PHÂN TÍCH THIẾT KẾ HƯỚNG ĐỐI TƯỢNG VỚI UML

Giảng viên: ThS. Nguyễn Đình Loan Phương

Email: phuongndl@uit.edu.vn

Giới thiệu môn học

- Lý thuyết : 45 tiết
- Thực hành, đồ án: 30 tiết
- Thang điểm:
  - Lý thuyết: 4/10
  - Đồ án : 4/10
  - Giữa kỳ : 2/10

Tài liệu tham khảo

- Giáo trình "Phân tích & thiết kế hướng đối tượng bằng UML" và "Qui trình phát triển phần mềm RUP" – ĐHKHTN, Dương Anh Đức
- Giáo trình "Phân tích & thiết kế hướng đối tượng bằng UML" – ĐHKHTN, Phạm Nguyễn Cương
- UML Fundamental, Dr. Ernest Cachia, 2001- 2004
- ….

- Các trang WEB
  - www.omg.org
  - www.rational.com
  - Các trang WEB về CASE Tools, OOAD & UML

Nội dung môn học

- Tổng quan về UML
- Xác định yêu cầu
- Tổng quan về phân tích thiết kế
- Mô hình hóa nghiệp vụ bằng UML

Giới thiệu

- Mục đích
  - Giới thiệu một số nét chính về lịch sử của UML, phạm vi và mục đích của UML và nội dung chính của môn học
- Nội dung chính
  - Động cơ đối với OOA/D
  - UML là gì, những gì không thuộc phạm vi của UML
  - Lịch sử của UML
  - Mục đích của UML
  - Các khung nhìn và lược đồ UML
  - Nội dung của môn học

Phân tích thiết kế hướng đối tượng

- "Tất cả các lược đồ chỉ là những bức tranh đẹp"
- "Người sử dụng sẽ không cảm ơn những bứctranh đẹp, những gì người sử dụng muốn là một phần mềm chạy tốt"
- Chúng ta không thể hiểu được các hệ thống phức tạp trong trạng thái nguyên vẹn của nó (phải chia nhỏ, mổ xẻ mô hình)
- Những biểu tượng được chọn lựa kĩ càng có thể:
  - Làm cho thông tin dễ tiếp cận ,dễ hiểu
  - Đưa ra cái nhìn thấu đáo vào hệ thống

Phương pháp luận

- Phương pháp là tập hợp các bước cần thực hiện để đạt được một mục đích nào đó.
- Phương pháp luận là môn khoa học chuyên nghiên cứu về các phương pháp.
- Hầu hết các tài liệu mô tả quá trình xây dựng phần mềm là phương pháp.
  - Phương pháp luận cấu trúc
  - Phương pháp luận hướng đối tượng

Phương pháp luận cấu trúc

- Phương pháp này còn gọi là phương pháp cổđiển
- Được nhìn nhận dưới sự phức tạp của chức năng hệ thống máy tính
- Chức năng được phân rã theo một hệ thống cấutrúc nhất định do người phân tích hệ thống đưa ra(cấu trúc phân nhánh, lặp...)
- Bao gồm mô hình quá trình chức năng cũng nhưcác mô hình dữ liệu. Sự liên kết giữa hai mô hìnhdữ liệu này còn đơn giản qua các mối liên kết và luồng thông tin từ quá trình chức năng này sang chức năng khác

Ưu/khuyết điểm của phương pháp

- Phân rã được chức năng, quá trình hoạt động phần mềm được thực hiện từng bước như thế nào, khá đơn giản và dễ hiểu.
- Việc dựa vào cấu trúc của quá trình chức năng dẫn đến khi chức năng hệ thống thay đổi, cấu trúc ấy có thể bị thay đổi rất nhiều, thậm chí phải thay đổi toàn bộ.
- Sự tách biệt giữa mô hình chức năng và mô hình dữ liệu dẫn đến những chức năng hoàn toàn giống nhau nhưng xử lý những kiểu dữ liệu khác nhau phải được viết lại liên tục.
- Thiếu linh động, phí phạm mã, khó mở rộng, khó thích nghi của phầm mềm xây dựng dựa vào phương pháp này.

Phương pháp luận hướng đối tượng

- Phương pháp này xác định rằng, cấu trúc thông tin trong hệ thống thông tin là ít thay đổi.
- Thế giới xung quanh dưới dạng đối tượng rời rạc. Phương pháp đưa ra khái niệm đối tượng để mô tả thôngtin.
- Giới thiệu thêm mối quan hệ kế thừa cha con. Các chức năng được xây dựng trên hệ cấu trúc đối tượng nhờ sự kếthợp thông tin và chức năng trên cấu trúc đối tượng.

Ưu/khuyết điểm của phương pháp

- Tăng cường tính sử dụng: qua mối liên kết kế thừa, không chỉ những hành vi, đoạn mã được tái sử dụng mà cả những thông tin tĩnh của lớp cha cũng được lớp con tái sử dụng.
- Tăng cường tính mở rộng: việc mở rộng chức năng có thể được thực hiện qua việc tạo lớp con. Vì vậy không ảnh hưởng đến cấu trúc thông tin đã có. Hơn thế nữa phần mềm trở nên linh động hơn hẳn.
- Do dựa vào cấu trúc thông tin thay vì chức năng: Nếu cấu trúc này thay đổi (lĩnh vực ứng dụng thay đổi) thì việc xây dựng lại một hệ thống khác là không tránh khỏi. Do đó phương pháp này thiếu sự linh động với sự thay đổi củathông tin.

Các khái niệm cơ bản - Trừu tượng hoá

- Xem xét các vấn đề cụ thể rồi thông qua quá trình tư duy trừu tượng để rút ra các khái niệm, nguyên tắc quyluật.
- Là công cụ chủ yếu để con người nhận thức và mô hình hoá thế giới.
- Trừu tượng hoá bao gồm hai quá trình chính : khái quát hoá và cụ thể hoá.

Khái quát hoá (Generalization)

- Khái quát hoá là quá trình tập trung vào những điểm chung cơ bản của các đối tượng, sự kiện công việc cụ thể, loại bỏ những điểm riêng có tính vụn vặt không quan trọng để tạo một khái niệm mới mang đặc tính chung liên quan đến tất cả những cái cụ thể ấy.
- Nói cách khác, khái quát hoá là phép đồng hoá những điểm chung của các sự việc cụ thể mà con người quan sát được.
- Khái quát hoá có thể được phân thành nhiều cấp độ khác nhau tuỳ vào mức độ khi thực hiện phép khái quát cũng như những điểm chung cơ bản được rút ra từ các đối tượng cụ thể.

Cụ thể hoá (refinement)

- Ngược với khái quát hoá là tinh chế hoá hay cụ thể hoá.
- Quá trình tinh chế là quá trình đi từ những khái niệm sự việc trừu tượng khái quát để mô tả chi tiết, cụ thể các đối tượng sự việc cụ thể hay các khái niệm sự việc trừu tượng ở mức thấp hơn. Nói cách khác, tinh chế là những cái khái quát trừu tượng cho các trường hợp cụ thể.
- Do tinh chế là quá trình ngược của khái quát hoá, nó bao gồm nhiều mức độ khác nhau tuỳ theo mức độ mô tả cụ thể vấn đề khái quát cũng như hướng mô tả.

Độ phức tạp

- Khi con người đối đầu với mọi bài toán họ luôn luôn phải đối đầu với độ phức tạp.
- Phương hướng chủ yếu là chia nhỏ đến mức có khả năng giải quyết được (cụ thể hoá) "chia để trị".
- Ngược lại, vấn đề khó khăn của phân rã độ phức tạp lại là khả năng tích hợp và quản lý các vấn đề nhỏ để giải quyết vấn đề lớn (khái quát hoá). Những bước phân rã ban đầu (tầm vĩ mô, khái quát nhất) thường tạo ra cái nhìn tổng quát nhất cho toàn bộ bài toán và được xem như là sườn cho toàn bộ bài toán.
- Những bước phân rã đó được gọi là phân rã kiến trúc. Mỗi phần của phần mềm được phân rã và tích hợp khác nhau tuỳ theo phương pháp luận khác nhau.

Che dấu thông tin

- Từ hai phương pháp cơ bản là trừu tượng hóa và phân rã độ phức tạp dẫn đến câu hỏi : "Làm thế nào để xây dựng được phần mềm với các mức độ phức tạp và trừu tượng khác nhau?"
- Nguyên tắc che dấu thông tin - thông tin của hai phần được che dấu nếu thật sự không cần thiết - được đưa ra nhằm đảm bảo các phần khác nhau của bài toán có thể tồn tại độc lập và bỏ qua những chi tiết không cần thiết trong mối quan hệ giữa chúng với nhau.
- Che dấu thông tin vì vậy đảm bảo được sự độc lập của từng phần của bài toán, chia bài toán thành từng phần trừu tượng khác nhau và giải quyết bài toán trên từng mức trừu tượng đó.

Chia sẻ và tái sử dụng

- Tính độc lập của từng bài toán có thể giúp cho bài toán được sử dụng lại trong nhiều hệ thống khác nhau mà không cần thiết phải giải lại.
- Kế thừa (inheritance)
  - Kế thừa cho phép chúng ta định nghĩa một lớp mới tương tự những lớp trước (đã có), ngoài ra bổ sung thêm những thuộc tính và các phương thức mô tả chi tiết hơn về một nhóm các đối tượng cụ thể.
  - Những lớp kế thừa gọi là lớp con (subclass) hoặc là lớp dẫn xuất(derived).
  - Những lớp được kế thừa còn gọi là lớp cha (supperclass).

Yêu cầu của mô hình hoá

- Không một mô hình đơn nào là đầy đủ, cần thiết phải có các khung nhìn khác nhau
  - Cách tốt nhất để tiếp cận mỗi hệ thống phức tạp là đi từ tập các mô hình nhỏ độc lập gần nhất
- Mỗi mô hình có thể được diễn đạt tại các mức độ phức tạp (độ thô) khác nhau
- Những mô hình tốt là cần thiết
  - Cho sự giao tiếp giữa những người thực hiện dự án với người sử dụng nó
  - Đảm bảo sự hợp lý, đúng đắn (soundness) trong kiến trúc

Lịch sử phát triển

- Vào những năm 1980s-các bước đầu tiên của lập trình hướng đối tượng
  - Smalltalk được chính thức chuyển từ phòng thí nghiệm ra phổ dụng
  - C++ được sinh ra
- Chuyển từ phương thức phân tích và thiết kế theo kiểu chức năng sang phương thức hướng đối tượng
- Các phương thức hướng đối tượng được phát triển vào những năm 1980s và giữa 1990s

Chuẩn hóa phương thức

- 1994- các phương thức đã gần như hoàn chỉnh và tương tự nhau
  - Cùng khái niệm (concepts): objects, classes, relationships, attributes, etc.
  - Cùng khái niện nhưng lại dùng kí hiệu (notation) khác nhau
  - Mỗi phương thức đều có các mặt mạnh và yếu
- Yêu cầu chuẩn hóa
  - Nhóm OMG (Object Management Group) đã thử và thất bại

Phương thức được hợp nhất

- Sự kiện lớn vào năm 1994 - James Rumbaugh liên kết với Grady Booch thành lập RationalSoftware Corporation
- Vào thời gian đầu, là sự hợp nhất của 2 phương pháp
  - Booch và OMT (Object Modelling Technique)
- Phương pháp này được gọi là Unified Method
- 1995 - Rational thông báo rằng đã mua Ivar, Jacobson's method Objectory. Jacobson muốn liên kết với Rational Software Corporation

3 nhân vật quan trọng

- Grady Booch
  - Làm việc cho ADA, sau đó là US Air Force Academy
  - Giám đốc khoa học của RSC
- Jame Rumbaugh
  - Nhân vật đứng đầu của General Electric
  - Tác giả của cuốn Object Modelling Technique
- Ivar Jacobson
  - Làm cho Ericson, sau đó sở hữu công ty Objective System ở Thụy Điển
  - Là cha đẻ của Use Case

Hợp nhất

- Ba nhân vật nói trên chuyển tên của "Unified method" thành "Unified Modelling Language"
- Mục đích của UML
  - Mô hình các hệ thống (không chỉ là phần mềm) bằng cách sử dụng các khái niệm hướng đối tượng
  - Thiết lập các hiện thực với khái niệm
  - Hướng tới các kế thừa phức tạp trong các hệ thống
  - Tạo ra một ngôn ngữ mô hình khả dụng cho cả người và máy

Công bố UML

- Một cách duy nhất để giành được sự chấp thuận của các phương pháp là đem UML ra cộng đồng
- Năm 1996: thiết lập cộng đồng UML
  - Dưới sự lãnh đạo của Rational SC
  - Một số công ty lớn khác như: I-Logic, Intellicorp, IBM,….

Động cơ thúc đẩy sử dụng UML

- Sự chấp nhận UML - Những tập đoàn thành viên (cộng tác) của UML

Microsoft	I-Logix	MCI
Oracle	IntelliCorp	Systemhouse
IBM	Softteam	ObjectTime
DECHP	Sterling	PlatiumTechnology
TI	Software	Ptech
Unisys	ICON Computing	ReichTechnologies
		...

Các hoạt động tiến tới chuẩn hóa

- Mục đích căn nguyên của UML là để trở thành một chuẩn hóa trên thực tế
- "Chấp thuận" bằng cách được sử dụng rộng rãi
- Nhưng OMG muốn có một chuẩn hóa thực sự
  - Yêu cầu một chuẩn hoá chính thức hơn là chỉ trên thực tế
  - Các định nghĩa rõ ràng của cú pháp và ngữ nghĩa
- OMG Task force được thiết lập cho vấn đề chuẩn hóa

UML và các khái niệm

- UML là một ngôn ngữ mô hình sử dụng các kí hiệu cho việc viết tài liệu, phân tích, thiết kế và thực hiện tiến trình phát triển hệ thống hướng đối tượng.
- Có 4+1 khung nhìn: Logical, Component, Process, Deployment và Use case

UML là gì?

- UML là một cách phân tích và thiết kế mô hình theo hướng đối tượng
  - Hiểu theo cách thông thường, UML bao gồm các mô hình đặc trưng cho việc phân tích và thiết kế
- UML không phải là một phương pháp, đơn thuần nó chỉ là một ngôn ngữ kí hiệu
  - Là một tập các kí hiệu
  - Là một tập các luật (cú pháp, ngữ nghĩa, kiểm tra) cho việc sử dụng các kí hiệu
  - Dùng để hiển thị, đặc tả, xây dựng, làm tài liệu

UML-NN mô hình hướng đối tượng

- UML được tạo ra phục vụ cho việc mô hình hóa hướng đối tượng
- Hướng đối tượng sản sinh ra các mô hình thể hiện một lĩnh vực
  - Một lĩnh vực kinh doanh, ví dụ: banking
  - Các thuật ngữ và đối tượng của lĩnh vực (ví dụ: tiền, séc)
- UML có thể được sử dụng để mô hình nhiều kiểu hệ thống khác nhau.

UML-Ngôn ngữ mô hình hoá trực quan

- Ngôn ngữ mô hình hóa trực quan là một phát minh lớn của những năm 1990s trong việc thiết kế phần mềm.
- Thể hiện trực quan là cách tốt nhất để giao tiếp và quản lí độ phức tạp
- Làm cho mô hình hóa ngày càng gần hơn với việc cài đặt

Ưu điểm của UML

- UML hợp nhất các mô hình của Booch, OMT, và Jacobson
  - Thống nhất hầu hết các khái niệm của 3 phương pháp trên
  - Thêm các ký hiệu và khái niệm chưa có ở trong 3 phương pháp này

Những điểm ngoài phạm vi UML

- UML không là một phương pháp
- UML không xác định/hướng vào (address) toàn bộ quá trình
- UML không quy định cách tiếp cận vào việc xác định các lớp, các phương thức và phân tích các mô hình...
- UML không bao gồm bất kỳ quy tắc thiết kế hay cách thức giải quyết vấn đề nào

Mục tiêu của UML

- Cung cấp một ngôn ngữ mô hình hoá trực quan có sẵn và gợi tả (ready to use,expressive), từ đó có thể phát triển và thay đổi các mô hình một cách hiệu quả
- Cung cấp các kỹ thuật chuyên môn để mở rộng các khái niệm cốt lõi (core concepts)
- Độc lập với các ngôn ngữ lập trình riêng biệt (particular) và các tiến trình phát triển

Mục tiêu của UML

- Cung cấp kiến thức cơ bản để hiểu ngôn ngữ mô hình hoá
- Ủng hộ/khuyến khích sự phát triển của môi trường sử dụng công cụ hướng đối tượng (the OO tools market)
- Hỗ trợ các khái niệm cho sự phát triển ở mức độ cao hơn như là sự cộng tác (collaborations), khung (frameworks), mẫu (patterns), thành phần (components)…

UML và sự phát triển phần mềm

- Giai đoạn nghiên cứu sơ bộ
- Giai đoạn phân tích
- Giai đoạn thiết kế
- Giai đoạn xây dựngThử nghiệm

Giai đoạn nghiên cứu sơ bộ

- UML đưa ra khái niệm Use Case để xác định các yêu cầu của khách hàng (người sử dụng).
  - Dùng biểu đồ Use case (Use Case Diagram) để nêu bật mối quan hệ cũng như sự giao tiếp với hệ thống.
- Qua phương pháp mô hình hóa Use case, các tác nhân (Actor) bên ngoài quan tâm đến hệ thống sẽ được mô hình hóa song song với chức năng mà họ đòi hỏi từ phía hệ thống (tức là Use case).
  - Mỗi Use case được mô tả trong tài liệu sẽ đặc tả các yêu cầu của khách hàng: khách hàng chờ đợi điều gì từ phía hệ thống mà không hề để ý đến việc chức năng này sẽ được thực thi ra sao.

Giai đoạn phân tích

- Quan tâm đến quá trình trừu tượng hóa đầu tiên (các lớp và các đối tượng), cơ chế hiện hữu trong phạm vi vấn đề.
- Sau khi nhận biết được các lớp thành phần và mối quan hệ giữa chúng, các lớp cùng các mối quan hệ sẽ được miêu tả bằng công cụ biểu đồ lớp (class diagram).
- Sự cộng tác giữa các lớp nhằm thực hiện các Use case cũng sẽ được miêu tả nhờ vào các mô hình động (dynamic models).

Giai đoạn thiết kế

- Kết quả của giai đoạn phân tích được mở rộng thành một giải pháp kỹ thuật.
  - Bổ sung các lớp mới => tạo thành hạ tầng cơ sở kỹ thuật: Giao diện người dùng, các chức năng để lưu trữ các đối tượng trong ngân hàng dữ liệu, giao tiếp với các hệ thống khác, giao diện với các thiết bị ngoại vi và các máy móc khác trong hệ thống, ...
  - Các lớp thuộc phạm vi vấn đề có từ giai đoạn phân tích sẽ được "nhúng" vào hạ tầng cơ sở kỹ thuật này, tạo ra khả năng thay đổi trong cả hai phương diện: Phạm vi vấn đề và hạ tầng cơ sở.

  => Kết quả là bản đặc tả chi tiết cho giai đoạn xây dựng hệ thống.

Giai đoạn xây dựng – lập trình

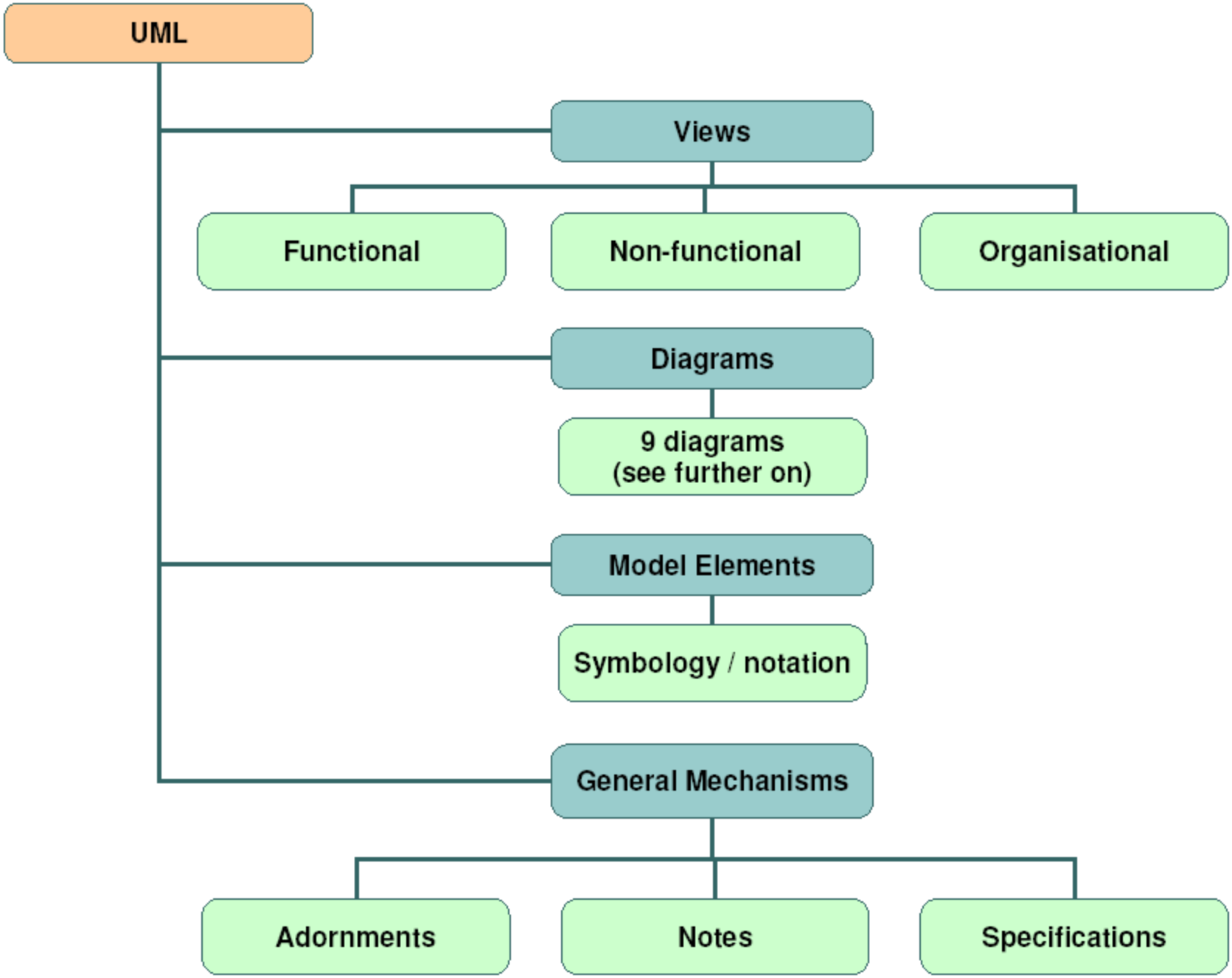
- Các lớp của giai đoạn thiết kế sẽ được biến thành những dòng code cụ thể trong một ngôn ngữ lập trình hướng đối tượng cụ thể
  - Phụ thuộc vào khả năng của ngôn ngữ được sử dụng, đây có thể là một công việc khó khăn hay dễ dàng.
  - Khi tạo ra các mô hình phân tích và thiết kế trong UML, tốt nhất nên cố gắng né tránh việc ngay lập tức biến đổi các mô hình này thành các dòng code.

Thử nghiệm

- Một hệ thống phần mềm thường được thử nghiệm qua nhiều giai đoạn và với nhiều nhóm thử nghiệm khác nhau.
- Các nhóm sử dụng nhiều loại biểu đồ UML khác nhau làm nền tảng cho công việc của mình
  - Thử nghiệm đơn vị sử dụng biểu đồ lớp (class diagram) và đặc tả lớp
  - Thử nghiệm tích hợp thường sử dụng biểu đồ thành phần (component diagram) và biểu đồ cộng tác (collaborationdiagram)
  - Thử nghiệm hệ thống sử dụng biểu đồ Use case (use case diagram) để đảm bảo hệ thống có phương thức hoạt động đúng như đã được định nghĩa từ ban đầu

Các thành phần của UML

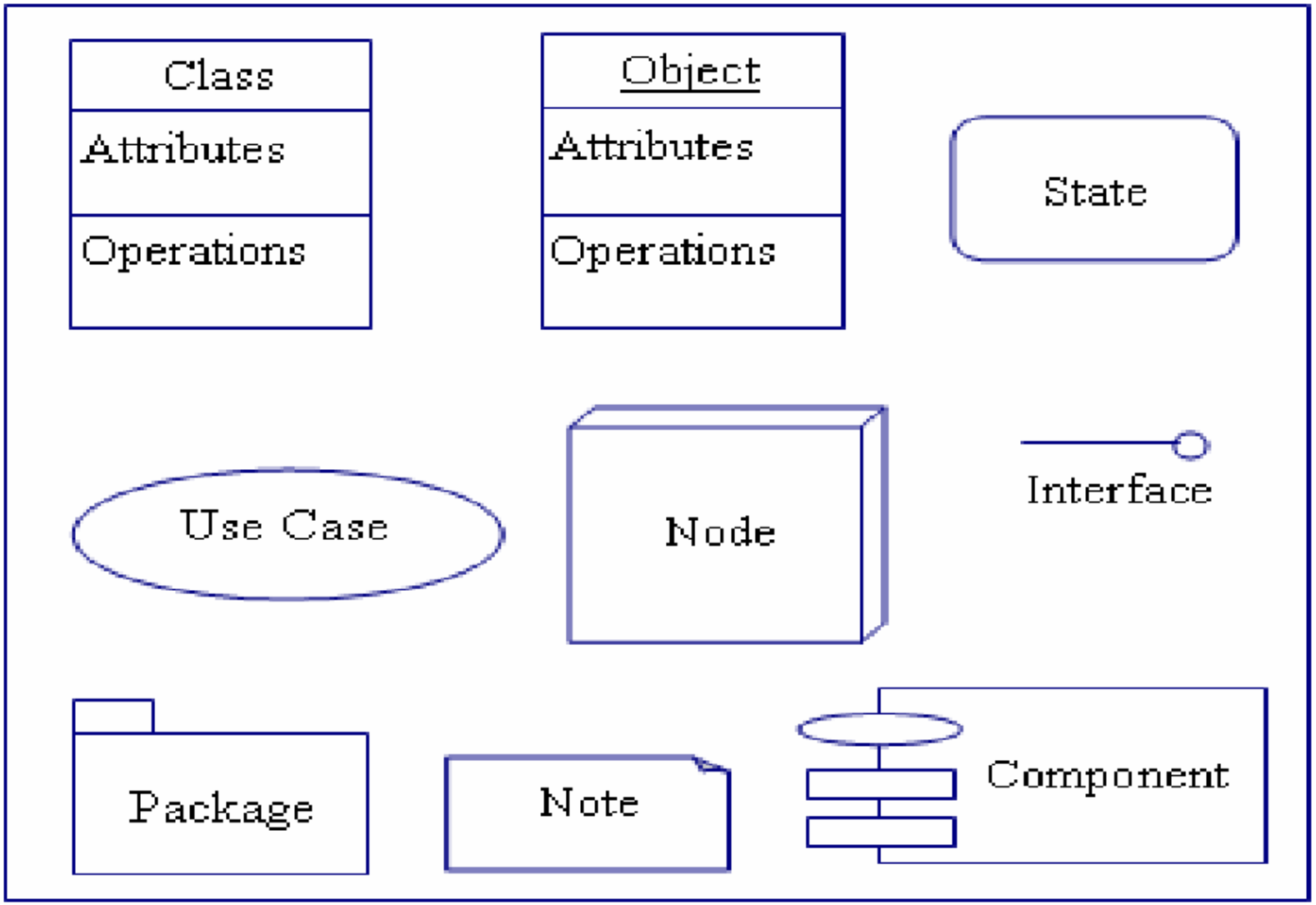
- Khung nhìn (view): chỉ ra những khía cạnh khác nhau của hệ thống cần mô hình hóa.
- Biểu đồ (diagram): các hình vẽ miêu tả nội dung trong một khung nhìn.
  - UML có tất cả 9 loại biểu đồ
- Phần tử mô hình hóa (model element): Các khái niệm sử dụng trong các biểu đồ được gọi là các phần tử mô hình
- Cơ chế chung: cung cấp thêm những lời nhận xét bổ sung, các thông tin cũng như các quy tắc ngữ pháp chung về một phần tử mô hình; chúng còn cung cấp thêm các cơ chế để có thể mở rộng ngôn ngữ UML cho phù hợp với một phương pháp xác định (một quy trình, một tổ chức hoặc một người dùng).


UML
Views
Functional
Non-functional
Organisational
Diagrams
9 diagrams
(see further on)
Model Elements
Symbology / notation
General Mechanisms
Adornments
Notes
Specifications

Phần tử mô hình trong UML

- Các khối để hình thành mô hình UML gồm ba loại
  - Phần tử
  - Quan hệ
  - Lược đồ.
- Phần tử mô hình (model element) : các khái niệm được sử dụng trong các biểu đồ
  - Có bốn loại phần tử mô hình: cấu trúc, hành vi, nhóm và chú thích.
- Quan hệ: gắn các phần tử này lại với nhau
- Lược đồ: tập hợp các phần tử.

Phần tử mô hình

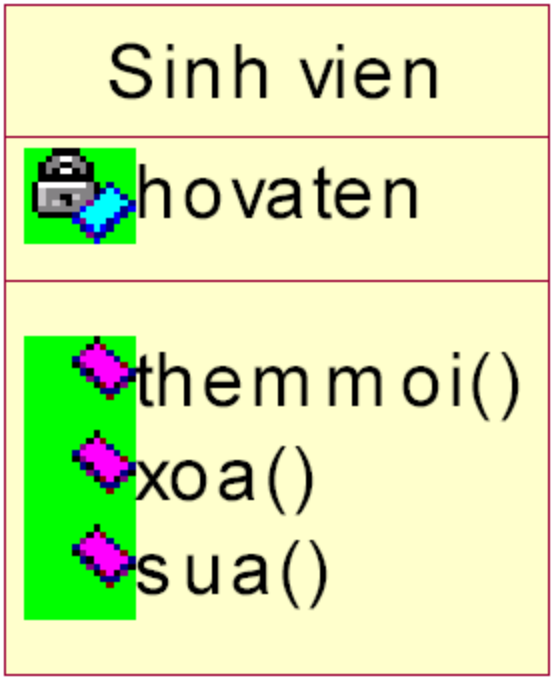


Phần tử cấu trúc

- Là bộ phận tĩnh của mô hình
- Biểu diễn các thành phần khái niệm hay vật lý
- Bao gồm:
  - Lớp
  - Giao diện
  - Phần tử cộng tác
  - Trường hợp sử dụng (Use case)
  - Thành phần (component), Nút (node): thể hiện thành phần vật lý, tồn tại khi chương trình chạy, biểu diễn các tài nguyên tính toán. Có thể đặt tập các thành phần trên nút và chuyển từ nút này sang nút khác, có thể là máy tính, thiết bị phần cứng.

Lớp

- Là mô tả tập các đối tượng cùng chung thuộc tính, thao tác, quan hệ và ngữ nghĩa.
- Lớp được minh hoạ bằng hình chữ nhật, thông thường chúng có tên, thuộc tính và thao tác

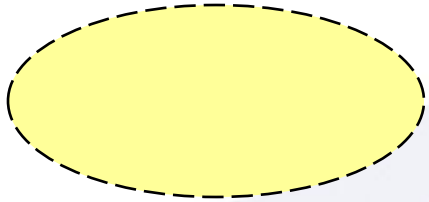


Giao diện

- Tập hợp các thao tác làm dịch vụ của lớp hay thành phần.
- Mô tả hành vi thấy được từ bên ngoài của thành phần.
- Biểu diễn toàn bộ hay một phần hành vi của lớp.
- Định nghĩa tập đặc tả thao tác, không định nghĩa cài đặt của chúng.

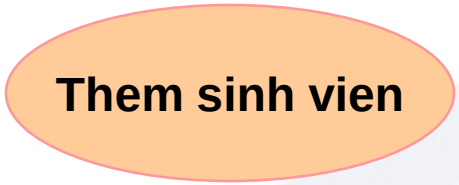
Phần tử cộng tác

- Mô tả ngữ cảnh của tương tác.
- Ký hiệu đồ hoạ: hình elip với đường vẽ nét đứt, kèm theo tên.
- Thể hiện giải pháp thi hành bên trong hệ thống, bao gồm các lớp, quan hệ và tương tác giữa chúng để đạt được một chức năng mong đợi của Use case.



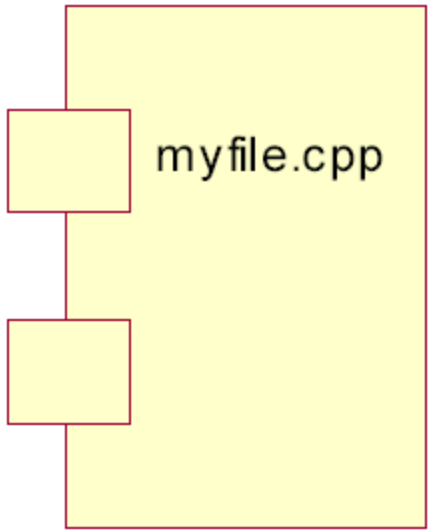
Trường hợp sử dụng (Use case)

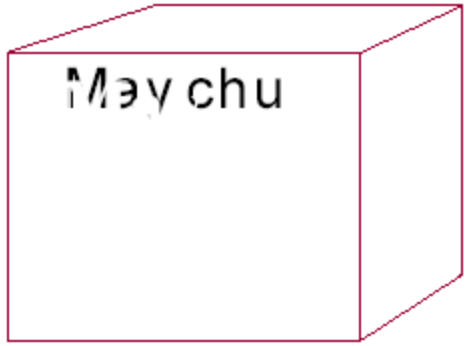
- Mô tả trình tự các hành động hệ thống sẽ thực hiện để đạt được một kết quả cho tác nhân nào đó.
- Tác nhân: những đối tượng bên ngoài tương tác với hệ thống.
- Tập hợp các Use case của hệ thống sẽ hình thành các trường hợp mà hệ thống được sử dụng.
- Sử dụng Use case để cấu trúc các phần tử có tính hành vi trong mô hình



Thành phần và nút

- Biểu diễn vật lý mã nguồn, các file nhị phân trong quá trình phát triển hệ thống.
- Nút (node): thể hiện thành phần vật lý, tồn tại khi chương trình chạy và biểu diễn các tài nguyên tính toán.
- Có thể đặt tập các thành phần trên nút và chuyển từ nút này sang nút khác, có thể là máy tính, thiết bị phần cứng.





Phần tử hành vi

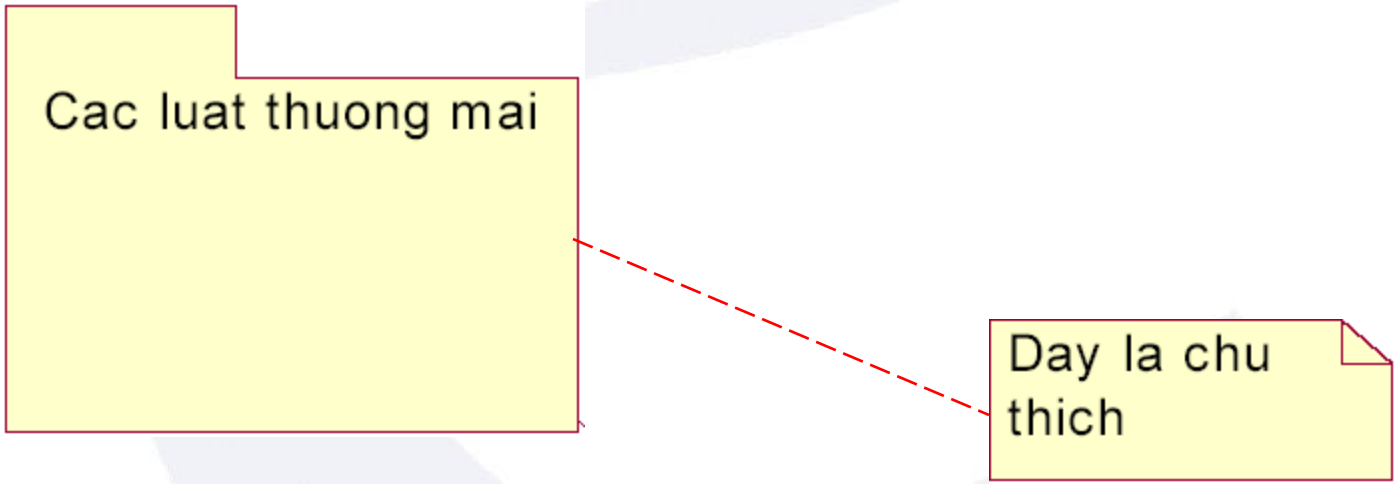
- Là bộ phận động hay các động từ của mô hình, biểu diễn hành vi theo thời gian và không gian dưới hai hình thức
  - Tương tác: là hành vi bao gồm tập các thông điệp trao đổi giữa các đối tượng trong ngữ cảnh cụ thể để thực hiện mục đích cụ thể. Hành vi của nhóm đối tượng hay của thao tác có thể được chỉ ra bằng tương tác.
  - Máy trạng thái: là hành vi chỉ ra trật tự các trạng thái mà đối tượng hay tương tác sẽ đi qua để đáp ứng sự kiện. Hành vi của lớp hay cộng tác của lớp có thể được xác định bằng máy trạng thái. Máy trạng thái kích hoạt nhiều phần tử, bao gồm trạng thái, chuyển tiếp từ trạng thái này sang trạng thái khác, sự kiện và các hoạt động đáp ứng sự kiện

Phần tử nhóm

- Là bộ phận tổ chức của mô hình.
- Chỉ có một phần tử thuộc nhóm này có tên là gói (package).
- Gói là cơ chế đa năng để tổ chức các phần tử vào nhóm.
- Các phần tử cấu trúc, hành vi và ngay cả phần tử nhóm có thể cho vào gói.
- Không giống thành phần (component), phần tử nhóm hoàn toàn là khái niệm, có nghĩa rằng chúng chỉ tồn tại vào thời điểm phát triển hệ thống chứ không tồn tại vào thời gian chạy chương trình.

Chú thích

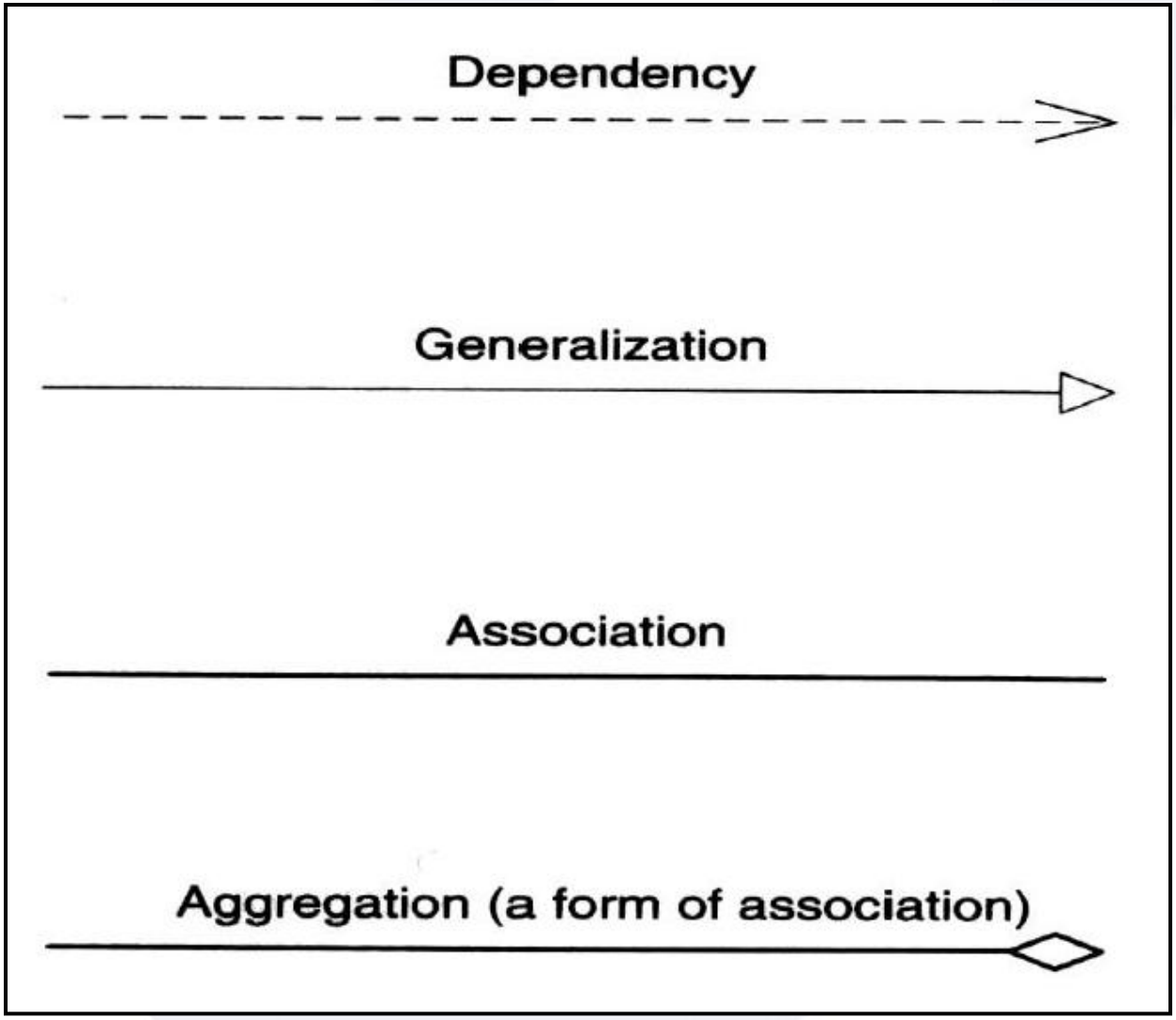
- Là bộ phận chú giải của mô hình UML.
- Phần tử chú thích được gọi là lời ghi chú (note) và dùng để mô tả các phần tử khác trong mô hình.



Các quan hệ trong UML

- Có bốn loại quan hệ trong UML
  - Kết hợp (Association) : nối các phần tử và các thực thể nối (link).
  - Khái quát hóa (Generalization): còn được gọi là tính thừa kế. Ý nghĩa: một phần tử này có thể là một sự chuyên biệt hóa của một phần tử khác.
  - Phụ thuộc (Dependency): chỉ ra rằng một phần tử này phụ thuộc trong một phương thức nào đó vào một phần tử khác.
  - Tụ hợp/bao gộp (Aggregation): Một dạng của nối kết, trong đó một phần tử này chứa các phần tử khác.
- Là cơ sở để xây dựng mọi quan hệ trong UML.

Quan hệ trong UML

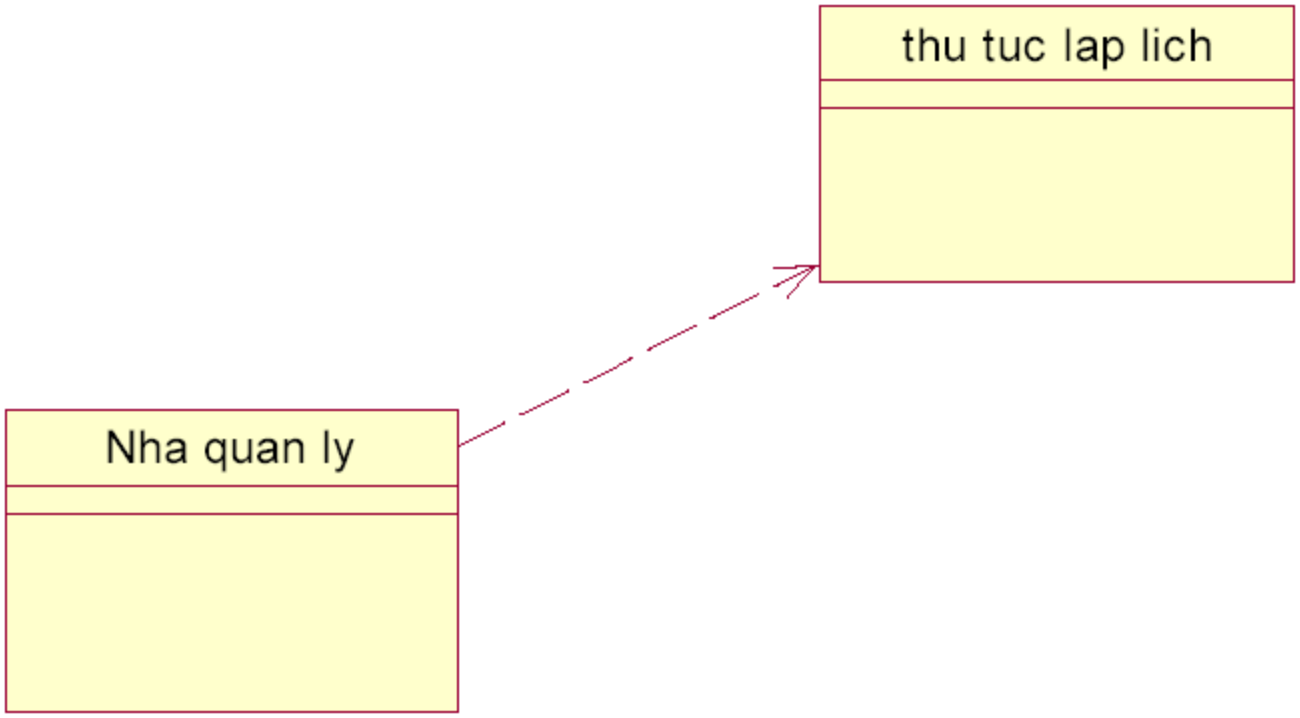


Quan hệ kết hợp (Association)

- Là quan hệ cấu trúc
- Mô tả tập liên kết (kết nối giữa các đối tượng).
- Khi đối tượng của lớp này gửi/nhận thông điệp đến/từ đối tượng của lớp kia thì ta gọi chúng là có quan hệ kết hợp.
- Chúng có thể chứa tên nhiệm vụ và tính nhiều (mutiplicity).

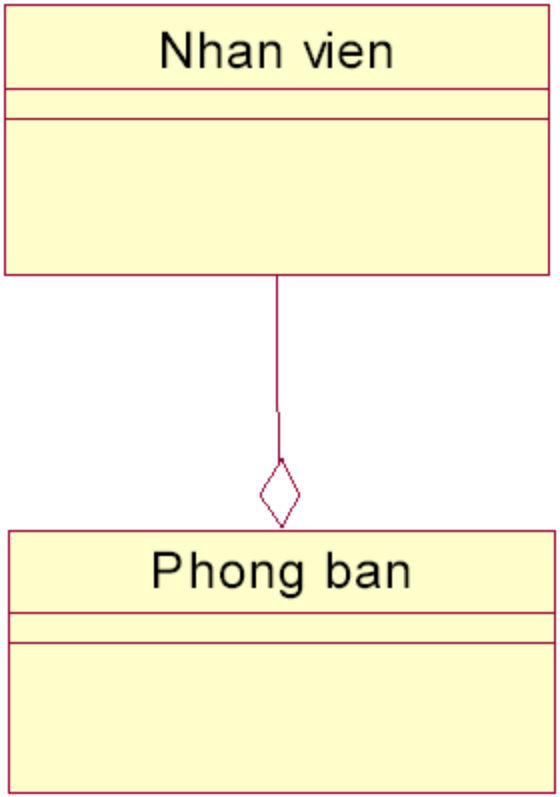
Quan hệ phụ thuộc (dependency)

- Là quan hệ ngữ nghĩa giữa hai phần tử trong đó thay đổi phần tử độc lập sẽ tác động đến ngữ nghĩa của phần tử phụ thuộc.



Tụ hợp/Bao gộp (aggregration)

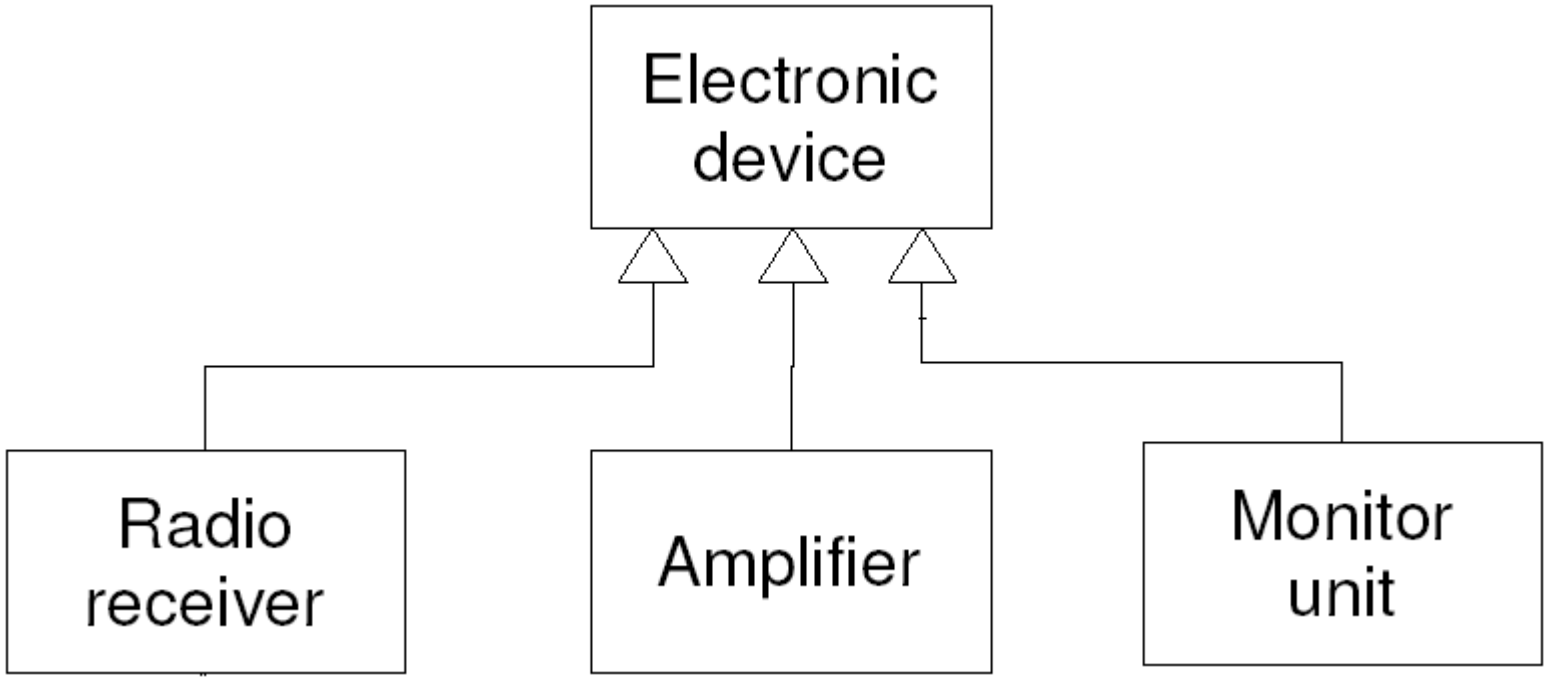
- Là dạng đặc biệt của kết hợp, biểu diễn quan hệ cấu trúc giữa toàn thể và bộ phận.
- Một dạng đặc biệt của tụ hợp là quan hệ hợp thành (composition), trong đó nếu như đối tượng toàn thể bị huỷ bỏ thì các đối tượng bộ phận của nó cũng bị huỷ bỏ theo.



Khái quát hóa và hiện thực hóa

- Khái quát hoá (generalization): là quan hệ đặc biệt hoá / khái quát hoá, đối tượng cụ thể kế thừa các thuộc tính và phương thức của đối tượng tổng quát.
- Hiện thực hoá (realization): là quan hệ ngữ nghĩa giữa giao diện và lớp hay thành phần hiện thực lớp; giữa use case và hợp tác hiện thực Use case.

Khái quát hóa và hiện thực hóa



Kiến trúc hệ thống

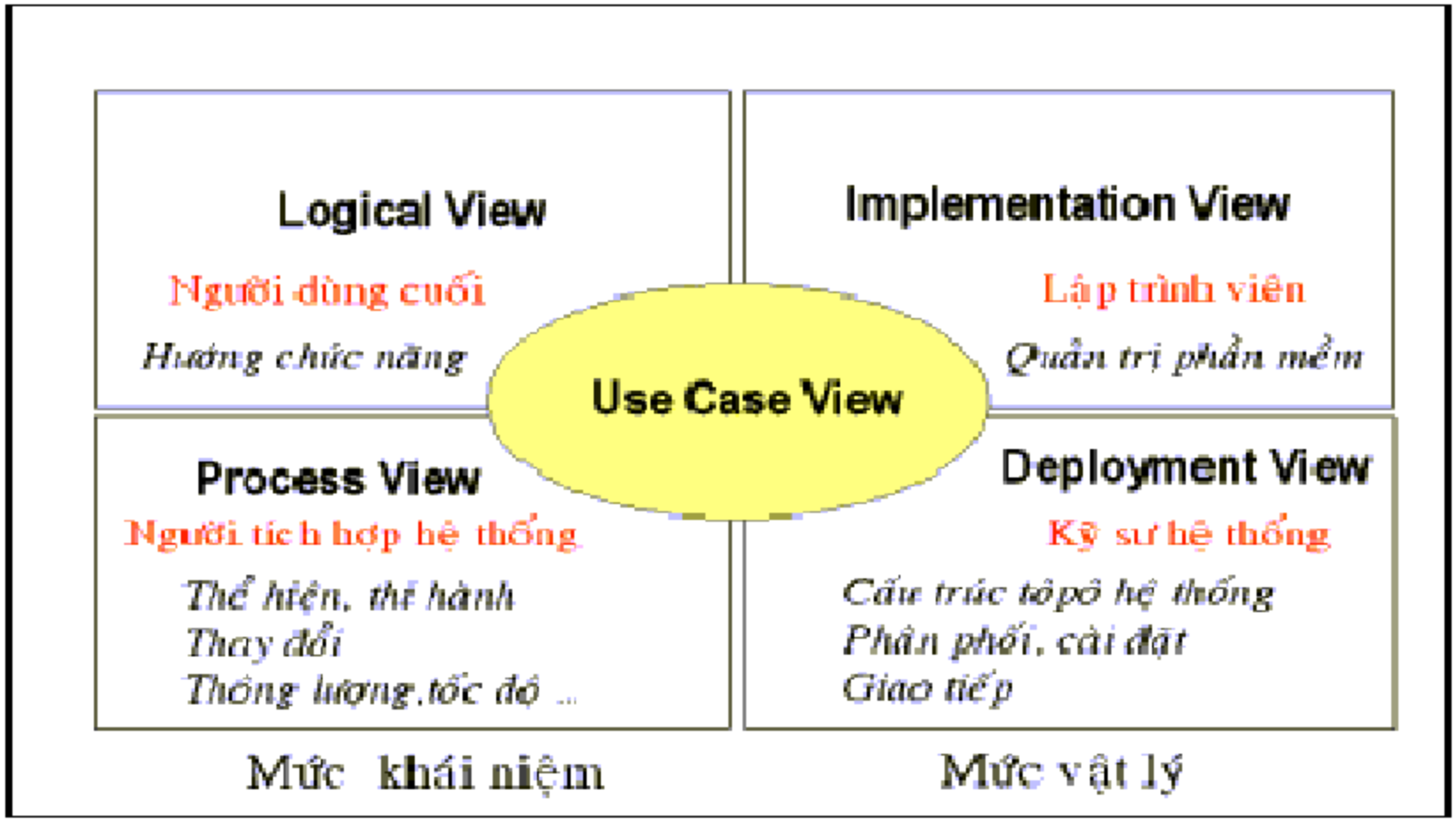
- Không thể mô hình hoá một hệ thống phức tạp chỉ bằng một mô hình hay lược đồ, hệ thống phải được phân tích dưới những góc độ khác nhau.
- Việc hiển thị, đặc tả, xây dựng và làm tài liệu hệ thống đòi hỏi hệ thống phải được xem xét từ nhiều khía cạnh khác nhau
- Kiến trúc hệ thống được sử dụng để quản lí các điểm nhìn khác nhau để điều khiển sự phát triển hệ thống tăng dần và lặp trong suốt chu kì sống.

Cấu trúc View

- Một hệ thống cần phải được miêu tả với một loạt các khía cạnh khác nhau
- Một hệ thống thường được miêu tả trong một loạt các hướng nhìn khác nhau, mỗi hướng/khung nhìn sẽ thể hiện một bức ảnh ánh xạ của toàn bộ hệ thống và chỉ ra một khía cạnh riêng của hệ thống.
- Mỗi View là một thể hiện của hệ thống được mô hình hoá, mỗi View có thể bao gồm nhiều loại biểu đồ khác nhau

4+1 khung nhìn của UML

- Một hệ thống được mô tả trong 1 số khía cạnh khác nhau



4+1 khung nhìn

- User model View (Use Case View hoặc Scenario View)- thể hiện các vấn đề và các giải pháp liên quan đến chức năng tổng quát của hệ thống.
- Structural model View (Static hoặc Logical View)- thể hiện các vấn đề liên quan đến cấu trúc thiết kế của hệ thống.
- Behavioral model View (Dynamic, Process Concurrent, hoặc Collaboration View) thể hiện các vấn đề liên quan đến việc xử lý giao tiếp và đồng bộ trong hệ thống.
- Implementation model View (Component View) thể hiện các vấn đề liên quan đến việc tổ chức các thành phần trong hệ thống.
- Environment model View (Deployment View) thể hiện các vấn đề liên quan đến việc triển khai hệ thống.

Khung nhìn Use Case

- Tác giả của Use case là Ivar Jacobson
- Use case trở thành một phương pháp được sử dụng cho việc nắm bắt các yêu cầu vào những năm 1990s
- Khung nhìn Use case miêu tả chức năng của hệ thống sẽ phải cung cấp do được tác nhân từ bên ngoài mongđợi.
- Use case không chỉ được sử dụng cho việc mô hình hóa các hệ thống kĩ thuật mà còn dùng để mô hình các hệ thống kinh doanh
- Use case thể hiện tương tác nổi bật giữa user và hệ thống.

Use case View

- Khung nhìn Use case là khung nhìn dành cho khách hàng, nhà thiết kế, nhà phát triển và người thử nghiệm; nó được miêu tả qua các biểu đồ Use case (use case diagram) và thỉnh thoảng cũng bao gồm cả các biểu đồ hoạt động (activity diagram).
- Khung nhìn Use case mang tính trung tâm, bởi nó đặt ra nội dung thúc đẩy sự phát triển các khung nhìn khác.
- Khung nhìn này cũng được sử dụng để thẩm tra (verify) hệ thống qua việc thử nghiệm xem khung nhìn Use case có đúng với mong đợi của khách hàng (Hỏi: "Đây có phải là thứ bạn muốn") cũng như có đúng với hệ thống vừa được hoàn thành (Hỏi: "Hệ thống có hoạt động như đã đặc tả?”).

Khung nhìn Logic (Logic View)

- Miêu tả phương thức mà các chức năng của hệ thống sẽ được cung cấp.
  - Chủ yếu nó được sử dụng cho các nhà thiết kế và nhà phát triển.
- Đi vào phía bên trong của hệ thống trong đó
  - Cấu trúc tĩnh được miêu tả bằng các biểu đồ lớp (class diagram) và biểu đồ đối tượng (object diagram).
  - Quá trình mô hình hóa động được miêu tả trong các biểu đồ trạng thái (state diagram), biểu đồ trình tự (sequence diagram), biểu đồ tương tác (collaboration diagram) và biểu đồ hoạt động (activity diagram).
- Định nghĩa các thuộc tính như trường tồn (persistency), song song (concurrency), giao diện cũng như cấu trúc nội tại của các lớp

Khung nhìn song song (Cuncurrency View)

- Khung nhìn song song nhắm tới việc chia hệ thống thành các qui trình (process) và các bộ xử lý (processor).
- Sử dụng hiệu quả các tài nguyên (efficient usage of resources)
  - Thực hiện song song
  - Xử lý các sự kiện không đồng bộ từ môi trường.
- Phải quan tâm đến vấn đề giao tiếp và đồng bộ hóa các tiểu trìnhđó.
- Dùng cho các người phát triển và tích hợp hệ thống
- Bao gồm các diagrams:
  - Biểu đồ động (trạng thái, trình tự, tương tác và hoạt động)
  - Biểu đồ thực thi (biểu đồ thành phần và biểu đồ triển khai).

Khung nhìn thành phần (Component View)

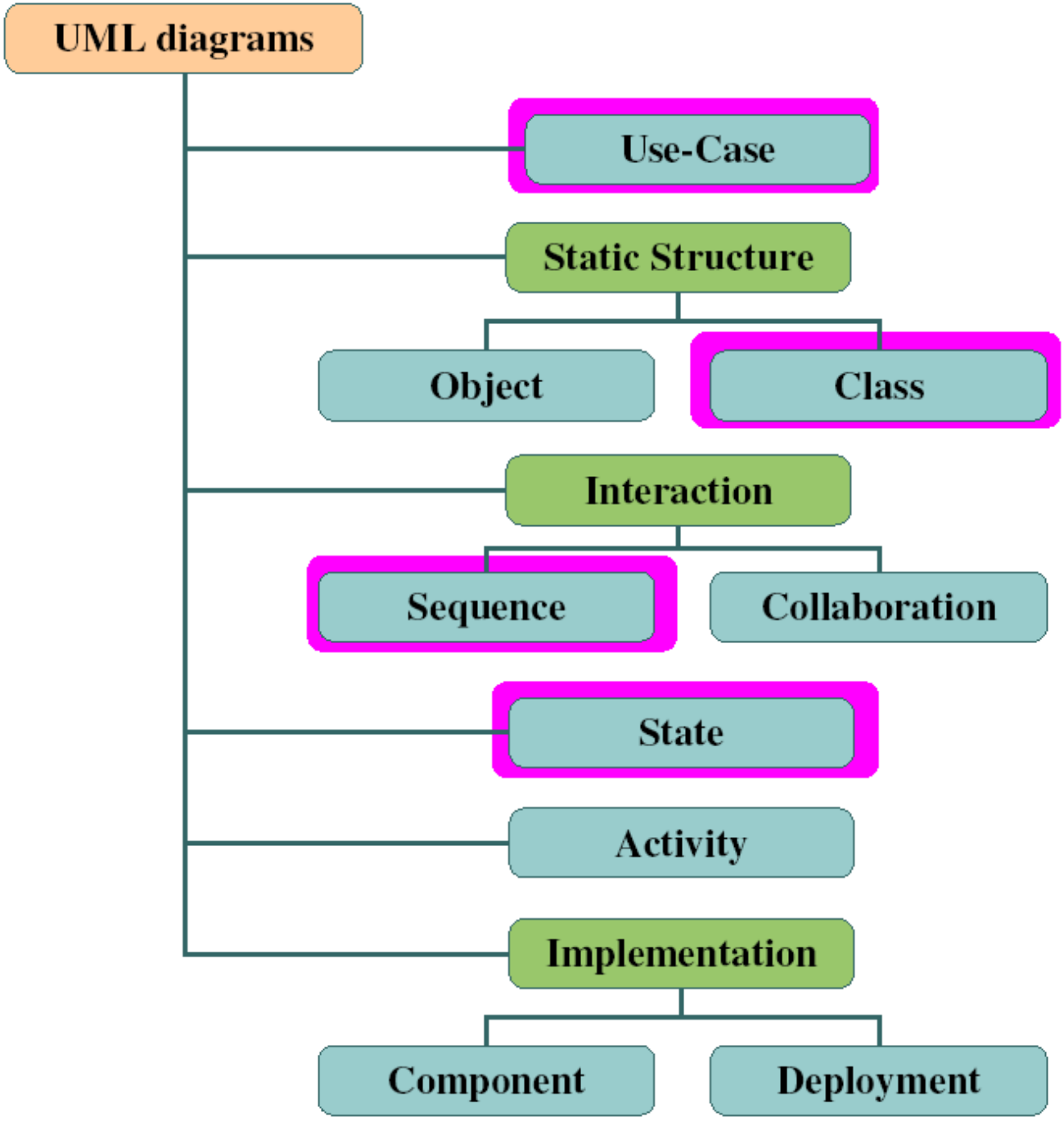
- Đặc tả của các mô đun cài đặt
  - Thường được sử dụng cho nhà phát triển, bao gồm nhiều biểu đồ thành phần.
- Thành phần ở đây là các modul thuộc nhiều loại khác nhau, được chỉ ra trong biểu đồ cùng với cấu trúc cũng như sự phụ thuộc của chúng.

