TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

NHẬP MÔN CNPM

Nội dung / Chương 8-3: Thiết kế chi tiết phần mềm

Thông tin GV

Nội dung chương

Phần 1. ABC

Phần 2. ABC

Phần 3. ABC

Phần 4. ABC

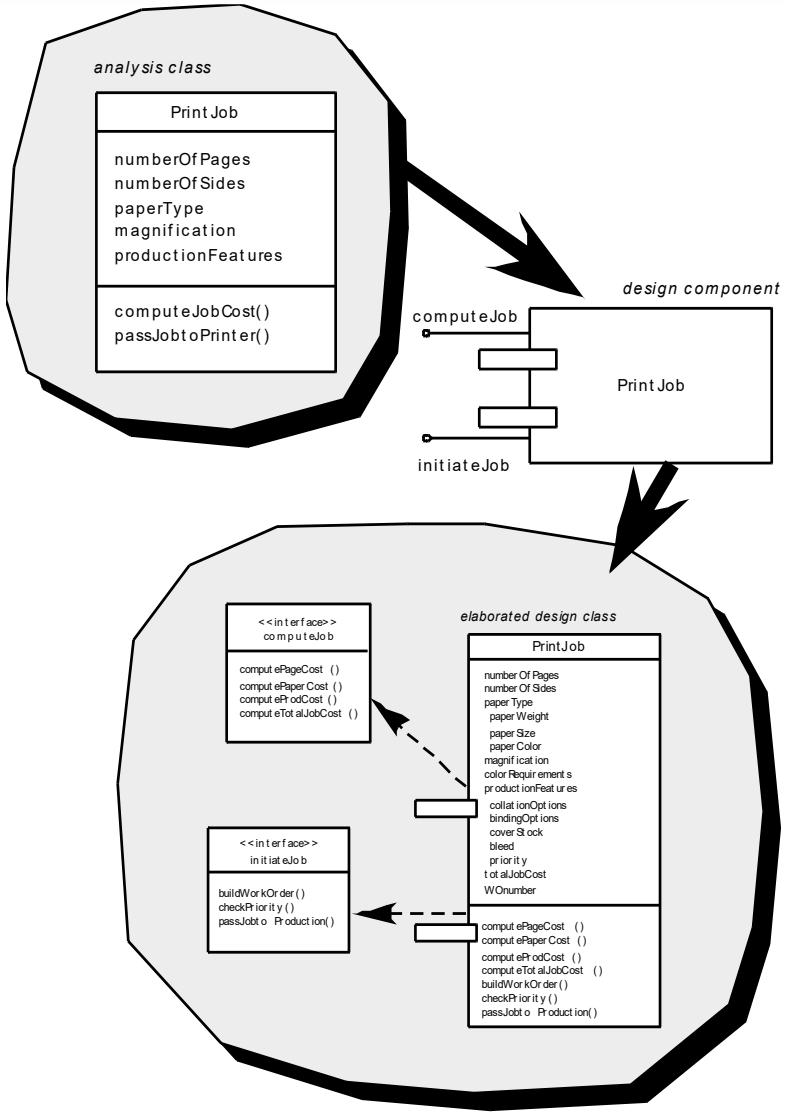
Phần 5. ABC

Phần 6. ABC

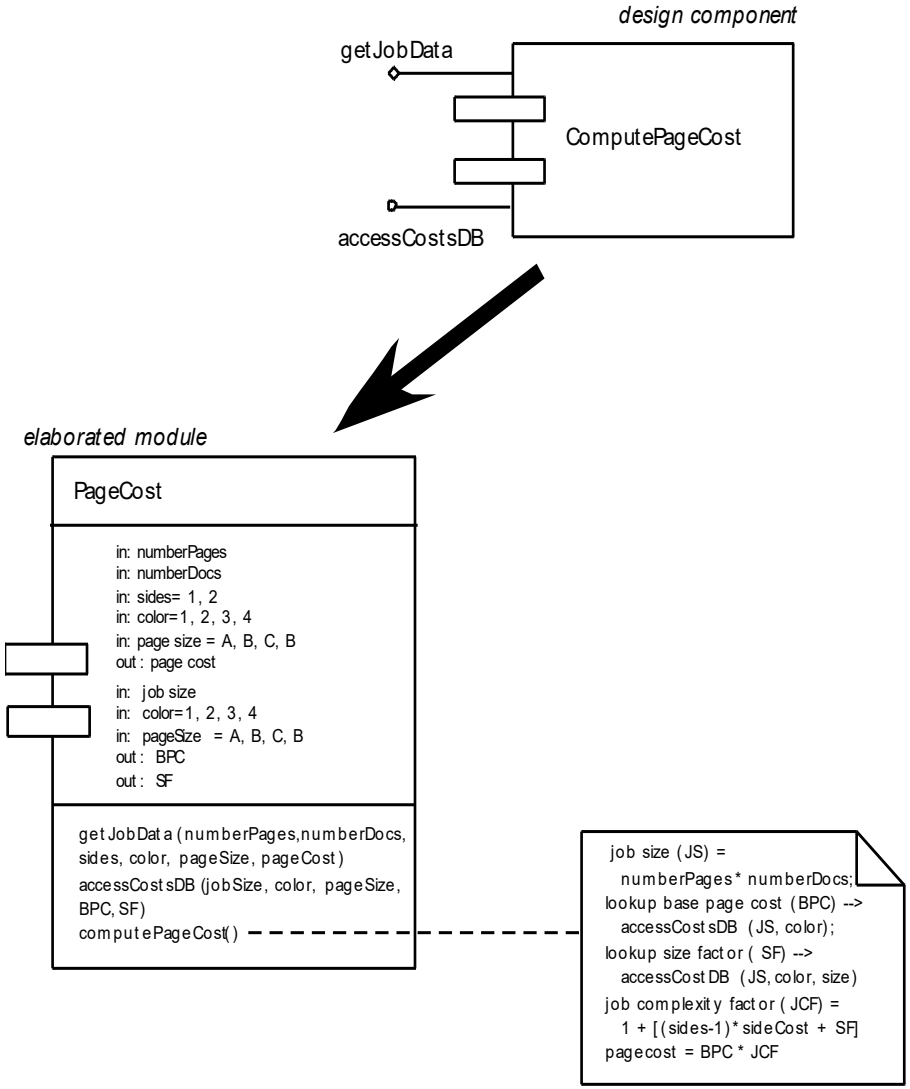
Thế nào là một thành phần?

