CHƯƠNG 5

PHÂN TÍCH THIẾTKẾ HƯỚNG ĐỐI TƯỢNG

Nội dung

- UML là gì?
- Sơ lược lịch sử phát triển của UML
- Các khung nhìn của UML
- Lược đồ của UML 2.0
- Case study : Hệ thống POS
- Mô hình Use-Case

UML

- UML- Unified modeling language
  (ngôn ngữ mô hình hóa thống nhất)
- UML là một ngôn ngữ mô hình (modeling language)
  - Vocabulary: phần tử hình ảnh
  - Grammar: quy tắc kết nối các phần tử

➔biểu diễn ý niệm và vật lý của một hệ thống

- Dùng UML để tạo và đọc các mô hình nhưng không thể cho biết tạo mô hình gì và khi nào thì tạo chúng

UML - Unified modeling language

- UML dùng để:
  - Hình tượng hóa (Visualizing)
  - Đặc tả (Specifying)
  - Xây dựng (Constructing)
  - Lưu trữ (Documenting)

UML là ngôn ngữ dùng để hình ảnh hóa

- Nó giúp các developer mô tả các ý tưởng, dễ dàng đọc được mô hình xây dựng bằng UML do một người khác viết
- Những cấu trúc mà việc nắm bắt thông qua đọc mã lệnh là khó khăn nay đã được thể hiện trực quan

UML là ngôn ngữ dùng để đặc tả

- UML có thể đặc tả tất cả các quyết định quan trọng trong phân tích, thiết kế và thực thi một hệ thống phần mềm

UML là ngôn ngữ dùng để xây dựng

- Các mô hình xây dựng bởi UML có thể ánh xạ tới một ngôn ngữ lập trình cụ thể như : Java, C++, VB... thậm chí cả các bảng trong một CSDL quan hệ hay CSDL hướng đối tượng

UML là ngôn ngữ dùng để lưu trữ tài liệu

- Dùng để ghi chép về:
  - Các yêu cầu của hệ thống
  - Kiến trúc của hệ thống
  - Thiết kế
  - Mã nguồn
  - Kế hoạch dự án
  - Tests
  - Các nguyên mẫu

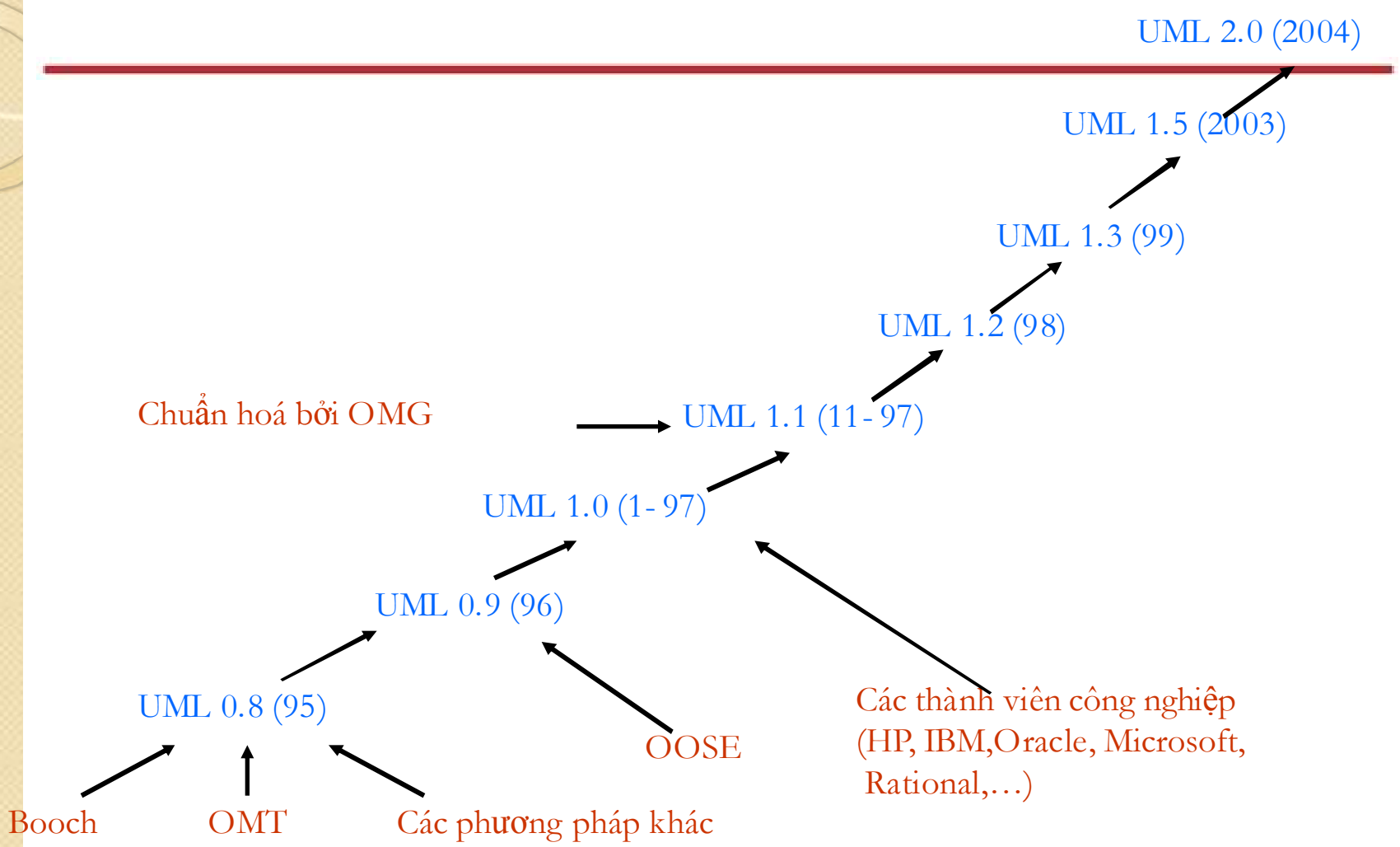
Lịch sử phát triển của UML

- Ngôn ngữ hướng đối tượng đầu tiên là Simula-67 (1967)
- 1967 – 1994: dư thừa quá nhiều phương pháp luận hướng đối tượng

