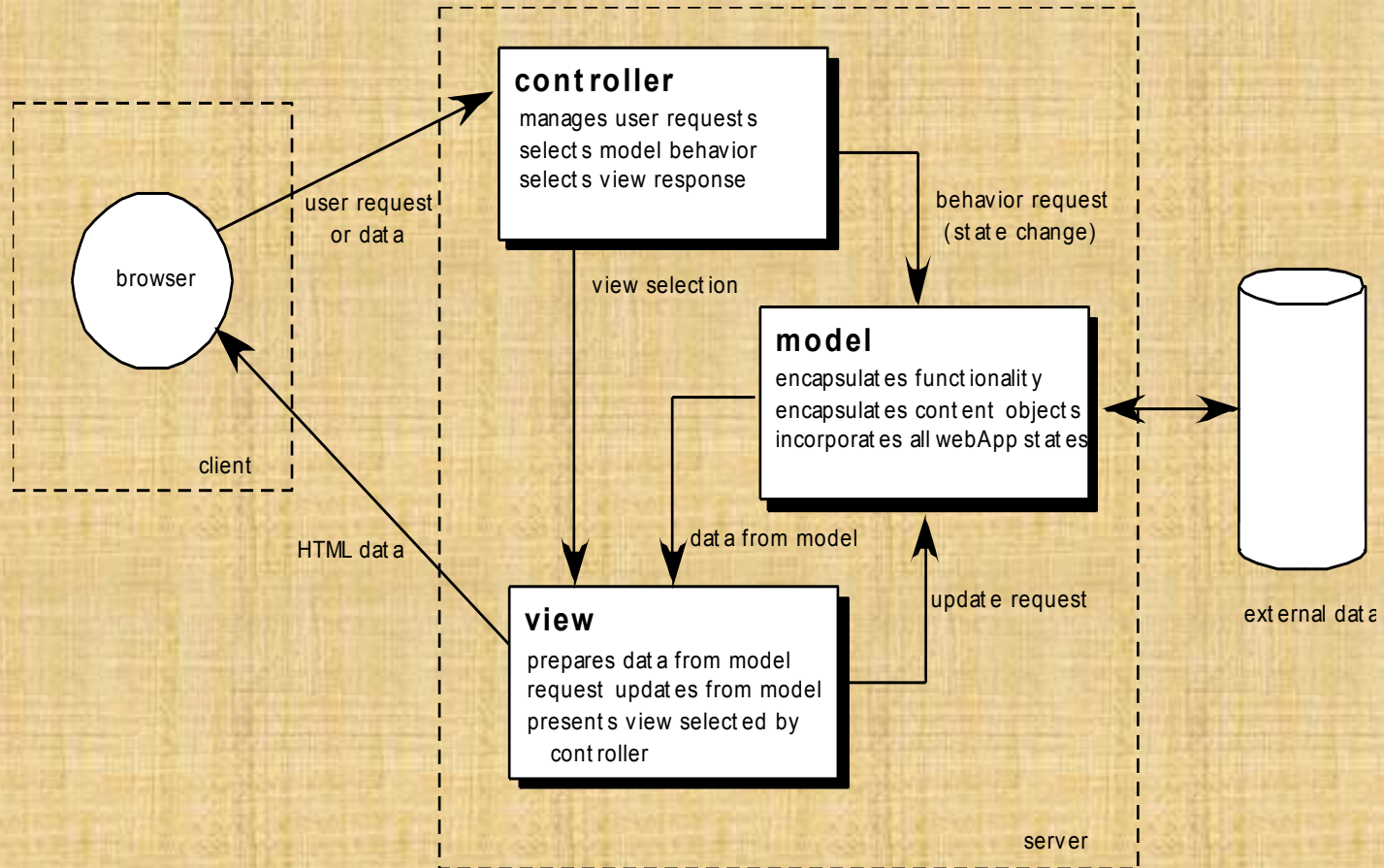
Thiết kế nội dung



Kiến trúc MVC

- **Model** chứa tất cả các nội dung cụ thể của ứng dụng và xử lý logic, bao gồm cả:
 - Mọi đối tượng nội dung
 - Truy cập dữ liệu bên ngoài
 - Mọi chức năng xử lý trong ứng dụng
- **View** chứa tất cả các giao diện chức năng cụ thể và cho phép:
 - Trình bày mọi nội dung và chức năng logic
 - Truy cập dữ liệu bên ngoài
 - Các chức năng được người dùng yêu cầu
- **Controller** cung cấp truy cập vào Model và View, điều khiển dòng dữ liệu