Khung nhìn triển khai (Deployment View)

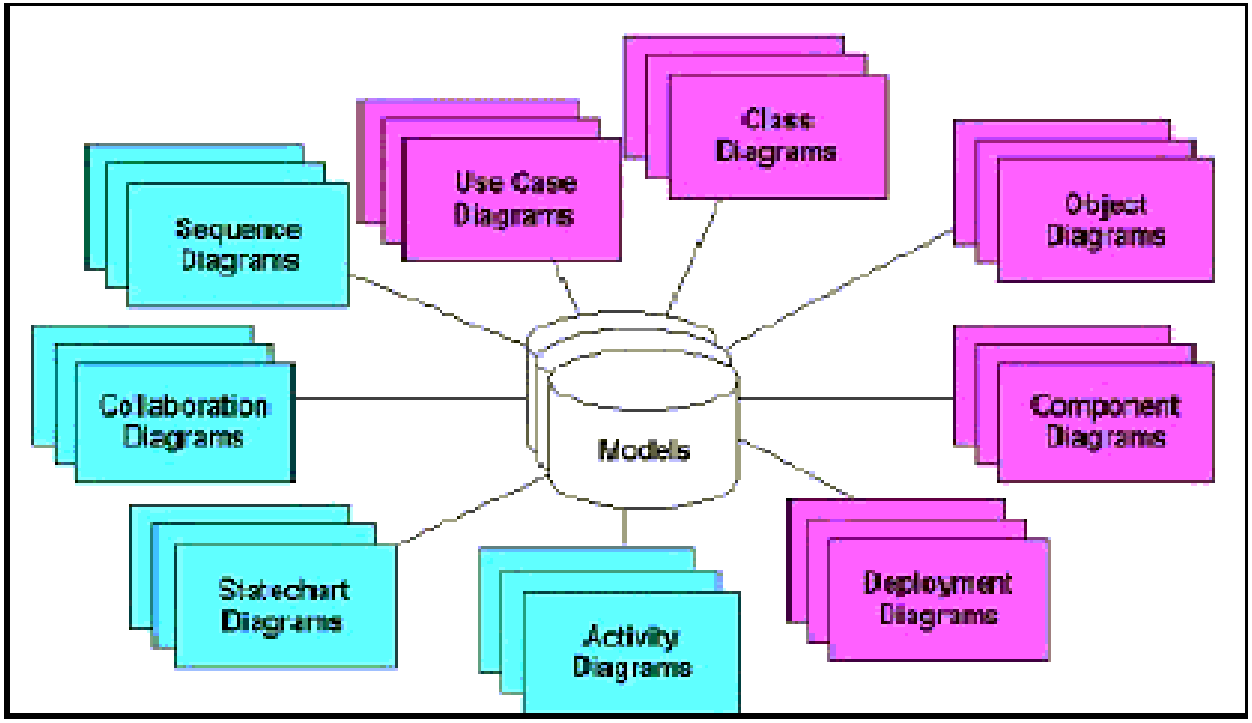
- Hướng nhìn triển khai chỉ cho chúng ta sơ đồ triển khai về mặt vật lý của hệ thống
  - Ví dụ: máy tính, máy móc và sự liên kết giữa chúng.
- Được sử dụng bởi người phát triển, tích hợp và test hệ thống
- Được thể hiện bằng các biểu đồ triển khai - Deployment diagram
- Bao gồm sự ánh xạ các thành phần của hệ thống vào cấu trúc vật lý
  - Ví dụ: chương trình nào hay đối tượng nào sẽ được thực thi trên máy tính nào.

Biểu đồ (diagram)

- Là các hình vẽ bao gồm các ký hiệu phần tử mô hình hóa được sắp xếp để minh họa một thành phần cụ thể hay một khía cạnh cụ thể của hệ thống.
- Một mô hình hệ thống thường có nhiều loại biểu đồ, mỗi loại có nhiều biểu đồ khác nhau. Một biểu đồ là một thành phần của một khung nhìn cụ thể; và khi được vẽ ra, nó thường được xếp vào một khung nhìn. Mặt khác, một số loại biểu đồ có thể là thành phần của nhiều khung nhìn khác nhau, tùy thuộc vào nội dung của biểu đồ.


UML diagrams
Use-Case
Static Structure
Object
Class
Interaction
Sequence
Collaboration
State
Activity
Implementation
Component
Deployment

Biểu đồ


Class Diagrams
Use Case Diagrams
Sequence Diagrams
Object Diagrams
Collaboration Diagrams
Models
Component Diagrams
Statechart Diagrams
Activity Diagrams
Deployment Diagrams

Biểu đồ

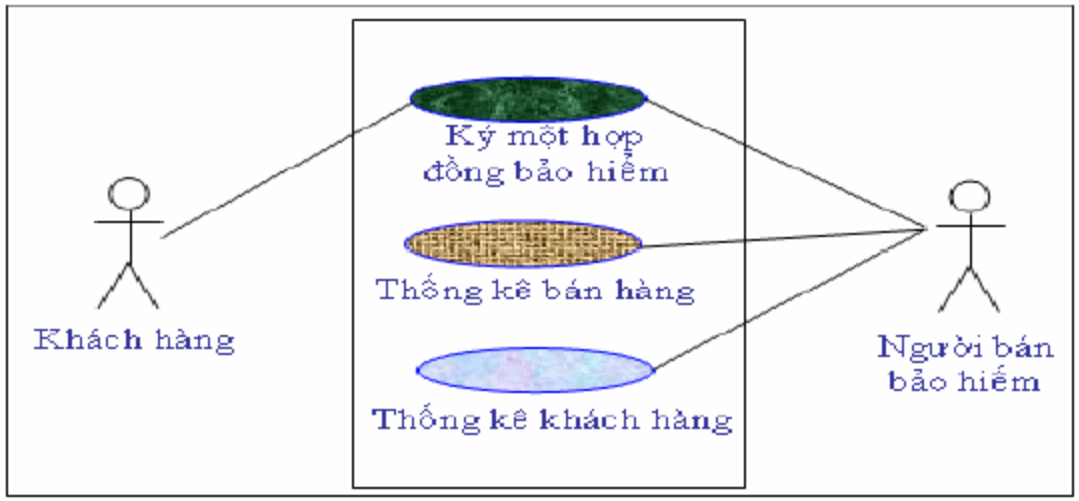
- Use case: nắm bắt các yêu cầu
- Sequence: Thể hiện thời gian tương tác giữa các đối tượng
- Collaboration: Làm nổi bật các quan hệ giữa các đối tượng và liên kết giữa chúng
- Class: thể hiện cấu trúc tĩnh của hệ thống
- State: các trạng thái của đối tượng
- Activity: các hoạt động song song và các tiến trình kinh doanh
- Component: cấu trúc vật lí của các thành phần chính
- Deployment: phân bố vật lí của hệ thống

Use Case Diagram

- Mô tả chức năng mà hệ thống cung cấp.
- Hình thức mô tả: văn bản, tài liệu, biểu đồ hoạt động
- Use case diagram thể hiện các Actor liên hệ như thế nào với các use case
  - Actors: là một người hay cái gì đó nằm ngoài hệ thống và tương tác với hệ thống
  - Use case có thể được tinh lọc từ các use case khác

Use Case Diagram

- Use case được miêu tả duy nhất theo hướng nhìn từ ngoài vào của các tác nhân (hành vi của hệ thống theo như sự mong đợi của người sử dụng), không miêu tả chức năng được cung cấp sẽ hoạt động nội bộ bên trong hệ thống ra sao.

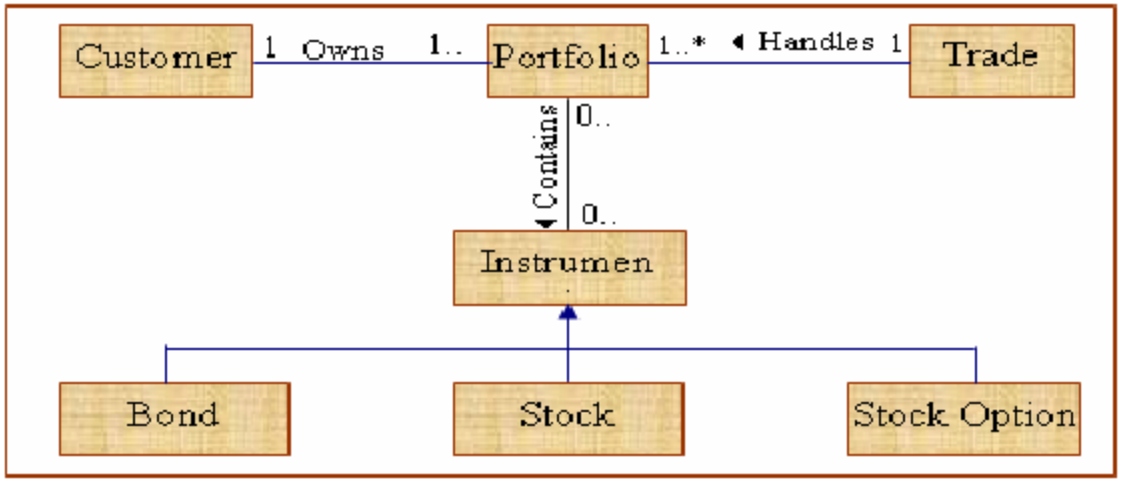


Biểu đồ lớp (Class Diagram)

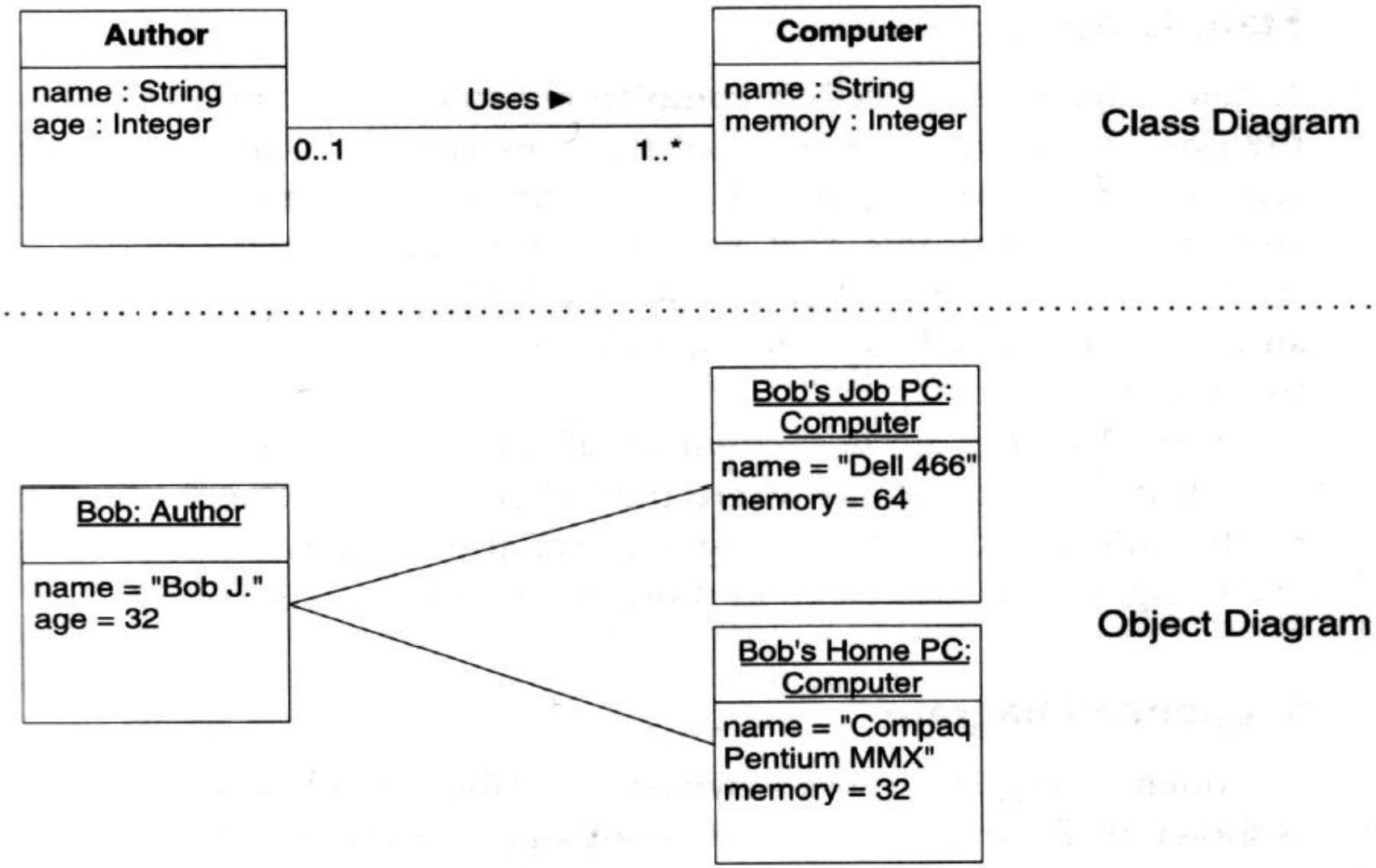
- Chuyển từ khung nhìn use case sang khung nhìn logic của hệ thống
- Class là thành phần chính dùng để xây dựng bất kì một hệ thống hướng đối tượng nào
- Class diagram quản lý các class và quan hệ giữa chúng
- Class là một mô tả của kiểu object, bao gồm cấu trúc và các tác động
- Các thể hiện của một class là các objects

Biểu đồ lớp (Class Diagram)

- Biểu đồ lớp chỉ ra cấu trúc tĩnh của các lớp trong hệ thống
- Một hệ thống thường sẽ có một loạt các biểu đồ lớp

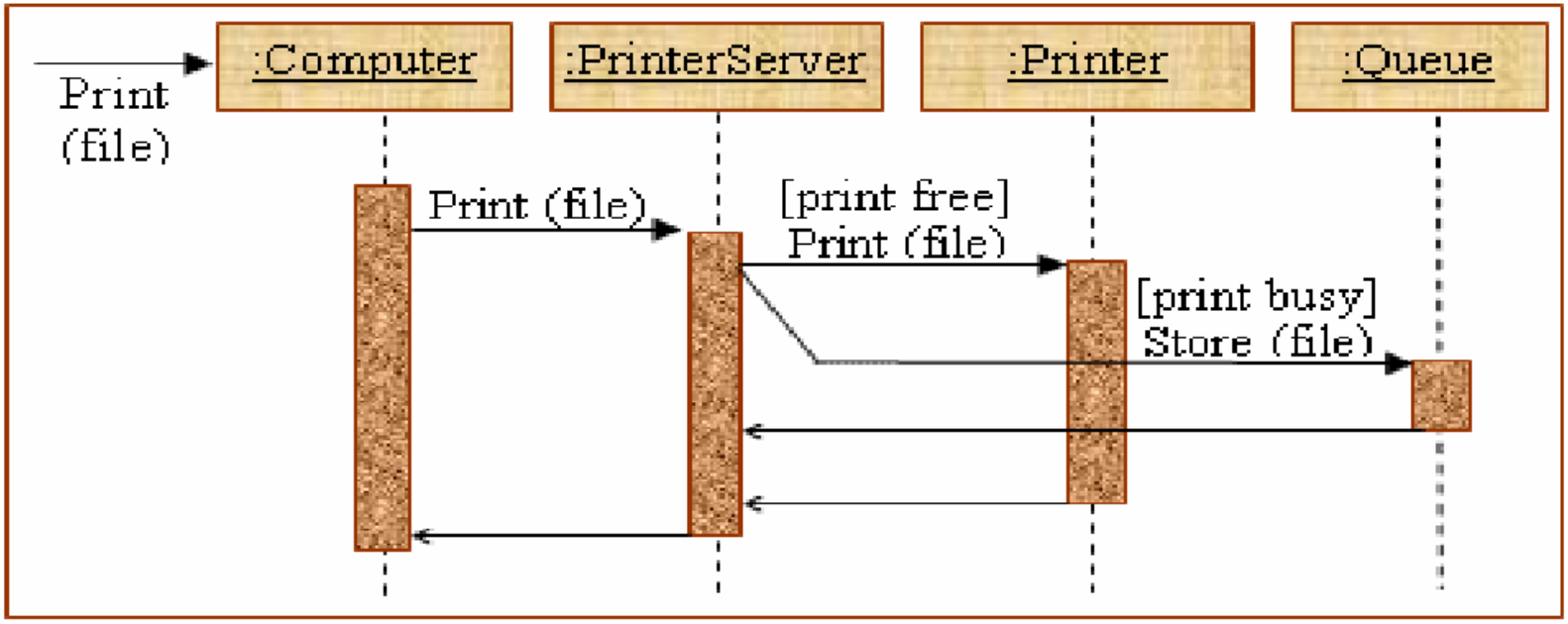


Biểu đồ đối tượng (Object Diagram)



Biểu đồ trình tự (Sequence Diagram)

- Một biểu đồ trình tự chỉ ra sự tương tác động giữa các đối tượng
- Khía cạnh quan trọng của biểu đồ này là chỉ ra trình tự các thông điệp (message) được gửi giữa các đối tượng.

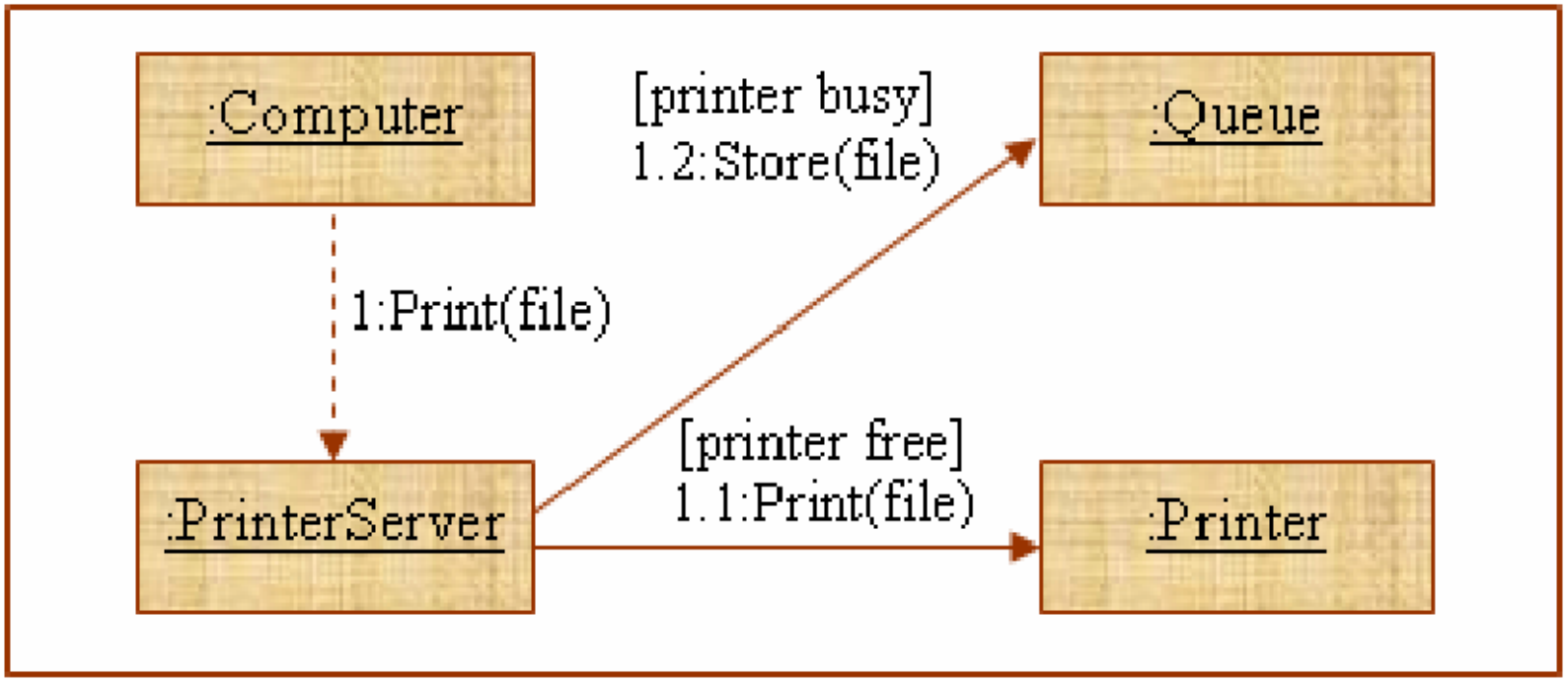


Biểu đồ cộng tác (CollaborationDiagram)

- Chỉ ra sự tương tác động, giống như biểu đồ trình tự.
- Lựa chọn:
  - Nếu thời gian hay trình tự là yếu tố quan trọng nhất cần nhấn mạnh => chọn biểu đồ trình tự
  - Nếu ngữ cảnh là yếu tố quan trọng hơn => chọn biểu đồ cộng tác.
  - Trình tự tương tác giữa các đối tượng được thể hiện trong cả hai loại biểu đồ này.

Biểu đồ cộng tác (CollaborationDiagram)

- Bên cạnh việc thể hiện sự trao đổi thông điệp (gọi là tương tác), biểu đồ cộng tác chỉ ra các đối tượng và quan hệ của chúng (đôi khi được gọi là ngữ cảnh).

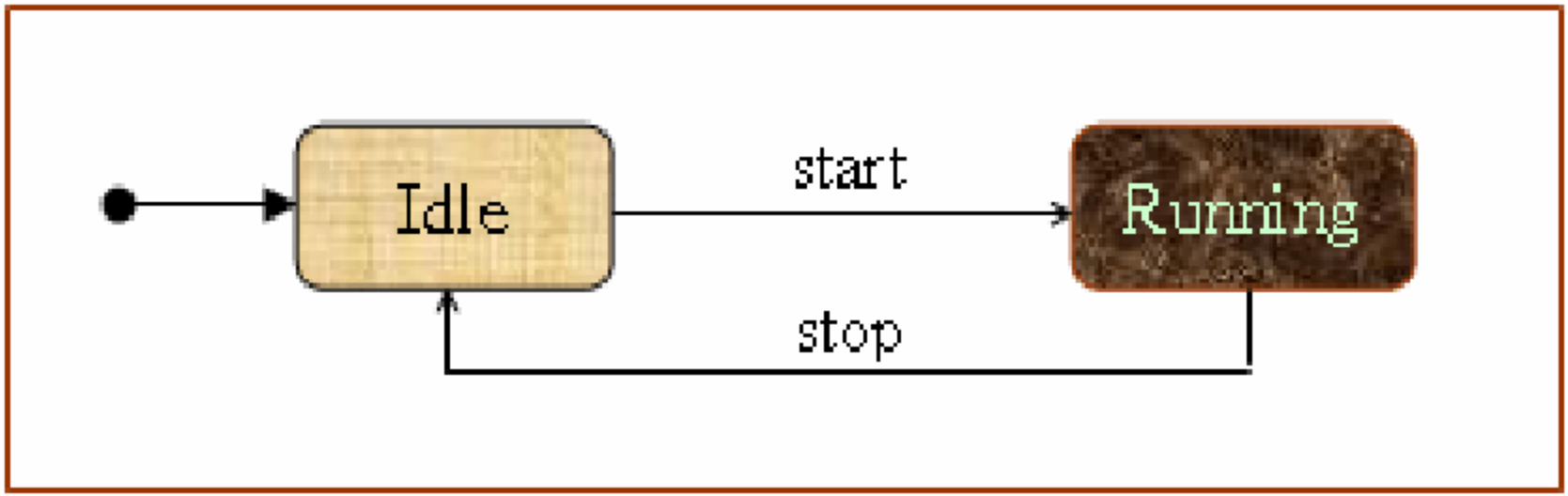


Biểu đồ trạng thái (State Diagram)

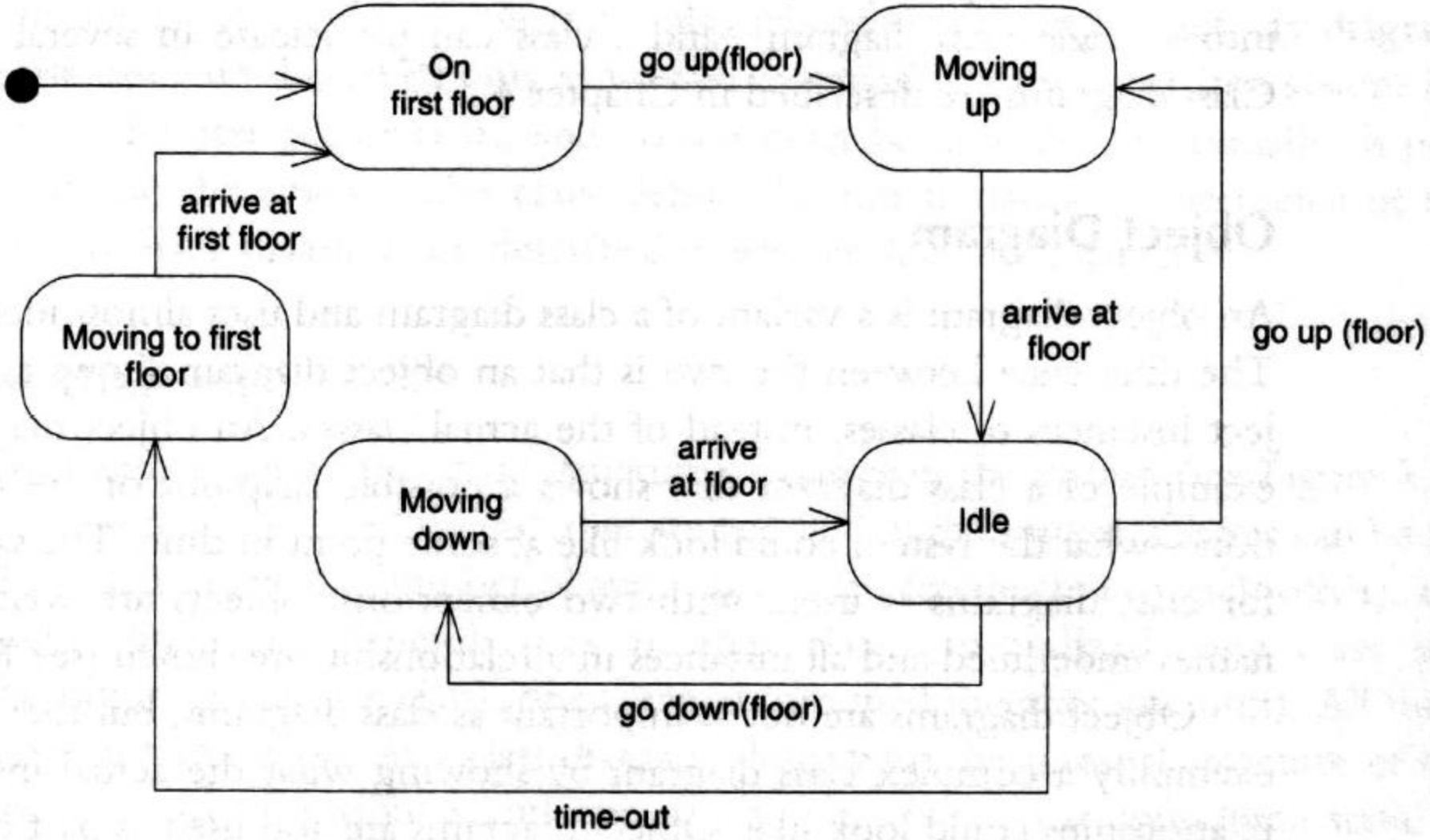
- Mô hình hóa các tác động bên trong đối tượng
- UML state diagram được xây dựng dựa trên StateCharts của David Harel
  - Nâng cấp các khả năng của state diagram để mô hình các hệ thống lớn
  - Một cách chính thức, state diagram được gọi là “statechart diagrams”
- State diagram nên được sử dụng cho các object với các hành vi động

Biểu đồ trạng thái (State Diagram)

- Chỉ sử dụng cho những lớp có số lượng các trạng thái được định nghĩa rõ ràng, hành vi bị ảnh hưởng và thay đổi qua các trạng thái khác nhau.



Biểu đồ trạng thái (State Diagram)



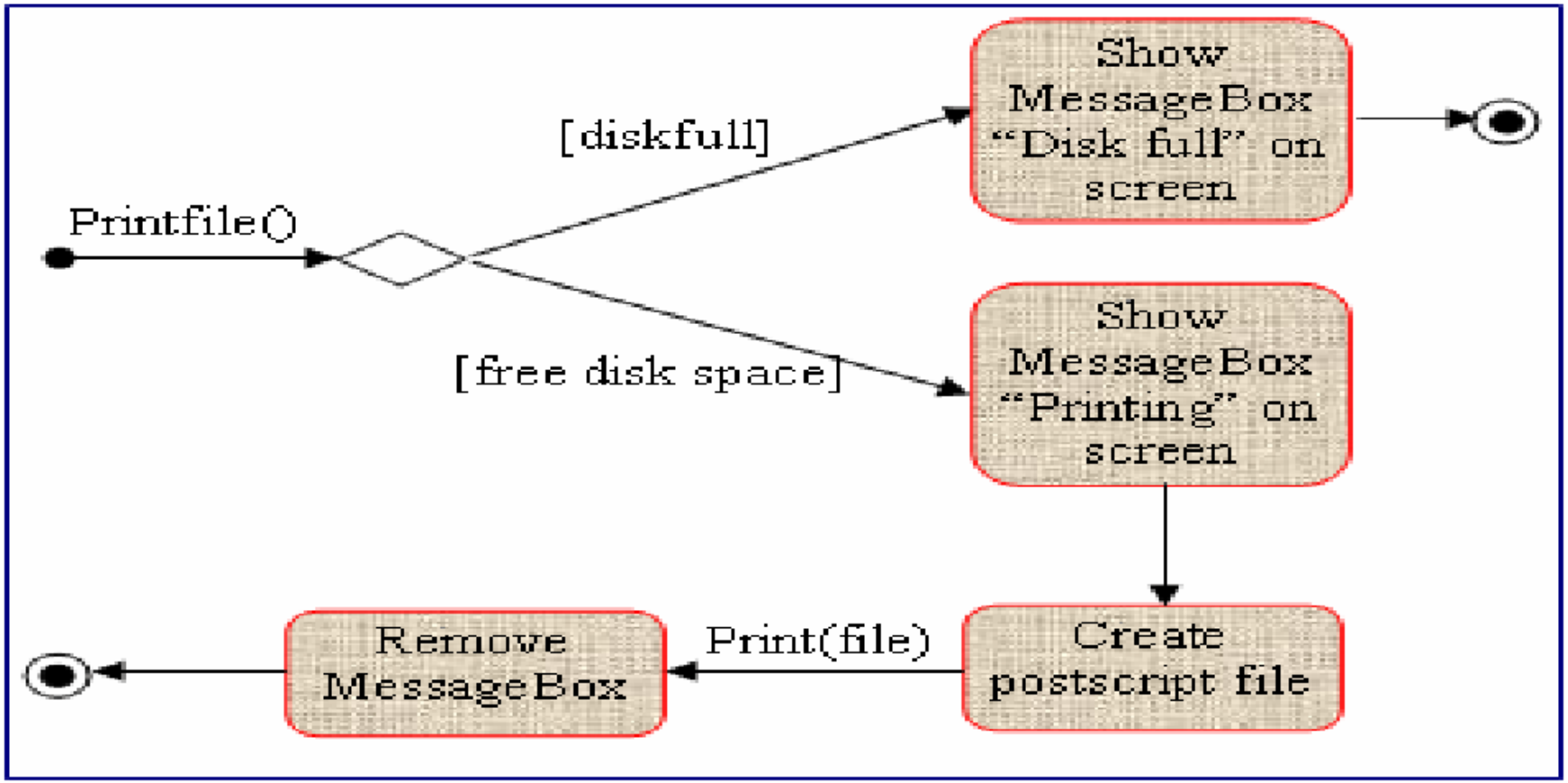
Biểu đồ hoạt động (Activity Diagram)

- Mô hình hóa các tương tác động của hệ thống
- Miêu tả các tiến trình song song của hệ thống
- Được hình hành bởi sự kết hợp các ý tưởng và khái niệm của nhiều nguồn:
  - BPR của Rummler-Brache
  - Event diagram của J Odell
  - SDL state
- Dùng để mô tả workflow

Biểu đồ hoạt động (Activity Diagram)

- Chỉ ra trình tự của các hoạt động (activity)
- Miêu tả các hoạt động được thực hiện trong một thủ tục
- Ngoài ra, dùng để miêu tả các dòng chảy hoạt động khác, ví dụ: luồng sự kiện trong một Use case hay trong một trình tự tương tác.
- Biểu đồ hoạt động bao gồm các trạng thái hành động, chứa đặc tả của một hoạt động (một hành động - action).
- Một trạng thái hành động sẽ qua đi khi hành động được thực hiện xong (khác với biểu đồ trạng thái: một trạng thái chỉ chuyển sang trạng thái khác sau khi đã xảy ra một sự kiện rõ ràng !)

Biểu đồ hoạt động (Activity Diagram)

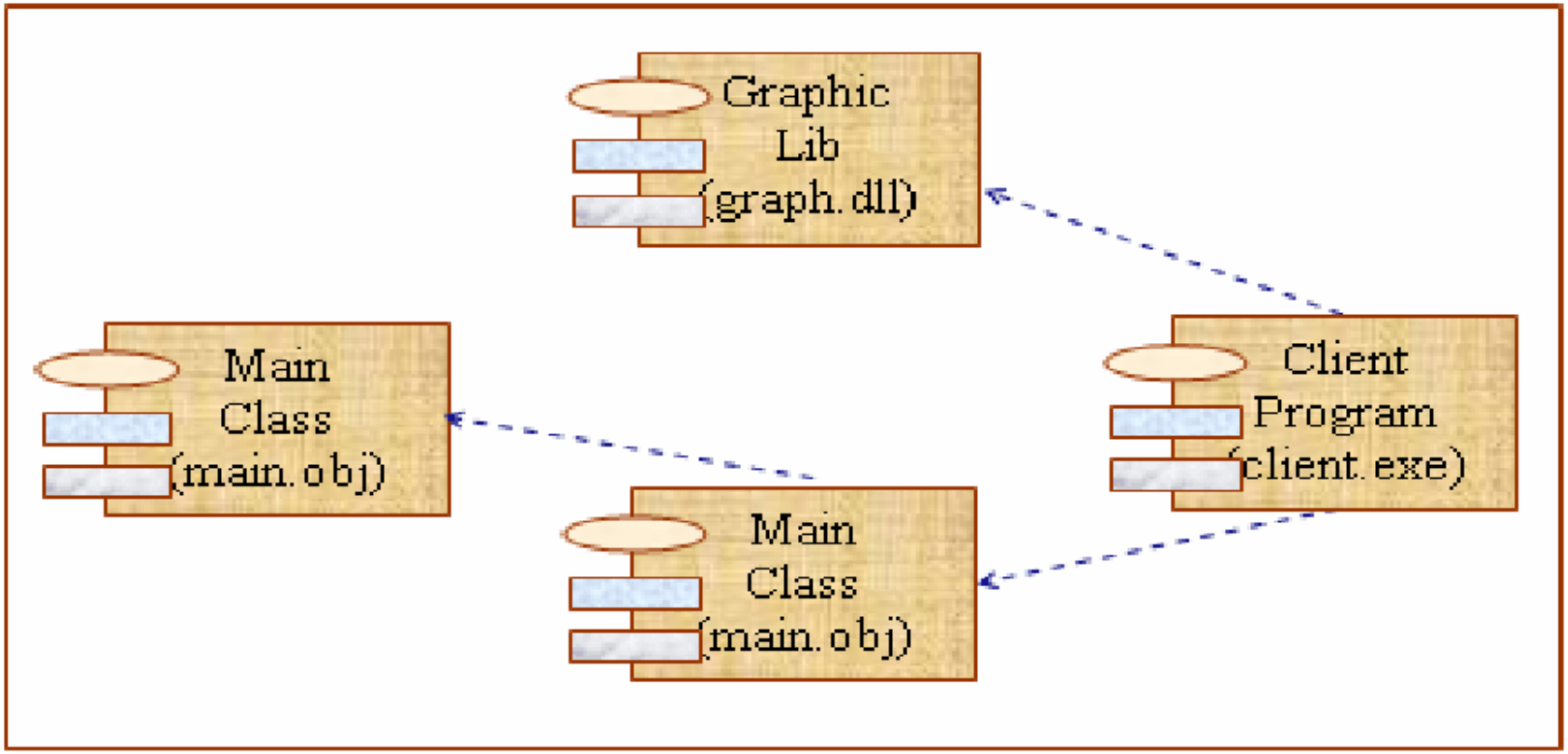


Biểu đồ thành phần (Component)

- Miêu tả cấu trúc vật lí của phần mềm
  - Chỉ ra cấu trúc vật lý của các dòng lệnh (code) theo khái niệm thành phần code
  - Một thành phần chứa các thông tin về các lớp logic hoặc các lớp mà nó thi hành, như thế có nghĩa là nó tạo ra một ánh xạ từ hướng nhìn logic vào hướng nhìn thành phần.
- Liên kết phụ thuộc nghĩa là một component cần một component khác để có một định nghĩa hoàn chỉnh
- Một component có thể định nghĩa các interface (giao diện) tới các component khác và thể hiện rõ ràng trong diagram

Biểu đồ thành phần (Component)

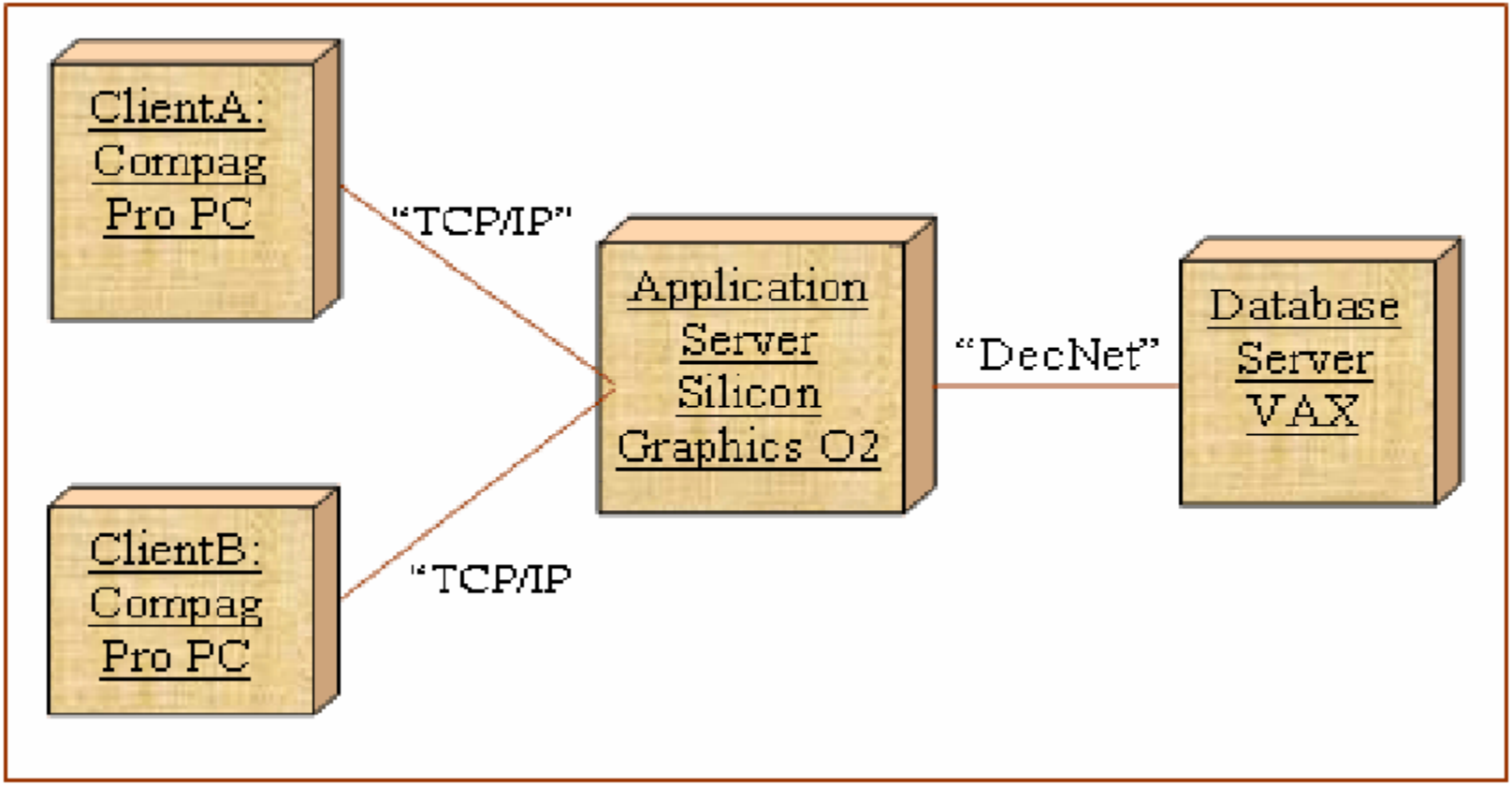
- Biểu đồ thành phần chỉ ra sự phụ thuộc giữa các thành phần mã



Biểu đồ triển khai (Deployment)

- Chỉ ra kiến trúc vật lý của phần cứng, phần mềm trong hệ thống
  - Nodes là các object vật lí, ví dụ như máy tính, máy in , vv
  - Connections là các đường dẫn giữa các node
- Chỉ ra hướng nhìn triển khai, miêu tả kiến trúc vật lý thật sự của hệ thống
  - Đây là một hướng nhìn rất xa lối miêu tả duy chức năng của hướng nhìn Use case. Mặc dù vậy, trong một mô hình tốt, có thể chỉ tất cả những con đường dẫn từ một nút mạng trong một kiến trúc vật lý cho tới những thành phần của nó, cho tới lớp mà nó thực thi, cho tới những tương tác mà các đối tượng của lớp này tham gia để rồi cuối cùng, tiến tới một Use case.

Biểu đồ triển khai (Deployment)



Q/A

PHÂN TÍCH THIẾT KẾ HƯỚNG ĐỐI TƯỢNG VỚI UML

Giảng viên: ThS. Nguyễn Đình Loan Phương
Email: phuongndl@uit.edu.vn

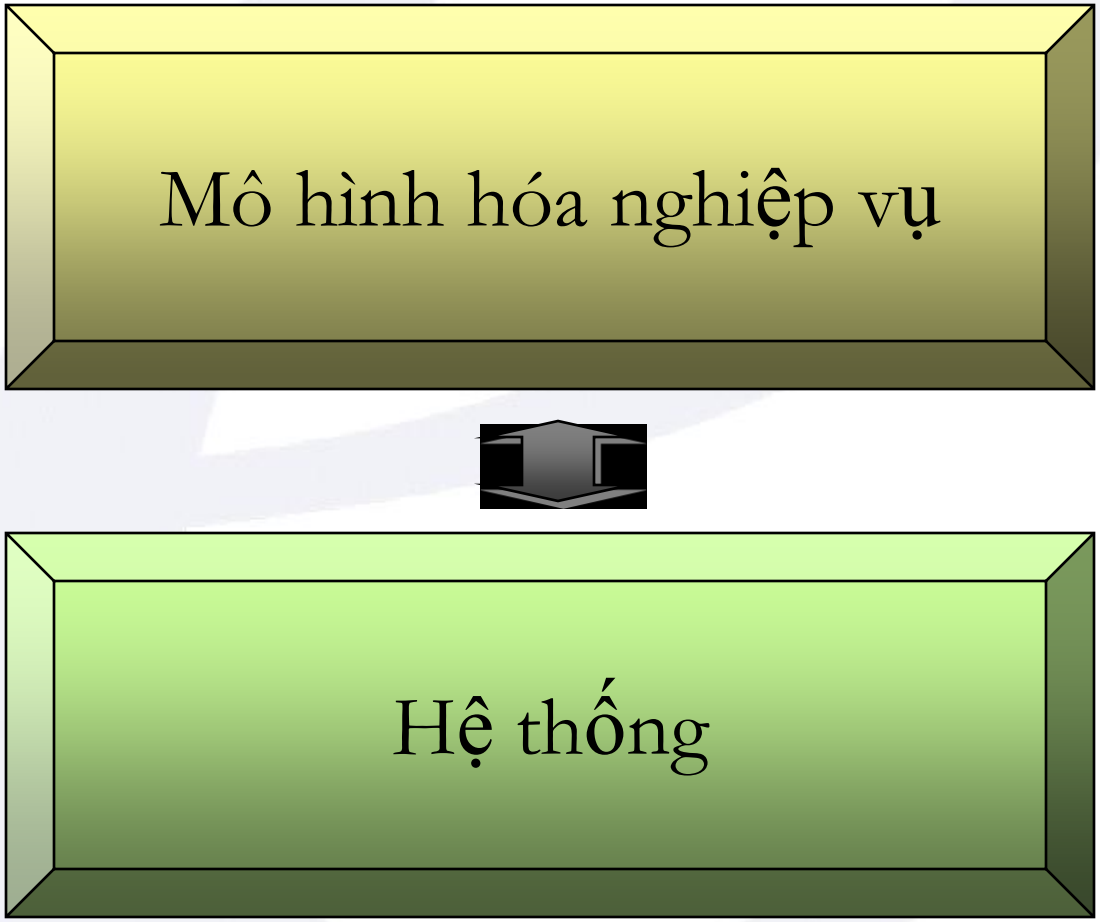
Mô hình hoá nghiệp vụ

- Mô hình hóa nghiệp là gì?
- Tại sao phải mô hình hóa nghiệp vụ
- Luồng công việc của mô hình hóa nghiệp vụ
- Phân tích qui trình nghiệp vụ
- Mô tả ràng buộc
- Thiết kế qui trình nghiệp vụ

Mô hình hóa nghiệp vụ là gì?

- Trực quan hóa những hệ thống phức tạp
  - Dễ giao tiếp, dễ truyền đạt
  - Giúp thực hiện những giải pháp dễ dàng hơn. Chúng ta có thể so sánh và tối ưu hóa
- Nắm bắt được các yêu cầu nghiệp vụ
- Xác định được phạm vi hệ thống
- Biểu diễn sự thay đổi, cải tiến qui trình đã tồn tại, hoặc xây dựng qui trình mới, hoặc nâng cấp môi trường, …

Mô hình hóa nghiệp vụ là gì?



Mô hình hoá nghiệp vụ

- Mô hình hóa nghiệp là gì?
- ➢ Tại sao phải mô hình hóa nghiệp vụ
- Luồng công việc của mô hình hóa nghiệp vụ
- Phân tích qui trình nghiệp vụ
- Xác định ràng buộc nghiệp vụ
- Thiết kế qui trình nghiệp vụ

Tại sao mô hình hóa nghiệp vụ?

- Nhằm đảm bảo những giải pháp, những hệ thống cần xây dựng đáp ứng thực sự nhu cầu khách hàng.
- Giảm thiểu rủi ro do những người phát triển không có thông tin đầy đủ về cách thức mà nghiệp vụ được thực hiện
- Xác định đúng vai trò trách nhiệm của con người cũng như định nghĩa những gì được xử lý bởi nghiệp vụ trong việc phát triển hệ thống

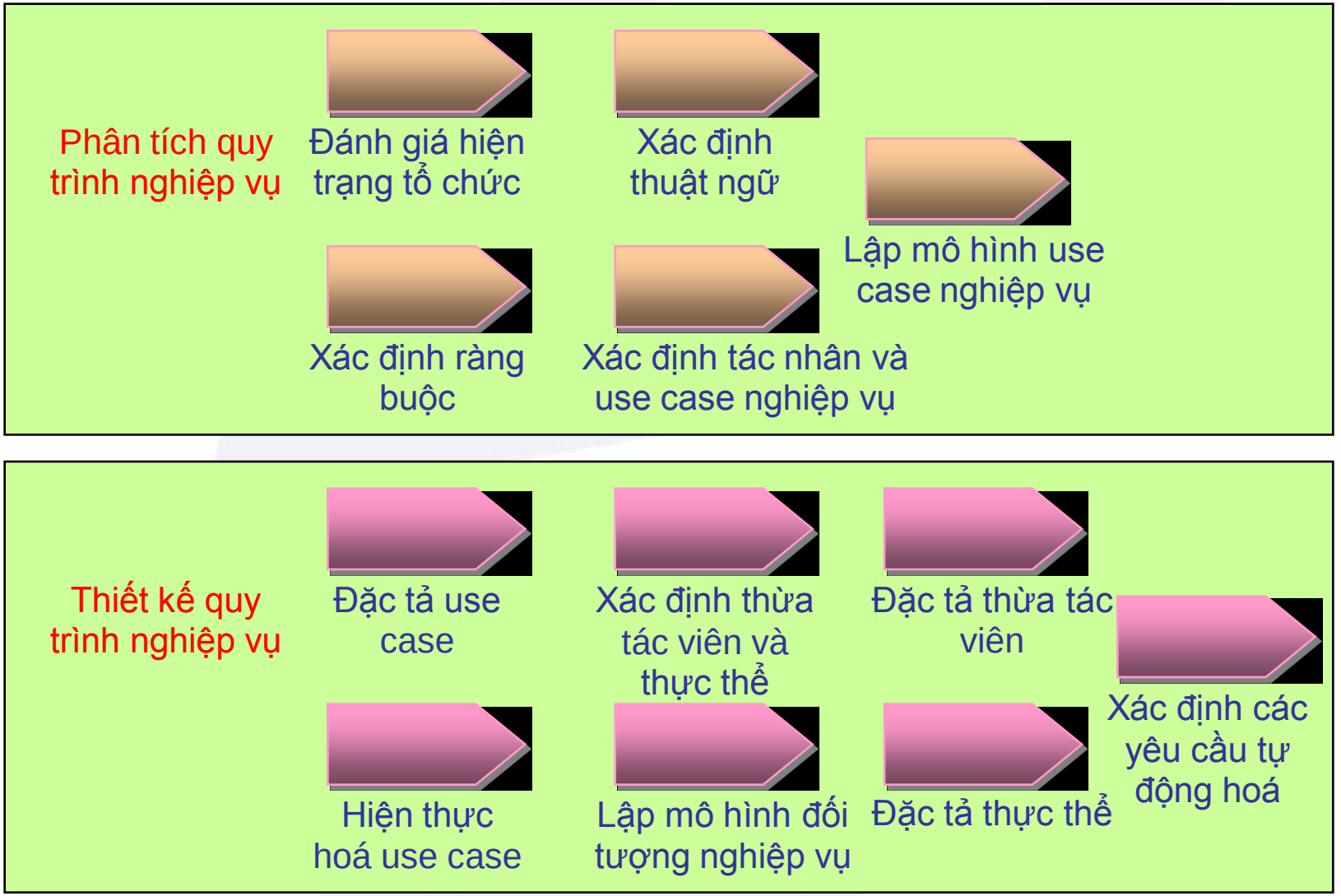
Mục đích mô hình hóa nghiệp vụ?

- Hiểu được cấu trúc và các hoạt động của tổ chức được triển khai hệ thống.
- Hiểu được các vấn đề hiện tại trong tổ chức và xác định các vấn đề cần cải tiến.
- Bảo đảm rằng các khách hàng, người dùng cuối, và các nhà phát triển có sự hiểu biết chung về tổ chức.
- Thiết lập các yêu cầu hệ thống nhằm hỗ trợ tổ chức

Mô hình hoá nghiệp vụ

- Mô hình hóa nghiệp là gì?
- Tại sao phải mô hình hóa nghiệp vụ
- ➢ Luồng công việc của mô hình hóa nghiệp vụ
- Phân tích qui trình nghiệp vụ
- Xác định ràng buộc nghiệp vụ
- Thiết kế qui trình nghiệp vụ

Luồng công việc



Mô hình hoá nghiệp vụ

- Mô hình hóa nghiệp là gì?
- Tại sao phải mô hình hóa nghiệp vụ
- Luồng công việc của mô hình hóa nghiệp vụ
- ➢ Phân tích qui trình nghiệp vụ
- Xác định ràng buộc nghiệp vụ
- Thiết kế qui trình nghiệp vụ

Phân tích qui trình nghiệp vụ

Đánh giá hiện trạng tổ chức

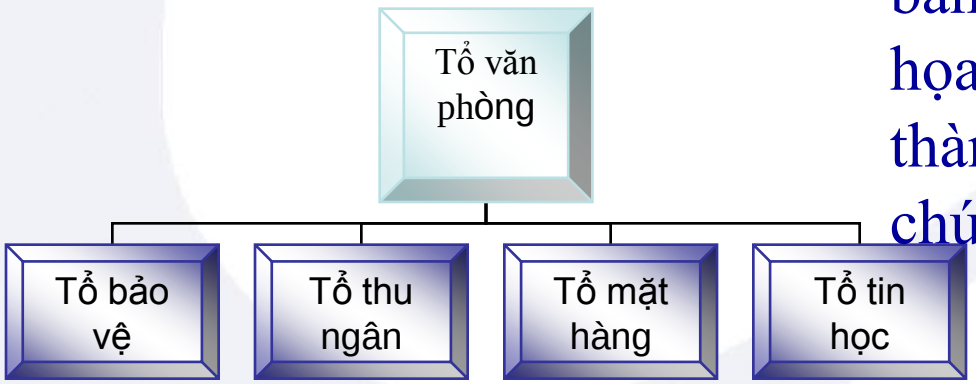
- Mục đích:
  - Đánh giá và nắm bắt thông tin về tổ chức.
  - Xác định các đối tượng liên quan (stakeholder) và khách hàng của hệ thống.
  - Định nghĩa phạm vi của việc mô hình hóa nghiệp vụ.
  - Tán thành những tiềm năng cải tiến và các mục tiêu mới của tổ chức.
  - Mô tả những mục tiêu chính của tổ chức.

Phân tích qui trình nghiệp vụ

Đánh giá hiện trạng tổ chức

- Nắm bắt thông tin về tổ chức:
  - Cơ cấu tổ chức, phân cấp và các vai trò trong hệ thống.
  - Mô tả ngắn gọn các thành phần và mối quan hệ này thông qua sơ đồ tổ chức

Cần mô tả ngắn gọn bằng văn bản vai trò và hoạt động của từng thành phần trong tổ chức



Sơ đồ tổ chức của siêu thị

Phân tích qui trình nghiệp vụ

Đánh giá hiện trạng tổ chức

- Nắm bắt thông tin về tổ chức – ví dụ: mô tả thông tin hoạt động

Tổ văn phòng: *Gồm 1 Giám Đốc và 2 phó Giám Đốc có nhiệm vụ điều phối toàn bộ hoạt động của siêu thị. Tổ phải nắm được tình hình mua bán, doanh thu của siêu thị để báo cáo lại cho ban giám đốc. Việc báo cáo được thực hiện hàng tháng, hàng quý hoặc cũng có khi báo cáo đột xuất theo yêu cầu*

Phân tích qui trình nghiệp vụ

Đánh giá hiện trạng tổ chức

- Nắm bắt và phân tích các yếu tố sau:
  - Hỗ trợ ra quyết định
  - Ưu thế cạnh tranh
  - Hoàn vốn đầu tư: chỉ ra các lợi ích kinh tế rõ ràng (phân tích lợi nhuận)
  - Giảm chi phí (phân tích chi phí)
  - Hỗ trợ cho việc quản lý nghiệp vụ
  - Khả năng thực hiện công việc phải nhanh hơn và tốt hơn

Phân tích qui trình nghiệp vụ

Đánh giá hiện trạng tổ chức

- **Xác định các đối tượng liên quan và khách hàng**
  - Đối tượng liên quan (stakeholder): là những cá nhân chịu ảnh hưởng trực tiếp từ các tác động của hệ thống
  - Khách hàng: người dùng hệ thống, có thể là các stakeholder

Phân tích qui trình nghiệp vụ

Đánh giá hiện trạng tổ chức

- Xác định các đối tượng liên quan và khách hàng

Tên	Đại diện	Vai trò
Người quản lý	Giám đốc, người quản lý siêu thị	Theo dõi tiến trình phát triển của dự án và theo dõi tình hình hoạt động của siêu thị.
Nhân viên bán hàng	Người nhập các thông tin trong hệ thống.	Chịu trách nhiệm trong khâu bán hàng ở siêu thị, duy trì hoạt động của siêu thị.

Tên	Mô tả	Đối tượng liên quan
Người quản lý	Đáp ứng các nhu cầu quản lý siêu thị như hàng hóa, khách hàng, doanh số.	Người quản lý
Nhân viên bán hàng	Đảm bảo rằng hệ thống sẽ đáp ứng các nhu cầu của công việc bán hàng.	Nhân viên bán hàng
Khách hàng	Đáp ứng nhu cầu tra cứu thông tin về hàng hóa có trong siêu thị.	

Phân tích qui trình nghiệp vụ

Đánh giá hiện trạng tổ chức

- Mô tả nhu cầu của từng đối tượng liên quan

Tên đối tượng liên quan/ khách hàng	Độ ưu tiên	Nhu cầu	Giải pháp hiện hành	Giải pháp đề xuất

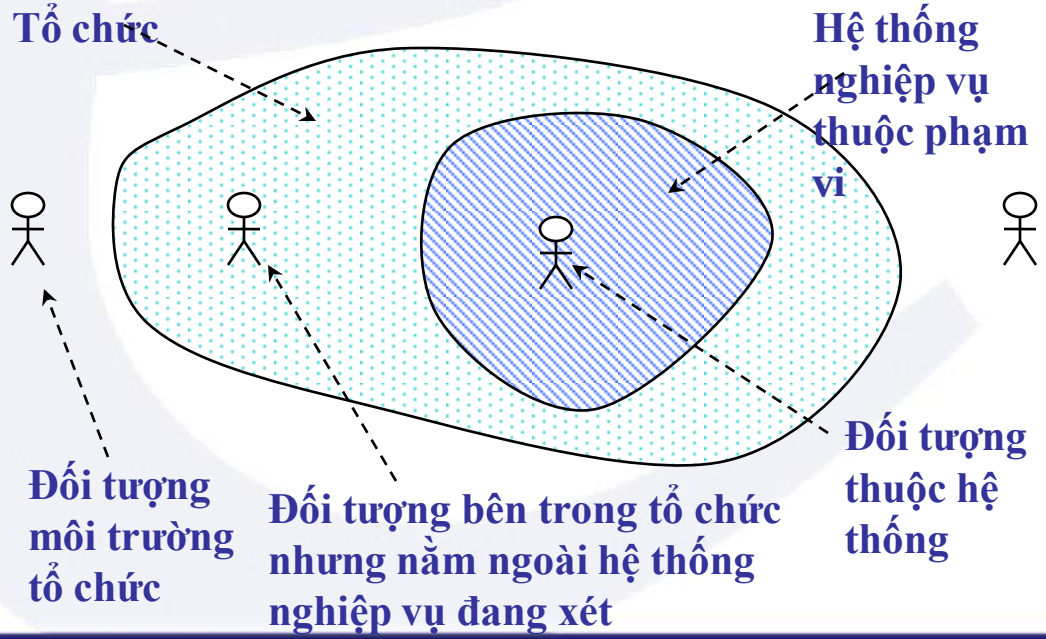
- Ví dụ:

Tên đối tượng liên quan/ khách hàng	Độ ưu tiên	Nhu cầu	Giải pháp hiện hành	Giải pháp đề xuất
Người quản lý	Cao	Xem các báo cáo thống kê theo các yêu cầu khác nhau	Báo cáo thống kê doanh thu	Hiển thị báo cáo theo nhiều tiêu chí khác nhau, thông tin bố trí dễ nhìn và đơn giản nhưng đầy đủ.

Phân tích qui trình nghiệp vụ

Đánh giá hiện trạng tổ chức

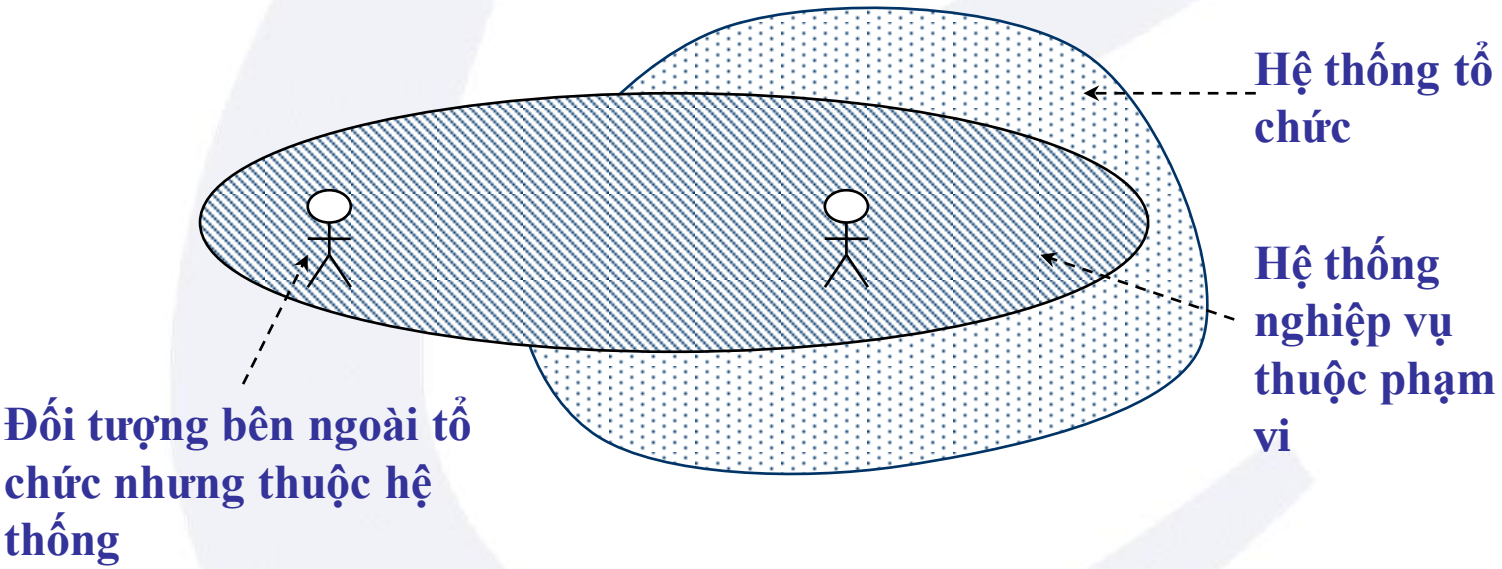
- Giới hạn hệ thống phát triển
  - Xác định ranh giới phát triển hệ thống bằng cách:
    - ✓ Chỉ ra những thực thể nằm ngoài hệ thống
    - ✓ Chỉ ra những thực thể bên trong tổ chức nhưng nằm ngoài hệ thống



Phân tích qui trình nghiệp vụ

Đánh giá hiện trạng tổ chức

- Giới hạn hệ thống phát triển



Các hệ thống thương mại điện tử e-Business, e-Commerce

Phân tích qui trình nghiệp vụ

Đánh giá hiện trạng tổ chức

- Trình bày vấn đề có hệ thống
  - Mẫu trình bày

Vấn đề	mô tả vấn đề
Đối tượng chịu tác động	các đối tượng liên quan bị ảnh hưởng bởi vấn đề
Ảnh hưởng của vấn đề	tác động ảnh hưởng của vấn đề
Một giải pháp thành công	liệt kê một vài lợi ích của một giải pháp thành công

Phân tích qui trình nghiệp vụ

Trình bày vấn đề có hệ thống

- Ví dụ:

Vấn đề	Cơ sở dữ liệu của các khách hàng thân thiết được lưu trữ ở nhiều nơi và không có sự đồng bộ .
Đối tượng chịu tác động	Khách hàng, người quản lý
Ảnh hưởng của vấn đề	Dịch vụ khách hàng thân thiết chỉ thiết lập được ở từng siêu thị. Điều này là bất hợp lý, làm rắc rối trong việc nâng cao dịch vụ khách hàng, làm giảm khả năng cạnh tranh của siêu thị.
Một giải pháp thành công	Nhân viên có thể sử dụng chung một tài khoản (account) cấp cho mỗi khách hàng được dùng ở tất cả siêu thị. Nâng cao khả năng chăm sóc khách hàng của siêu thị tốt hơn từ đó thu hút được khách hàng nhiều hơn, tăng doanh thu của siêu thị. Giúp người quản lý có thể làm tốt công tác quản lý khách hàng, theo dõi tình hình phục vụ khách hàng một cách dễ dàng.

Phân tích qui trình nghiệp vụ

Xác định các thuật ngữ nghiệp vụ

Thuật ngữ	Diễn giải

Thuật ngữ	Diễn giải
Người quản lý	Người quản lý siêu thị và cũng là người quản trị hệ thống. Nguoiquanly được gọi chung cho những người được cấp quyền là "Quản lý", có thể bao gồm giám đốc, phó giám đốc, kế toán, nhân viên tin học, …
Nhân viên bán hàng	Là nhân viên làm việc trong siêu thị. Nhân viên bán hàng, đứng ở quầy thu tiền và tính tiền cho khách hàng. Thông qua các mã vạch quản lý trên từng mặt hàng được nhân viên bán hàng nhập vào hệ thống thông qua một đầu đọc mã vạch.
Khách hàng thân thiết	Khách hàng thân thiết của siêu thị hay khách hàng đăng ký tham gia chương trình khách hàng thân thiết của siêu thị.

Mô hình hóa nghiệp vụ

Các khái niệm

Tác nhân (Business Actor)	Một người hay vật bên ngoài quy trình nghiệp vụ tương tác với nghiệp vụ đó.
Mô hình hóa Nghiệp vụ	Bao gồm toàn bộ các kỹ thuật mô hình hóa để giúp ta lập mô hình nghiệp vụ một cách trực quan.
Mô hình đối tượng (Business Object)	Đây là một mô hình mô tả việc hiện thực hóa business use case.
Quy trình nghiệp vụ (Business Process)	Một nhóm các hành động có quan hệ với nhau, sử dụng tài nguyên của tổ chức để cung cấp các kết quả rõ ràng cho các mục tiêu của tổ chức. Trong RUP, các business process được xác định thông qua các business use case và các hiện thực hóa business use-case.
Business Use Case	Một business use case xác định một tập hợp các thể hiện business use-case. Mỗi thể hiện là một chuỗi các hành động tuần tự mà nghiệp vụ thực hiện để đem lại một kết quả rõ ràng cho một business actor cụ thể. Một lớp business use-case chứa tất cả các luồng công việc chính và phụ có liên quan để tạo ra kết quả trên.