- *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



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 tráo 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

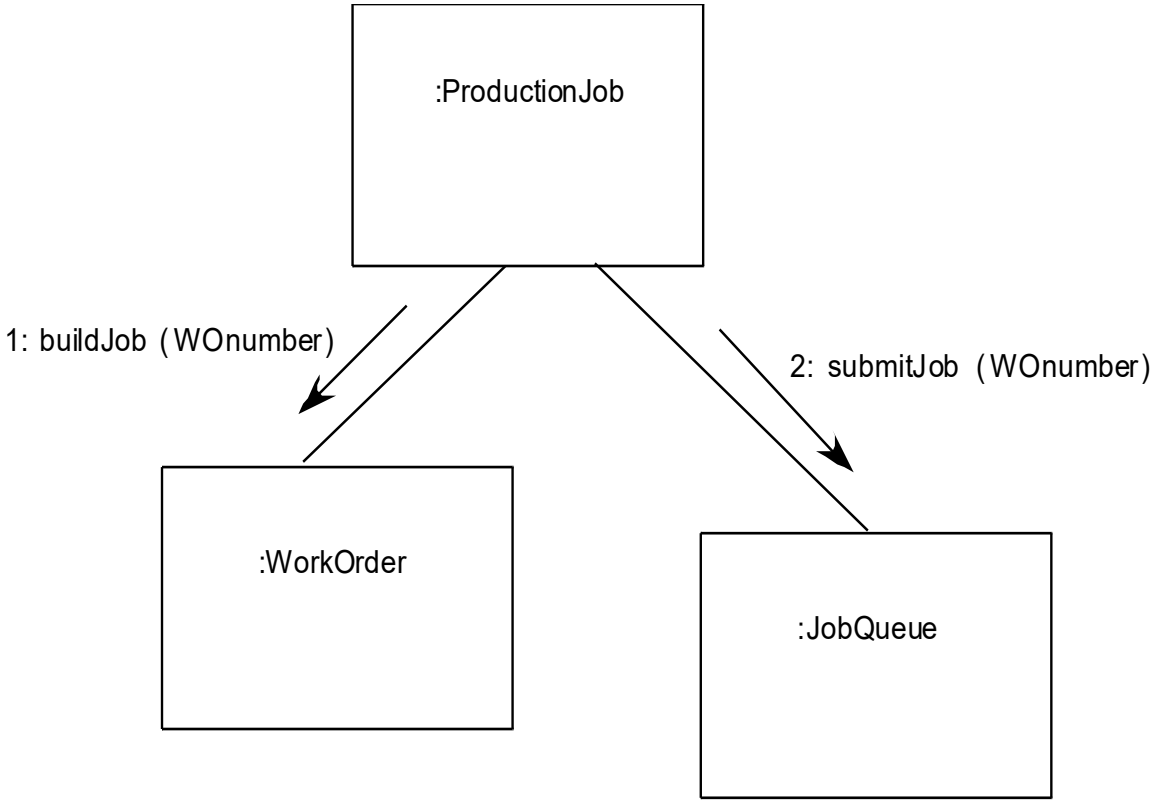
Thiết kế mức thành phần - I

- 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.

