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

NHẬP MÔN CNPM

Nội dung / Chương 7: Mô hình các yêu cầu: Flow, Behavior, Patterns, and WebApps

Thông tin GV

Nội dung chương

Phần 1. ABC

Phần 2. ABC

Phần 3. ABC

Phần 4. ABC

Phần 5. ABC

Phần 6. ABC

Các giải pháp mô hình các yêu cầu

- Một khung nhìn của mô hình các yêu cầu được gọi là *structured analysis (phân tích có cấu trúc),* suy xét dữ liệu các tiến trình mà thay đổi dữ liệu như các thực thể riêng biệt.
  - Data objects(dữ liệu đối tượng) được mô hình bằng cách định nghĩa các trạng thái và quan hệ giữa chúng.
  - Các tiến trình thao tác đối với dữ liệu đối tượng được mô hình để cho thấy làm thế nào biến đổi dữ liệu như dữ liệu đối tượng thông qua hệ thống.
- Một đề xuất thứ hai để phân tích mô hình, được gọ là object-oriented analysis (*phân tích hướng đối tượng*), tập trung vào:
  - Định nghĩa các lớp
  - Cách thức mà chúng liên kết với nhau để thực hiện các yêu cầu của khách hàng.

Flow-Oriented Modeling

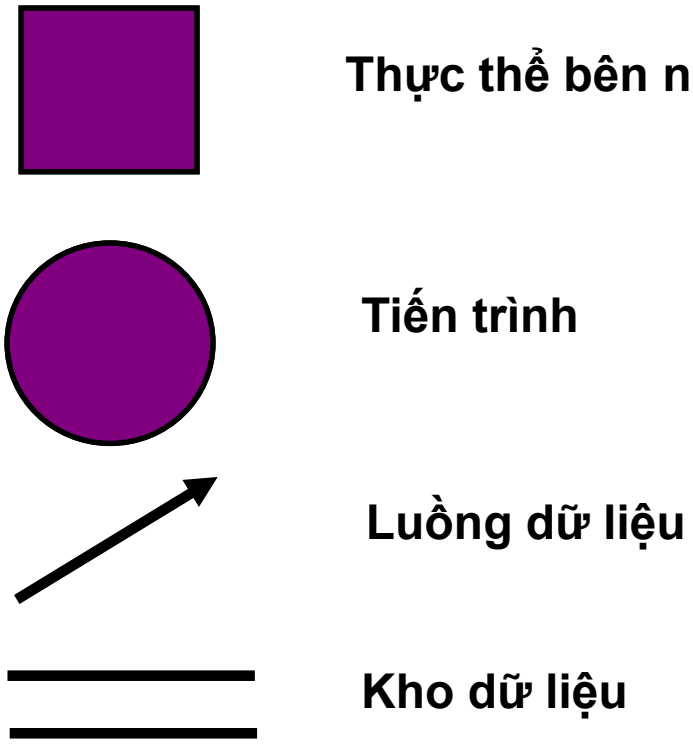
- Trình diễn làm thế nào dữ liệu đối tượng được biến đổi thông qua hệ thống data flow diagram (biểu đồ luồng dữ liệu) (DFD) là giản đồ mẫu được sử dụng
- Được nhiều người nhìn nhận là một cách tiếp cận "old school“ (trường phái cũ), nhưng nó vẫn tiếp tục cung cấp một khía cạnh duy nhất nó có của hệ thống nên vẫn được sử dụng để bổ sung các yếu tố vào các mô hình phân tích khác.

The Flow Model

- Mỗi hệ thống máy tính cơ bản là một hệ thống biến đổi thông tin

Các ký hiệu mô hình luồng DL

- Mỗi hệ thống máy tính cơ bản là một hệ thống biến đổi thông tin



Thực thể ngoài

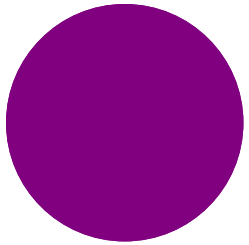
Một sản xuất hoặc tiêu thụ dữ liệu

Ví dụ: một người, một thiết bị, một cảm biến...

Ví dụ khác: hệ thống máy tính cơ bản

Dữ liệu luôn luôn phải xuất phát từ một nơi nào đó và luôn luôn phải được gửi đến một cái gì đó.

Tiến trình

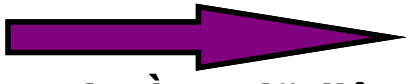


Một sự biến đổi dữ liệu(thay đổi đầu vào qua đầu ra)

Ví dụ: *tính thuế, xác định diện tích,*
Định dạng báo cáo, hiển thị đồ thị

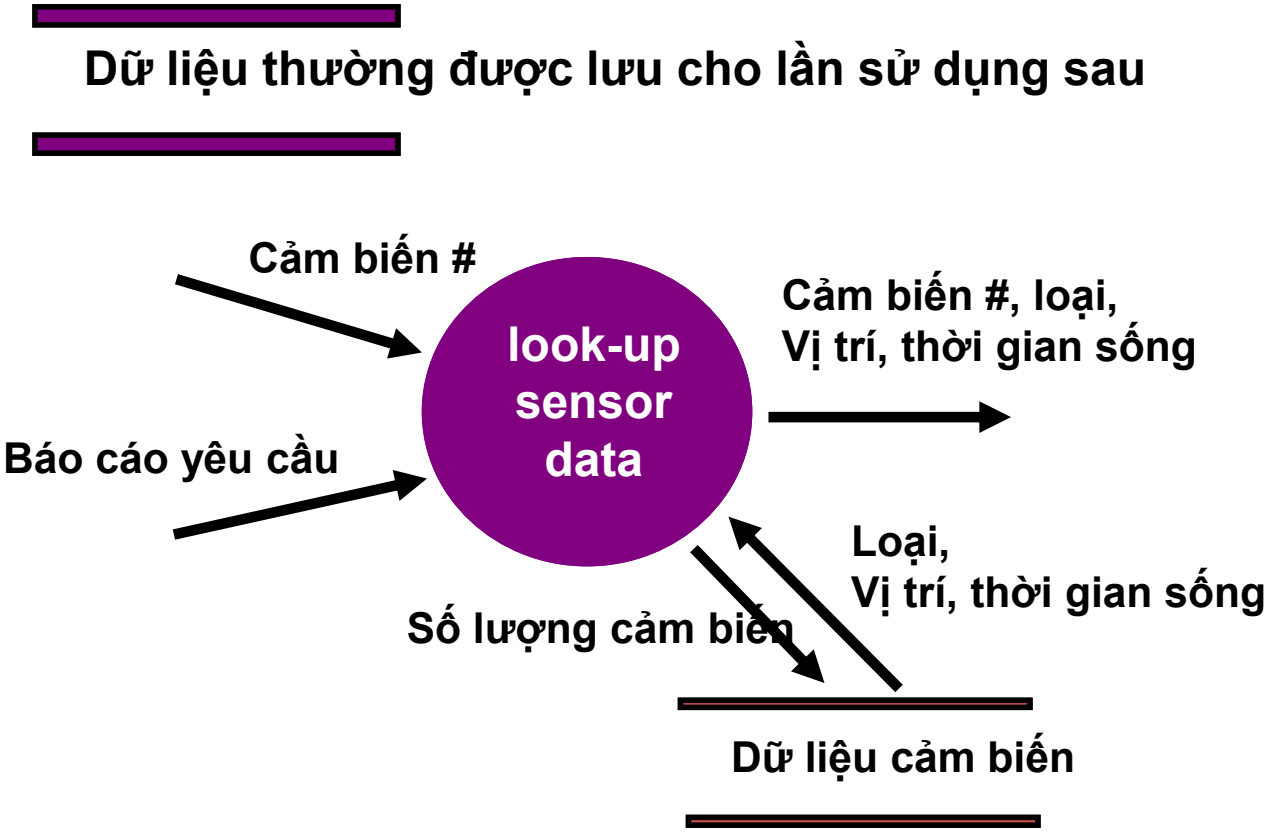
Dữ liệu phải luôn luôn được xử lý bằng cách nào đó để đạt được chức năng của hệ thống

Luồng dữ liệu



Luồng dữ liệu thông qua một hệ thống, bắt đầu như đầu vào và biến đổi thành đầu ra.

