Phân tích thiết kế hệ thống UML

Gv: Nguyễn Ngọc Tú

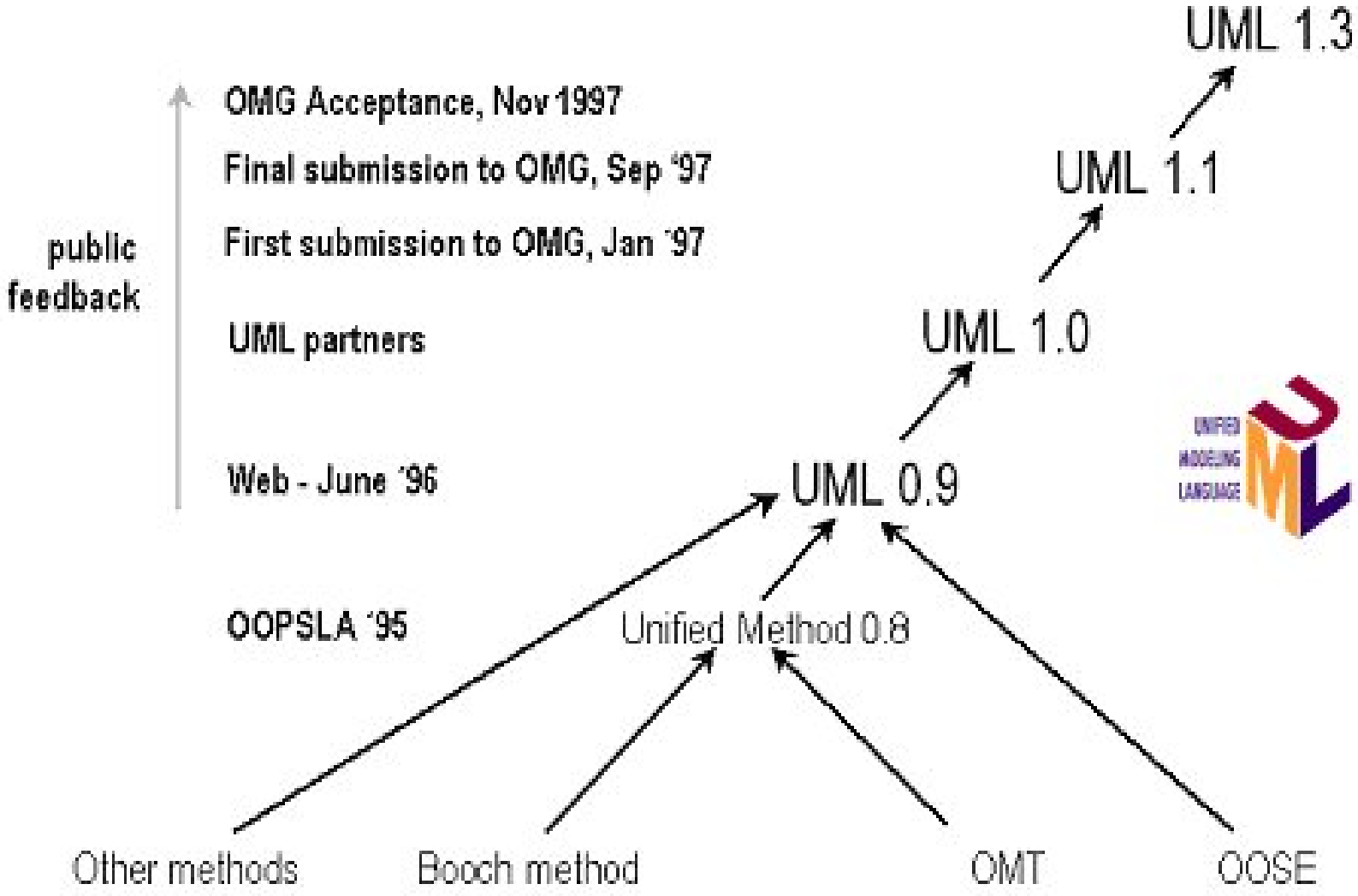
Email: nntu@hoasen.edu.vn

UML là gì?

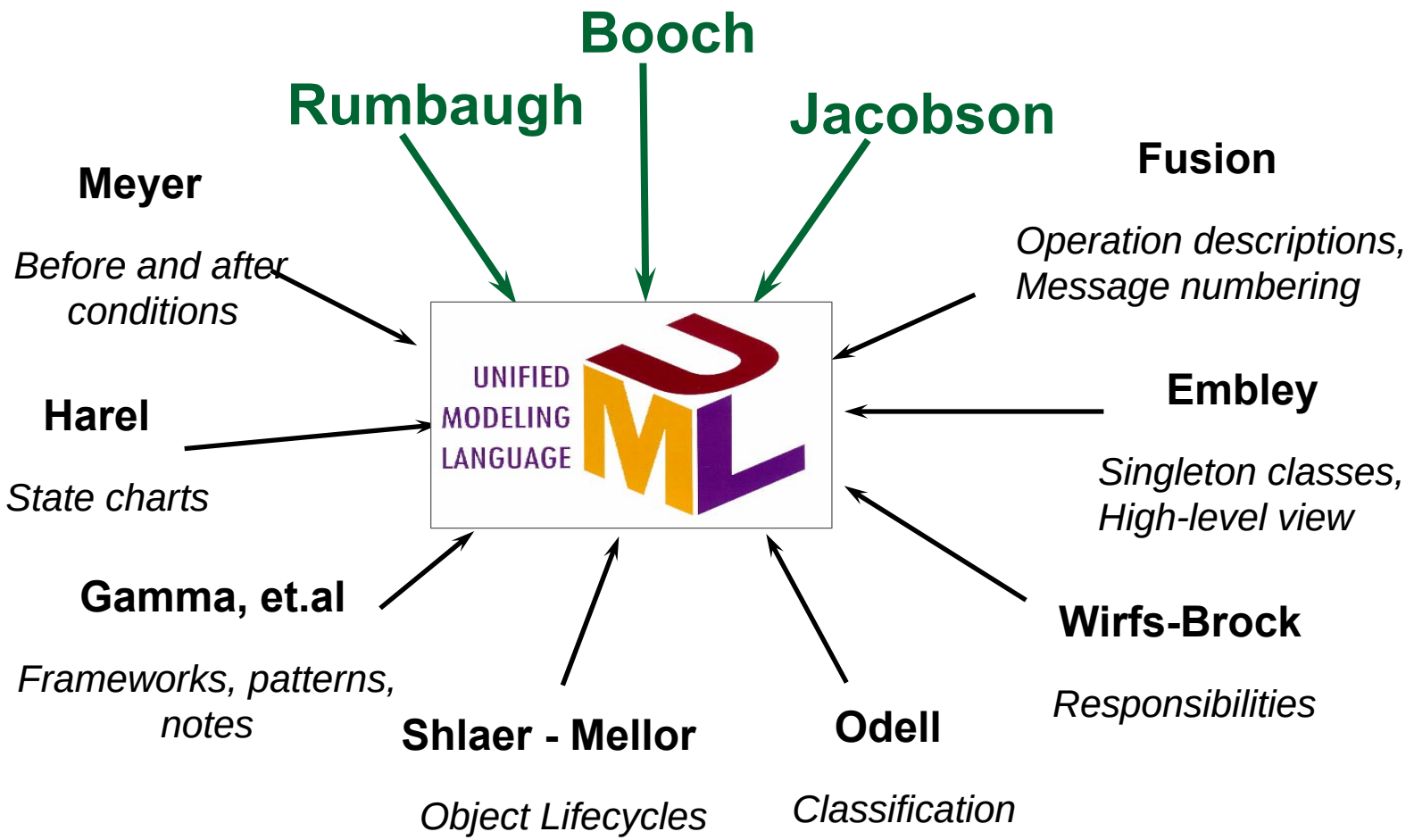
- Unified Modeling Language (UML) là ngôn ngữ giúp
  - đặc tả
  - trực quan hóa
  - xây dựng
  - làm sưu liệu

  các artifact của một hệ thống phần mềm

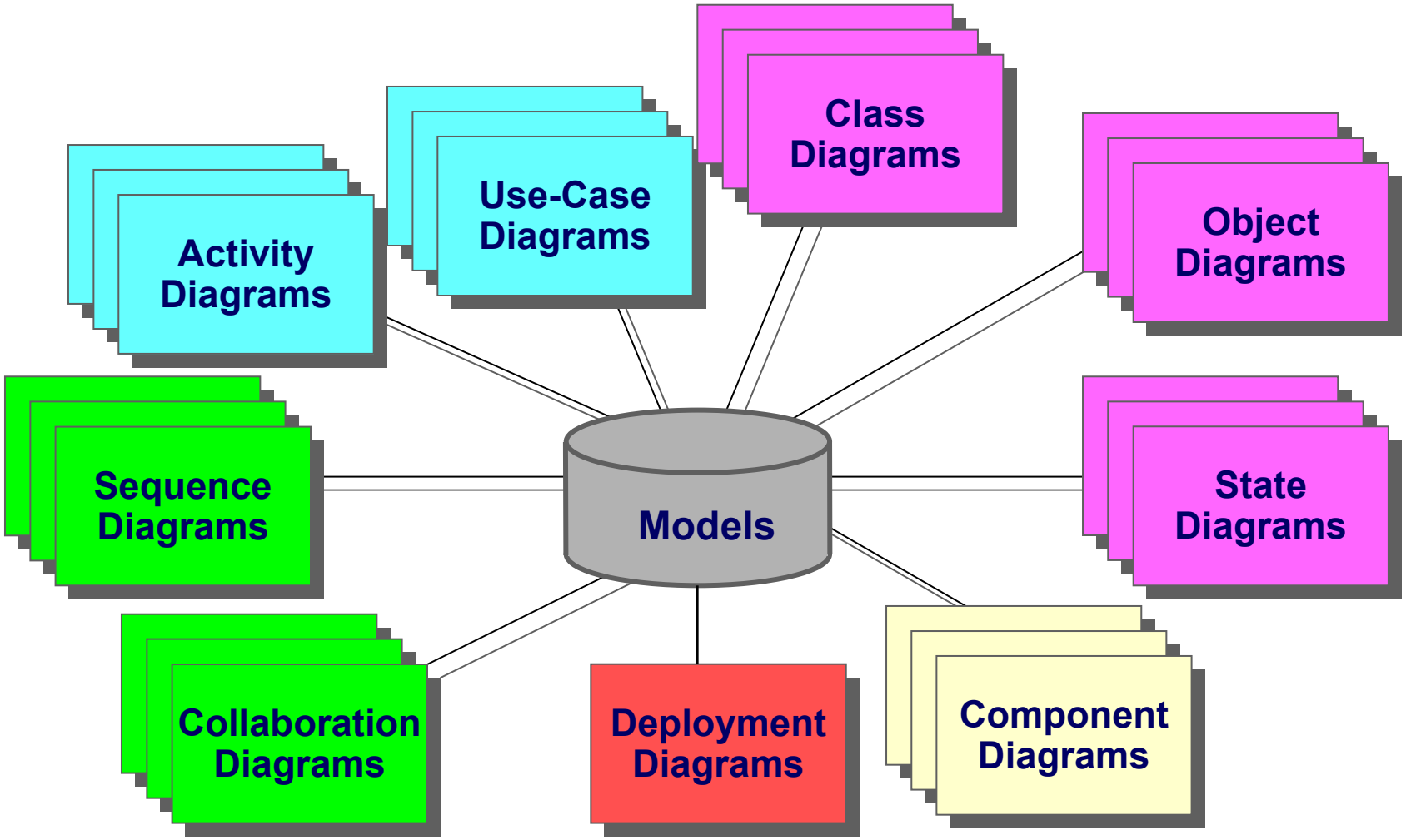
Lịch sử của UML



Đầu vào của UML

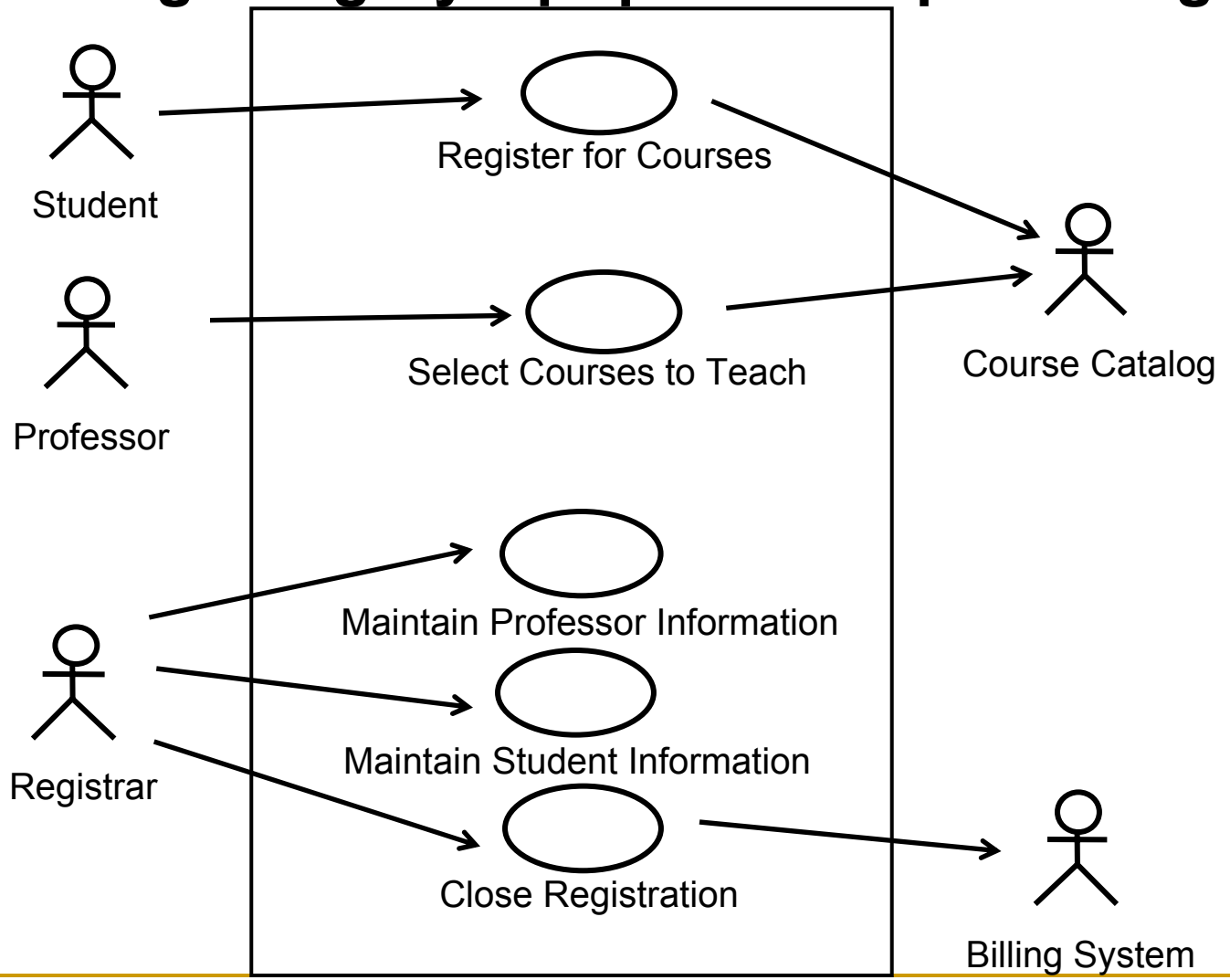


UML cung cấp các lược đồ chuẩn



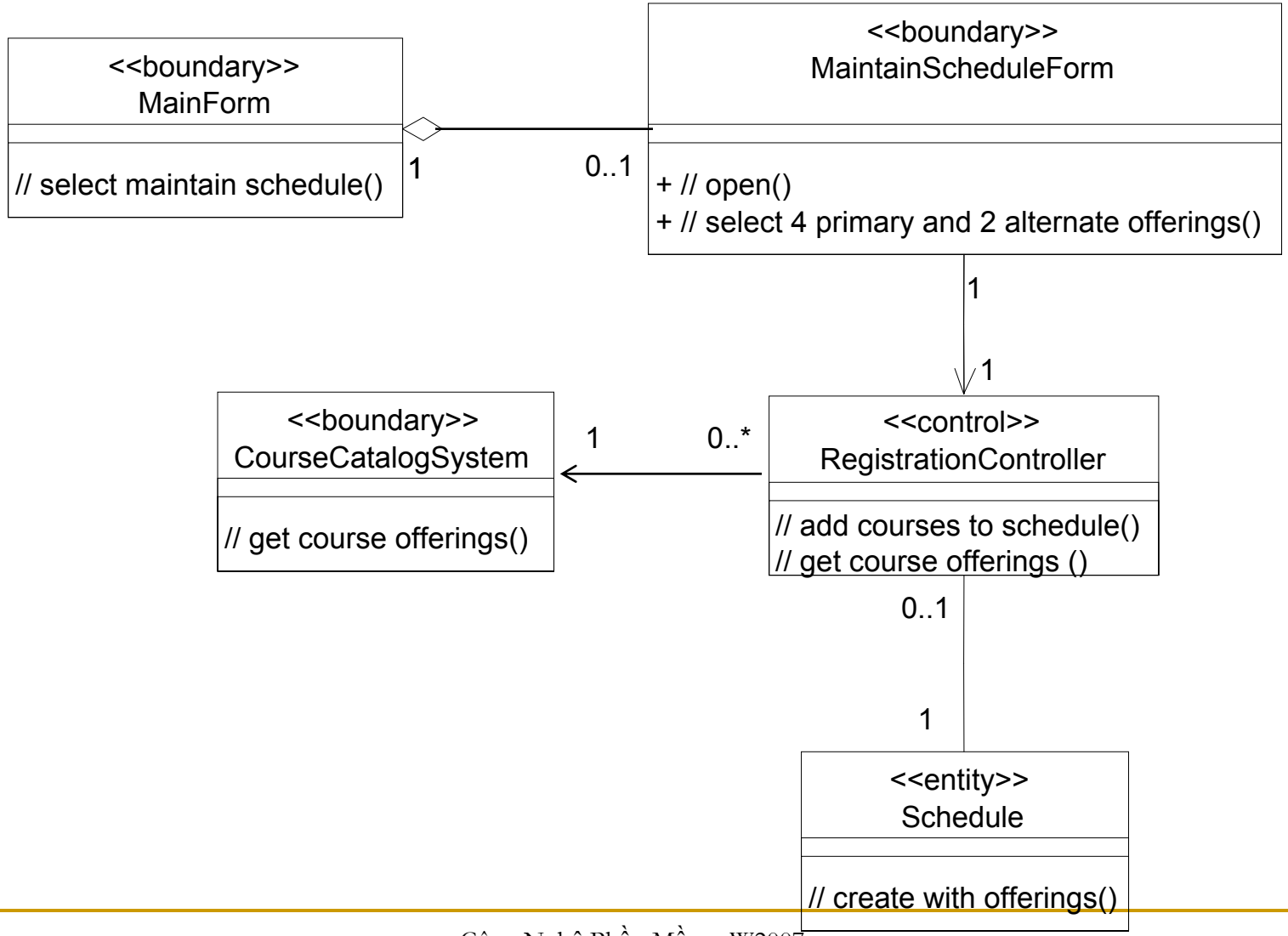
Ví dụ : Use-Case

Hệ thống Đăng Ký học phần ở một Trường ĐH



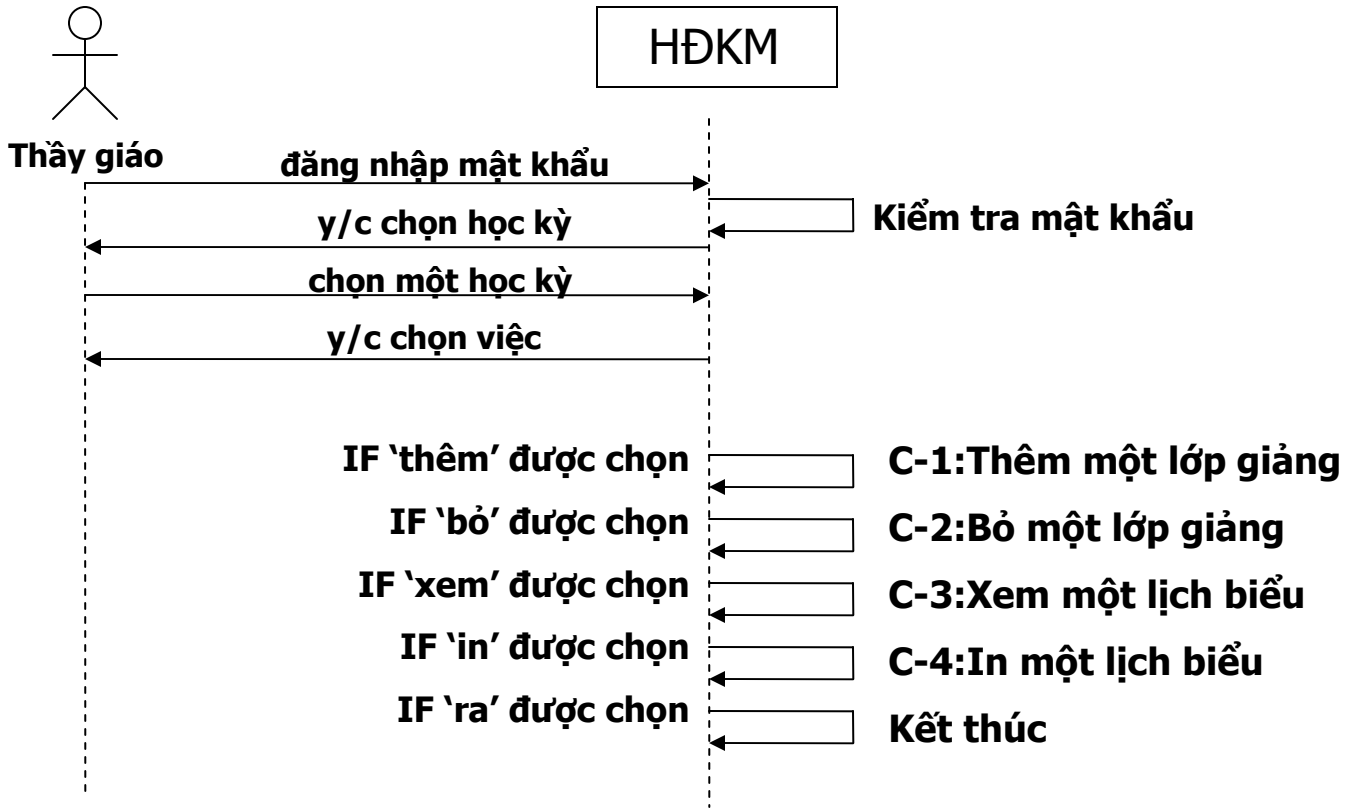
Ví dụ lược đồ Class

Hệ thống Đăng Ký học phần ở một Trường ĐH

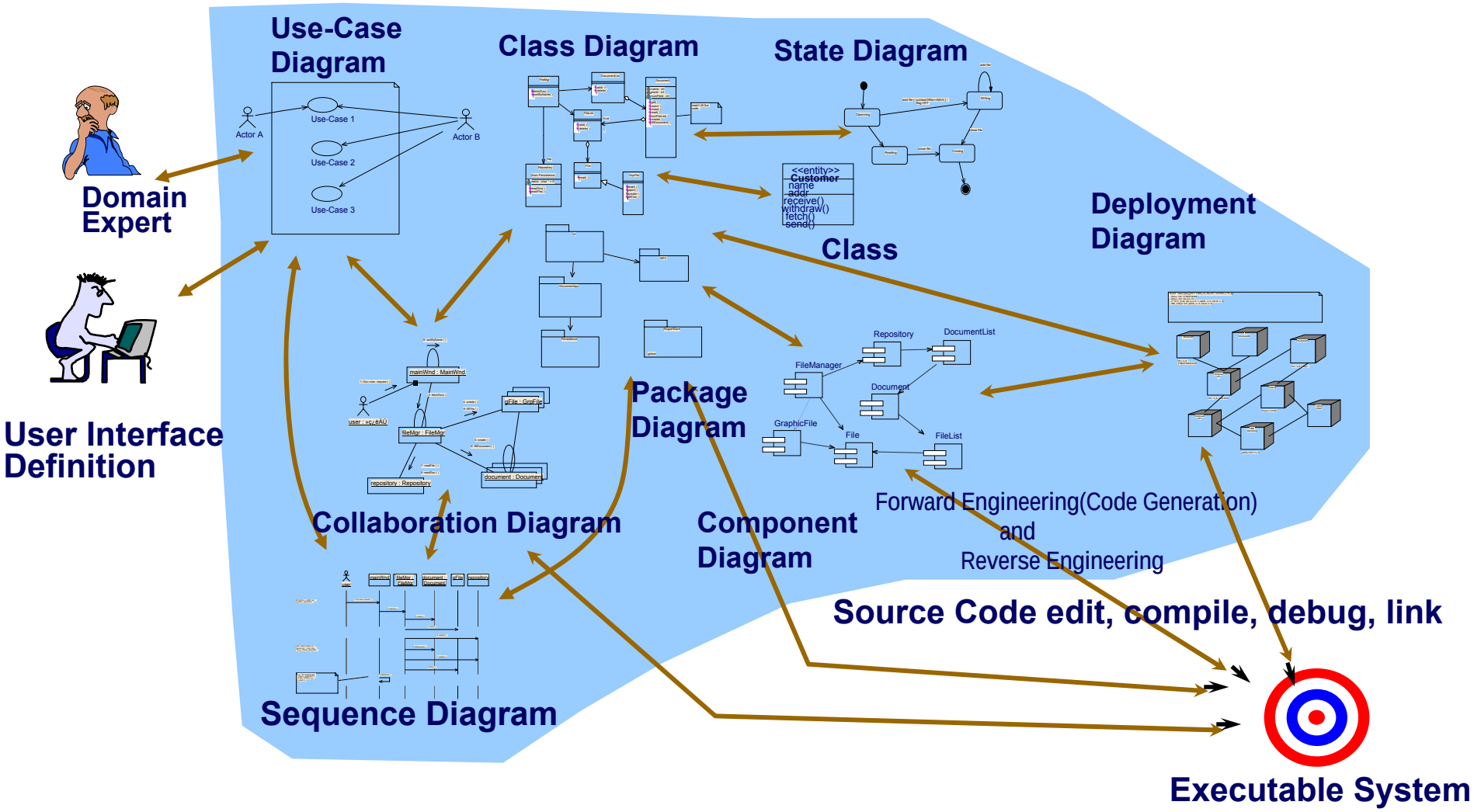


Ví dụ: lược đồ tuần tự

Dùng biểu đồ trình tự hệ thống:

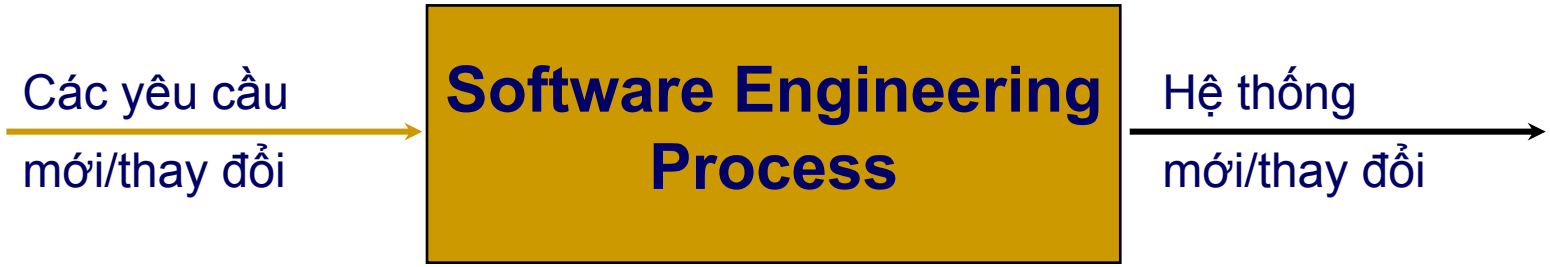


Các Artifact then chốt



Qui trình là gì?