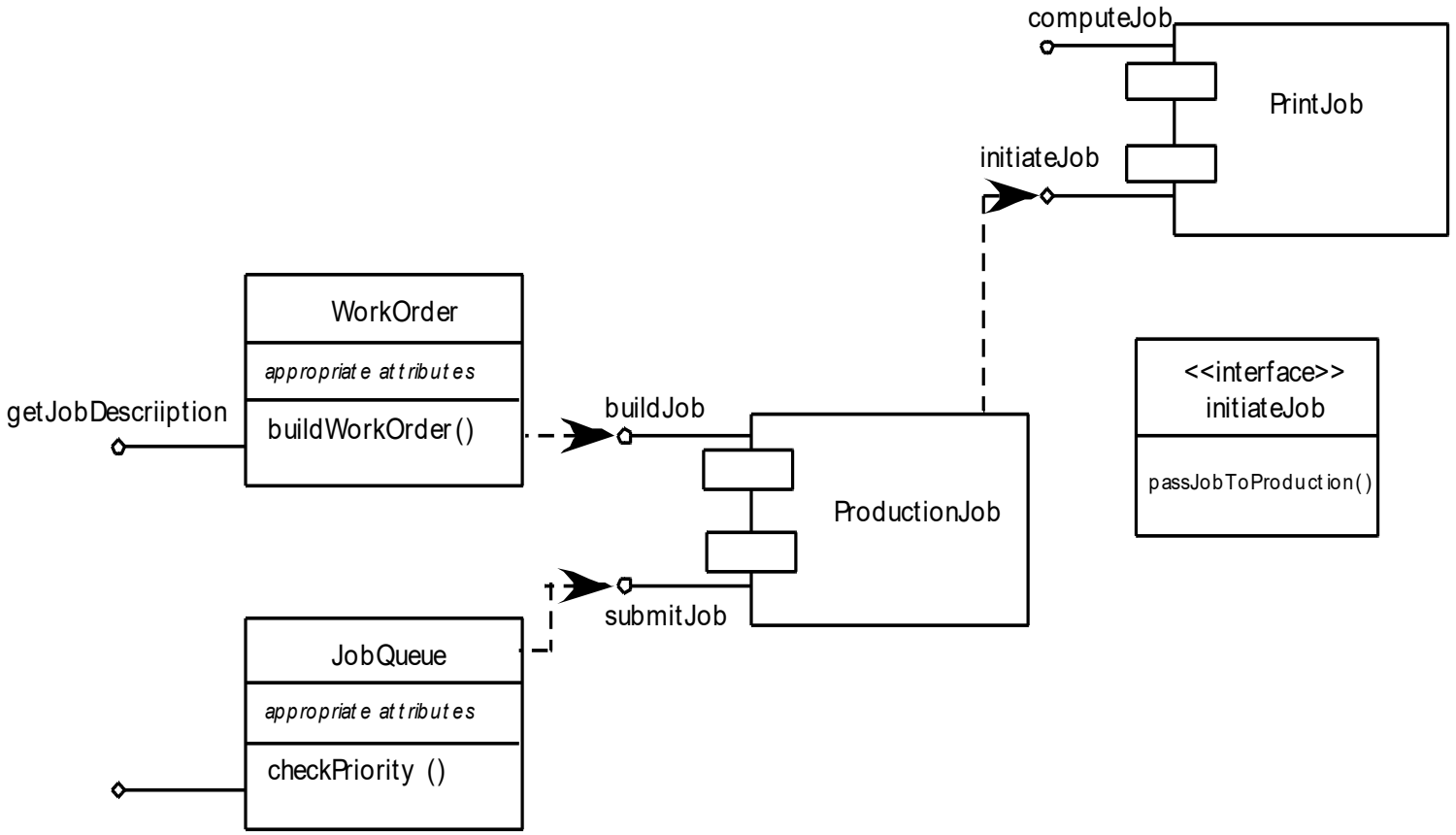
Thiết kế mức thành phần - II

- 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.