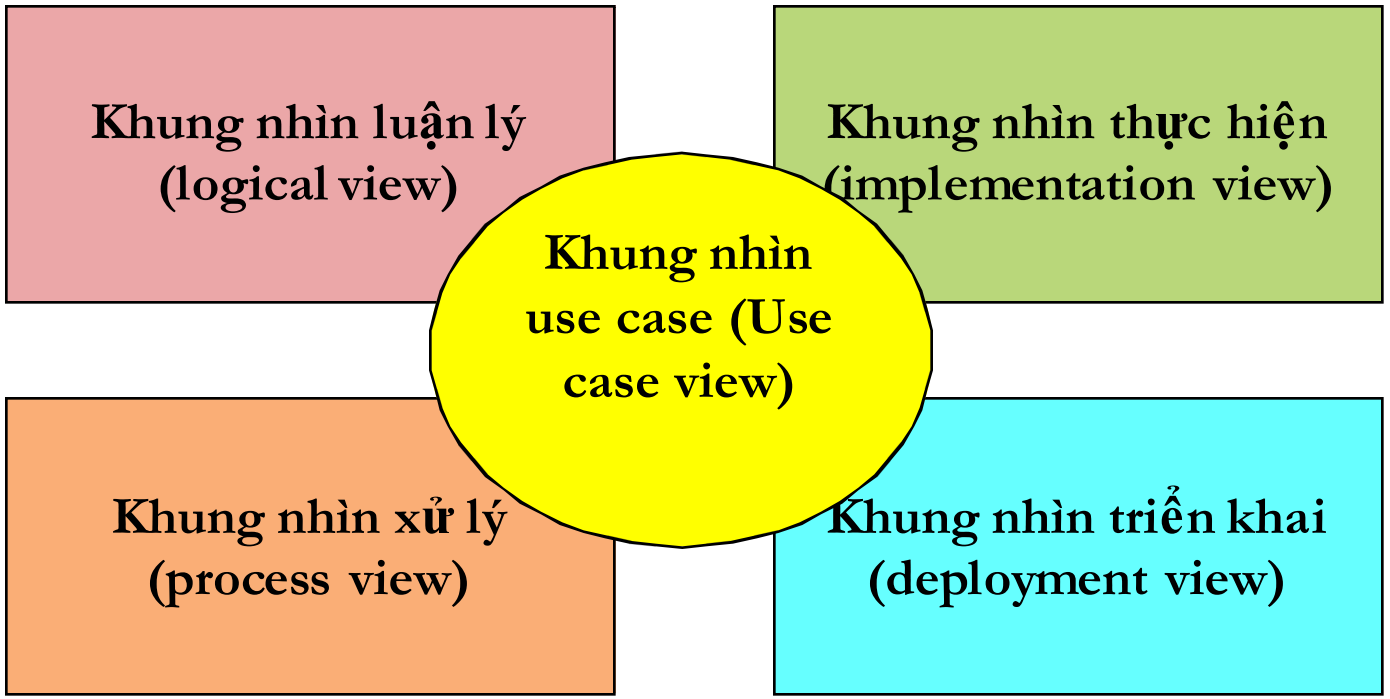
➔ UML được phát triển với nổ lực làm đơn giản và hợp nhất các phương pháp

- Phương pháp Booch + phương pháp OMT → UP (Unified Process) (1994)
- Jacobson đã nỗ lực tích hợp phương pháp UP + OOSE → UML đầu tiên (1996)
- UML 1.0 công bố (1/1997)
- UML 2.0 công bố (2004)

Lịch sử phát triển của UML



Các khung nhìn (view) của UML



Use-Case View

- Chứa các use case mô tả hành vi của hệ thống dưới góc nhìn của người dùng cuối, nhà phân tích hay người kiểm thử hệ thống.
- Không xét tổ chức bên trong của phần mềm, mà chỉ làm rõ các chức năng chính của hệ thống
- Dạng tĩnh:
  - Use Case diagrams
- Dạng động:
  - Activity diagrams
  - Sequence diagrams
  - Collaboration diagrams
- Khi bắt đầu dự án, lược đồ use case được dùng để thống nhất hệ thống giữa khách hàng và nhà phát triển hệ thống

Logical View (hay design view)

- Hỗ trợ cho các yêu cầu chức năng của hệ thống dưới dạng các dịch vụ (service) mà hệ thống cung cấp cho người dùng cuối.
- Để tạo khung nhìn thiết kế thường theo hai bước.
  - Bước 1: nhận ra các lớp phân tích (analysis class) độc lập với ngôn ngữ lập trình
  - Bước 2: chuyển các lớp phân tích thành các lớp thiết kế (design class) phụ thuộc theo ngôn ngữ.

Process View

- Chia hệ thống thành các tiến trình (process) và luồng(thread), mô tả việc đồng bộ hóa và các xử lý đồng thời.
- Dành cho việc thực thi hệ thống

Implementation View & Deployment View

- **Implementation View**: Bao gồm các component và file tạo nên hệ thống vật lý. Biểu đồ được sử dụng là component diagram
- **Deployment View:** Chỉ ra cấu hình phần cứng mà hệ thống sẽ chạy trên đó. Nó thể hiện sự phân tán, cài đặt các phần mà tạo nên kiến trúc vật lý của hệ thống. Biểu đồ được sử dụng là Deployment diagram.

Các loại sơ đồ (diagram) khác nhau của UML

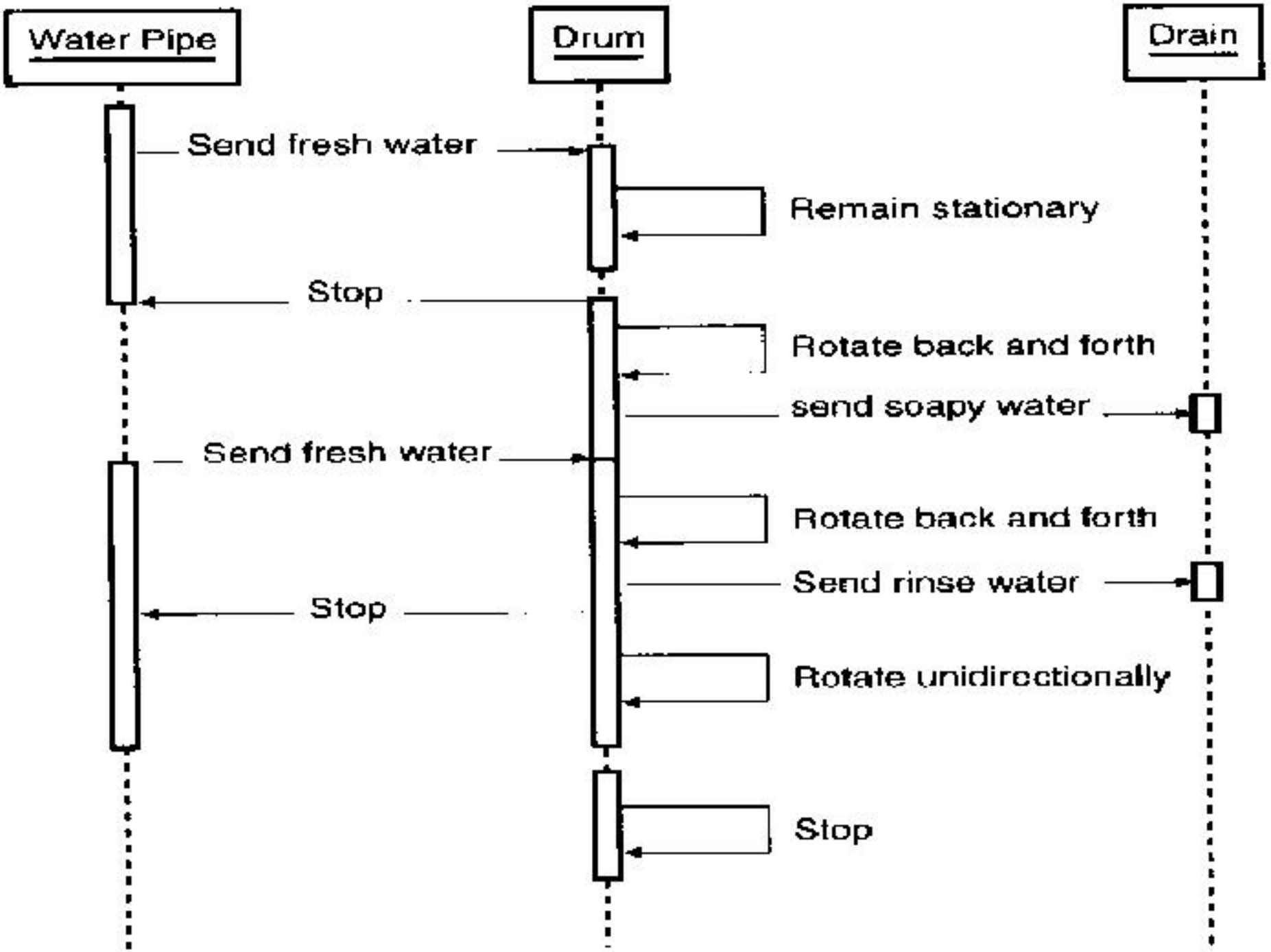
- **Sơ đồ trạng thái (State Diagram)**
  - **Tại thời điểm nào thì một object cũng phải có một trạng thái xác định. Một máy giặt có thể đang ngâm (soak), giặt (wash), giũ (rinse), vắt (spin) hoặc không hoạt động.**

Các loại sơ đồ (diagram) khác nhau của UML

- **Sơ đồ trình tự (Sequence Diagram)**
  - Sơ đồ trình tự của UML (sequence diagram) biểu diễn sự tương tác theo thời gian.
  - Các thành phần của máy gồm một ống cấp nước (cấp nước sạch), một trống giặt (chứa áo quần) và một ống thoát. Chúng dĩ nhiên cũng là các object. Điều gì xảy ra khi ta kích hoạt use case “wash clothes”?