Một qui trình xác định **Ai** làm **Gì**, **Khi nào** và **Như thế nào** để đạt được mục đích cuối cùng. Trong Công nghệ phần mềm đích là xây dựng một sản phẩm phần mềm hoặc nâng cấp một sản phẩm có sẵn

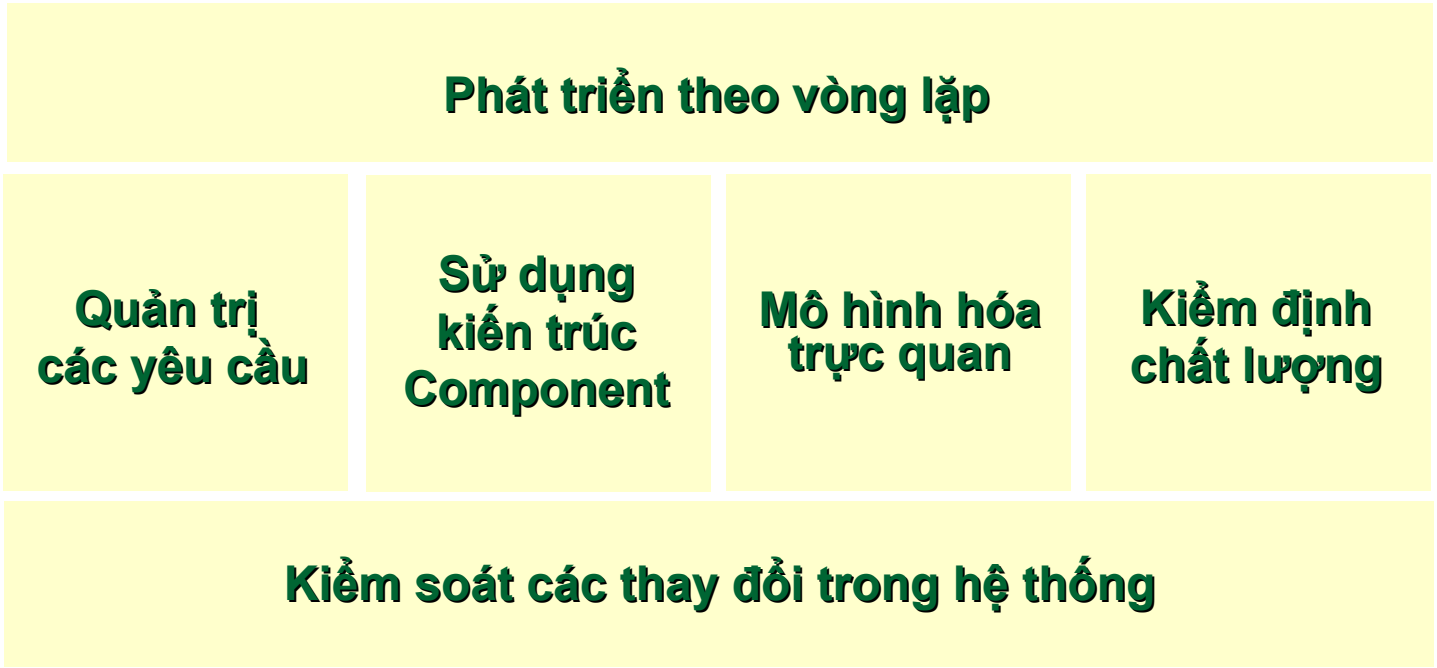


Một qui trình hiệu quả ...

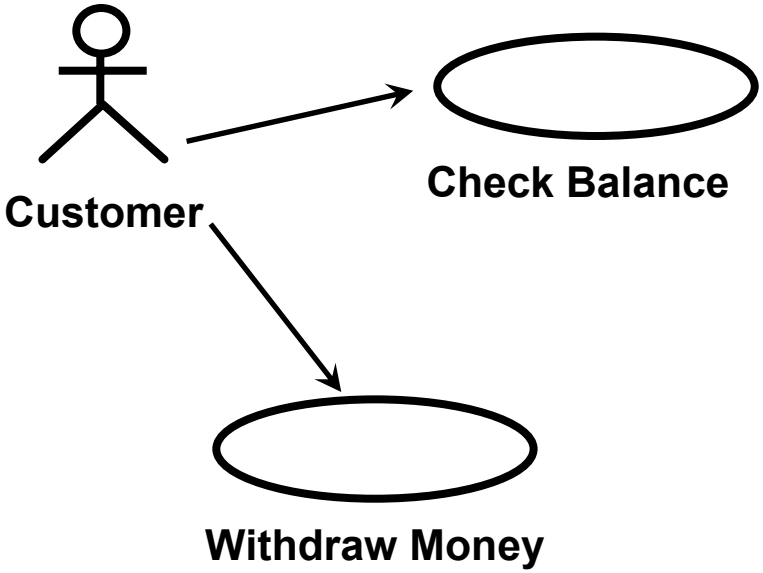
- Cung cấp các chỉ dẫn để phát triển một cách hiệu quả một phần mềm có chất lượng
- Giảm thiểu rủi ro tăng khả năng tiền định
- Nắm giữ và thể hiện các kinh nghiệm tốt
  - Học từ các kinh nghiệm khác
  - Mentor on your desktop
  - Mở rộng các tài liệu huấn luyện (Extension of training material)
- Promotes common vision and culture
- Cung cấp hướng dẫn về cách dùng các công cụ ứng dụng
- Chuyển tải thông tin trực tuyến, chỉ dẫn chi tiết

RUP chuyển tải các kinh nghiệm

RUP mô tả cách ứng dụng hiệu quả 6 kinh nghiệm quí dành cho quá trình phát triển phần mềm



RUP định hướng bởi các Use-Case



Các Use-Case của một Cash Machine

Một **actor** là một người hoặc một cái gì đó bên ngoài hệ thống tương tác với hệ thống

Một **Use-Case** là một chuỗi các hành động mà hệ thống thực hiện mang lại một kết quả quan sát được đối với một actor.

Use-Case chứa luồng các sự kiện

Luồng các sự kiện (Flow of events) của Use-Case rút tiền (Withdraw Money):

1. Use-Case bắt đầu khi khách hàng đưa thẻ tín dụng vào. Hệ thống đọc và thẩm tra thông tin của thẻ.
2. Hệ thông nhắc nhập số PIN. Hệ thống kiểm tra số PIN.
3. Hệ thống hỏi tác vụ nào khách hàng muốn thực hiện. Khách hàng chọn "Rút tiền."
4. Hệ thống hỏi số lượng. Khách hàng nhập số lượng.
5. Hệ thống yêu cầu nhập kiểu tài khoản. Khách hàng chọn checking hoặc savings.
6. Hệ thống liên lạc với ATM network . . .

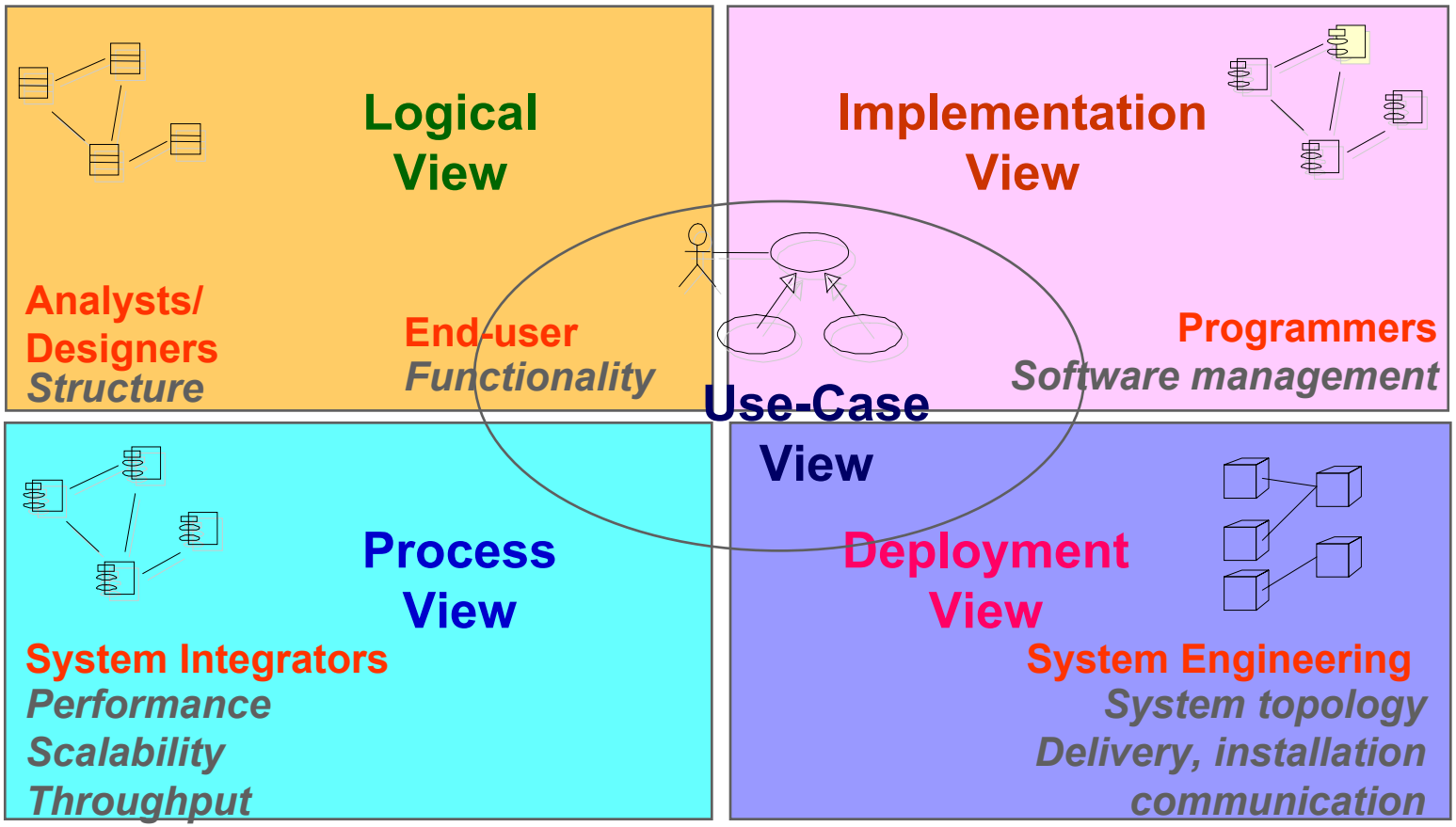
Lợi ích của Use-Case

- Use-Case ngắn gọn, đơn giản và, dễ hiểu đối với
  - End users, developers, … hiểu các yêu cầu chức năng của hệ thống
- Use-Case định hướng nhiều hoạt động trong qui trình:
  - Tạo và thẩm định mô hình thiết kế (design model)
  - Định nghĩa các test case và các thủ tục của test model
  - Qui hoạch các vòng lặp
  - Tạo sưu liệu cho người dùng
  - Triển khai hệ thống
- Use-Case giúp đồng bộ hóa nội dung các mô hình (model) khác nhau

RUP là qui trình Architecture-Centric

- Kiến trúc là trọng tâm của các vòng lặp đầu tiên
  - Xây dựng, thẩm tra, và xđ giới hạn của kiến trúc tạo thành mục tiêu chính của giai đoạn triển khai (elaboration)
- Prototype của kiến trúc xác nhận tính hợp lệ của kiến trúc và đóng vai trò baseline cho phần còn lại của qui trình phát triển
- Sưu liệu về kiến trúc phần mềm là artifact chính mô tả kiến trúc được chọn
- Các artifact khác kế thừa từ kiến trúc:
  - Design guidelines bao gồm cách sử dụng pattern và idiom
  - Cấu trúc sản phẩm
  - Cấu trúc của đội ngũ phát triển phần mềm

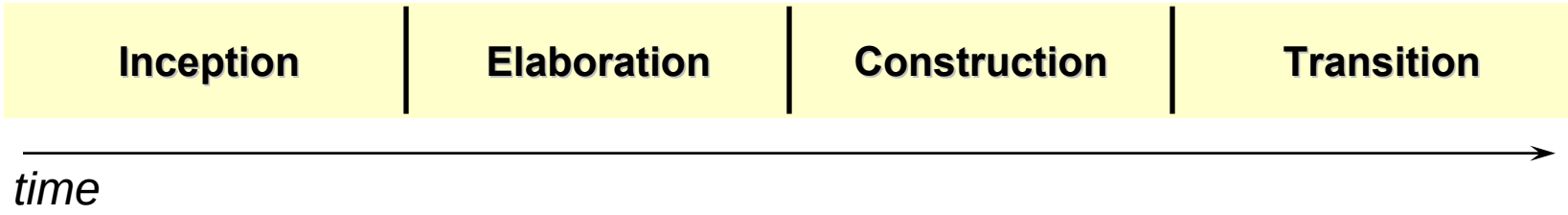
Biểu diễn kiến trúc : Mô hình 4+1 View



Lợi ích qui trình Architecture-Centric

- Giúp đạt và giữ vững sự kiểm soát tốt dự án, quản lý độ phức tạp của nó, và duy trì sự tích hợp của hệ thống
- Cung cấp khả năng tái sử dụng hiệu quả ở nhiều mức độ khác nhau
- Cung cấp nền tảng cho quản lý dự án
- Giúp dễ dàng phát triển theo hướng component-based
  - Một component đảm trách một chức năng rõ ràng trong khuôn khổ một kiến trúc được định nghĩa tốt
  - Một component thích ứng và cung cấp mọt hiện thực hóa vật lý của một tập các giao diện (interface)
  - Các component tồn tại tương ứng với một kiến trúc cụ thể

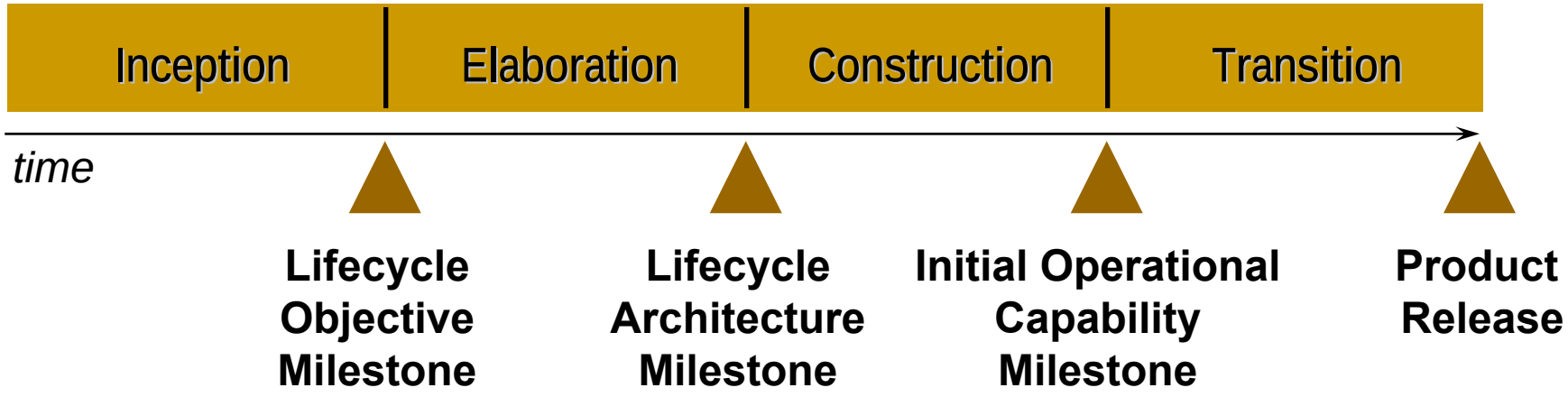
Các Phase trong chu kỳ sống



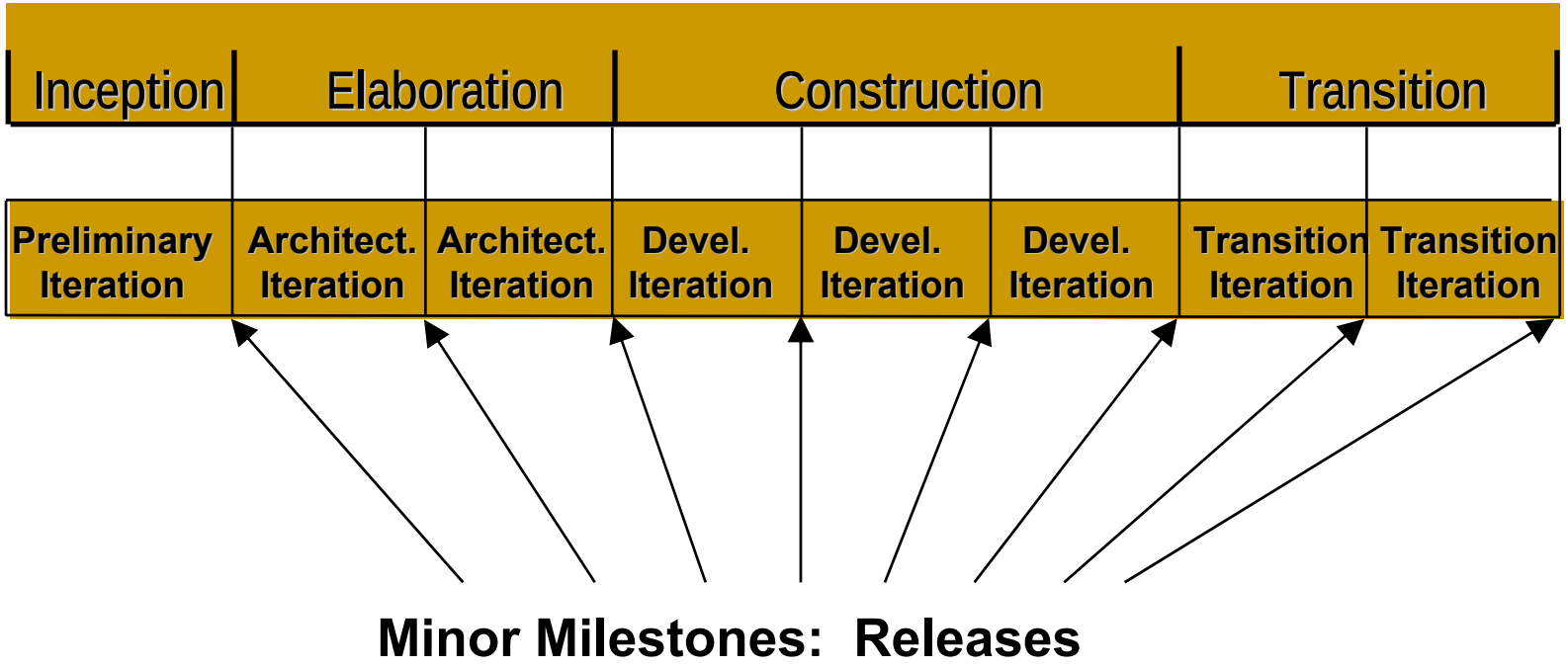
RUP có 4 phase:

- **Inception** - Định nghĩa phạm vi của dự án
- **Elaboration** - Lập kế hoạch dự án, mô tả các đặc tính, định ranh giới kiến trúc
- **Construction** – Xây dựng sản phẩm
- **Transition** - Chuyển giao sản phẩm cho người dùng