Các kho dữ liệu



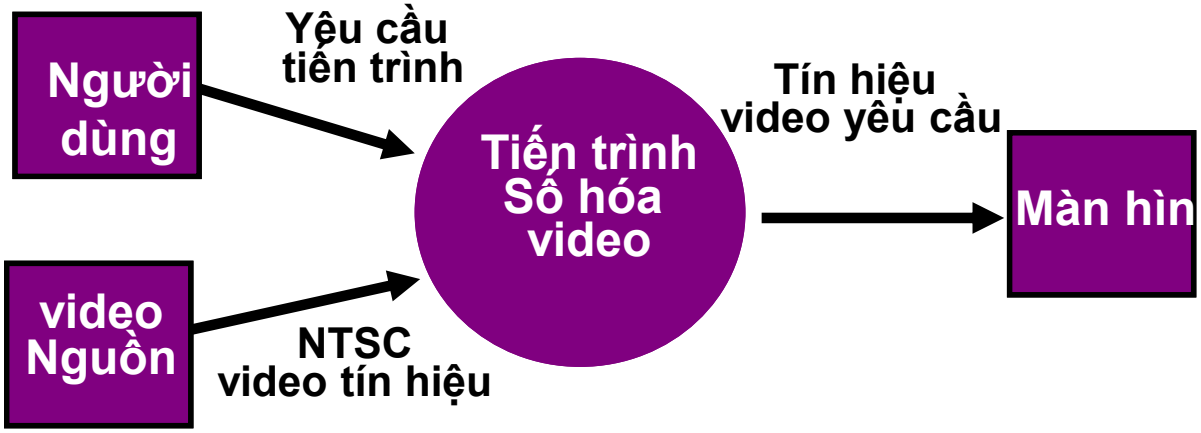
Biểu đồ luồng dữ liệu: hướng dẫn

- Tất cả các biểu tượng phải được gán nhãn có nghĩa đầy đủ
- Biểu đồ DFD phát triển thông qua số lượng mức các chi tiết
- Luôn luôn bắt đầu với một mức nội dung (gọi là level 0)
- Luôn luôn chỉ ra các thực thể ngoài tại mức 0
- Luôn luôn gán nhãn cho hướng luồng dữ liệu
- Không thể hiện các thủ tục logic

Xây dựng một DFD—I

- Xem xét kịch bản người dùng và/hoặc để tách biện các dữ liệu đối tượng và sử dụng một phân tích ngữ pháp để xác định "hoạt động"
- Xác định thực thể ngoài (các sản xuất và tiêu thụ của dữ liệu)
- Tạo mức 0 DFD

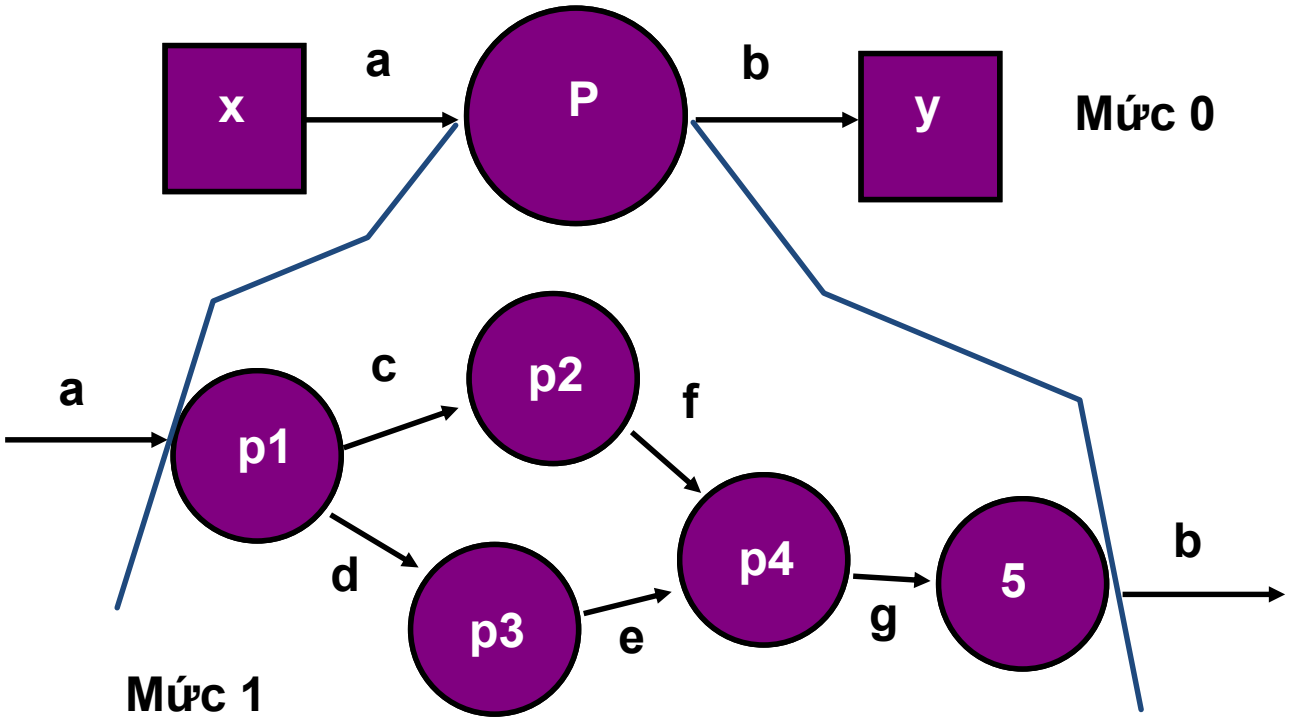
Ví dụ mức 0 DFD



Xây dựng một DFD—II

- Viết một kịch bản mô tả biến đổi.
- Phân tích cú pháp để xác định biến đổi ở cấp độ tiếp theo
- “Cân bằng" luồng để duy trì luồng dữ liệu liên tục
- Phát triển một mức 1DFD
- Sử dụng một tỷ lệ 1:5 để mở rộng (xấp xỉ).