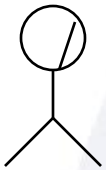
Mô hình hóa nghiệp vụ

Các khái niệm

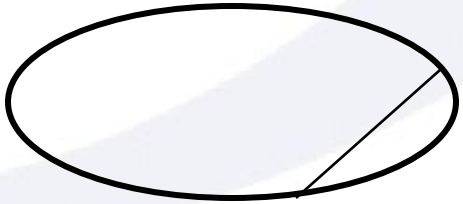
Mô hình use case (Business Use-case model)	Đây là một mô hình của các chức năng nghiệp vụ. Nó được dùng làm đầu vào chủ yếu để xác định các vai trò trong tổ chức.
Hiện thực hóa Business Use-case	Mô tả cách thức mà luồng công việc của một business use case được hiện thực hóa như thế nào trong mô hình Business Object, dưới dạng các business object cộng tác với nhau.
Thừa tác viên (Business Worker)	Một vai trò hoặc một tập hợp các vai trò bên trong nghiệp vụ. Một business worker tương tác với những business worker khác và thao tác với những business entity khi tham gia vào các hiện thực hóa business use-case.

Mô hình hóa nghiệp vụ

Xác định business actor và business use case



Tác nhân (Business actor)



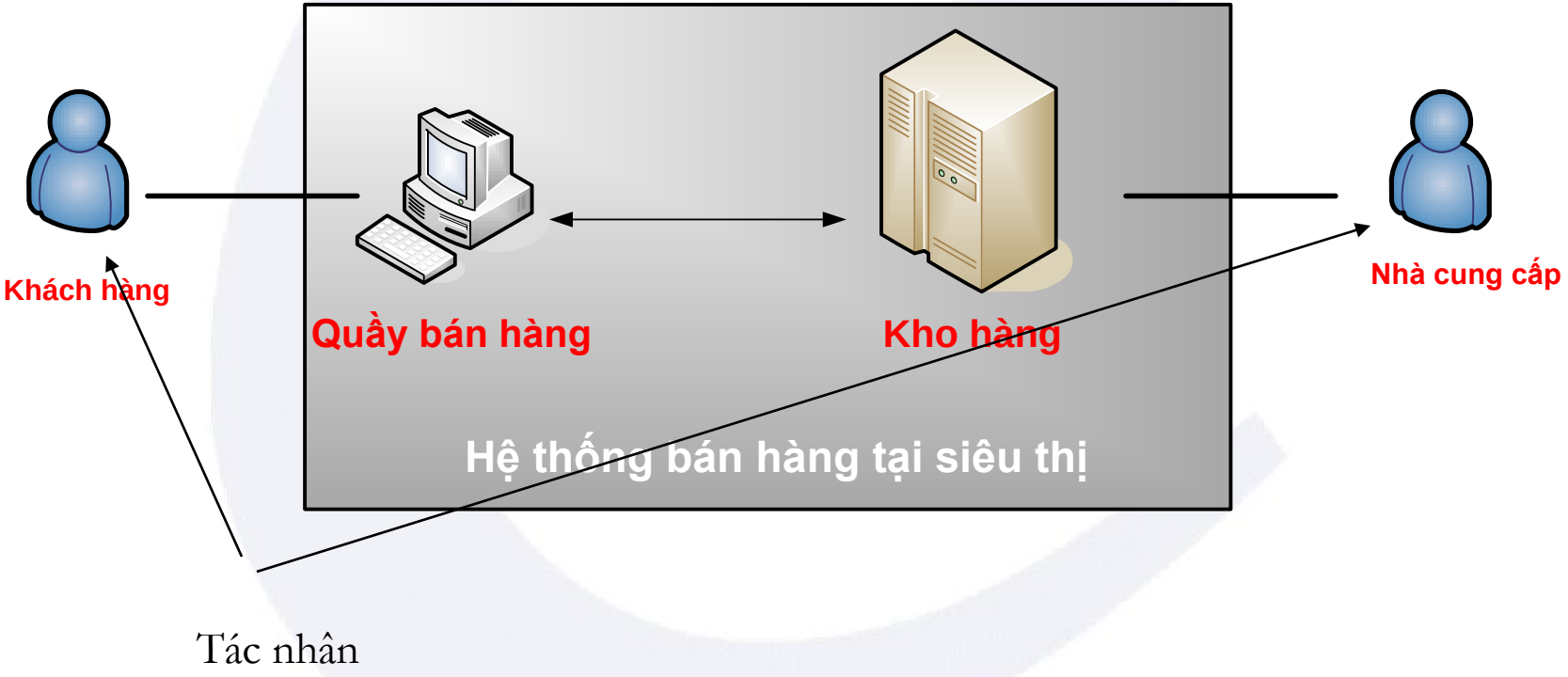
Use case (Business use case)

- **Tác nhân**: bất kỳ đối tượng nào bên ngòai tổ chức nghiệp vụ:
  - khách hàng, nhà cung cấp, đối tác, đồng nghiệp ở những nghiệp vụ không được mô hình hóa,…
  - Một hệ thống hay một tổ chức khác

Mô hình hóa nghiệp vụ

Xác định tác nhân và use case nghiệp vụ

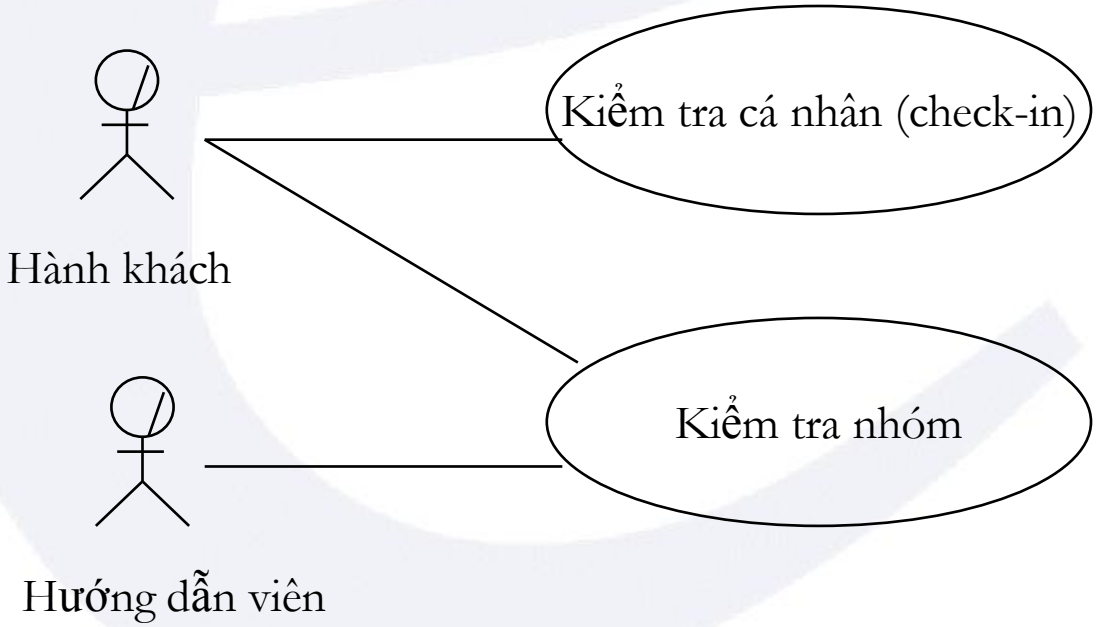
- Ví dụ:



Mô hình hóa nghiệp vụ

Xác định tác nhân và use case nghiệp vụ

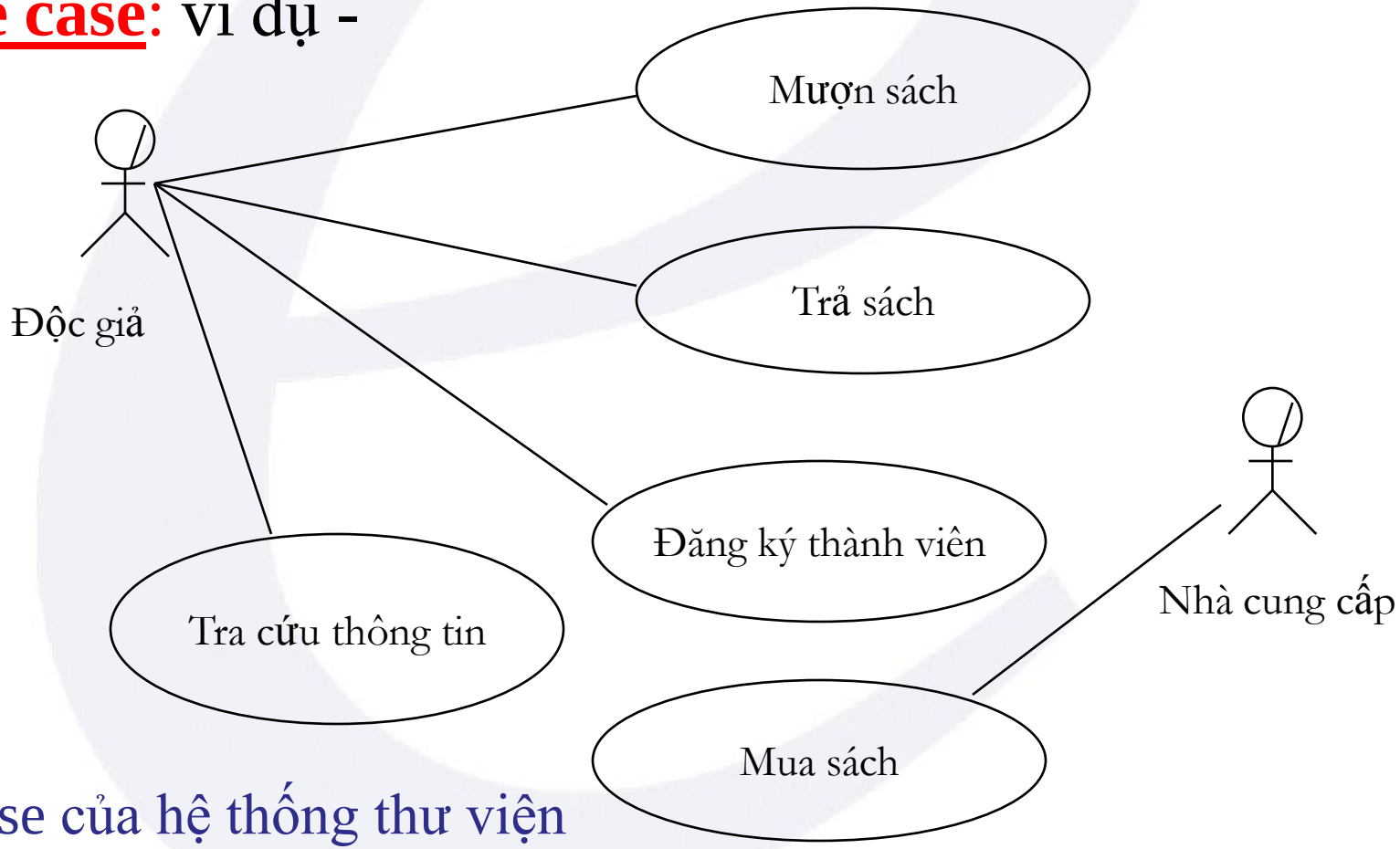
- **Use case**:
  - use-case là một chuỗi các hành động được thực hiện trong nghiệp vụ và tạo ra một giá trị kết quả có thể quan sát được cho một tác nhân riêng lẻ của nghiệp vụ



Mô hình hóa nghiệp vụ

Xác định tác nhân và use case nghiệp vụ

- **Use case**: ví dụ -



Các use case của hệ thống thư viện

Mô hình hóa nghiệp vụ

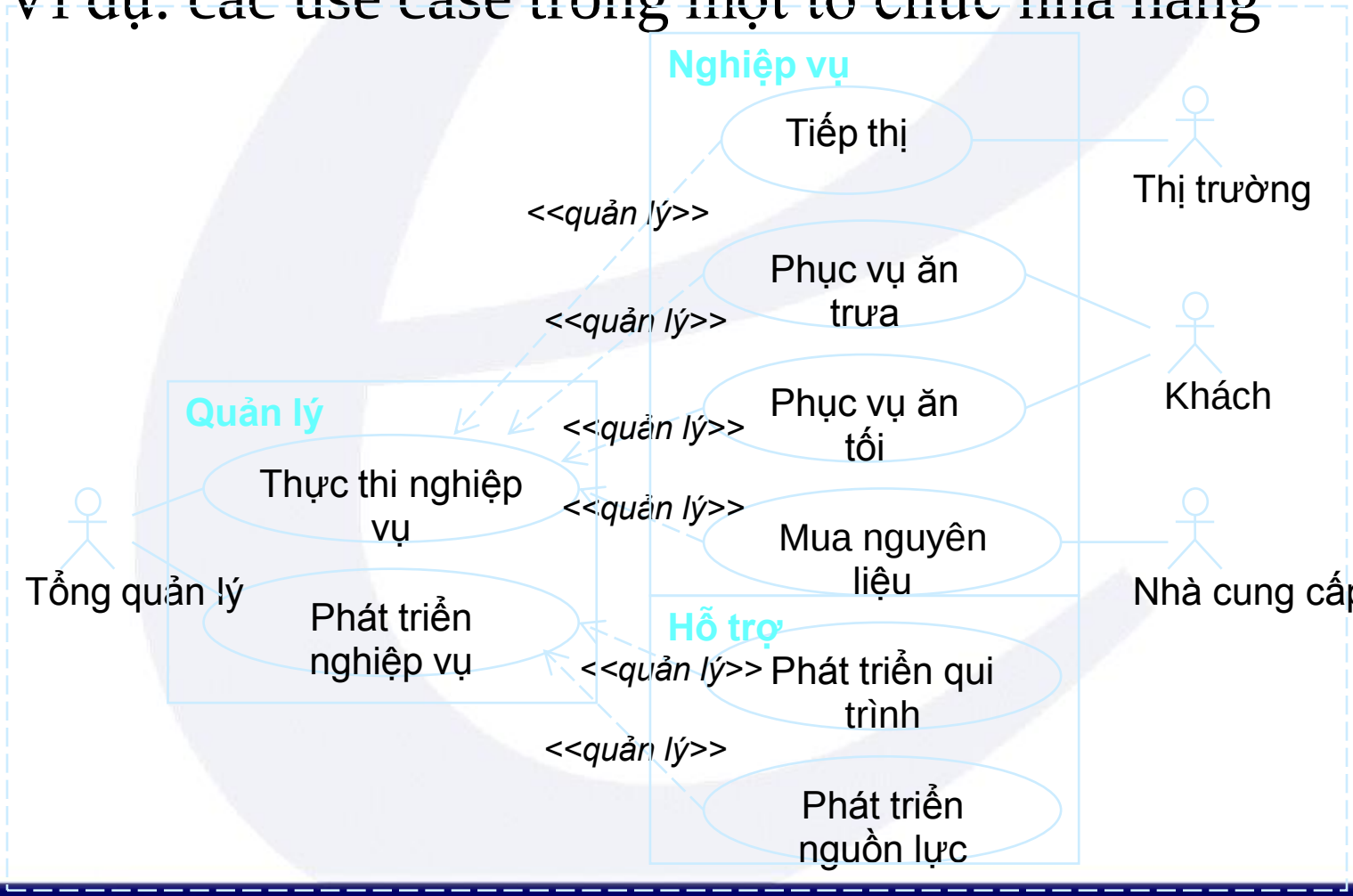
Xác định business actor và business use case

- Phân loại use case nghiệp vụ:
  - Các hoạt động liên quan đến công việc của tổ chức, thường được gọi là các qui trình nghiệp vụ
  - Các hoạt động mang đặc điểm hỗ trợ: quản trị hệ thống, dọn dẹp, an ninh ,…
  - Công việc quản lý

Mô hình hóa nghiệp vụ

Xác định tác nhân và use case nghiệp vụ

- Ví dụ: các use case trong một tổ chức nhà hàng



Mô hình hóa nghiệp vụ

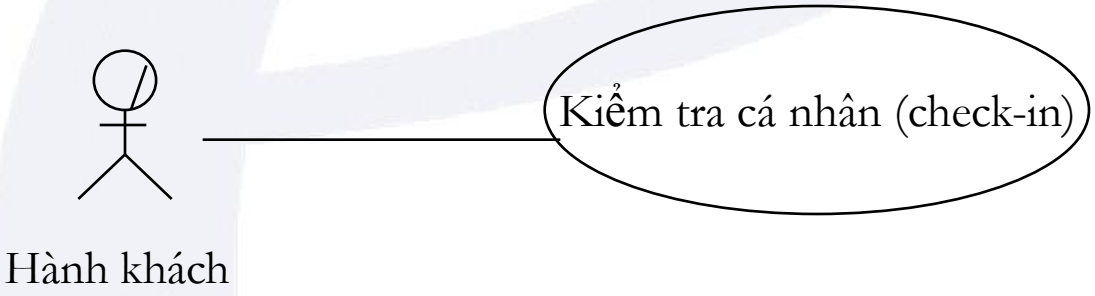
Xác định tác nhân và use case nghiệp vụ

- Thể hiện của các use case: khi mô tả một business use case nên tránh mô tả cụ thể cho một thể hiện mà dựa trên một tập các thể hiện
- Luồng công việc trong use case: biểu diễn sử dụng văn bản mô tả hoặc sơ đồ hoạt động

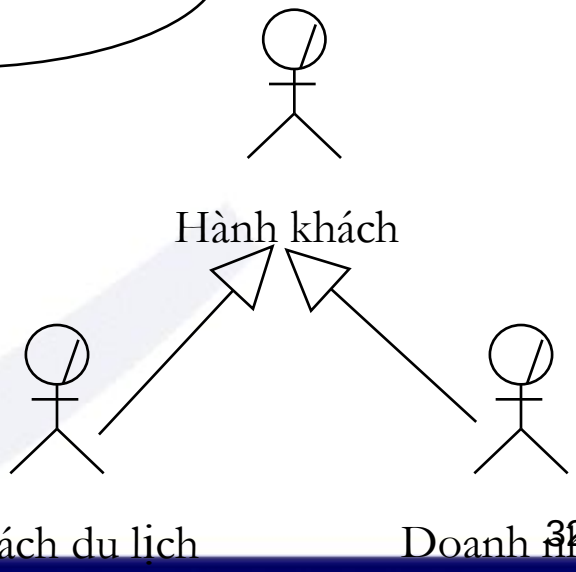
Mô hình hóa nghiệp vụ

Cấu trúc mô hình use case

- Mối kết hợp
  - Mối kết hợp giao tiếp (communicates-association):



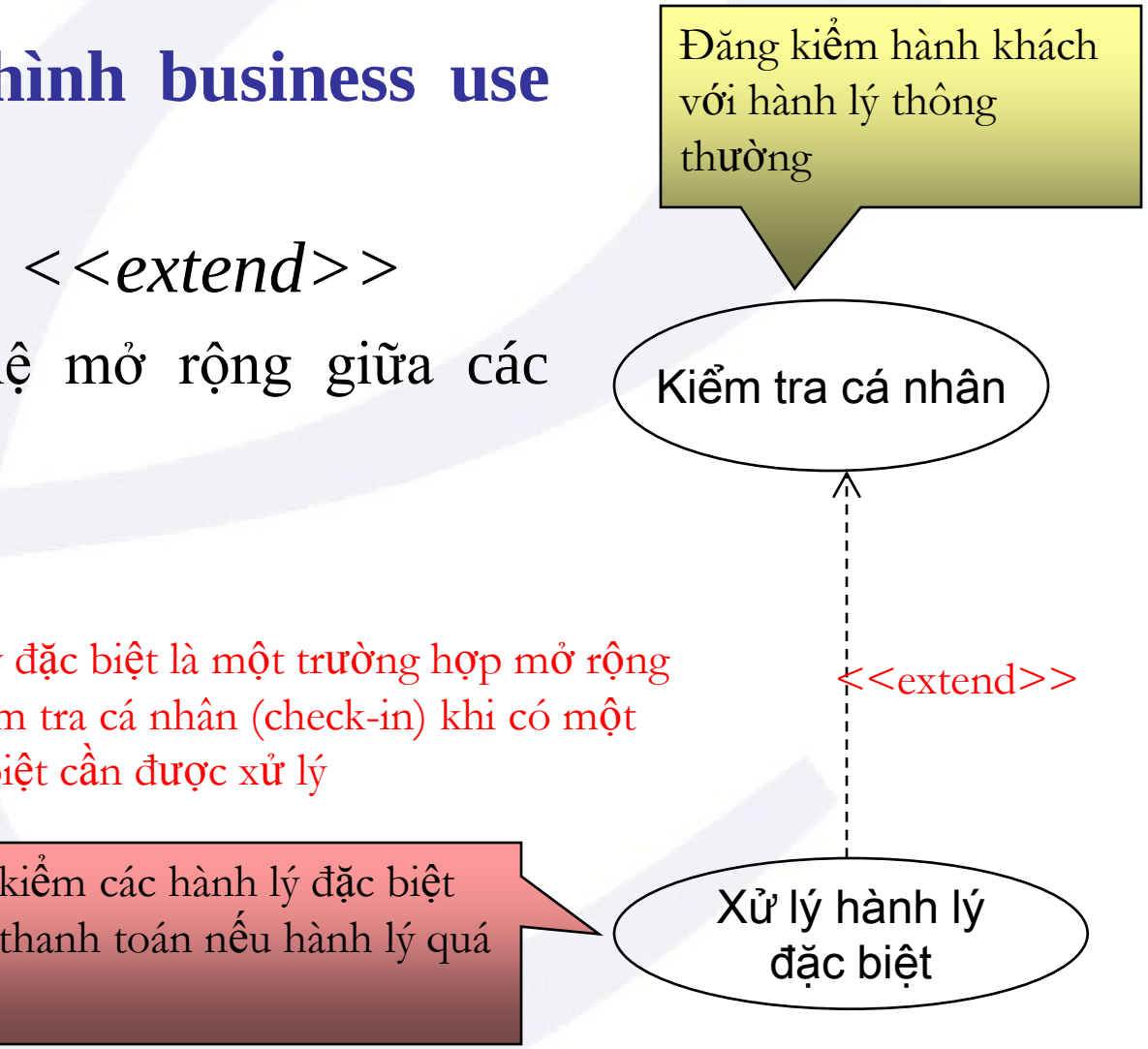
  - Mối quan hệ tổng quát hóa giữa các tác nhân



Mô hình hóa nghiệp vụ

Cấu trúc mô hình business use case

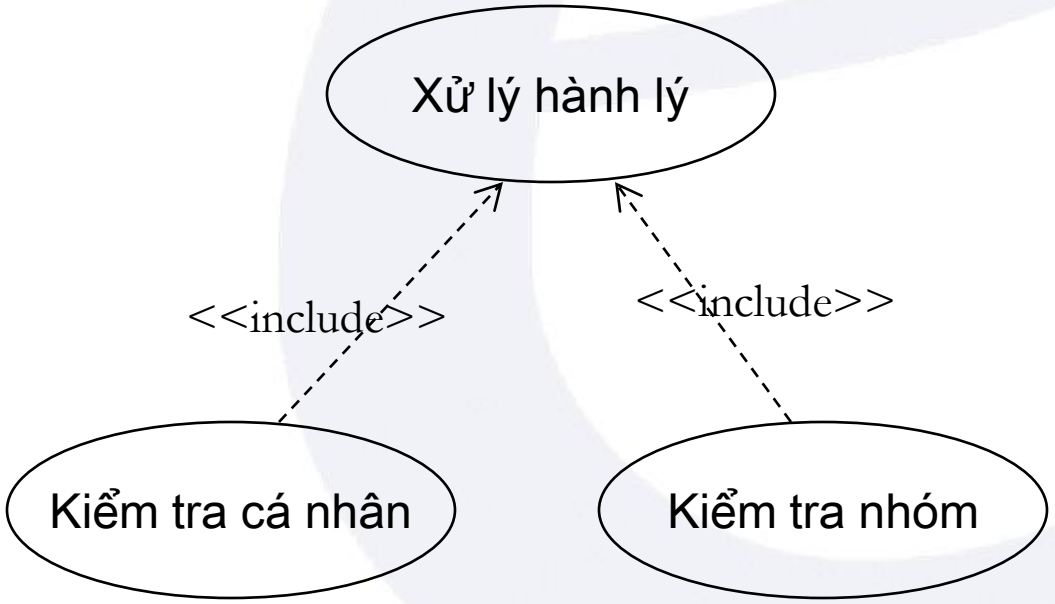
- Mối kết hợp - <<*extend*>>
  - Mối quan hệ mở rộng giữa các Use Case



Mô hình hóa nghiệp vụ

Cấu trúc mô hình business use case

- Mối kết hợp
  - Mối quan hệ bao hàm giữa các Use Case

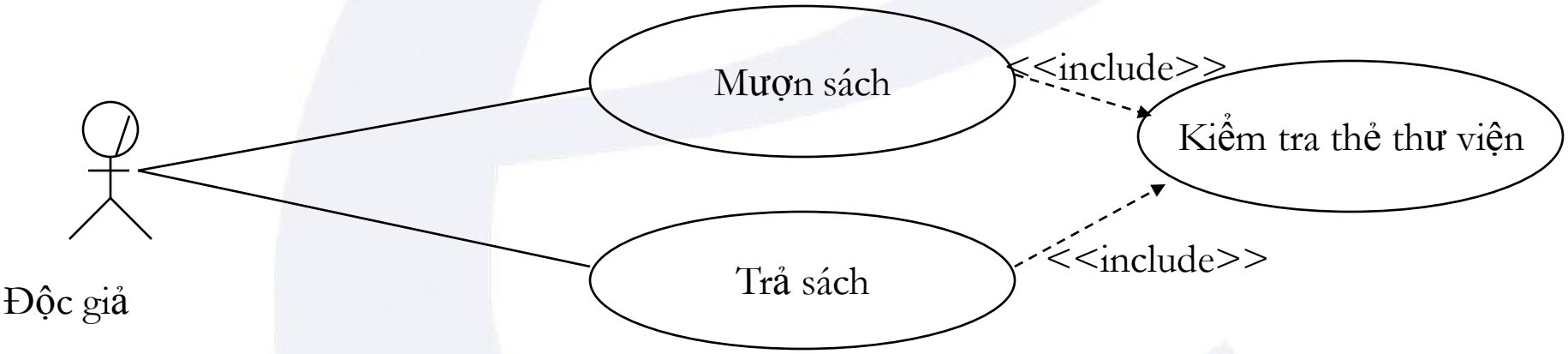


Các hoạt động kiểm tra cá nhân và kiểm tra nhóm đều xử lý hành lý của hành khách. Do đó, tách hoạt động này thành một use case và tạo mối liên kết <<include>> từ hai use case kia tới nó

Mô hình hóa nghiệp vụ

Cấu trúc mô hình use case

- Mối kết hợp
  - Mối quan hệ bao hàm giữa các Use Case

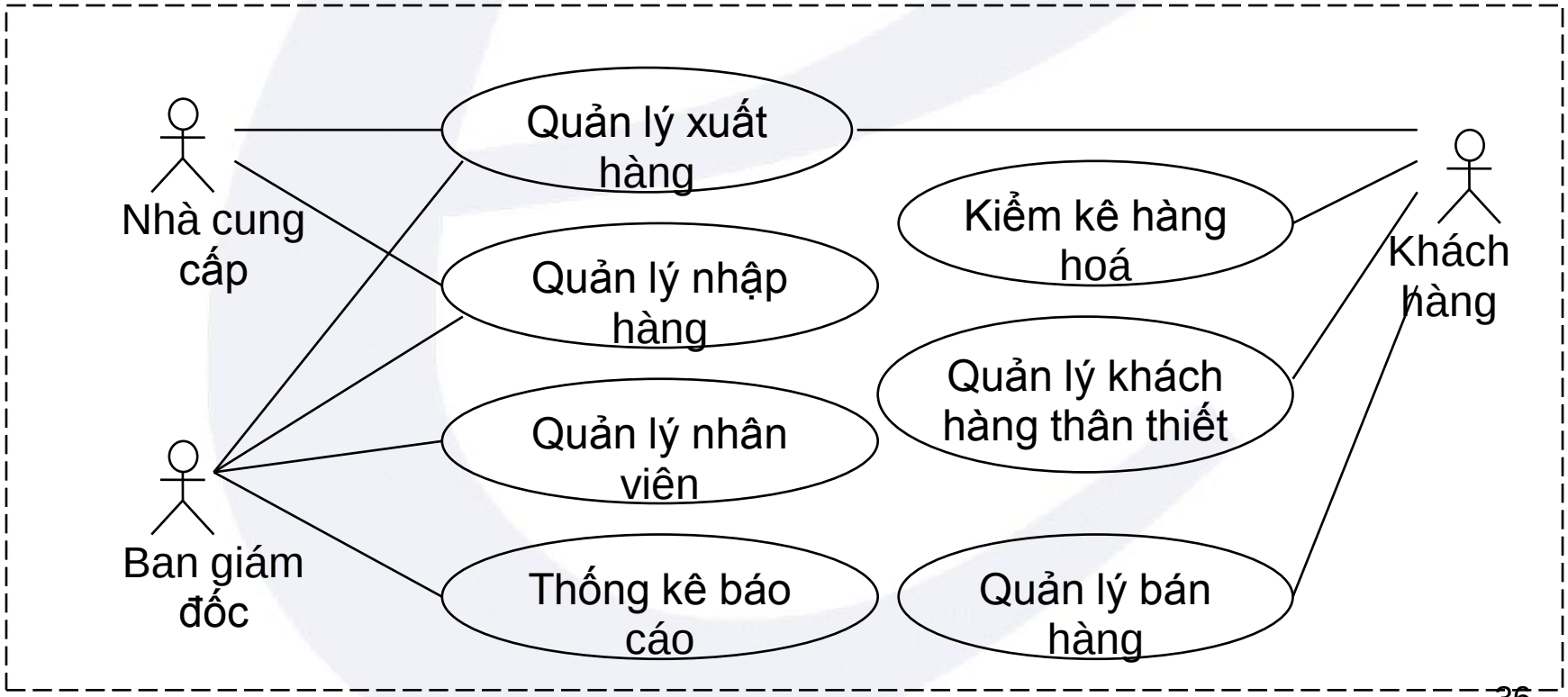


Khi độc giả đến mượn hoặc trả sách thì thư viện đều kiểm tra thẻ thư viện của độc giả → tạo một use case Kiểm tra thẻ thư viện và tạo liên kết <<include>> từ hai use case đó đến use case này

Mô hình hóa nghiệp vụ

Cấu trúc mô hình use case

- Ví dụ: mô hình use case của siêu thị - sơ đồ ngữ cảnh



Mô hình hóa nghiệp vụ

Mô tả use case

- Nội dung của một use case được mô tả ban đầu theo hai phần:

Giới thiệu về use case
Các dòng cơ bản (basic flow): bao gồm các hoạt động chính và thứ tự mô tả nội dung chính của use case
Các thay thế (alternative flow): mô tả các nhánh hoạt động bất thường để xử lý ngoại lệ ngoài các dòng chính

Mô hình hóa nghiệp vụ

Mô tả use case

- Ví dụ: mô tả use case mượn sách

Use case bắt đầu khi một có đọc giả đến mượn sách. Mục tiêu của use case nhằm xử lý mượn sách cho đọc giả
Các dòng cơ bản: 1. Xác định thẻ thư viện của đọc giả: nhân viên yêu cầu đọc giả xuất trình thẻ thư viện để kiểm tra 2. Xác định thông tin nợ sách: kiểm tra thông tin các sách đang nợ của đọc giả 3. Ghi nhận thông tin lần mượn: cập nhật vào hệ thống thông tin về lần mượn của đọc giả 4. Gởi sách cho đọc giả và thông báo ngày giới hạn trả sách

Mô hình hóa nghiệp vụ

Mô tả use case

- Ví dụ: mô tả use case mượn sách

Các dòng thay thế:

-Xử lý thẻ hết hạn: nếu thẻ sinh viên của đọc giả hết hạn, thủ thư sẽ thông báo cho đọc giả và yêu cầu làm thẻ mới

-Xử lý không cho mượn: nếu số lượng sách mà đọc giả đang mượn >3, thủ thư sẽ từ chối lần mượn của đọc giả

Mô hình hóa nghiệp vụ

Mô tả use case

- Ví dụ: mô tả use case Đăng ký

Use case mô tả hoạt động đăng kiểm tại quầy đăng ký khi hành khách tới để đăng ký đi chuyến bay của mình.
Các dòng cơ bản: 1. Tìm kiếm chỗ ngồi: sau khi nhận vé từ hành khách, nhân viên sẽ tìm một chỗ ngồi cho hành khách từ hệ thống. Hệ thống sẽ đánh dấu chỗ đó không còn trống. 2. In thẻ lên máy bay: in thẻ lên máy bay cho hành khách. 3. Xử lý hành lý: kiểm tra vá xác nhận hành lý, in ra thẻ đánh dấu hành lý và thẻ kiểm soát hành lý cho nhân viên.
Các dòng thay thế 1. Xử lý hành lý đặc biệt: xử lý các hành lý chứa một loại hàng đặc biệt hoặc quá nặng (được mô tả trong use case ….).

Mô hình hóa nghiệp vụ

Đánh giá kết quả

- Tất cả các nghiệp vụ cần thiết đã được xác định chưa?
- Có xác định được use case dư thừa nào không?
- Hành vi của mỗi use case có theo đúng thứ tự không?
- Luồng công việc của mỗi use case có hoàn chỉnh không?
- Tìm thấy được tất cả các use case ?

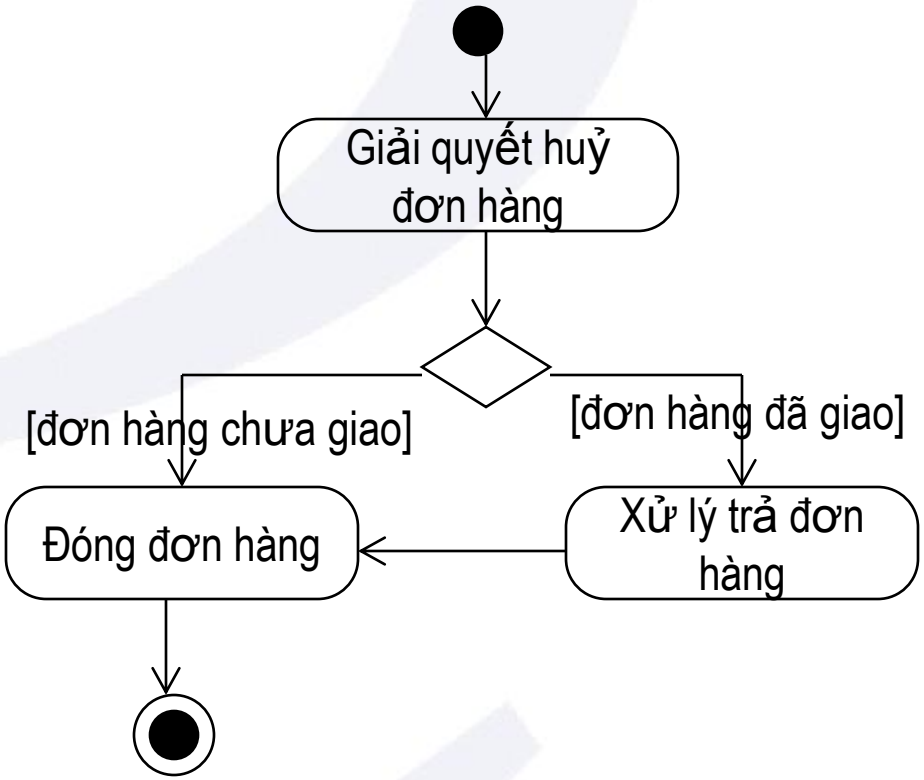
Mô hình hoá nghiệp vụ

- Mô hình hóa nghiệp là gì?
- Tại sao phải mô hình hóa nghiệp vụ
- Luồng công việc của mô hình hóa nghiệp vụ
- Phân tích qui trình nghiệp vụ
- ➢ Xác định ràng buộc nghiệp vụ
- Thiết kế qui trình nghiệp vụ

Xác định ràng buộc nghiệp vụ

Nguyên tắc ràng buộc

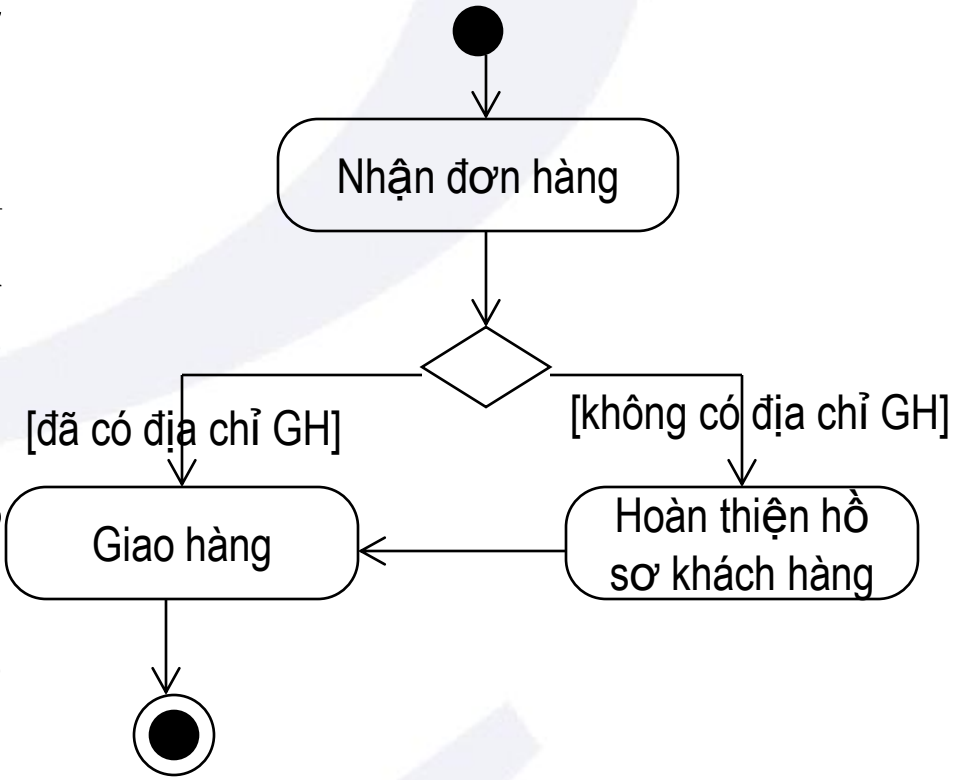
- Nguyên tắc kích hoạt và phản ứng:
  - Là những ràng buộc hay điều kiện xác định khi nào một hành động xảy ra
  - Ví dụ: ràng buộc trong hoạt động huỷ một đơn hàng



WHEN một đơn đặt hàng bị hủy bỏ
IF hàng chưa được vận chuyển
THEN kết thúc đơn đặt hàng

Xác định ràng buộc nghiệp vụ

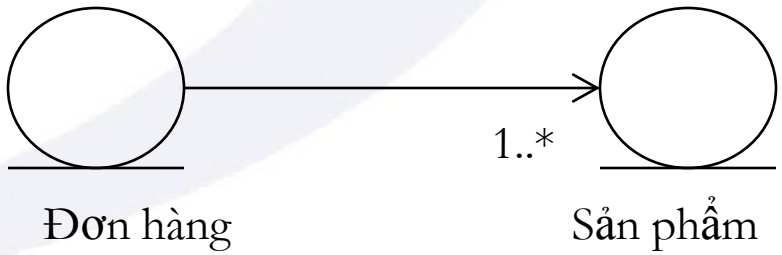
- Nguyên tắc ràng buộc thao tác
  - Đây là những điều kiện phải thỏa trước và sau thao tác để bảo đảm thao tác đó hoạt động đúng
  - Ví dụ: Trong một tổ chức quản lý đặt hàng, nguyên tắc sau đây có thể xảy ra



Vận chuyển Hàng hóa đến chỗ Khách hàng
ONLY IF Khách hàng có địa chỉ

Xác định ràng buộc nghiệp vụ

- Nguyên tắc ràng buộc cấu trúc
  - Xác định các quy định và điều kiện về các lớp, đối tượng, và các mối quan hệ giữa chúng
  - Ví dụ:



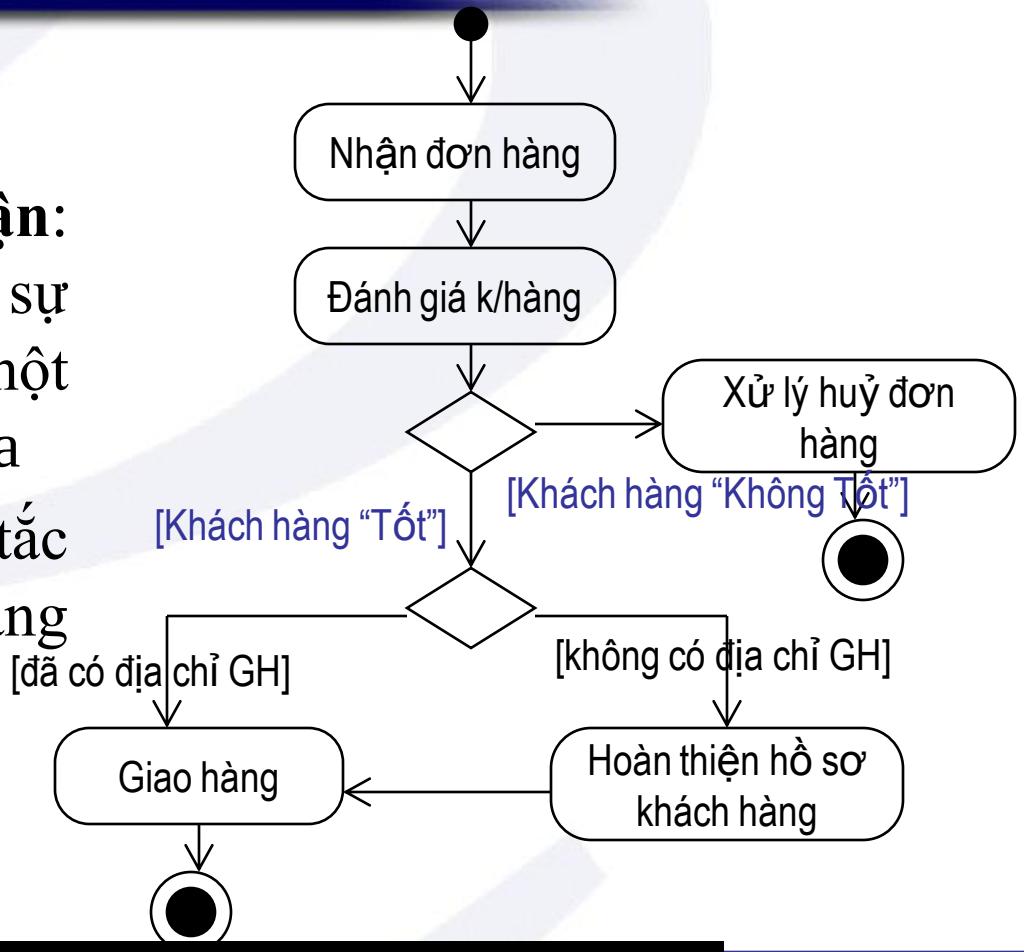
IT MUST ALWAYS HOLD THAT

Một đơn đặt hàng tham chiếu đến tối thiểu một sản phẩm

Xác định ràng buộc nghiệp vụ

Nguyên tắc diễn dịch:

- **Các nguyên tắc suy luận**: Xác định rằng nếu một số sự kiện nhất định là đúng, một kết luận có thể được suy ra
- Ví dụ: Thiết lập nguyên tắc sau đây để xác định trạng thái của một khách hàng

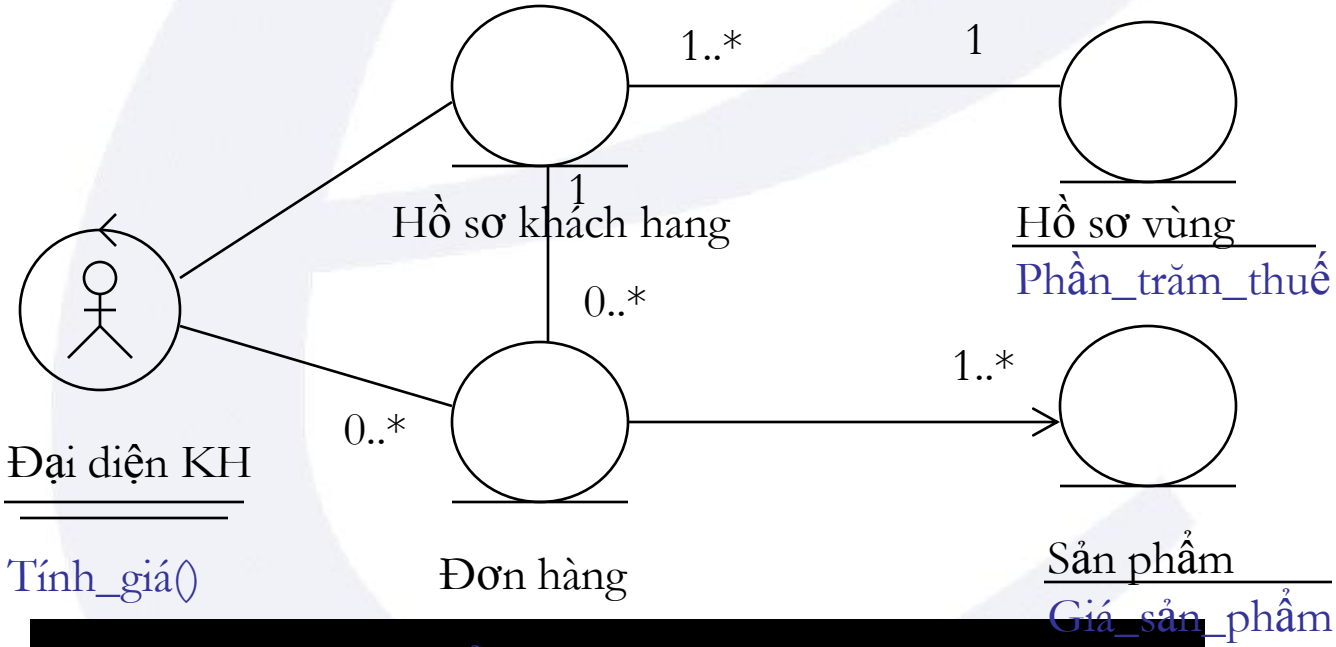


Một Khách hàng là một Khách hàng Tốt IF AND ONLY IF
Những hóa đơn chưa thanh toán gửi đến Khách hàng đều ít hơn 30 ngày

Xác định ràng buộc nghiệp vụ

Nguyên tắc diễn dịch:

- **Nguyên tắc tính toán các sự kiện:**



Giá một sản phẩm được tính toán như sau:
giá sản phẩm * (1 + phần trăm thuế/ 100)

Mô hình hoá nghiệp vụ

- Mô hình hóa nghiệp là gì?
- Tại sao phải mô hình hóa nghiệp vụ
- Luồng công việc của mô hình hóa nghiệp vụ
- Phân tích qui trình nghiệp vụ
- Xác định ràng buộc nghiệp vụ

➢ Thiết kế qui trình nghiệp vụ

Thiết kế qui trình nghiệp vụ

- Đặc tả use case nghiệp vụ
- Xác định thừa tác viên (worker) vụ thực thể (entity) nghiệp vụ
- Hiện thực hóa use case nghiệp vụ
- Lập cấu trúc mô hình đối tượng nghiệp vụ (business object)
- Đặc tả thừa tác viên nghiệp vụ
- Đặc tả thực thể nghiệp vụ
- Xác định các yêu cầu tự động hóa

Thiết kế qui trình nghiệp vụ

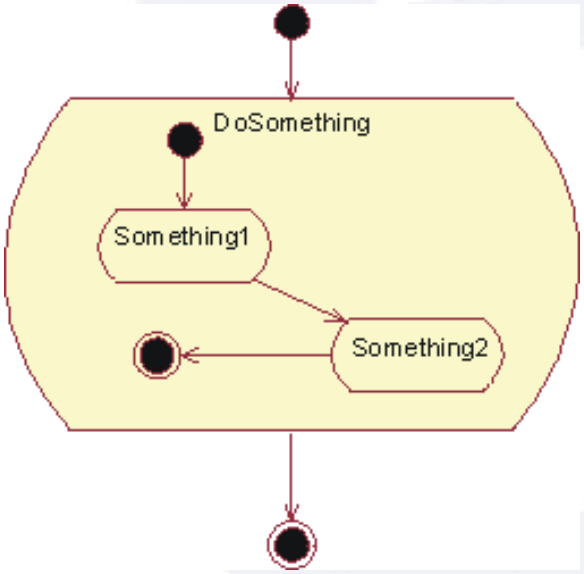
Đặc tả use case nghiệp vụ

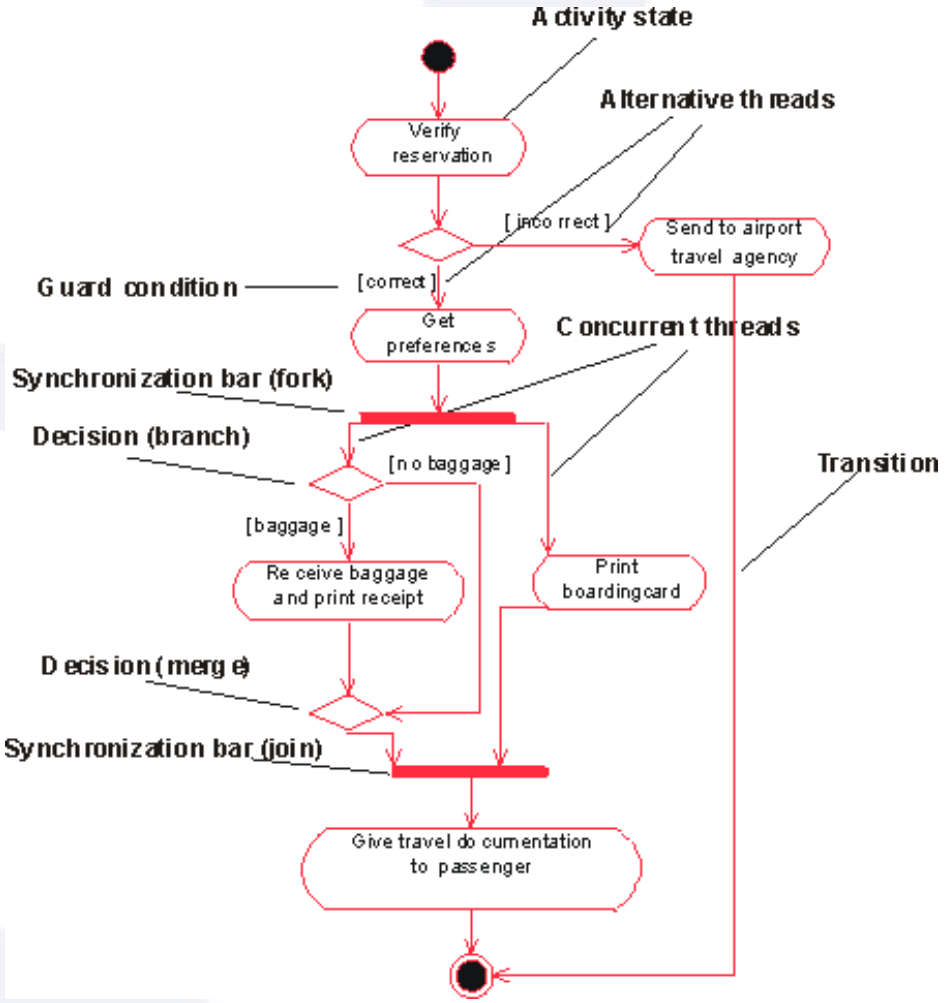
- Xác định sự tương tác giữa tác nhân và use case nghiệp vụ
- Mô tả luồng công việc chính
- Những luồng bất thường và tùy chọn:
  - Những luồng sự kiện con tham gia phần lớn luồng công việc chính.
  - Những luồng công việc bất thường giúp luồng công việc chính rõ ràng hơn.
  - Những luồng sự kiện con xảy ra ở những khoảng thời gian khác nhau trong cùng một luồng công việc và chúng có thể được thực thi

Thiết kế qui trình nghiệp vụ

Đặc tả use case nghiệp vụ

- Dùng sơ đồ hoạt động (activity) để mô tả luồng công việc

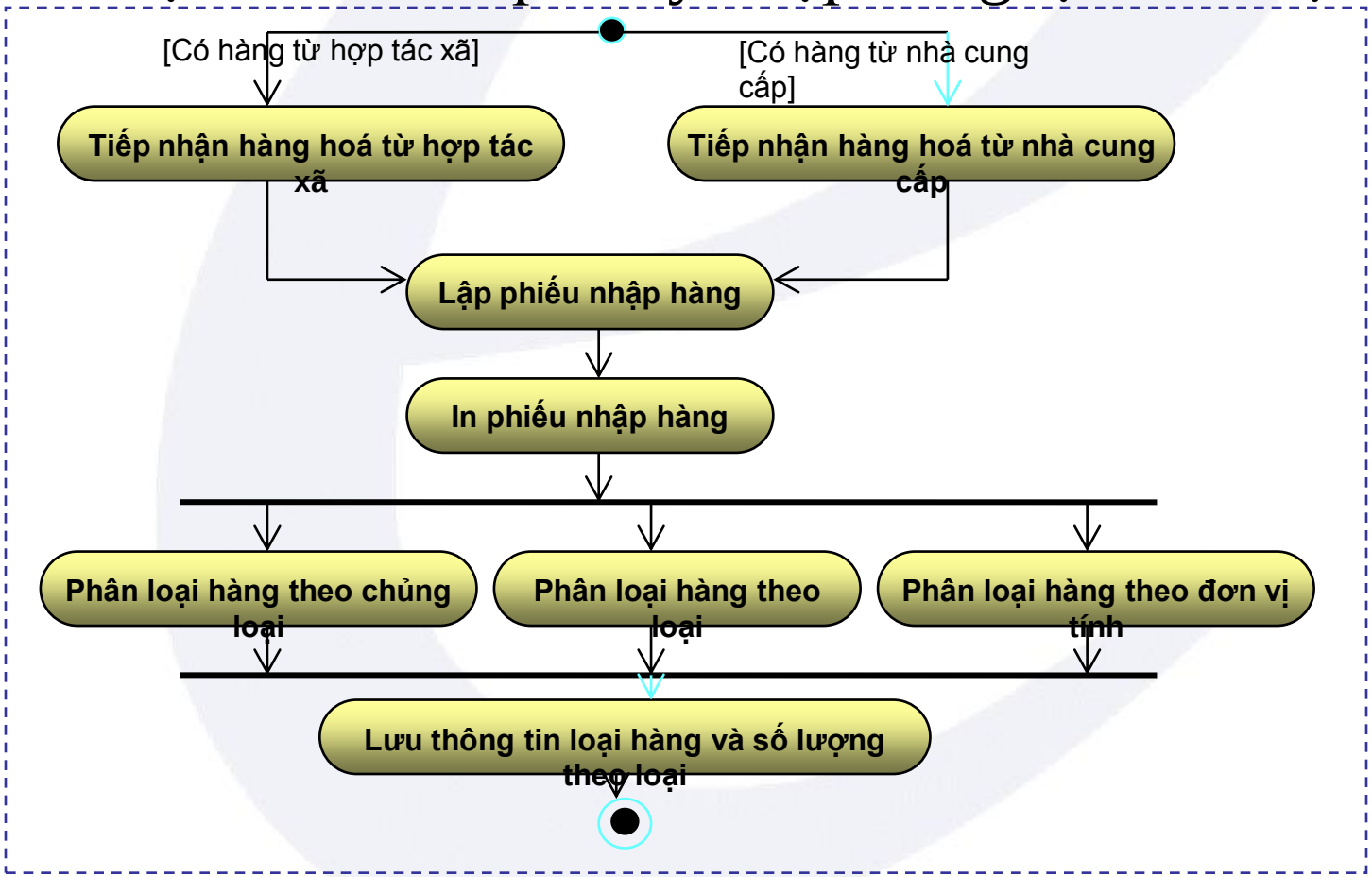




Thiết kế qui trình nghiệp vụ

Đặc tả use case nghiệp vụ

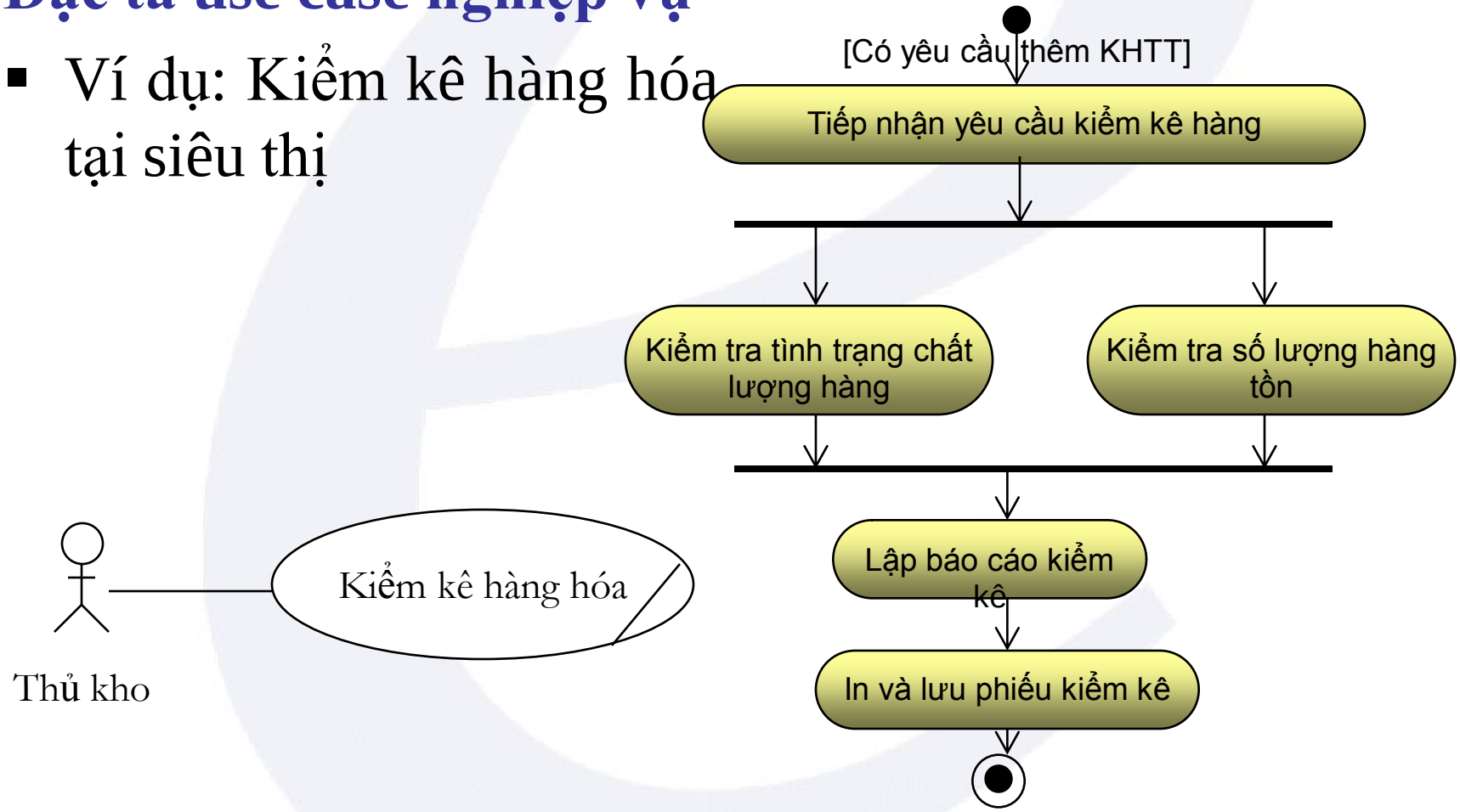
- Ví dụ: Use case quản lý nhập hàng tại siêu thị