Các mốc chính đặt tại ranh giới các Phase

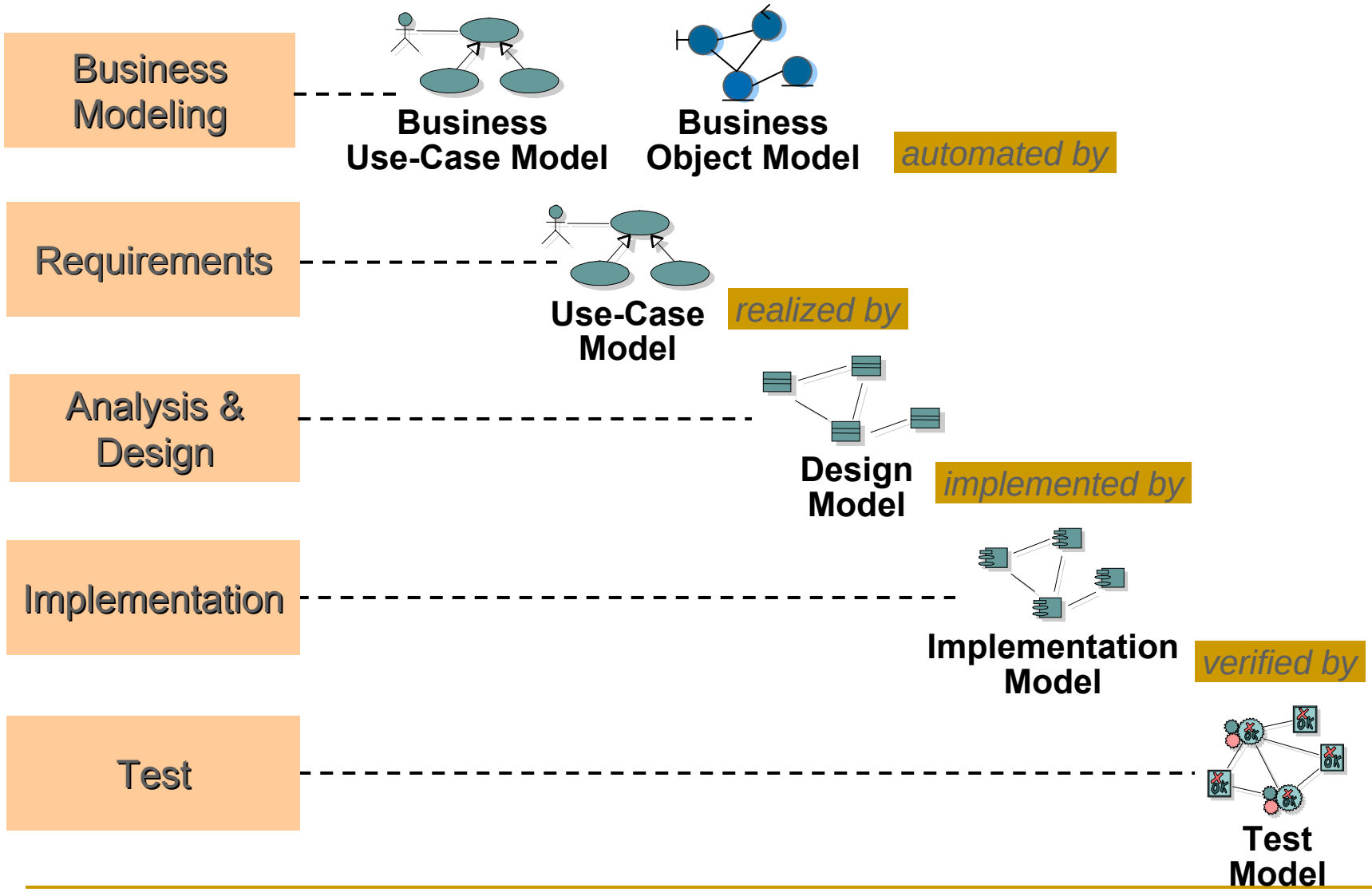


Các Iteration và Phase

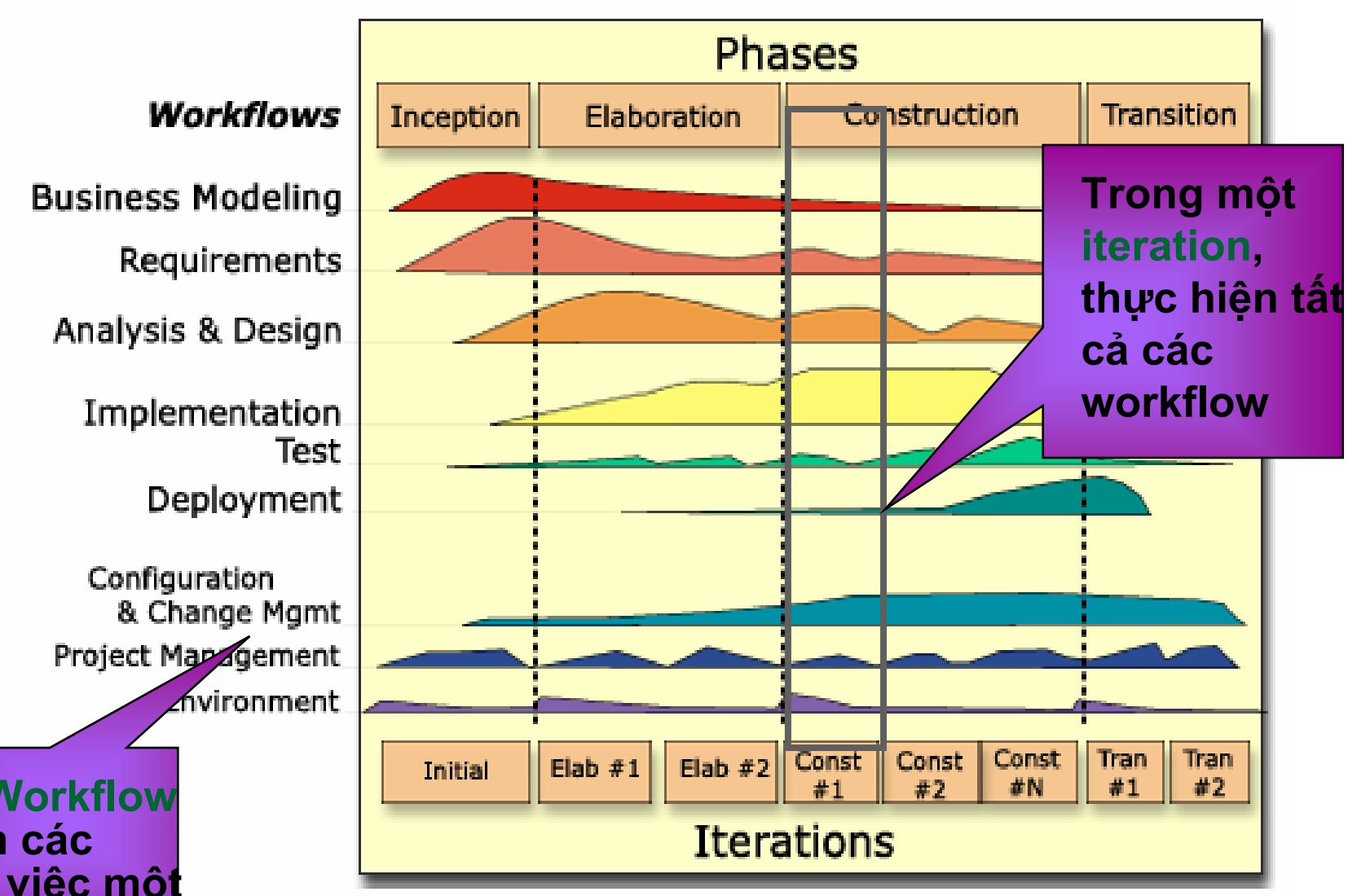


Một iteration là một chuỗi các hoạt động với một kế hoạch lập sẵn và một tiêu chuẩn lượng giá, có kết quả là một phiên bản release (internal hay external)