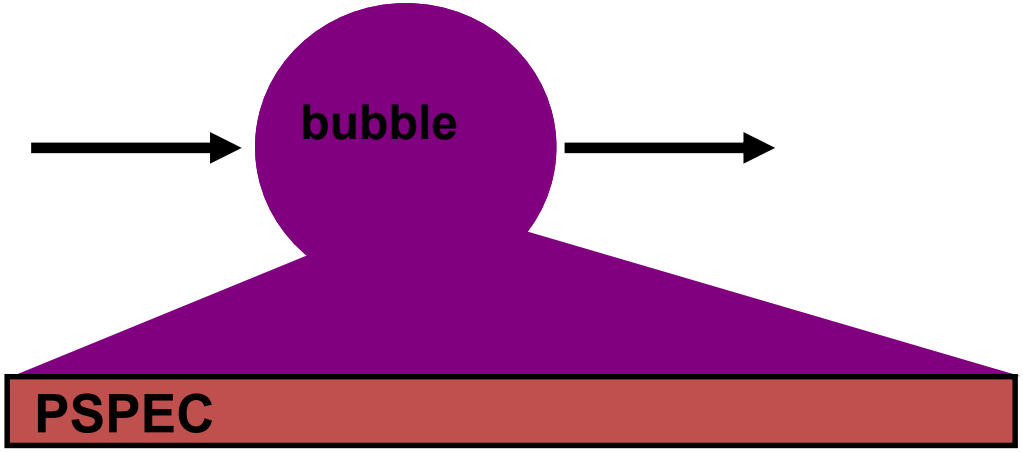
Hệ thống phân cấp luồng dữ liệu



Các ghi chú

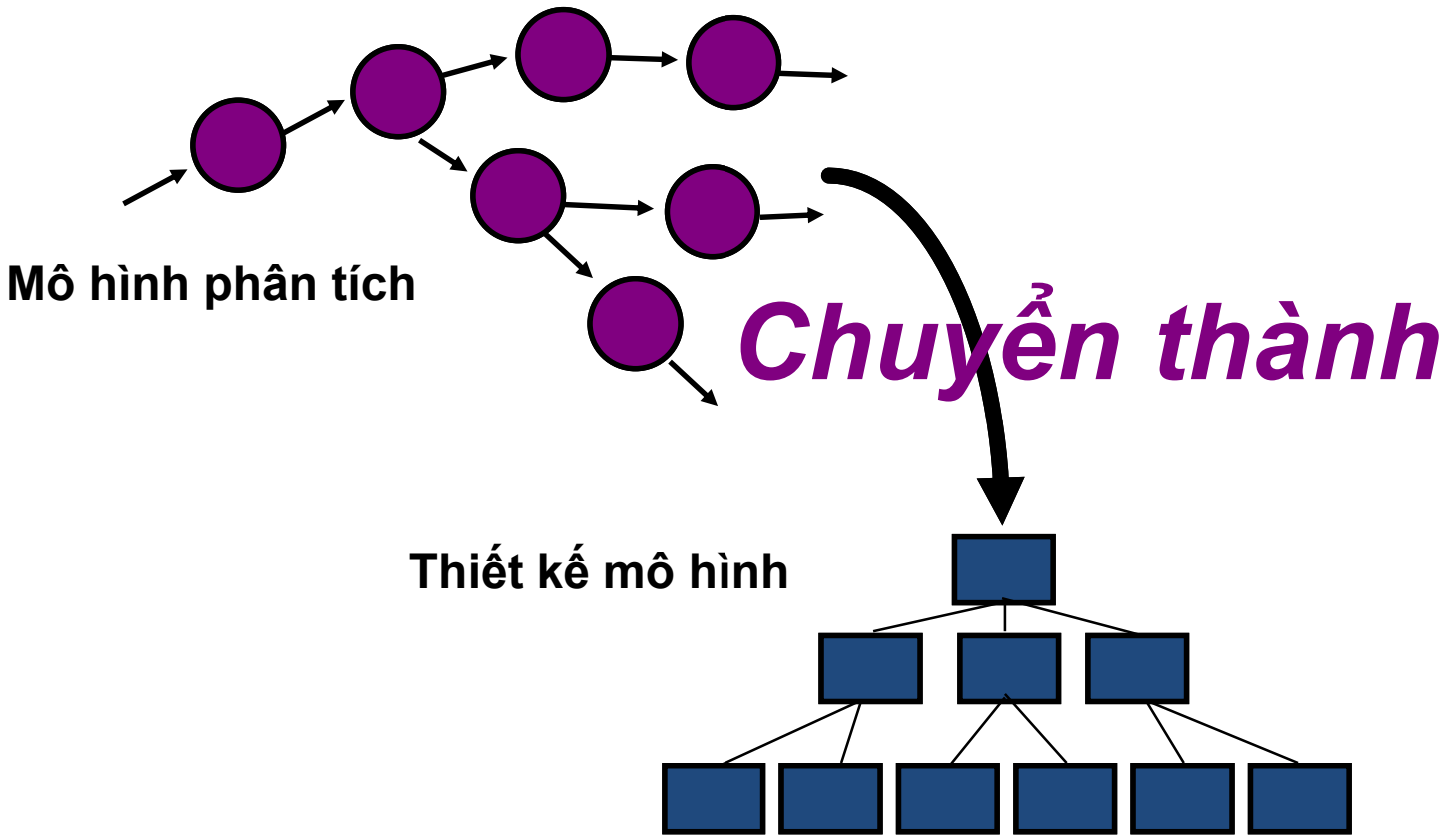
- Mỗi ô tròn là đơn giá trị cho đến khi nó làm gì đấy
- Tỷ lệ mở rộng giảm khi số lượng mức tăng
- Phần lớn hệ thống yêu cầu giữa 3 và 7 cấp cho một mô hình đầy đủ luồng
- Một mục lưu lượng dữ liệu duy nhất (mũi tên) có thể được mở rộng như cấp độ tăng(dữ liệu từ điển cung cấp thông tin)

Đặc tả Tiến trình (PSPEC)



- Kịch bản
- Mã giả(PDL)
- Phương trình
- Các bảng
- Biểu đồ và/hoặc

DFDs: tương lai



Mô hình luồng Điều khiển

- Biểu diễn các "sự kiện" và các tiến trình mà quản lý các sự kiện
- Một "sự kiện" là một điều kiện đúng/sai mà có thể chắc chắn bởi:
  - Nghe tất cả các cảm biến mà "đọc" bởi phần mềm
  - Nghe tất cả các ngắt điều kiện.
  - Nghe tất cả các "ngắt" mà được dẫn động bởi hệ thống.
  - Nghe tất cả các dữ liệu điều kiện.
  - Nhắc lại phân tích danh từ/động từ được áp dụng cho kịch bản sản xuất, xem xét tất cả "kiểm soát mục" như có thể CSPEC đầu vào/đầu ra..

Đặc tả Điều khiển (CSPEC)

CSPEC có thể là:

- Biểu đồ trạng thái (biểu đồ tuần tự)
- Bảng chuyển đổi trạng thái
- Bảng quyết định
- Bảng kích hoạt