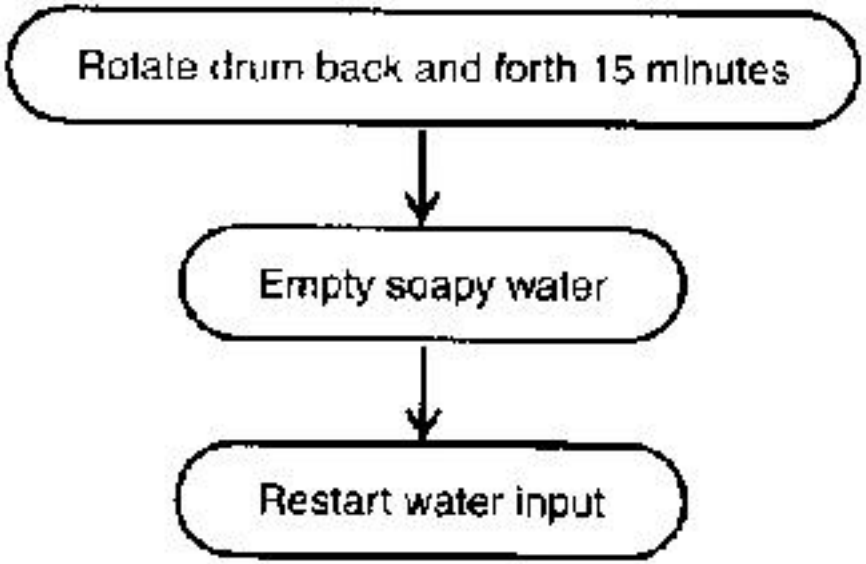
các bước tiến hành

1. Nước vào trống thông qua ống cấp
2. Trống đứng yên trong 5 phút
3. Nước ngừng cấp
4. Trống quay ngược và xuôi trong 15 phút
5. Nước xà phòng thoát ra thông qua ống thoát
6. Nước sạch được cấp lại
7. Trống tiếp tục quay ngược và xuôi
8. Nước ngừng cấp
9. Nước vắt thoát ra thông qua ống thoát
10. Trống quay 1 chiều nhanh dần trong vòng 5 phút
11. Trống ngừng quay và việc giặt hoàn tất.


Water Pipe
Drum
Drain
Send fresh water
Remain stationary
Stop
Rotate back and forth
send soapy water
Send fresh water
Rotate back and forth
Send rinse water
Stop
Rotate unidirectionally
Stop

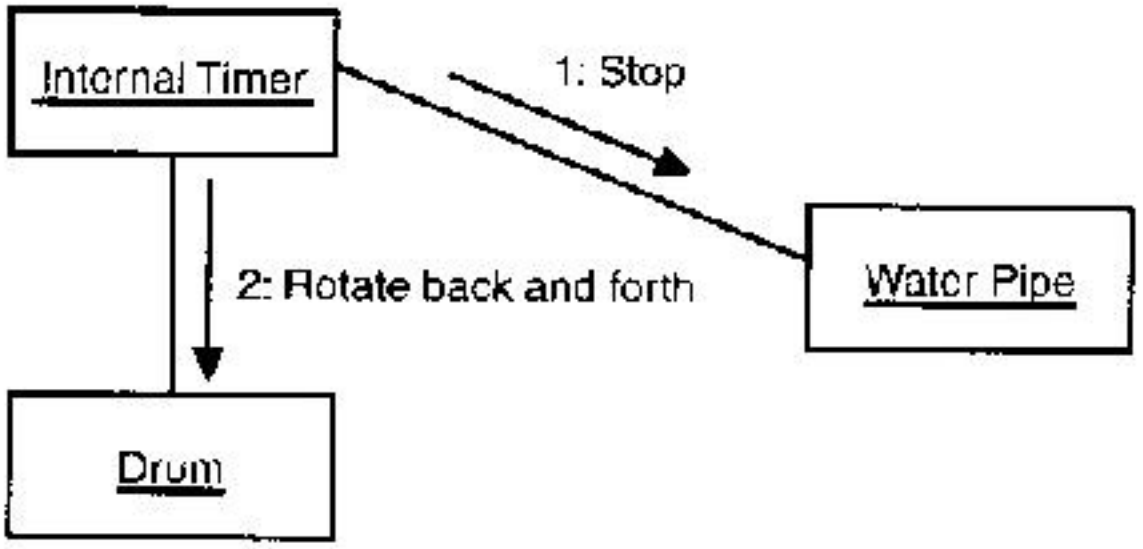
Các loại sơ đồ (diagram) khác nhau của UML

- **Sơ đồ hoạt động (Activity Diagram)**
  - Các hoạt động xuất hiện trong một use case hoặc trong một hành vi của object thường diễn ra theo một trình tự như 11 bước ở ví dụ trên.



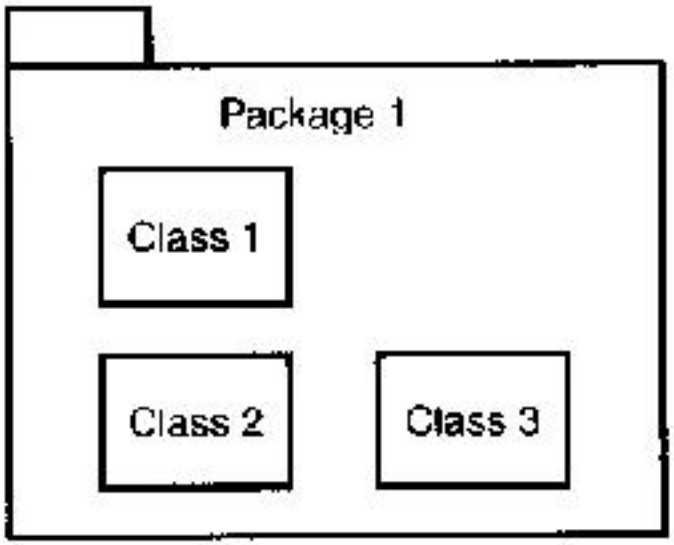
Các loại sơ đồ (diagram) khác nhau của UML

- **Sơ đồ cộng tác (Collaboration Diagram)**
  - Các thành phần của hệ thống làm việc với nhau để hoàn thành mục tiêu đặt ra và một ngôn ngữ mô hình hóa cần có cách biểu diễn sự hợp tác này