Thiết kế qui trình nghiệp vụ

Đặc tả use case nghiệp vụ

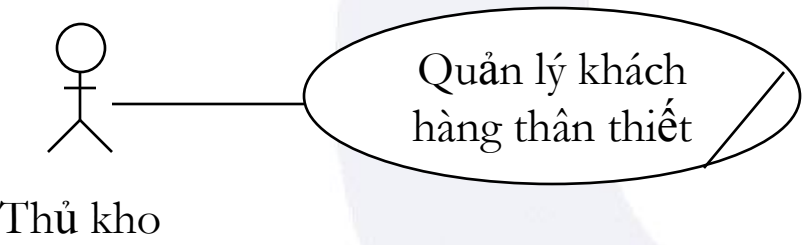
- Ví dụ: Kiểm kê hàng hóa tại siêu thị

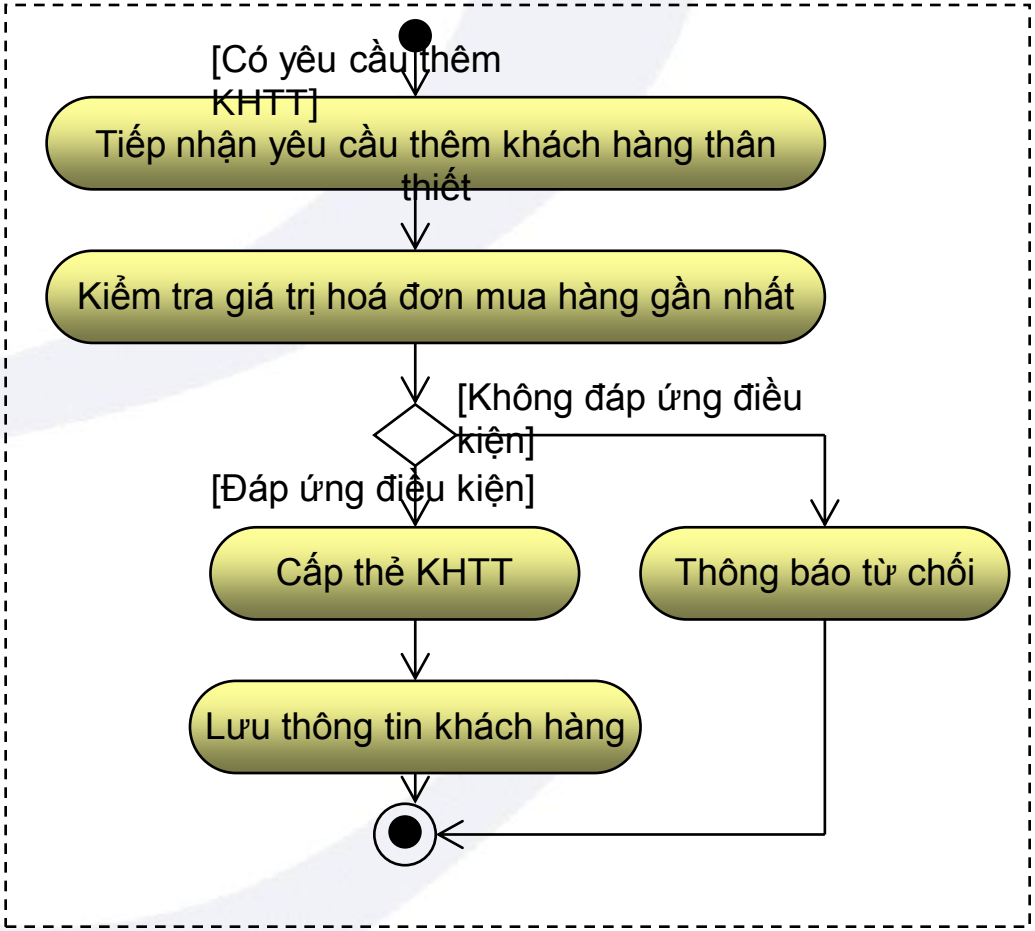


Thiết kế qui trình nghiệp vụ

Đặc tả use case nghiệp vụ

- Ví dụ: Kiểm kê hàng hóa tại siêu thị

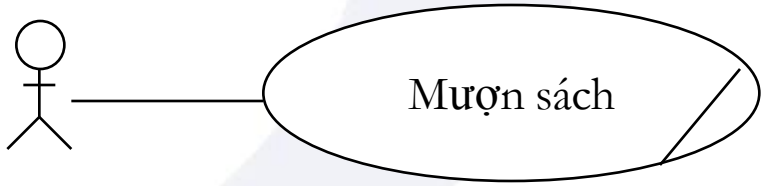


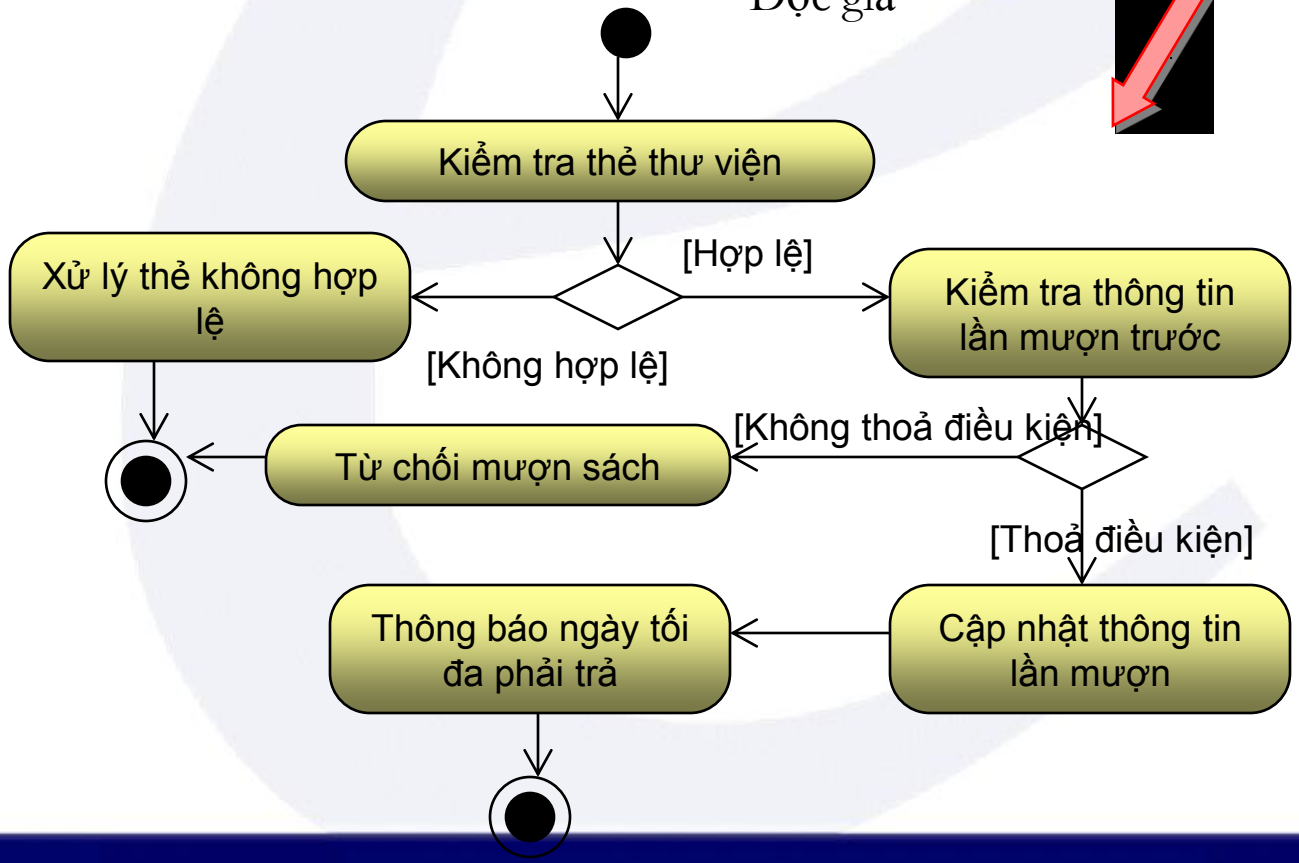


Thiết kế qui trình nghiệp vụ

Đặc tả use case nghiệp vụ

- Ví dụ: Use case mượn sách

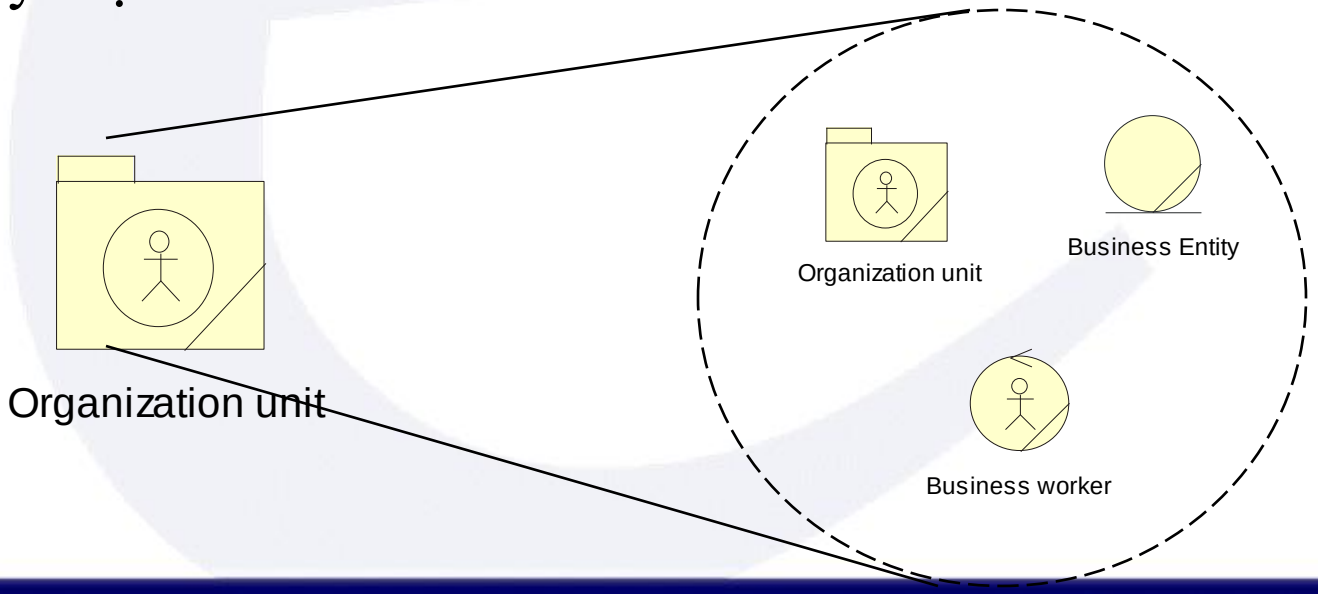




Thiết kế qui trình nghiệp vụ

Xác định thừa tác viên và thực thể nghiệp vụ

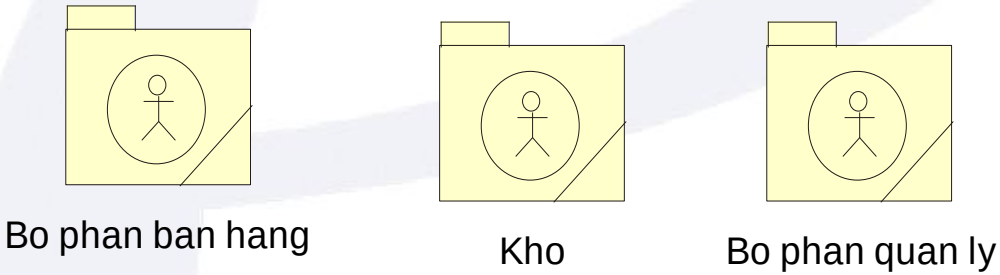
- Xác định đơn vị tổ chức:
  - Một đơn vị tổ chức bao gồm các thừa tác viên, thực thể, và các đơn vị tổ chức khác có liên quan đến nhau theo một số tiêu chí nào đó
  - Ký hiệu:



Thiết kế qui trình nghiệp vụ

Xác định thừa tác viên và thực thể nghiệp vụ

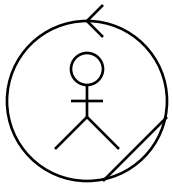
- Xác định đơn vị tổ chức:



Thiết kế qui trình nghiệp vụ

Xác định thừa tác viên và thực thể nghiệp vụ

- Xác thừa tác viên nghiệp vụ:
  - Một thừa tác viên biểu diễn một vai trò hay một tập các vai trò trong nghiệp vụ
  - Tương tác với các thừa tác viên khác và thao tác với các thực thể trong khi tham gia hoạt động của use case
  - Ký hiệu:

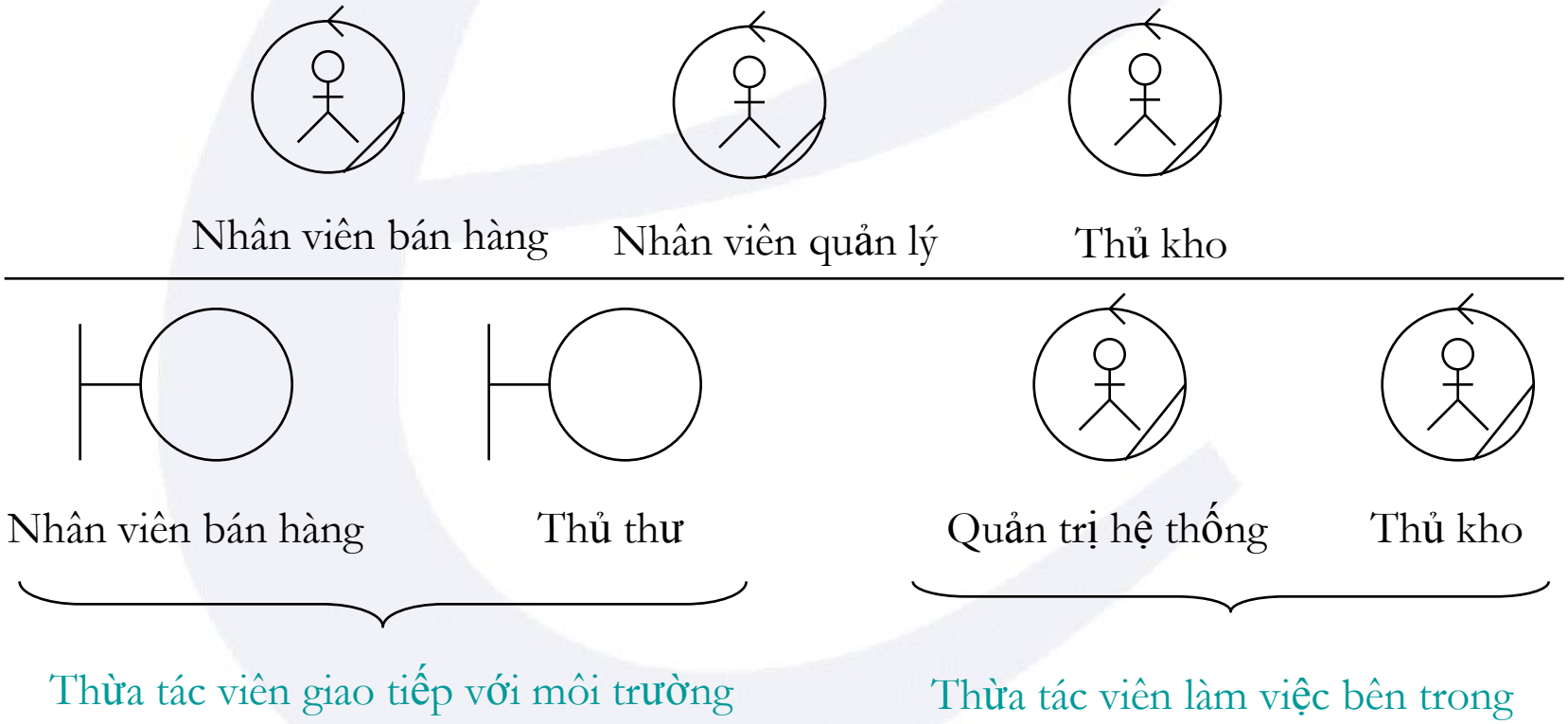


Thừa tác viên

Thiết kế qui trình nghiệp vụ

Xác định thừa tác viên và thực thể nghiệp vụ

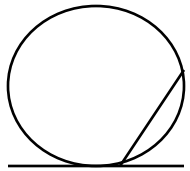
- Xác định thừa tác viên:
  - Ví dụ:



Thiết kế qui trình nghiệp vụ

Xác định thừa tác viên và thực thể nghiệp vụ

- Xác thực thể nghiệp vụ:
  - Một thực thể biểu diễn một sự vật được xử lý hoặc sử dụng bởi các thừa tác viên
  - Ký hiệu

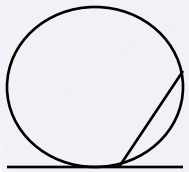


Thực thể nghiệp vụ

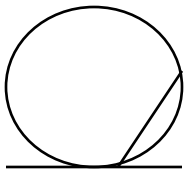
Thiết kế qui trình nghiệp vụ

Xác định thừa tác viên và thực thể nghiệp vụ

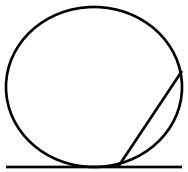
- Xác thực thể nghiệp vụ:
  - Các sự vật có thể là:
    - ✓Đối tượng thông tin: các đối tượng dùng để chứa thông tin dữ liệu hệ thống như là: sổ sách, chứng từ, hồ sơ, giấy tờ, thẻ, báo cáo, tập tin, CSDL,…



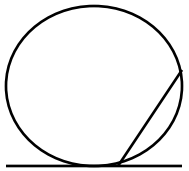
Thẻ thư viện



Hồ sơ khách hàng



Hoá đơn

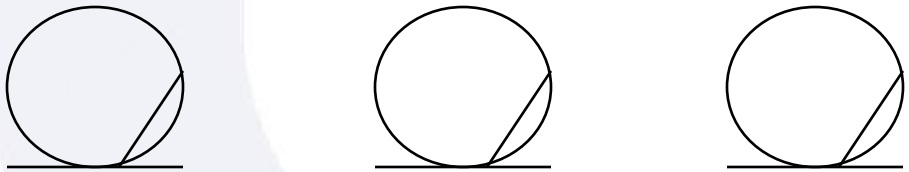


Sổ Nký bán hàng

Thiết kế qui trình nghiệp vụ

Xác định thừa tác viên và thực thể nghiệp vụ

- Xác thực thể nghiệp vụ:
  - Các sự vật có thể là:
    - ✓ Đối tượng sự vật: các đối tượng mô tả các sự vật trong hoạt động nghiệp vụ như là: các đối tượng trong quá trình sản xuất, các trang thiết bị, …

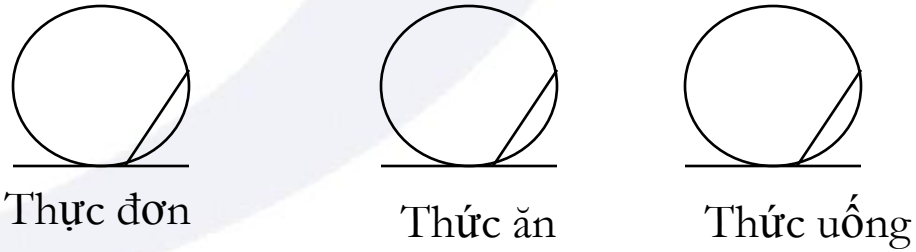


Hàng hoá Nguyên vật liệu Sách

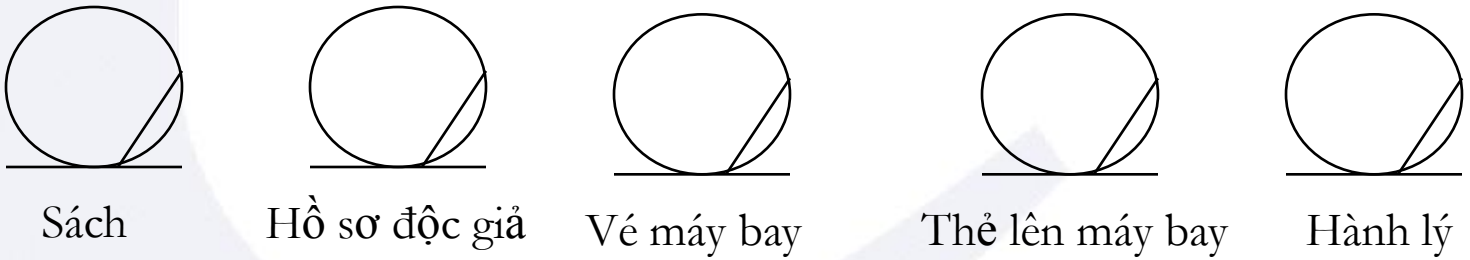
Thiết kế qui trình nghiệp vụ

Xác định thừa tác viên và thực thể nghiệp vụ

- Xác thực thể:
  - Ví dụ:



Tại nhà hàng

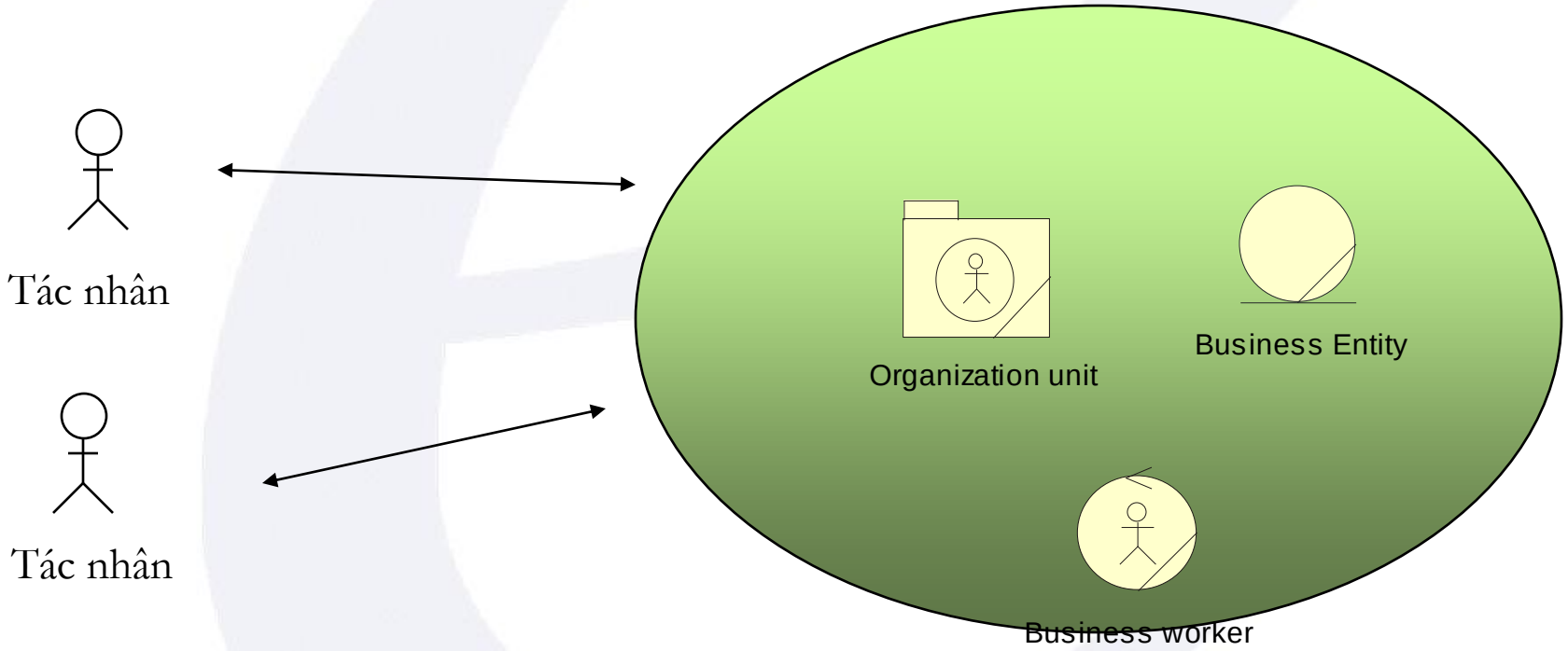


Tại thư viện

Tại sân bay

Thiết kế qui trình nghiệp vụ

Xác định thừa tác viên và thực thể nghiệp vụ



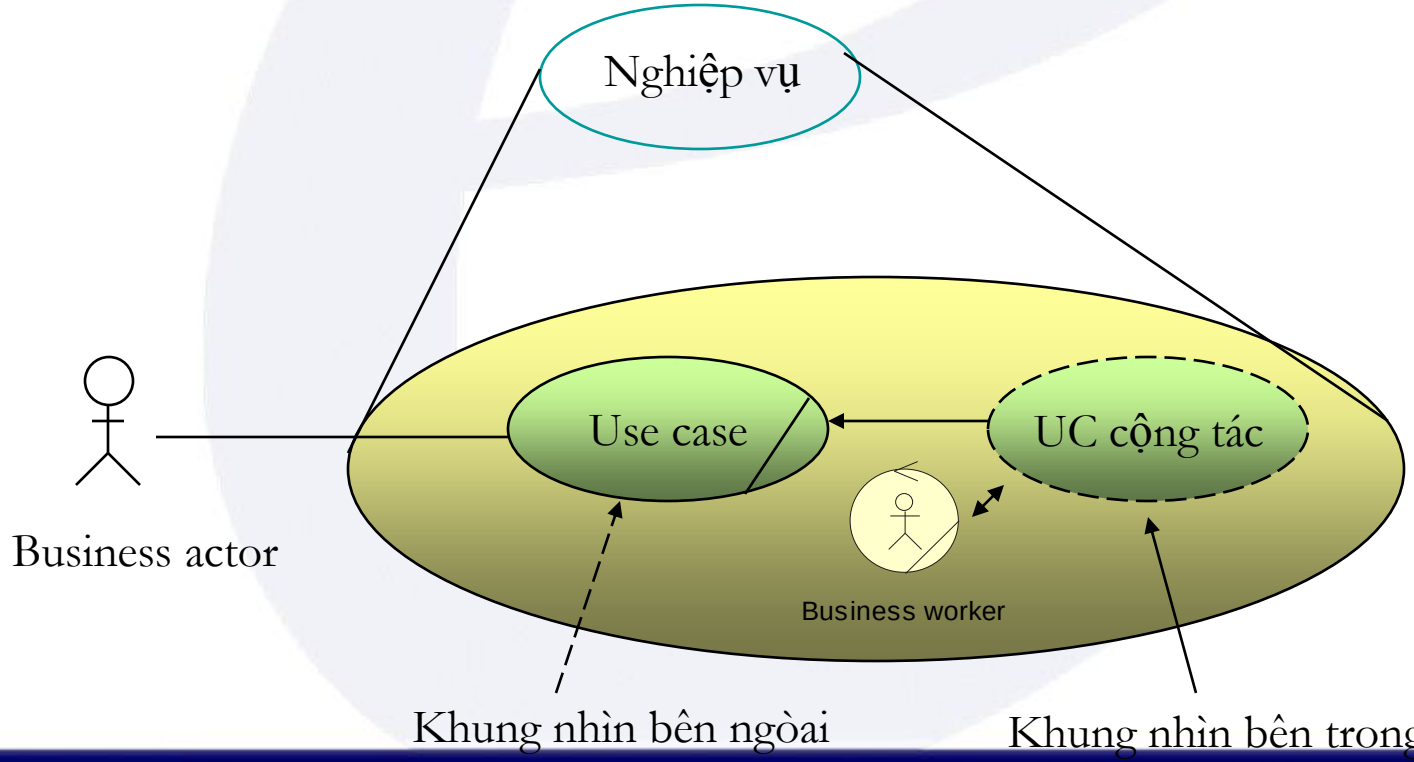
Thiết kế qui trình nghiệp vụ

- Đặc tả use case nghiệp vụ
- Xác định thừa tác viên vụ thực thể nghiệp vụ
- **Hiện thực hóa use case nghiệp vụ**
- Lập cấu trúc ô hình đối tượng nghiệp vụ (business object)
- Đặc tả thừa tác viên nghiệp vụ
- Đặc tả thực thể nghiệp vụ
- Xác định các yêu cầu tự động hóa

Thiết kế qui trình nghiệp vụ

Hiện thực hóa use case nghiệp vụ

- Một hiện thực hóa use-case nghiệp vụ mô tả cách thức một use case cụ thể được hiện thực hóa bên trong mô hình đối tượng dưới dạng: các đối tượng cộng tác với nhau thực hiện hoạt động của use case

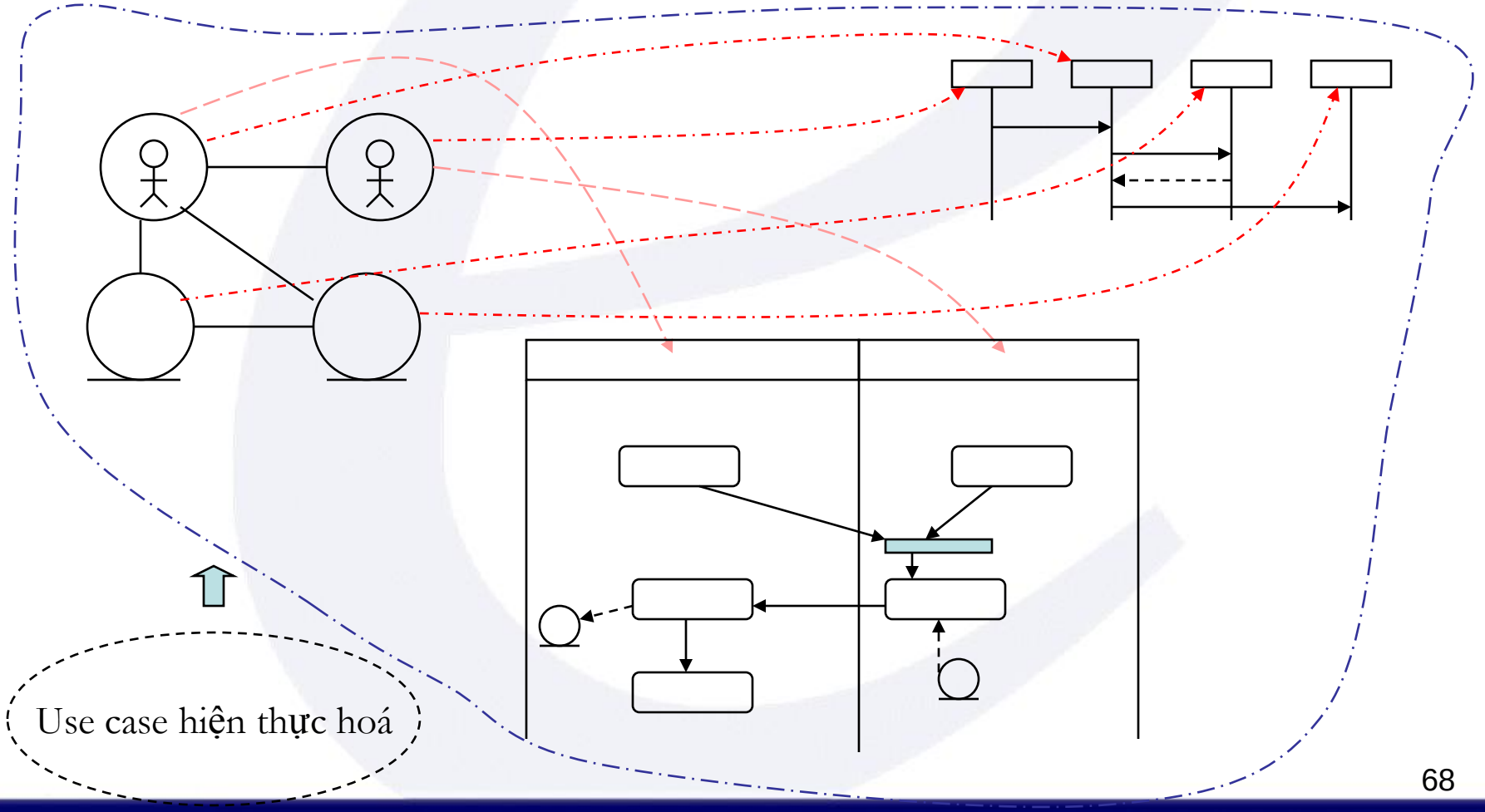


Thiết kế qui trình nghiệp vụ

- Đặc tả use case nghiệp vụ
- Xác định thừa tác viên vụ thực thể nghiệp vụ
- Hiện thực hóa use case nghiệp vụ
- Lập cấu trúc mô hình đối tượng nghiệp vụ (business object)
- Đặc tả thừa tác viên nghiệp vụ
- Đặc tả thực thể nghiệp vụ
- Xác định các yêu cầu tự động hóa

Lập cấu trúc mô hình đối tượng nghiệp vụ

- Các hiện thực hóa use-case nghiệp vụ:



Lập cấu trúc mô hình đối tượng nghiệp vụ

- Mô tả trừu tượng cách thức các thừa tác viên và thực thể liên kết và cộng tác với nhau để thực hiện nghiệp vụ
- Các thành phần chính:
  - Các thừa tác viên: cho thấy các trách nhiệm của một nhân viên
  - Các thực thể: biểu diễn đầu ra, tài nguyên, sự vật được sử dụng
  - Các hiện thực hóa use-case nghiệp vụ

Lập cấu trúc mô hình đối tượng nghiệp vụ

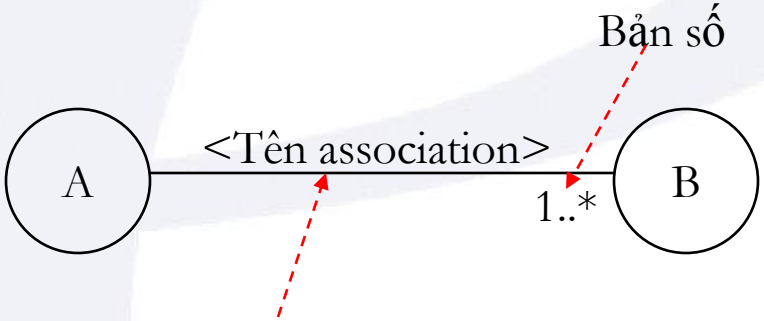
- Mục đích:
  - thống nhất về những gì về lĩnh vực nghiệp vụ được mô tả dưới dạng các đối tượng, thuộc tính, trách nhiệm
  - làm rõ những yêu cầu được hỗ trợ bởi hệ thống thông tin đang xây dựng
  - chuyển tiếp lối tư duy về các vấn đề nghiệp vụ sang lối tư duy về các ứng dụng phần mềm

Lập cấu trúc mô hình đối tượng nghiệp vụ

- Xây dựng lược đồ lớp (class diagram)
  - Các lược đồ lớp cho thấy các mối kết hợp, kết tập và tổng quát hóa giữa **thừa tác viên** và **thực thể**
    - ✓Các hệ thống phân cấp kế thừa
    - ✓Các mối kết tập của thừa tác viên và thực thể.
    - ✓Cách thức các thừa tác viên và thực thể liên quan đến nhau thông qua các mối kết hợp

Lập cấu trúc mô hình đối tượng nghiệp vụ

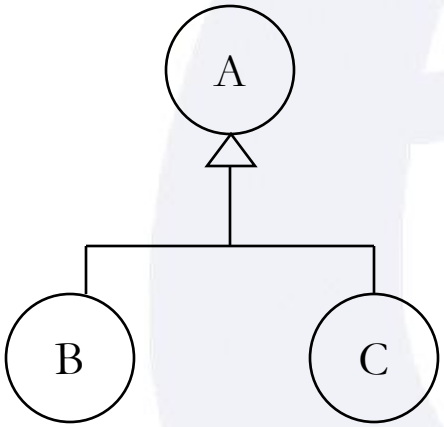
- Xây dựng lược đồ lớp (class diagram)
  - Mối liên kết:
    - ✓ Association

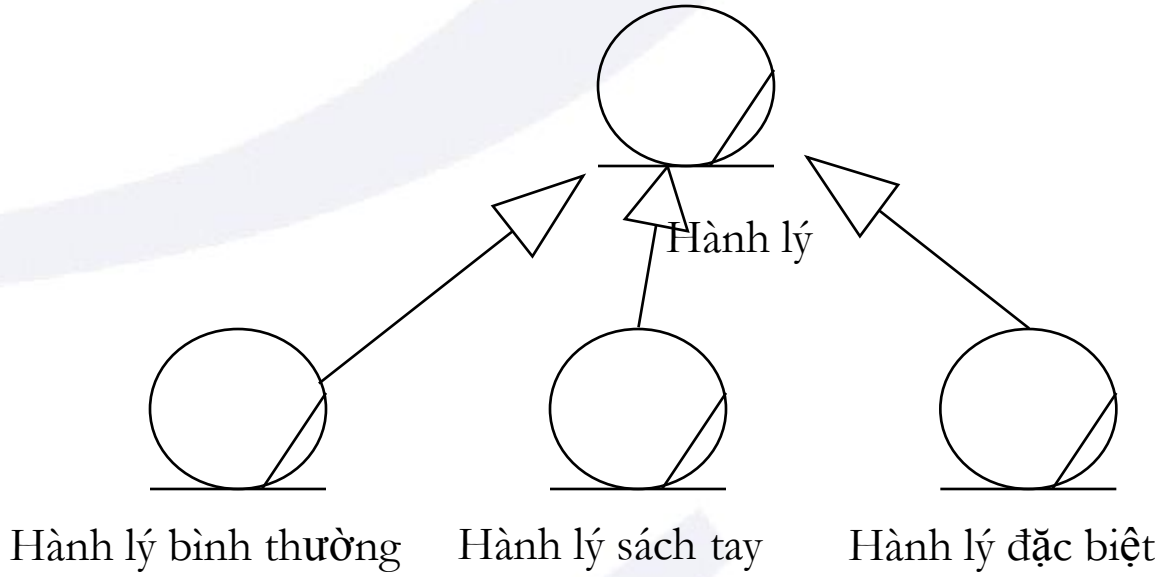


A hoặc đối tượng của A chứa một sự tham chiếu đến B hoặc các đối tượng của B

Lập cấu trúc mô hình đối tượng nghiệp vụ

- Xây dựng lược đồ lớp (class diagram)
  - Mối liên kết:
    - ✓ Specialization:



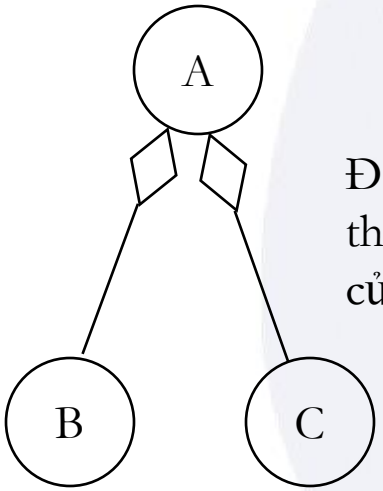


Hành lý chia thành các loại: Hành lý bình thường, hành lý đặc biệt, hành lý đặc biệt

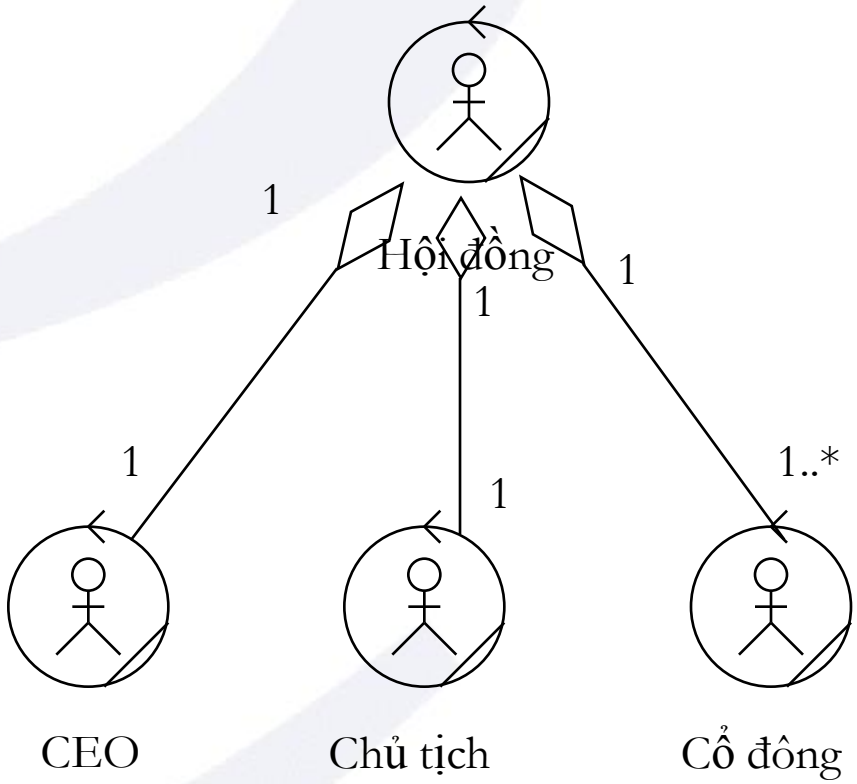
Chỉ sử dụng các mối tổng quát hóa giữa các lớp có cùng stereotype

Lập cấu trúc mô hình đối tượng nghiệp vụ

- Xây dựng lược đồ lớp (class diagram)
  - Mối liên kết:
    - ✓ Aggregration:



Đố tượng A được tạo thành từ những đối tượng của B và C



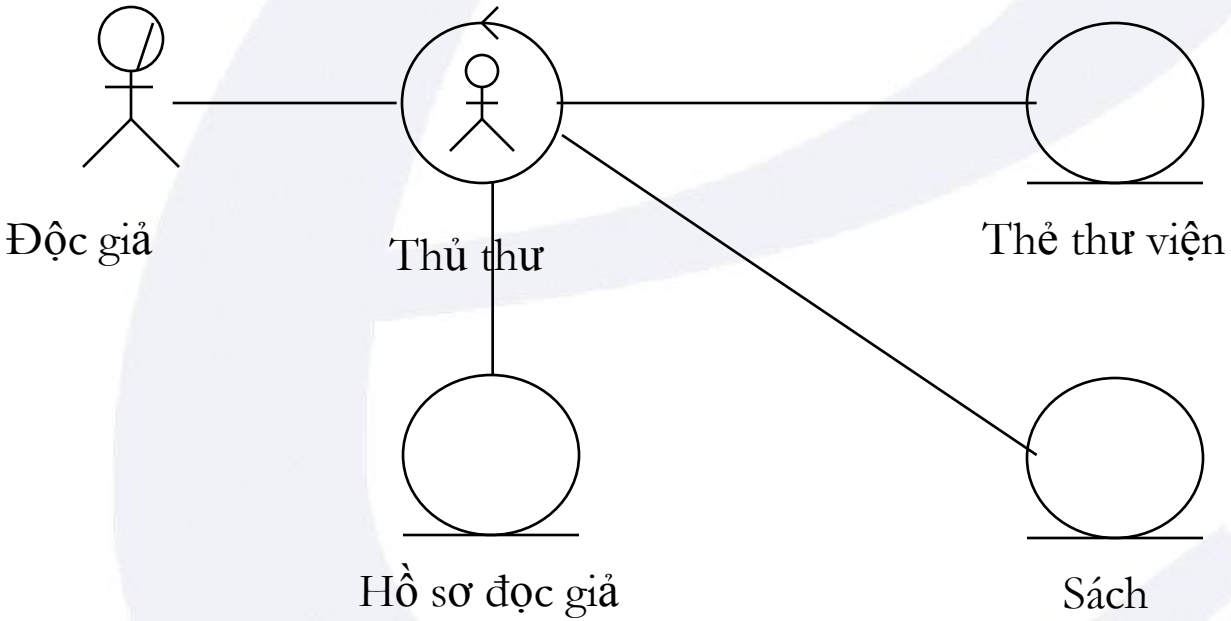
Hội đồng của một công ty bao gồm chủ tịch, giám đốc (CEO), và một vài cổ đông đại diện

Lập cấu trúc mô hình đối tượng nghiệp vụ

- Xây dựng sơ đồ lớp (class diagram)
  - Một sơ đồ lớp các thừa tác viên, các thực thể và gói trong một hoặc nhiều use case
  - Liên kết bao gồm: mối kết hợp, kết tập và tổng quát hóa giữa thừa tác viên và thực thể:
    - ✓ Sự phân cấp kế thừa
    - ✓ Các mối kết tập của thừa tác viên và thực thể
    - ✓ Cách thức các thừa tác viên và thực thể liên quan đến nhau thông qua các mối kết hợp

Lập cấu trúc mô hình đối tượng nghiệp vụ

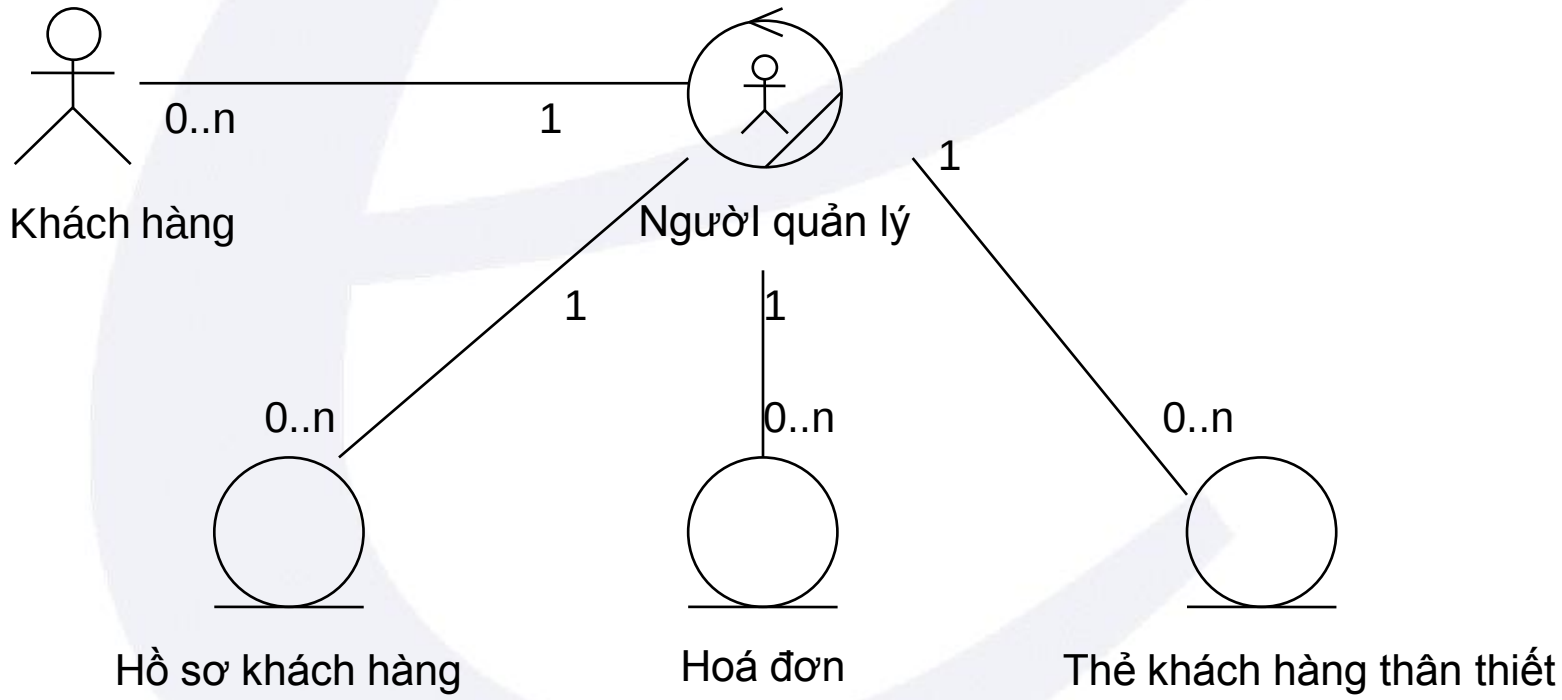
- Xây dựng sơ đồ lớp (class diagram)



Sơ đố lớp cho use case Muợn sách

Lập cấu trúc mô hình đối tượng nghiệp vụ

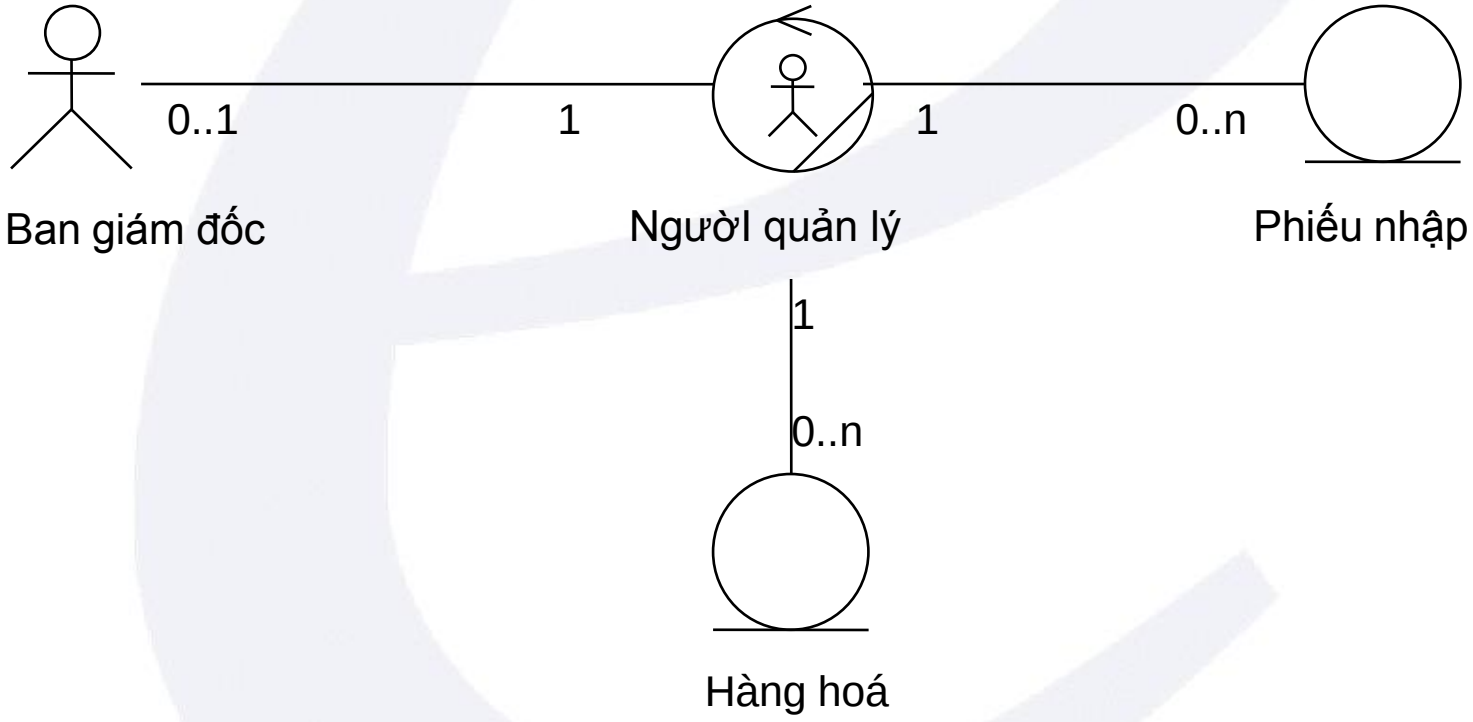
- Xây dựng sơ đồ lớp (class diagram)



Lược đồ lớp cho use case Quản lý khách hàng thân thiết

Lập cấu trúc mô hình đối tượng nghiệp vụ

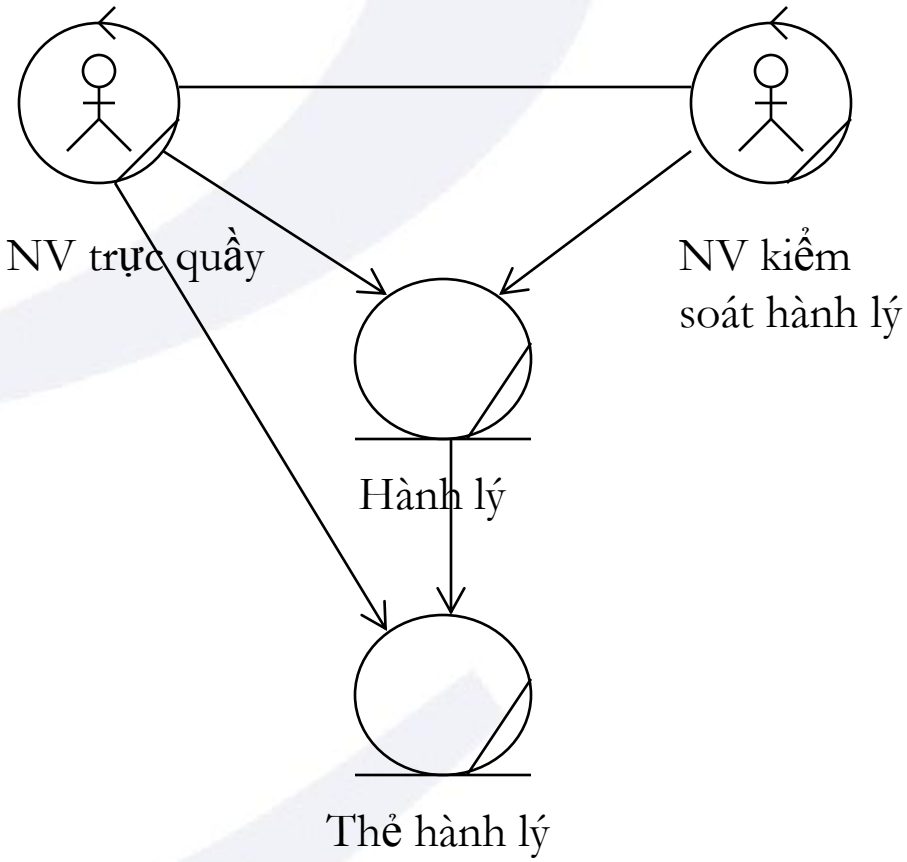
- Xây dựng sơ đồ lớp (class diagram)



Lược đồ lớp cho use case Quản lý nhập hàng

Lập cấu trúc mô hình đối tượng nghiệp vụ

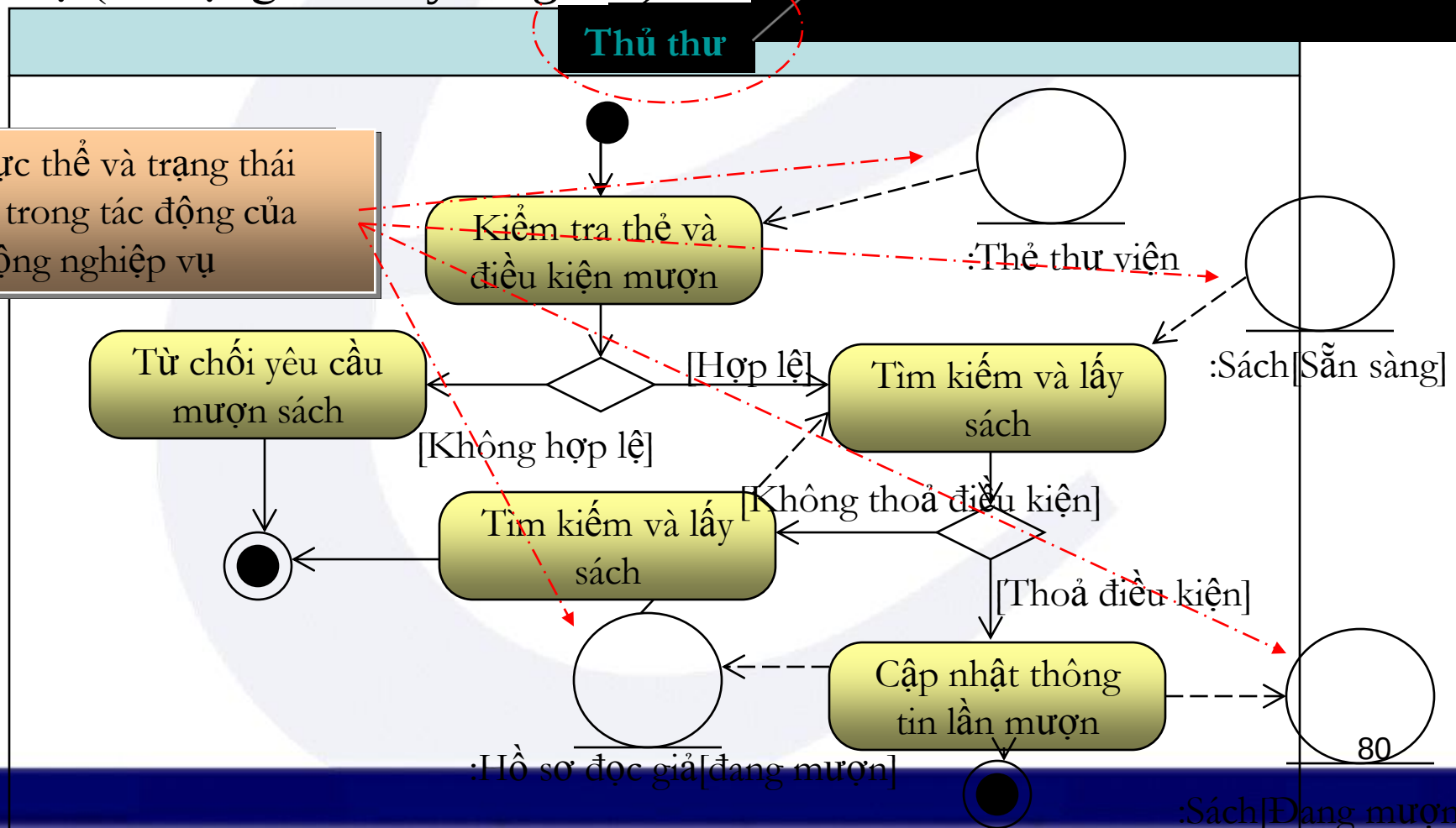
- Xây dựng sơ đồ lớp (class diagram)
  - Ví dụ: Một lược đồ lớp cho thấy các thừa tác viên và thực thể tham gia trong use case Đăng ký Hành khách



Thiết kế qui trình nghiệp vụ

Hiện thực hóa use case nghiệp v

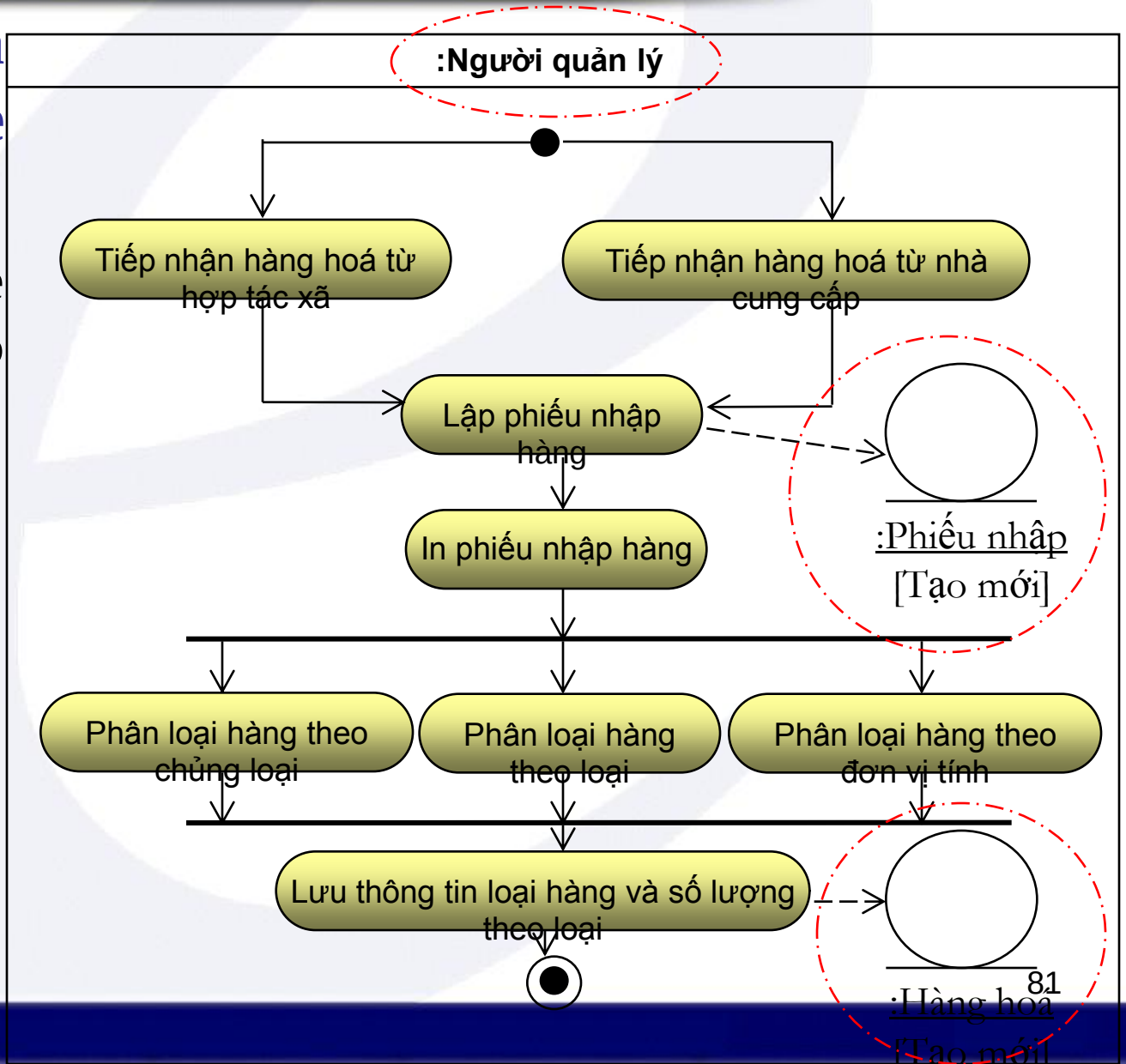
- Đặc tả luồng công việc của hiện
  vụ (sử dụng activity diagram)



Thiết kế qui trình nghiệp vụ

Hiện thực hóa use case nghiệp vụ

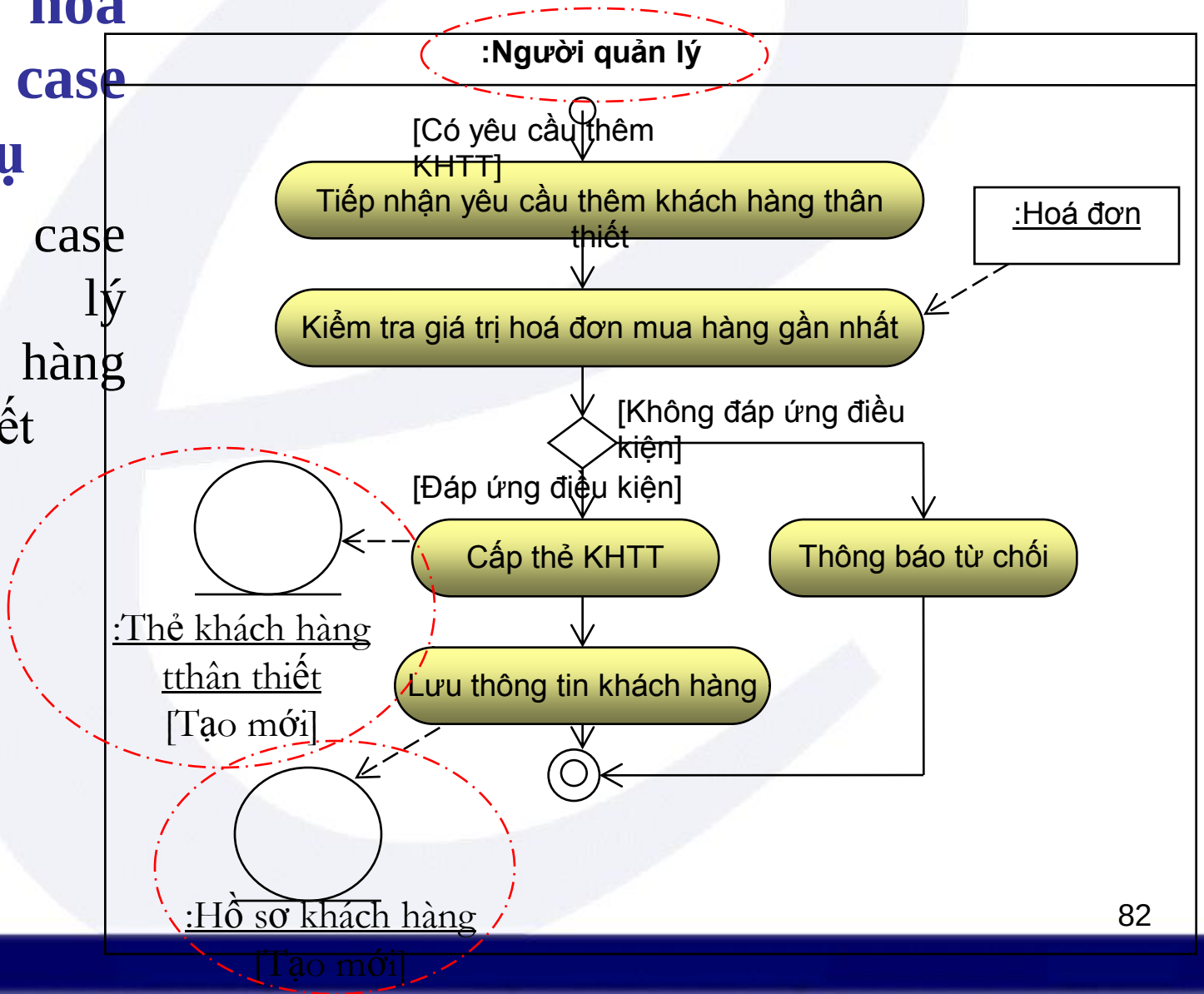
- Use case Quản lý nhập hàng



Thiết kế qui trình nghiệp vụ

Hiện thực hóa use case nghiệp vụ

- Use case Quản lý khách hàng thân thiết



Thiết kế qui trình nghiệp vụ

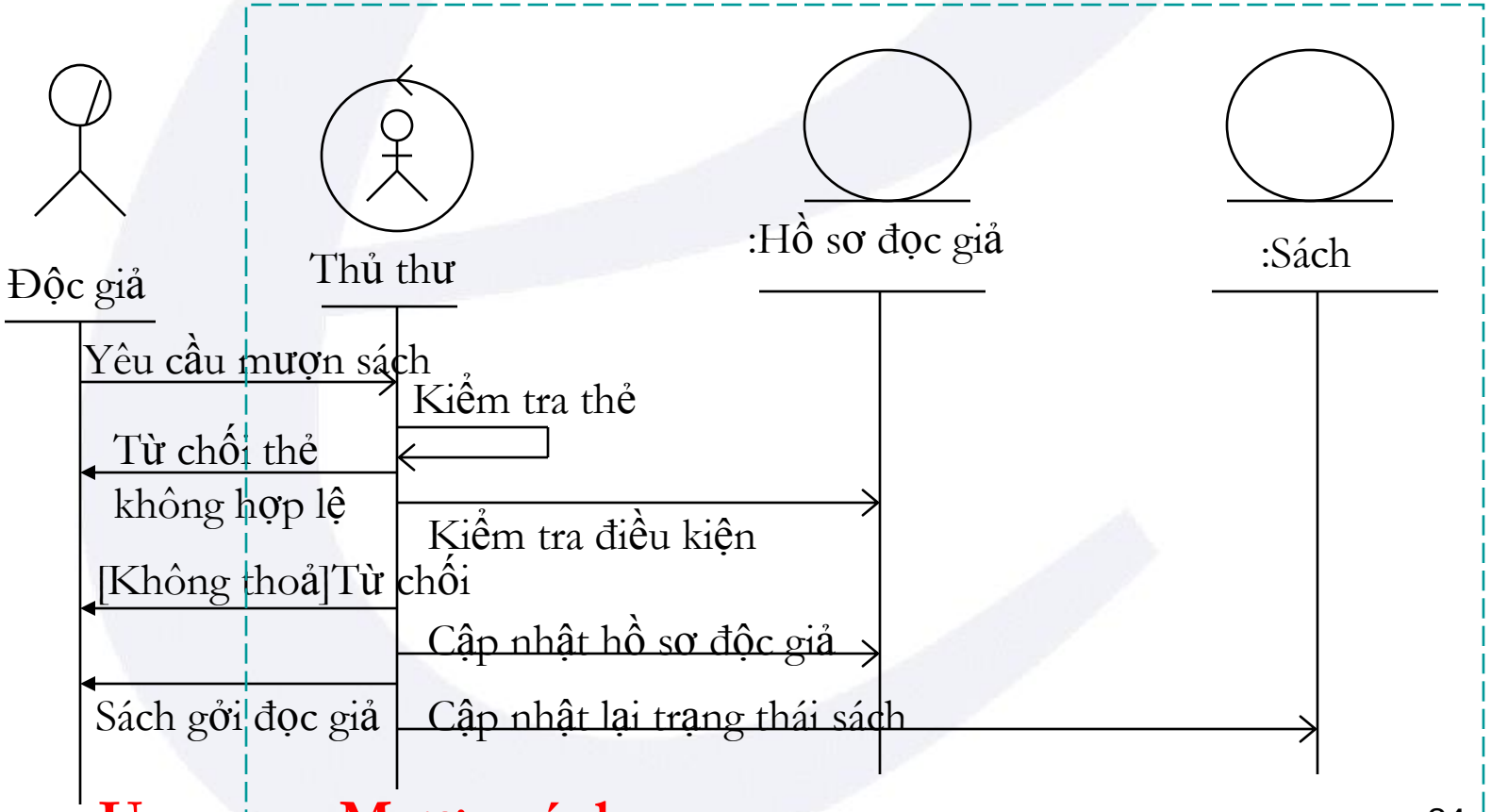
Hiện thực hóa use case nghiệp vụ

- Hiện thực hoá use case nghiệp vụ sử dụng sơ đồ tương tác (interaction diagram)
  - ✓ Lược đồ tuần tự mô tả rõ ràng trình tự các sự kiện. Với các kịch bản phức tạp, thì lược đồ tuần tự thích hợp hơn so với các lược đồ hoạt động.
  - ✓ Lược đồ hợp tác trình bày các mối liên kết giao tiếp và những thông điệp giữa các đối tượng. Chúng phù hợp hơn trong việc giúp ta hiểu được tất cả các tác động trên một đối tượng cho trước.
  - ✓ Nếu có ít luồng rẽ nhánh, nhưng có nhiều thực thể tham gia, thì lược đồ tương tác thường là một sự lựa chọn tốt hơn so với lược đồ hoạt động nhằm để trình bày hiện thực hóa của luồng công việc