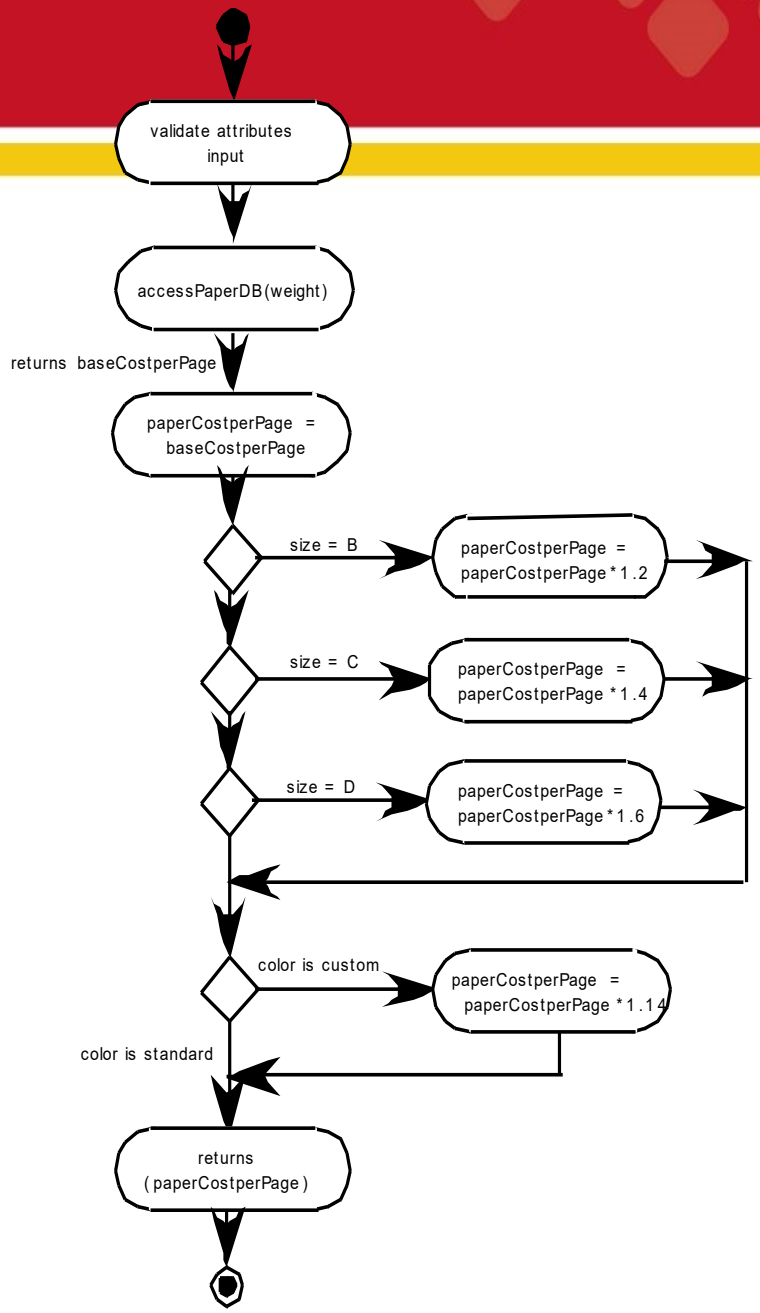
Sơ đồ cộng tác



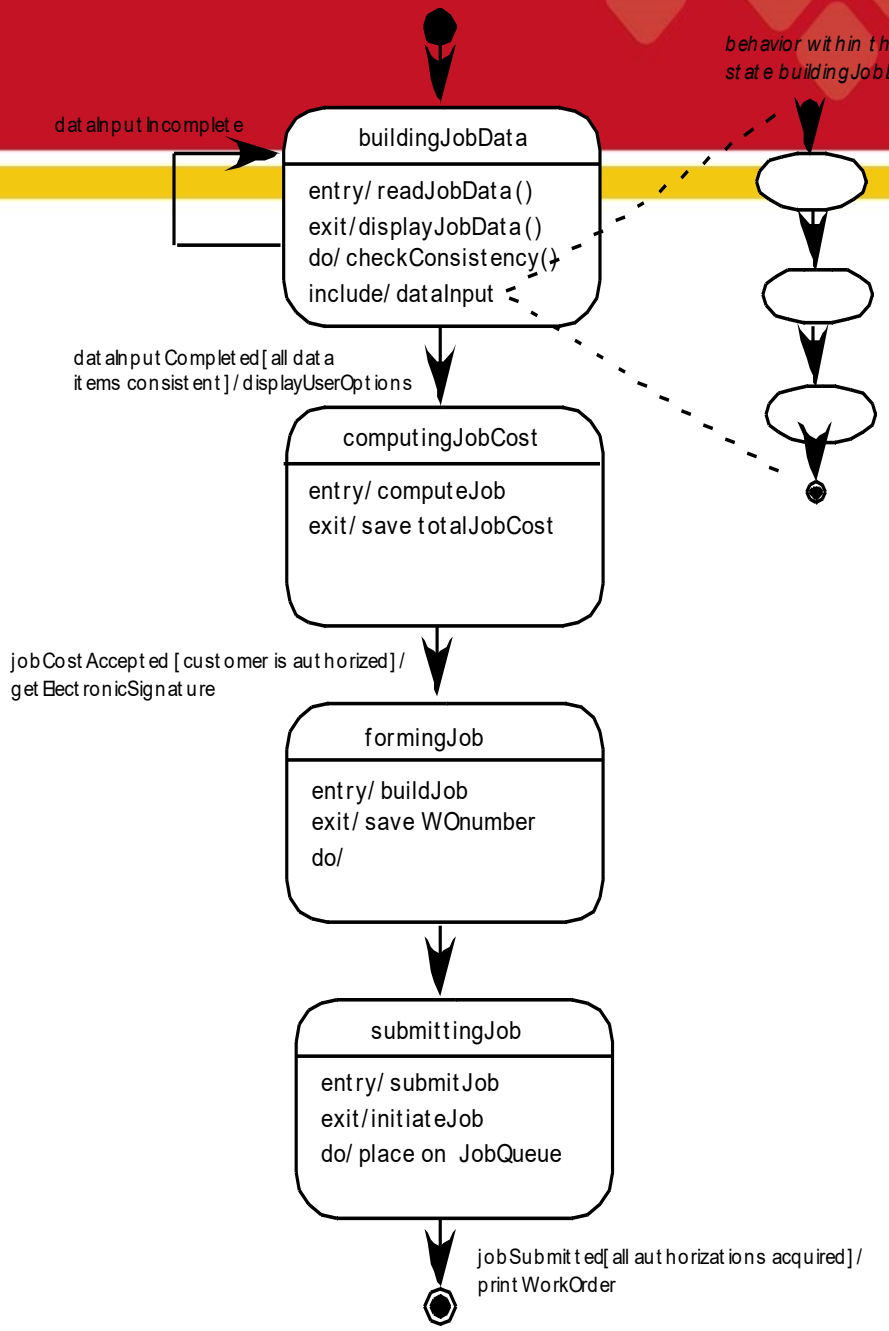
Tái cấu trúc


computeJob
PrintJob
initiateJob
WorkOrder
appropriate attributes
buildWorkOrder()
getJobDescriiption
buildJob
ProductionJob
<<interface>>
initiateJob
passJobToProduction()
submitJob
JobQueue
appropriate attributes
checkPriority ()

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 lí 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 lí 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.