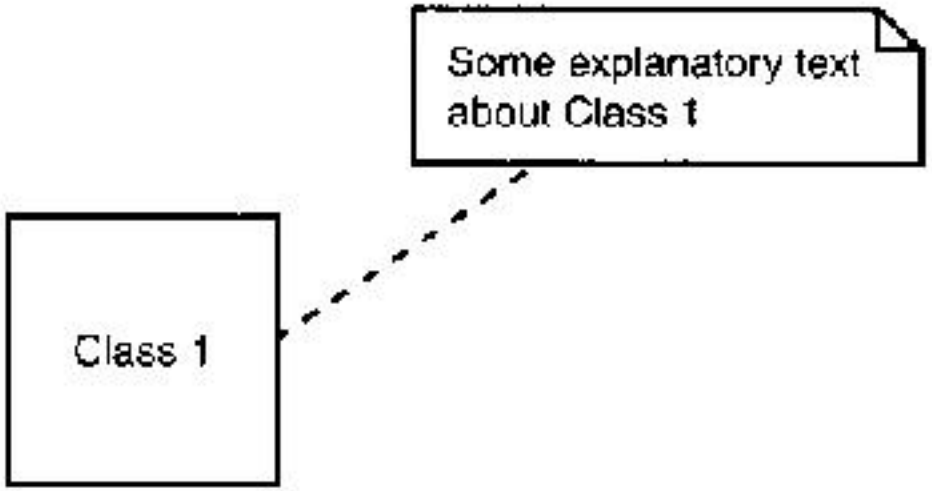
Một số tính năng khác

- **Gói (package)**
  - Khi cần tổ chức các thành phần của một diagram vào trong một nhóm (group), ta đưa chúng vào trong một ***package***



Một số tính năng khác

- **Ghi chú (Note)**
  - **Thuật ngữ**: Ghi chú (***note***) của UML giúp tránh một thành phần nào đó trong diagram bị hiểu nhầm.



Case study 1: Hệ thống POS

- Hệ thống POS (Point-Of-Sale) là một ứng dụng máy tính hóa được dùng để lưu trữ lại hồ sơ bán hàng và quản lý việc thanh toán. Hệ thống được dùng cho các cửa hàng bán lẻ.
- Yêu cầu phần cứng chỉ gồm máy tính và máy quét mã vạch (bar code scanner).
- Phần mềm có thể giao kết được với các ứng dụng khác như tính thuế, quản lý kho,... Hệ thống cũng cần có khả năng hoạt động ngay cả khi có lỗi kết nối với các dịch vụ khác chẳng hạn như khi hệ thống quản lý kho hay dịch vụ thanh toán từ xa tạm thời không kết nối được thì hệ thống POS vẫn có thể quản lý việc bán hàng và thanh toán bằng tiền mặt.

Một số khái niệm mở rộng trong UML

- Stereotypes
- Tagged values

Stereotype

- Stereotype dùng để xác định một loại phần tử mới dựa vào một phần tử hiện có. Stereotype giống như phần tử cũ nhưng có thêm 1 số ngữ nghĩa khác.
- Stereotype có thể được tạo ra từ tất cả các phần tử cơ bản của UML: class, node, component, packages,…
- Một số stereotype được định nghĩa sẵn giúp cho UML đơn giản

Stereotype (tt)

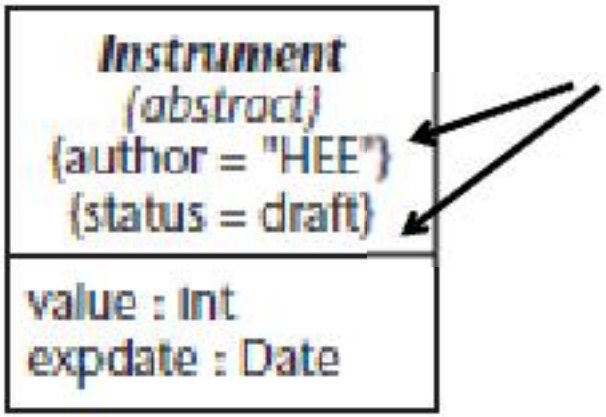
- Ký hiệu của Stereotype:

  <<StereotypeName>>

- Ví dụ: một hệ thống thường có rất nhiều class khác nhau thuộc các loại khác nhau, dùng stereotype để phân loại các class này. UML định nghĩa sẵn 1 số stereotype cho class như sau: <<artifact>>, <<window>>, <<source>>…

Tagged values

- Là bất kỳ loại thông tin nào mà người dùng muốn gắn vào phần tử.
- Được thực hiện bằng cách tạo thuộc tính mới cho phần tử và gán giá trị cho thuộc tính đó.



Hai tagged values là author và status

Bài tập

Tìm hiểu hai khái niệm:

- Artifact
- Stakeholder

mô hình Use-Case

Nội dung

- Yêu cầu hệ thống
- Mô tả use case
  - Actor
  - Scenario
  - Use case
- Lược đồ use case
- Lược đồ gói

Thu thập yêu cầu (Requirement gathering)

- Khách hàng và người dùng cuối thường có các mục tiêu (goal) và muốn hệ thống máy tính giúp họ hoàn thành mục tiêu này.
- Use case là cơ chế giúp diễn đạt các mục tiêu này đơn giản và dễ hiểu.
- Các bước trong công đoạn:
  1. Thu thập yêu cầu của người dùng,
  2. Use case là cơ chế giúp diễn đạt yêu cầu
  3. Tạo mô hình use case

Use case "Process Sales" **(dạng đơn giản)**

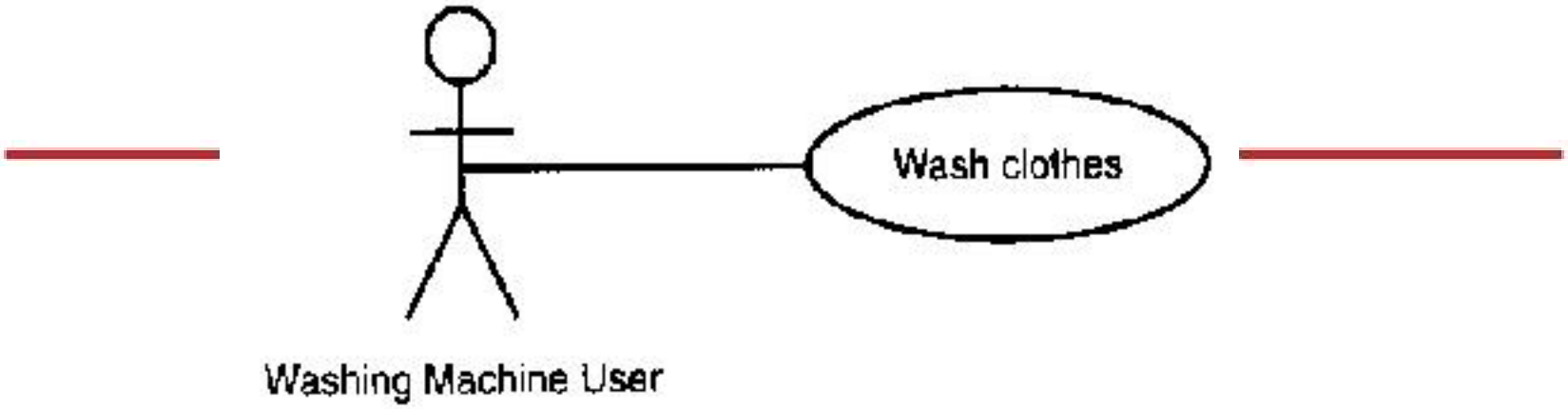
- Khách hàng (customer) đến quầy tính tiền với các hàng hóa (item) đã chọn mua. Thu ngân (cashier) sử dụng hệ thống POS để nhập các mặt hàng đã mua. Hệ thống sẽ đưa ra tổng thành tiền và chi tiết mỗi mặt hàng được mua. Khách hàng sẽ cung cấp thông tin cho việc trả tiền (payment) và hệ thống sẽ kiểm tra tính hợp lệ và ghi nhận lại. Sau đó, hệ thống sẽ cập nhật kho trong khi đó khách hàng nhận hóa đơn (receipt) và ra về cùng với hàng hóa đã mua

Một số khái niệm chính

- **Actor:** là 1 cái gì đó hoạt động như con người, hệ thống máy tính,…
- **Scenario (kịch bản)** là 1 chuỗi các hành động (action) và tương tác (interaction) giữa các actor và hệ thống.
- Scenario còn gọi là use case instance (điển hình của use case).
- Có nhiều cách để xác định scenario nhưng cách đơn giản nhất là dùng lược đồ activity.