Thiết kế qui trình nghiệp vụ

Hiện thực hóa use case nghiệp vụ

- Đặc tả luồng công việc của hiện thực hóa use-case (sử dụng sequence diagram)

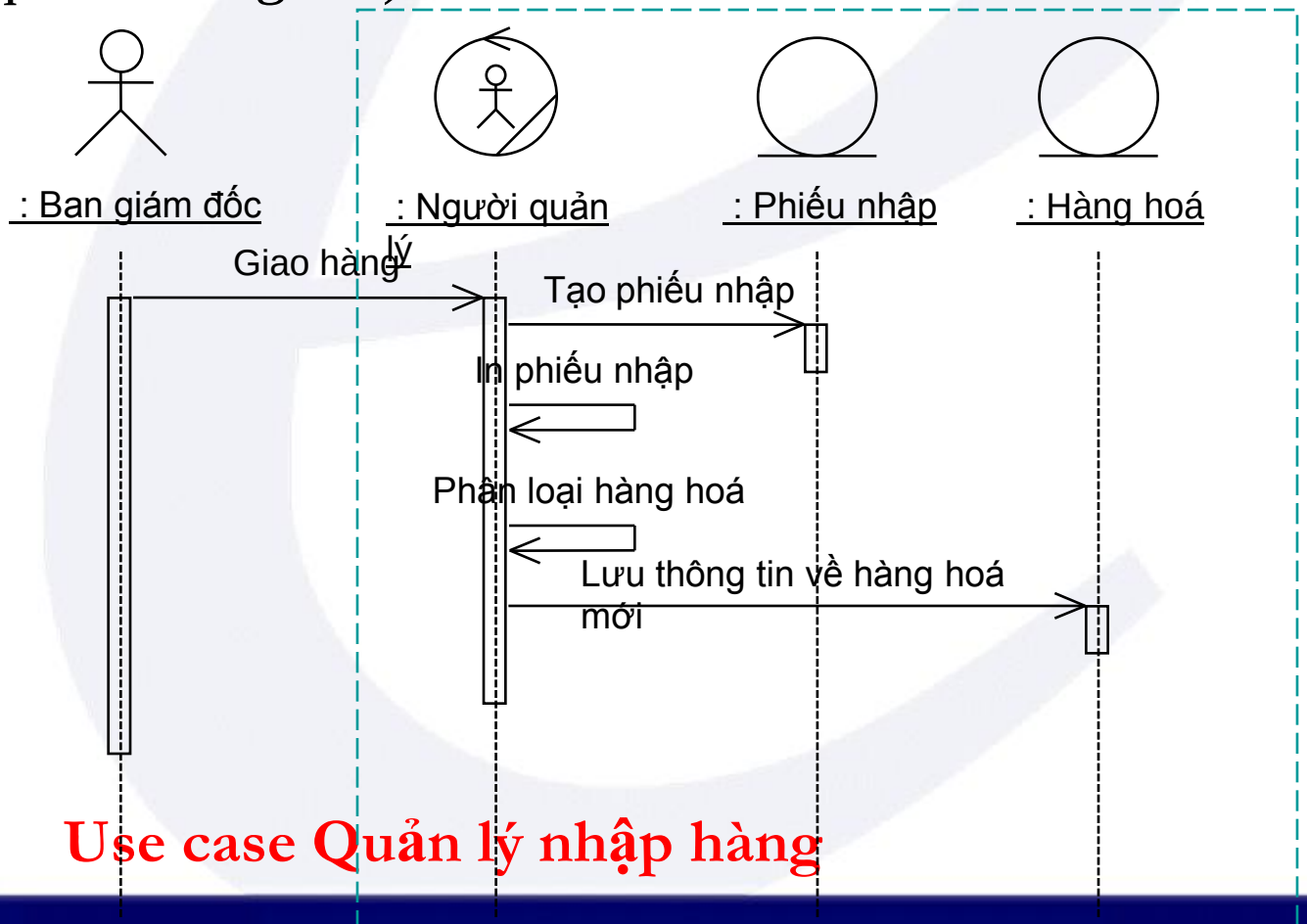


Use case Mượn sách

Thiết kế qui trình nghiệp vụ

Hiện thực hóa use case nghiệp vụ

- Đặc tả luồng công việc của hiện thực hóa use-case (sử dụng sequence diagram)

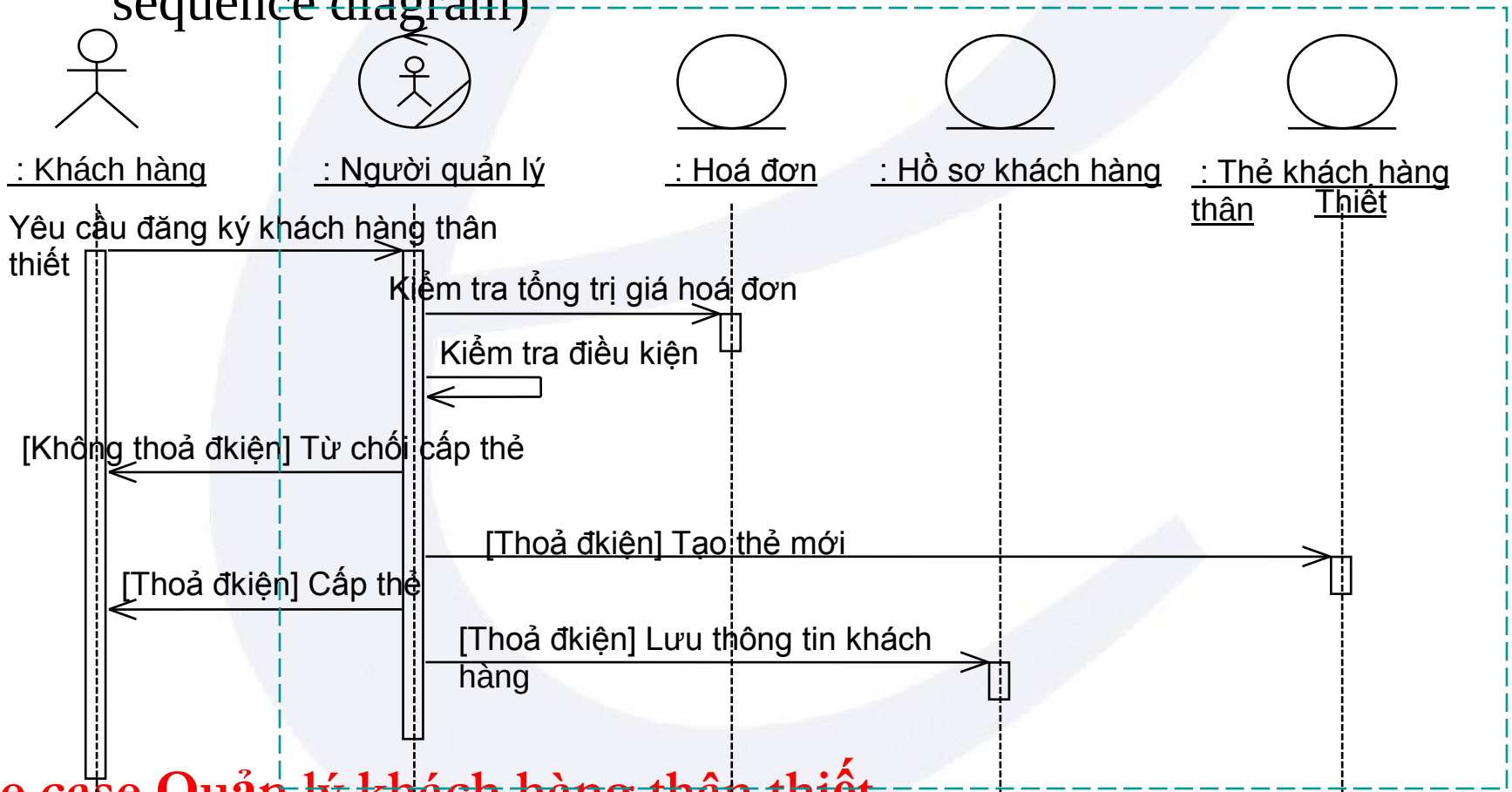


Use case Quản lý nhập hàng

Thiết kế qui trình nghiệp vụ

Hiện thực hóa use case nghiệp vụ

- Đặc tả luồng công việc của hiện thực hóa use-case (sử dụng sequence diagram)



Use case Quản lý khách hàng thân thiết

Thiết kế qui trình nghiệp vụ

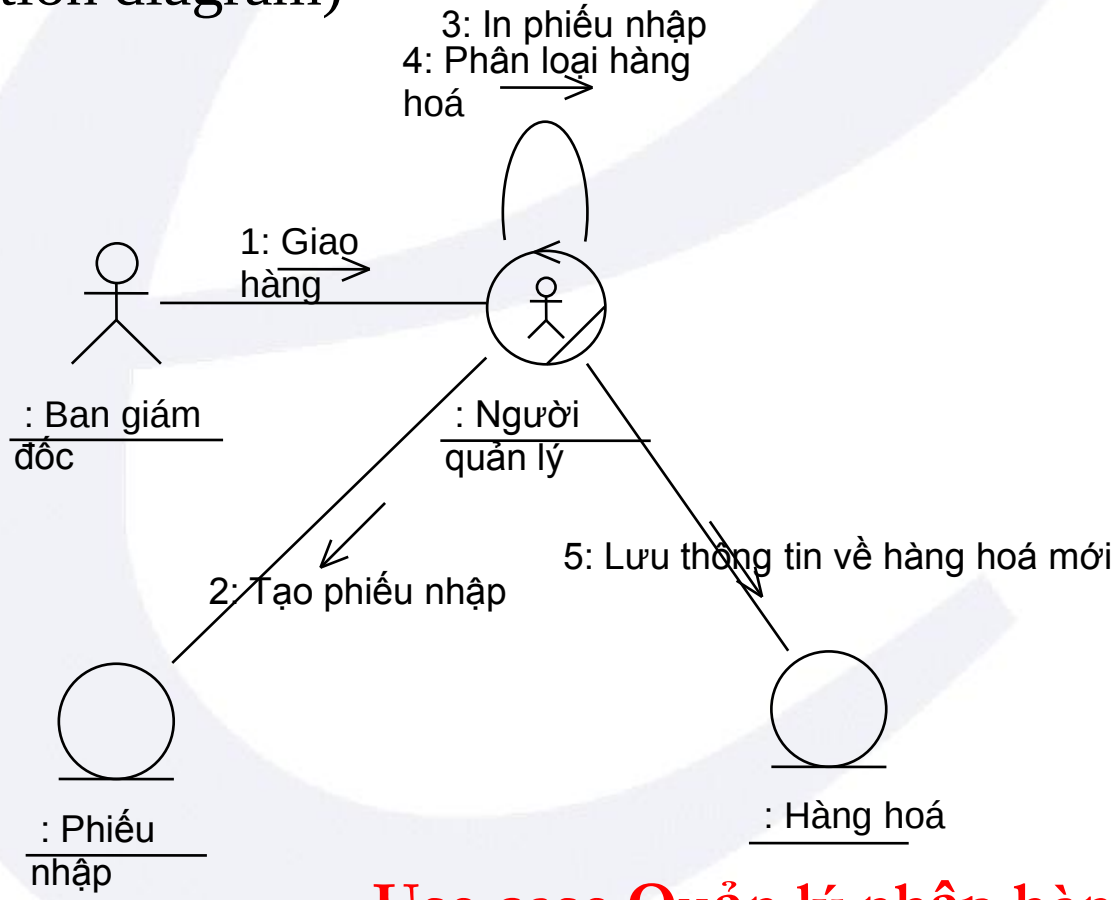
Hiện thực hóa use case nghiệp vụ

- Đặc tả luồng công việc của hiện thực hóa use-case (sử dụng collaboration diagram):
  - ✓ Biểu diễn tương tác giữa các đối tượng trong đó nhấn mạnh sự tham gia của các đối tượng thông qua những mối liên kết giữa chúng và những thông điệp (message) chúng gởi cho nhau.
  - ✓ Tập trung vào các đối tượng trong khi lược đồ tuần tự tập trung vào tương tác và tuần tự các tượng tác

Thiết kế qui trình nghiệp vụ

Hiện thực hóa use case nghiệp vụ

- Đặc tả luồng công việc của hiện thực hóa use-case (sử dụng collaboration diagram)

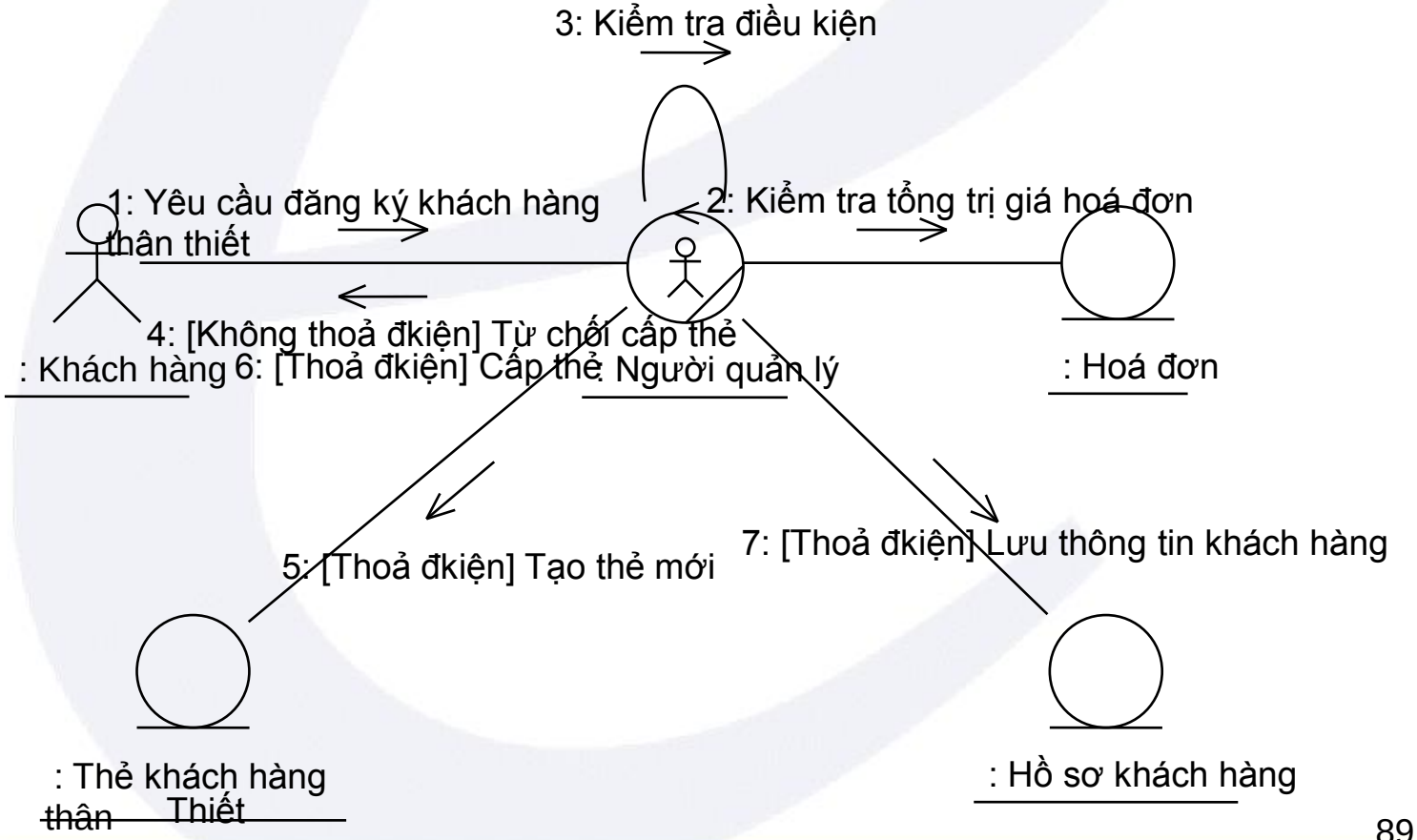


Use case Quản lý nhập hàng

Thiết kế qui trình nghiệp vụ

Hiện thực hóa use case nghiệp vụ

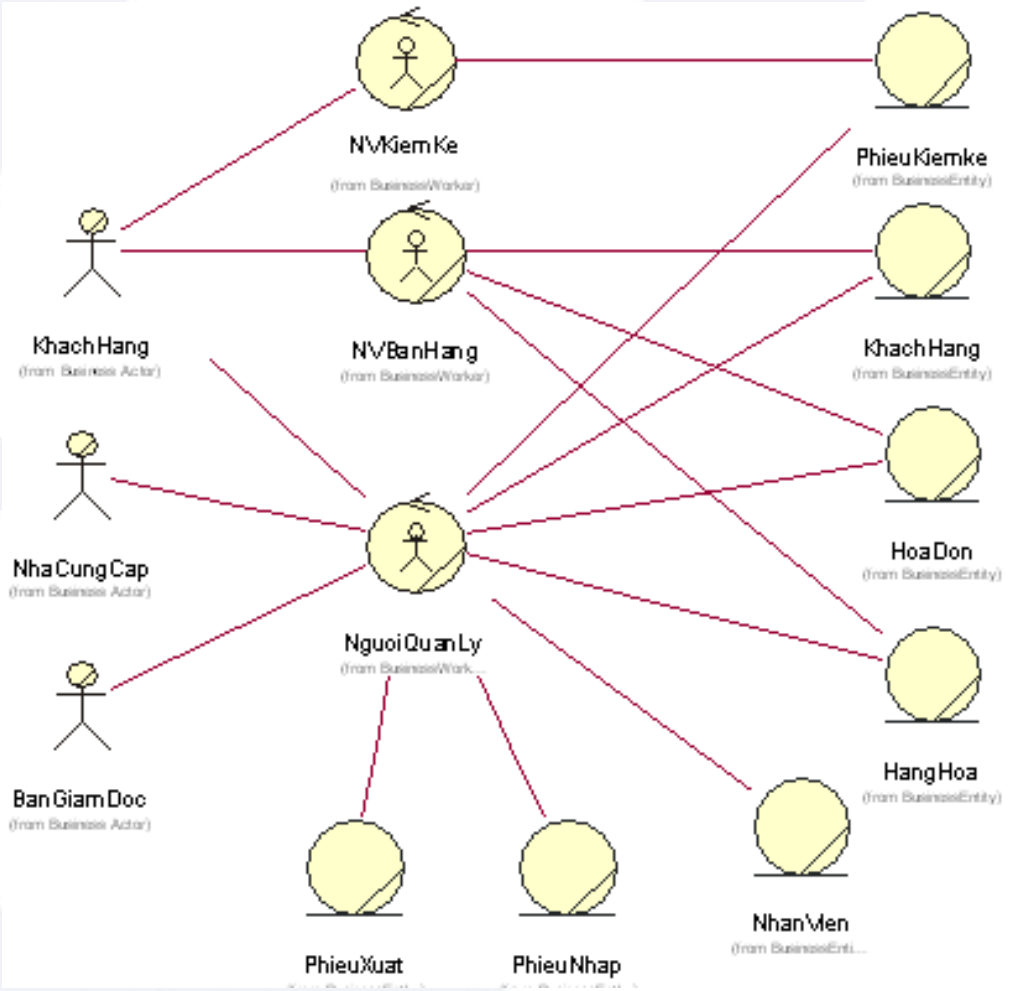
- Đặc tả luồng công việc của hiện thực hóa use-case (sử dụng collaboration diagram)



Use case Quản lý khách hàng thân thiết

Lập cấu trúc mô hình đối tượng nghiệp vụ

- Xây dựng lược đồ lớp (class diagram)
  - Ví dụ:



Thiết kế qui trình nghiệp vụ

- Đặc tả use case nghiệp vụ
- Xác định worker vụ entity nghiệp vụ
- Hiện thực hóa use case nghiệp vụ
- Lập cấu trúc ô hình đối tượng nghiệp vụ (business object)
- Đặc tả worker nghiệp vụ
- Đặc tả entity nghiệp vụ
- Xác định các yêu cầu tự động hóa

Thiết kế qui trình nghiệp vụ

- Đặc tả use case nghiệp vụ
- Xác định worker vụ entity nghiệp vụ
- Hiện thực hóa use case nghiệp vụ
- Lập cấu trúc ô hình đối tượng nghiệp vụ (business object)
- Đặc tả worker nghiệp vụ
- Đặc tả entity nghiệp vụ
- Xác định các yêu cầu tự động hóa

Thiết kế qui trình nghiệp vụ

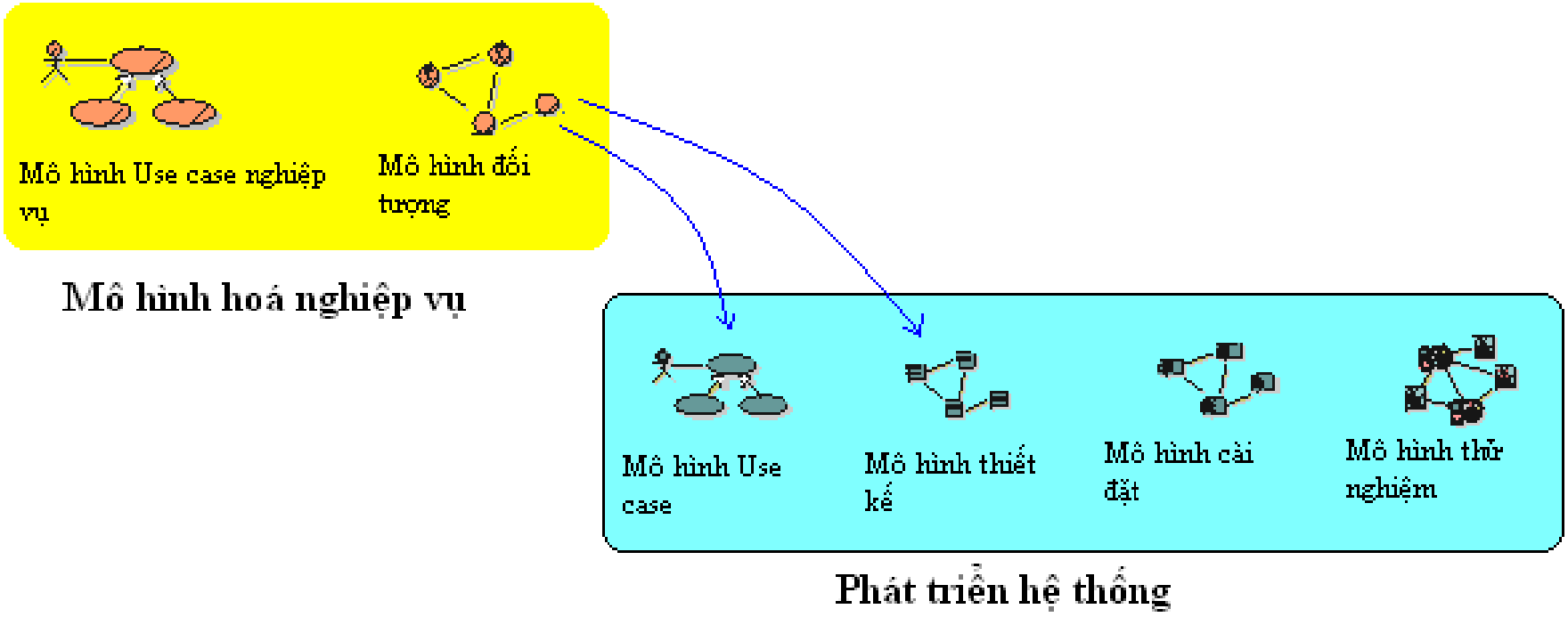
- Đặc tả use case nghiệp vụ
- Xác định worker vụ entity nghiệp vụ
- Hiện thực hóa use case nghiệp vụ
- Lập cấu trúc ô hình đối tượng nghiệp vụ (business object)
- Đặc tả worker nghiệp vụ
- Đặc tả entity nghiệp vụ
- Xác định các yêu cầu tự động hóa

Xác định các yêu cầu tự động hóa

- Mục đích
  - Hiểu được cách thức sử dụng các công nghệ mới cải thiện hoạt động hiệu quả của tổ chức.
  - Xác định mức độ tự động hóa trong tổ chức.
  - Thiết lập các yêu cầu hệ thống từ những artifact mô hình hóa nghiệp vụ.

Xác định các yêu cầu tự động hóa

- Xác định tác nhân và use case hệ thống

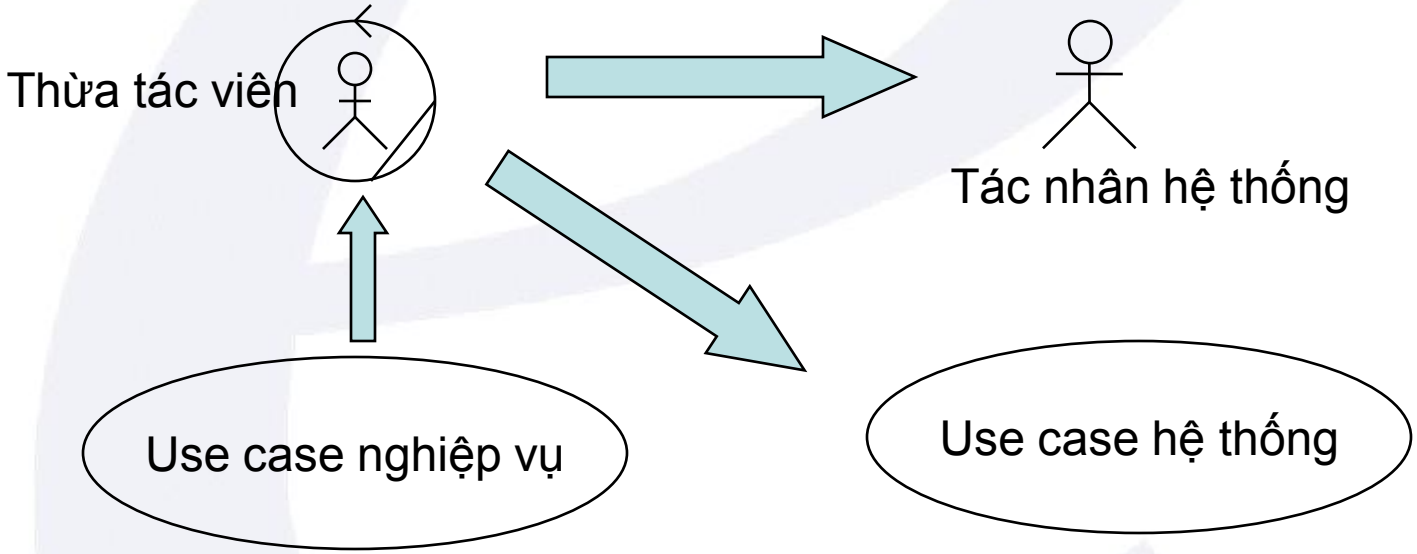


Xác định các yêu cầu tự động hóa

- Xác định actor và use case hệ thống
  - Xác định xem thừa tác viên sẽ sử dụng hệ thống thông tin không?
  - Nếu có, xác định một tác nhân cho hệ thống thông tin trong mô hình use-case của hệ thống thông tin. Đặt tên tác nhân với tên của thừa tác viên.
  - Đối với mỗi use case nghiệp vụ mà thừa tác viên tham gia, tạo một use case hệ thống và mô tả vắn tắt.
  - Xem xét các mục tiêu về tốc độ thực thi hay những thông tin bổ sung cho thừa tác viên cần được chú thích như là một yêu cầu đặc biệt của use case hệ thống.

Xác định các yêu cầu tự động hóa

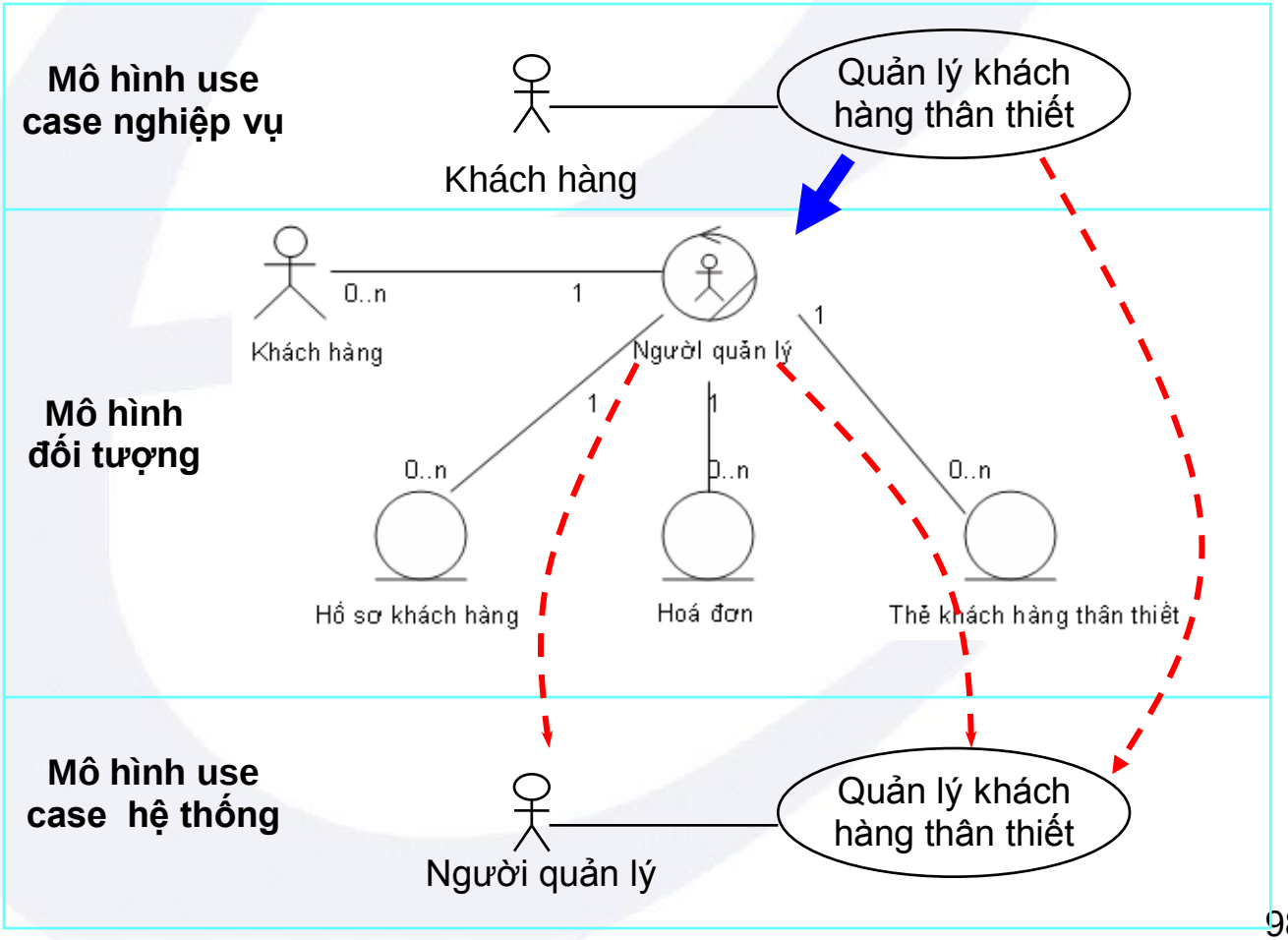
- Xác định actor và use case hệ thống



Đối với mỗi thừa tác viên, xác định một tác nhân hệ thống ứng viên. Đối với mỗi use case nghiệp vụ mà tác nhân tham gia vào, tạo ra một use case hệ thống ứng cử viên

Xác định các yêu cầu tự động hóa

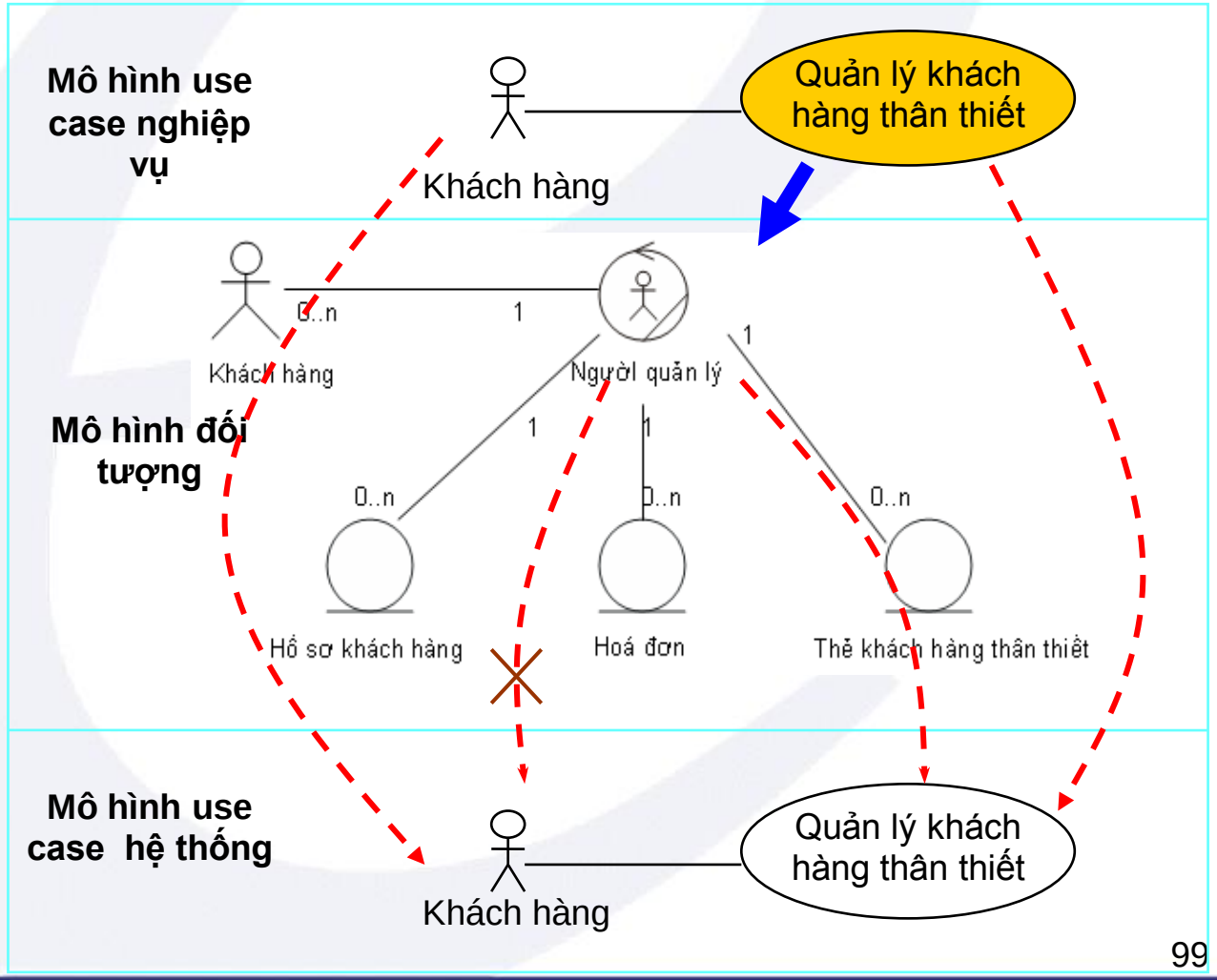
- Xác định tác nhân và use case hệ thống
  - Ví dụ:



Xác định các yêu cầu tự động hóa

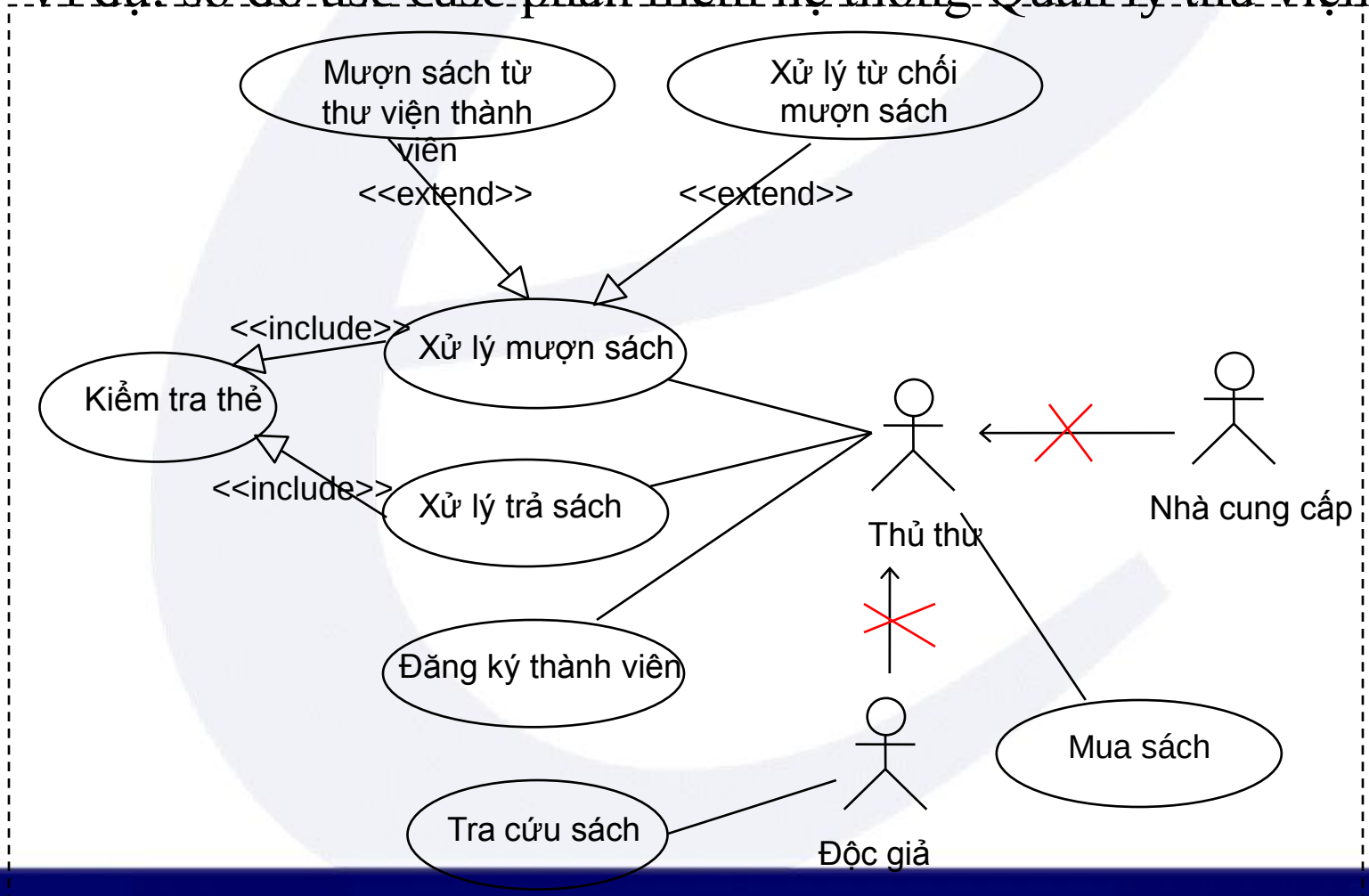
- Xác định tác nhân và use case hệ thống
  - Ví dụ:

Trường hợp tác nhân nghiệp vụ là tác nhân hệ thống: nghiệp vụ được tự động hóa hòan tòan (các ứng dụng thượng mại điện tử)



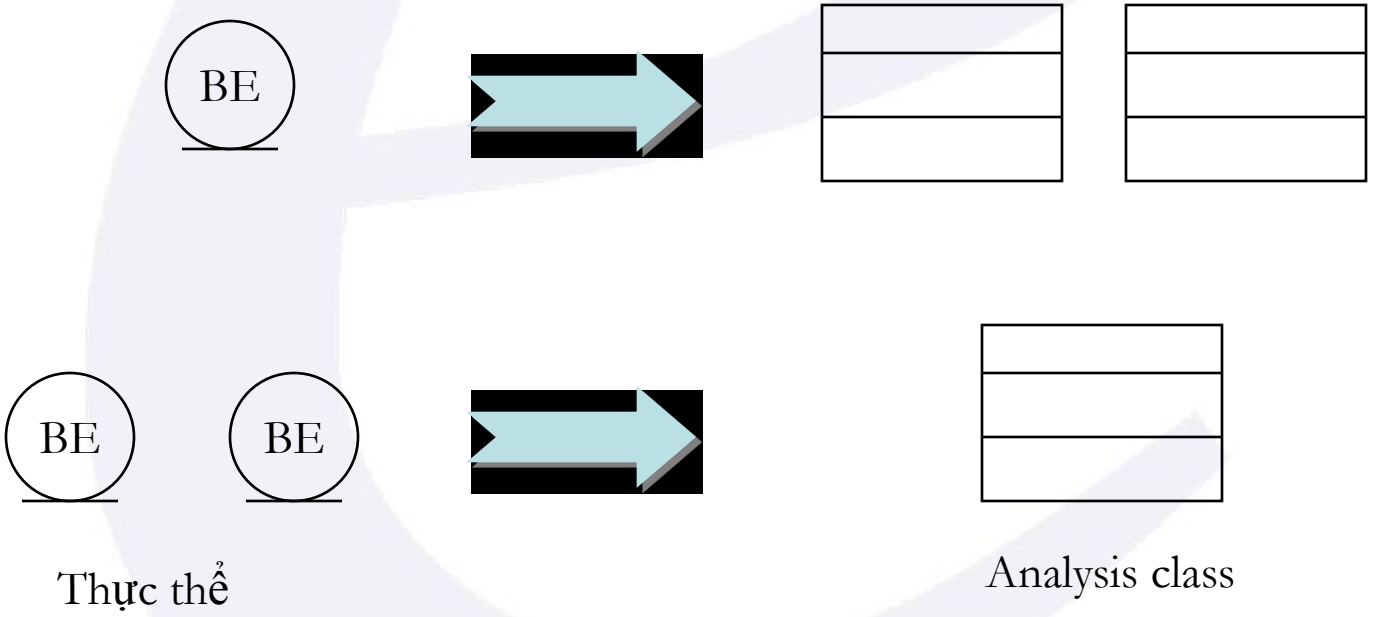
Xác định các yêu cầu tự động hóa

- Xác định tác nhân và use case hệ thống
  - Ví dụ: sơ đồ use case phần mềm hệ thống Quản lý thư viện



Xác định các yêu cầu tự động hóa

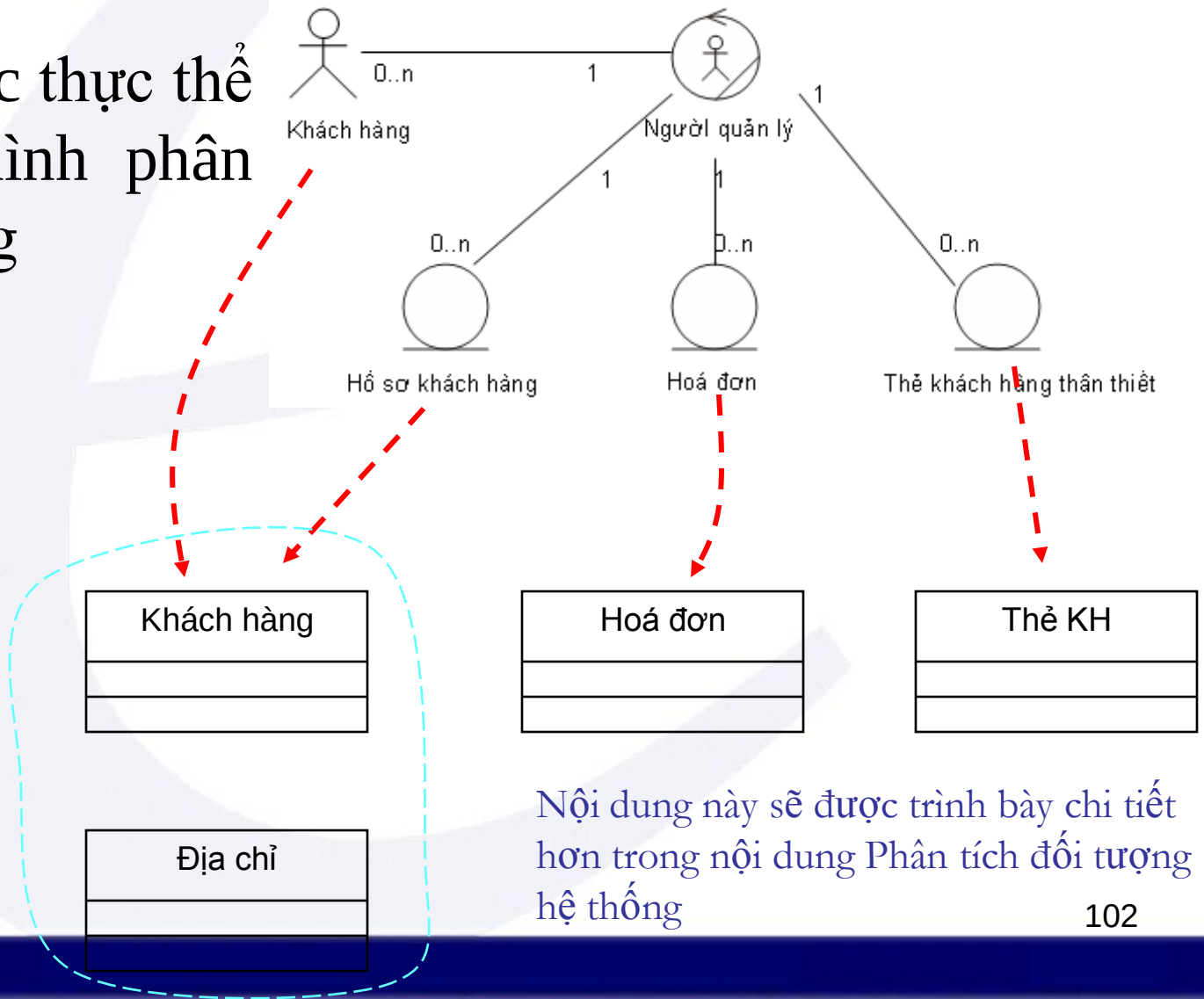
- Xác định các thực thể trong mô hình phân tích hệ thống



Mỗi thực thể được quản lý bởi HTTT sẽ được chuyển thành class trong mô hình phân tích

Xác định các yêu cầu tự động hóa

- Xác định các thực thể trong mô hình phân tích hệ thống
  - Ví dụ:



Nội dung này sẽ được trình bày chi tiết hơn trong nội dung Phân tích đối tượng hệ thống

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN – KHOA HTTT

PHÂN TÍCH THIẾT KẾ HƯỚNG ĐỐI TƯỢNG VỚI UML

Giảng viên: ThS. Nguyễn Đình Loan Phương
Email: phuongndl@uit.edu.vn

Slides: ĐHKHTN, ĐHBK, ĐH Hoa Sen, ĐH Huế

XÁC ĐỊNH YÊU CẦU NGƯỜI DÙNG

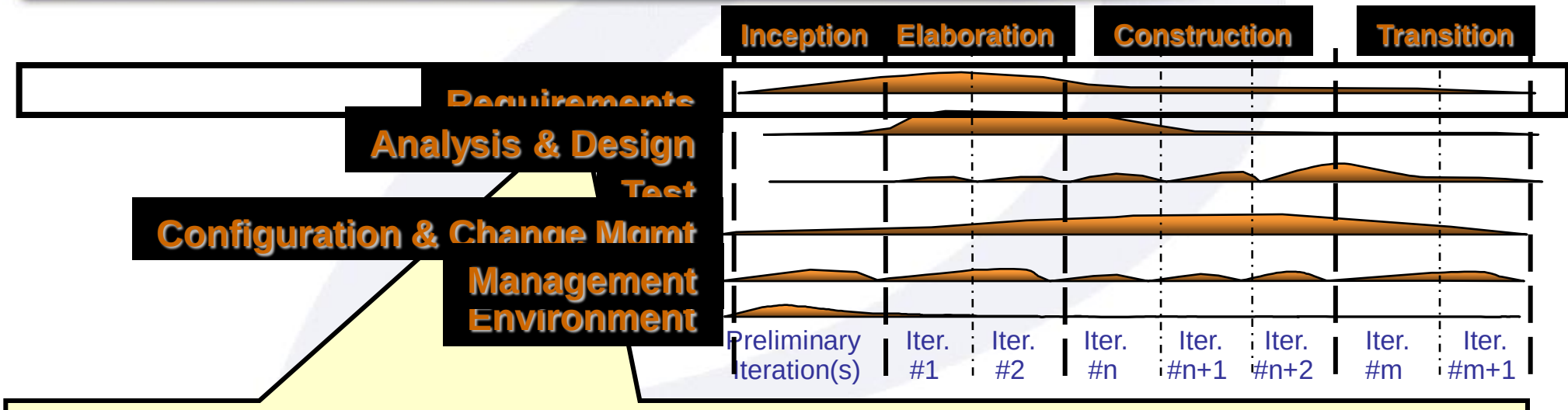
Mục tiêu

- Tìm hiểu các khái niệm cơ bản về xác định yêu cầu người dùng và tác dụng của chúng lên Phân tích và Thiết kế
- Tìm hiểu cách ghi nhận và diễn dịch các yêu cầu của người dùng, là những thông tin được dùng để bắt đầu việc phân tích và thiết kế

Nội dung

- **Giới thiệu**
- Các khái niệm chính
  - Actor
  - Use Case
  - Use Case Model
- Phát biểu bài toán
- Bảng chú giải
- Các đặc tả bổ sung
- Checkpoints

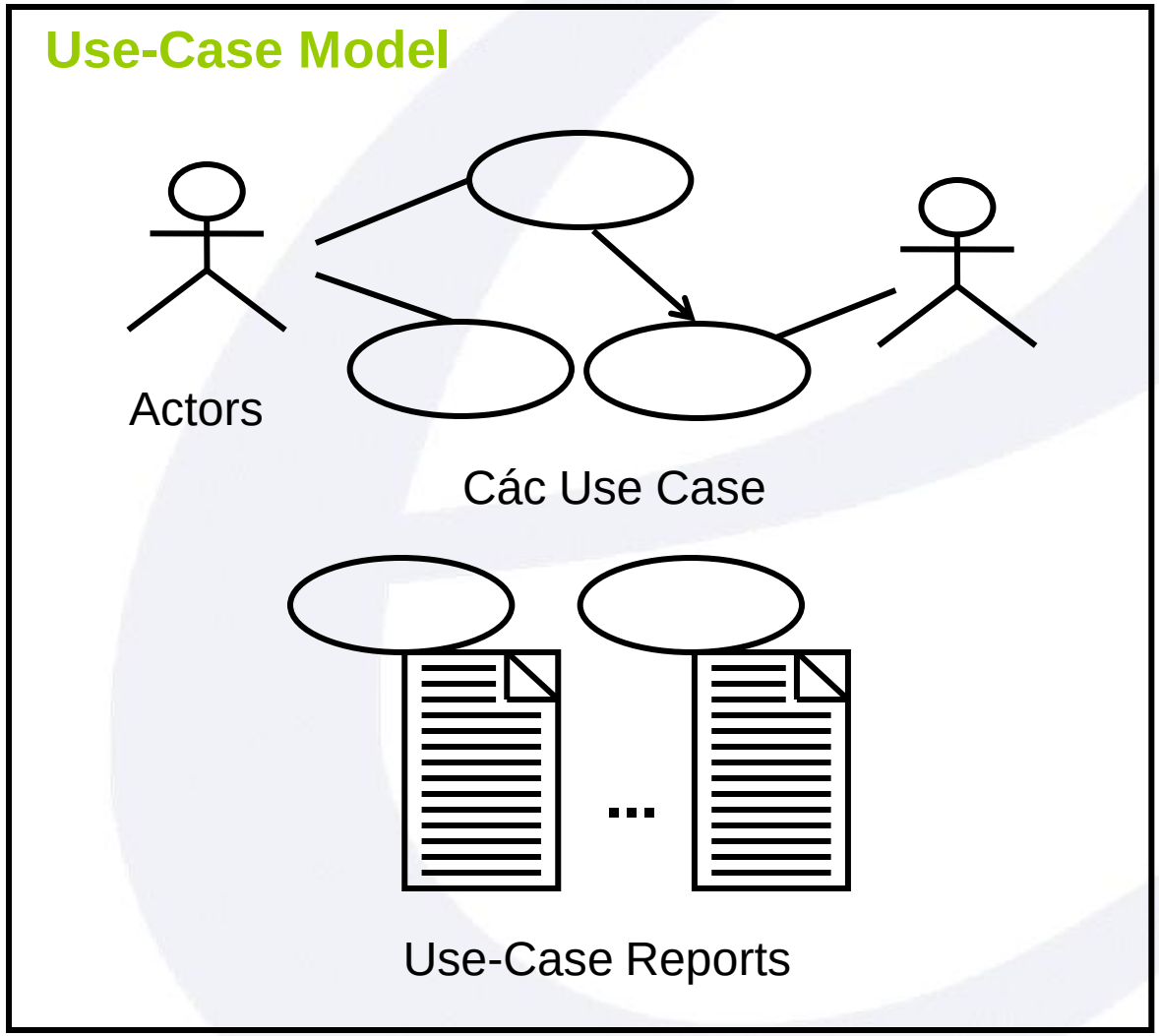
Yêu cầu người dùng trong ngữ cảnh

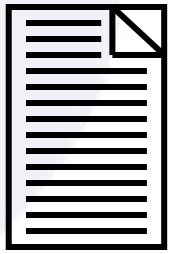


Mục đích của bước xác dịnh yêu cầu người dùng là:

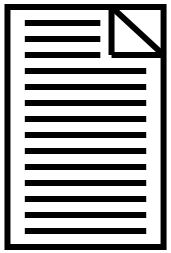
- Đi đến thỏa thuận với khách hàng và người dùng về các chức năng của hệ thống (những gì hệ thống phải thực hiện).
- Cho phép các nhà phát triển hệ thống (system developer) hiểu rõ hơn các yêu cầu đối với hệ thống.
- Phân định các ranh giới của hệ thống.
- Cung cấp cơ sở để hoạch định nội dung kỹ thuật của các vòng lặp.
- Xác định giao diện người dùng cho hệ thống.

Các dạng thông tin về yêu cầu người dùng





Bảng chú giải

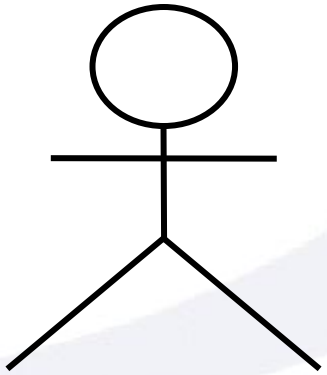


Các đặc tả bổ sung

Nội dung

- Giới thiệu
- **Các khái niệm chính**
  - Actor
  - Use Case
  - Use Case Model
- Phát biểu bài toán
- Bảng chú giải
- Các đặc tả bổ sung
- Checkpoints

Khái niệm - Actor



Actor (Tác nhân)

Các Actor nằm BÊN NGOÀI hệ thống

Actor

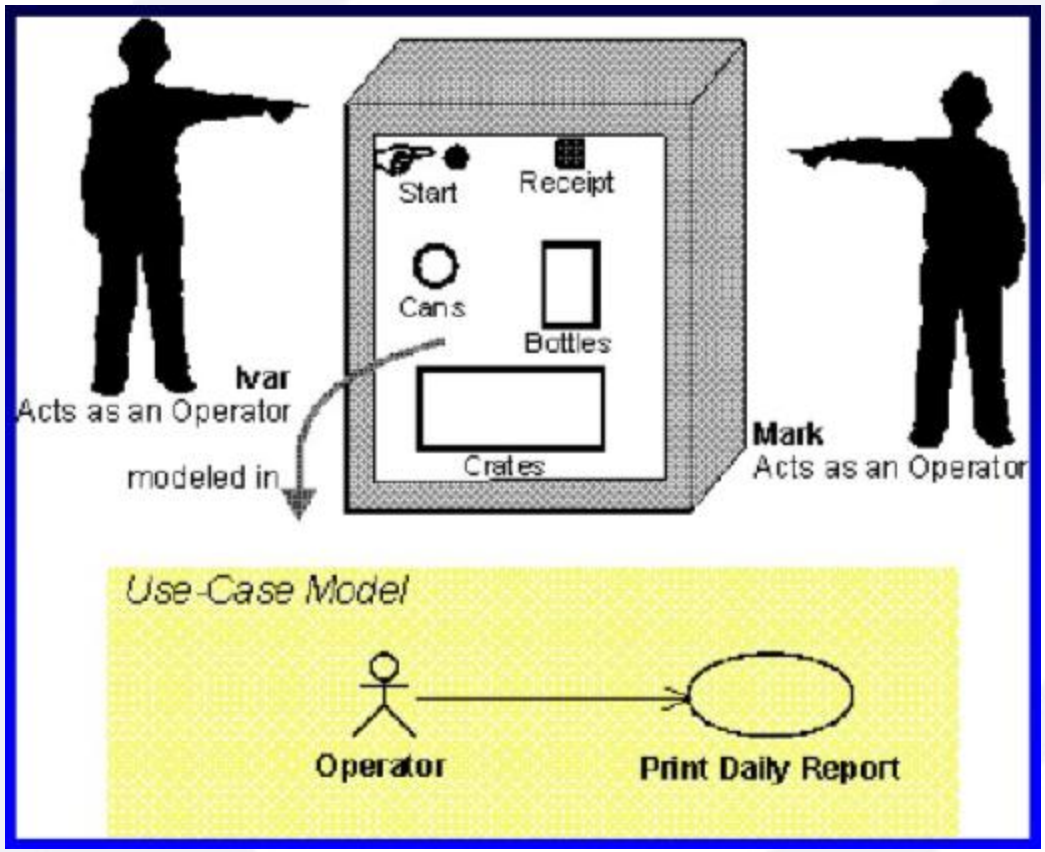
- Một actor xác định một **tập các vai trò** khi người sử dụng **tương tác** với hệ thống. Người sử dụng có thể là một cá nhân hay **một hệ thống khác**
- Để có thể hiểu một cách đầy đủ hệ thống cần xây dựng, bạn cần phải biết hệ thống **phục vụ cho ai**, có nghĩa là ai sẽ là người sử dụng hệ thống. Những loại người dùng khác nhau sẽ được biểu diễn bởi các tác nhân trong mô hình.
- Một tác nhân là một cái gì đó **trao đổi dữ liệu với hệ thống**. Tác nhân có thể là **người sử dụng**, một **thiết bị phần cứng bên ngoài**, hoặc có thể là một **hệ thống khác**.

Phân biệt Actor – Instant Actor ?

- Sự khác biệt giữa một tác nhân và một người sử dụng độc lập trong hệ thống là tác nhân **biểu diễn một lớp (một tập) người sử dụng** chứ không phải là một cá nhân cụ thể nào.
- Một vài người sử dụng có thể đóng cùng một vai trò đối vơí hệ thống => chỉ thiết kế một tác nhân biểu diễn cho các người dùng trên

=> Mỗi người dùng cụ thể là một thể hiện của tác nhân (Instant Actor)

Actor – Ví dụ


Start
Receipt
Cans
Bottles
Crates
Ivar
Acts as an Operator
modeled in
Mark
Acts as an Operator
Use-Case Model
Operator
Print Daily Report

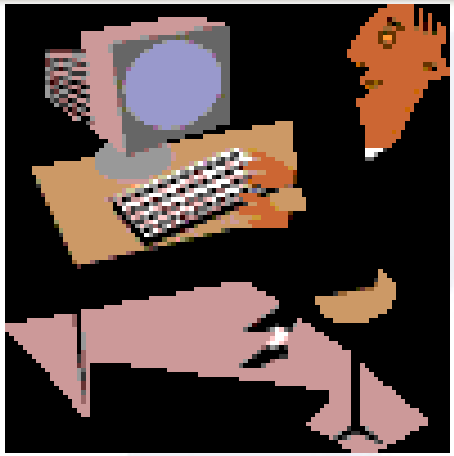
Actor – Ví dụ

- Một hệ thống quản lý thư viện, cho phép người dùng có thể tra cứu thông tin của các quyển sách có trong thư viện.
- Hai sinh viên A và B sử dụng hệ thống để tra cứu thông tin

=> Chỉ có một tác nhân là "Người sử dụng“

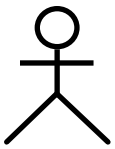
=> A và B là hai thể hiện của tác nhân này.

Vai trò (Role)



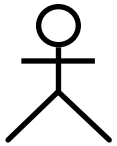
Charlie

Charlie có vai trò như
một sinh viên



Student

Charlie có vai trò như
một giáo sư



Professor

=> Một User có thể có nhiều vai trò

Vai trò (Role)

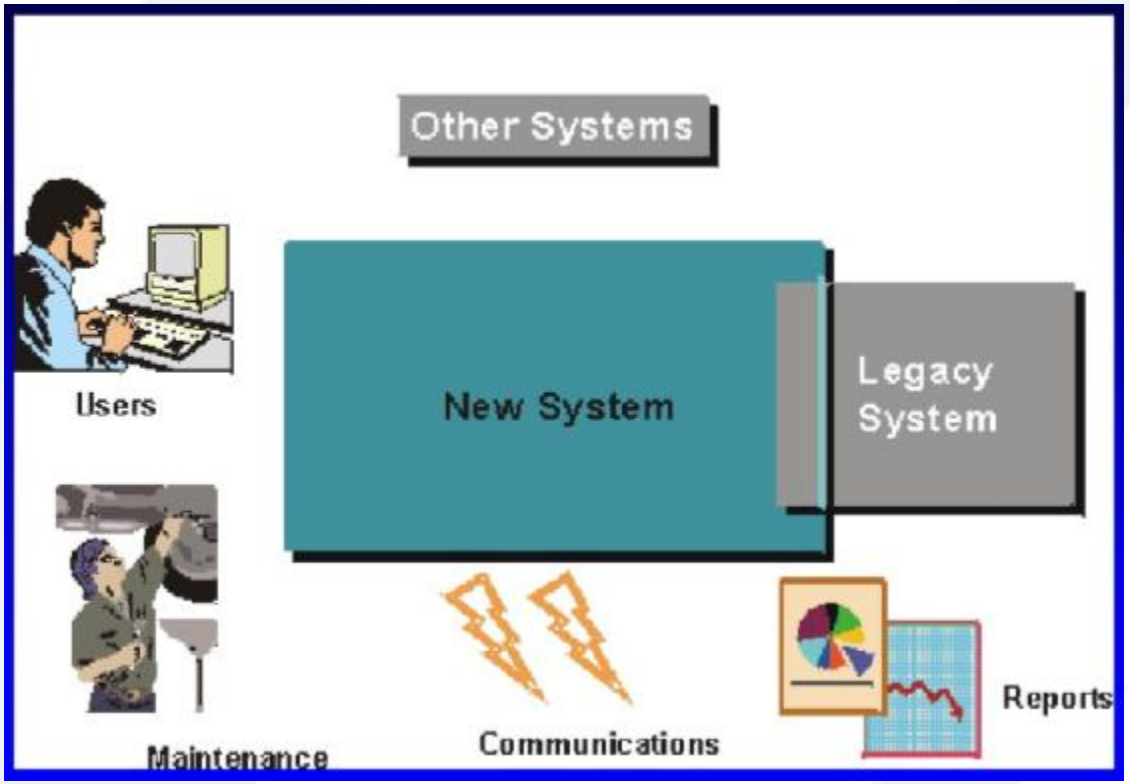
- Trong một vài tình huống, một người đóng một vai trò nào đó được mô hình hóa thành một actor trong hệ thống. Ví dụ như quản trị hệ thống.
- Cũng có trường hợp cùng một người dùng nhưng là thể hiện của nhiều tác nhân (trong trường hợp một cá nhân có nhiều vai trò).
- Ví dụ: người thủ thư tên A có thể có hai vai trò khác nhau trong hệ thống quản lý thư viện
  - Tác nhân **“Người sử dụng"** bình thường
  - Tác nhân **"Người thủ thư"**.

Làm thế nào để xác định Actor?

- Những gì xung quanh hệ thống sẽ trở thành tác nhân của hệ thống ?
- Những cá nhân độc lập sẽ sử dụng hệ thống.
- Phân loại ?

=> nghĩ tới một vài cá nhân nào đó và đảm bảo rằng các tác nhân được thiết kế đáp ứng hầu hết các nhu cầu của họ.

Làm thế nào để xác định Actor?