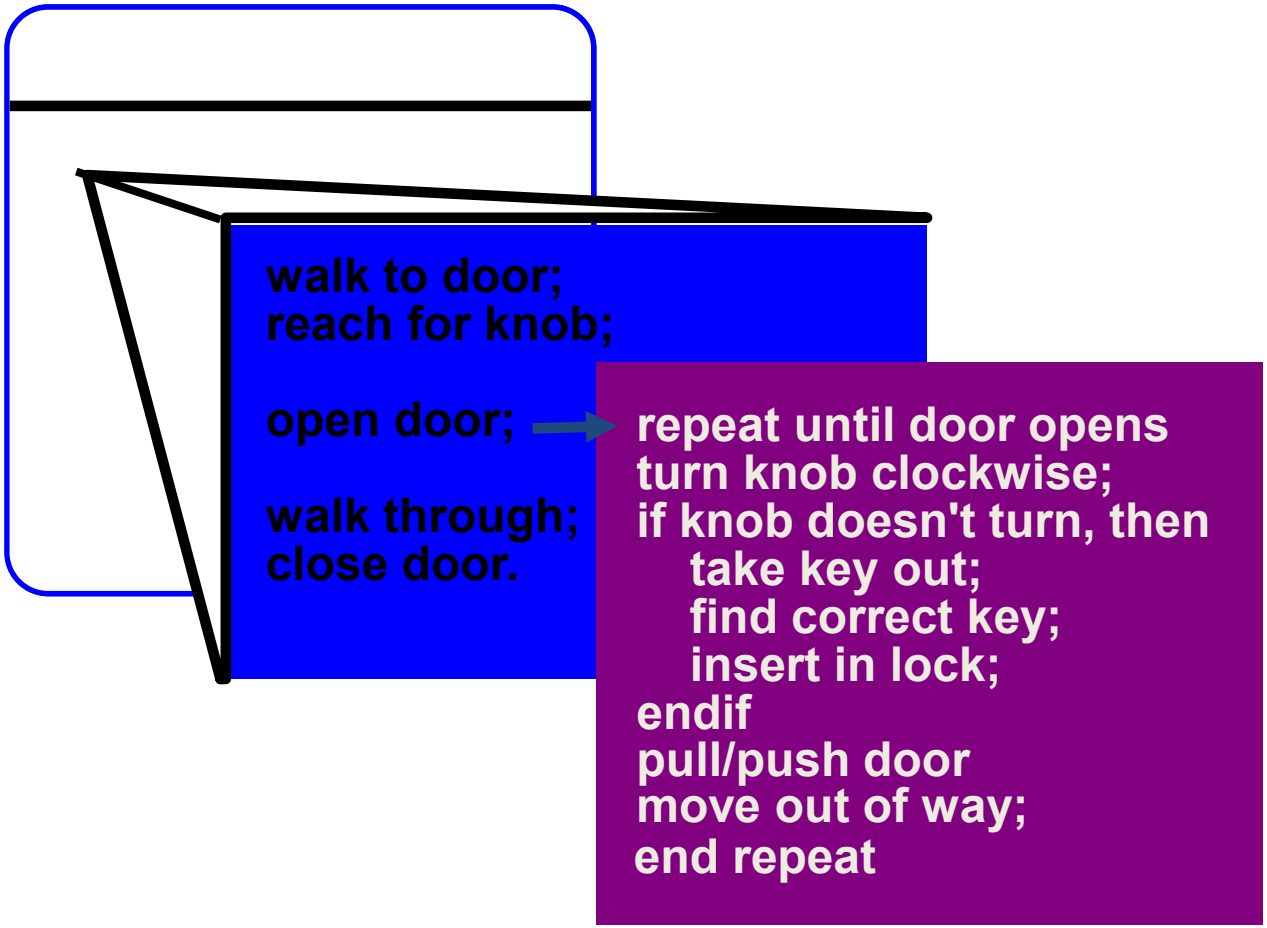
Thiết kế thành phần quy ước

- 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



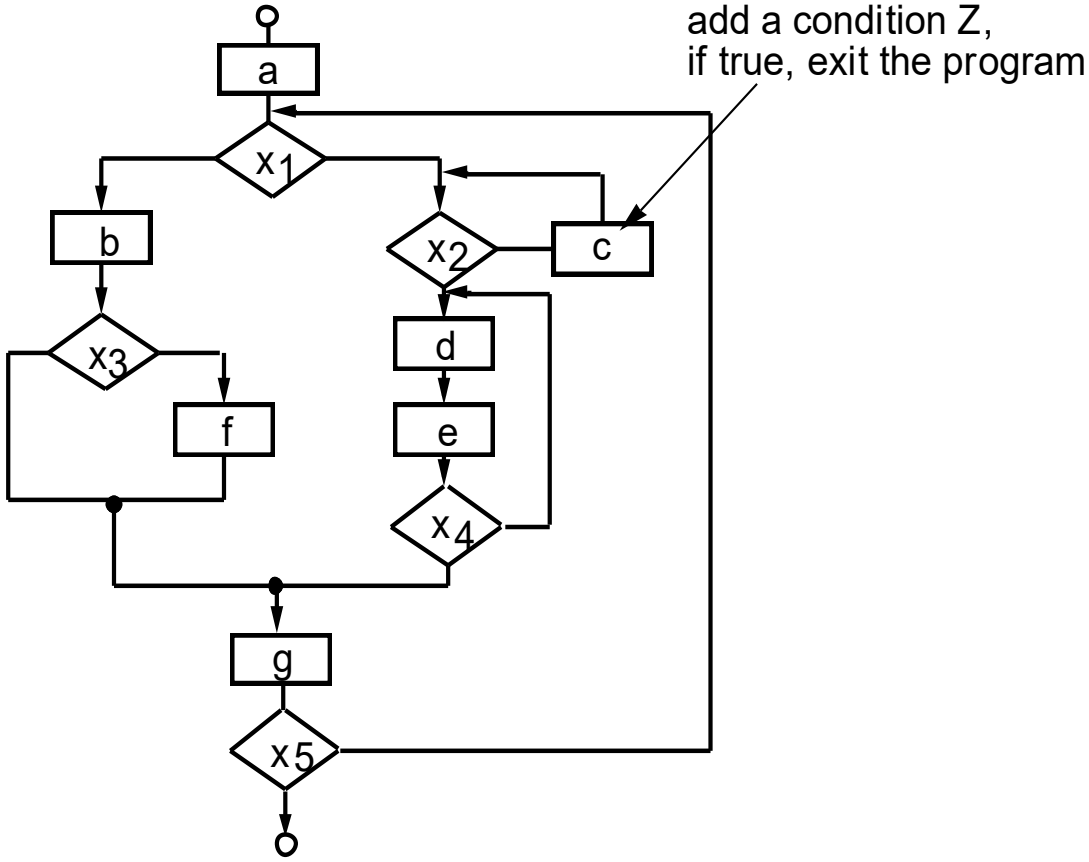
Mô hình thiết kế thuật toán

- 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.