Các kịch bản của use case "Process Sales"

- Mua thành công các hàng hóa
- Không mua được hàng do không thanh toán được bằng thẻ tín dụng.



Thuật ngữ: Hình nhân (stick) nhỏ tượng trưng cho người dùng máy giặt được gọi là tác nhân (***actor***). Hình ellipse biểu diễn use case. Chú ý rằng actor–thực thể kích hoạt use case–có thể là con người hoặc một hệ thống khác.

Xác định use case

- Các yêu cầu có thể được nhóm thành nhiều mức. Vậy nên dùng use case ở mức nào và phạm vi nào?
- Xét 3 use case sau:

1. Thỏa thuận hợp đồng với nhà cung cấp
2. Xử lý việc bán hàng.
3. Đăng nhập

Use case nào phù hợp với phạm vi và mục tiêu của hệ thống POS?????

Xử lý nghiệp vụ cơ bản (Elementary business processes - EBP)

- EBP là một nhiệm vụ được thực thi bởi một người nào đó tại 1 vị trí và 1 thời điểm xác định nhằm đáp ứng 1 sự kiện nghiệp vụ và phải cho kết quả là 1 giá trị nghiệp vụ và giữ cho giá trị này trong trạng thái nhất quán

Xác định use case

- Để use case ở mức EBP nên có:
  - Scenario chính chứa từ 5 đến 10 bước, không nên kéo dài nhiều ngày và chứa quá nhiều phần.
  - Chỉ nên là 1 nhiệm vụ, có thể thực thi trong vài phút hoặc vài giờ.
- Thường thì có thể tạo các use case con biểu diễn các nhiệm vụ con (sub-task) trong 1 use case cơ bản

Xác định use case

- "Thỏa thuận hợp đồng với nhà cung cấp": không thể là use case mức EBP vì nó kéo dài nhiều ngày và liên quan đến nhiều thành phần khác.
- "Đăng nhập vào hệ thống" có vẽ gần với mục tiêu người dùng, nhưng nó không cho thêm được 1 giá trị nghiệp vụ.Thu ngân có thể đăng nhập 20 lần/ngày nhưng không phục vụ gì cho việc bán hàng, nên nó chỉ là mục tiêu thứ cấp

Xác định use case

- Chỉ có "xử lý việc bán hàng" phù hợp với chuẩn EBP.
- Các actor đều có mục tiêu (goal) và họ sử dụng CTUD để giúp thỏa mãn mục tiêu. Vì vậy use case ở mức EBP còn được gọi là use case ở mức mục tiêu người dùng (user-goal).

Quy trình xác định actor và use case

1. Xác định phạm vi hệ thống (system boundary)
2. Xác định tác nhân (actor) chính
3. Xác định các mục tiêu của mỗi actor chính ở mức EBP
4. Xác định use case thỏa mãn mục tiêu người dùng (user-goal), đặt tên theo tên mục tiêu. Thường use case sẽ ánh xạ 1-1 với mục tiêu.

Xác định actor

- Actor là 1 ai đó hay 1 cái gì đó tương tác (interact) với hệ thống.
- Tương tác = actor sẽ gửi hay nhận các thông báo từ hệ thống.
- Actor được xem như 1 loại (type) nào đó, không phải là 1 điển hình cụ thể, nó biểu diễn 1 vai trò (role) chứ không nhằm vào một cá nhân nào của hệ thống.

Xác định actor

- Trong hệ thống POS, John là nhân viên, vai trò của anh ta là thu ngân, vai trò của anh ta sẽ được mô hình hóa chứ không phải bản thân anh ta.
- Thực tế một người có thể là nhiều actor khác nhau trong hệ thống phụ thuộc vào vai trò của người đó.
- Ví dụ cũng là John nhưng anh ta có thể là actor “thu ngân”, hay actor “ khách hàng”.

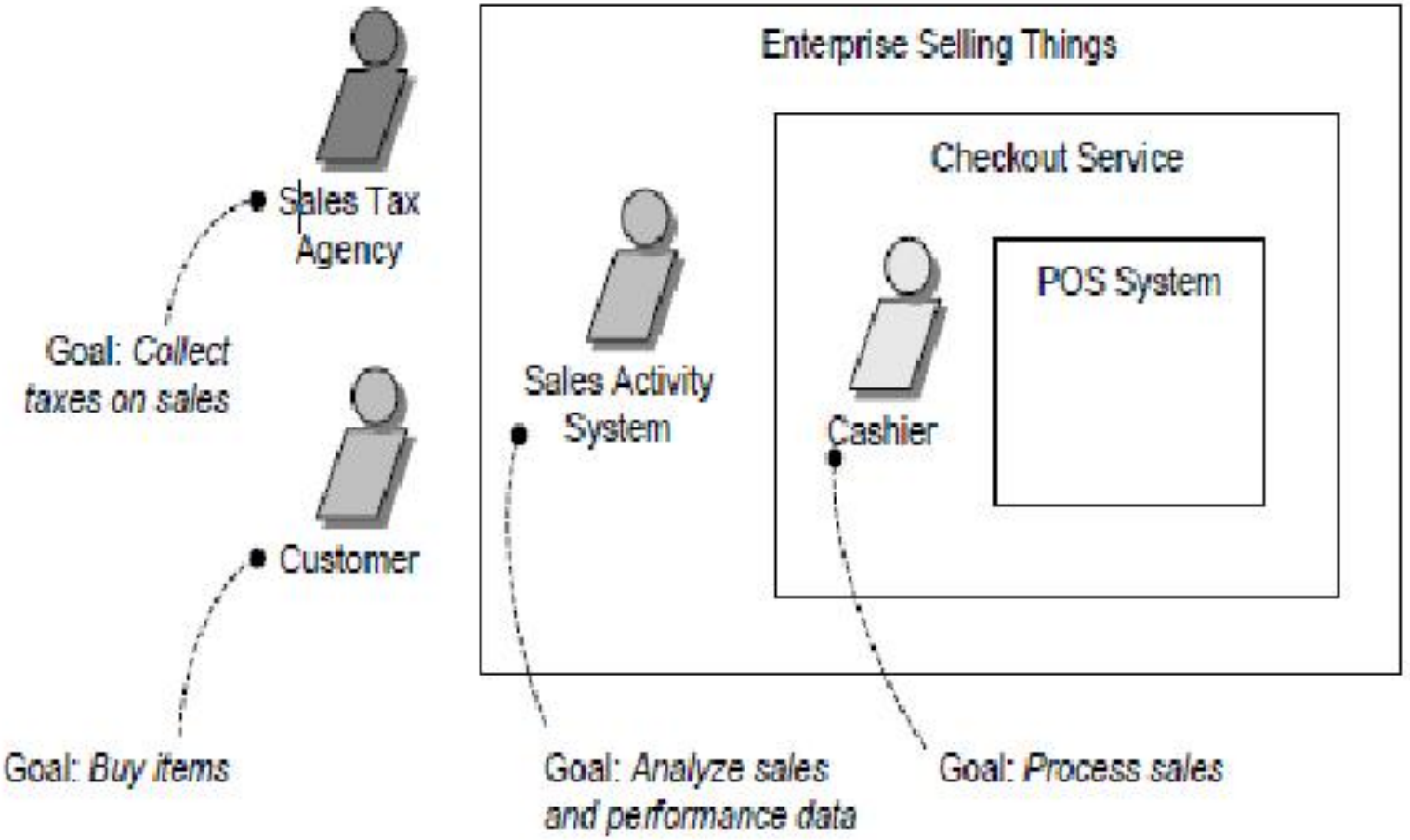
Xác định actor

- Mỗi actor cần có tên, và tên của actor nên phản ánh vai trò (role) của actor đó, không nên phản ánh chức năng của actor đó.
- Ba loại actor chính:
  - User
  - Different systems
  - Time (thời gian)