Mô hình MVC

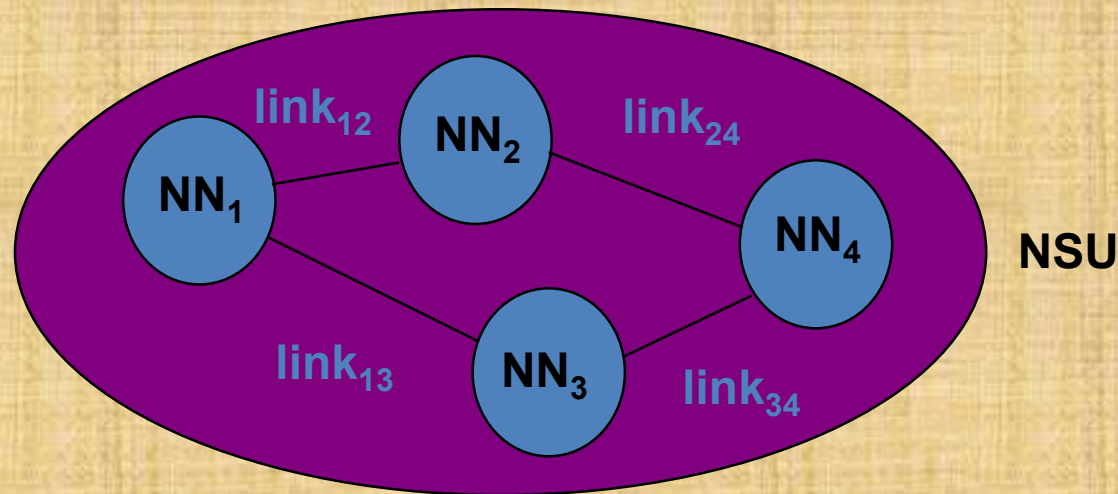


Thiết kế điều hướng

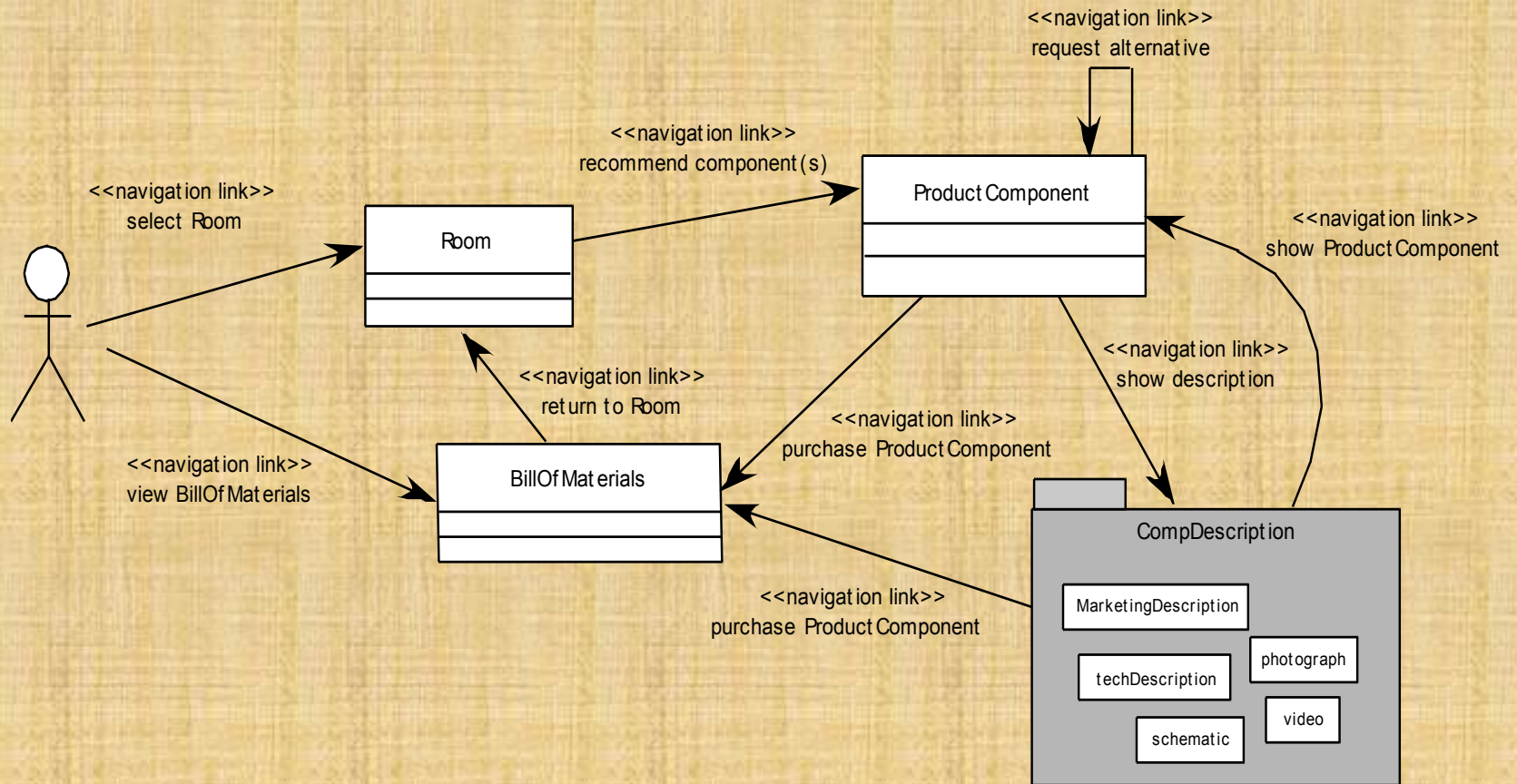
- Bắt đầu với việc xem xét cây phân cấp người dùng và use-case liên quan
 - Mỗi người dùng có thể sử dụng WebApp khác nhau và có yêu cầu điều hướng khác nhau
- Mỗi người dùng tương tác với WebApp sẽ xác định một **navigation semantic units (NSUs)**
 - NSU — “Tập thông tin, cấu trúc danh mục liên quan đến việc thực hiện một tập con của yêu cầu người dùng”

Navigation Semantic Units

- Navigation semantic unit
 - Cách điều hướng — đại diện cho cách chuyển hướng tốt nhất để đạt được mục tiêu, gồm các nút điều hướng liên kết bởi các cạnh điều hướng



Tạo một NSU



Cú pháp điều hướng

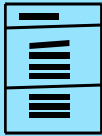
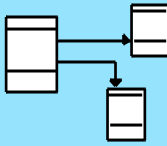
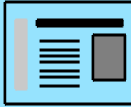

- **Liên kết điều hướng đơn** — liên kết dạng text, biểu tượng, nút, ...
- **Thanh điều hướng ngang** — danh sách các nội dung, chức năng có liên kết thích hợp (khoảng 4-7 loại)
- **Cột điều hướng dọc**
 - Liệt kê các nhóm nội dung, chức năng
 - Liệt kê tất cả các đối tượng, nội dung chính trong WebApp
- **Tabs** — một ẩn dụ tương tự như thanh/cột điều hướng, hiển thị trong một số tình huống
- **Site maps** — cung cấp một tab bao gồm tất cả các chuyển hướng đến các nội dung trong web