Tổ hợp

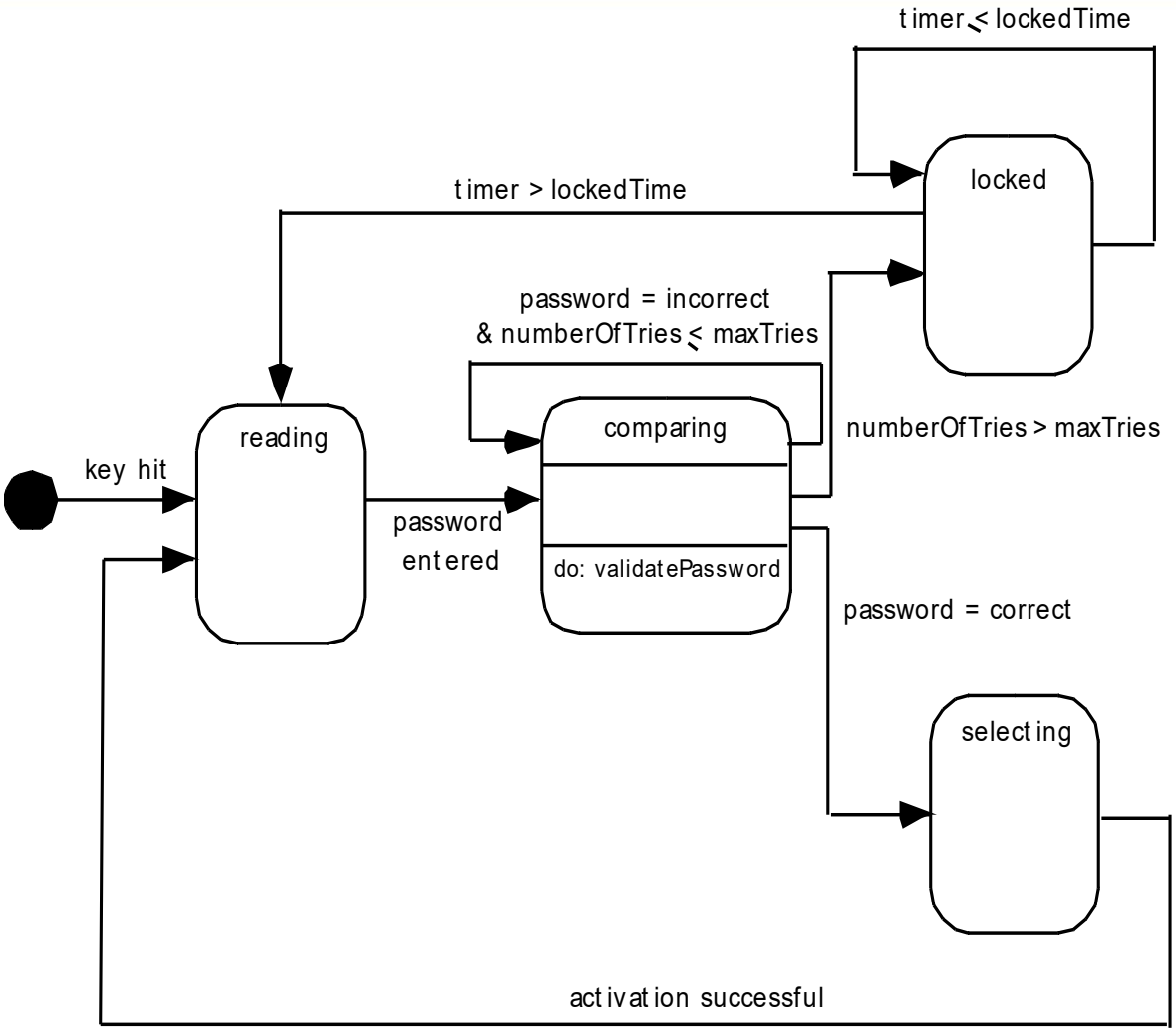
Mô hình hành vi

- Mô hình hành chỉ ra cách phần mềm sẽ phản ứng với các sự kiện hoặc kích thích bên ngoài. Để tạo ra các mô hình, các nhà phân tích phải thực hiện theo các bước sau:
  - Đánh giá tất cả users-case để hiểu đầy đủ trình tự của các tương tác trong hệ thống.
  - Xác định sự kiện điều khiển dãy tương tác và hiểu làm thế nào những sự kiện liên quan đến các đối tượng cụ thể..
  - Tạo ra chuỗi cho mỗi use-case.
  - Xây dựng một biểu đồ trạng thái cho hệ thống.
  - Xem lại các mô hình hành vi để xác minh tính chính xác và nhất quán.

Biểu diễn trạng thái

- Trong bối cảnh của mô hình hành vi, 2 đặc điểm khác nhau của trạng thái phải được xem xét:
  - trạng thái của mỗi lớp là hệ thống thực hiện chức năng của nó và
  - trạng thái của hệ thống như quan sát từ bên ngoài như hệ thống thực hiện chức năng của nó
- Trạng thái của một lớp có trên cả hai đặc tính chủ động và thụ động[CHA93].
  - Một trạng thái thụ động chỉ đơn giản là tình trạng hiện tại của tất cả các thuộc tính của một đối tượng.
  - Các trạng thái hoạt động của một đối tượng cho biết tình trạng hiện tại của đối tượng như nó trải qua một sự biến đổi hoặc tiến trình tiếp tục.

Biểu đồ trạng thái cho các lớp ControlPanel



Các trạng thái của một hệ thống

- Trạng thái—1 tập của các tình huống mà đặc trưng cho hành vi của 1 hệ thống tại một thời điểm nhất định
- Chuyển đổi trạng thái—sự di chuyển từ một trạng thái tới trạng thái khác.
- Sự kiện—Một biến cố mà nhờ đó hệ thống có thể dự đoán được hành vi
- Hành động—tiến trình mà xuất hiện như một chuỗi các biến đổi

Mô hình hành vi

- Tạo ra một danh sách của các trạng thái khác nhau của hệ thống (làm thế nào để hệ thống xử lý các hành vi?)
- chỉ ra cách hệ thống làm cho một trạng thái chuyển đổi từ một trạng thái khác (Làm thế nào hệ thống chuyển đổi trạng thái?)
  - Chỉ ra ssuwj kiện
  - Chỉ ra hành động
- Vẽ một biểu dồ trạng thái hoặc một biểu đồ trình tự