Actor là thời gian (time)

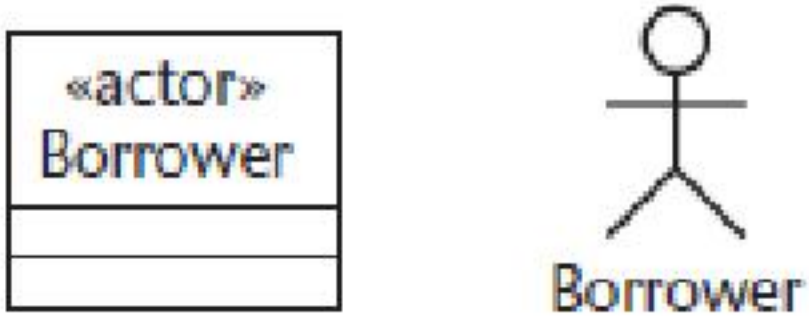
- Thời gian sẽ trở thành actor của hệ thống nếu sau 1 khoảng thời gian nào đó thì nó kích khởi (triger) một số sự kiện (event).
- Ví du:
  - Hệ thống POS, cứ vào 5 giờ chiều ngày thứ bảy thì hệ thống sẽ tự động thống kê tình hình bán hàng trong tuần và in phiếu đặt hàng mới.
  - Hệ thống đặt vé tự động, cứ 3 giờ chiều mỗi ngày, hệ thống sẽ tự động chọn ngẫu nhiên 1 khách hàng để tặng vé khuyến mãi

Actor và mục tiêu



Biểu diễn actor

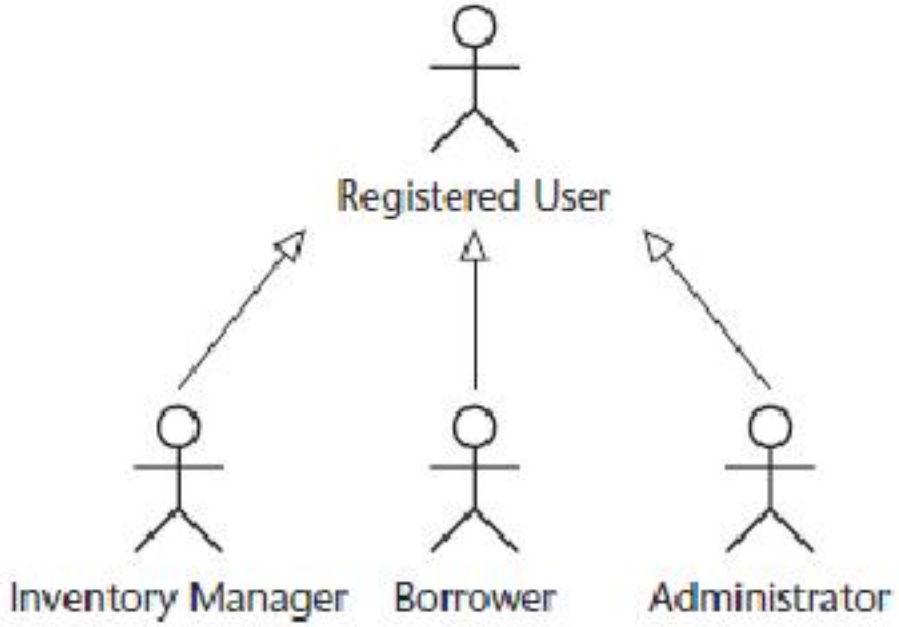
- Được biểu diễn trong lược đồ UML dưới 1 trong 2 dạng sau



- Tên actor thường là một danh từ (noun)

Khái quát hóa (generalization)

- Một actor "con" (child) có thể làm mọi việc mà actor cha (parent) làm và có thể làm thêm 1 số việc khác nữa

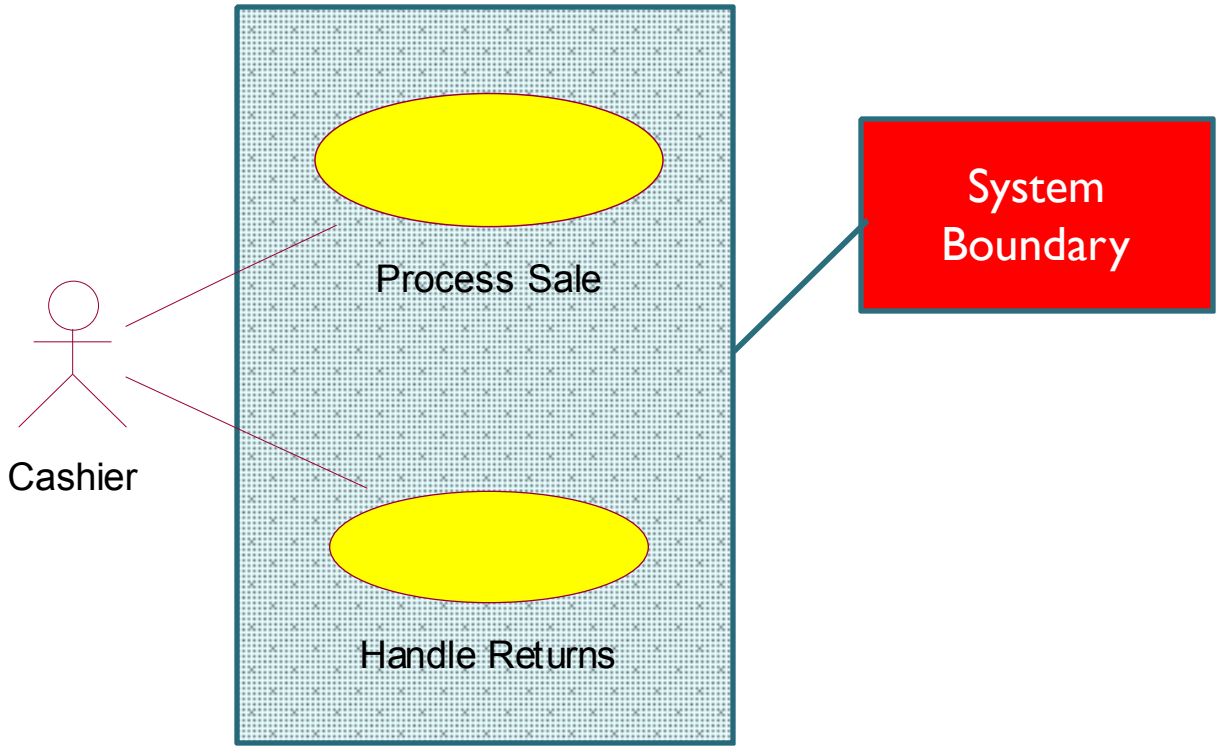


Biểu diễn Use case

- "Một tập hợp các hành động (action) được thực thi bởi hệ thống, để tạo ra một kết quả có giá trị nào đó cho 1 hay nhiều actor hay stakeholder khác của hệ thống"
- Tên use case phải bắt đầu bằng một động từ, thường là 1 phrase