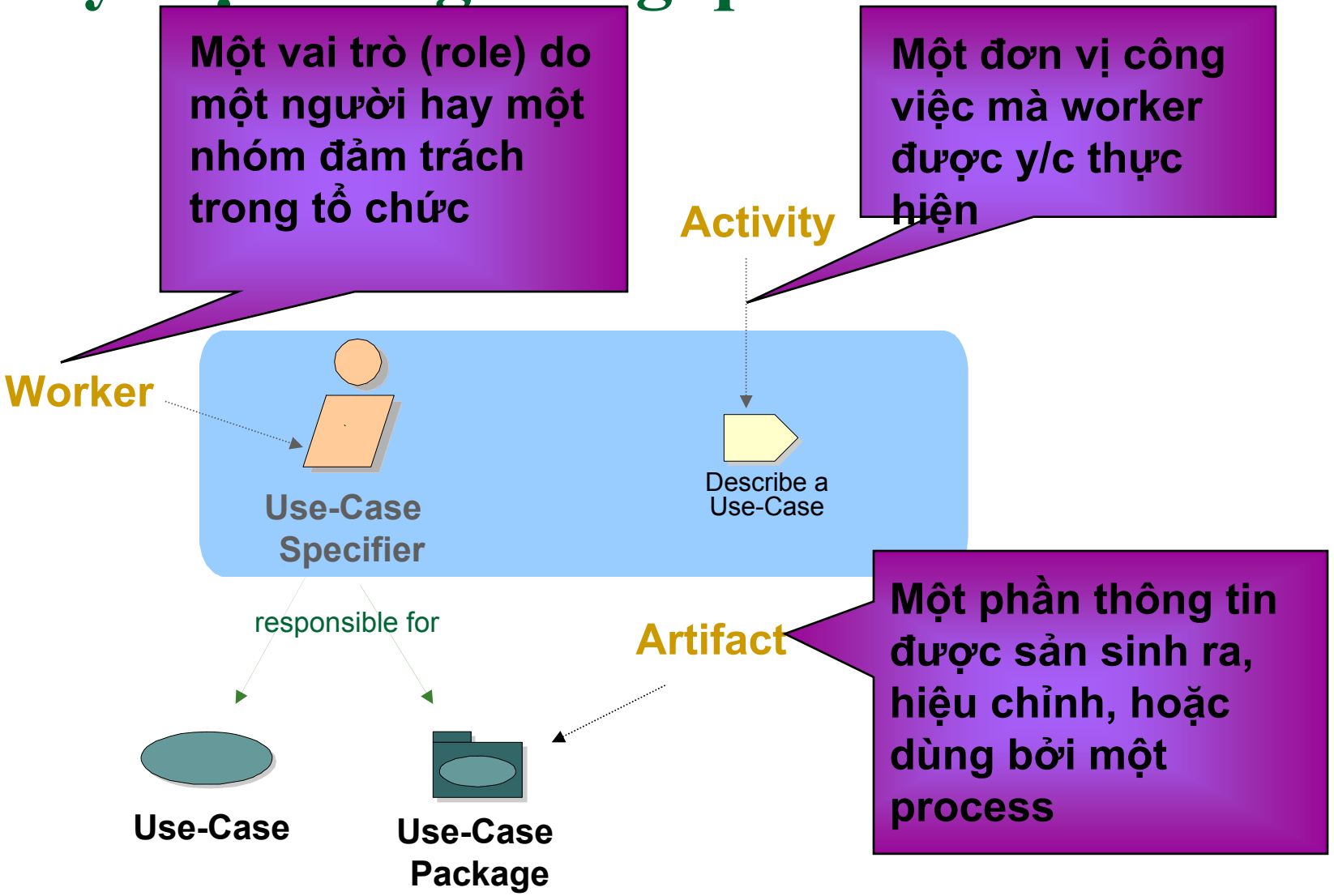
Các Workflow chính



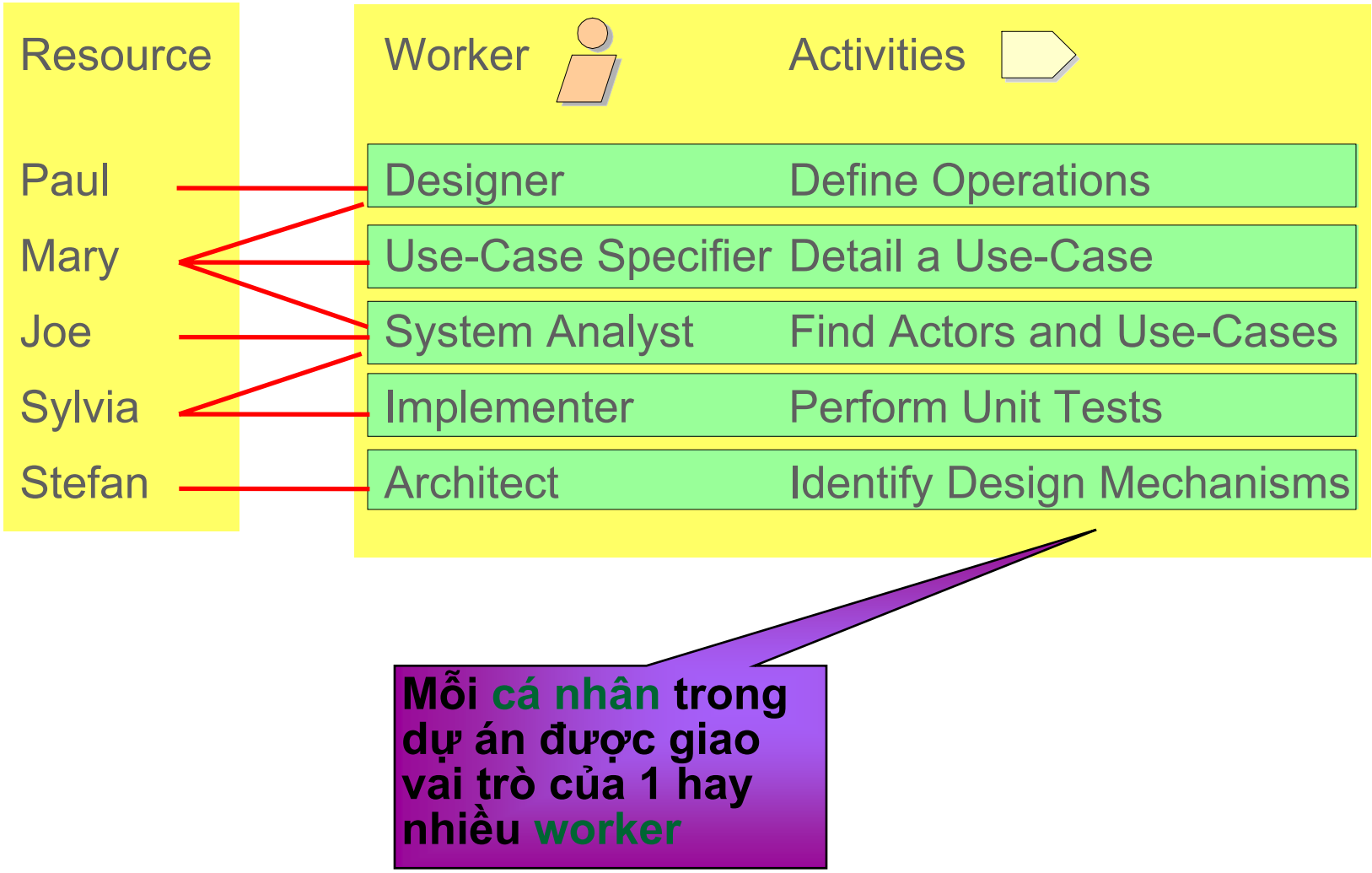
Mô hình tích hợp



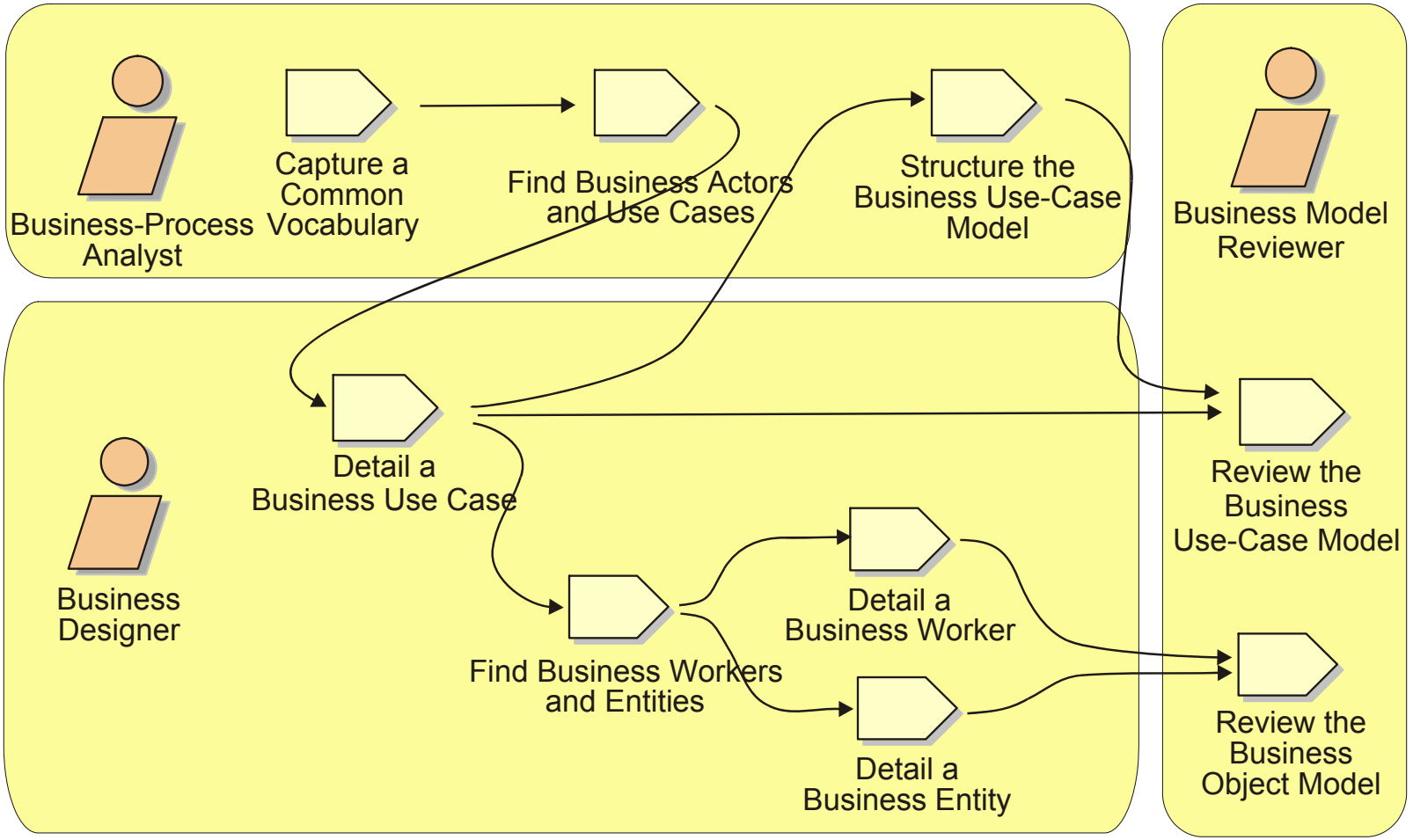
Các ký hiệu dùng trong qui trình



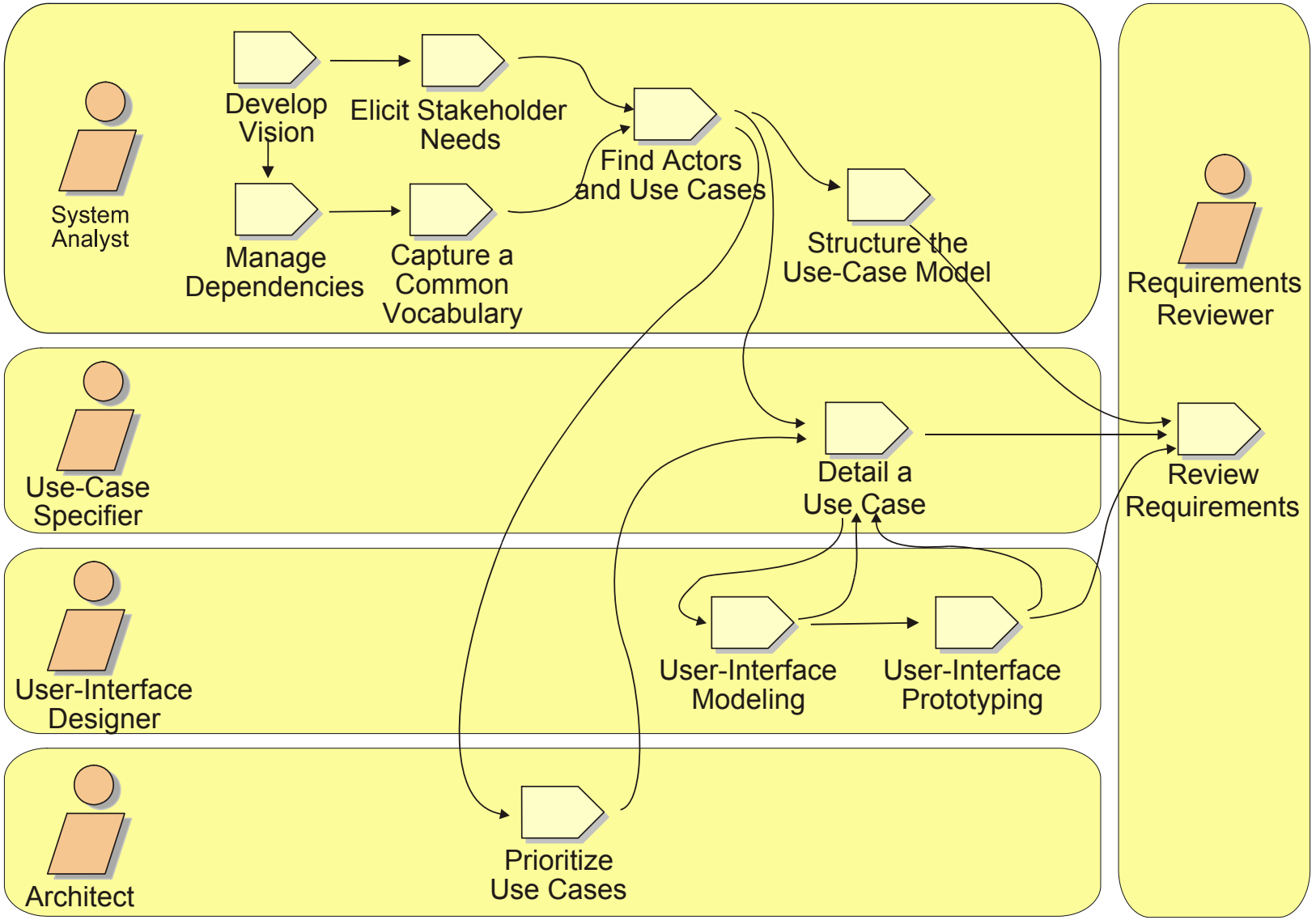
Phân công công việc



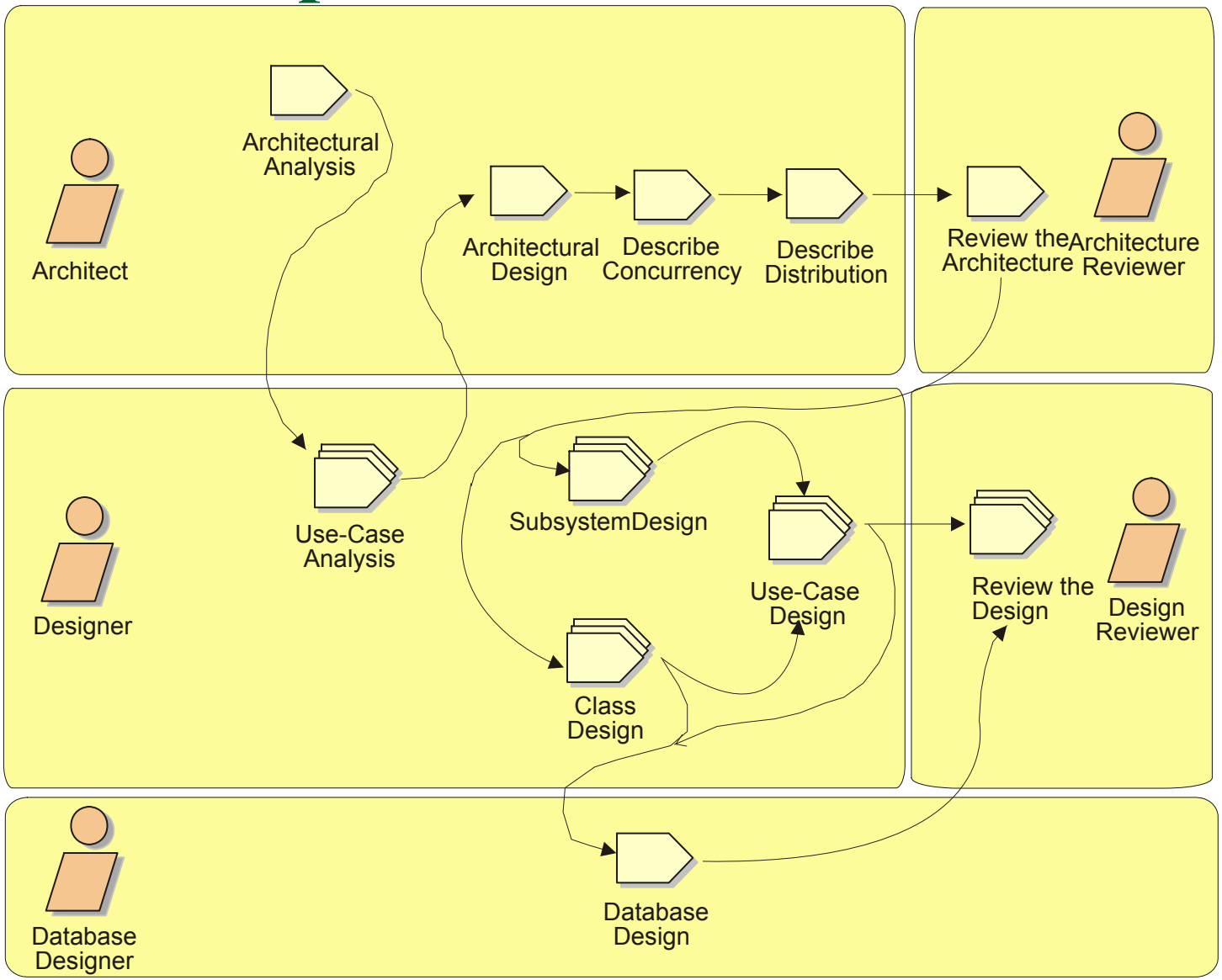
Workflow mô hình hóa nghiệp vụ



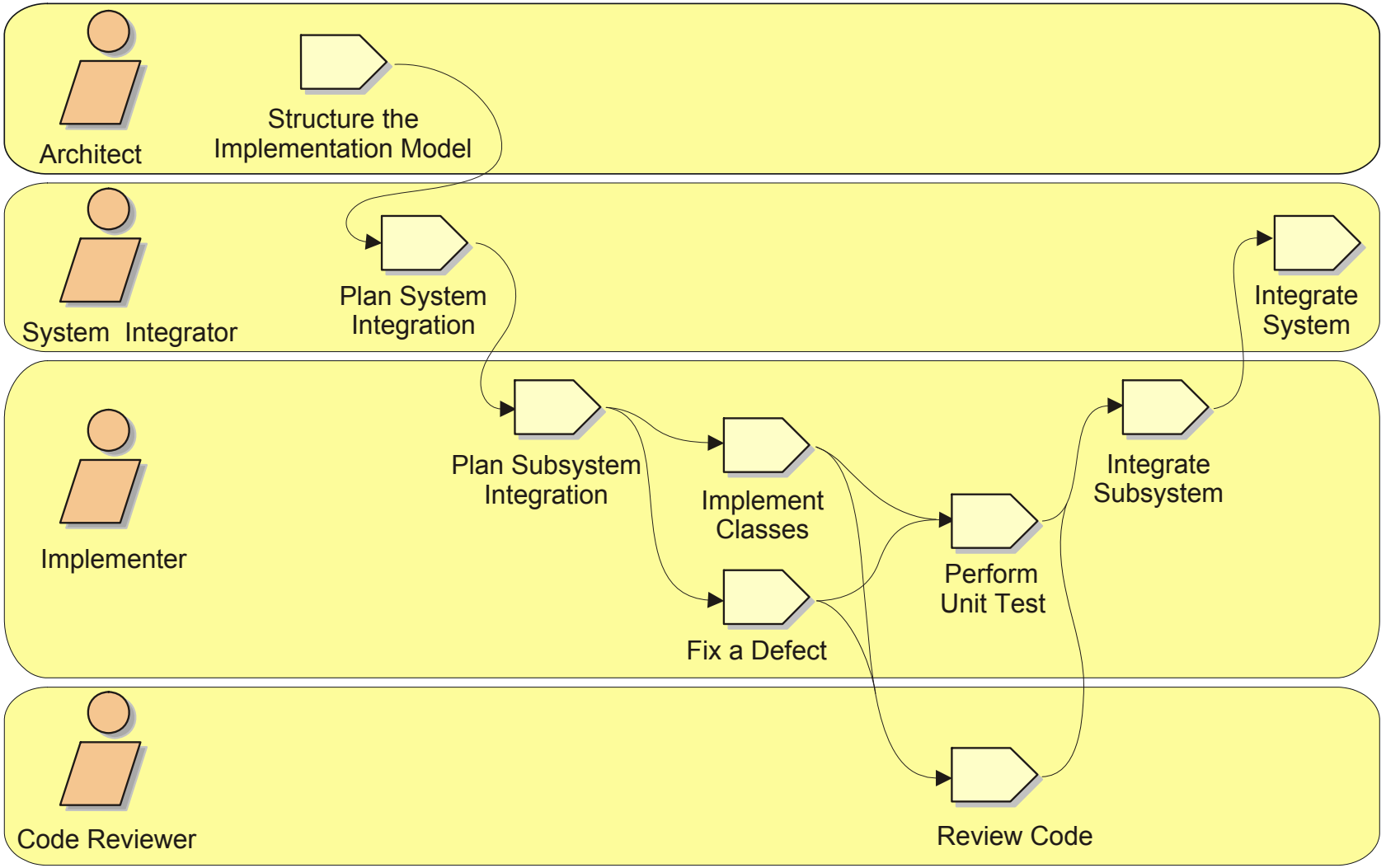
Workflow xác định yêu cầu



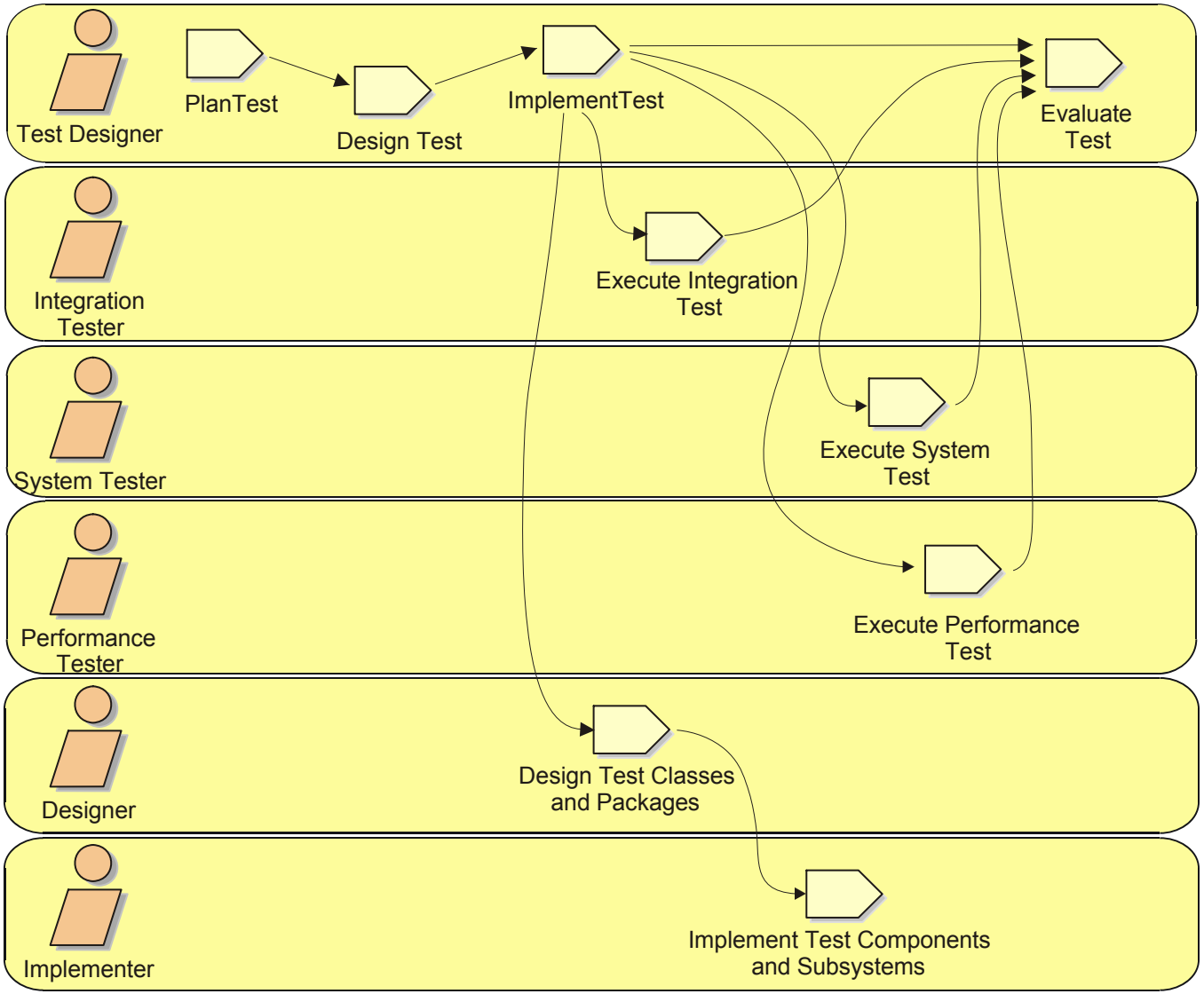
Workflow phân tích và thiết kế



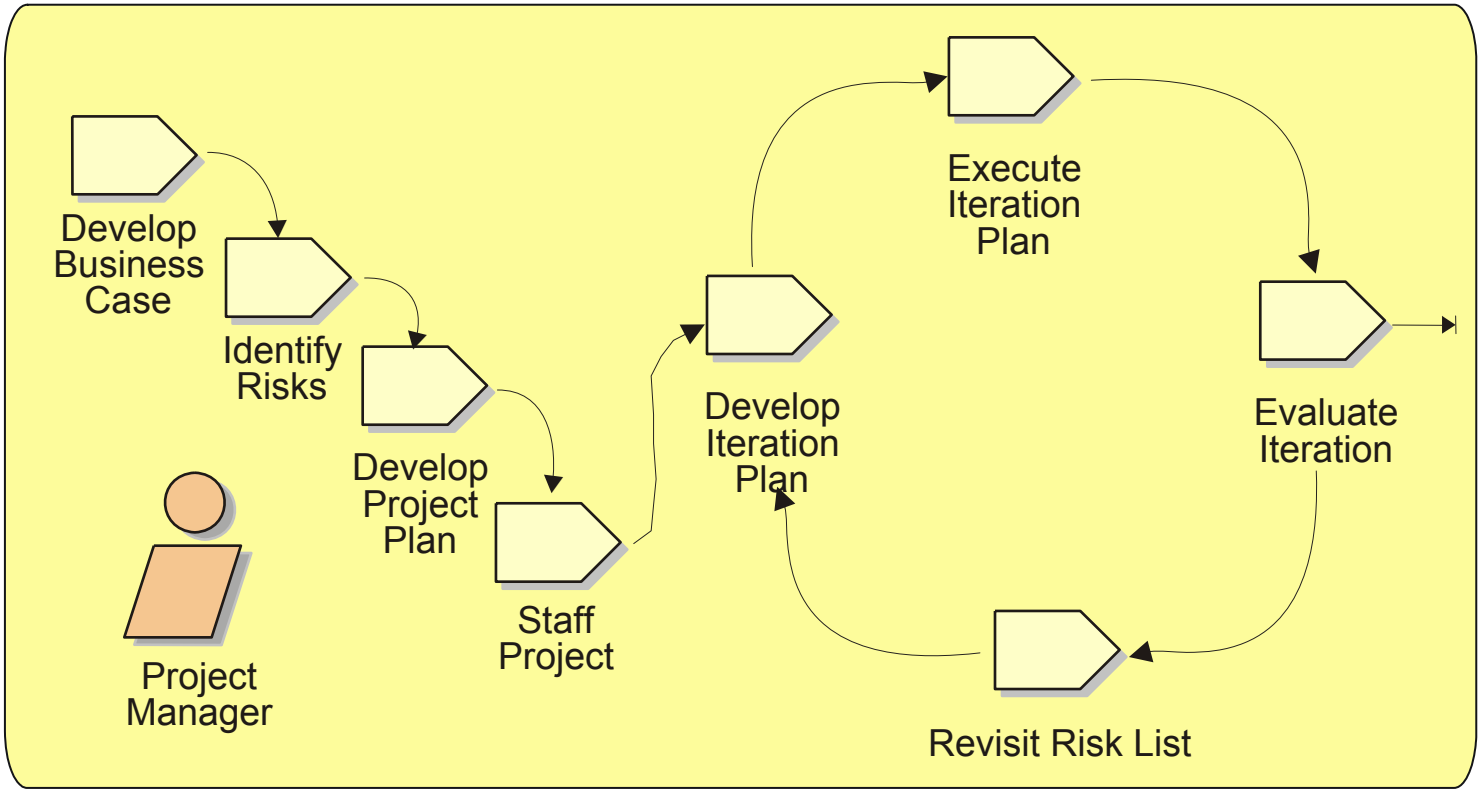
Workflow cài đặt



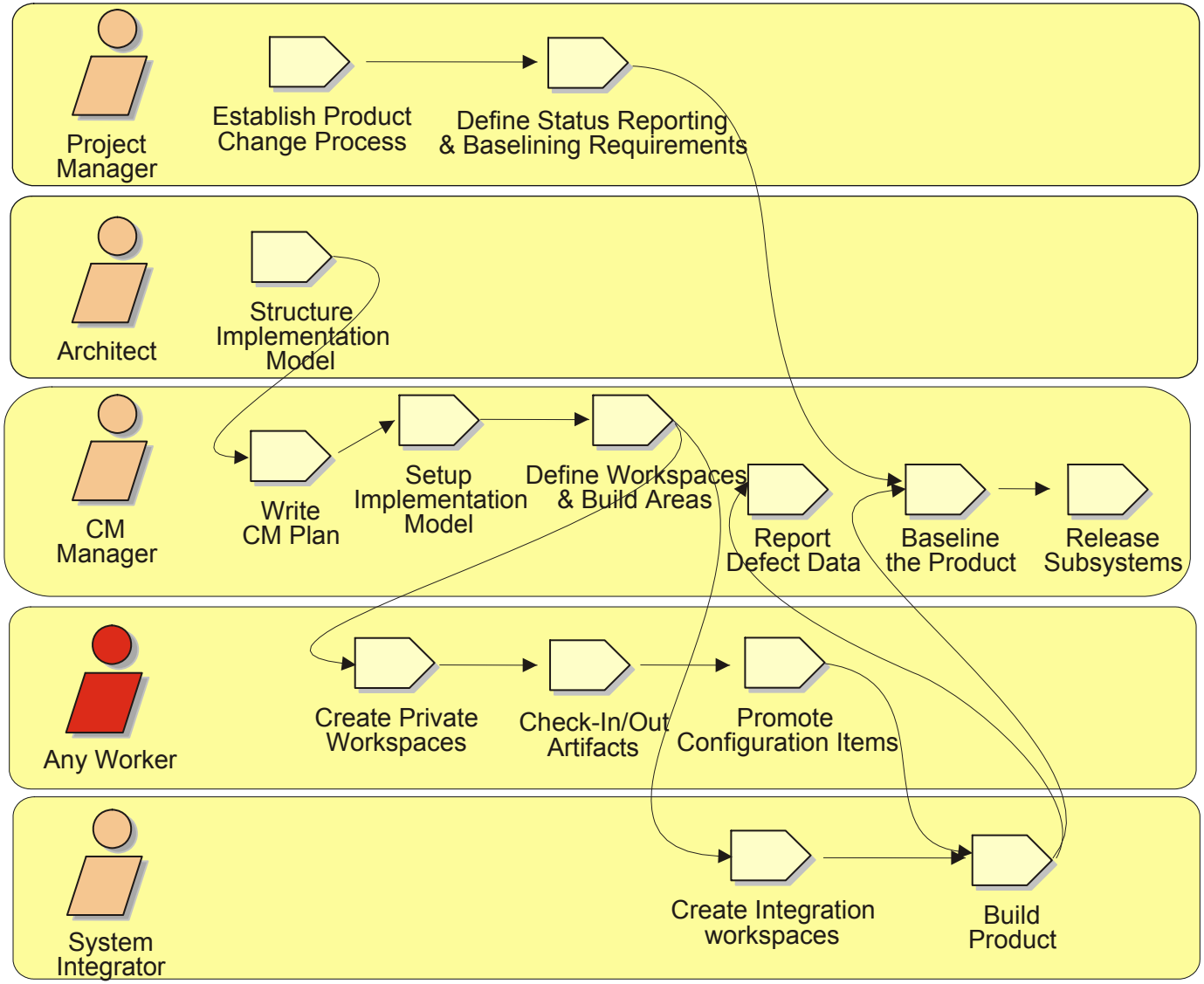
Workflow kiểm chứng



Workflow quản trị dự án



Workflow quản trị cấu hình & các thay đổi



Workflow quản trị môi trường

- Cấu hình quá trình PTPM
- Nâng cao chất lượng quá trình PTPM
- Chọn và thu thập các công cụ
- Tinh chỉnh, bổ sung các công cụ
- Hỗ trợ quá trình phát triển
- Huấn luyện

Khái niệm Guideline, Mentor, và Template

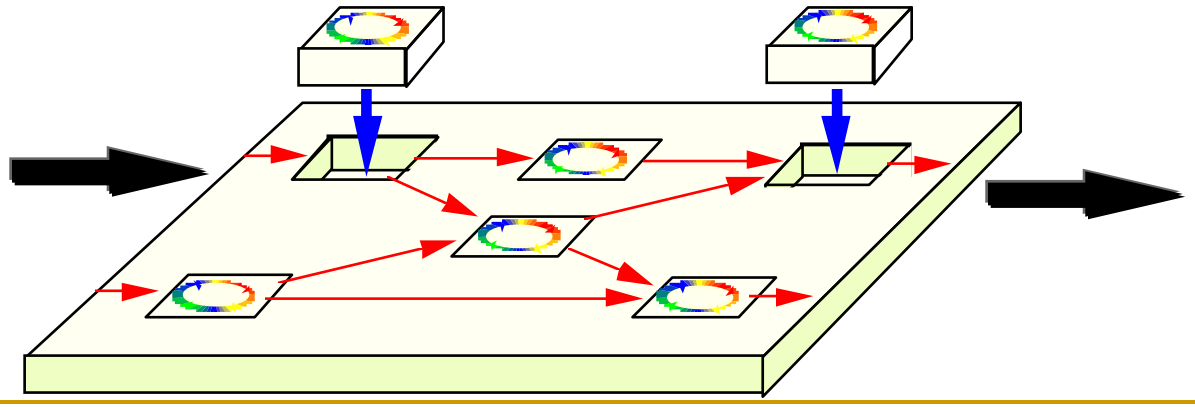
- Guidelines là các luật, gợi ý, và heuristics hỗ trợ cho các hoạt động
  - Ví dụ, modeling và programming guidelines
- Tool mentors diễn tả các dùng một công cụ cụ thể để thực hiện một hoạt động hoặc các bước trong 1 HĐ
  - Ví dụ, xây dựng 1 design model dùng Rational Rose
- Templates là các artifact định nghĩa sẵn
  - Ví dụ, một Rational SoDA template cho 1 Use-Case Report
- Guidelines, tool mentors và templates giúp dễ dàng ứng dụng qui trình đúng đắn và nhất quán

Các công cụ hỗ trợ

Process Workflows	
Business Modeling	Requisite Pro, Rose, SoDA
Requirements	Requisite Pro, Rose, SoDA
Analysis and Design	Rose, SoDA, Apex
Implementation	Rose, Apex, SoDA, Purify, ...
Test	SQA TeamTest, Quantify, PerformanceStudio,...
Deployment	SoDA, ClearCase, ...
Supporting Workflows	
Config. & Change Mgmt.	ClearCase, ClearQuest
Project Management	Unified Process, Microsoft® Project, ...
Environment	Unified Process, Rational Tools

Thích nghi hóa một qui trình

- Việc thích nghi một qui trình bao gồm cấu hình và cài đặt qui trình
- Khi **cấu hình** qui trình, process framework được làm cho thỏa mãn các yêu cầu và ràng buộc của tổ chức tiếp nhận
  - Kết quả được ghi nhận trong "Development Case"
- Khi **cài đặt** qui trình, thực tế của tổ chức được thay đổi để dùng hiệu quả qui trình



Rational Unified Process

- Unified Modeling Language (UML) là ngôn ngữ dùng để đặc tả, trực quan hóa, xây dựng, và làm sưu liệu về các artifact của một hệ thống phần mềm
- Một qui trình phát triển phần mềm định nghĩa **Ai** làm **Gì, Khi nào** và **Như thế nào** để xây dựng một sản phẩm phần mềm
- RUP có 4 phase: **Inception, Elaboration, Construction và Transition**
- Mỗi phase chấm dứt tại các mốc chính và gồm 1 hay nhiều iteration
- Một **iteration** là một chuỗi các hoạt động với một kế hoạch lập sẵn và một tiêu chuẩn lượng giá, có kết quả là một phiên bản release

Rational Unified Process

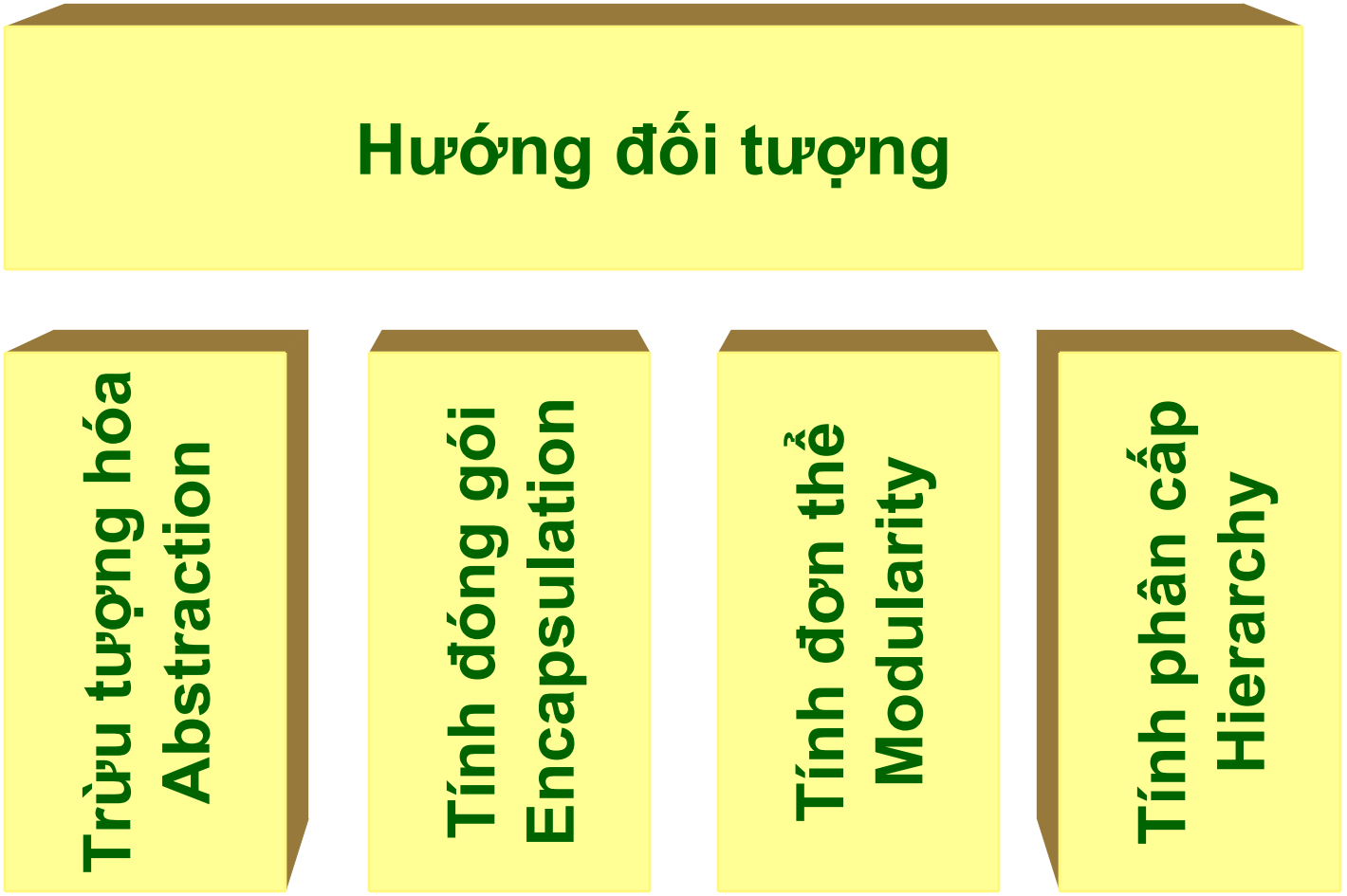
- Một **workflow** nhóm các hoạt động liên quan với nhau
- Mỗi workflow được thực hiện trong 1 **iteration** và cho kết quả là một model được sản sinh theo từng bước
- Một **artifact** là một phần thông tin được sản sinh ra, hiệu chỉnh, hoặc dùng bởi một process
- Một **worker** là một vai trò do một các nhân hay 1 nhóm đảm trách trong 1 tổ chức phát triển phần mềm.
- Một **activity** là một đơn vị công việc có thể được yêu cầu thực hiện

OOD

Hướng đối tượng

- **Các nguyên tắc cơ bản của OO**
- **Các khái niệm cơ bản của OO**
- **Sức mạnh của OO**
- **Các cơ chế mô hình hoá cơ bản của UML**

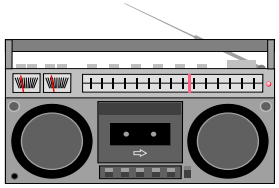
Các nguyên tắc cơ bản của OO



Thế nào là trừu tượng hóa?