Thiết kế cấp thành phần

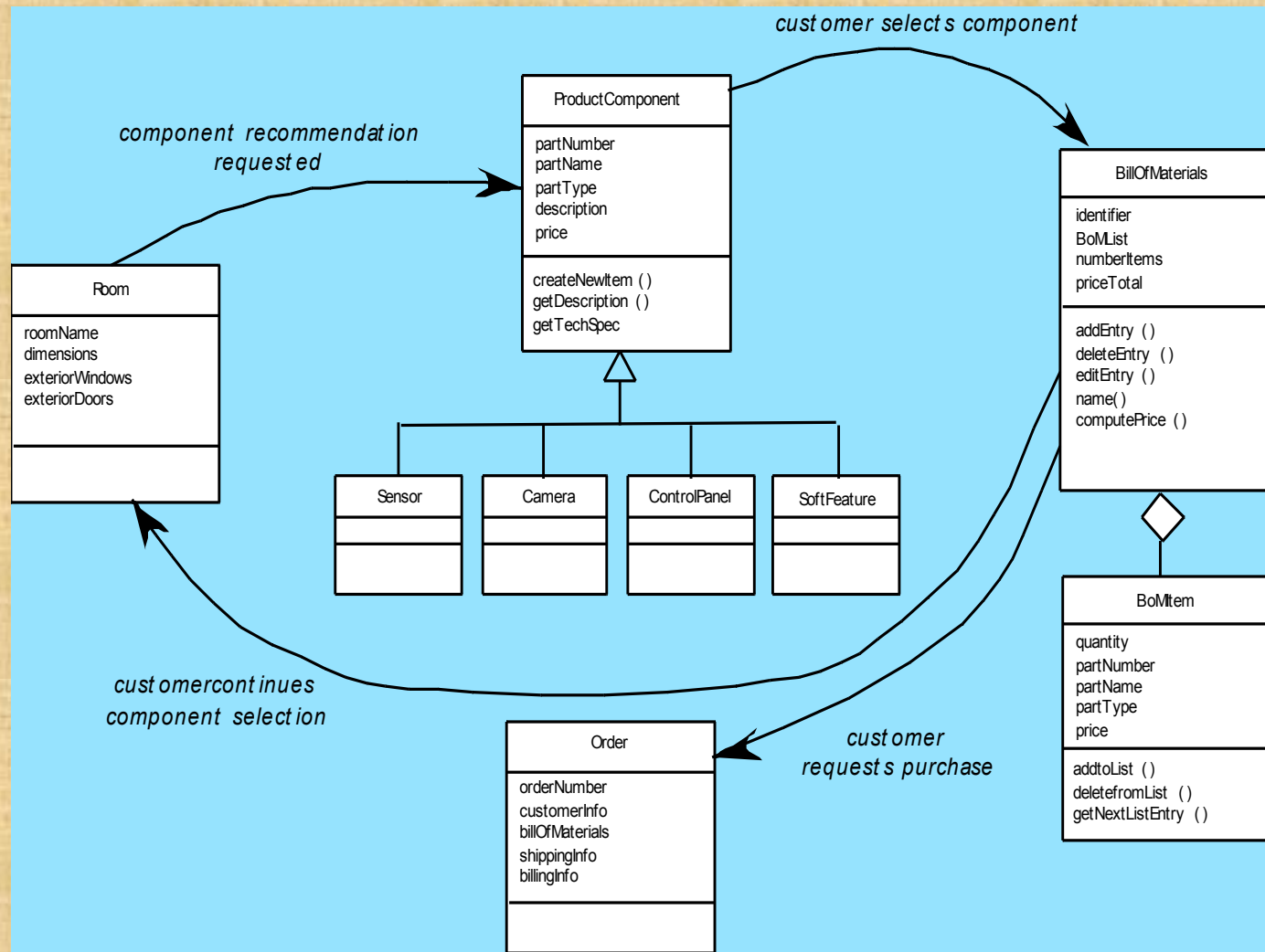
- Các thành phần của WebApp thực hiện chức năng:
 - Thực hiện xử lý địa phương, tạo nội dung, chuyển hướng
 - Cung cấp khả năng tính toán, xử lý dữ liệu
 - Cung cấp truy vấn cơ sở dữ liệu
 - Thiết lập giao diện dữ liệu với các hệ thống ngoài

OOHDM

- Object-Oriented Hypermedia Design Method (OOHDM)

				
	conceptual design	navigational design	abstract interface design	implementation
work products	Classes, sub-systems, relationships, attributes	Nodes, links, access structures, navigational contexts, navigational transformations	Abstract interface objects, responses to external events, transformations	executable WebApp
design mechanisms	Classification, composition, aggregation, generalization specialization	Mapping between conceptual and navigation objects	Mapping between navigation and perceptible objects	Resource provided by target environment
design concerns	Modeling semantics of the application domain	Takes into account user profile and task. Emphasis on cognitive aspects.	Modeling perceptible objects, implementing chosen metaphors. Describe interface for navigational objects	Correctness; Application performance; completeness

Lược đồ khái niệm



Tài liệu tham khảo

- Slide Set to accompany Software Engineering: A Practitioner's Approach, 7/e by Roger S. Pressman
- <http://bit.ly/2ihOLB4>