Biểu đồ trình tự

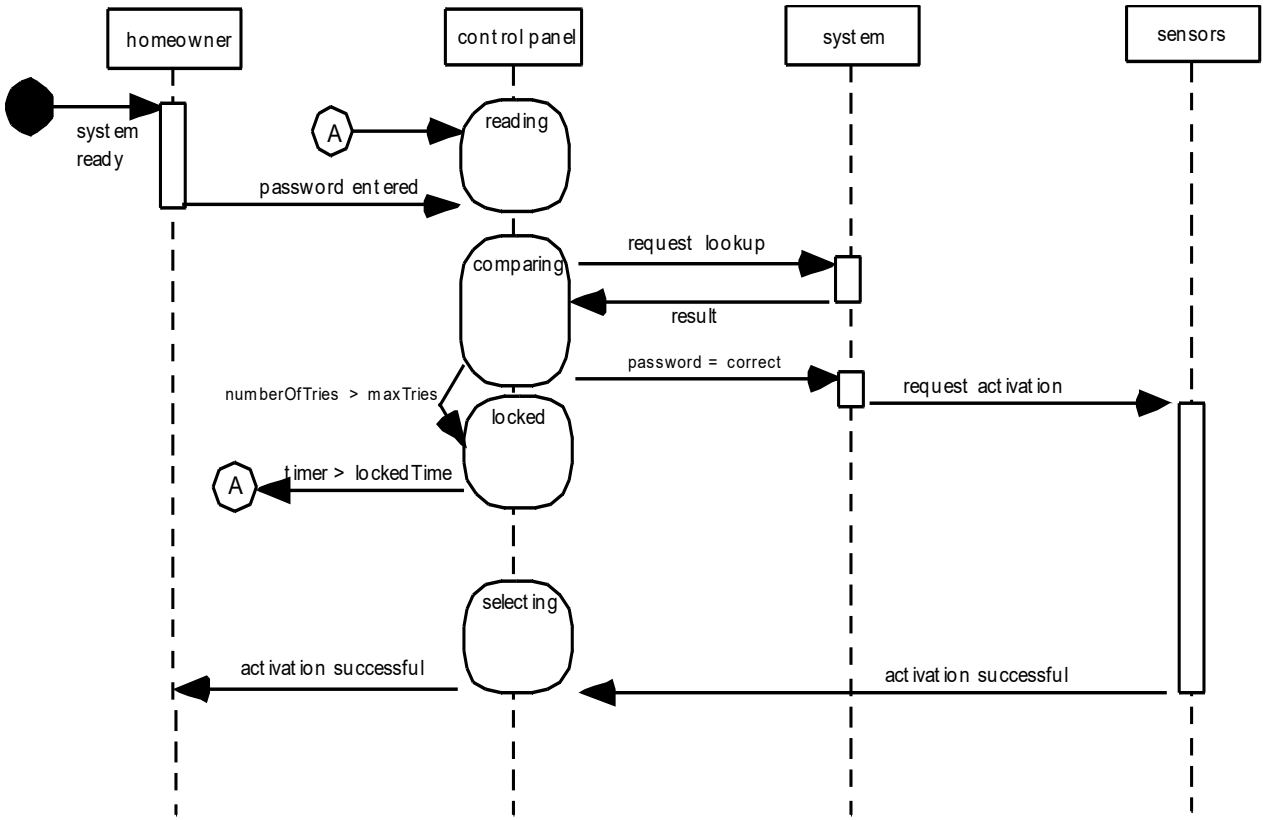


Figure 8.27 Sequence diagram (partial) for *SafeHome* security function

Viết các thông số phần mềm

Mô hình cho mô hình các yêu cầu

- Mô hình phần mềm là một cơ chế để thu nhận tri thức mà cho phép nó được áp dụng lại khi xuất hiện một vấn đề mới
  - Miền kiến thức có thể được áp dụng cho một vấn đề mới trong cùng miền ứng dụng .
  - các miền kiến thức được lư lại bởi một mô hình có thể được áp dụng bằng cách tương tự vào một miền ứng dụng khác mà hoàn toàn khác nhau.
- Ban đầu, các tác giả của một mô hình phân tích không "tạo ra" các mô hình, nhưng sau đó, họ phát hiện ra đó là công việc yêu cầu kỹ thuật đang được tiến hành.
- Khi mô hình đã được khảo sát, nó đã được chứng thực

Khảo sát mô hình phân tích

- Các yếu tố cơ bản nhất trong mô tả của một mô hình yêu cầu là các use-case.
- Một tập hợp chặt chẽ các use-case có thể phục vụ là cơ sở cho việc phát hiện một hoặc nhiều các mẫu phân tích thêm.
- Một mô hình phân tích ngữ nghĩa (SAP) "là một mô hình mô tả một tập hợp nhỏ các use-case mạch lạc theo cùng mô tả một ứng dụng cơ bản." [Fer00]

Một ví dụ

- Hãy xem xét các use-case sơ bộ sau đây cho phần mềm cần thiết để kiểm soát và giám sát một máy ảnh thực sự xem và cảm biến khoảng cách cho một chiếc điện thoại tự động :

Use case: Theo dõi chuyển động ngược

Mô tả: Khi chiếc xe được đặt tại số lùi, các phần mềm điều khiển cho phép một nguồn cấp dữ liệu video từ một máy quay phim phía sau được đặt để hiển thị bảng điều khiển. Phần mềm điều khiển tính toán các khoảng cách và định hướng trên màn hình hiển thị để các nhà điều hành xe có thể duy trì định hướng khi xe di chuyển theo chiều ngược lại. Phần mềm điều khiển cũng theo dõi một cảm biến để xác định xem một đối tượng nằm trong 10 feet phía sau của chiếc xe. Nó sẽ tự động dừng chiếc xe nếu các cảm biến khoảng cách chỉ ra một đối tượng trong 3 feet phía sau của xe.

Một ví dụ

- Use-case này với một loạt các chức năng đó sẽ được phân tích và xây dựng (thành một bộ mạch lạc của use-case) trong quá trình thu thập yêu cầu và mô hình hóa.
- Bất kể bao nhiêu xây dựng được thực hiện, use-case(s) đề nghị một SAP-giám sát dựa trên phần mềm đơn giản, chưa áp dụng rộng rãi và kiểm soát các cảm biến và cơ cấu chấp hành trong một hệ thống vật lý.
- Trong trường hợp này, các cảm biến cung cấp thông tin xung quanh khoảng cách và thông tin video. Các "thiết bị truyền động" là hệ thống dừng của xe (gọi nếu một đối tượng là rất gần với xe).
- Nhưng trong trường hợp tổng quát hơn, một mô hình được áp dụng rộng rãi được phát hiện -> (Thiết bị truyền động-cảm biến) Actuator-Sensor

Mô hình Actuator-Sensor—I

Tên mô hình: ***Actuator-Sensor***

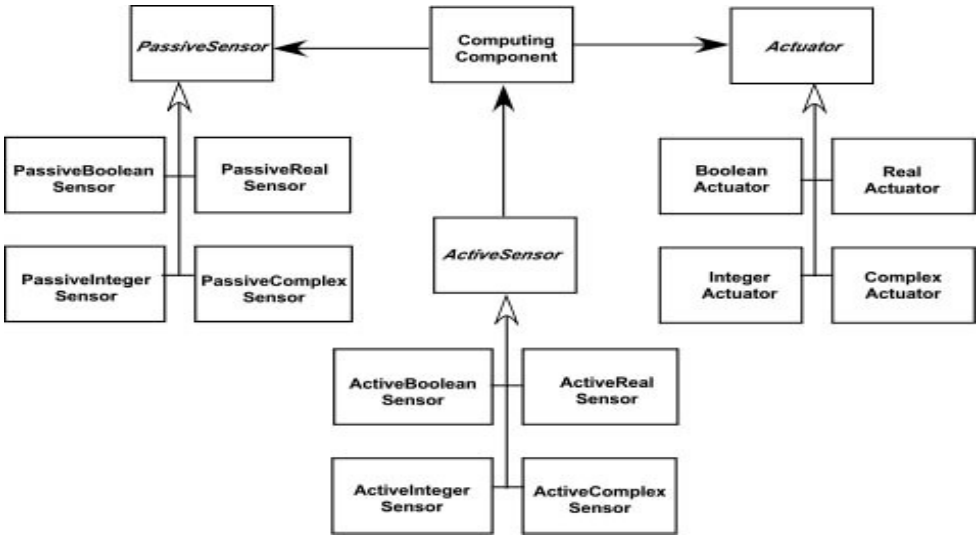
1. Ý định: Xác định các loại cảm biến và cơ cấu chấp hành trong một hệ thống nhúng.
2. Cải tiến: Các hệ thống nhúng thường có các loại cảm biến và thiết bị truyền động. Những cảm biến và thiết bị truyền động là trực tiếp hoặc gián tiếp kết nối với một bộ điều khiển. Mặc dù nhiều trong số các cảm biến và cơ cấu chấp hành trông khá khác nhau, hành vi của chúng là giống nhau đủ để cấu trúc chúng thành một mô hình. Các mô hình chỉ ra làm thế nào để xác định các cảm biến và thiết bị truyền động cho một hệ thống, bao gồm cả các thuộc tính và các hoạt động. Các mô hình Actuator-Sensor sử dụng một cơ chế kéo (yêu cầu rõ ràng cho thông tin) cho PassiveSensors và một cơ chế đẩy (phát sóng thông tin) cho ActiveSensors.

Hạn chế:

1. Mỗi cảm biến thụ động phải có một số phương pháp để đọc đầu vào và các thuộc tính đại diện cho các giá trị cảm biến.
2. Mỗi bộ cảm biến hoạt động phải có khả năng phát sóng thông điệp cập nhật khi thay đổi giá trị của nó.
3. Mỗi bộ cảm biến hoạt động phải gửi một life tick, một thông báo trạng thái ban hành trong một khung thời gian quy định, để phát hiện trục trặc.
4. Mỗi thiết bị truyền động phải có một số phương pháp để gọi các phản ứng thích hợp được xác định bởi các thành phần máy tính(Computing Component).
5. Mỗi bộ cảm biến và thiết bị truyền động nên có một chức năng thực hiện để kiểm tra tình trạng hoạt động của riêng mình.
6. Mỗi bộ cảm biến và thiết bị truyền động cần có thể kiểm tra tính hợp lệ của các giá trị nhận được hoặc được gửi đi và thiết lập trạng thái hoạt động của nó nếu các giá trị là ở bên ngoài của các thông số kỹ thuật.