Người bán hàng

Khách hàng

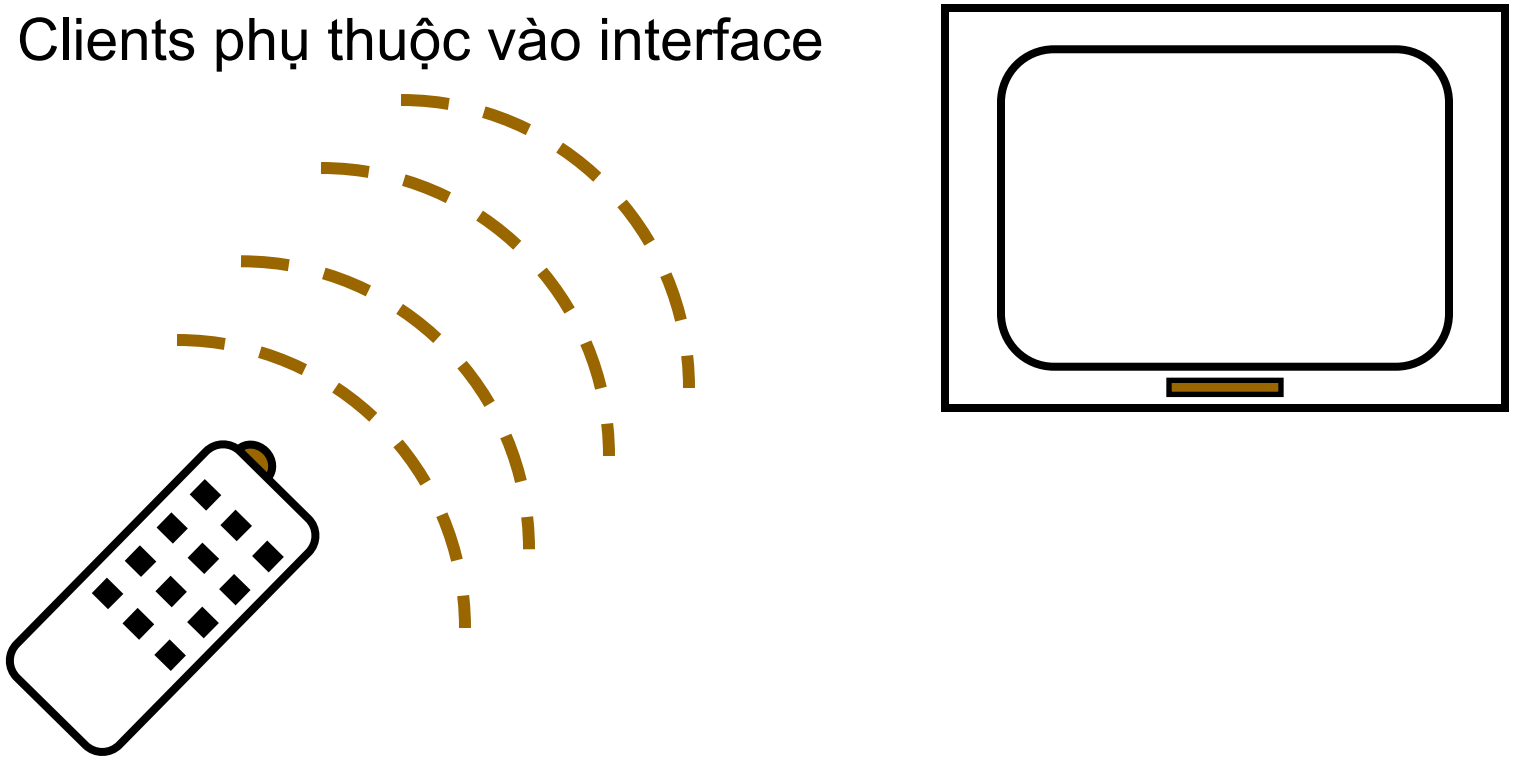


Sản phẩm

Quản lý được độ phức tạp

Đóng gói (Encapsulation) là gì?

- **Che dấu cài đặt bên trong với clients**
  - Clients phụ thuộc vào interface



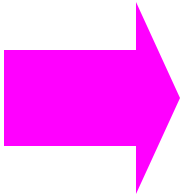
Tăng tính mềm dẻo

Tính đơn thể là gì?

- **Phân chia nhỏ một vấn đề phức tạp thành nhiều phần nhỏ, đơn giản hơn quản lý được**

Nhận
đơn đặt hàng

Hệ thống xử lý
đơn đặt hàng



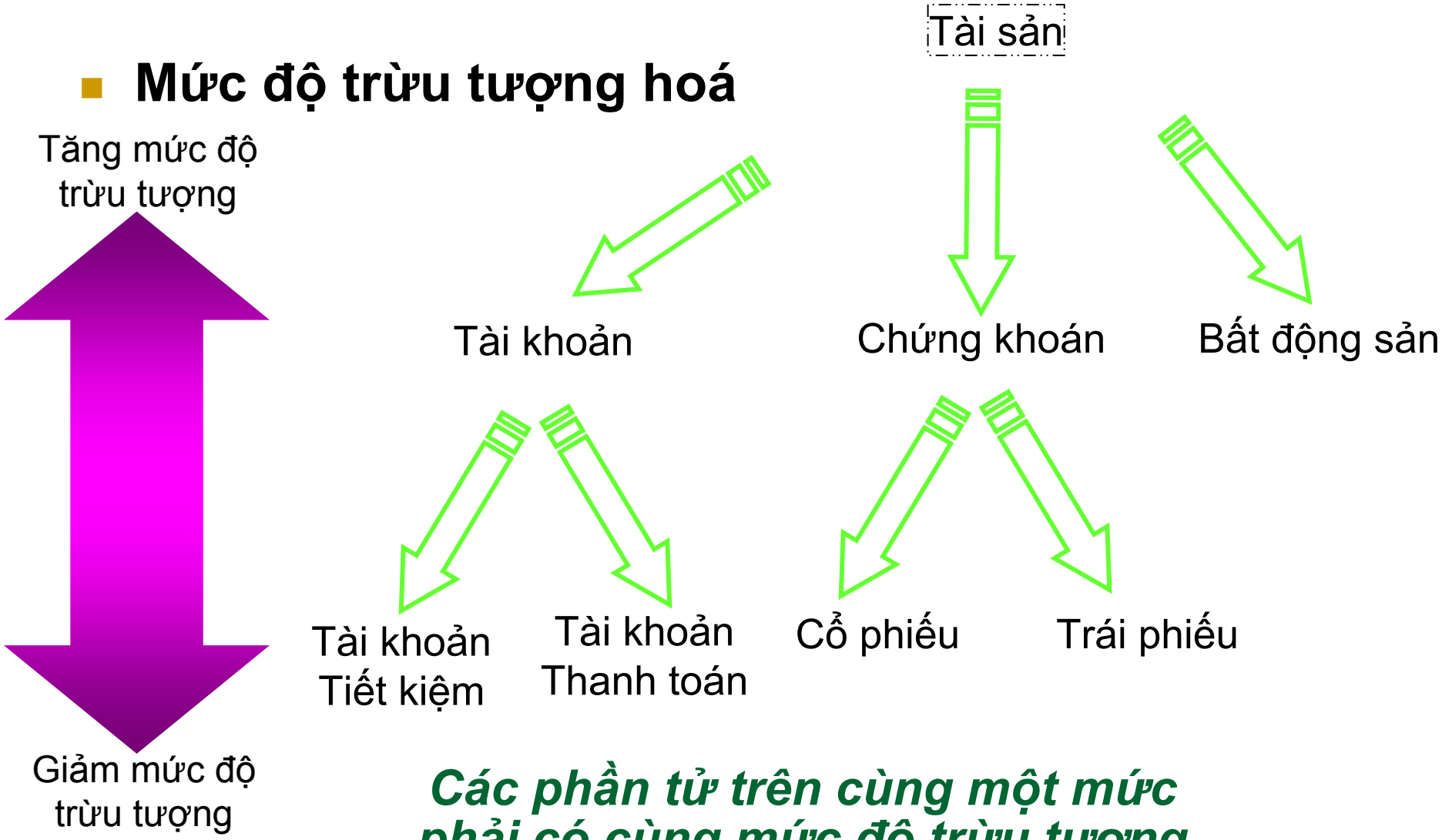
Thực hiện
đơn đặt hàng

Tính tiền

Quản lý được độ phức tạp

Sự phân cấp (Hierarchy) là gì?

- **Mức độ trừu tượng hoá**



Các phần tử trên cùng một mức phải có cùng mức độ trừu tượng

Hướng đối tượng

- Các nguyên tắc cơ bản của OO
- **Các khái niệm cơ bản của OO**
- Sức mạnh của OO
- Các cơ chế mô hình hoá cơ bản của UML

Các khái niệm cơ bản

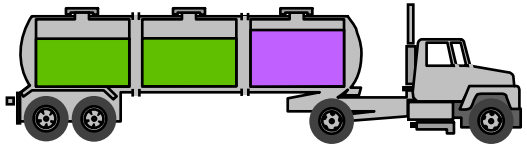
- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- Package
- Subsystem
- Relationships

Các khái niệm cơ bản

- **Object**
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- Package
- Subsystem
- Relationships

Object là gì?

- **Một cách không hình thức, một đối tượng biểu diễn một thực thể, dạng vật lý, khái niệm, hoặc phần mềm**
  - Thực thể vật lý

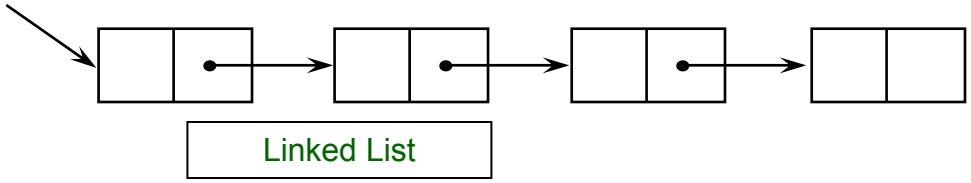


Truck

  - Thực thể khái niệm

Chemical Process

  - Thực thể phần mềm

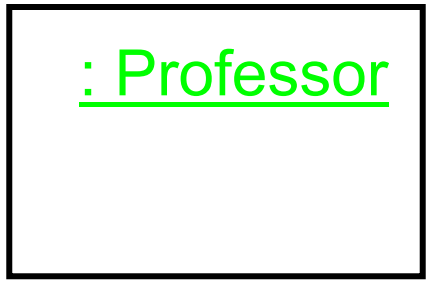


Một định nghĩa hiệu quả hơn

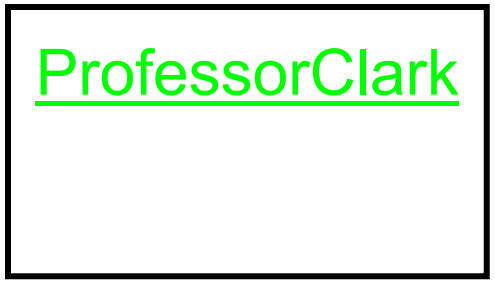
- **Một đối tượng là một khái niệm, sự trừu tượng, hoặc một vật với giới hạn rõ ràng và có ý nghĩa với một ứng dụng cụ thể**
- Một đối tượng có:
  - Trạng thái
  - Hành vi
  - Định danh (Identity)

Biểu diễn đối tượng

- **Một đối tượng được biểu diễn bởi một hình chữ nhật với tên được gạch dưới**

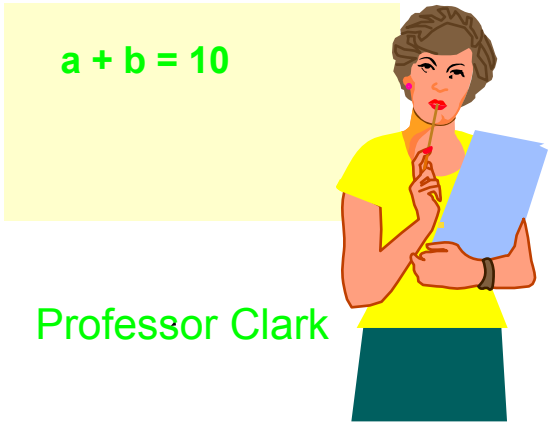


Chỉ có tên Class



Chỉ có tên đối tượng

Tên class và tên đối tượng



Các khái niệm cơ bản

- Object
- **Class**
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- Package
- Subsystem
- Relationships

Class là gì?

- Class là mô tả của một nhóm đối tượng có chung các thuộc tính (attributes), hành vi (operations), các mối quan hệ và ngữ nghĩa
  - Một đối tượng là một thể hiện của class
- Một class là sự trừu tượng mà trong đó:
  - Nhấn mạnh các tính chất quan trọng
  - Bỏ qua các tính chất khác

Nguyên tắc OO: Trừu tượng hóa

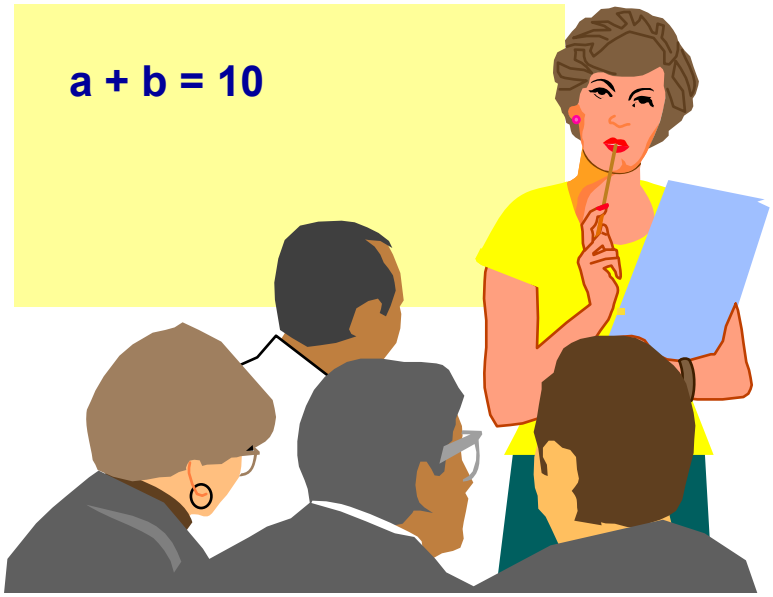
Ví dụ về Class

Class

Course

Properties

Tên
Địa điểm
Thời gian
Số tín chỉ
Giờ bắt đầu
Giờ kết thúc

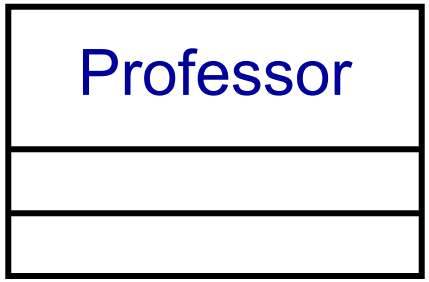


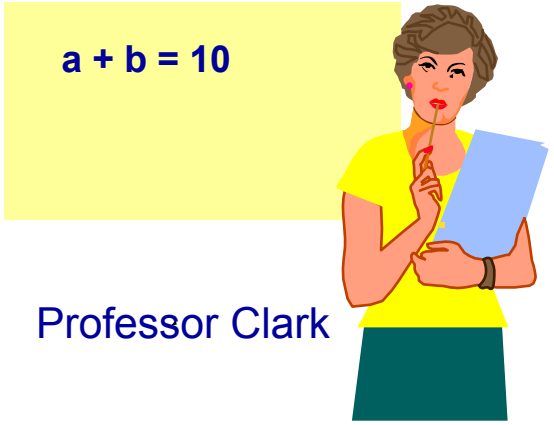
Behavior

Thêm một sinh viên
Huỷ một sinh viên
Lấy danh sách giáo sư
Xác định hết chỗ chưa

Biểu diễn Class

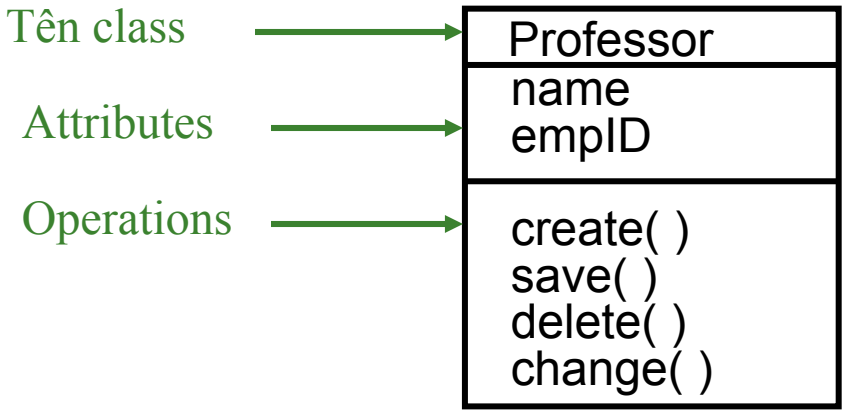
- Một class biểu diễn bằng một hình chữ nhật gồm ba phần





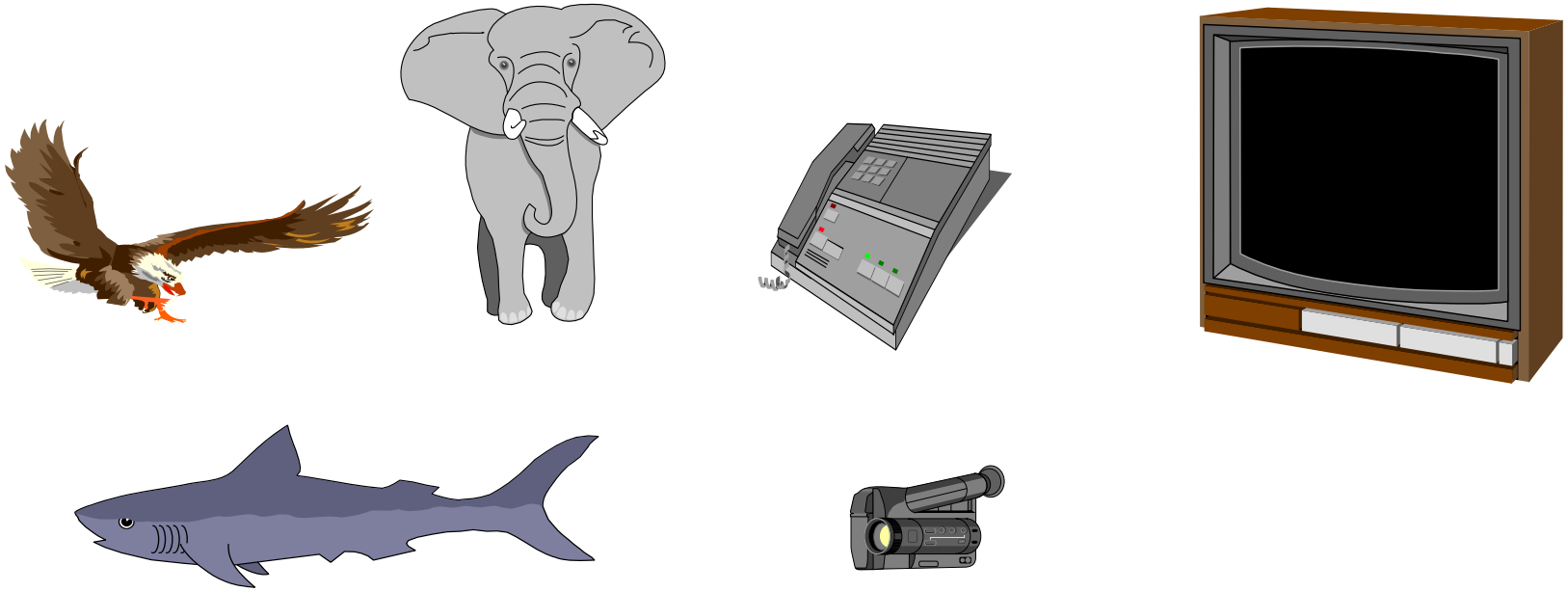
Các phần trong một Class

- Một class bao gồm ba phần
  - Phần đầu chứa tên class
  - Phần thứ hai cho thấy cấu trúc của lớp (attributes)
  - Phần thứ ba cho thấy các hành vi của lớp (operations)



Các lớp đối tượng

- **Bạn nhìn thấy có bao nhiêu class?**



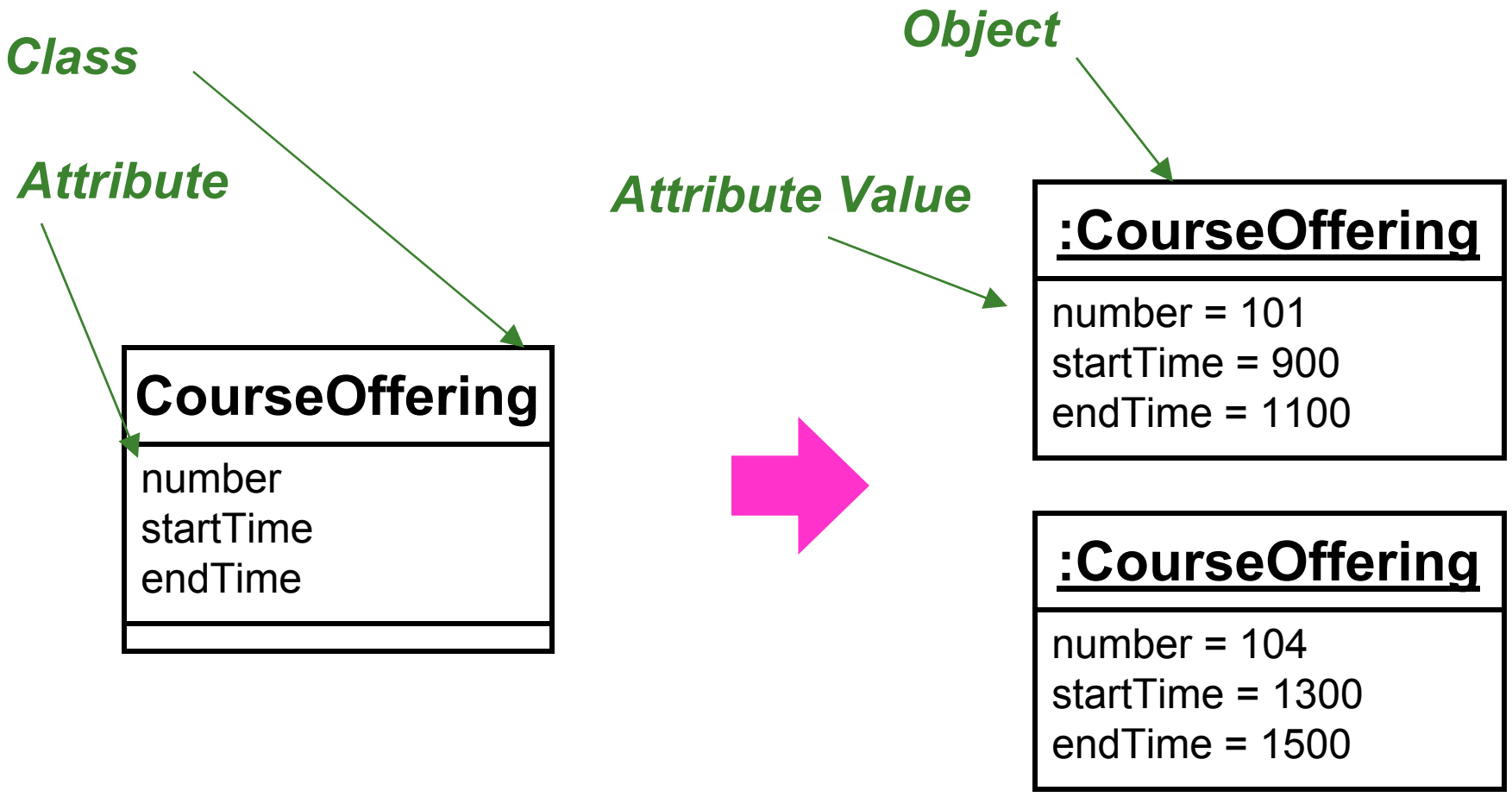
Quan hệ giữa class và đối tượng

- Một class là một định nghĩa trừu tượng của một đối tượng
  - Nó định nghĩa cấu trúc và hành vi của mỗi đối tượng trong lớp
  - Nó được dùng như khuôn mẫu để tạo đối tượng
- Các đối tượng được nhóm thành các class

Các khái niệm cơ bản

- Object
- Class
- **Attribute**
- Operation
- Interface (Polymorphism)
- Component
- Package
- Subsystem
- Relationships

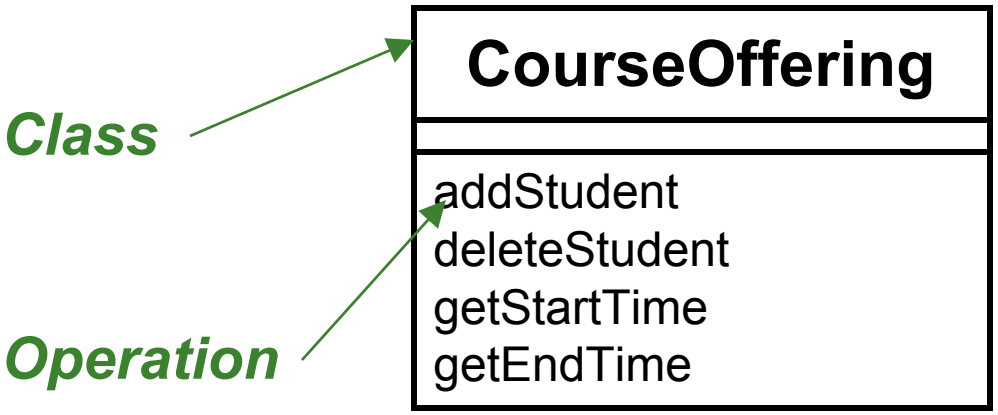
Thuộc tính (Attribute) là gì?



Các khái niệm cơ bản của Hướng đối tượng

- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- Package
- Subsystem
- Relationships

Hành vi (Operation) là gì?