Actor và use case

- Mối liên hệ giữa actor và use case thường là quan hệ hai chiều

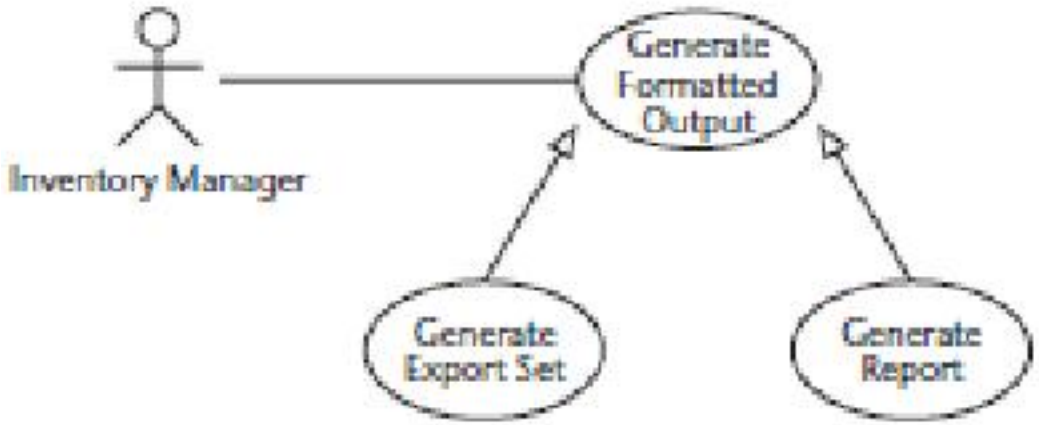


Quan hệ giữa các use case

- Vì mỗi use case biểu diễn một đơn vị đầy đủ, nên giữa các use case sẽ không có sự kết hợp (association) giữa các use case nhưng có mối quan hệ (relationship) giữa chúng và được phân thành 3 loại sau:
  - Extend
  - Include
  - Generalization

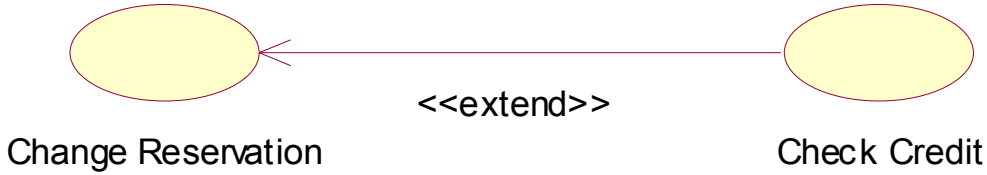
Quan hệ khái quát hóa(Generalization)

- Là mối quan hệ từ use case con đến use case cha, xác định một con có thể chuyên biệt hóa mọi hành vi (behavior) và đặc tính của cha.
-



Quan hệ Extend

- Được dùng trong hai trường hợp sau:
  - Khi hệ thống đang phát triển có nhiều thay đổi cho các hành vi của UC.
  - Khi UC đã triển khai nhưng chưa xác định đầy đủ chức năng cho nó.

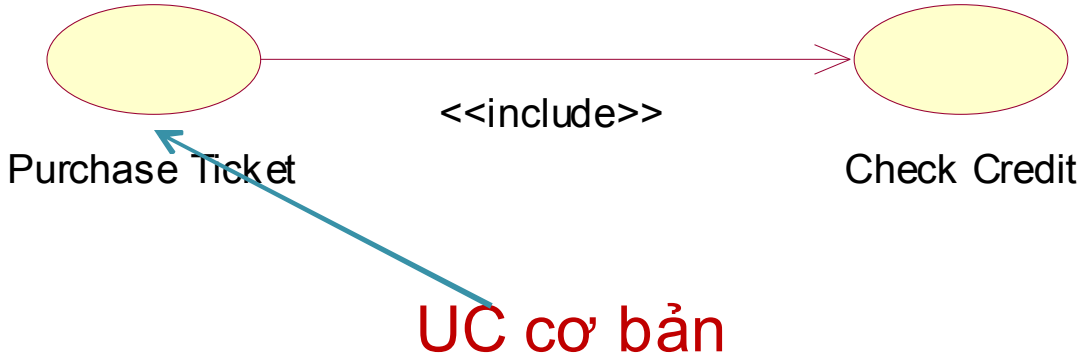


- Khi "Change Reservation" đang vận hành, thì "Check Credit" chạy nếu và chỉ nếu việc đặt trước có thay đổi.

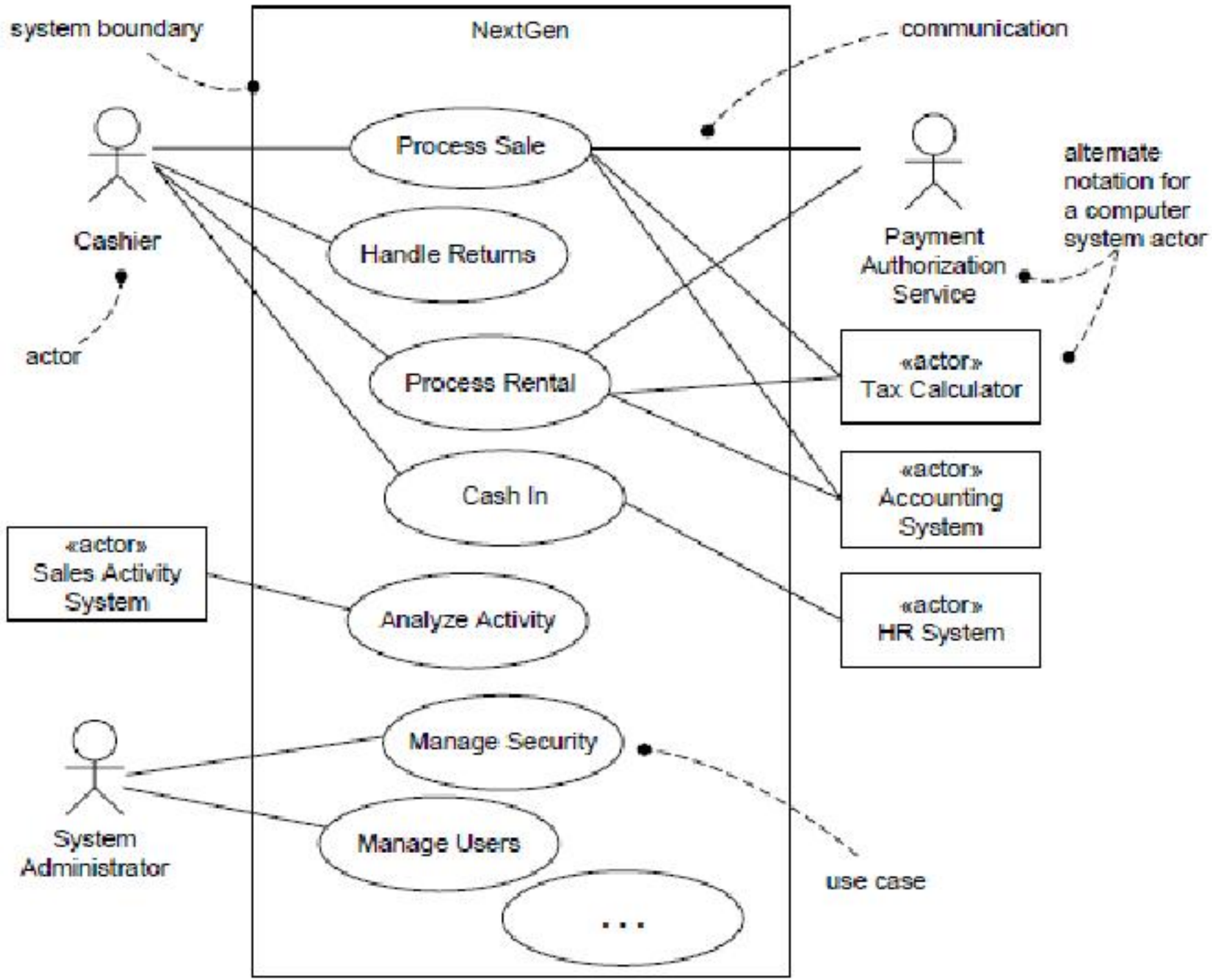
Quan hệ Include

- cho phép UC này sử dụng chức năng được cung cấp bởi UC khác.
- Nếu hai hay nhiều UC có chung chức năng nào đó, thì có thể tách riêng chức năng đó ra thành 1 UC mới. Khi đó UC cơ bản sẽ có quan hệ “include” với UC mới này.