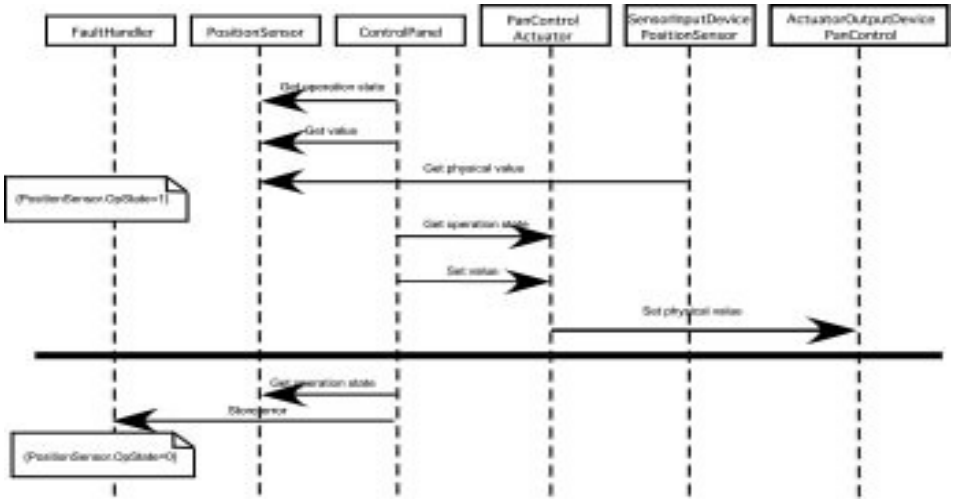
Actuator-Sensor Pattern—II

1. **Khả năng áp dụng**: Hữu ích trong bất kỳ hệ thống mà trong đó nhiều cảm biến và thiết bị truyền động có mặt.
2. **Kiến trúc**: Một biểu đồ UML lớp cho mô hình *Actuator-Sensor* ở hình 7.8. Actuator, PassiveSensor và ActiveSensor là lớp trừu tượng và ký hiệu bằng chữ in nghiêng. Có bốn loại khác nhau của cảm biến và thiết bị truyền động trong mô hình này. Các Boolean, số nguyên, và các lớp thực sự đại diện cho những loại phổ biến nhất của cảm biến và thiết bị truyền động. Các lớp phức tạp là những cảm biến hoặc thiết bị truyền động mà sử dụng các giá trị mà không thể dễ dàng đại diện trong các đặc trưng của các kiểu dữ liệu nguyên thủy, chẳng hạn như một thiết bị radar. Tuy nhiên, các thiết bị này vẫn cần kế thừa giao diện từ các lớp trừu tượng vì chúng cần có các chức năng cơ bản như truy vấn trạng thái hoạt động.



Actuator-Sensor Pattern—III

Hành vi: Hình 7.9 trình bày một sơ đồ trình tự UML cho một ví dụ về các mô hình *Actuator-Sensor* vì nó có thể được áp dụng cho các chức năng **SafeHome** điều khiển vị trí (ví dụ, pan, thu phóng) của một camera an ninh. Ở đây, các **ControlPanel** truy vấn một bộ cảm biến (cảm biến vị trí thụ động) và một thiết bị truyền động (điều khiển pan) để kiểm tra trạng thái hoạt động cho mục đích chẩn đoán trước khi đọc hoặc thiết lập một giá trị. Các thông điệp **đặt Giá trị vật lý và Nhận Giá trị vật lý** không phải là thông điệp giữa các đối tượng. Thay vào đó, chúng mô tả sự tương tác giữa các thiết bị vật lý của hệ thống với các đối tác phần mềm của chúng. Trong phần dưới của biểu đồ, dưới đường ngang, các **PositionSensor(vị trí cảm biên)** chỉ ra trạng thái hoạt động là số không. Các **ComputingComponent (thành phần tính toán)**sau đó gửi mã lỗi cho cảm biến lỗi vị trí để **các FaultHandler(điều khiển lỗi)** đó sẽ quyết định báo lỗi này ảnh hưởng đến hệ thống như thế nào và những hành động được yêu cầu. nó nhậnđược các dữ liệu từ các cảm biến và tính toán đáp ứng yêu cầu cho các thiết bị truyền động



Actuator-Sensor Pattern—III

- Xem SEPA, 7/e cho các thông tin sau:
  - Các thành phần
  - Các liên kết
  - Các kết quả

Requirements Modeling for WebApps

- Phân tích nội dung. Các phổ đầy đủ các nội dung được cung cấp bởi các WebApp(ứng dụng Web) được xác định, bao gồm cả văn bản, đồ họa và hình ảnh, video, âm thanh và dữ liệu. Mô hình hóa dữ liệu có thể được sử dụng để xác định và mô tả một trong các dữ liệu đối tượng.
- Phân tích Tương tác. Cách thức mà người dùng tương tác với các WebApp được mô tả chi tiết. Biểu đồ ca sử dụng(Use-Case) có thể được phát triển để cung cấp mô tả chi tiết của tương tác này.
- Phân tích chức năng. Các use-case được tạo ra như một phần của phân tích tương tác xác định các hoạt động mà sẽ được áp dụng cho nội dung WebApp và bao hàm các chức năng xử lý khác. Tất cả các hoạt động và chức năng được mô tả chi tiết.
- Phân tích cấu hình. Môi trường và cơ sở hạ tầng, trong đó vị trí WebApp được mô tả chi tiết.

Khi nào chúng ta tiến hành phân tích?

- Trong một số tình huống WebE, phân tích và thiết kế hợp nhất. Tuy nhiên, một phân tích hoạt động rõ ràng xuất hiện khi…
  - các WebApp sẽ được xây dựng nhiều và/hoặc phức tạp.
  - số lượng các bên liên quan là lớn
  - số lượng kỹ sư Web và đóng góp khác quá lớn
  - các mục tiêu và đối tượng (được xác định trong quá trình lập) cho các WebApp sẽ ảnh hưởng đến “kinh doanh "
  - sự thành công của các WebApp sẽ có một ảnh hưởng mạnh mẽ đến sự thành công của doanh nghiệp

Mô hình nội dung

- Đối tượng nội dung được lấy từ biểu đồ ca sử dụng
  - kiểm tra các mô tả kịch bản để tham khảo trực tiếp và gián tiếp đến nội dung
- Các trạng thái của một đối tượng nội dung được định nghĩa
- Các mối quan hệ giữa các đối tượng nội dung và/hoặc, hệ thống phân cấp của nội dung được duy trì bởi một WebApp
  - Sơ đồ mối quan hệ thực thể-mối quan hệ hoặc UML
  - Cây phân cấp dữ liệu hoặc UML