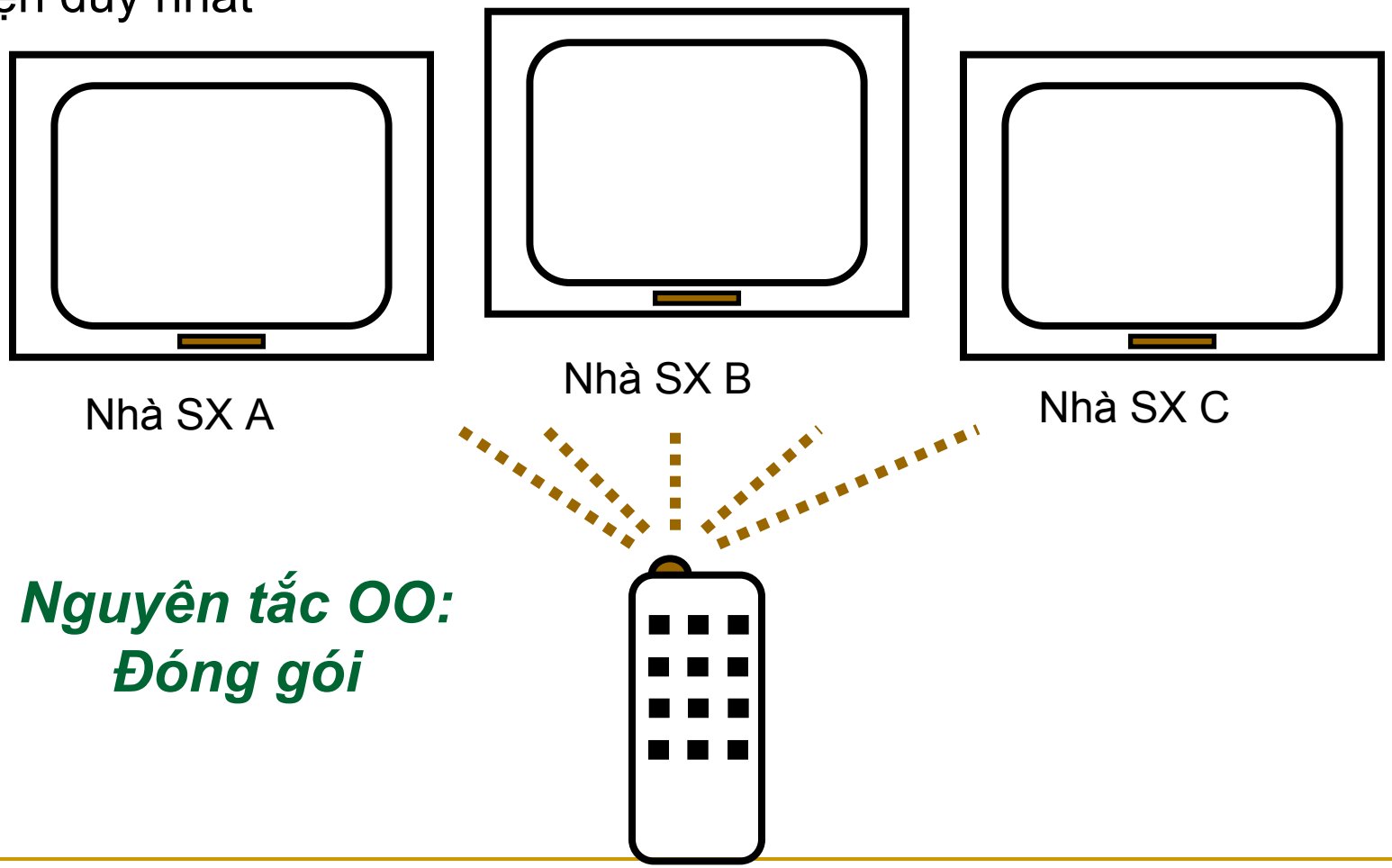


Các khái niệm cơ bản

- Object
- Class
- Attribute
- Operation
- **Interface (Polymorphism)**
- Component
- Package
- Subsystem
- Relationships

Polymorphism là gì?

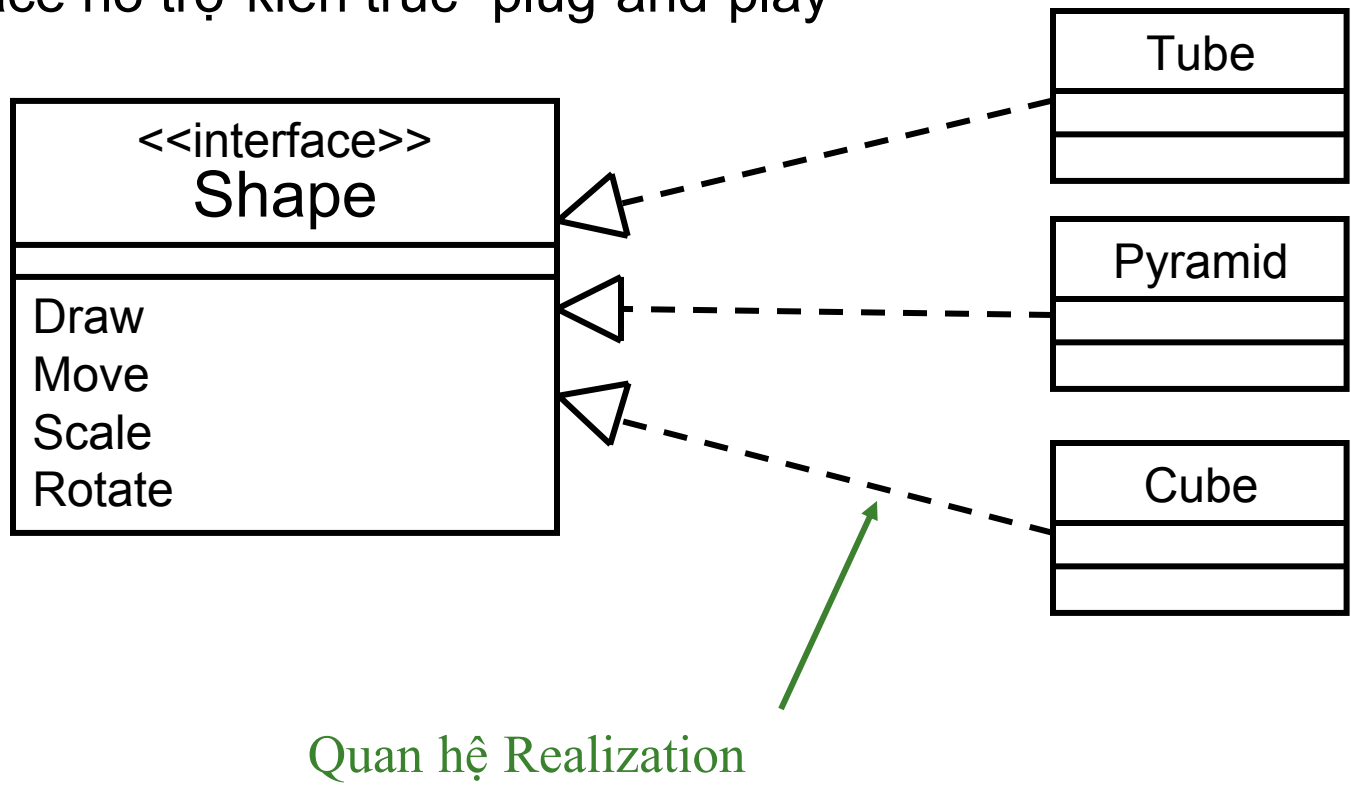
- Khả năng che dấu nhiều cài đặt khác nhau bên dưới một giao diện duy nhất



Nguyên tắc OO: Đóng gói

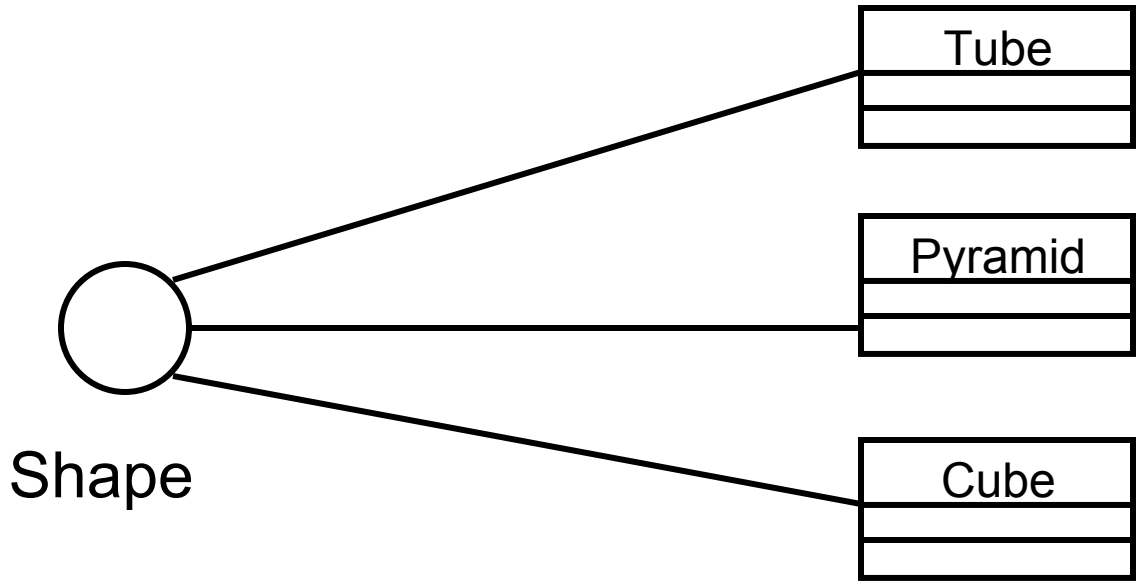
Interface là gì?

- Interface hình thức hoá polymorphism
- Interface hỗ trợ kiến trúc “plug-and-play”

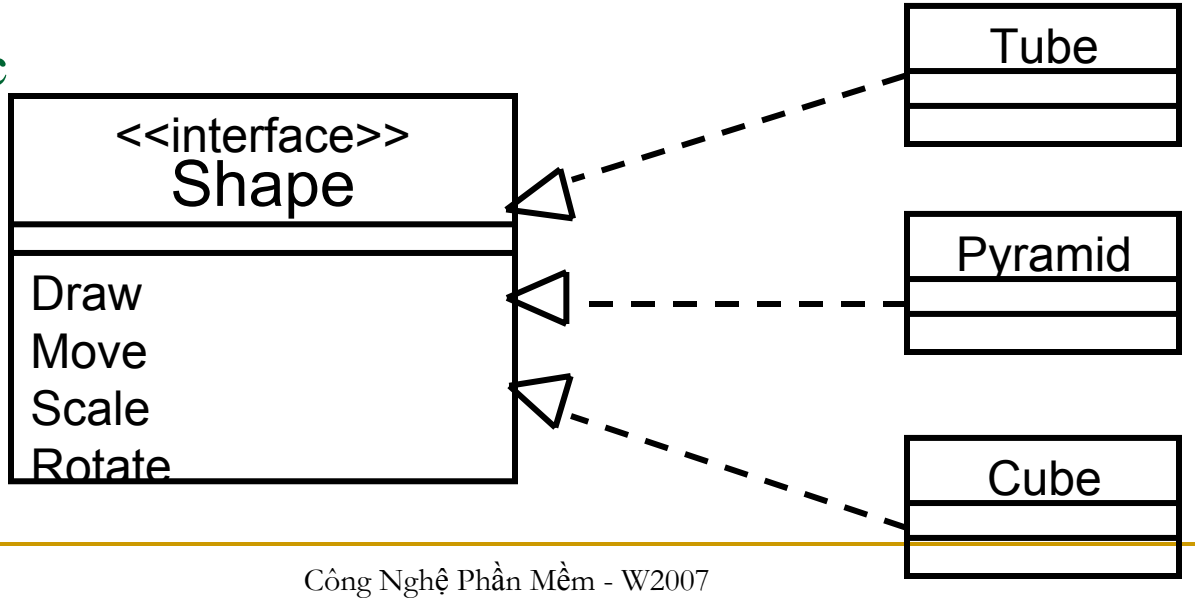


Biểu diễn Interface

Biểu diễn rút gọn



Biểu diễn chính tắc
(Class/Stereotype)



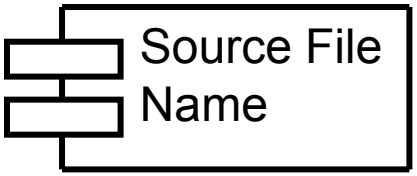
Các khái niệm cơ bản

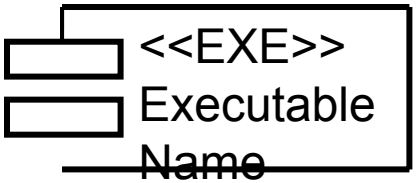
- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- **Component**
- Package
- Subsystem
- Relationships

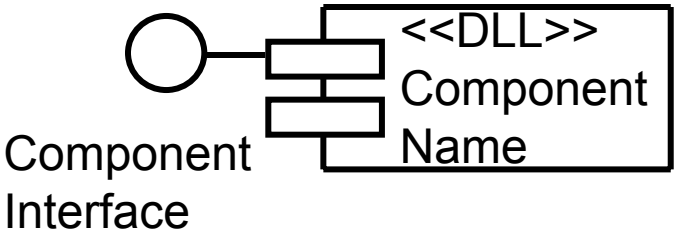
Component là gì?

- Một phần không tầm thường của hệ thống, gần như độc lập và có thể thay thế được, giữ một chức năng rõ ràng trong hệ thống
- Một component có thể là
  - Một source code component
  - Một run time components hoặc
  - Một executable component

Nguyên tắc OO: Đóng gói





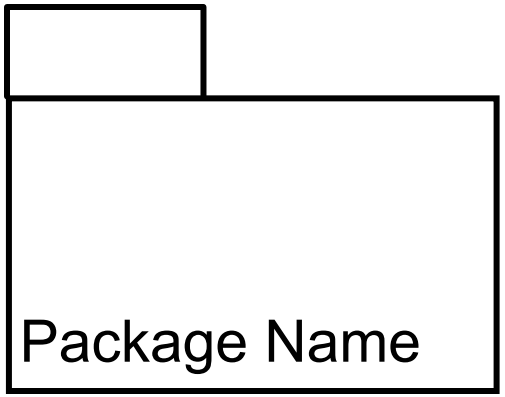


Các khái niệm cơ bản

- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- **Package**
- Subsystem
- Relationships

Package là gì?

- Một package là một cơ chế để tổ chức các phần tử vào thành các nhóm
- Một phần tử trong mô hình có thể chứa các phần tử khác



Nguyên tắc OO: Tính đơn thể

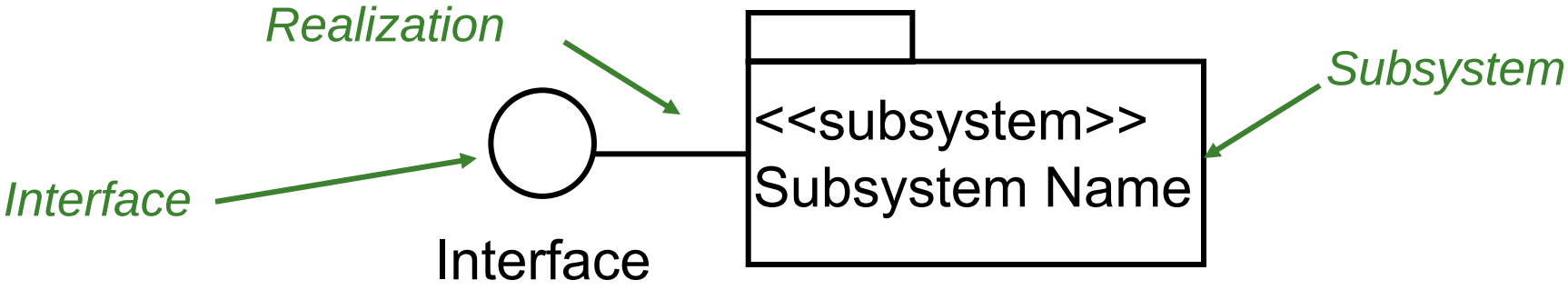
- Dùng để
  - Tổ chức mô hình đang phát triển
  - Một đơn vị trong quản trị cấu hình

Các khái niệm cơ bản

- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- Package
- **Subsystem**
- Relationships

Subsystem là gì?

- Tổ hợp của một package (có thể chứa các phần tử khác trong mô hình) và một class (có hành vi)
- Hiện thực hoá một hoặc nhiều interface định nghĩa cho hành vi của nó

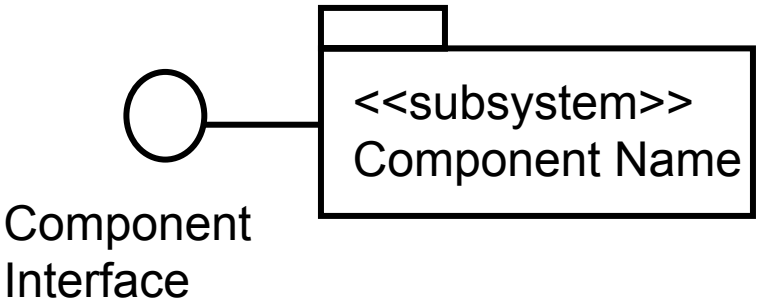


Nguyên tắc OO: Đóng gói và Tính đơn thể

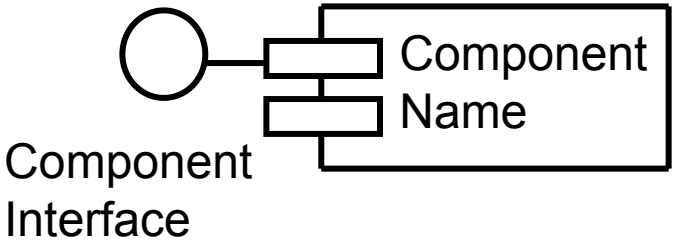
Subsystem và Component

- Component là thể hiện ở mức vật lý của một khái niệm trừu tượng trong thiết kế
- Subsystem có thể dùng để biểu diễn các component trong thiết kế

Design Model



Implementation Model



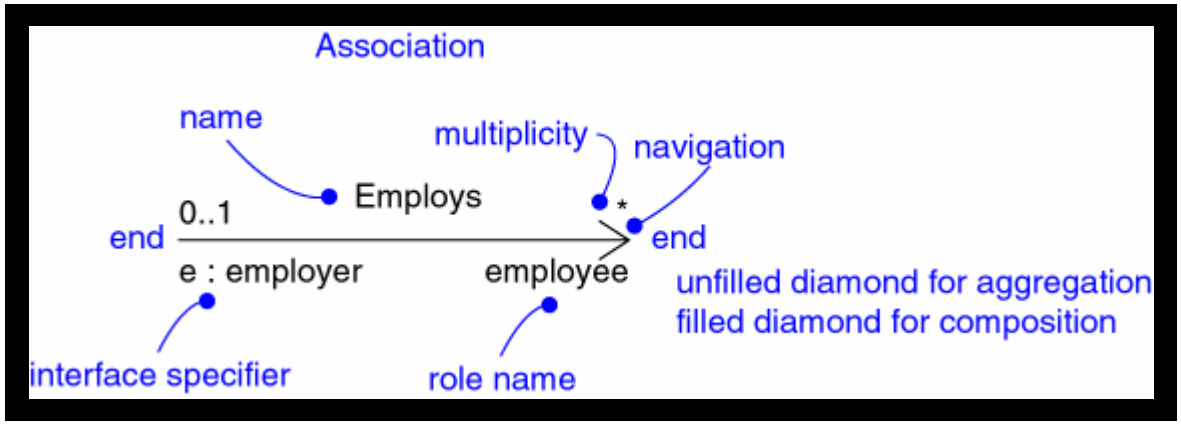
Nguyên tắc OO: Đóng gói và Tính đơn thể

Các khái niệm cơ bản của Hướng đối tượng

- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Component
- Package
- Subsystem
- **Relationships**

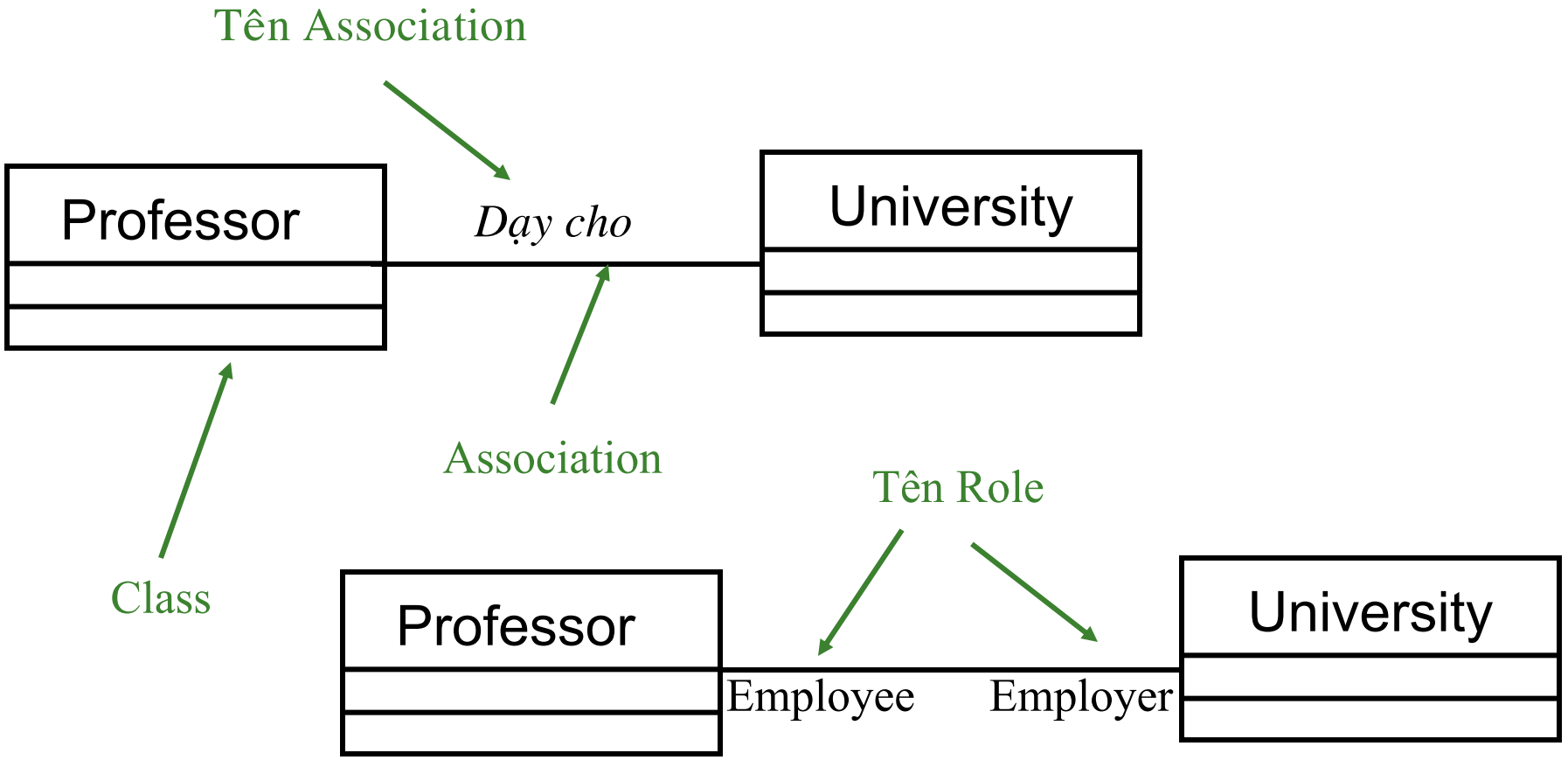
Các mối quan hệ

- Association (Kết hợp)
  - Aggregation (Thu nạp/Bao gộp/Tụ hợp)
  - Composition (Cấu thành)
- Dependency (Phụ thuộc)
- Generalization (Tổng quát hóa)
- Realization (Hiện thực hoá)