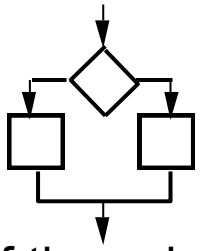
Một thiết kế thủ tục có cấu trúc



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-then-else

```
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

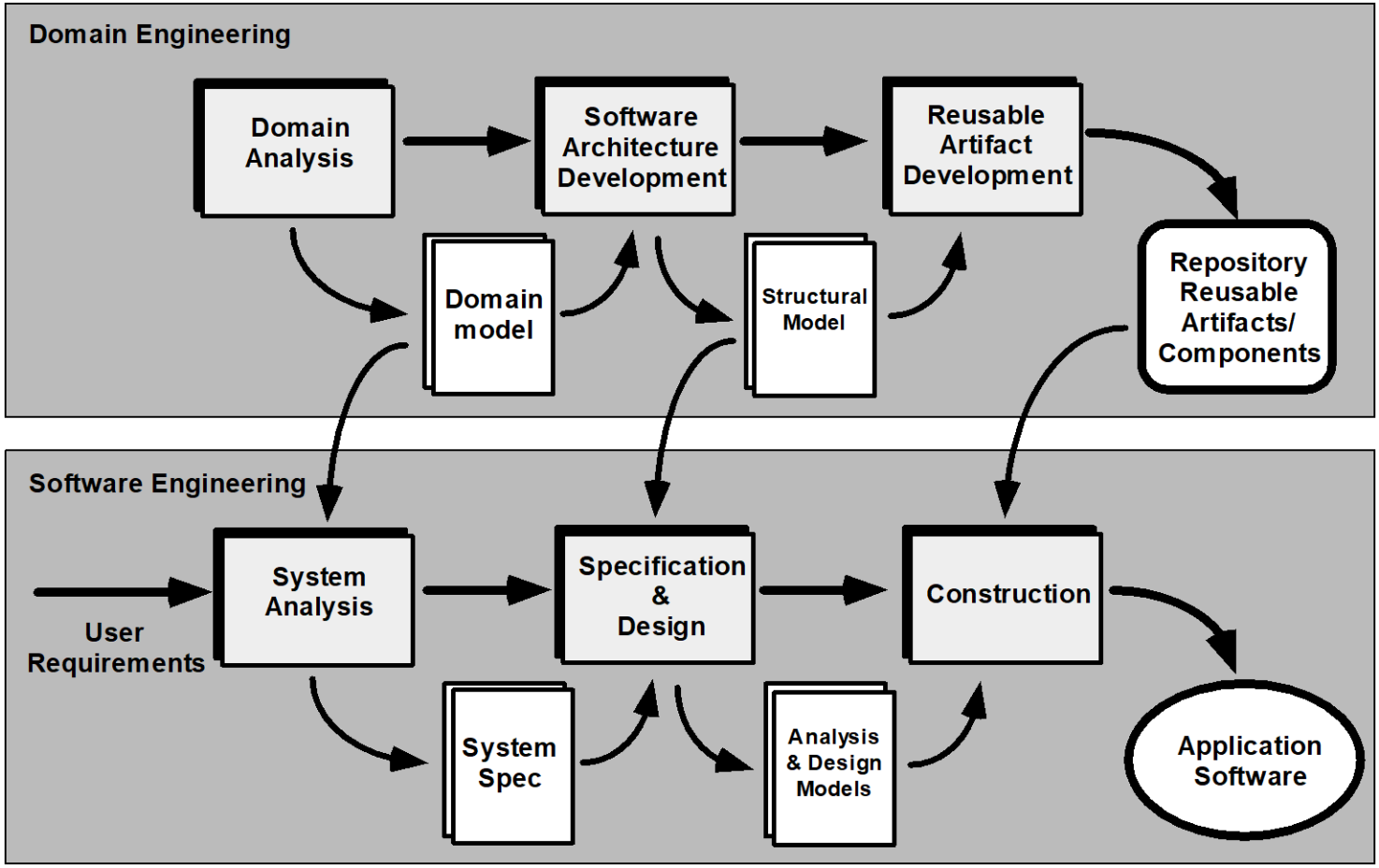
Sự phát triển dựa trên thành phần

- 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



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

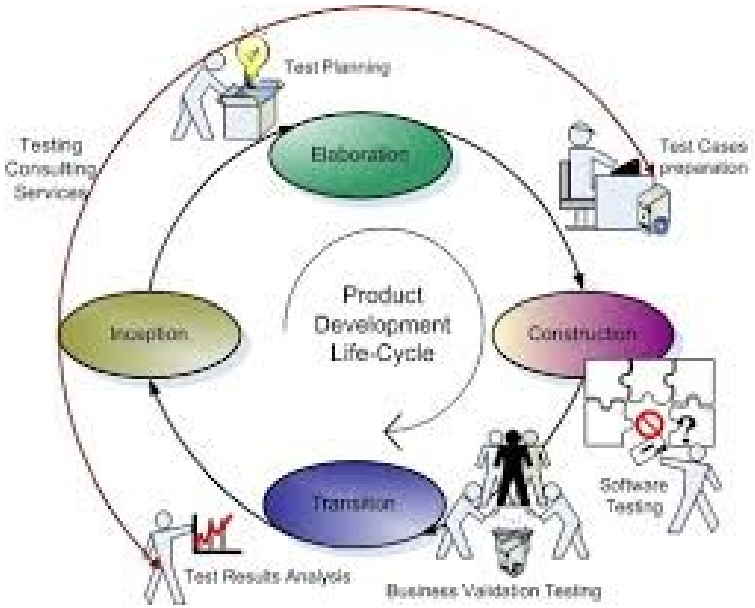
- Chức năng thành phần có cần thiết cho việc triển khai trong tương lai?
- Mức độ phổ biến của chức năng của các thành phần trong các tên miền là như thế nào?
- Có trùng lặp chức năng của các thành phần trong các tên miền không?
- Thành phần có phụ thuộc phần cứng không?