Cây dữ liệu

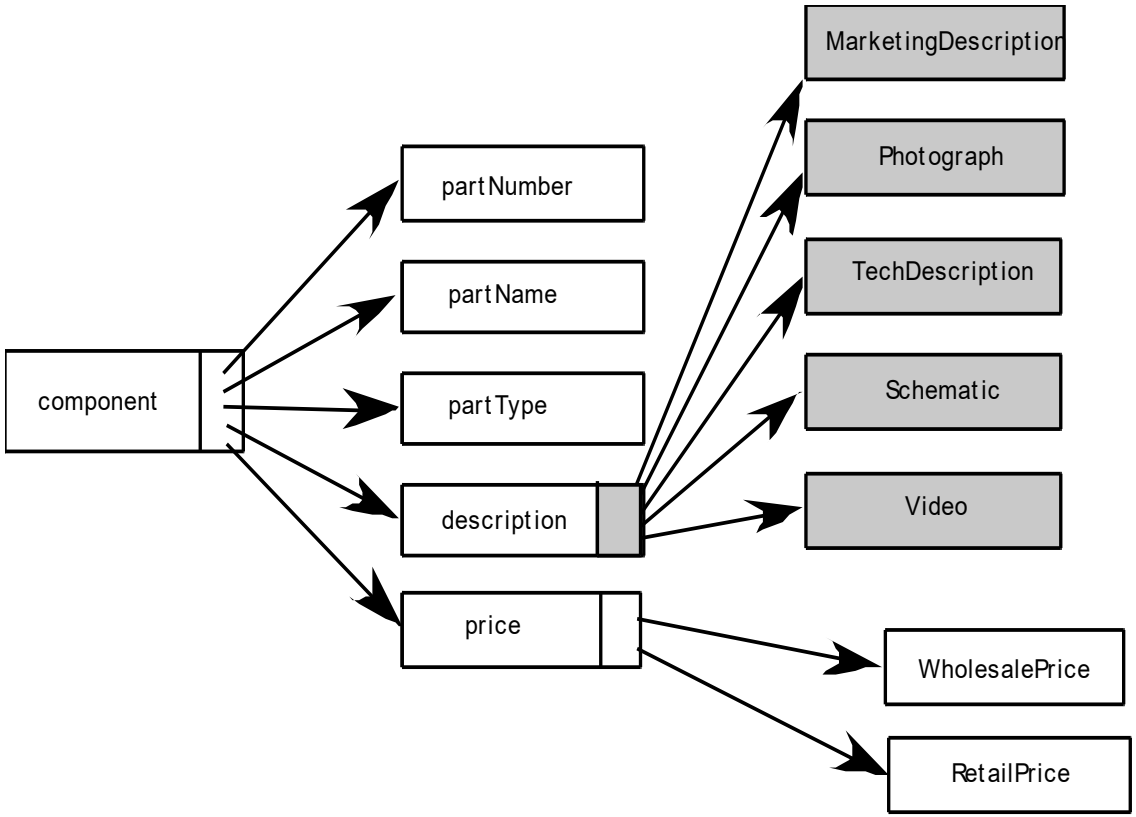


Figure 18.3 Data tree for a *SafeHome* component

Mô hình tương tác

- Bốn nguyên tố cấu thành:
  - use-cases(biểu đồ ca sử dụng)
  - sequence diagrams(biểu đồ trình tự)
  - state diagrams(biểu đồ trạng th
  - a user interface prototype(giao diện người dùng)
- Mỗi trong số này là một nguyên tố UML quan trọng và được mô tả trong Phụ lục I