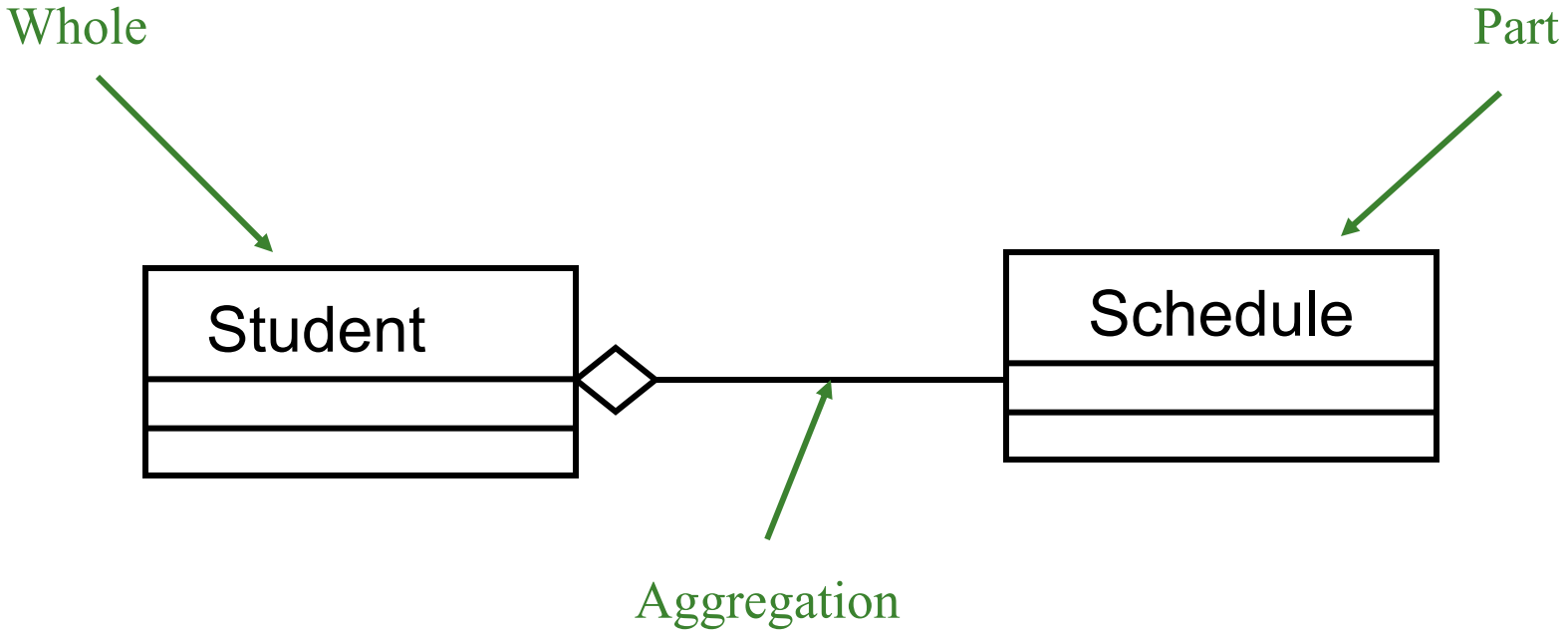
Mối quan hệ: Association

- Mô hình hoá một liên kết ngữ nghĩa giữa các class



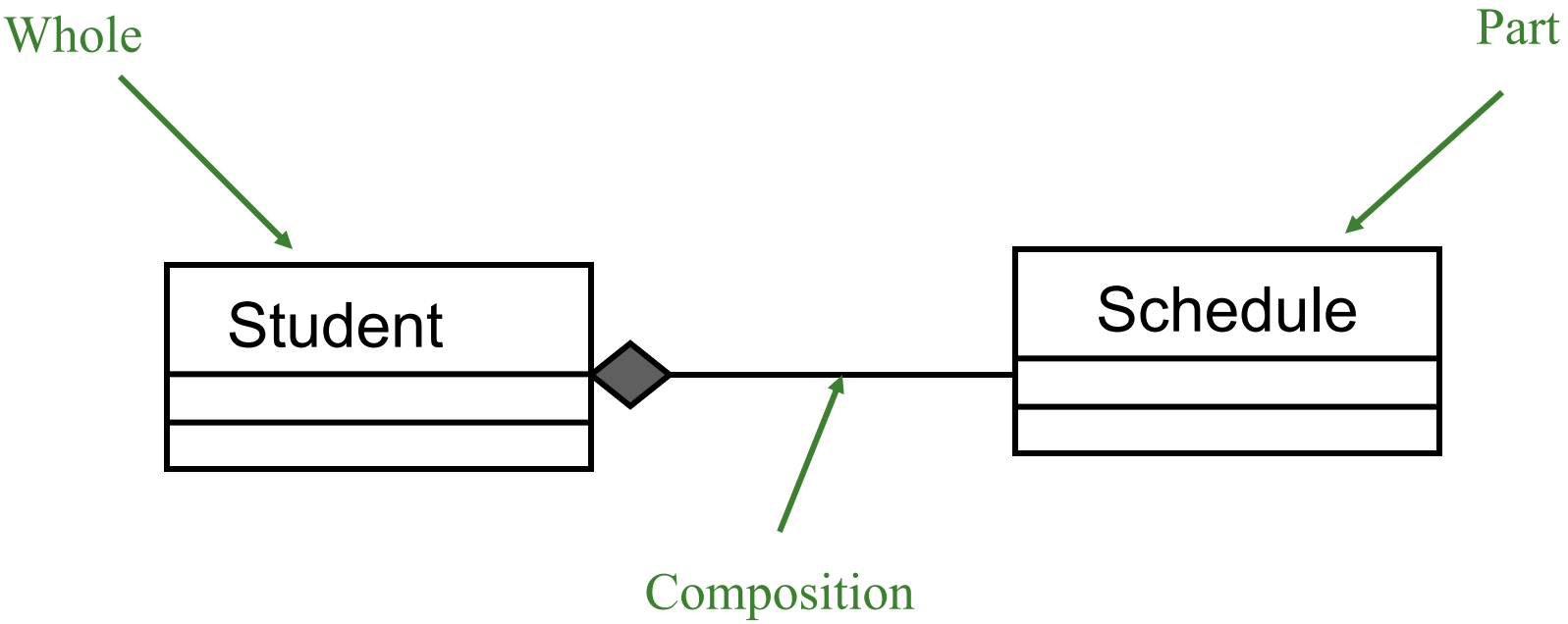
Mối quan hệ: Aggregation

- Một dạng đặc biệt của association mô hình hoá mối quan hệ toàn thể-bộ phận giữa một thực thể và các bộ phận của nó



Mối quan hệ: Composition

- Một dạng aggregation có tính sở hữu cao và cùng chu kỳ sống
  - Các bộ phận không thể sống lâu hơn thực thể



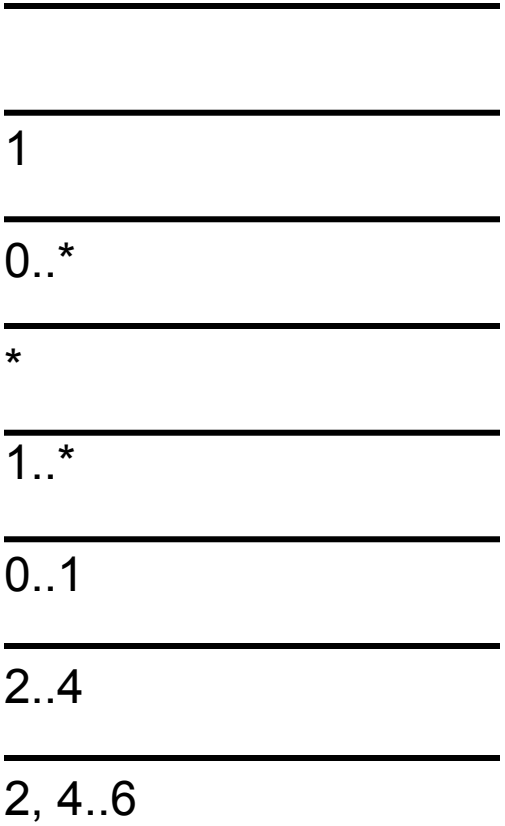
Association: Bản số và Chiều

- Bản số xác định số đối tượng tham gia vào một mối quan hệ
  - Số các thể hiện của một class quan hệ với MỘT thể hiện của một class khác
  - Được chỉ ra ở mỗi đầu của quan hệ association
- Association và aggregation mặc định là hai chiều, nhưng người ta thường giới hạn theo một chiều
  - Mũi tên được thêm vào để chỉ chiều của mối quan hệ

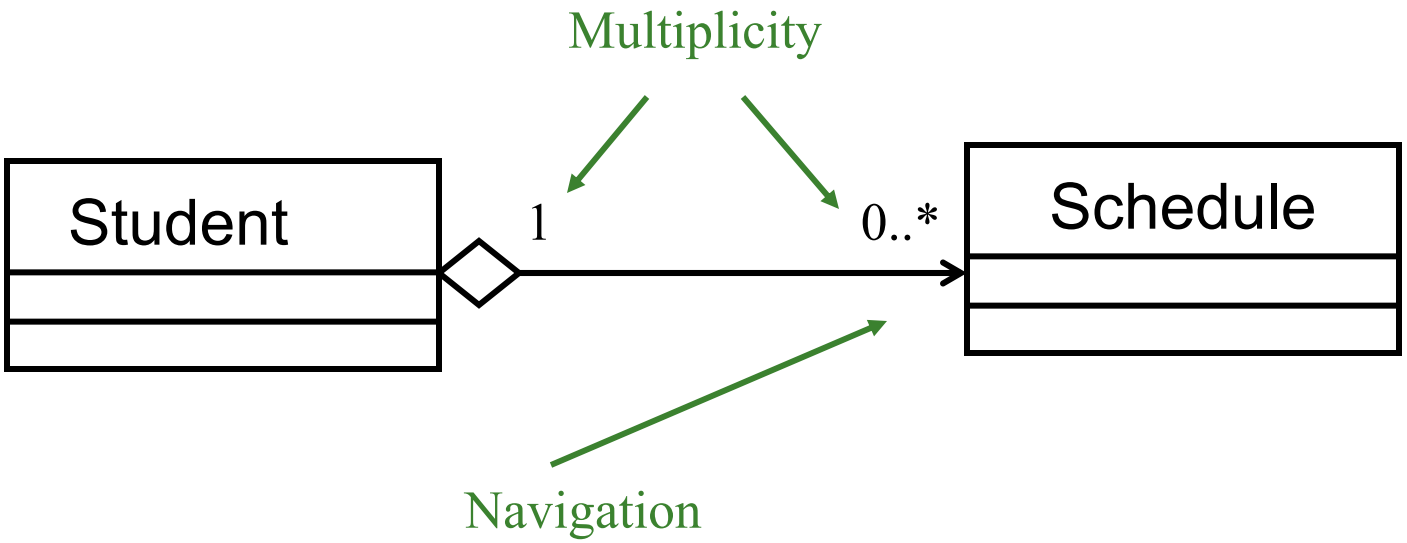
Association: Bản số

- Không xác định
- Chỉ một
- Không hoặc nhiều

- Một hoặc nhiều
- Không hoặc một
- Khoảng được chỉ định
- Các khoảng không liên tục

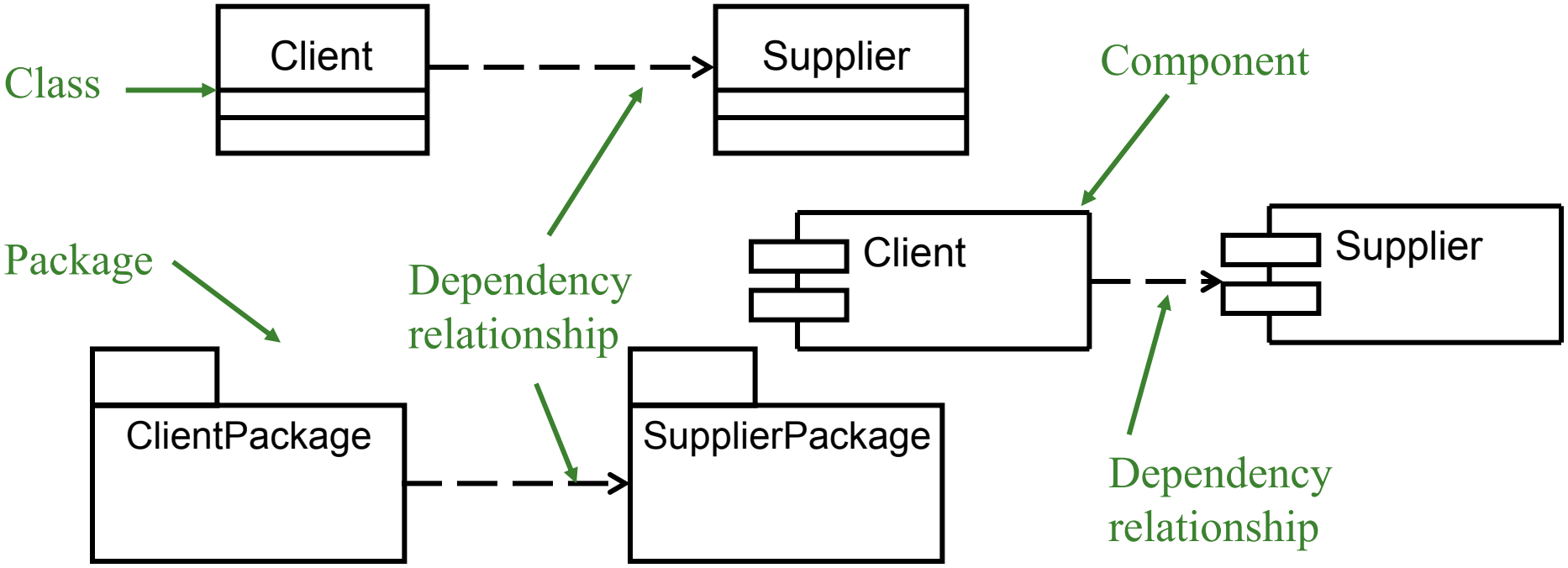


Ví dụ: Bản số và Chiều



Mối quan hệ: Dependency

- Quan hệ giữa hai phần tử trong mô hình mà thay đổi ở phần tử này có thể gây ra thay đổi ở phần tử kia
- Quan hệ "sử dụng", không cấu trúc

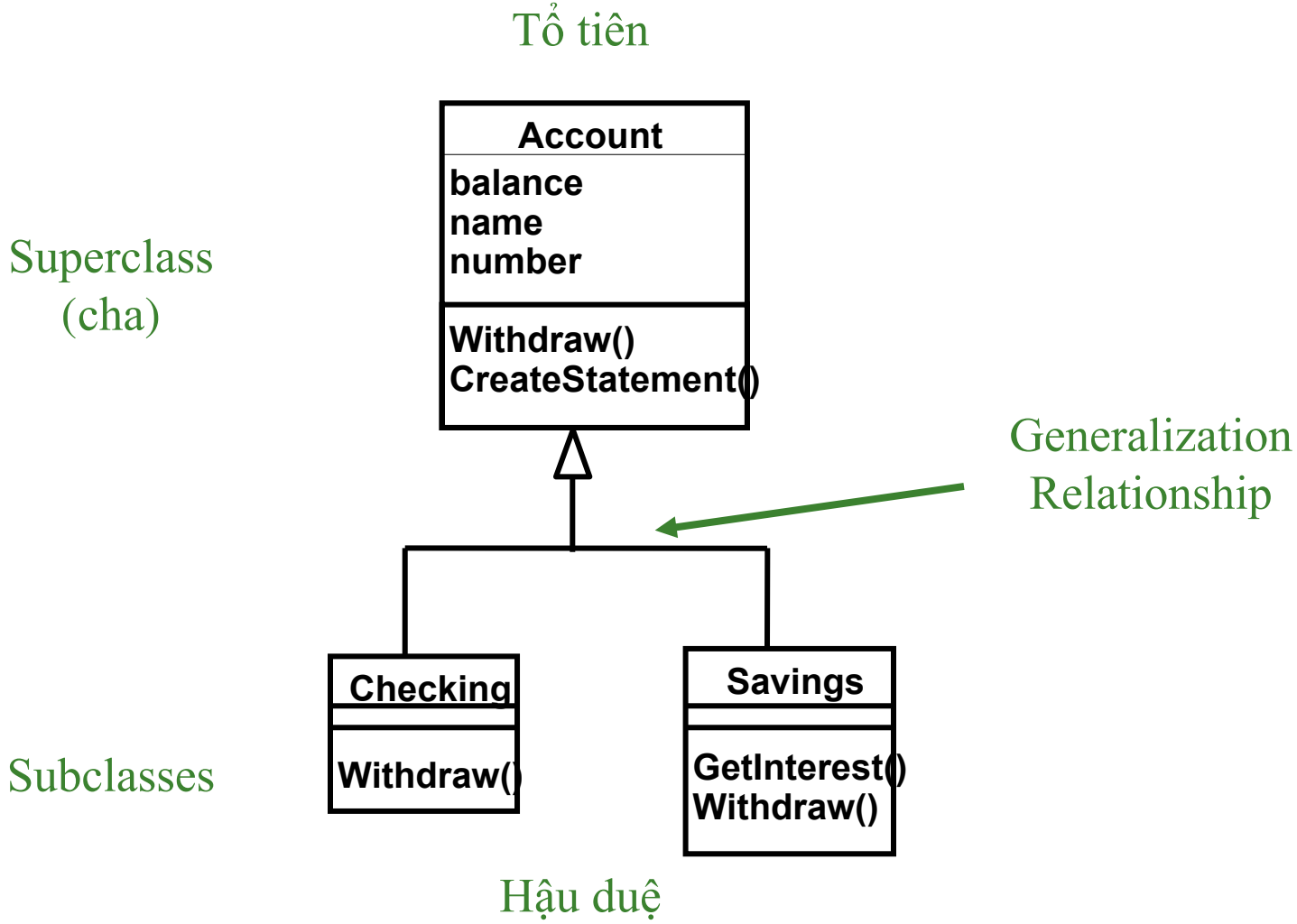


Mối quan hệ: Generalization

- Quan hệ giữa các class trong đó một lớp chia sẻ cấu trúc và/hoặc hành vi của một hoặc nhiều class khác
- Xác định một sự phân cấp các mức độ trừu tượng trong đó một subclass kế thừa từ một hoặc nhiều superclass
  - Đơn kế thừa
  - Đa kế thừa
- Generalization là quan hệ "là một dạng của"

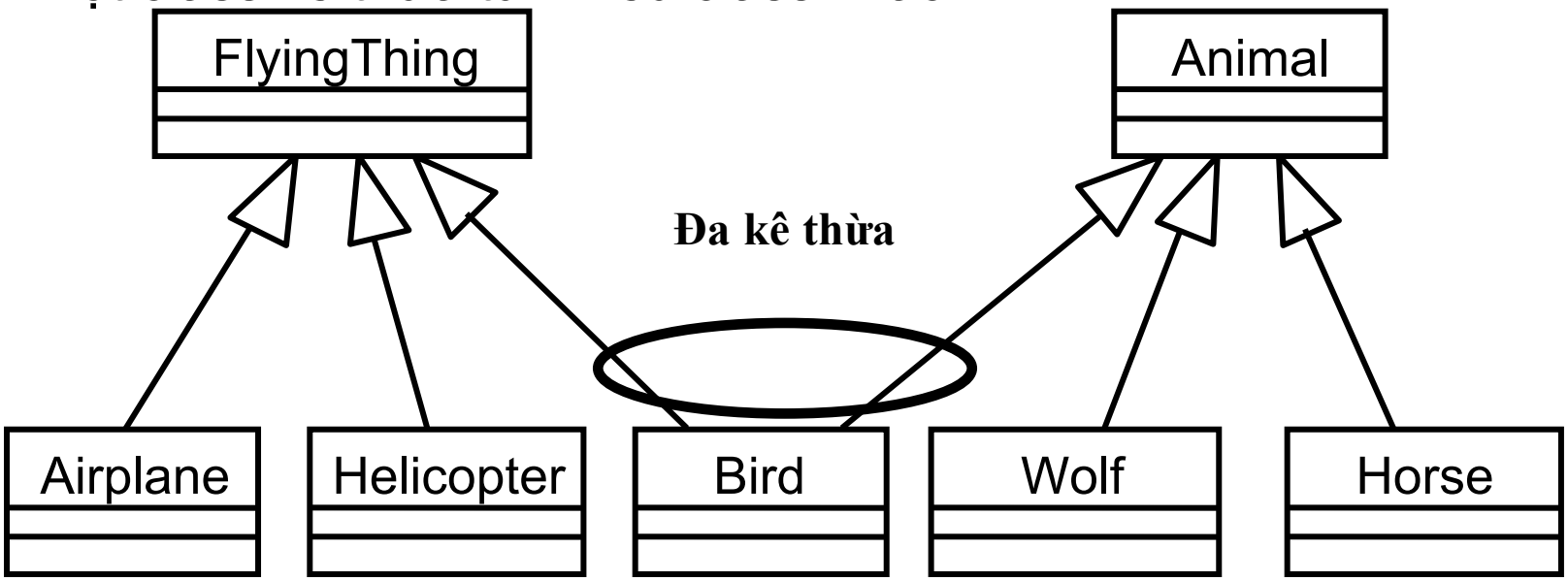
Ví dụ: Đơn kế thừa

- Một class kế thừa từ một class khác



Ví dụ: Đa kế thừa

- Một class kế thừa từ nhiều class khác



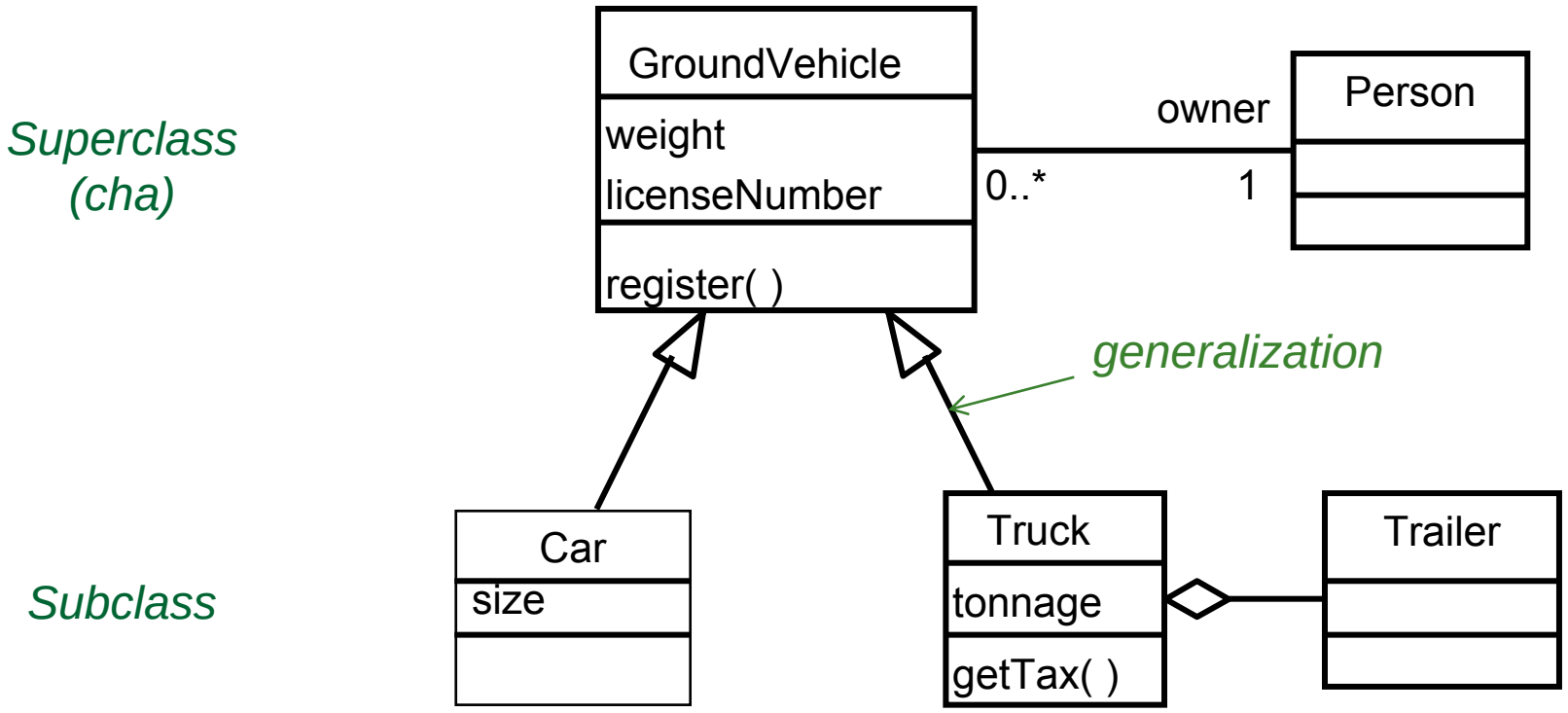
Chỉ sử dụng đa kế thừa khi thật cần, và luôn phải cẩn thận !

Cái gì được kế thừa?