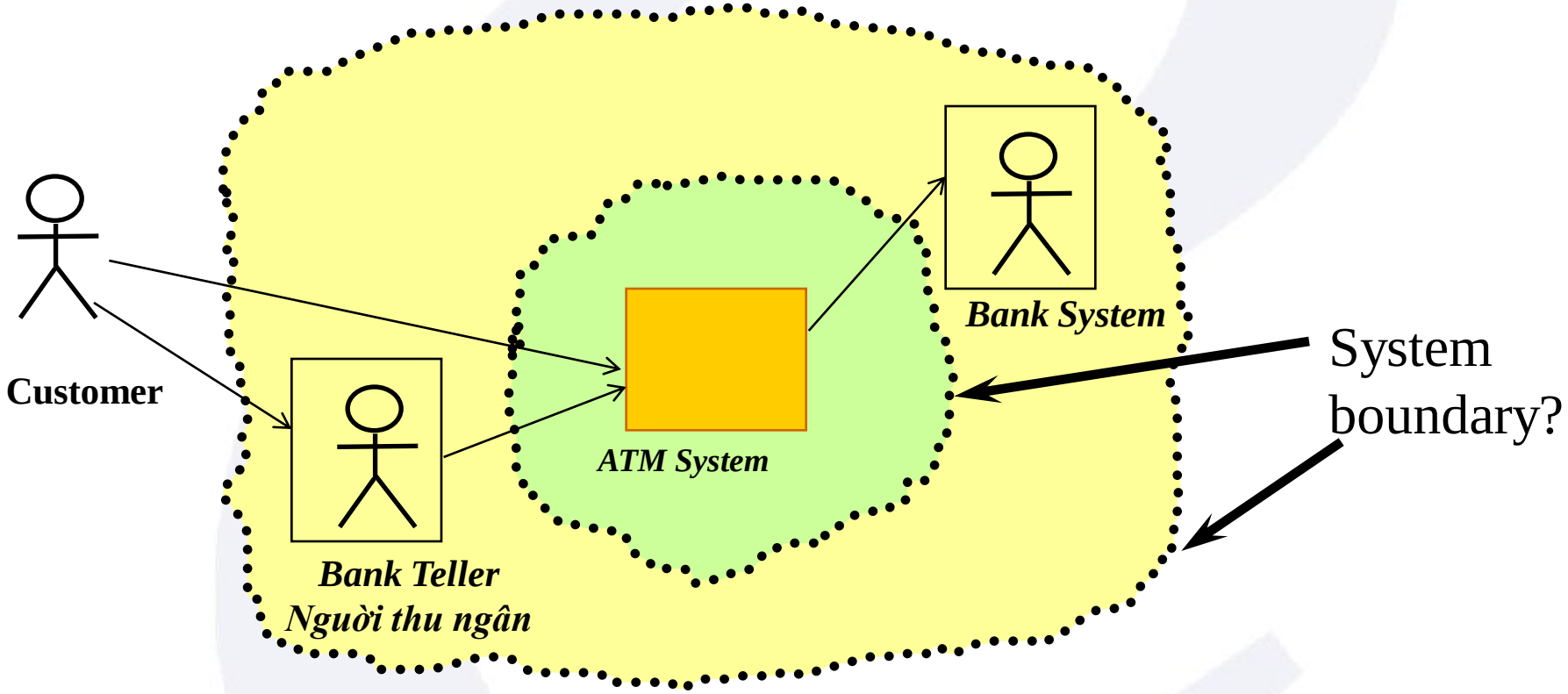
Làm thế nào để xác định Actor? Câu hỏi

- Ai là người cung cấp, sử dụng hoặc lấy thông tin từ hệ thống ?
- Ai sẽ sử dụng các tính năng của chương trình ?
- Người quan tâm tới một yêu cầu nào đó ?
- Nơi nào trong tổ chức(phòng ban, công ty) sẽ sử dụng hệ thống ?
- Ai là người duy trì, bảo dưỡng và quản lý hệ thống? Những tài nguyên bên ngoài hệ thống là gì ?
- Có những hệ thống nào khác tương tác với hệ thống này không?

Ví dụ

- Người dùng những chức năng chính của hệ thống
- Người dùng những chức năng phụ, như là quản trị hệ thống.
- Những thiết bị phần cứng bên ngoài
- Những hệ thống khác có tương tác trao đổi thông tin với hệ thống. Nếu xây dựng ứng dụng trên nền internet, có thể có tác nhân "vô danh"

Actors và giới hạn hệ thống (System Boundary)

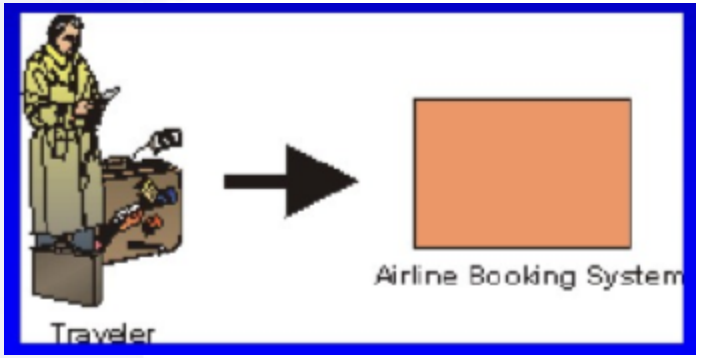


Actors và giới hạn hệ thống

- Tìm kiếm tác nhân cũng có nghĩa là xác định phạm vi của hệ thống, giúp ta xác định mục đích và qui mô của hệ thống cần xây dựng.
- Chỉ những người nào có tương tác trực tiếp với hệ thống mới được xem là tác nhân
- Ví dụ: trong hệ thống đăng ký vé, cần xét các trường hợp
  - Khách hàng mua vé thông qua nhân viên du lịch (travel agent) => **không là tác nhân** của hệ thống.
  - Khách hàng có thể đăng ký vé trực tiếp thông qua internet => **tác nhân.**

Ví dụ


Traveler
Travel Agent
Airline Booking System

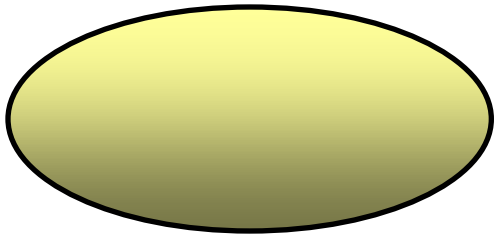

Traveler
Airline Booking System

Đặc tả Actor

- Mô tả
  - Mô tả ngắn gọn về Actor
- Đặc điểm
  - Những đặc điểm chính của Actor
- Mối quan hệ
  - Những mối quan hệ liên quan đến Actor
    - ✓ Quan hệ với các UseCase
    - ✓ Quan hệ tổng quát hoá với các Actor khác
- Lược đồ
  - Lược đồ những thành phần liên quan

Khái niệm – Use-Case

- Một use case xác định một tập các thể hiện use case
- Trong đó **mỗi thể hiện** là một chuỗi các hành động được hệ thống thực hiện và đem lại một kết quả thấy được có ý nghĩa đối với một actor cụ thể nào đó.



Use-Case

Giải thích

- Thể hiện Use Case và Use Case
  - Hệ thống thực hiện thao tác đăng nhập của nhân viên A
  - Hệ thống thực hiện thao tác đăng nhập của nhân viên B

  => “đăng nhập” là một Use Case

Làm thế nào để xác định Use Case ?

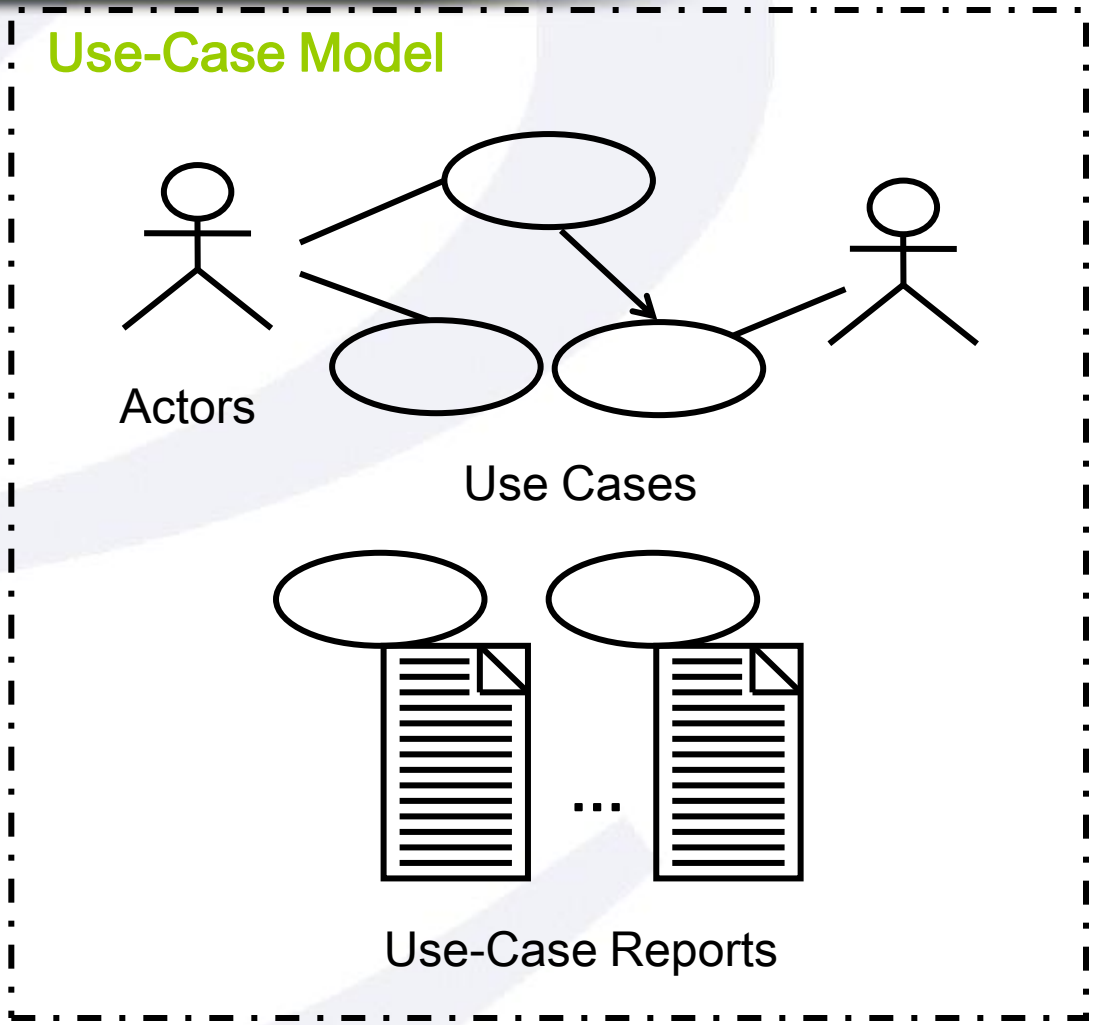
- Đối với mỗi Actor xác định, những công việc nào liên quan đến Actor này?
- Hệ thống hỗ trợ những nghiệp vụ nào trong thế giới thực ?
- Những thông tin nào cần được quản lý trong hệ thống ?
- Những thông tin nào cần được kết xuất ra khỏi hệ thống ?

Các bước thực hiện

- Xác định qui trình nghiệp vụ được hỗ trợ
- Xác định các đối tượng thông tin cần quản lý
  - Các Use Case dạng quản lý, tra cứu, kết xuất liên quan đến các đối tượng thông tin này
- Các nghiệp vụ, các xử lý chính
- Các báo cáo, kết xuất
- Các nghiệp vụ liên quan đến quản lý, duy trì thông tin hệ thống
- Các chức năng liên quan đến các yêu cầu đặc biệt (an toàn, bảo mật, thay đổi giao diện, màu sắc…)

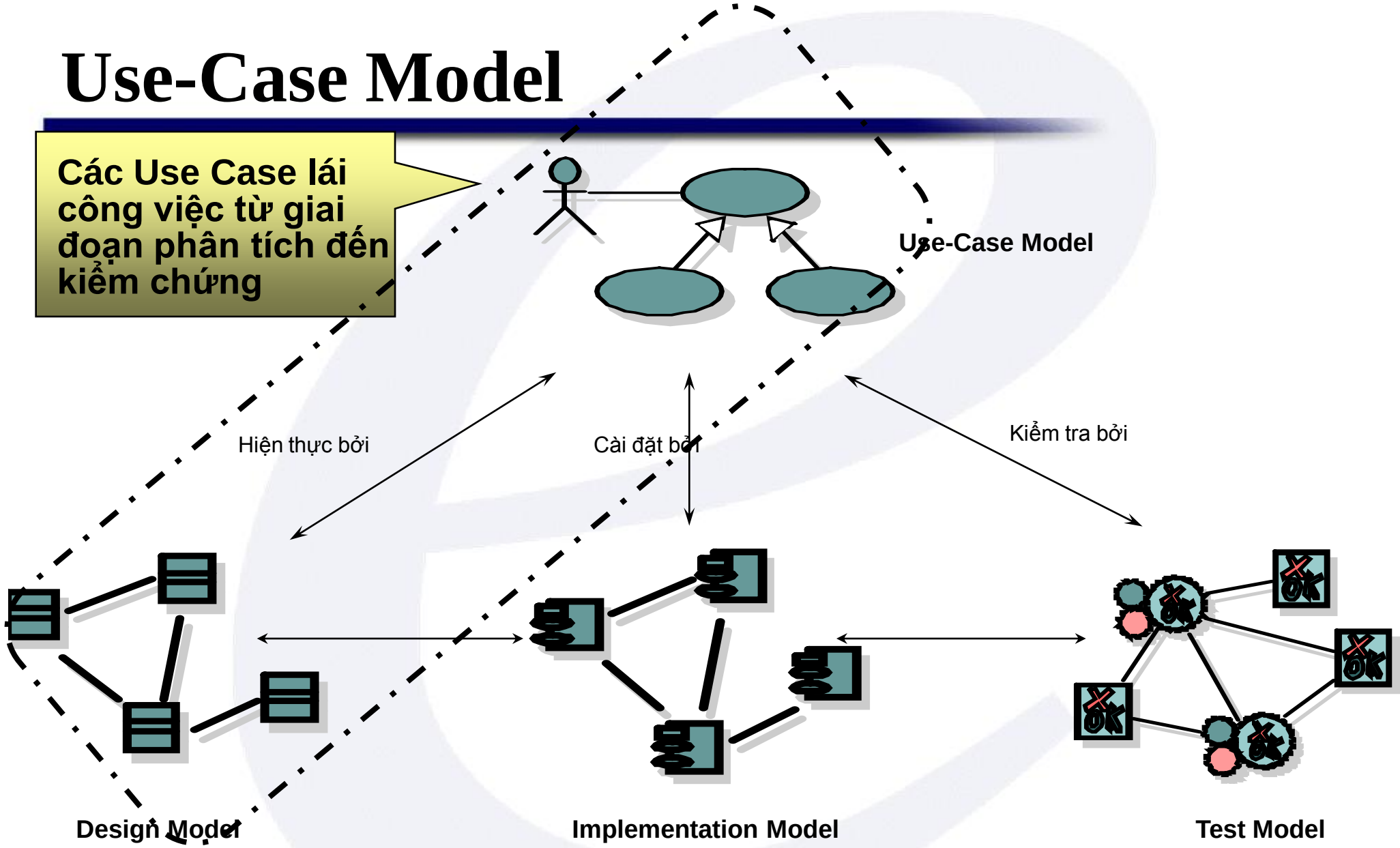
Use-Case Model

- Giới thiệu
- Survey Description
- Use-Case Packages
- Use Cases
- Actors
- Relationships
- Diagrams
- Use-Case View

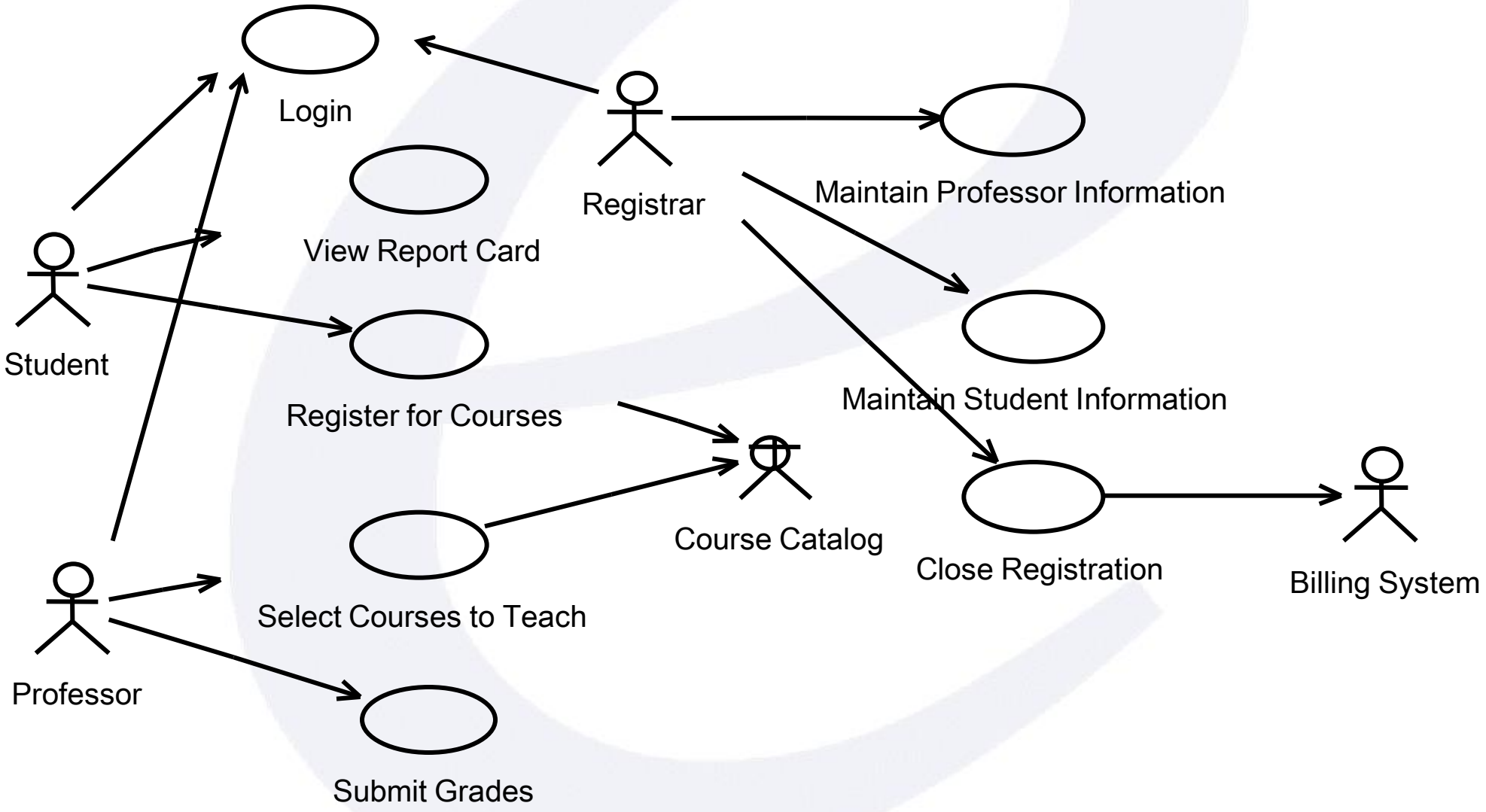


Use-Case Model

Các Use Case lái công việc từ giai đoạn phân tích đến kiểm chứng

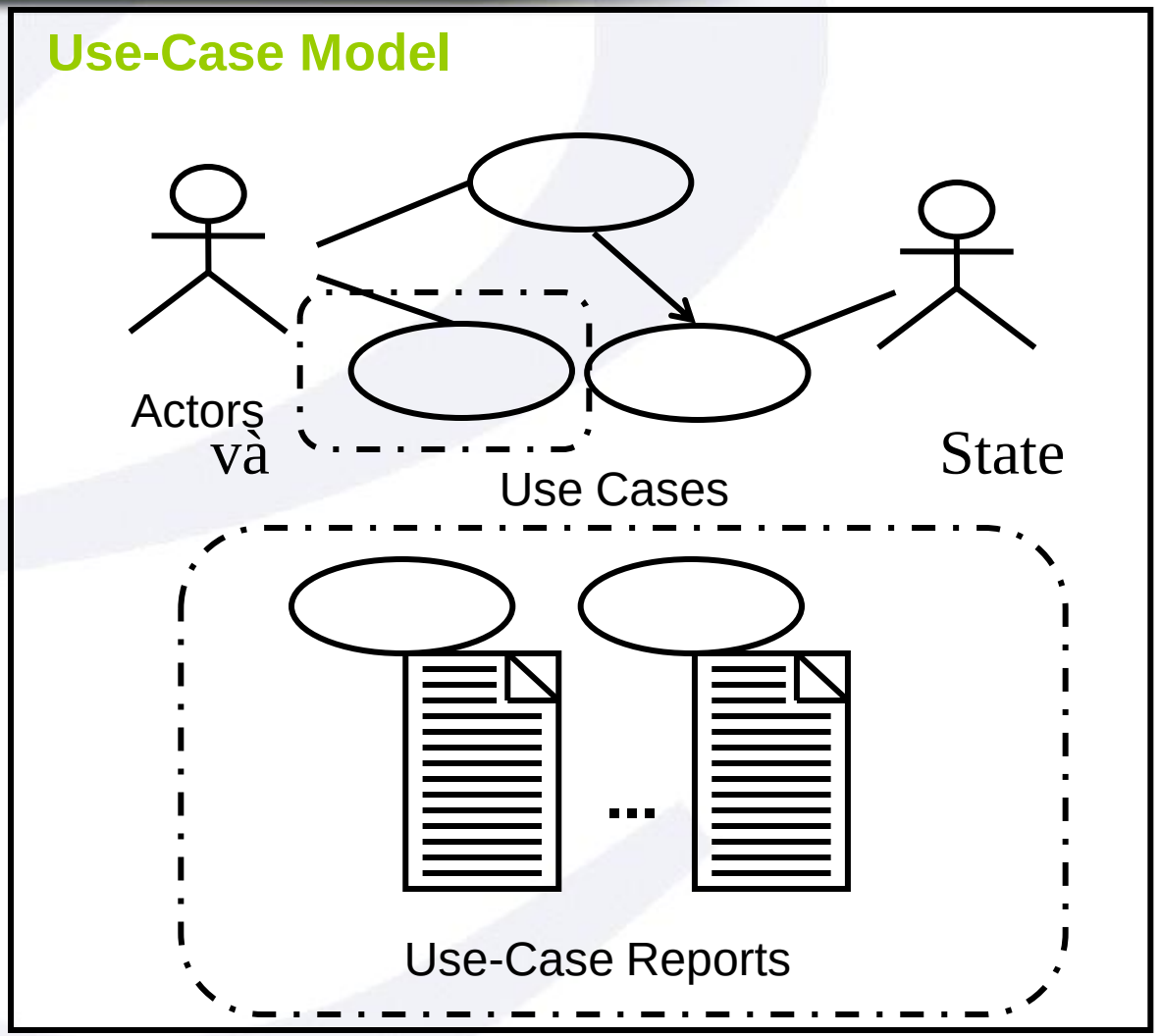


Ví dụ: Use-Case Model: Use-Case Diagram



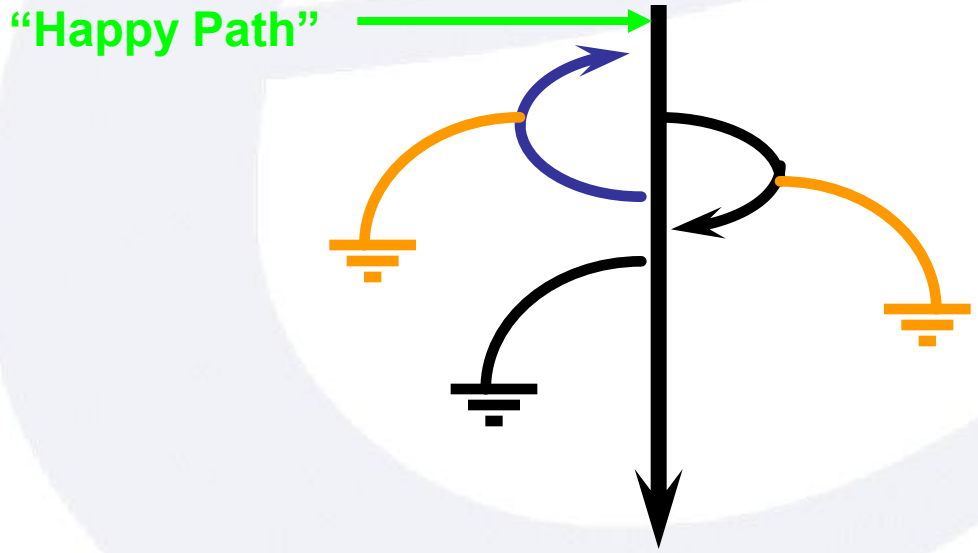
Đặc tả Use Case

- Tên
- Brief description
- Luồng các sự kiện
- Relationships
- Activity diagrams
- Use-Case diagrams
- Special requirements
- Preconditions
- Postconditions
- Các diagram khác



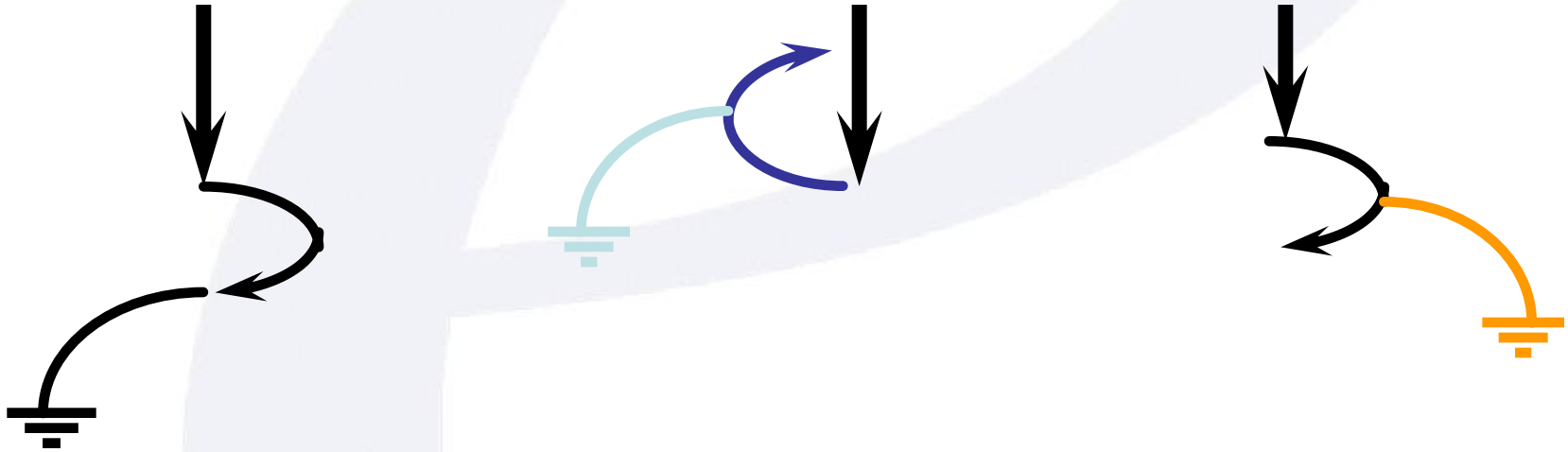
Luồng sự kiện (Use-Case Flows of Events)

- Của một *basic flow* (“Happy Path”)
- Một số *alternative flows*
  - Các biến thể thường gặp (Regular variants)
  - Các trường hợp bất thường (Odd cases)
  - **Exceptional flows** xử lý các tình huống lỗi

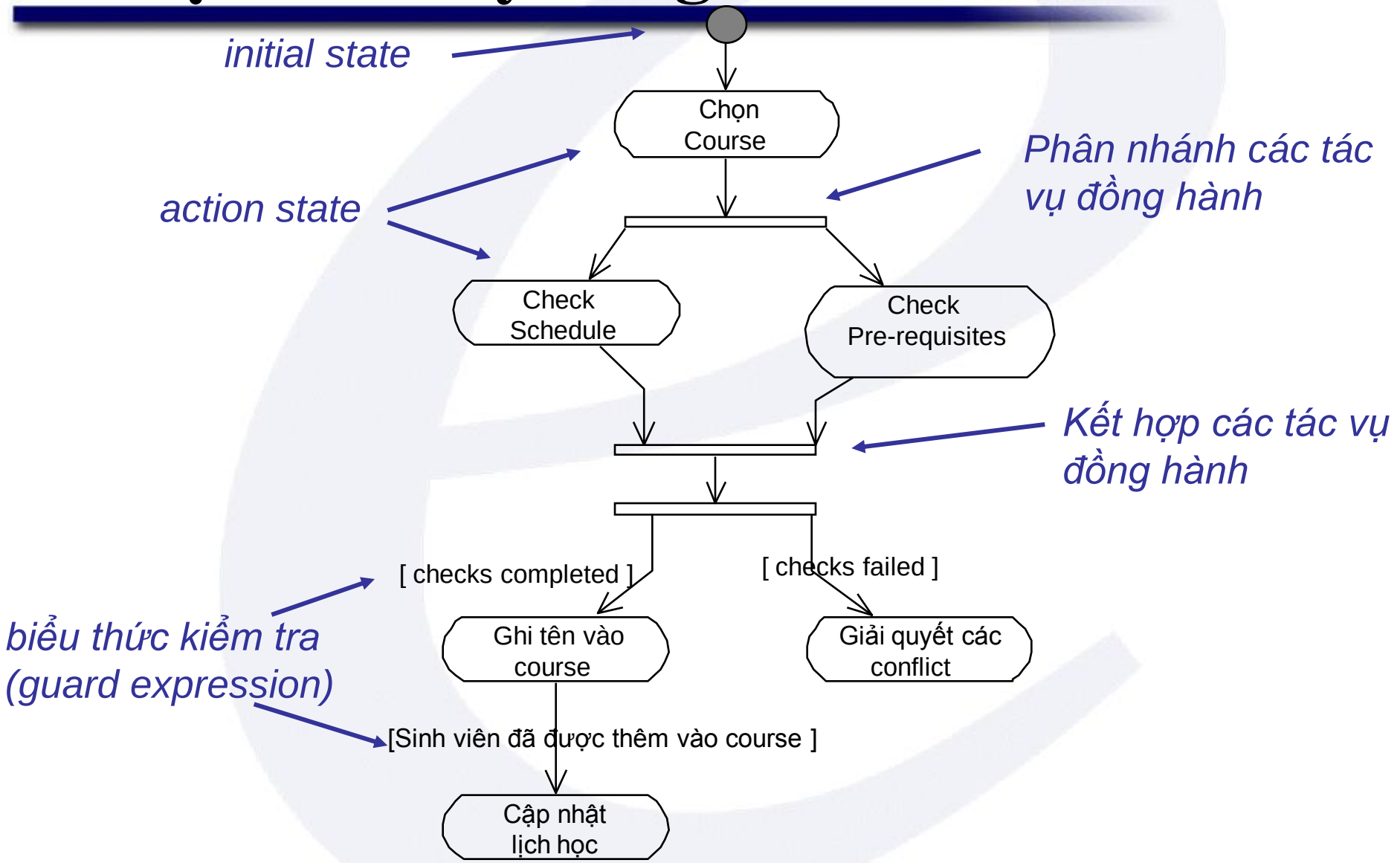


Scenarios là gì ?

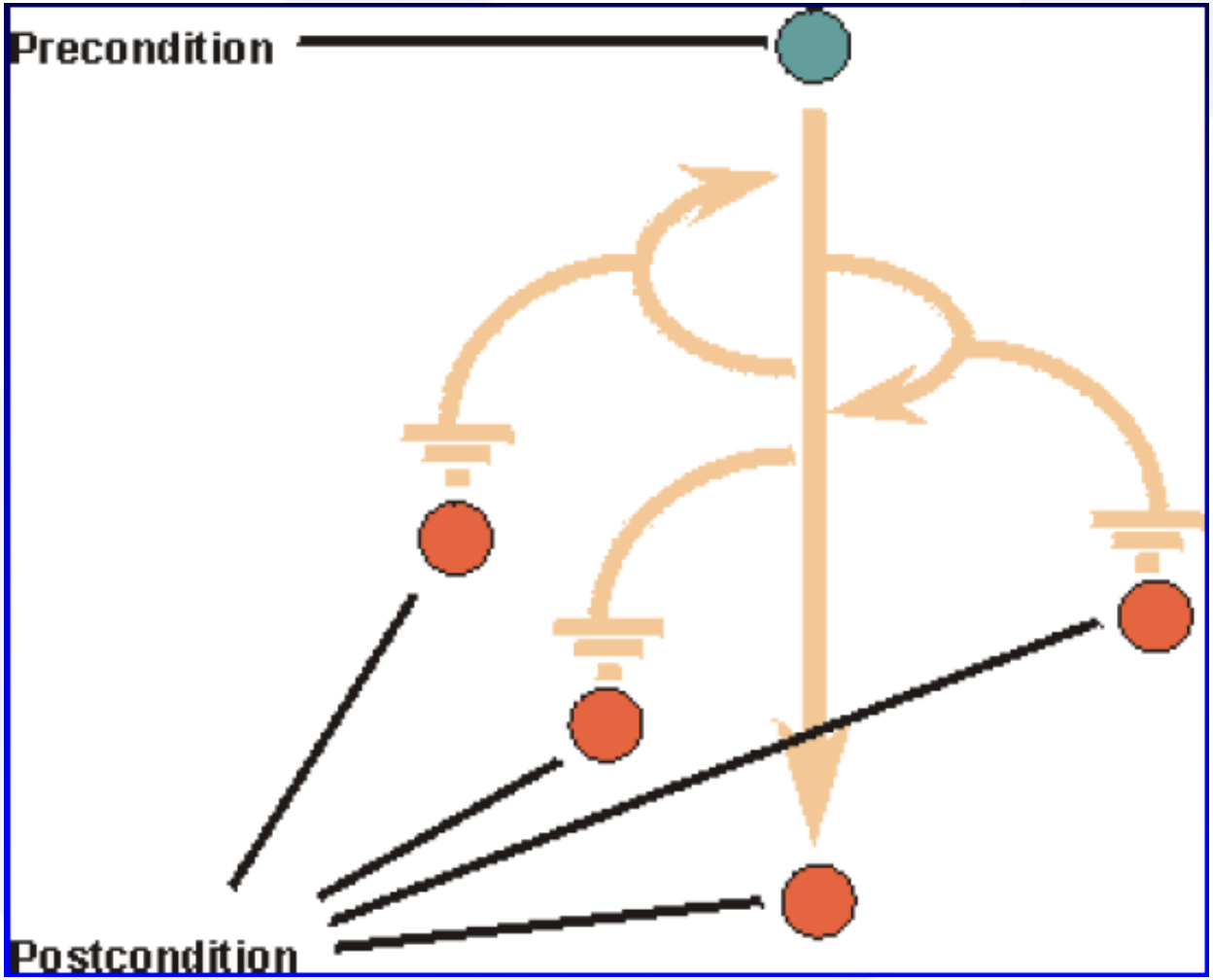
- Scenario là một thể hiện của use case



Ví dụ: Activity Diagram



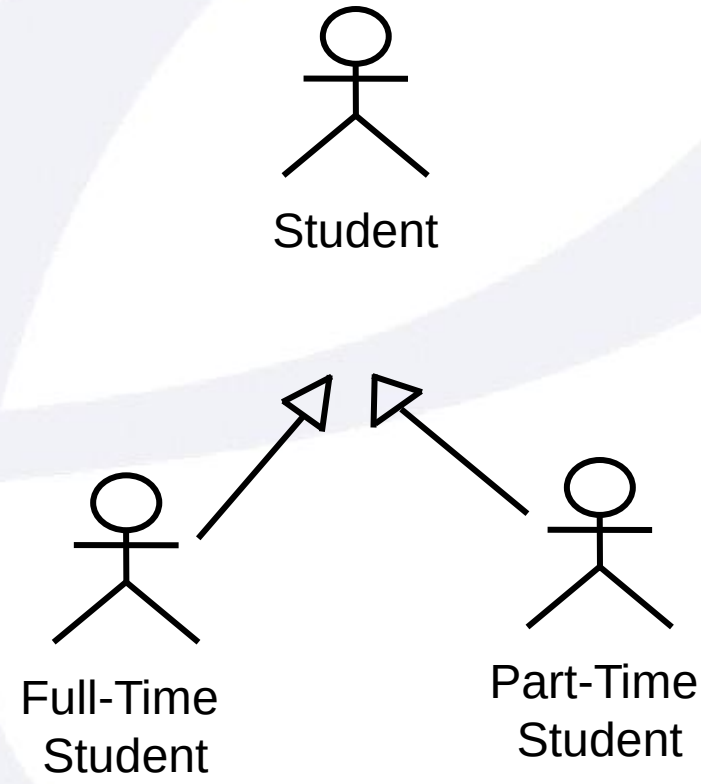
Pre và Post Conditions



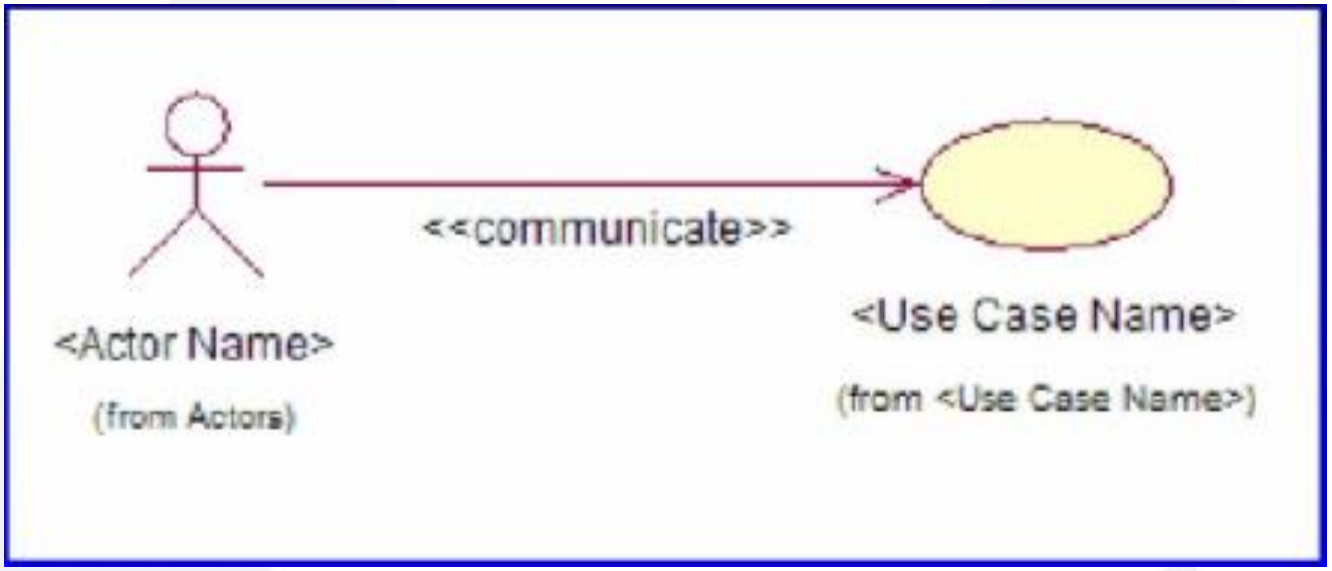
Relationship

- Quan hệ giữa các Actor
- Quan hệ giữa Actor và Use Case
- Quan hệ giữa các Use Case

Actor Generalization (Tổng quát hóa)

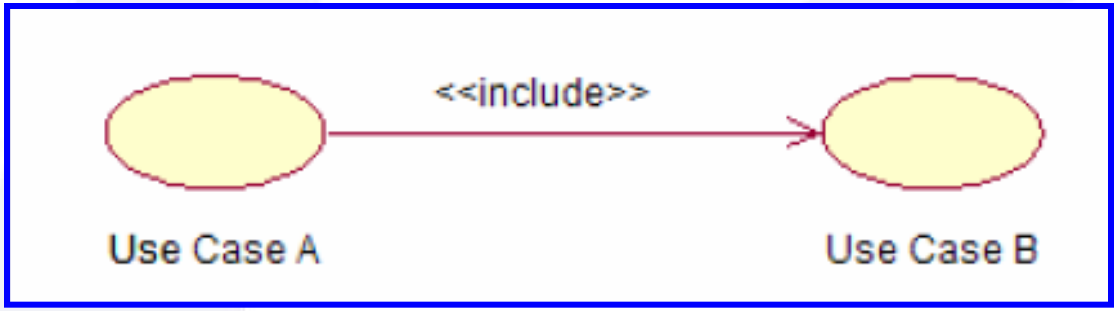


Actor & Use Case Relationship

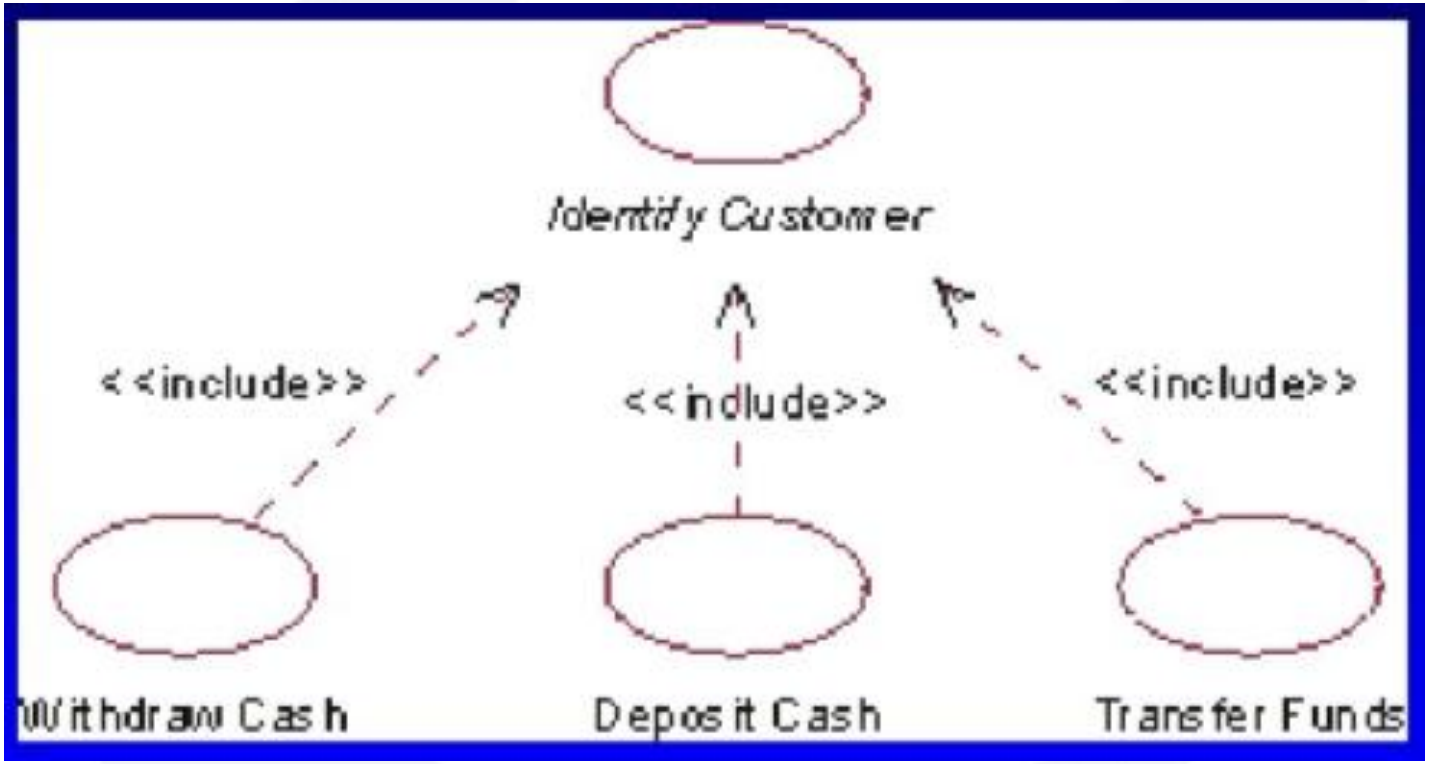


Use Case Relationship - Include

- <<Include>> - bao hàm, sử dụng

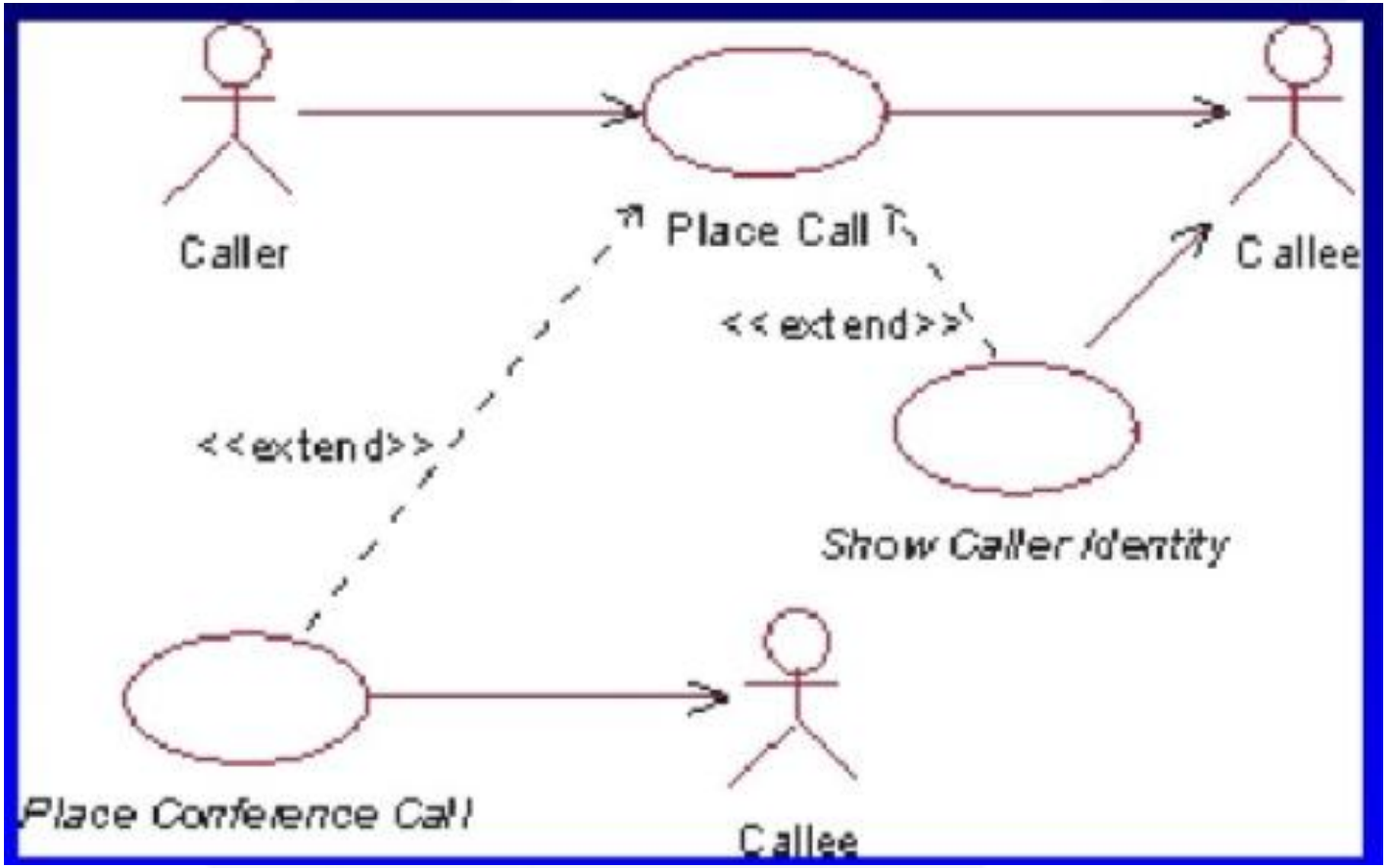


Ví dụ

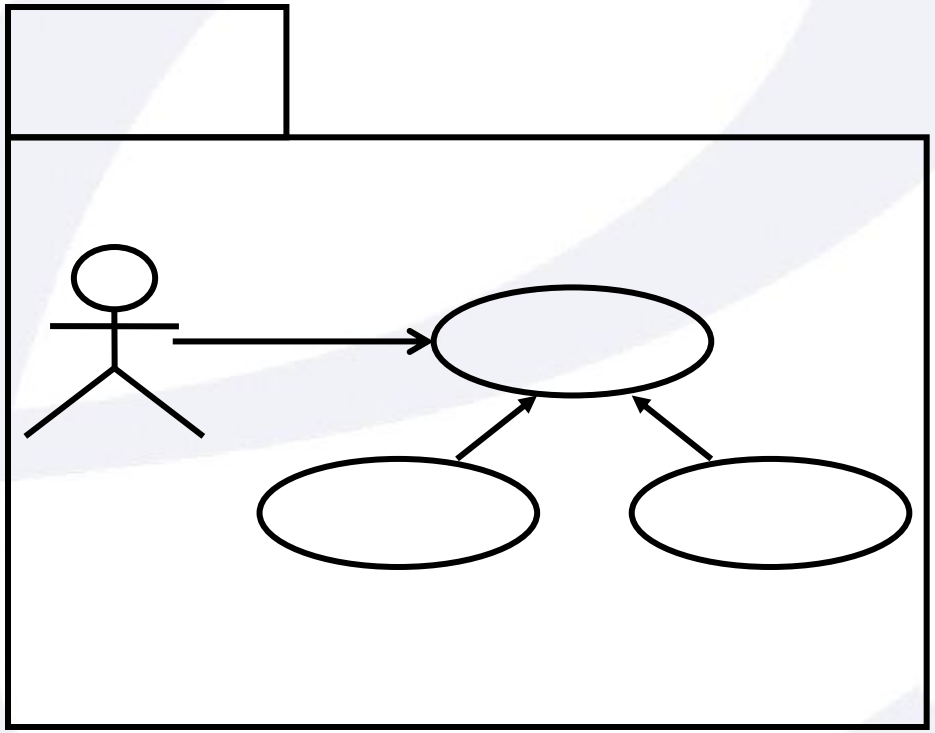

Identify Customer
<<include>>
<<include>>
<<include>>
Withdraw Cash
Deposit Cash
Transfer Funds

Use Case Relationship - Extend

- <<Extend>> - mở rộng



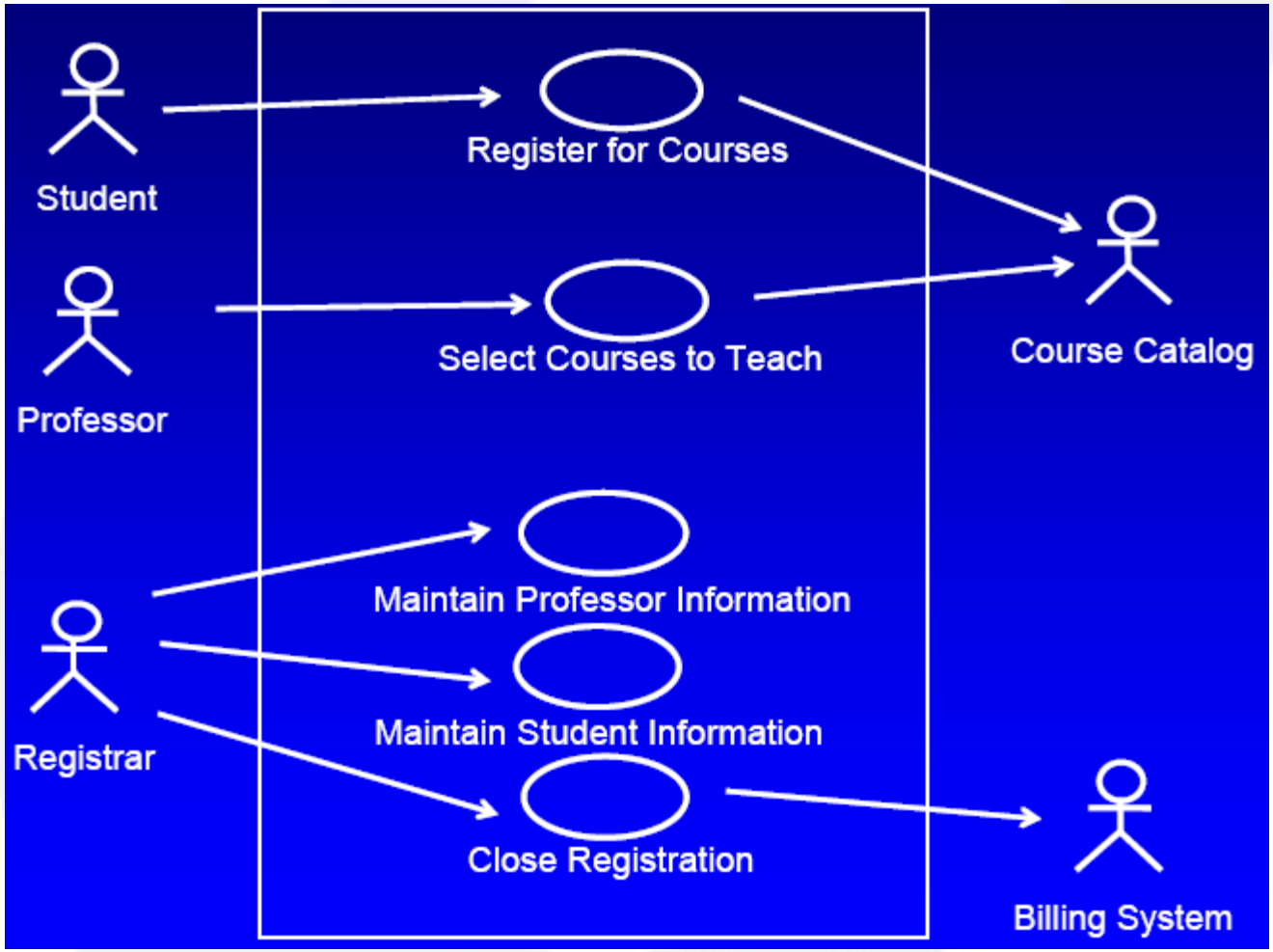
Package trong Use-Case Model



Ví dụ: Đặc tả Use-Case

- Điểm lại đặc tả của một use-case hoàn chỉnh, mô tả các yêu cầu của ứng dụng “Course Registration”

Ví dụ - Course Registration



Phân tích qua ví dụ

- Hệ thống quản lý giải vô địch bóng đá
- Hệ thống hỗ trợ thi trắc nghiệm qua mạng
- Hệ thống quản lý siêu thị
- …….

Nội dung

- Giới thiệu
- Các khái niệm chính
  - Actor
  - Use Case
  - Use Case Model
- **Phát biểu bài toán (Problem Statement)**
- Bảng chú giải
- Các đặc tả bổ sung
- Checkpoints

Ví dụ: Course Registration Problem Statement

- Là người phụ trách Tin học của trường Đại học KHTN, bạn được yêu cầu phát triển một hệ thống đăng ký học phần mới. Hệ thống mới cho phép sinh viên đăng ký học phần và xem phiếu điểm từ bất kì máy tính cá nhân nào được kết nối vào mạng nội bộ của trường. Các giáo sư cũng có thể truy cập hệ thống này để đăng ký lớp dạy và nhập điểm cho các môn học.
- Do kinh phí bị giảm nên trường không đủ khả năng thay đổi toàn bộ hệ thống trong cùng một lúc. Trường sẽ giữ lại cơ sở dữ liệu (CSDL) sẵn có về danh mục học phần mà trong đó lưu trữ toàn bộ thông tin về các học phần. Đây là một CSDL quan hệ và có thể truy cập bằng các câu lệnh SQL thông qua các server của trường. Hiệu suất của hệ thống cũ này rất kém nên hệ thống mới phải bảo đảm truy cập dữ liệu trên hệ thống cũ một cách hợp lý hơn. Hệ thống mới sẽ đọc các thông tin học phần trên CSDL cũ nhưng sẽ không cập nhật chúng. Phòng Đào tạo sẽ tiếp tục duy trì thông tin các học phần thông qua một hệ thống khác.
- Ở đầu mỗi học kỳ, sinh viên có thể yêu cầu danh sách các học phần được mở trong học kì đó. Thông tin về mỗi học phần, ví dụ như tên giáo sư, khoa, các học phần tiên quyết sẽ được cung cấp để giúp sinh viên chọn lựa.

Ví dụ: Course Registration Problem Statement

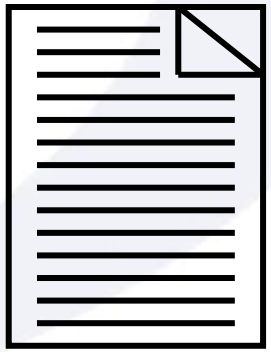
- Hệ thống mới cho phép sinh viên chọn bốn học phần được mở trong học kỳ tới. Thêm vào đó mỗi sinh viên có thể đưa ra hai môn học thay thế trong trường hợp không thể đăng ký theo nguyện vọng chính. Các học phần được mở có tối đa là 100 và tối thiểu là 30 sinh viên. Các học phần có ít hơn 30 sinh viên sẽ bị hủy. Đầu mỗi học kỳ, sinh viên có một khoảng thời gian để thay đổi các học phần đã đăng ký. Sinh viên chỉ có thể thêm hoặc hủy học phần đã đăng ký trong khoảng thời gian này. Khi quá trình đăng ký hoàn tất cho một sinh viên, hệ thống đăng ký sẽ gửi thông tin tới hệ thống thanh toán (billing system) để sinh viên có thể đóng học phí. Nếu một lớp bị hết chỗ trong quá trình đăng ký, sinh viên sẽ được thông báo về sự thay đổi trước khi xác nhận việc đăng ký học phần.
- Ở cuối học kỳ, sinh viên có thể truy cập vào hệ thống để xem phiếu điểm. Bởi vì thông tin về điểm của mỗi sinh viên cần được giữ kín, nên hệ thống cần có cơ chế bảo mật để ngăn chặn những truy cập không hợp lệ.
- Các giáo sư có thể truy cập vào hệ thống để đăng ký những học phần mà họ sẽ dạy. Họ có thể xem danh sách các sinh viên đã đăng ký vào lớp của họ, cũng như nhập điểm sau mỗi khóa học.

Nội dung

- Giới thiệu
- Các khái niệm chính
  - Actor
  - Use Case
  - Use Case Model
- Phát biểu bài toán
- **Bảng chú giải**
- Các đặc tả bổ sung
- Checkpoints

Từ điển thuật ngữ (Glossary)

- Giới thiệu
- Bảng chú giải



Glossary

Ví dụ: Glossary

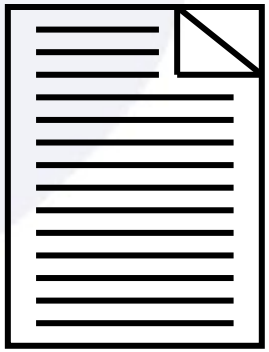
- **Bảng chú giải của ứng dụng "Course Registration":**
- Course (Học phần): Một môn học được dạy trong trường.
- Course Offering (Lớp): Một lớp học cụ thể được mở trong một học kỳ cụ thể – cùng một học phần có thể được mở song song nhiều lớp trong một học kỳ. Thông tin gồm các ngày học trong tuần và giờ học.
- Course Catalog (Danh mục học phần): Danh mục đầy đủ của tất cả các học phần được dạy trong trường.
- Faculty: Toàn bộ cán bộ giảng dạy của trường..
- Finance System (Hệ thống thanh toán): Hệ thống dùng để xử lý các thông tin thanh toán học phí.
- Grade (Điểm số): Sự đánh giá cho một sinh viên cụ thể trong một lớp cụ thể.
- Professor (Giáo sư): Người giảng dạy trong trường.
- Report Card (Phiếu diểm): Toàn bộ điểm số cho tất cả học phần một sinh viên đã học trong một học kỳ xác dịnh.
- Roster (Danh sách sinh viên đăng ký): Tất cả sinh viên đăng ký vào một lớp học cụ thể.
- Student (Sinh viên): Người đăng ký vào học các lớp của trường.
- Schedule (Lịch học): Các học phần mà một sinh viên đã chọn học trong học kỳ hiện tại.
- Transcript (Bản sao học bạ): Bản sao tất cả điểm số cho tất cả các học phần của một sinh viên cụ thể được chuyển cho hệ thống thanh toán để hệ thống này lập hóa đơn cho sinh viên.

Nội dung

- Giới thiệu
- Các khái niệm chính
  - Actor
  - Use Case
  - Use Case Model
- Phát biểu bài toán
- Bảng chú giải
- **Các đặc tả bổ sung**
- Checkpoints

Các đặc tả bổ sung

- Functionality
- Tính khả dụng (Usability)
- Tính tinh cậy (Reliability)
- Tính hiệu năng (Performance)
- Tính hỗ trợ (Supportability)
- Các ràng buộc thiết kế



Supplementary Specification

Ví dụ: Các đặc tả bổ sung

- **Tài liệu tham khảo**
  - Không có.
- **Chức năng**
  - Hỗ trợ nhiều người dùng làm việc đồng thời.
  - Nếu một lớp bị hết chỗ khi một sinh viên đang đăng ký học của lớp đó thì sinh viên này phải được thông báo.
- **Tính khả dụng**
  - Giao diện người dùng tương thích Windows 95/98.
- **Tính ổn định**
  - Hệ thống phải hoạt động liên tục 24 giờ/ngày, 7 ngày/tuần, với thời gian ngừng hoạt động không quá 1**0%.**
- **Hiệu suất**
  - Hệ thống phải hỗ trợ đến 2000 người dùng truy xuất CSDL trung tâm đồng thời bất kỳ lúc nào, và đến 500 người dùng truy xuất các server cục bộ.
  - Hệ thống phải cho phép truy xuất đến CSDL danh mục học phần cũ với độ trễ không quá 10 giây.
  - Hệ thống phải có khả năng hoàn tất 80% giao dịch trong vòng 2 phút.
- **Sự hỗ trợ**
  - Không có.
- **Tính bảo mật**
  - Hệ thống phải ngăn chặn sinh viên thay đổi lịch học của người khác, và ngăn các giáo sư thay đổi lớp dạy của giáo sư khác.
  - Chỉ có giáo sư mới có thể nhập điểm cho sinh viên.
  - Chỉ có cán bộ đào tạo mới được phép thay dổi thông tin của sinh viên.
- **Các ràng buộc thiết kế**
  - Hệ thống phải tích hợp với hệ thống có sẵn, Hệ thống danh mục học phần, một CSDL RDBMS.
  - Hệ thống phải cung cấp giao diện dựa trên Windows.

Nội dung

- Giới thiệu
- Các khái niệm chính
- Phát biểu bài toán
- Bảng chú giải
- Use-Case Model
- Các đặc tả bổ sung
- **Checkpoints**

Checkpoints: Requirements: Use-Case Model

- Use-case model có dễ hiểu không?
- Sau khi nghiên cứu use-case model, bạn có hình thành được một ý tưởng rõ ràng về các chức năng của hệ thống và cách thức mà chúng liên hệ với nhau ?
- Đã xác định hết tất cả các actor? Tất cả các yêu cầu chức năng được thỏa hay chưa?
- Use-case model có chứa các hành vi vô dụng nào không?
- Việc chia model thành các use-case package có xác đáng?

Checkpoints: Requirements: Actors

- Đã xác định hết tất cả các actor?
- Mỗi actor có tham gia vào ít nhất một use case?
- Mỗi actor thật sự có một vai trò (role)? Có cần ghép hoặc tách các actor không?
- Có tồn tại 2 actor dùng cùng một vai trò đối với 1 use case không?
- Tên của các actor có gợi nhớ không? Users và customers có hiểu tên của chúng?

Checkpoints: Requirements: Use-Cases

- Mỗi use case có ít nhất một actor tương tác?
- Các use case có độc lập với nhau?
- Tồn tại các use case có các luồng sự kiện và các hành vi tương tự nhau không?
- Liệu các use case có tên duy nhất, gợi nhớ, và dễ hiểu để chúng không bị nhầm lẫm trong các giai đoạn sau?
- Các khách hàng và người dùng có hiểu tên và mô tả của các use case không?

Checkpoints: Requirements: đặc tả Use-Case

- Use case có đủ rõ ràng đối với những người muốn hiên thực?
- Mục đích của use-case có rõ ràng?
- Mô tả sơ lược (Brief description) có cho ta hình ảnh trung thực của use-case?
- Có xác định rõ luồng sự kiện của use-case như thế nào và khi nào bắt đầu và kết thúc?
- Chuỗi các giao tiếp giữa actor và use-case có tiện nghi không (từ góc nhìn của người dùng)?
- Các tương tác và các thông tin trao đổi của actor có rõ ràng?
- Có use-case nào quá phức tạp không?
- Các luồng sự kiện (basic và alternative) được mô hình đúng đắn?

Checkpoints: Requirements: Glossary

- Các thuật ngữ có định nghĩa rõ ràng và súc tích?
- Mỗi thuật ngữ có dùng đâu đó trong các mô tả use-case?
- Các thuật ngữ có được sử dụng hợp lý trong các mô tả ngắn về các actor và use case?

Review

- Các thành phần chính của Đặc tả yêu cầu (Requirements)?
- Các thành phần của đặc tả yêu cầu được dùng để làm gì?
- Use-case model là gì?
- Actor là gì?
- Use case là gì? Liệt kê một số thuộc tính của use case.
- Sự khác nhau giữa use-case và scenario?
- Supplementary specification là gì và nó chứa những gì?
- Glossary là gì và nó chứa những gì?