Quan hệ Include



"Check Credit" sẽ kiểm tra tài khoản thẻ tín dụng có đủ tiền để giao dịch hay không. Vì chức năng này luôn luôn được dùng mỗi khi "Purchase Ticket " được xử lý, nó luôn được include vào "Purchase Ticket"


system boundary
NextGen
communication
Process Sale
Handle Returns
Process Rental
Cash In
Analyze Activity
Manage Security
Manage Users
...
Cashier
actor
Payment Authorization Service
alternate notation for a computer system actor
«actor» Tax Calculator
«actor» Accounting System
«actor» HR System
«actor» Sales Activity System
System Administrator
use case

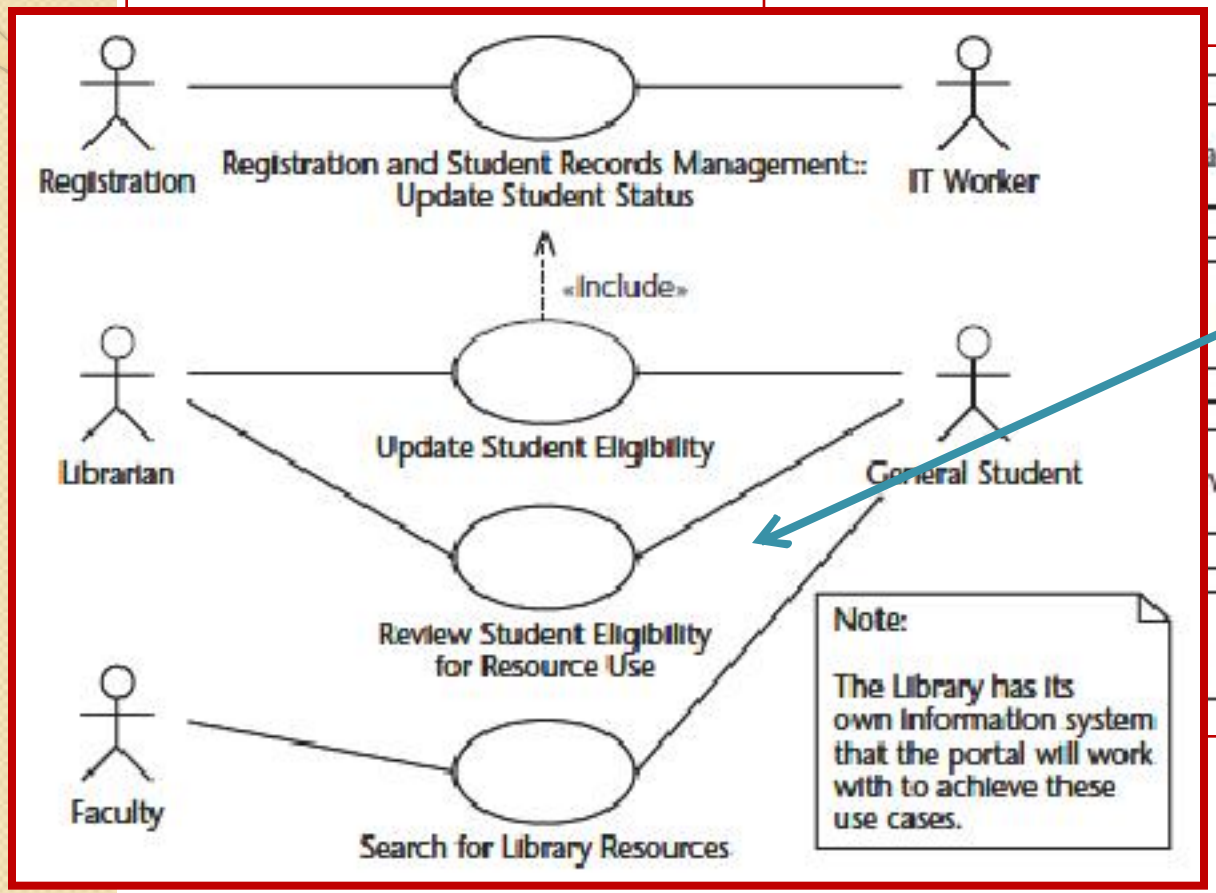
Tổ chức các use case

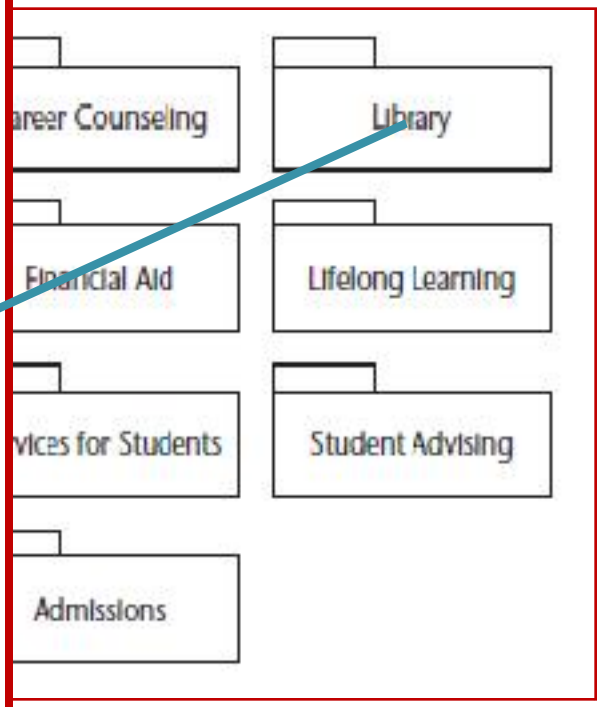
- Mô hình use case có thể chứa rất nhiều lược đồ use case.

➔Nên sắp xếp các UC sao cho nó có ý nghĩa cho khách hàng cũng như cho đội dự án.

- Thường thì nên xếp các UC và actor hoặc theo cụm chức năng hoặc theo actor chính

Đóng gói UC


Registration
Registration and Student Records Management::
Update Student Status
IT Worker
«Include»
Librarian
Update Student Eligibility
General Student
Review Student Eligibility
for Resource Use
Faculty
Search for Library Resources
Note:
The Library has its own information system that the portal will work with to achieve these use cases.


Library
Lifelong Learning
Student Advising
Admissions

Phân loại use case

- Phân biệt hai loại use case:
  - Use case hệ thống (system use case)
  - Use case nghiệp vụ (Business use case)
- Cả hai đều được tạo ra trong công đoạn Requirements nhưng loại UC nghiệp vụ ít thông dụng hơn.

Use case nghiệp vụ (Business use case)

- Nằm trong mô hình nghiệp vụ (Business use case) để giúp hiểu được toàn bộ nghiệp vụ của tổ chức, nhằm hoàn thành các mục tiêu của actor nghiệp vụ (business actor).
- Một tổ chức lớn thường có rất nhiều nghiệp vụ khác nhau và mô hình use case nghiệp vụ sẽ mô tả toàn bộ các nghiệp vụ này,

Use case hệ thống (system use case)

- UC hệ thống chỉ tập trung mô tả các chức năng chính của hệ thống mà thôi.
- Ví dụ hãng hàng không có hàng chục nghiệp vụ khác nhau nhưng hệ thống phần mềm " quản lý đặt chỗ trước " chỉ để thực hiện 1 phần trong các nghiệp vụ trên.