- Liệu các phần cứng có giữ nguyên 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



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 "dễ dàng 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.

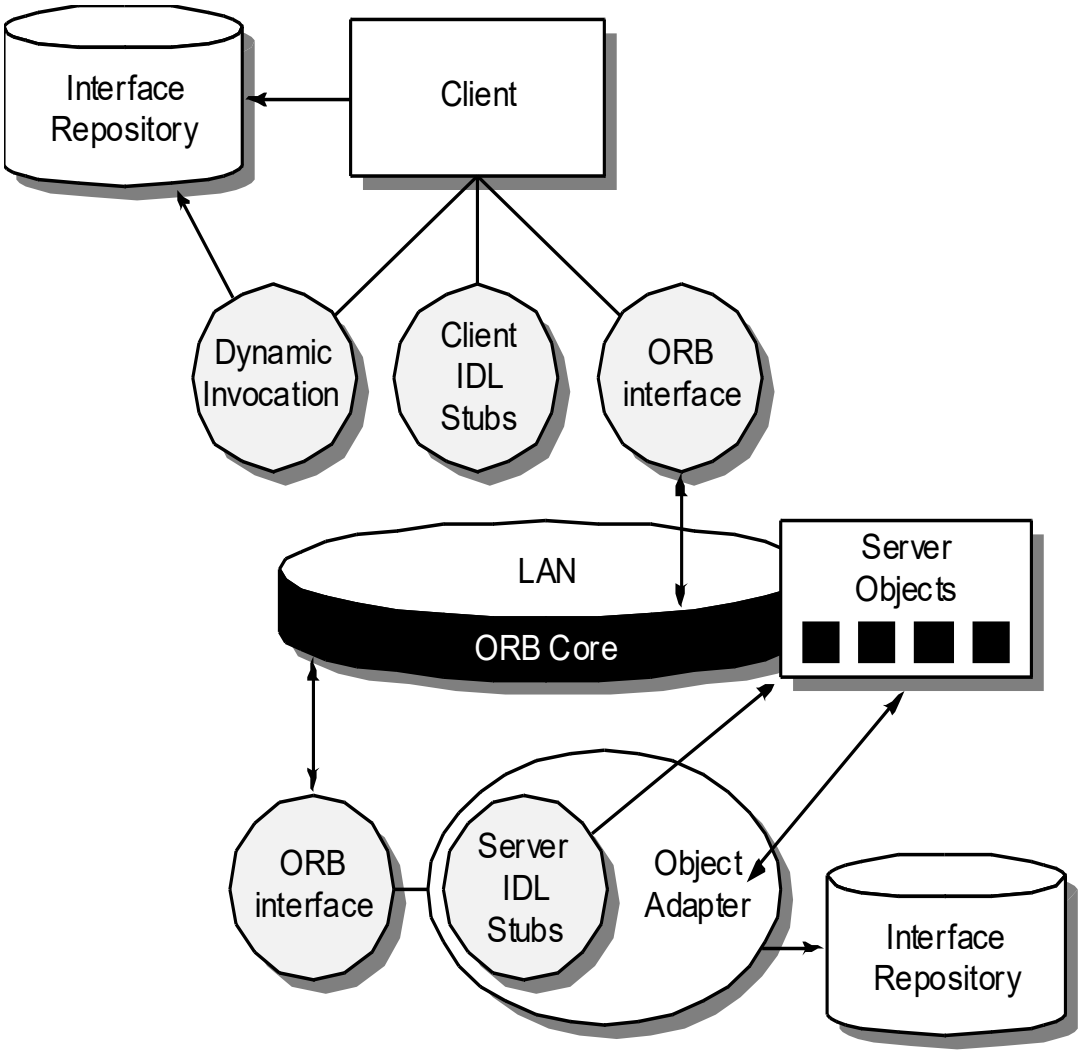
Ghép nối (composition)

- 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.

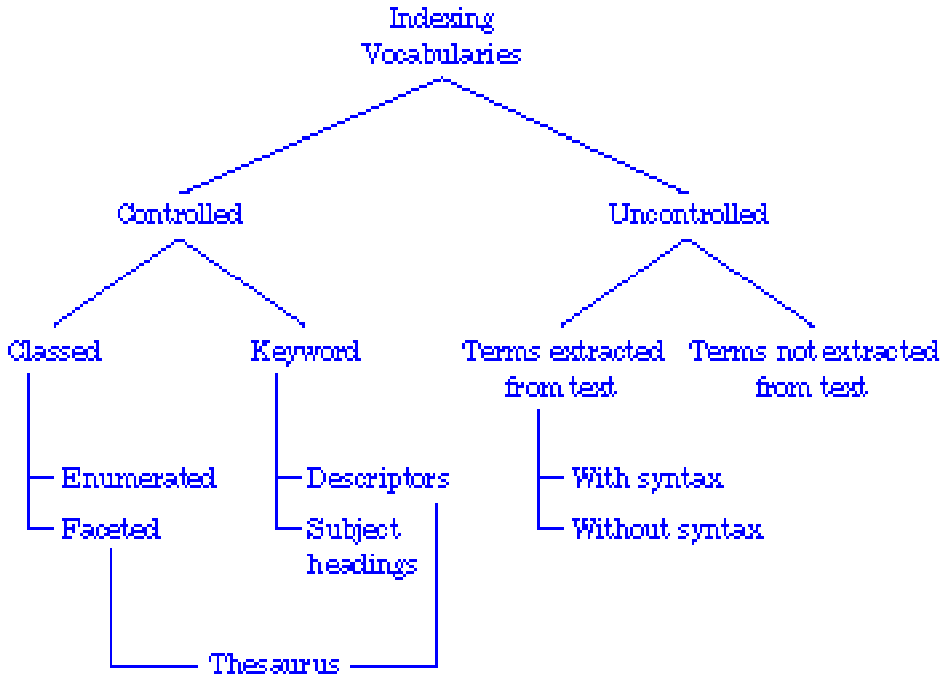
Sun JavaBeans

- 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


Indexing Vocabularies
Controlled
Uncontrolled
Classed
Keyword
Terms extracted from text
Terms not extracted from text
Enumerated
Faceted
Descriptors
Subject headings
With syntax
Without syntax
Thesaurus

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.

Tài liệu tham khảo

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