Bài tập

- Mô tả các thành phần chính của Requirements:
  - Problem statement
  - Use-case model main diagram
  - Supplementary specification
  - Glossary
- Điểm lại các thông tin về yêu cầu người dùng được sử dụng trong mô hình, ghi chú tất cả các câu hỏi, các vấn dề còn tranh cãi, mâu thuẫn

Q/A

Biểu đồ ca sử dụng (Use case diagrams)

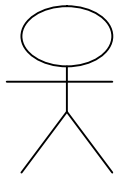
Mục đích của use case

- Ca sử dụng biểu diễn những chức năng mà hệ thống cần làm
- Các ca sử dụng cho phép:
  - Biết được hành vi của hệ thống mà không cần xác định làm thế nào hành vi này thực hiện
  - Định nghĩa những hạn chế chính xác của hệ thống
  - Cho người phát triển hiểu rõ hơn những gì mà khách hàng và người sử dụng chờ đợi

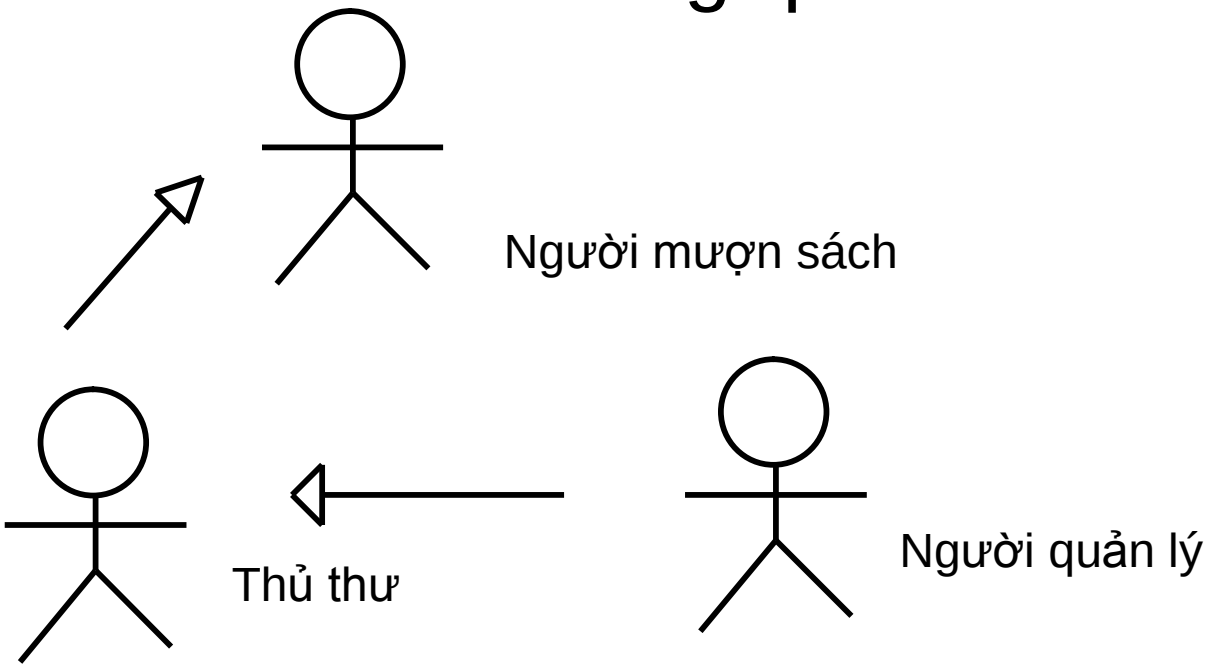
Mô hình ca sử dụng

- Một biểu đồ ca sử dụng định nghĩa:
  - Các tác nhân
  - Các ca sử dụng
  - Quan hệ giữa các tác nhân và các ca sử dụng
- Một mô hình ca sử dụng được định nghĩa bởi:
  - Các biểu đồ ca sử dụng
  - Phần mô tả bằng lời các kịch bản sử dụng
  - Phần mô tả các kịch bản dùng:
    - Biểu đồ tuần tự
    - Biểu đồ tương tác

Các tác nhân



- Một tác nhân là một người hoặc một thiết bị có phản ứng với hệ thống
- Quan hệ giữa các tác nhân: tổng quát hóa (thừa kế)

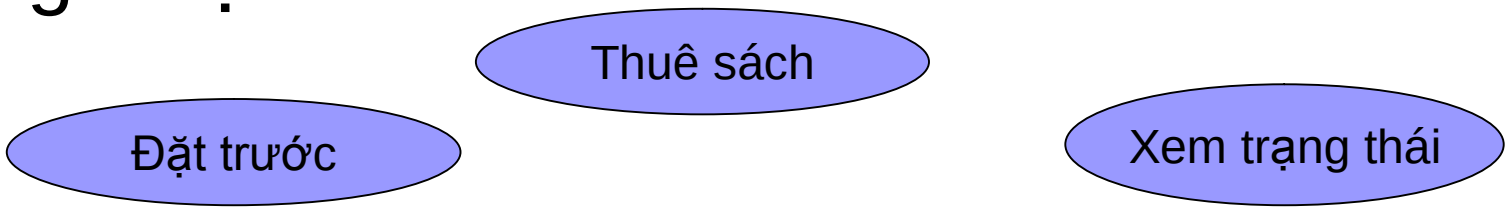


Tìm kiếm tác nhân như thế nào?

- Hãy trả lời các câu hỏi sau để tìm ra tác nhân hệ thống
  - Ai sẽ sử dụng chức năng chính của hệ thống?
  - Ai giúp hệ thống làm việc hàng ngày?
  - Ai quản trị, bảo dưỡng để hệ thống làm việc liên tục?
  - Hệ thống quản lý thiết bị phần cứng nào?
  - Hệ thống đang xây dựng tương tác với hệ thống khác nào?
  - Ai hay cái gì quan tâm đến kết quả hệ thống trả lại?

Các ca sử dụng

- Một ca sử dụng là một phương tiện để thể hiện các khả năng khác nhau sử dụng hệ thống
- Nó biểu diễn một chuỗi tương tác giữa tác nhân và ứng dụng
- Nó định nghĩa một chức năng có thể sử dụng được bởi tác nhân



Tìm kiếm UC như thế nào?

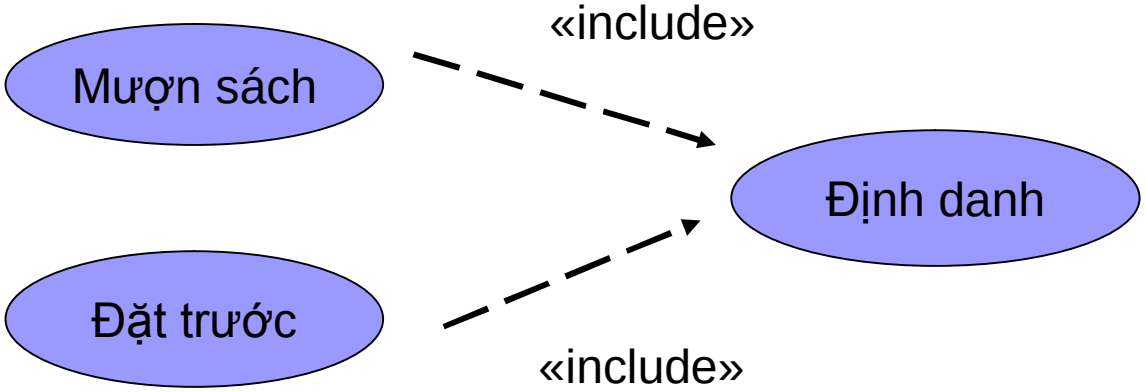
- Với mỗi tác nhân đã tìm ra, hãy trả lời các câu hỏi sau để tìm ra các Use case hệ thống
  - Tác nhân yêu cầu hệ thống thực hiện chức năng nào?
  - Tác nhân cần đọc, tạo lập, bãi bỏ, lưu trữ, sửa đổi các thông tin nào trong hệ thống?
  - Tác nhân cần thông báo cho hệ thống sự kiện xảy ra trong nó?
  - Hệ thống cần thông báo cái gì đó cho tác nhân?
  - Hệ thống cần vào/ra nào? Vào/ra đi đến đâu hay từ đâu?
- Đặt tên UC hệ thống
  - Theo khái niệm nghiệp vụ của tổ chức
  - Không sử dụng từ kỹ thuật, chuyên môn
  - Sử dụng các động từ, cụm từ ngắn gọn
- Tùy theo tầm cỡ dự án mà mỗi hệ thống có từ 20-70 UC

Đã tìm đầy đủ UC cho hệ thống?

- Các câu hỏi sau giúp xác định đã tìm đầy đủ UC?
  - Mỗi yêu cầu chức năng ở trong ít nhất một UC?
    - Nếu yêu cầu chức năng không ở trong UC nào thì nó sẽ không được cài đặt sau này.
  - Đã khảo sát mọi tác nhân tương tác với hệ thống?
  - Tác nhân cung cấp cho hệ thống thông tin nào?
  - Tác nhân nhận thông tin nào từ hệ thống?
  - Đã nhận biết mọi hệ thống bên ngoài tương tác với hệ thống đang xây dựng?
  - Thông tin nào hệ thống bên ngoài nhận và gửi cho hệ thống đang xây dựng?

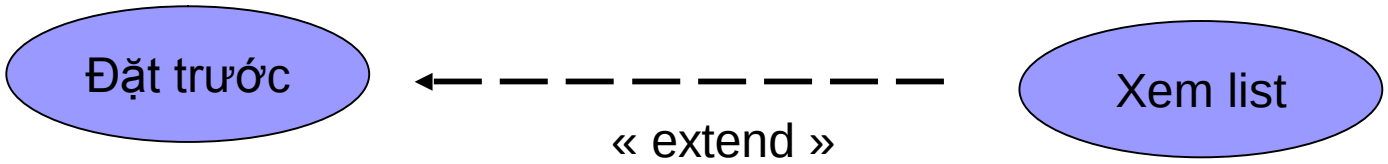
Tổ chức các ca sử dụng: include

- Quan hệ «*include*» biểu diễn một ca sử dụng chứa hành vi định nghĩa trong một ca sử dụng khác
- Quan hệ này cho phép biểu diễn phần chung các hành vi của nhiều ca sử dụng



Tổ chức các ca sử dụng: extend

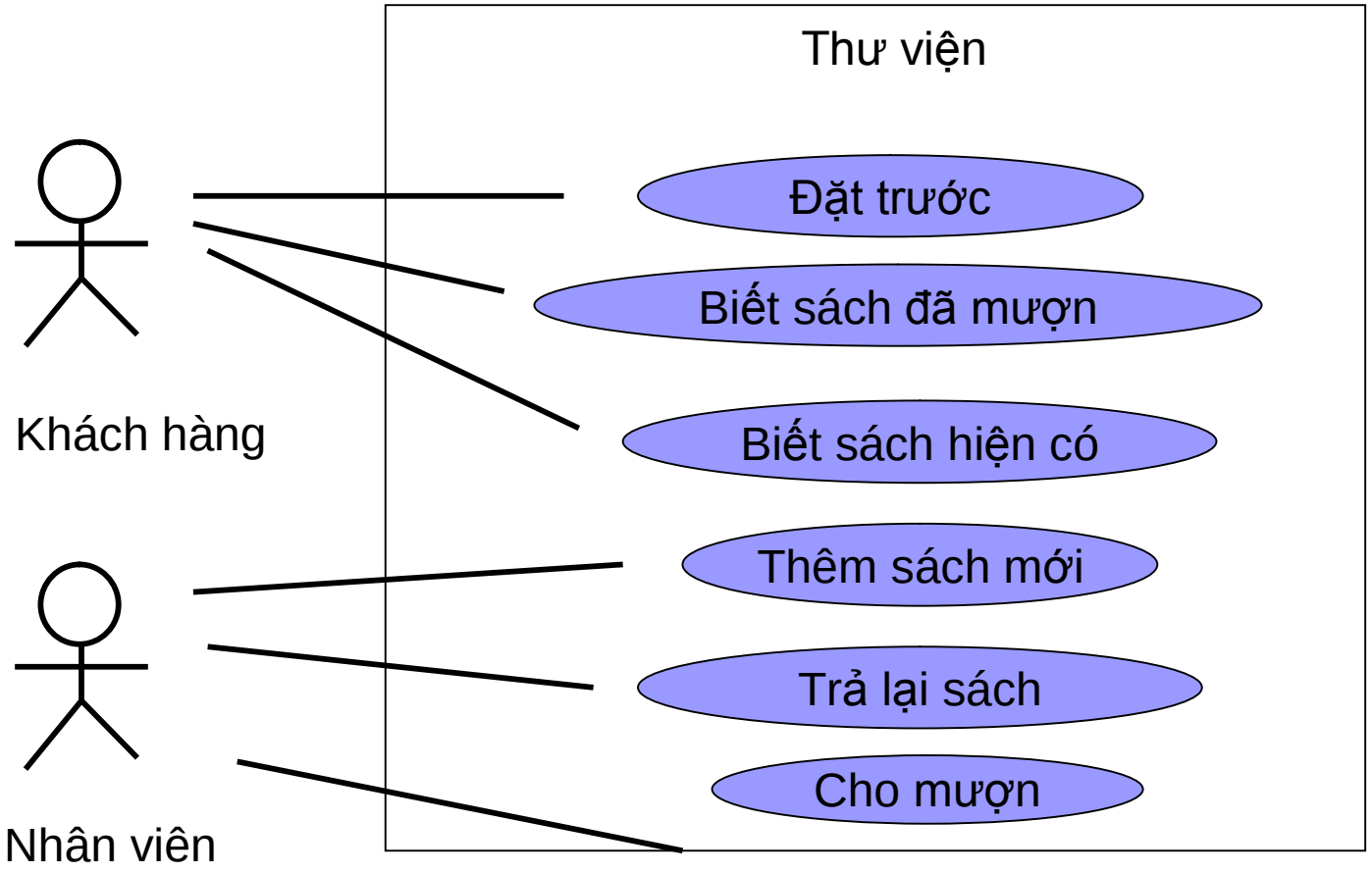
- Một UC tùy ý mở rộng chức năng do UC khác cung cấp
- Sử dụng để mô hình hóa một vài chức năng dùng chung, sử dụng lại giữa hai hay nhiều UC



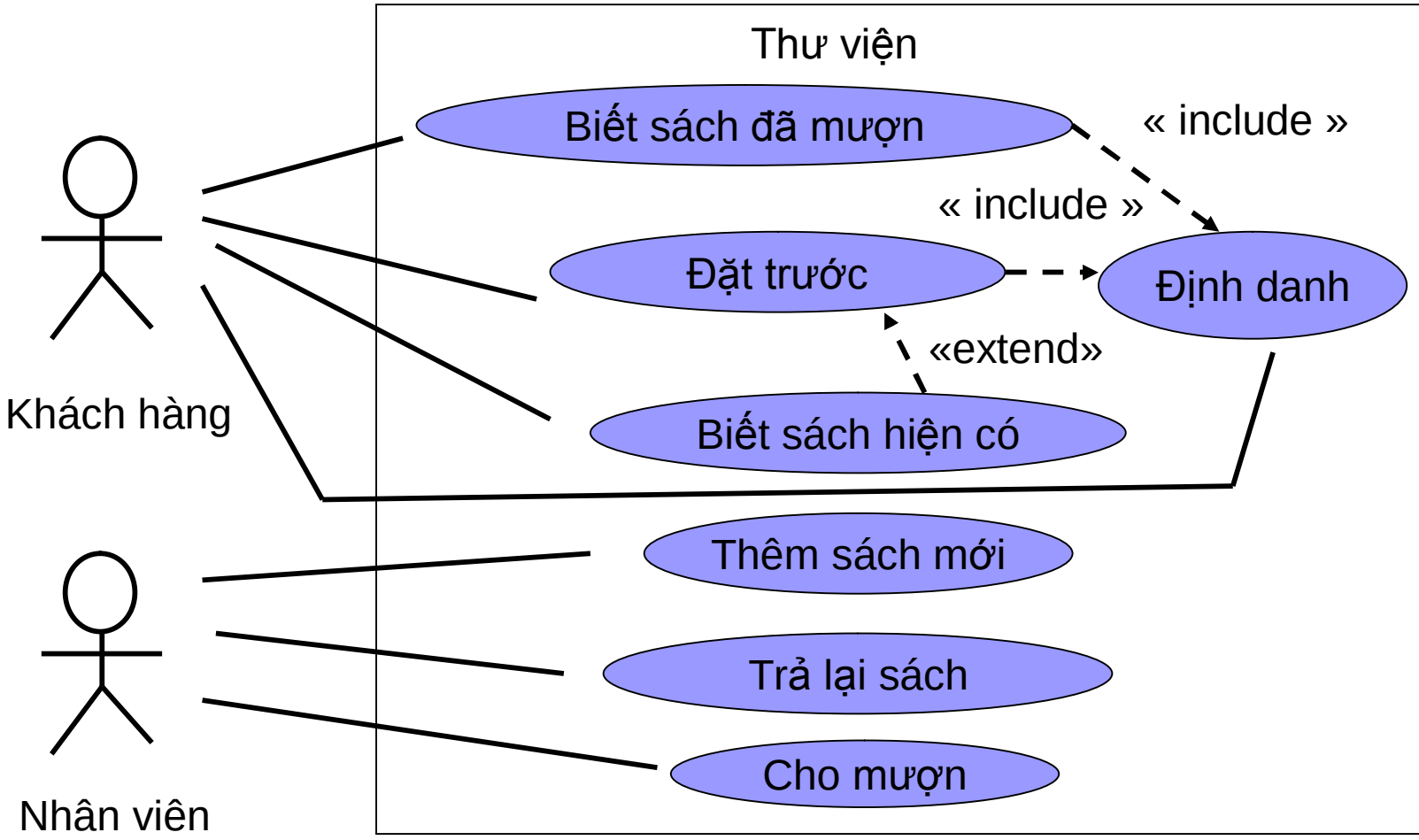
Mô hình hóa một hệ thống: xác định ca sử dụng

- Xác định các tác nhân sử dụng
- Với mỗi tác nhân, tìm kiếm các ca sử dụng với hệ thống. Đặc biệt những gì thay đổi trạng thái của hệ thống hoặc chờ đợi phản ứng từ hệ thống
- Tổ chức lại các ca sử dụng với các quan hệ sử dụng, mở rộng…

Biểu đồ ca sử dụng



Biểu đồ ca sử dụng



Luồng sự kiện trong UC

- Tài liệu luồng sự kiện (flow of events) mô tả hành vi của UC
  - mô tả luồng logic đi qua UC
  - mô tả người sử dụng làm gì, hệ thống làm gì
  - Trong một UC có nhiều luồng sự kiện: luồng chính, luồng phụ
- Kịch bản (Scenario)
  - Một luồng sự kiện trong một hiện thực của UC
  - Là trình tự hành động cụ thể để mô tả hành vi
  - Kịch bản đi xuyên suốt UC theo nhánh chính, nhánh phụ, nhánh đặc biệt

Tài liệu luồng sự kiện

- Tài liệu luồng sự kiện bao gồm
  - Mô tả vắn tắt UC
    - Mô tả ngắn gọn UC làm gì?
    - Những ai sử dụng UC?
    - Nó trả lại kết quả gì?
  - Tiền điều kiện (pre-condition)
    - Điều kiện cần thực hiện trước khi UC khởi động
    - Không phải UC nào cũng có tiền điều kiện
  - Luồng sự kiện chính và luồng sự kiện rẽ nhánh
  - Hậu điều kiện (post-condition)

Thí dụ tài liệu luồng sự kiện

- Làm tài liệu các luồng sự kiện cho UC “Purchase Ticket”
  - Các bước trong luồng sự kiện chính
    1. **UC bắt đầu khi customer chọn chức năng xem thông tin chuyến bay**
    2. **Hệ thống hiển thị thành phố đến, đi và thời gian hạ cánh, cất cánh**
    3. **User nhập nơi đến, đi, thời gian ngày tháng khởi hành và trở về**
    4. **Hệ thống hiển thị danh sách chuyến bay và giá vé**

       A1. Không còn chuyến bay
    5. **User chọn chuyến bay để đặt trước**
    6. **Hệ thống hiển thị các loại vé để user chọn**
    7. **User chọn giá vé**

       A2. User chọn giá vé cho thành viên frequent-flyer

Thí dụ tài liệu luồng sự kiện

8. Hệ thống hiển thị giá vé sẽ bán cho khách hàng
9. User khẳng định giá vé
10. Hệ thống hiển thị loại thẻ tín dụng, số thẻ, thời gian hết hạn
11. User nhập loại thẻ tín dụng, số thẻ, thời gian hết hạn
12. Hệ thống trình mua bằng thẻ
    - A3. Không thấy tài khoản
    - A4. Không đủ tiền
13. Hệ thống dành chỗ cho user
14. Hệ thống phát sinh và hiển thị mã xác thực cho user
15. User khẳng định đã nhận mã
16. Use case kết thúc

Thí dụ tài liệu luồng sự kiện

- Luồng phụ

  A1. Không có chuyến bay

  1. Hệ thống hiển thị thông điệp thông báo không có chuyến bay

  2. User khẳng định thông điệp

  3. Trở lại luồng chính Bước 2.

  A2. Vé dành cho thành viên frequent-flyer

  1. Hệ thống hiển thị số hiệu frequent-flayer

  2. User nhập số

  3. Hệ thống khẳng định tính hợp lệ của số

  A3. Số không hợp lệ

  …

Kịch bản của một ca sử dụng

- Khách hàng đứng trước máy vi tính
  1. Hệ thống hiển thị một thông điệp chào mừng
  2. Khách hàng chọn lựa thao tác đặt trước
  3. Hệ thống yêu cầu đăng nhập
  4. Khách hàng đưa định danh
  5. Hệ thống yêu cầu chọn sách
  6. Khách hàng chọn sách muốn mượn
  7. Hệ thống chuyển trạng thái sách thành đặt trước

References

- http://www.agilemodeling.com/essays/useCaseReuse.htm

Bài tập

- Vẽ biểu đồ use cases cho các hệ thống phần mềm:
  - Bán vé máy bay
  - Máy rút tiền ATM

Biểu đồ lớp

Class Diagrams

Trương Ninh Thuận

Biểu đồ lớp là gì?

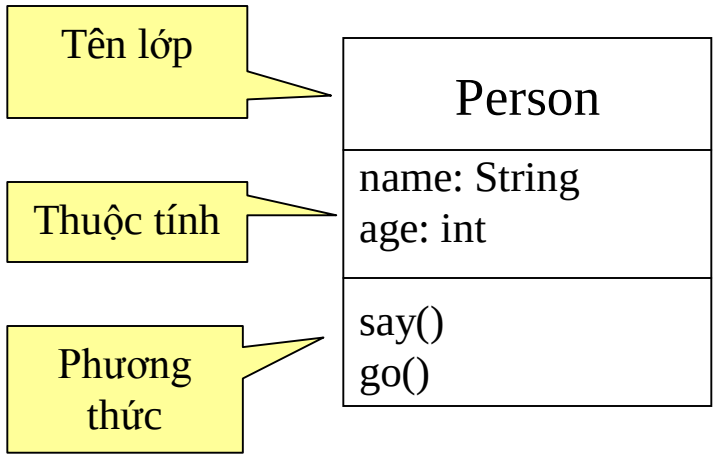
- Biểu đồ lớp mô tả kiểu của các đối tượng trong hệ thống và các loại quan hệ khác nhau tồn tại giữa chúng
- Là một kỹ thuật mô hình hóa tồn tại ở tất cả các phương pháp phát triển hướng đối tượng
- Biểu đồ hay dùng nhất trong UML.

Các phần tử của biểu đồ lớp

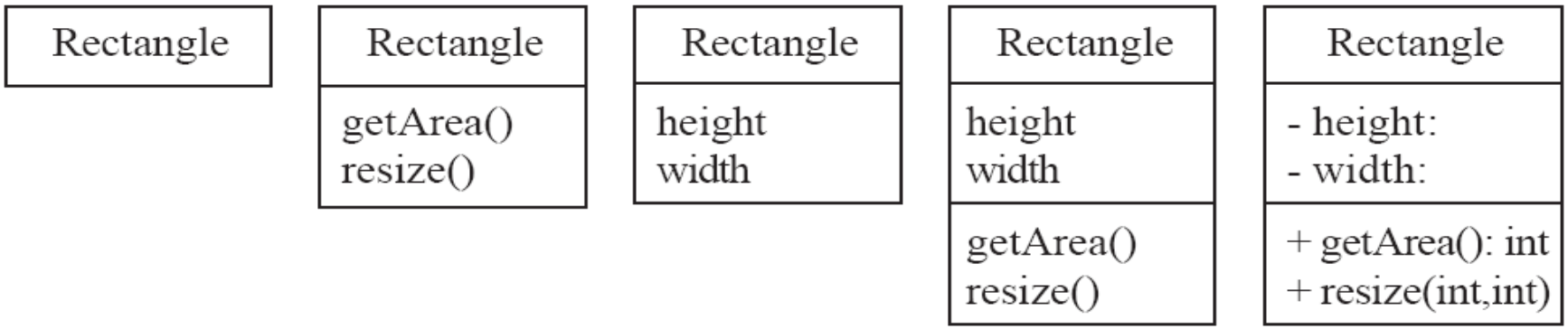
- Lớp
- Thuộc tính
- Phương thức
- Quan hệ
  - Liên kết (Associations)
  - Tổng quát hóa (Generalization)
  - Phụ thuộc (Dependency)
  - Thực hiện (Realization)
- Các luật ràng buộc và ghi chú

Lớp

- Một lớp là một mô tả của một tập các đối tượng có chung thuộc tính, phương thức và quan hệ



Biểu diễn lớp trong UML



Đặc tả thuộc tính lớp

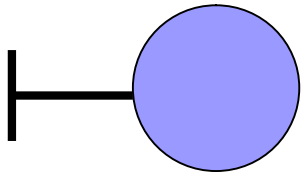
- Visibility
  - Đóng gói trong lập trình hướng đối tượng
  - Bốn lựa chọn phạm vi cho thuộc tính
    - Public: Mọi lớp đều nhìn thấy thuộc tính (+)
    - Private: Lớp khác không nhìn thấy thuộc tính (-)
    - Protected: Các lớp kế thừa có thể nhìn thấy (#)
    - Package và Implementation: Thuộc tính là public đối với các lớp trong cùng gói

Stereotype của lớp

- Trong biểu đồ lớp, stereotype là cơ chế để phân nhóm lớp
- UML có sẵn nhiều stereotype để sử dụng
- Ba stereotype lớp cơ sở sử dụng trong pha phân tích là
  - Boundary
  - Entity
  - Control

Stereotype của lớp

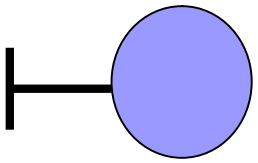
- Ba stereotype lớp cơ sở sử dụng trong pha phân tích là
  - Boundary
    - Dành cho lớp nằm trên biên hệ thống với thế giới còn lại
    - Chúng có thể là form, report, giao diện với phần cứng như máy in, scanner...
    - Khảo sát biểu đồ UC để tìm kiếm lớp biên
  - Entity
  - Control



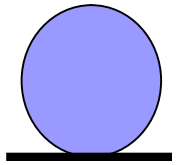
Form

Stereotype của lớp

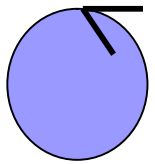
- Ba stereotype lớp cơ sở sử dụng trong pha phân tích là
  - Boundary
  - Entity
    - Lớp thực thể là lớp lưu trữ thông tin sẽ ghi vào bộ nhớ ngoài
    - Thông thường phải tạo ra bảng CSDL cho lớp loại này
  - Control
    - Có trách nhiệm điều phối hoạt động của các lớp khác
    - Thông thường mỗi UC có một lớp điều khiển
    - Nó không thực hiện chức năng nghiệp vụ nào
    - Các lớp điều khiển khác: điều khiển sự kiện liên quan đến an ninh và liên quan đến giao dịch CSDL



BoundaryClass



EntityClass

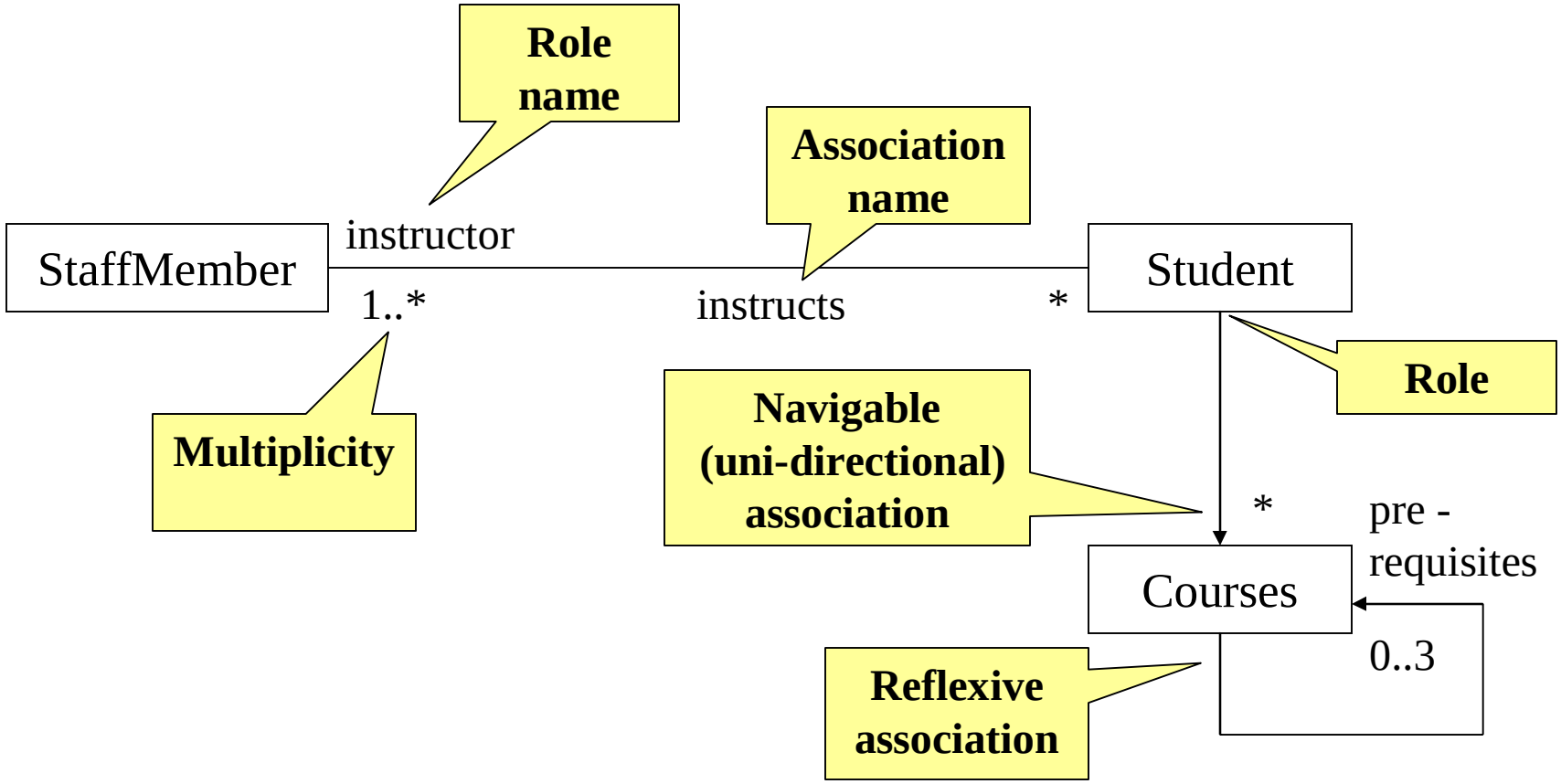


ControlClass

Liên kết

- Một quan hệ ngữ nghĩa giữa hai hoặc nhiều lớp có mối liên hệ với nhau giữa các đối tượng
- Một quan hệ cấu trúc, đặc tả rằng các đối tượng của một lớp kết nối với đối tượng của lớp khác hoặc chính lớp đó.
- Ví dụ: “Một nhân viên làm việc cho một công ty”
- Một Liên kết giữa các lớp chỉ ra rằng đối tượng ở một đầu của liên kết nhận ra đối tượng của đầu kia và có thể gửi thông điệp cho nhau

Liên kết (cont.)



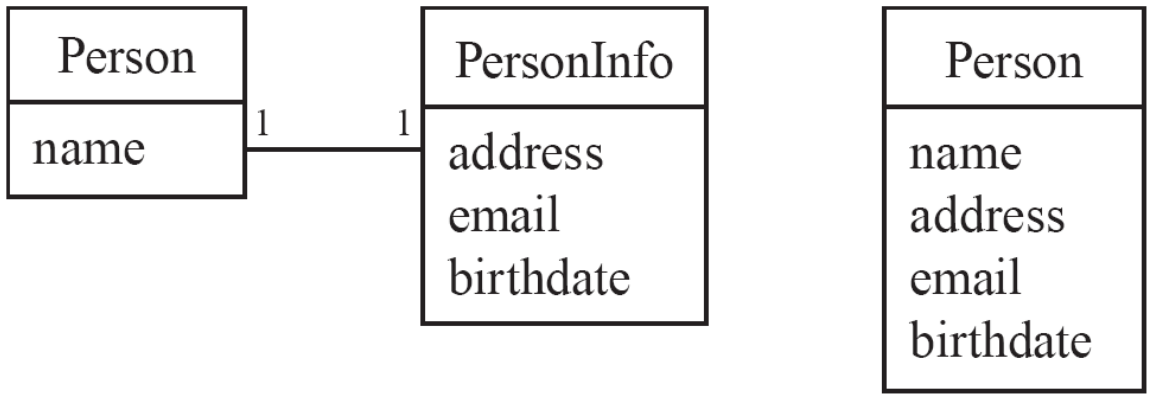
Liên kết (cont.)

- Multiplicity

Chỉ có 1 đối tượng	1
0 hoặc nhiều (unlimited)	* (0..*)
1 hoặc nhiều	1..*
0 hoặc 1 (optional association)	0..1
Khoảng xác định	2..4
Nhiều khoảng	2, 4..6, 8

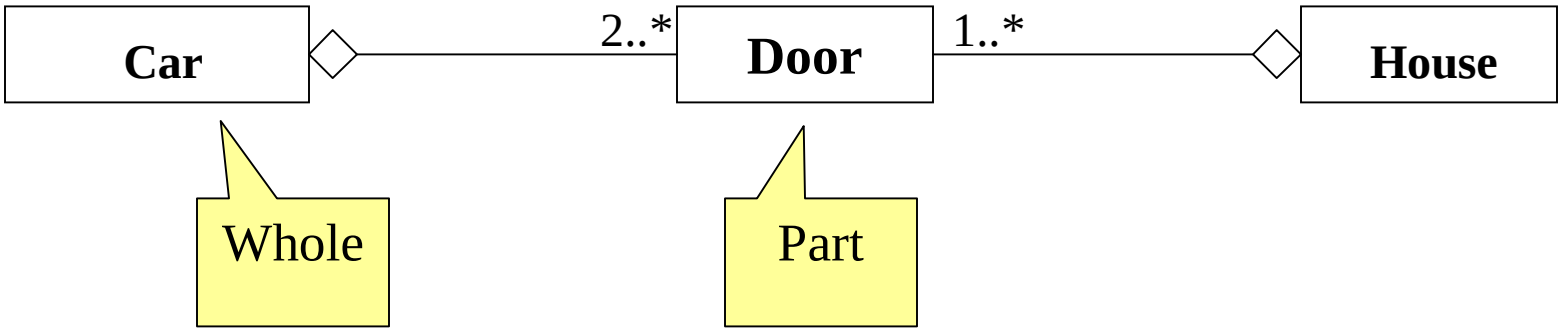
Phân tích và kiểm định quan hệ

- Tránh sử dụng quan hệ 1-1 không cần thiết trong biểu đồ lớp



Kết tập (aggregation)

- Một kiểu đặc biệt của liên kết, dùng để mô hình hóa quan hệ toàn thể - bộ phận giữa một kết tập và bộ phận của nó

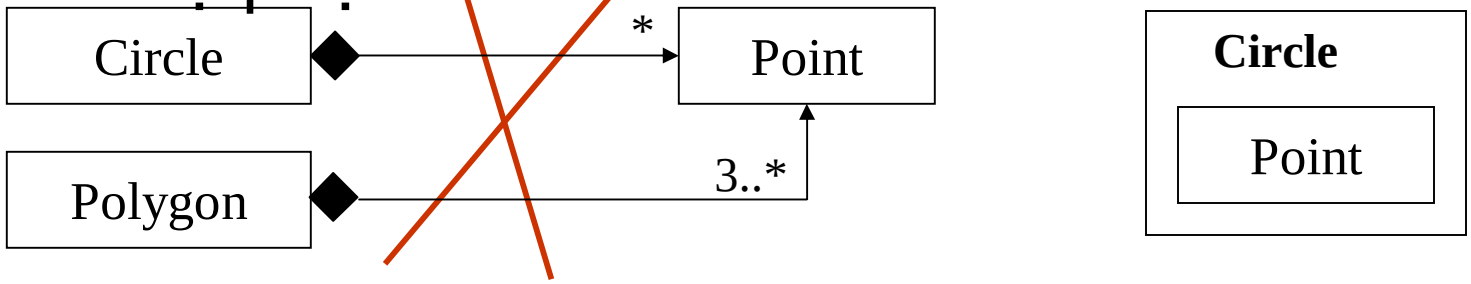


Kết tập (cont.)

- Kiểm tra kết tập:
  - Cụm từ “bộ phận của” (part of) được sử dụng để mô tả quan hệ?
    - Cánh cửa là một bộ phận của xe hơi
  - Có phải một số hành vi của toàn thể được áp dụng tự động cho bộ phận của nó?
    - Xe hơi di chuyển, cửa di chuyển.
  - Có phải một vài giá trị thuộc tính của toàn thể kéo theo một số thuộc tính của bộ phận?
    - Xe hơi màu xanh nên cửa màu xanh.
  - Có tồn tại sự không đảo chiều giữa các lớp cho quan hệ kết tập?
    - Cửa là bộ phận của xe hơi. Xe hơi không là bộ phận của cửa.

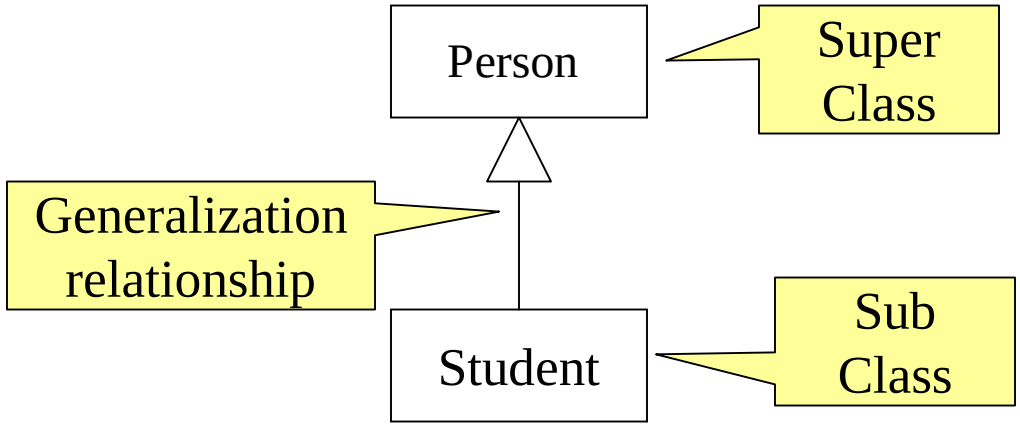
Hợp thành (Composition)

- Một dạng đặc trưng của kết tập
  - Toàn thể là sở hữu duy nhất của bộ phận
  - Số cá thể ở phía lớp toàn thể phải là 0 hoặc 1.
  - Thời gian sống của (lớp) bộ phận phụ thuộc vào (lớp) toàn thể.
  - Toàn thể phải quản lý việc tạo và hủy các bộ phận của nó.



Tổng quát hóa

- Đối tượng của lớp chuyên biệt (lớp con) có thể thay thế bởi các đối tượng của lớp tổng quát (lớp cha).
  - Quan hệ “is a …”.

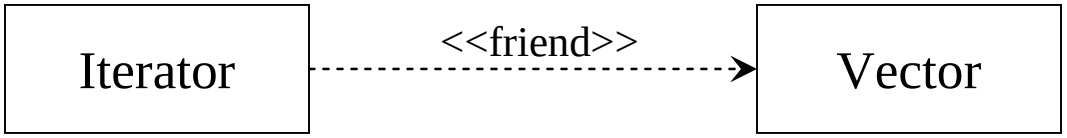


Tổng quát hóa

- Lớp con thừa kế lớp cha:
  - Thuộc tính
  - Phương thức
  - Quan hệ
- Lớp con có thể
  - Thêm thuộc tính và phương thức
  - Thêm quan hệ
  - Ghi đè các phương thức thừa kế

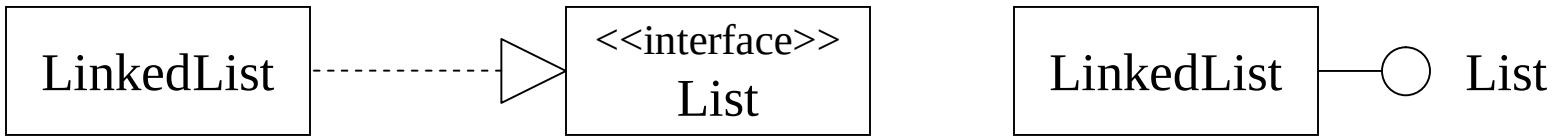
Phụ thuộc

- Sự phụ thuộc chỉ ra một quan hệ ngữ nghĩa giữa hai hoặc nhiều lớp trong đó sự thay đổi của lớp này bắt buộc sự thay đổi của lớp khác mặc dù giữa chúng không có một sự liên kết rõ ràng



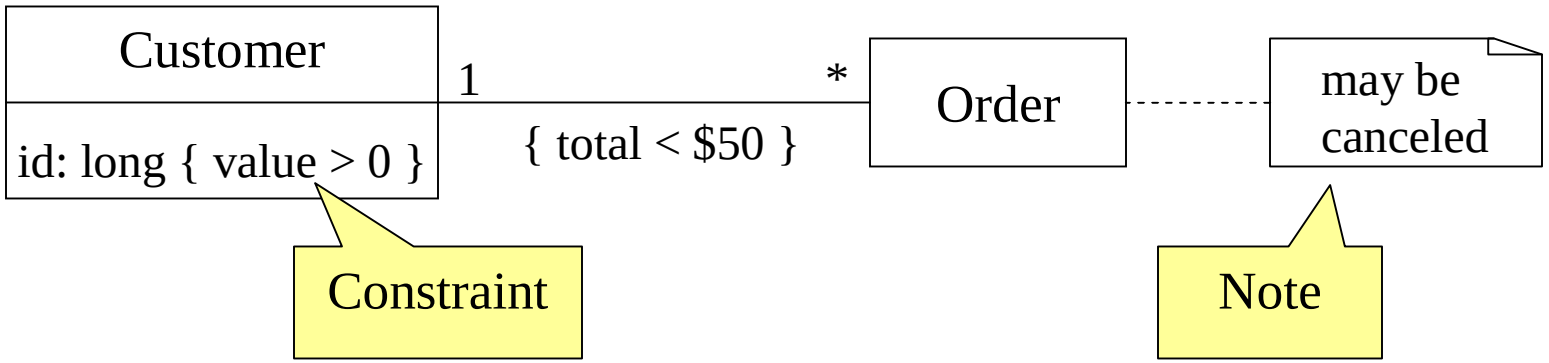
Thực hiện (Realization)

- Một quan hệ thực hiện chỉ ra một lớp thực thi hành vi đặc tả bởi một lớp khác (thường là một giao diện)
- Một giao diện có thể được thực thi bởi nhiều lớp
- Một lớp có thể thực thi nhiều giao diện



Các ràng buộc và ghi chú

- **Ràng buộc** và **chú thích** các liên kết, thuộc tính, phương thức và các lớp
- Các ràng buộc là các hạn chế ngữ nghĩa được viết dưới dạng biểu thức Boolean



Tips

- Không cố gắng sử dụng tất cả các ký hiệu khác nhau
- Không vẽ mô hình cho mọi thứ, tập trung vào các thông tin quan trọng

Biểu đồ lớp của hệ thống quản lý thư viện

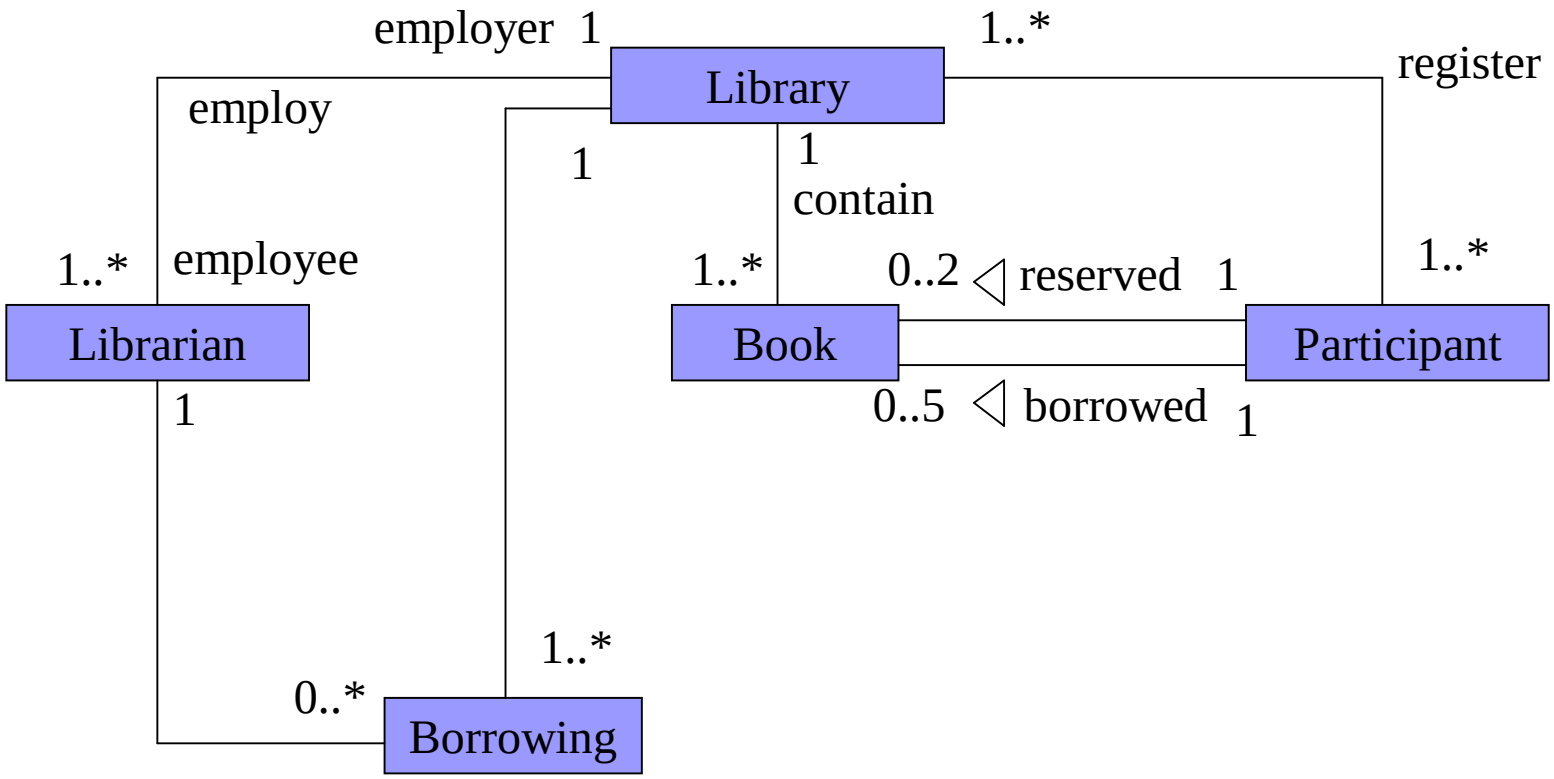
Các giai đoạn của mô hình hóa đối tượng

- Tìm kiếm các lớp
- Xác định liên kết giữa các lớp
- Xác định các thuộc tính
- Tổ chức và đơn giản hóa các lớp bằng cách sử dụng quan hệ thừa kế
- Xóa các liên kết thừa
- Kiểm tra xem biểu đồ đã bao gồm tất cả các yêu cầu của tài liệu hay chưa?
- Lặp lại và làm mịn mô hình
- Nhóm các lớp thành các modules (gói)

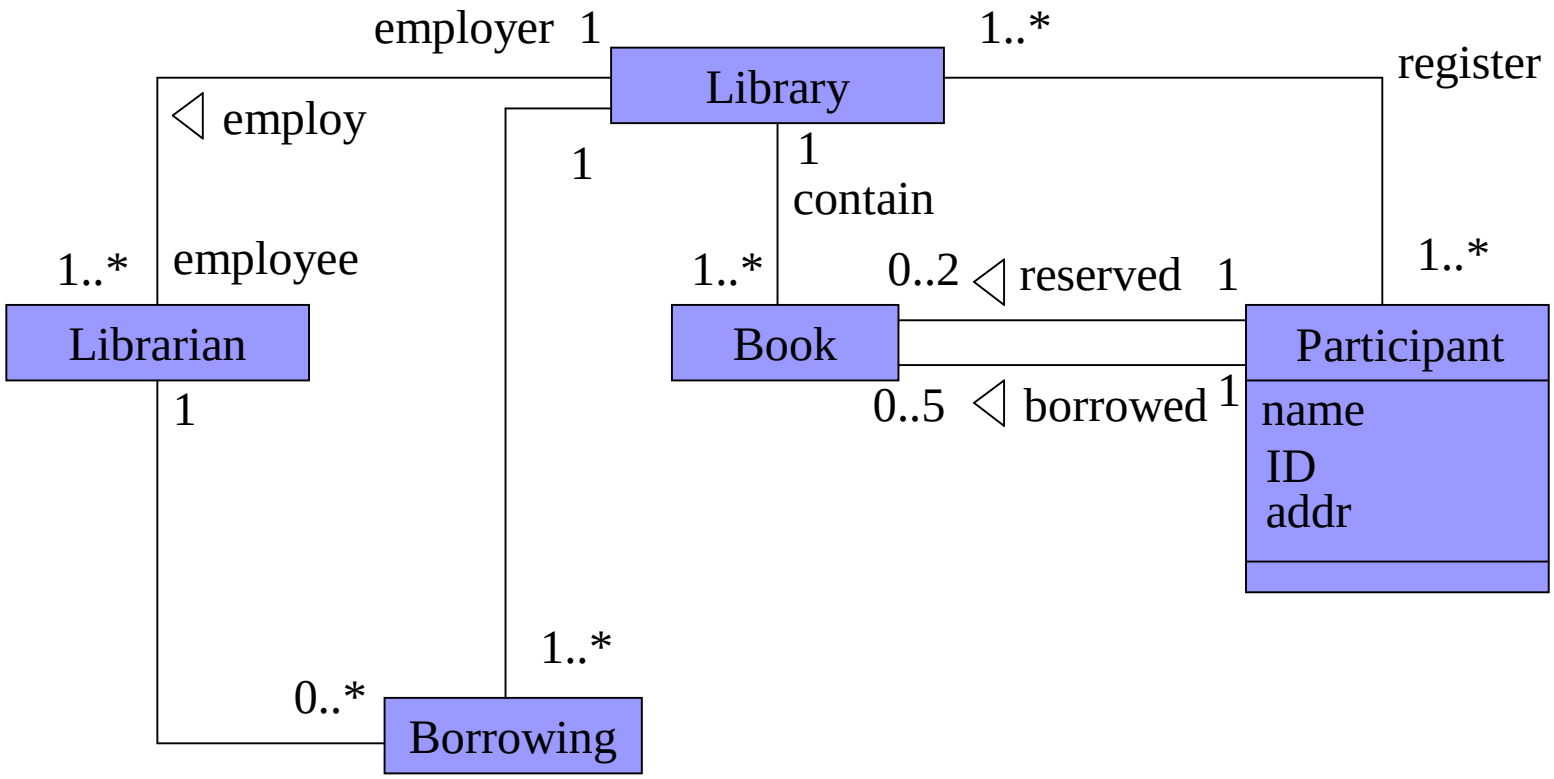
Xác định các lớp

- Người quản lý thư viện mong muốn tự động hóa việc mượn sách
- Họ yêu cầu một phần mềm cho phép người sử dụng biết sách hiện có, có thể đặt mượn 2 quyển sách, những người tham gia mượn sách có thể biết sách nào đã mượn hoặc đã đặt
- Những người tham gia mượn sách sở hữu một password để truy nhập
- Việc mượn sách được thực hiện bởi các thủ thư, sau khi xác định người mượn sách, họ biết được người này có được phép mượn hay không? (tối đa 5 quyển), người này được ưu tiên? (đã đặt trước)

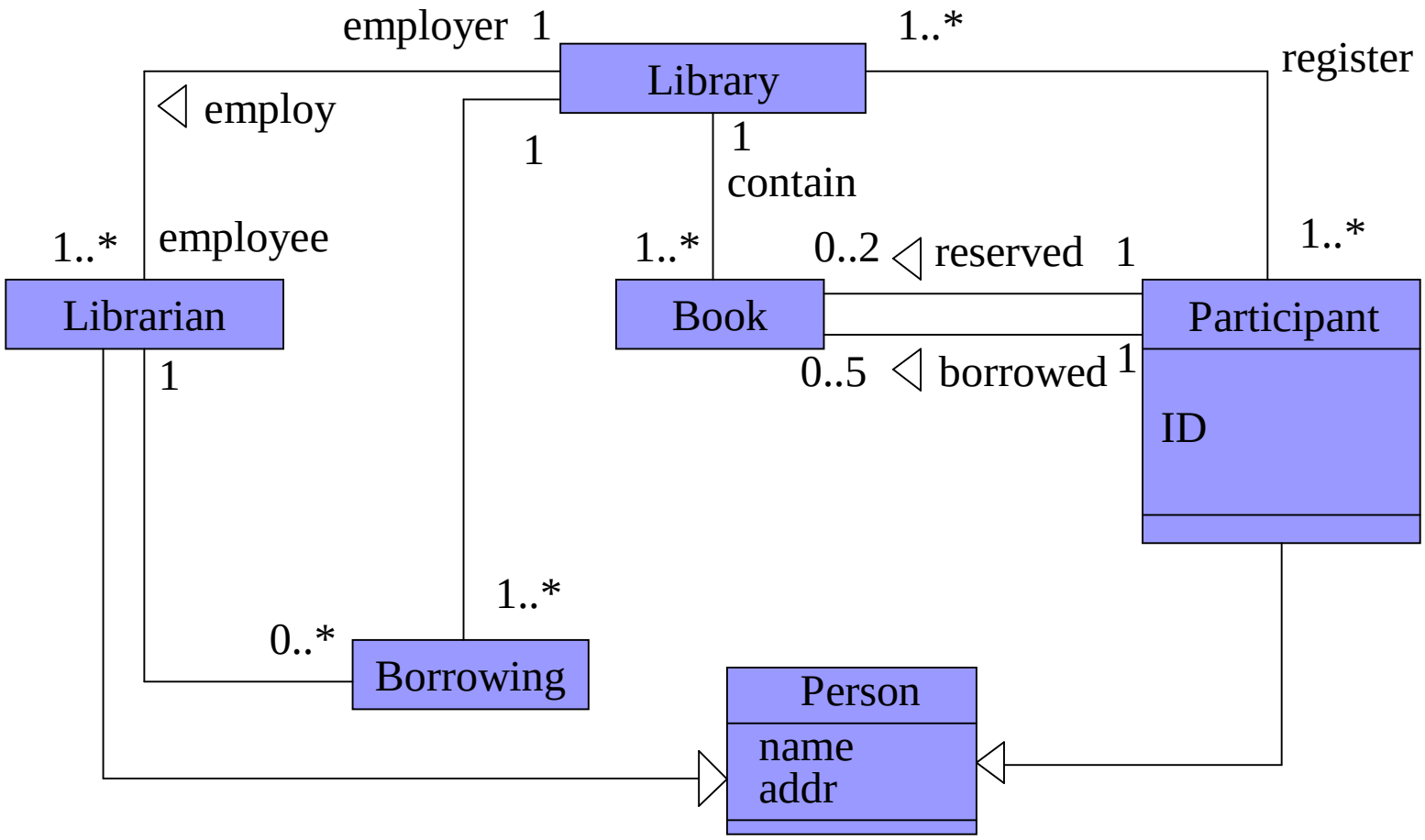
Xác định các liên kết



Xác định các thuộc tính



Tổng quát hóa bằng thừa kế



Bài tập

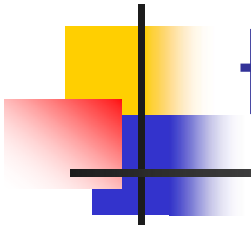
- **Câu 1**. Xác định quan hệ giữa các lớp: Keyboard, Mouse, Laptop và Desktop.
- **Câu 2**. Xác định quan hệ giữa các lớp: Person, StaffMember và Company. Nếu hệ thống có thêm lớp University thì quan hệ này sẽ thay đổi thế nào (StaffMember là bộ phận của Company và cũng là bộ phận của University).
- **Câu 3**. Xác định quan hệ giữa các lớp: Car, Door và Truck.

Mô hình hóa tương tác đối tượng

Truong Ninh Thuan

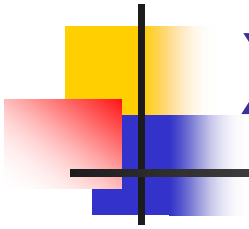
Mô hình hóa tương tác đối tượng

- Hai loại biểu đồ được sử dụng để mô hình hóa tương tác đối tượng
  - Biểu đồ trình tự (Sequence diagrams)
  - Biểu đồ cộng tác (Collaboration diagrams)
  - Biểu đồ trình tự và biểu đồ cộng tác đều chỉ ra cùng loại thông tin, còn gọi là biểu đồ tương tác (Interaction diagram)
  - Biểu đồ tương tác giúp xác định hệ thống làm việc như thế nào?



Đối tượng?