- Một subclass kế thừa các thuộc tính, hành vi và các mối quan hệ từ cha nó
- Một subclass có thể:
  - Bổ sung thuộc tính, hành vi và các mối quan hệ
  - Định nghĩa lại các hành vi (nên cẩn thận!)
- Các thuộc tính, hành vi và các mối quan hệ chung được đặt ở mức cao nhất có thể trong cấu trúc phân cấp

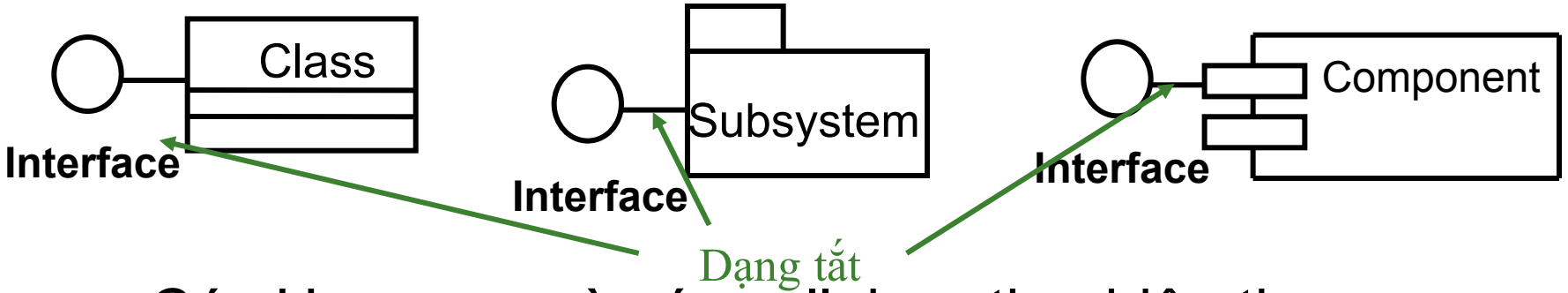
Sự kế thừa làm nổi bật các điểm tương đồng giữa các class

Ví dụ: Cái gì được kế thừa

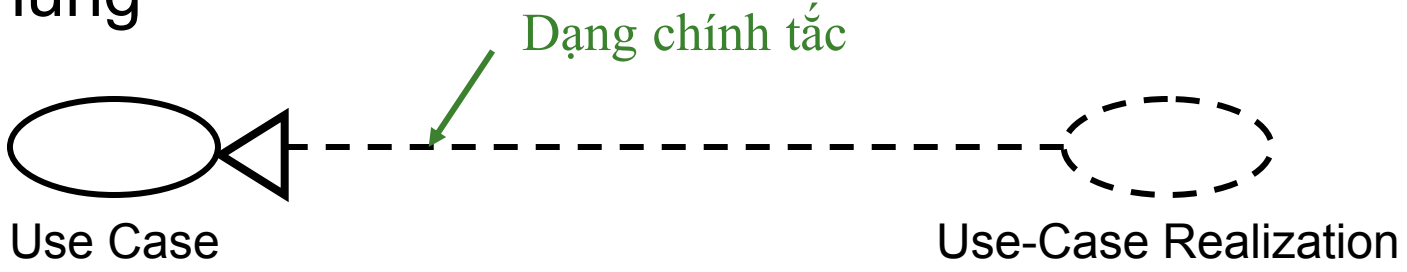


Mối quan hệ: Realization

- Một classifier đóng vai trò một hợp đồng mà một classifier khác đồng ý thực hiện
- Xuất hiện giữa:
  - Các Interface và các classifier hiện thực chúng



  - Các Use case và các collaboration hiện thực chúng



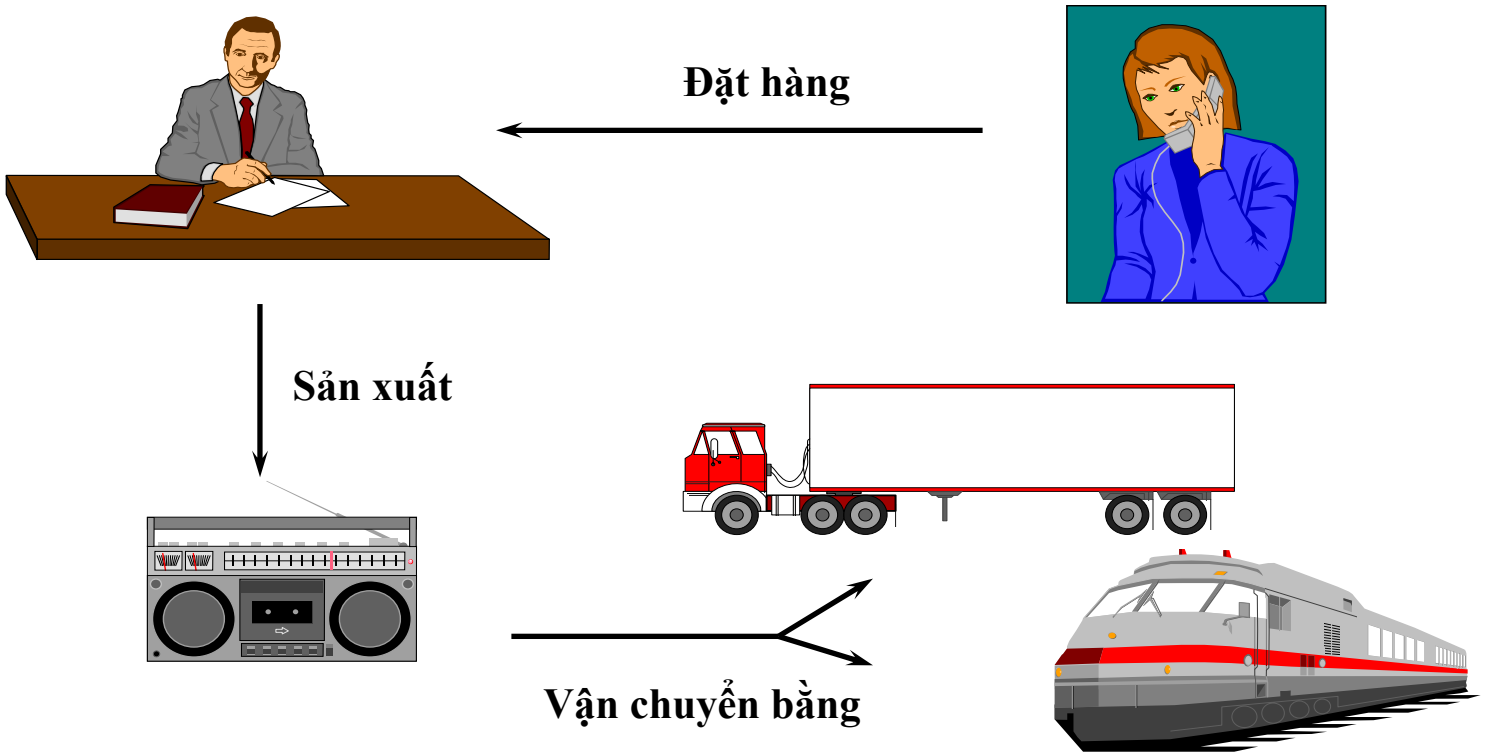
Giới thiệu về Hướng đối tượng: Các chủ đề

- Các nguyên tắc cơ bản của OO
- Các khái niệm cơ bản của OO
- **Sức mạnh của OO**
- Các cơ chế mô hình hoá cơ bản của UML

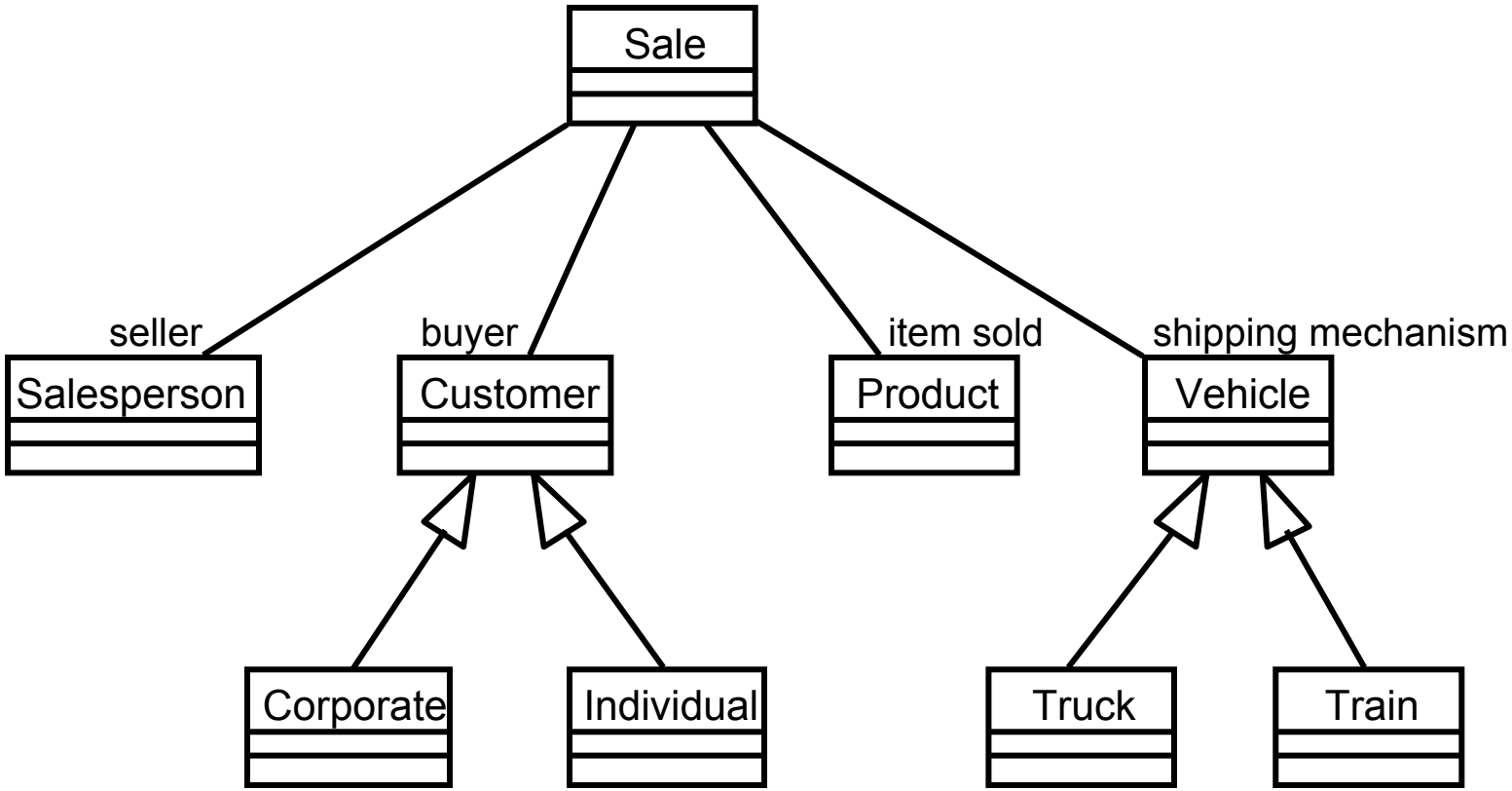
Sức mạnh của Hướng đối tượng

- Một mô hình chung
- Có tính dễ dùng lại
- Mô hình phản ánh chính xác thế giới thực
  - Mô tả chính xác hơn các tập dữ liệu và các xử lý
  - Được phân rã dựa trên các phân chia tự nhiên
  - Dễ hiểu và dễ bảo trì
- Tính ổn định
  - Một thay đổi nhỏ trong yêu cầu không gây ra sự thay đổi lớn trong hệ thống đang phát triển

Ví dụ: Sales Order System

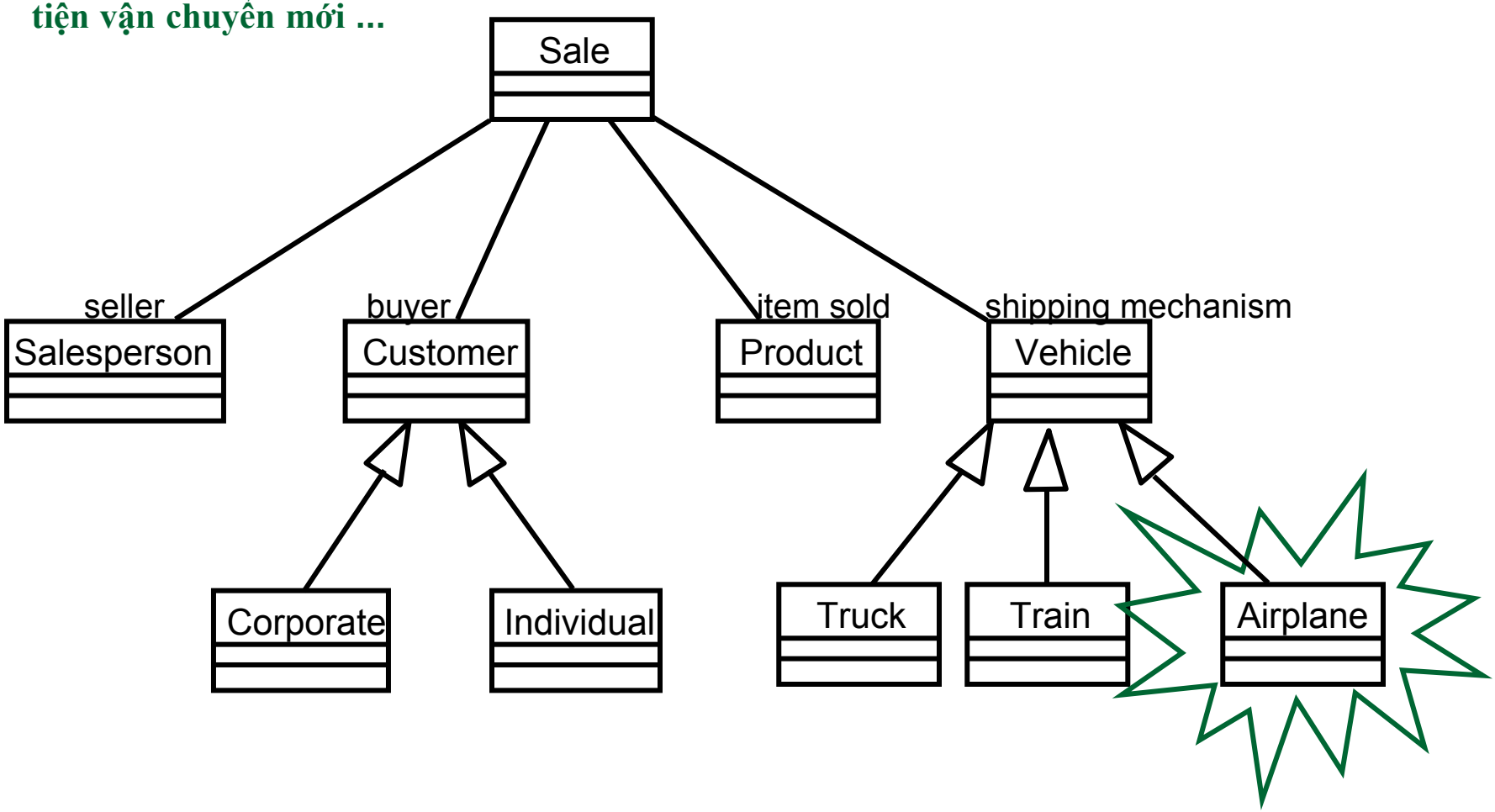


Class Diagram của ví dụ “bán hàng”



Hiệu ứng của sự thay đổi yêu cầu

Giả sử bạn cần phương tiện vận chuyển mới ...



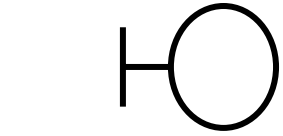
Việc thay đổi liên quan đến việc thêm 1 subclas mới

Hướng Đối Tượng

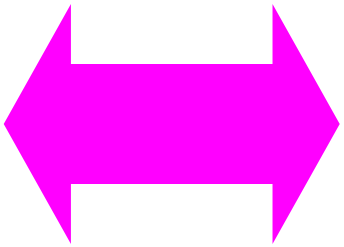
- Các nguyên tắc cơ bản của OO
- Các khái niệm cơ bản của OO
- Sức mạnh của OO
- **Các cơ chế mô hình hoá cơ bản của UML**

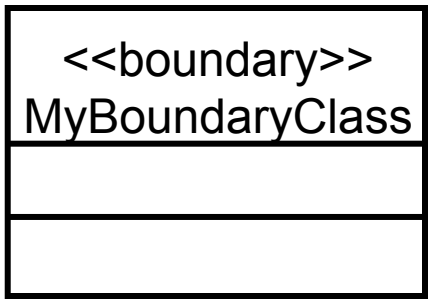
Các khuôn mẫu (Stereotype)

- Phân lớp và mở rộng các phần tử trong hệ thống ký hiệu UML
- Định nghĩa một phần tử của mô hình mới dựa trên một phần tử khác
- Có thể áp dụng cho mọi phần tử mô hình
- Được biểu diễn với tên đặt trong dấu << >> hoặc bằng các icon khác

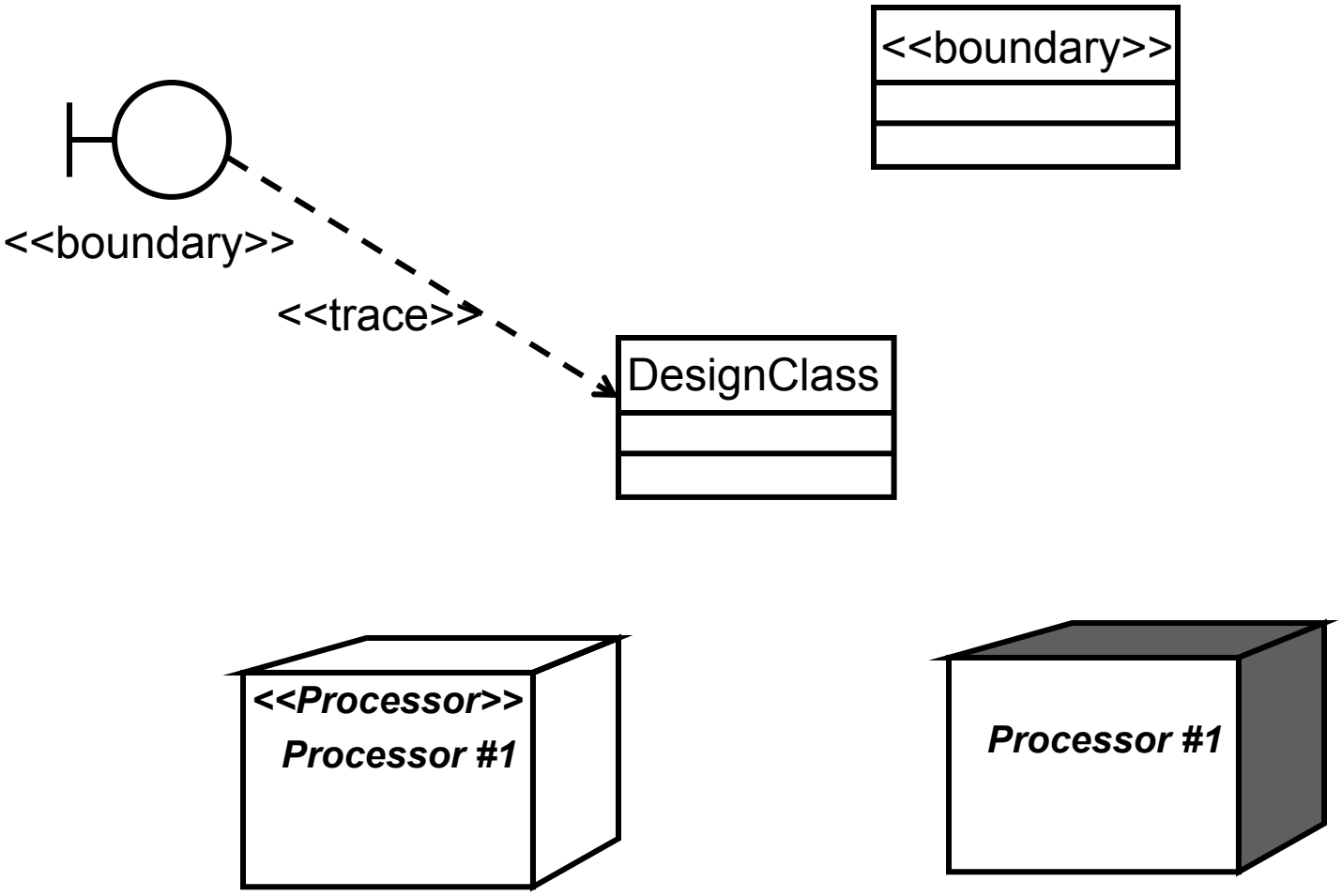


MyBoundaryClass



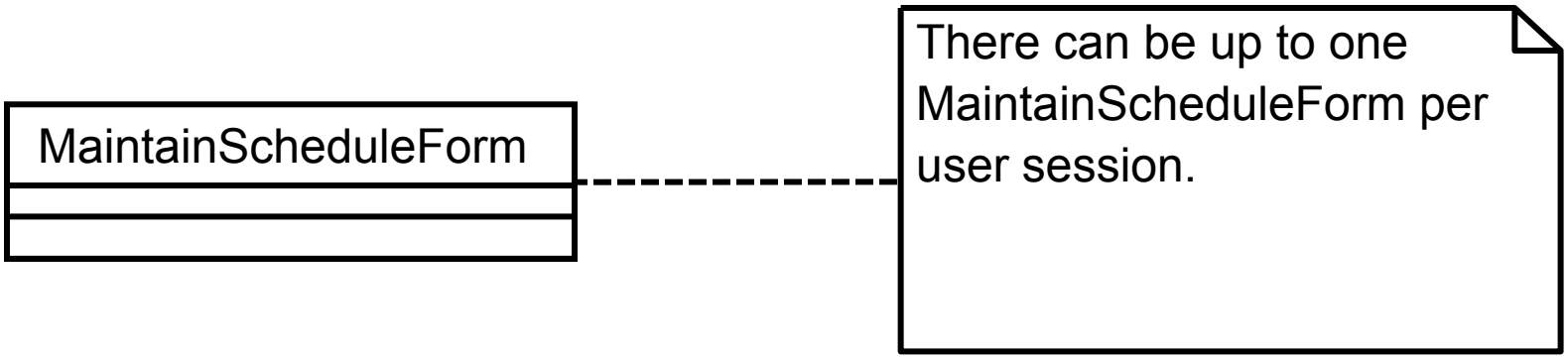


Ví dụ: Stereotype



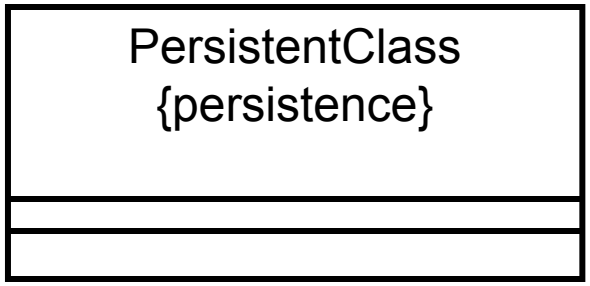
Các ghi chú (note)

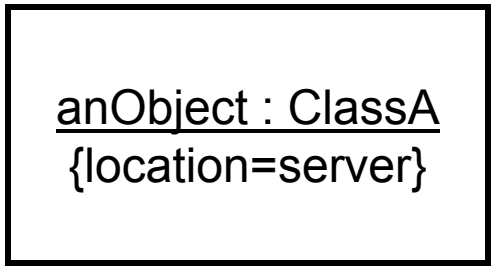
- Có thể đặt ghi chú cho mọi phần tử UML
- Ghi chú dùng để thêm thông tin cho các lược đồ
- Nó là hình chữ nhật bị bẻ góc
- Ghi chú có thể móc nối với một phần tử bằng một đường đứt nét



Các giá trị đính (Tagged Values)

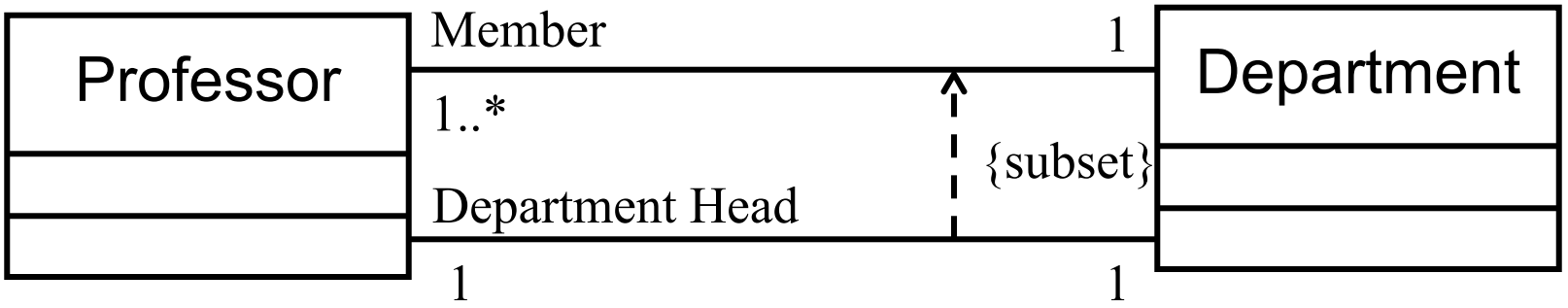
- Là sự mở rộng của các thuộc tính hoặc của các phần tử UML
- Là một số thuộc tính được định nghĩa sẵn bởi UML
  - Persistence
  - Location (chẳng hạn client, server)
- Là các thuộc tính có thể được tạo bởi các nhà mô hình hoá UML phục vụ cho mục đích bất kỳ





Các ràng buộc (Constraints)

- Hỗ trợ việc thêm các luật mới hoặc hiệu chỉnh các luật đang tồn tại



Q/A