Biểu đồ trình tự

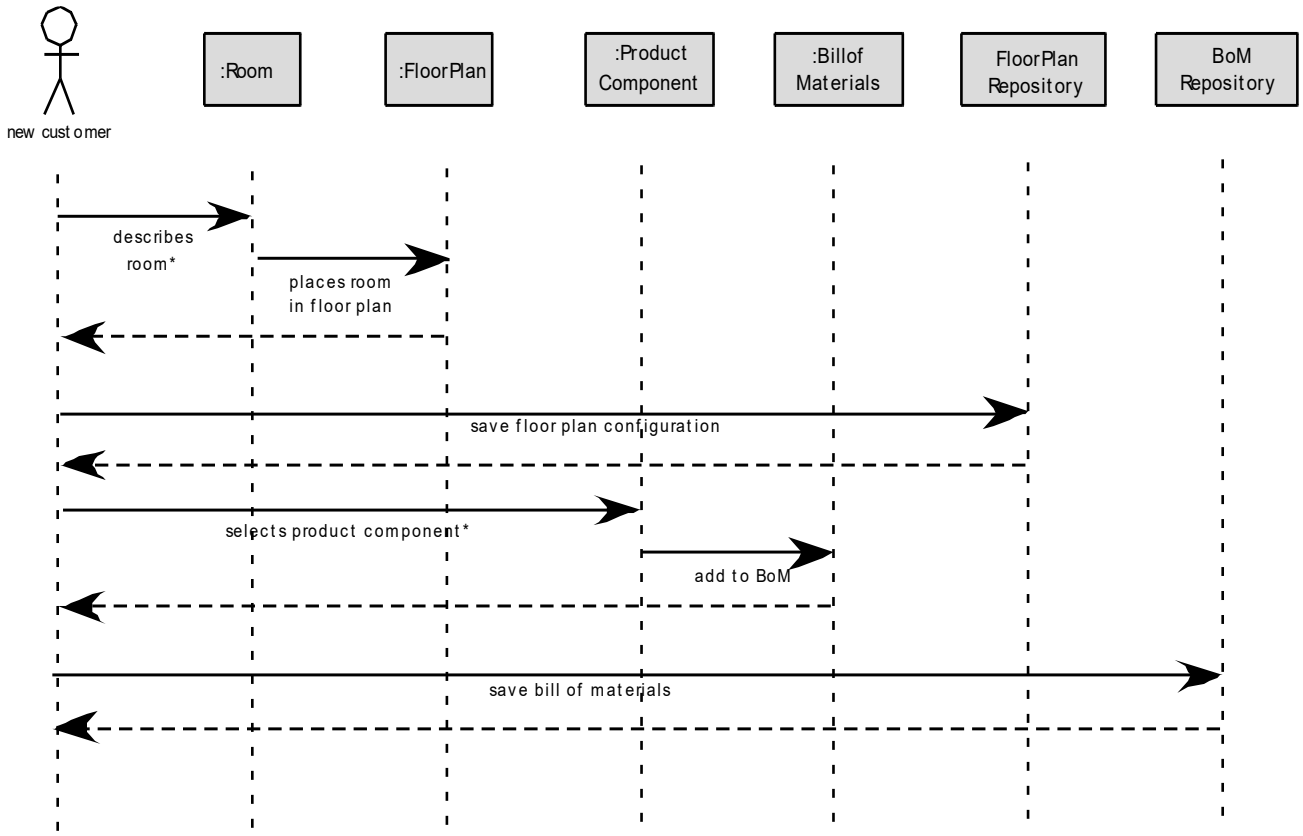


Figure 18.5 Sequence diagram for use-case: *select SafeHome components*

Biểu đồ trạng thái

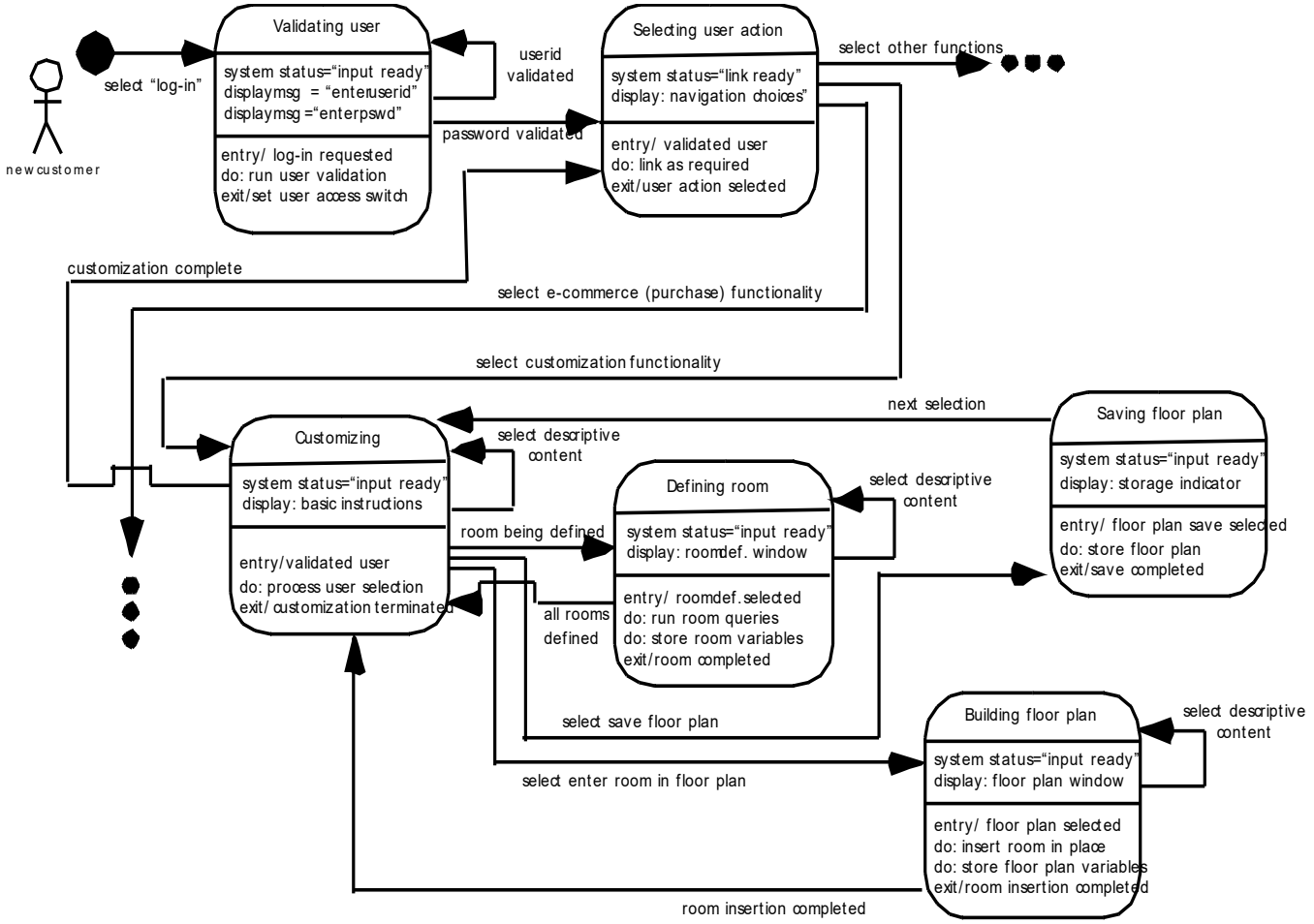


Figure 18.6 Partial state diagram for **new customer** interaction

Mô hình chức năng

- Các mô hình chức năng giải quyết hai yếu tố xử lý của WebApp
  - Khảo sát chức năng người dùng được phân phối bởi các WebApp cho người dùng cuối
  - các hàm trong lớp thực hiện hành vi kết hợp với lớp.
- Một biểu đồ hoạt động có thể được sử dụng để biểu diễn luồng tiến trình

Biểu đồ hoạt động

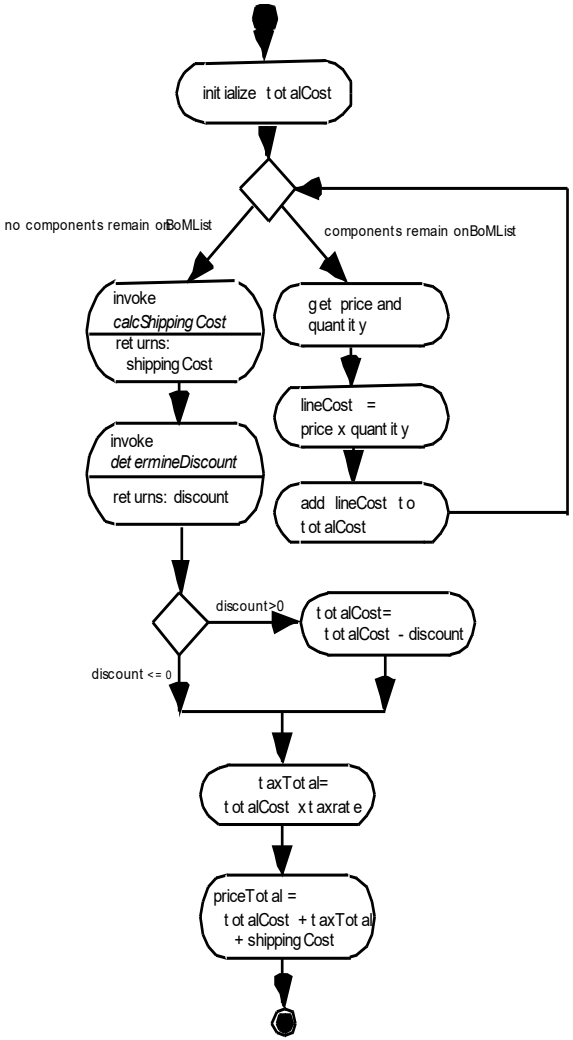


Figure 18.7 Activity diagram for ***computePrice()*** operatio

Mô hình cấu hình

- Máy chủ
  - Phần cứng máy chủ và môi trường hệ điều hành phải được xác định
  - Cân nhắc khả năng tương tác trên phía máy chủ phải được xem xét
  - Giao diện thích hợp, giao thức truyền thông và thông tin hợp tác liên quan phải được quy định
- Khách hàng(client)
  - Vấn đề cấu hình trình duyệt phải được xác định
  - Yêu cầu thử nghiệm cần được xác định

Mô hình điều hướng

- Một số yếu tố nên được dễ dàng để đạt được hơn (yêu cầu các bước chuyển hướng ít hơn)những người khác? Ưu tiên cho các trình diễn là gì?
- Một số yếu tố cần được nhấn mạnh để buộc người dùng điều hướng theo hướng của họ?
- Làm thế nào lỗi điều hướng cần được xử lý?
- Chuyển hướng tới các nhóm có liên quan cần được ưu tiên hơn hướng đến một yếu tố cụ thể.
- Chuyển hướng nên được thực hiện thông qua các liên kết, thông qua truy cập dựa trên tìm kiếm, hoặc bằng một số phương tiện khác?
- Các yếu tố có nhất định nên giới thiệu đến người dùng dựa vào bối cảnh của hành động chuyển hướng trước?
- Một bản ghi chuyển hướng nên được duy trì cho người sử dụng?

Mô hình điều hướng

- Nên một bản đồ chuyển hướng đầy đủ hoặc thực đơn (như trái ngược với một “back" liên kết duy nhất hoặc con trỏ) có sẵn ở mọi điểm trong sự tương tác của người dùng?
- Thiết kế chuyển hướng nên được thúc đẩy bởi các hành vi sử dụng dự kiến phổ biến nhất hoặc bởi tầm quan trọng của các yếu tố nhận thức WebApp ?
- Có thể sử dụng một “lưu trữ" chuyển hướng trước đó thông qua các WebApp để tiến hành sử dụng trong tương lai?
- Nhóm người dùng điều hướng tối ưu nên được thiết kế?
- Làm thế nào kết nối các đường link bên ngoài để các WebApp được xử lý? Chồng lên các cửa sổ trình duyệt hiện hành? Mở một cửa sổ trình duyệt mới? Mở một khung riêng biệt?

Tài liệu tham khảo

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