- Đối tượng là các sự vật xung quanh
- Thí dụ Máy bay VN358 là đối tượng
  - Có các thông tin
  - Có các hành vi
- Thông tin được lưu trữ bởi thuộc tính (Attribute)
- Hành vi của đối tượng được gọi là thao tác (Operation)



Xây dựng biểu đồ tương tác

- Bắt đầu từ luồng sự kiện
- Các bước xây dựng biểu đồ tương tác
  - Tìm kiếm đối tượng
  - Tìm kiếm tác nhân
  - Bổ sung thông điệp vào biểu đồ

Tìm kiếm đối tượng

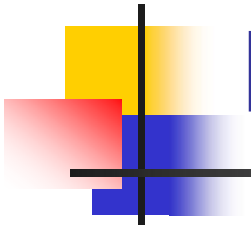
- Có thể hình thành các biểu đồ tương tác
  - Ở mức cao: để chỉ ra hệ thống giao tiếp như thế nào
  - Ở mức rất thấp: để chỉ ra lớp nào cần tham gia vào kịch bản
- Nên xem xét các nhóm đối tượng sau khi tìm kiếm chúng
  - Đối tượng thực thể (Entity)
  - Đối tượng biên (Boundary)
  - Đối tượng điều khiển (Control)

Tìm kiếm tác nhân

- Tác nhân trong biểu đồ tương tác là sự kích hoạt từ ngoài để khởi động luồng công việc của luồng sự kiện
- Tìm kiếm tác nhân trong luồng sự kiện
- Có thể có nhiều tác nhân cho một biểu đồ tương tác
- Nếu tác nhân nhận hay gửi thông điệp cho hệ thống theo kịch bản nào đó thì chúng phải có mặt trong biểu đồ tương tác của kịch bản đó

Xây dựng biểu đồ tương tác

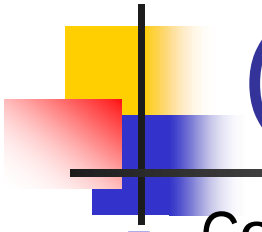
- Biểu đồ tương tác (Interaction diagrams) bao gồm các thành phần sau
  - Đối tượng (Objects)
  - Thông điệp (Messages)
  - Liên kết (Links)
  - Chú thích (Notes) và ràng buộc



Biểu đồ tuần tự

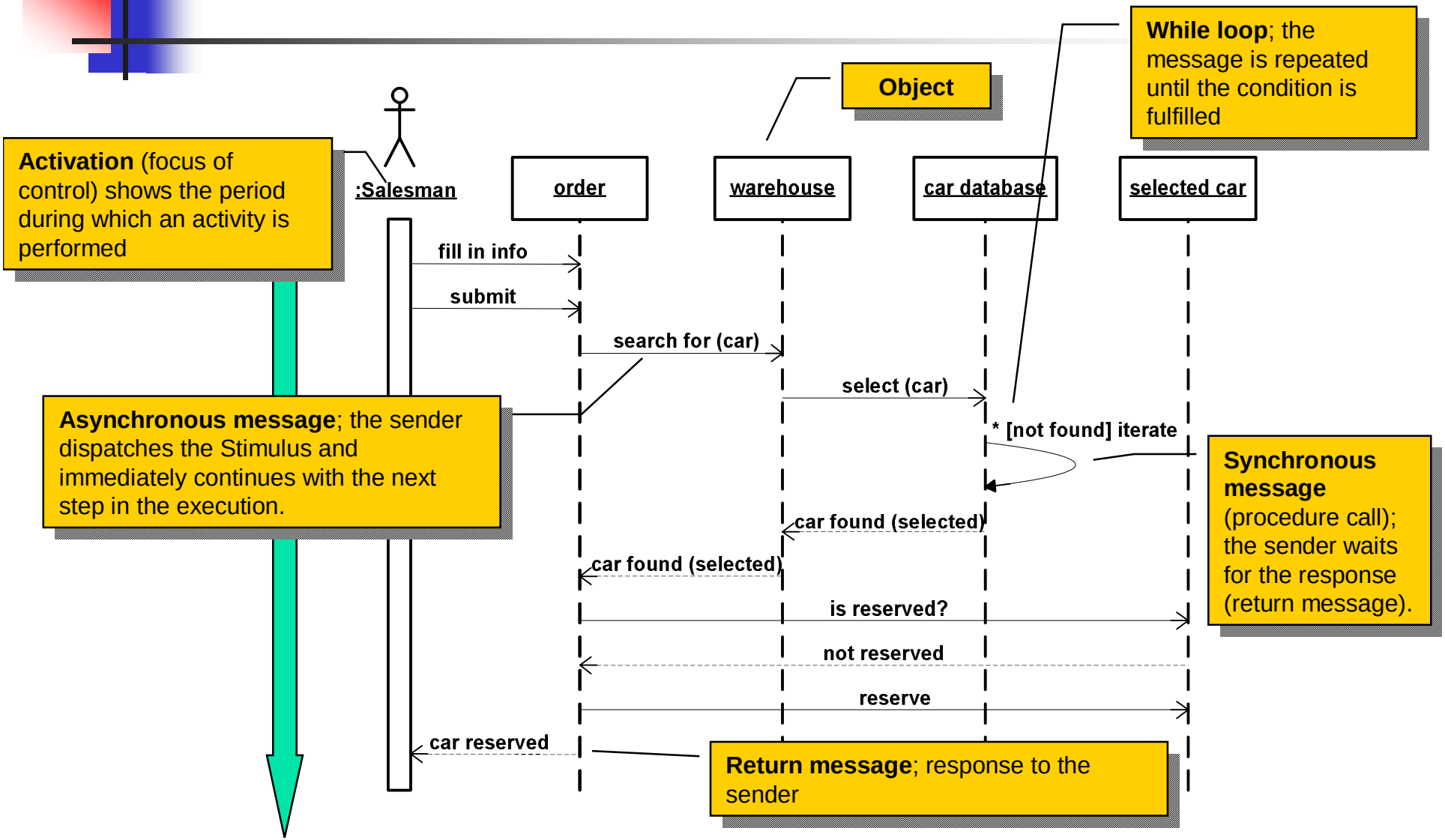
- Biểu đồ tuần tự là biểu đồ theo thứ tự thời gian
  - Đọc biểu đồ từ đỉnh xuống đáy
  - Mỗi đối tượng có vòng đời (Lifeline)
    - Bắt đầu khi hình thành đối tượng, kết thúc khi phá hủy đối tượng
    - Thông điệp được vẽ giữa hai đối tượng – thể hiện đối tượng gọi phương thức của đối tượng khác
    - Thông điệp phản thân

Biểu đồ cộng tác (Collaboration diagrams)

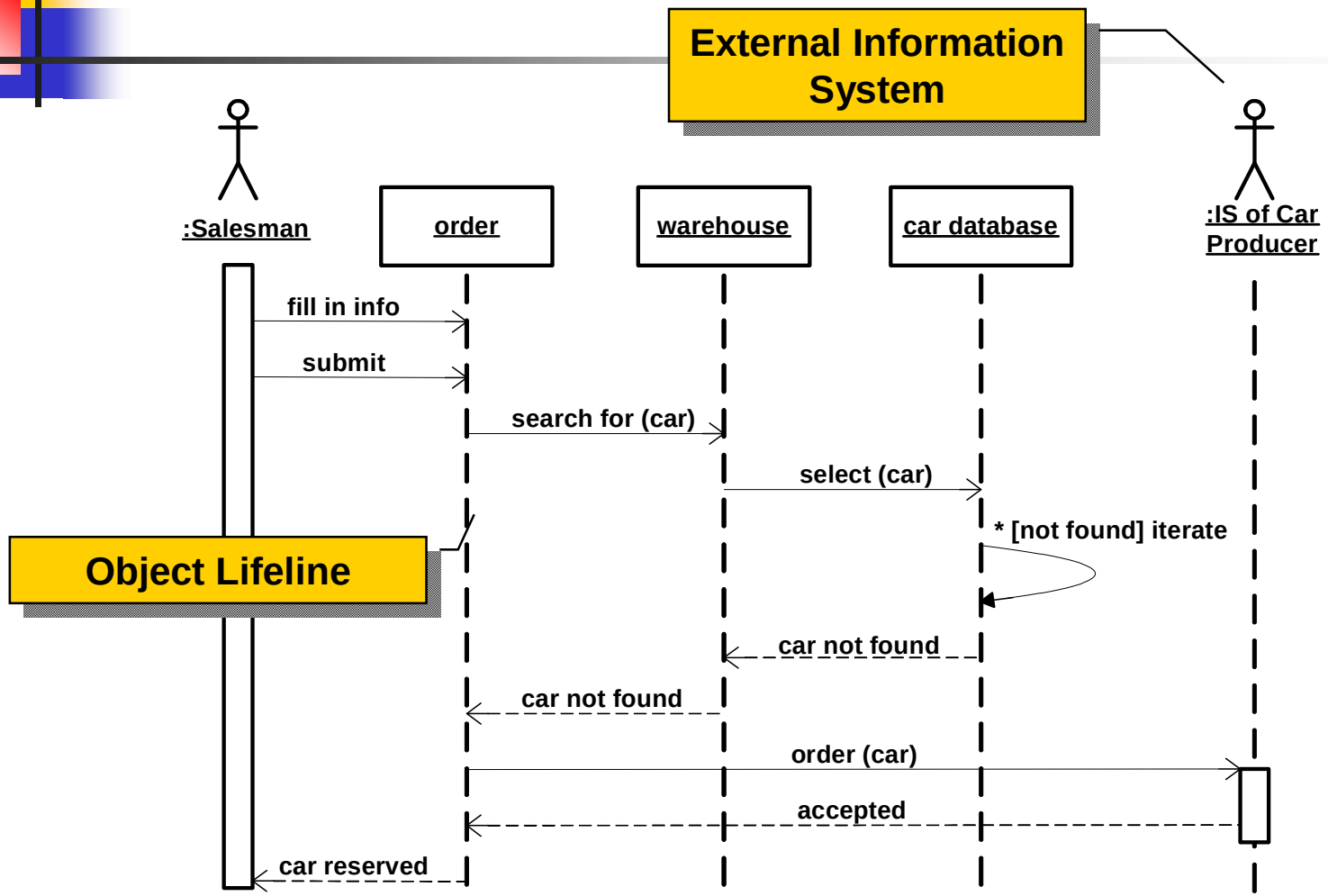


- Communication diagram (UML 2.0)
- Mô tả việc truyền thông điệp giữa các lớp và định nghĩa các liên kết
- Tương đồng về mặt ngữ nghĩa với biểu đồ tuần tự
  - Biểu đồ tuần tự chú trọng về thứ tự, thời gian các thông điệp
  - Biểu đồ tương tác biểu diễn thứ tự, quan hệ giữa các đối tượng
- Class roles: Đối tượng tham gia tương tác
- Link: Một thực thể của liên kết
- Messages: Gửi theo các links

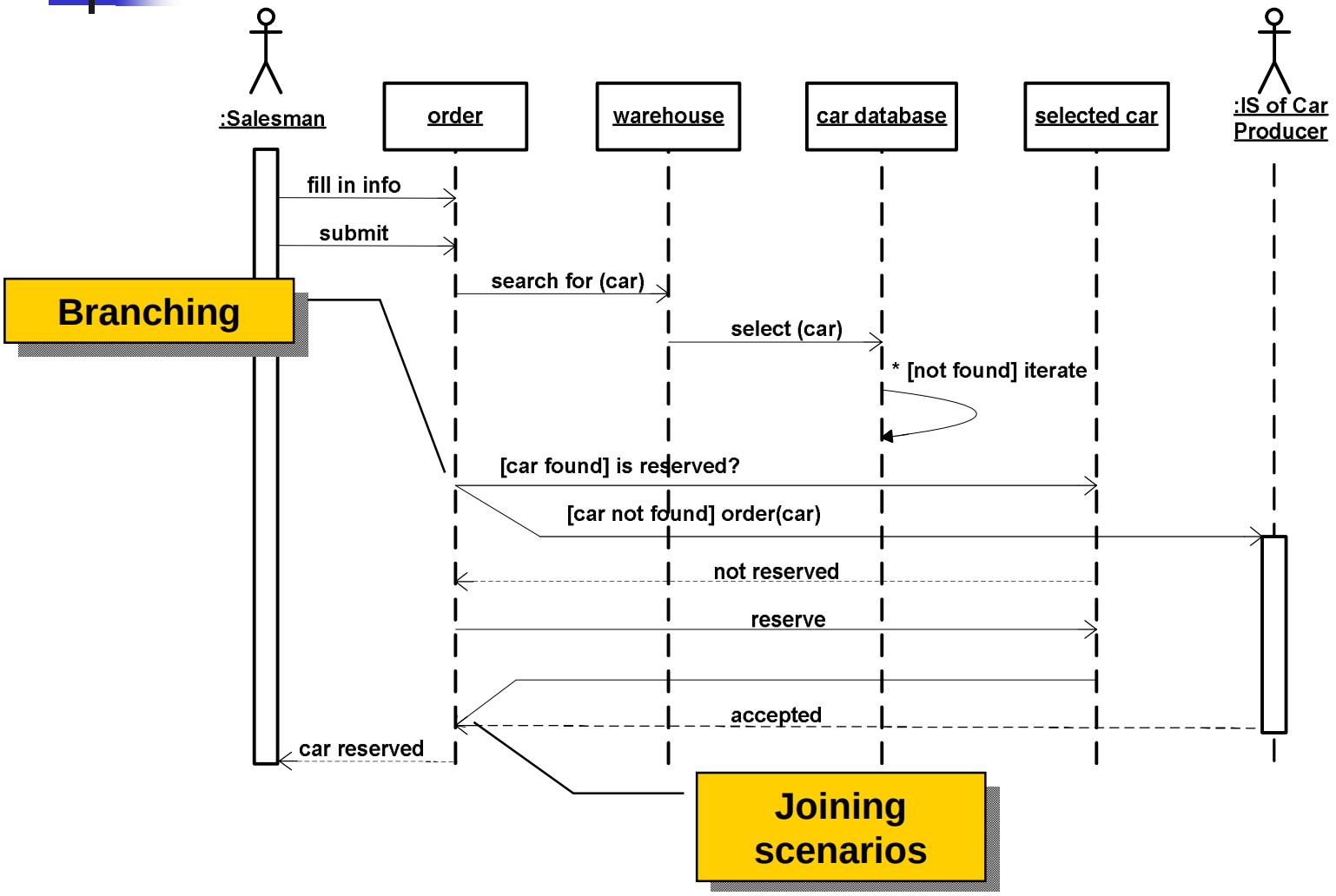
Biểu đồ tuần tự: Car Ordering



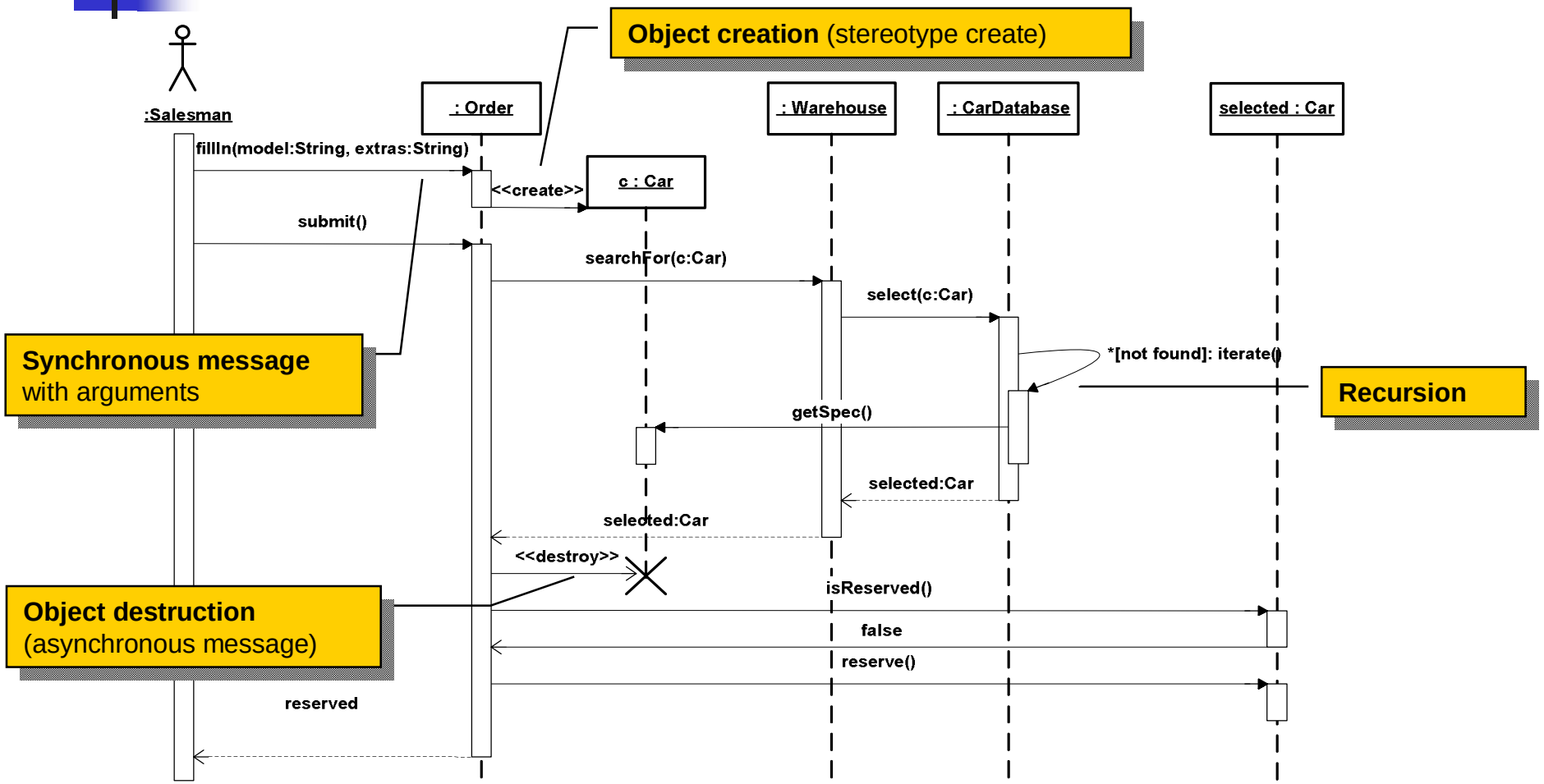
Kịch bản lựa chọn



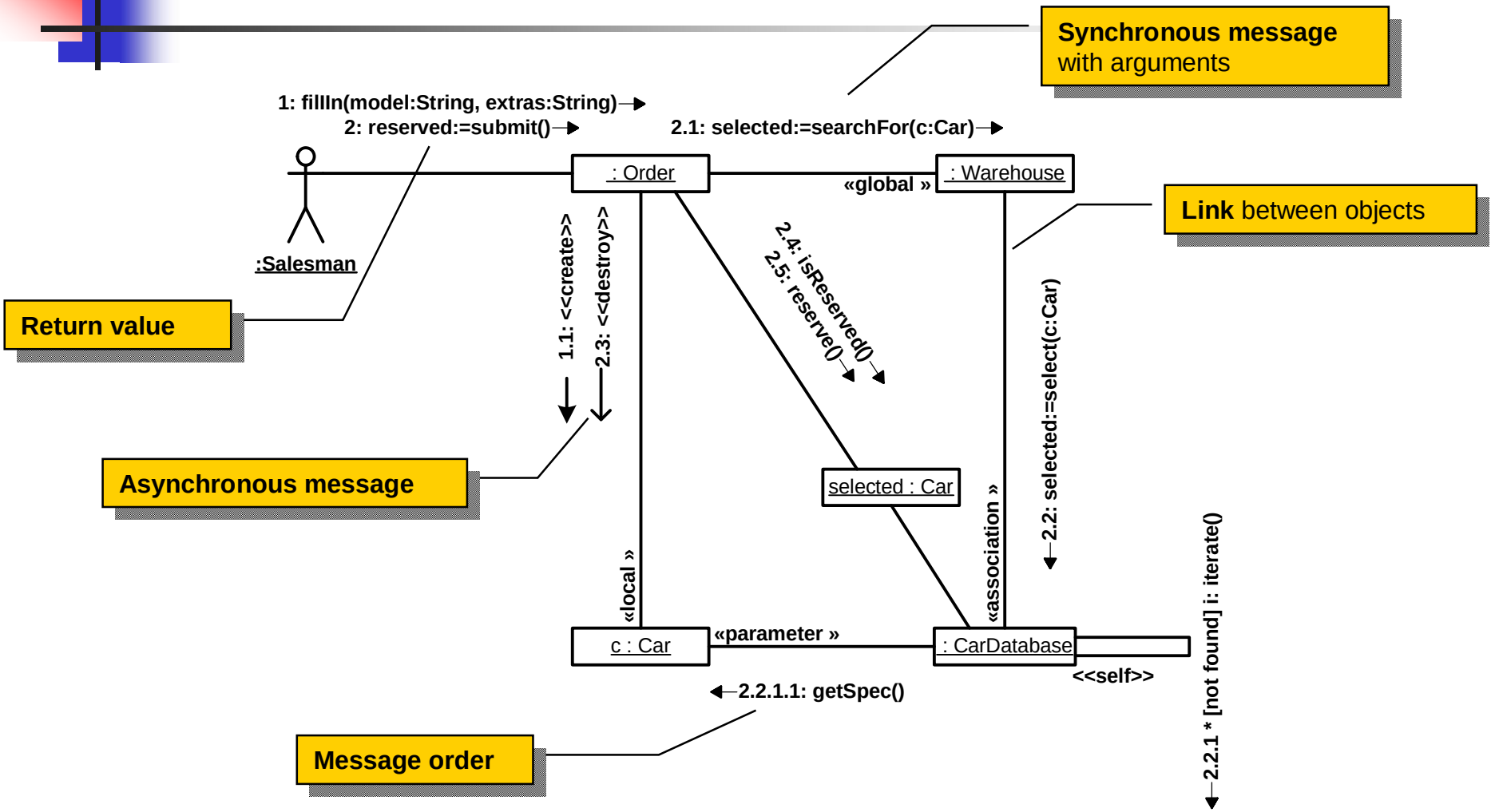
Hợp nhất các kịch bản



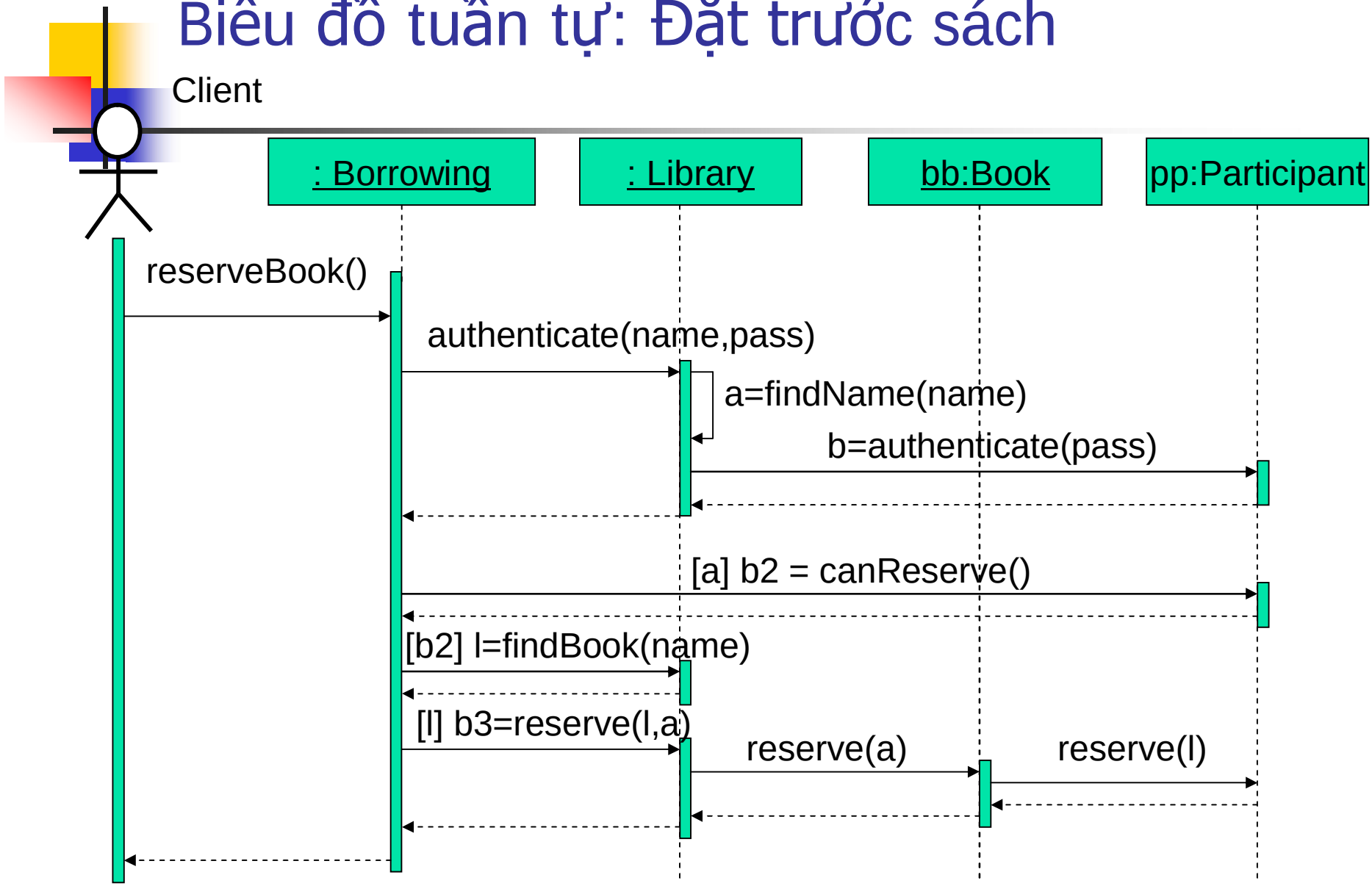
Làm mịn biểu đồ tuần tự



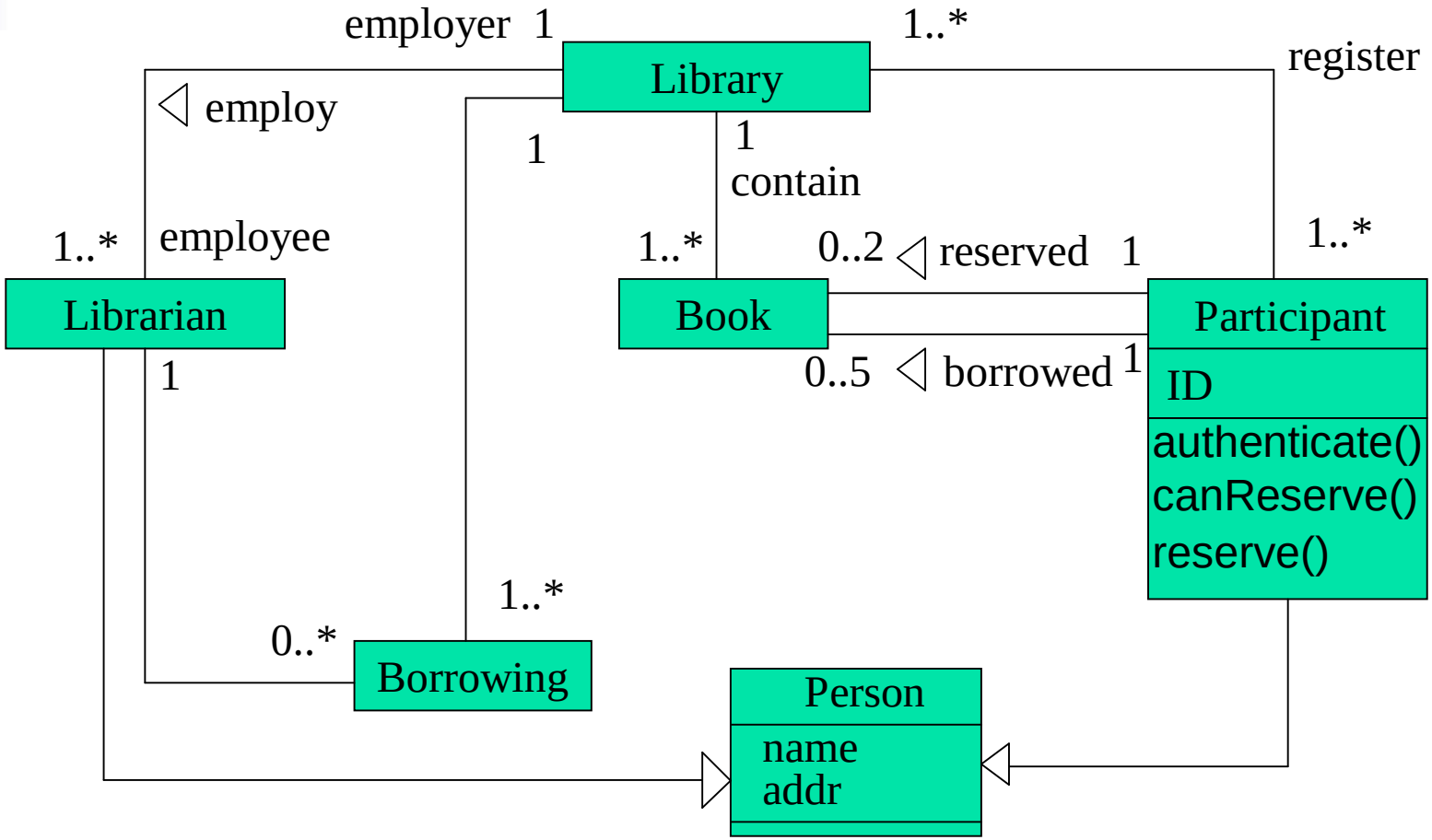
Biểu đồ cộng tác



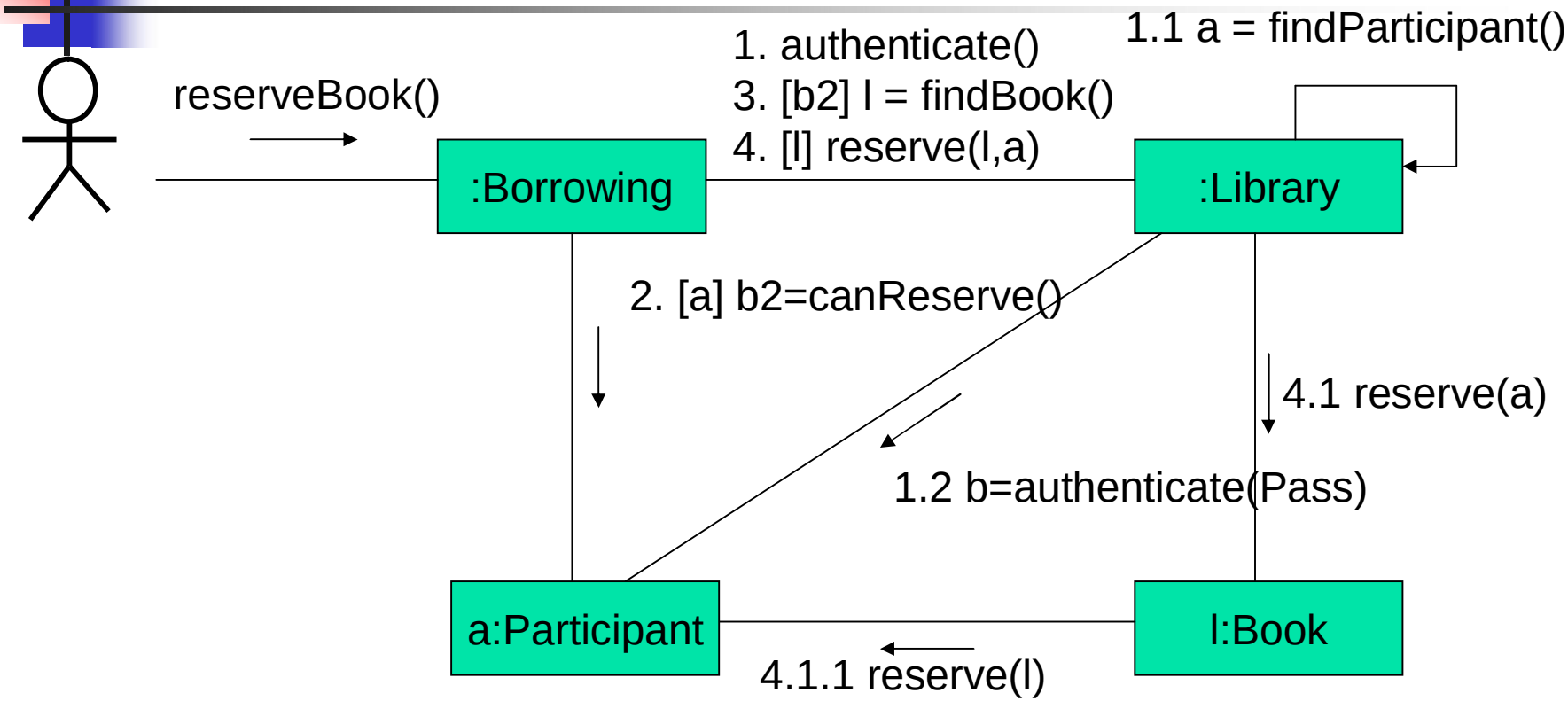
Biểu đồ tuần tự: Đặt trước sách



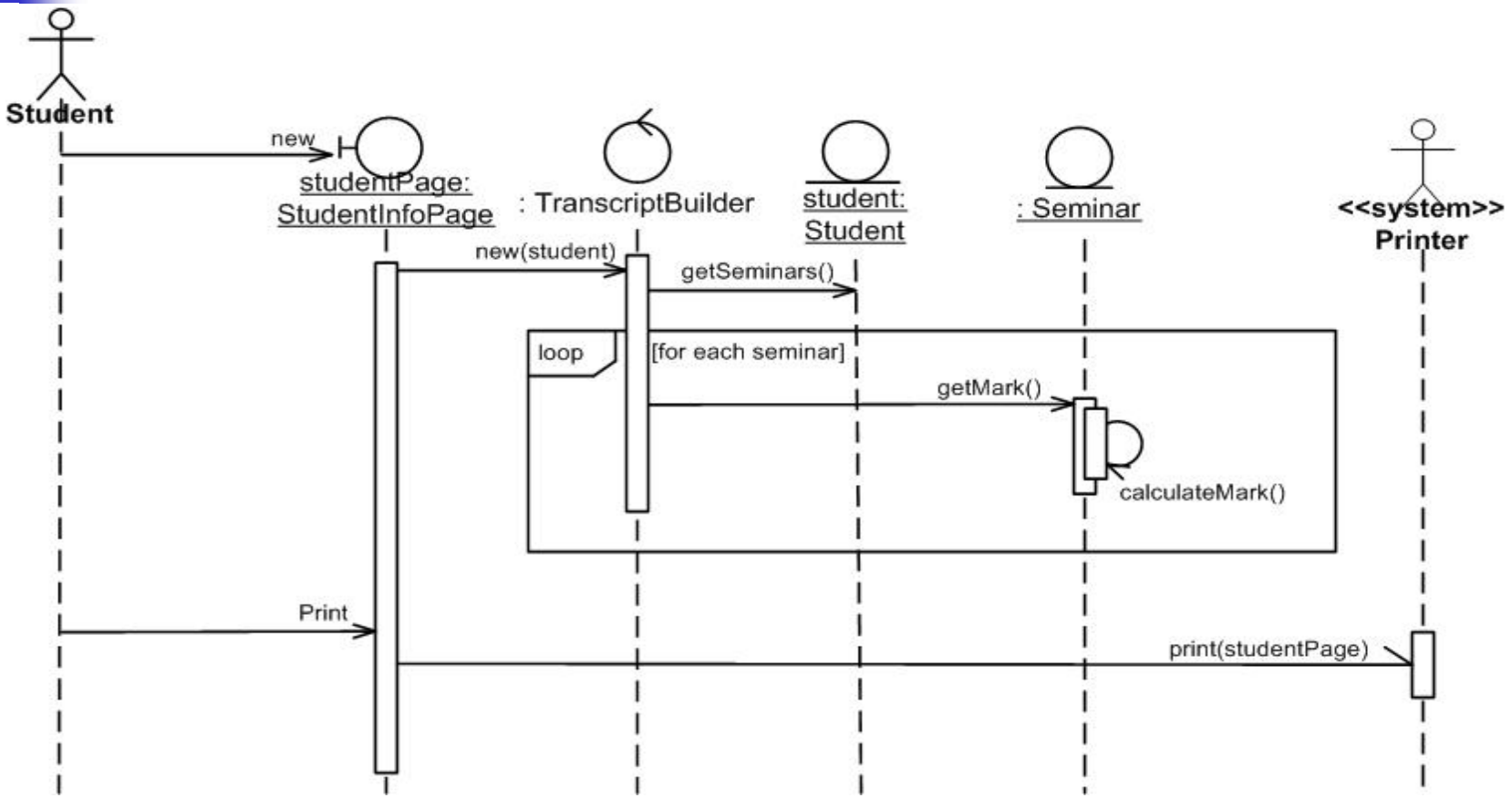
Thêm các phương thức vào các lớp

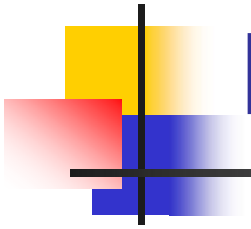


Biểu đồ cộng tác: Đặt sách



Enrolling in a seminar





Bài tập

- Bài 1. Vẽ biểu đồ tuần tự của kịch bản in một file ra máy in
- Bài 2. Chuyển biểu đồ tuần tự trên sang biểu đồ tương tác

Biểu đồ trạng thái
State diagrams

Trạng thái đối tượng

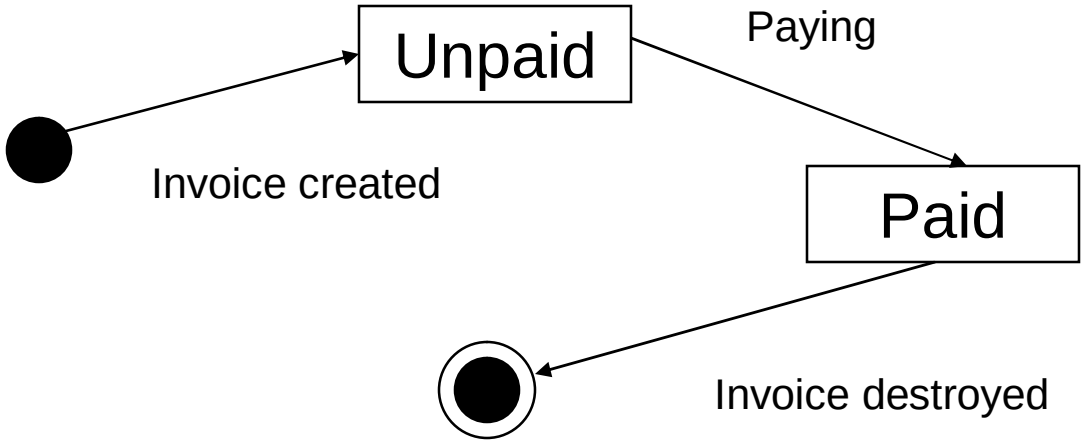
- Trạng thái đối tượng là kết quả của các hoạt động trước đó của đối tượng
- Đối tượng luôn ở trong một trạng thái xác định tại một thời điểm
- Thí dụ
  - Con người cụ thể của lớp Person có các trạng thái: Lao động, Thất nghiệp, Về hưu
  - Hóa đơn mua hàng: Đã thanh toán, chưa thanh toán
  - Xe ô tô: Đang chạy, Đang đứng
- Thay đổi trạng thái đối tượng
  - Có sự kiện xảy ra

Biểu đồ trạng thái

- Mô tả chu kỳ tồn tại của đối tượng từ khi nó sinh ra đến khi nó bị phá hủy
- Sử dụng để mô hình hóa khía cạnh động của lớp
- Biểu đồ bao gồm các thông tin sau
  - Các trạng thái của đối tượng
  - Hành vi của đối tượng
  - Sự kiện tác động làm thay đổi trạng thái
- Thông thường
  - Xây dựng biểu đồ chuyển trạng thái cho một vài đối tượng của lớp có nhiều hành vi động trong dự án
  - Không phải mọi dự án sử dụng biểu đồ loại này

Biểu đồ trạng thái

- Thí dụ biểu đồ trạng thái



- Biểu đồ trạng thái dùng để
  - Phân tích viên, người thiết kế và người sử dụng hiểu hành vi đối tượng
  - Người phát triển hiểu hành vi đối tượng để cài đặt nó

Biểu đồ trạng thái

- Các phần tử đồ họa
  - Trạng thái khởi đầu: Khi đối tượng được tạo ra

  - Trạng thái kết thúc: Khi đối tượng bị phá hủy

Trạng thái

- Trạng thái có các hành động kết hợp sau:
  - **OnEntry**/ các hành động này sẽ thực hiện khi trạng thái đạt tới
  - **Do**/ các hành động này thực hiện trong lúc trạng thái tồn tại
  - **OnEvent**/ các hành động này thực hiện để phản ứng lại một sự kiện
  - **OnExit**/ các hành động thực hiện khi thoát khỏi trạng thái

Chuyển trạng thái

- Chuyển trạng thái là bước chuyển từ trạng thái này sang trạng thái khác
- Chuyển trạng thái là một bộ ba:

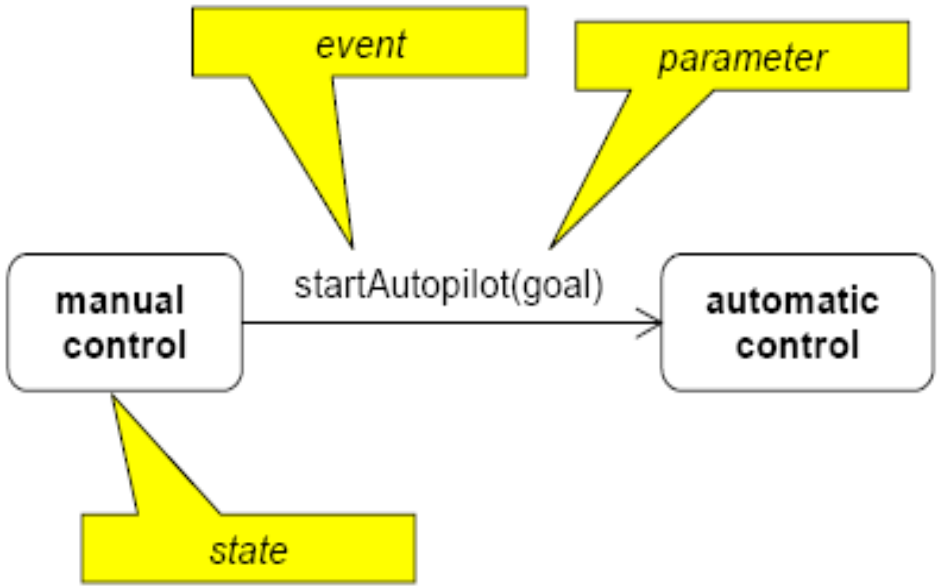
  Event[Condition]/Action

  các thành phần này đều không bắt buộc có

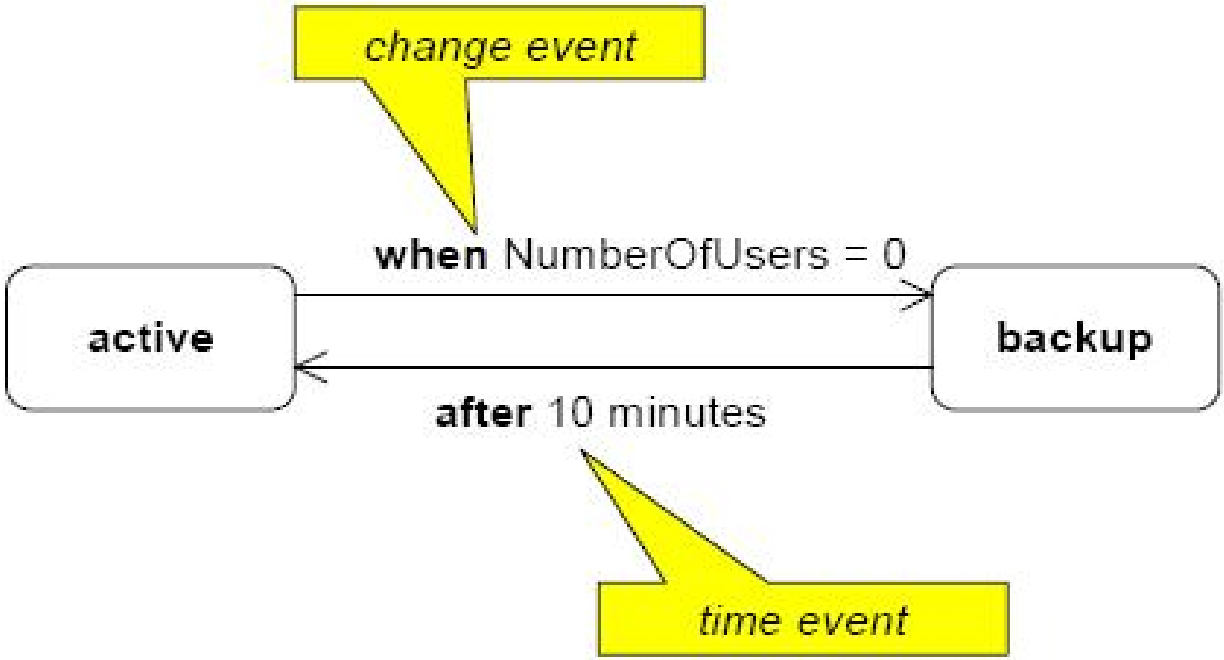
Sự kiện

- **Sự kiện** là nguyên nhân của chuyển trạng thái
- Một sự kiện có thể kích hoạt một hoặc nhiều hành động bởi một tác nhân
- Các kiểu sự kiện trong UML:
  - **Change events** xuất hiện khi điều kiện thỏa mãn
  - **Signal events** chỉ ra việc nhận một tín hiệu ngoài từ một đối tượng (hoặc tác nhân) sang đối tượng khác
  - **Call events** chỉ ra việc nhận một lời gọi hàm bởi một đối tượng hoặc tác nhân
  - **Time events** đánh dấu việc chuyển trạng thái sau một khoảng thời gian

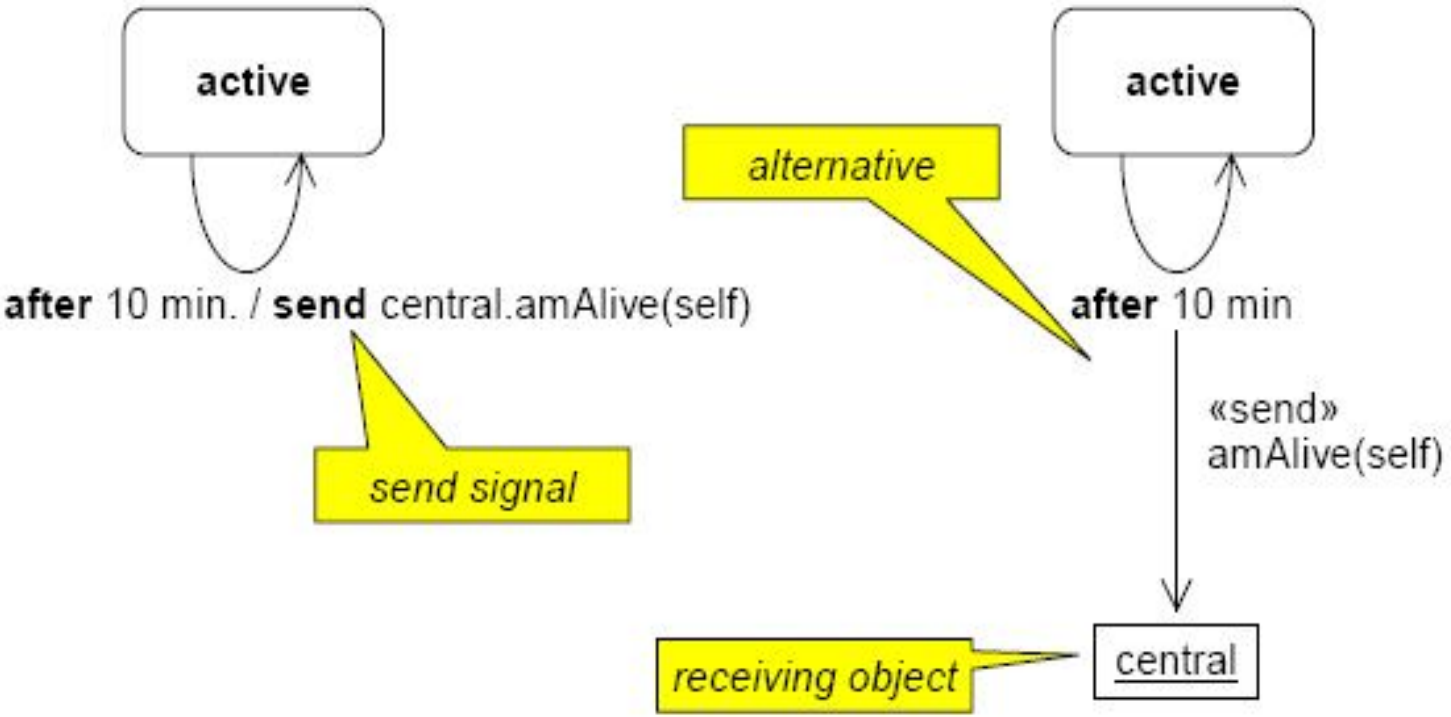
Call events


event
parameter
manual control
startAutopilot(goal)
automatic control
state

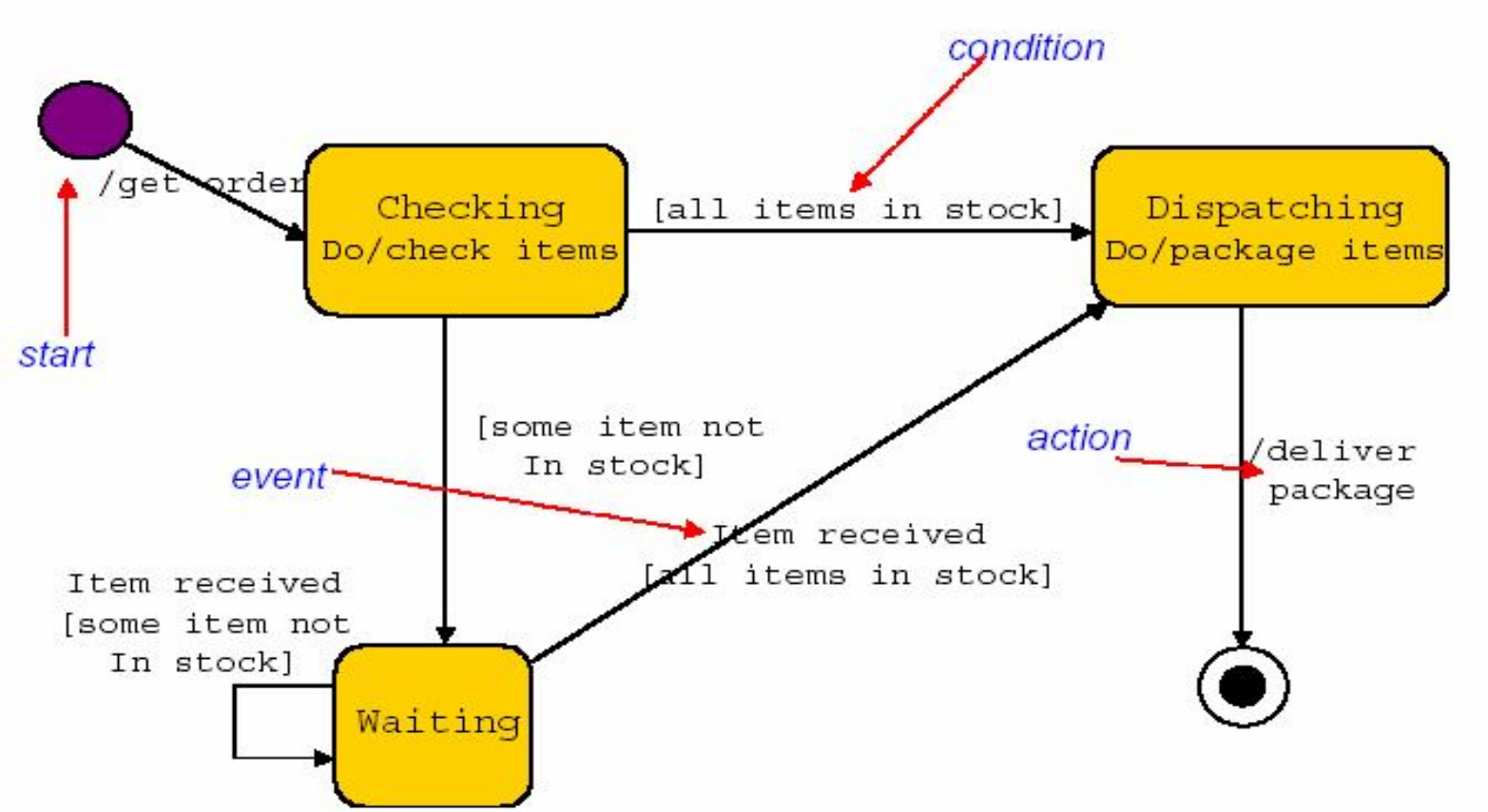
Change and time events



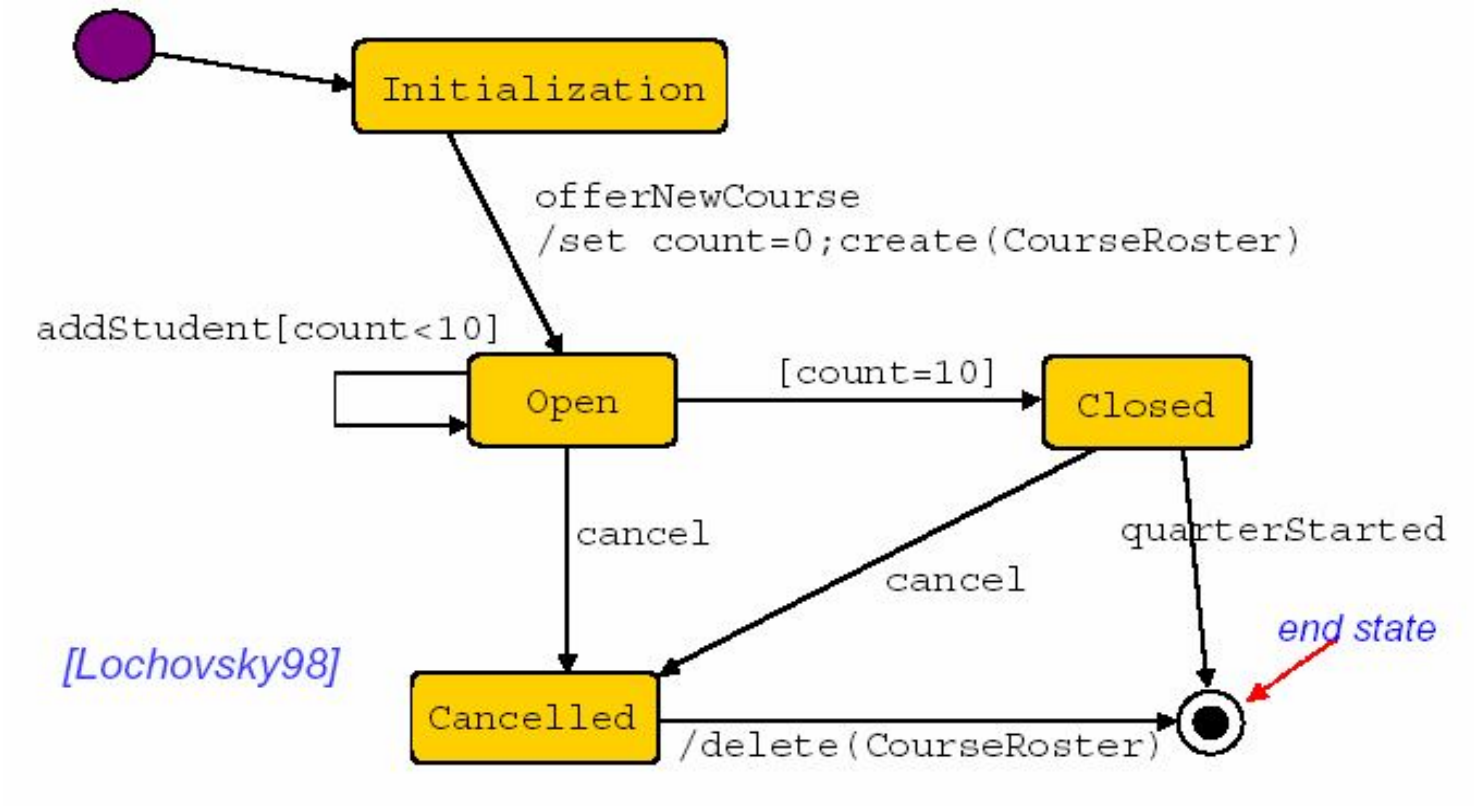
Sending signals



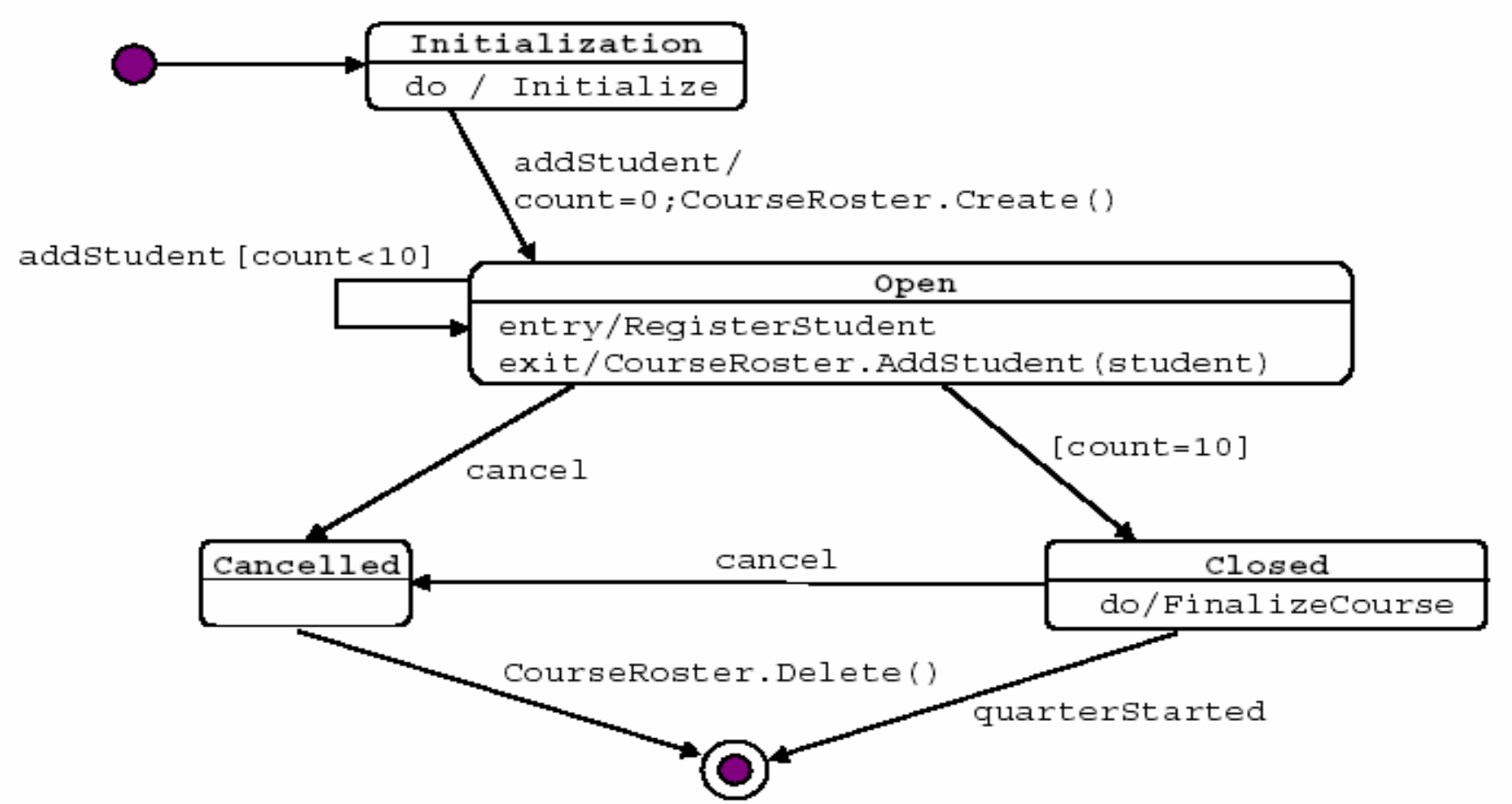
Biểu đồ trạng thái: Đặt mua hàng Purchase order



Biểu đồ trạng thái - Course



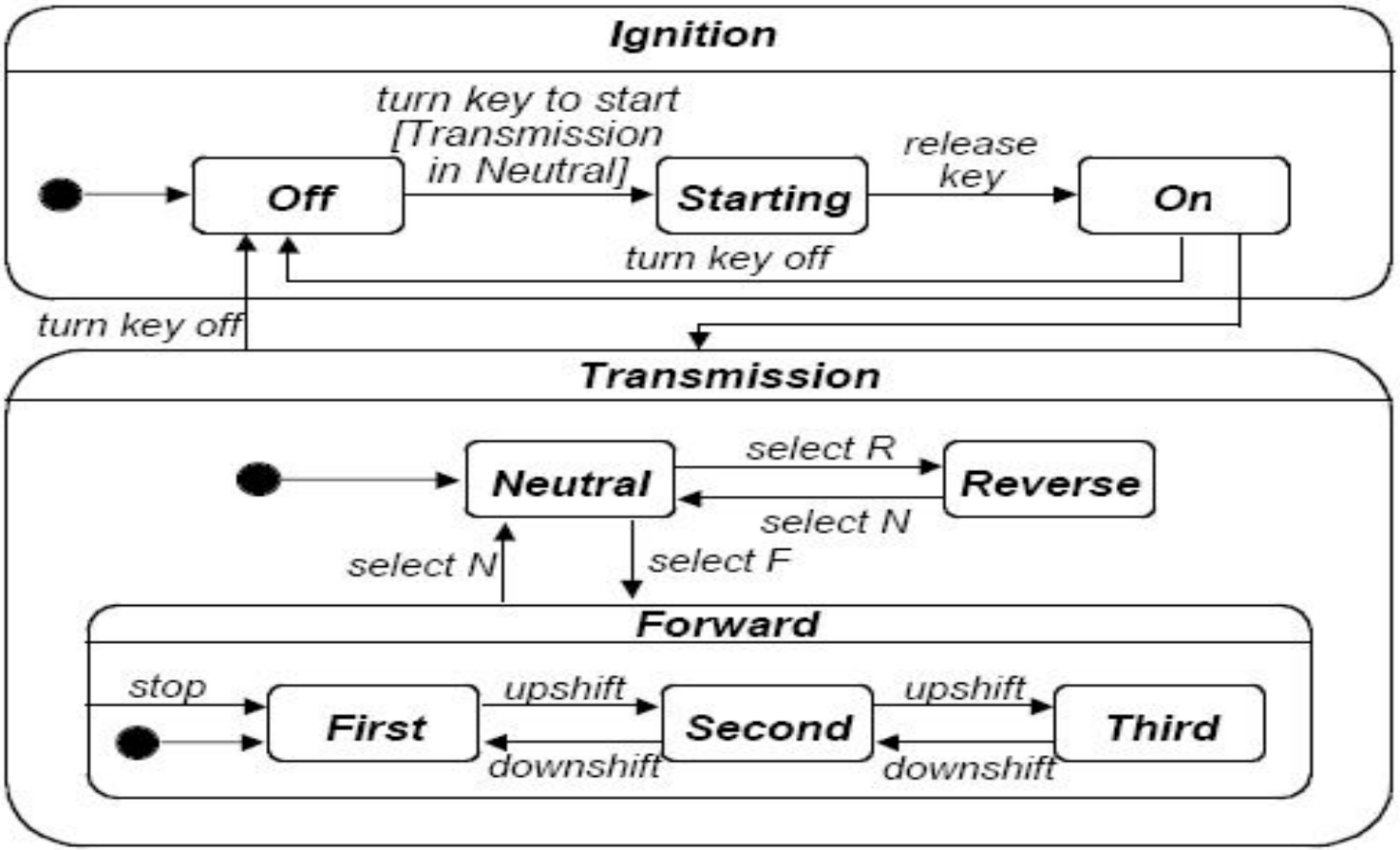
Biểu đồ trạng thái – Course (cont.)



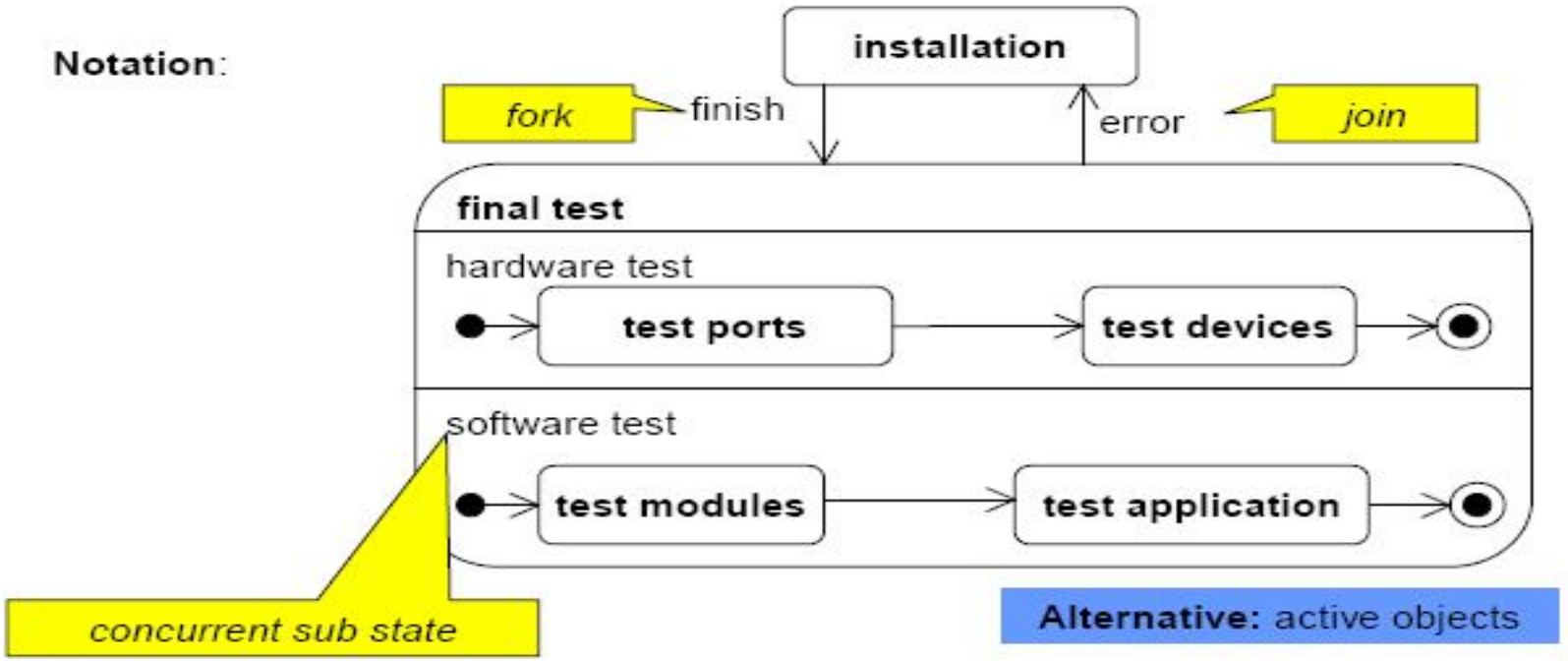
Trạng thái cha - Superstates

- Để giảm quá nhiều trạng thái trong biểu đồ ta có thể lồng trạng thái vào trong trạng thái khác
  - Trạng thái con (Substate), trạng thái cha (Superstate)
- Sự kết hợp này cho phép UML biểu diễn biểu đồ trạng thái theo các mức trừu tượng khác nhau
- Trạng thái cha chứa trong nó một hoặc nhiều trạng thái

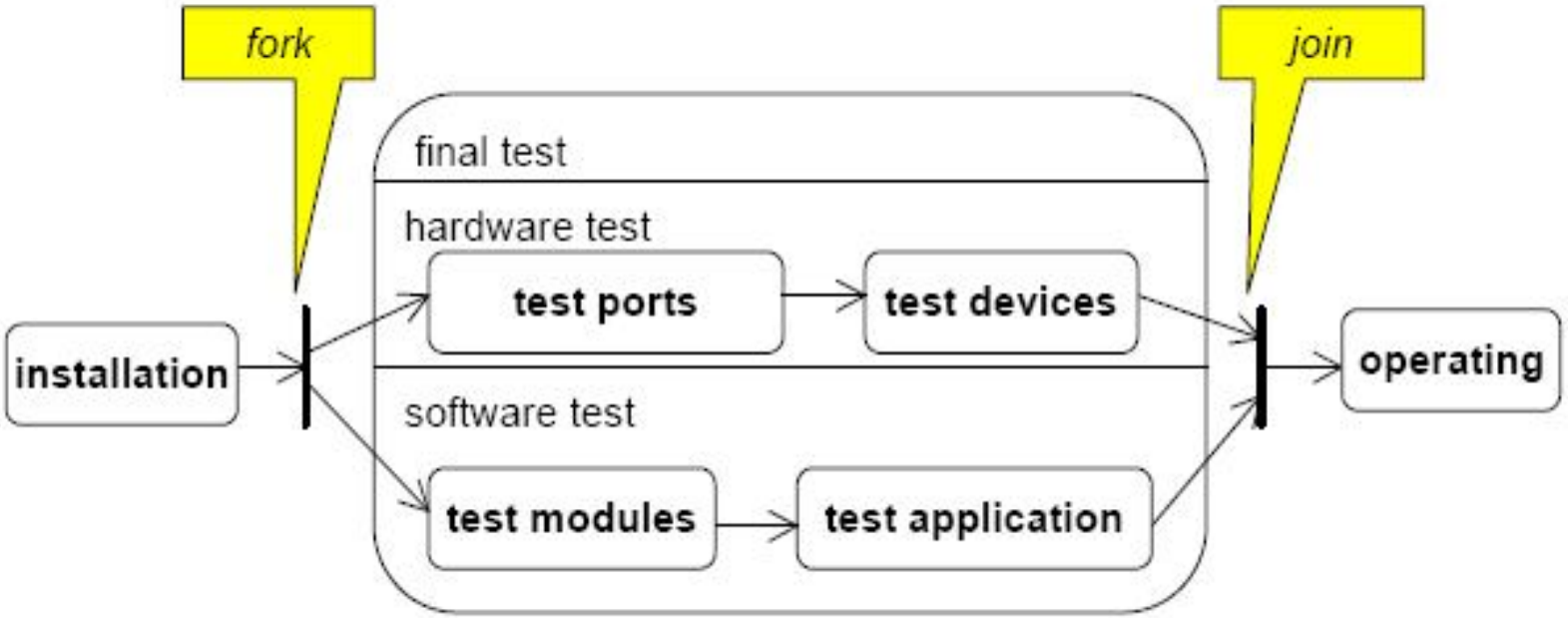
Bộ truyền động ô tô



Trạng thái con đồng thời



Trạng thái con đồng thời: Lựa chọn



Bài tập

- Vẽ biểu đồ chuyển trạng thái của một nồi cơm điện
- Vẽ biểu đồ chuyển trạng thái của máy điều hòa không khí

Biểu đồ thành phần và Biểu đồ triển khai

Biểu đồ thực thi

- Biểu diễn các khía cạnh của mô hình thực thi, bao gồm kiến trúc mã nguồn, kiến trúc mã thực thi
- Các loại biểu đồ:
  - Biểu đồ thành phần (component diagram)
  - Biểu đồ triển khai (deployment diagram)

Biểu đồ thành phần

- Biểu diễn việc tổ chức và phụ thuộc giữa các thành phần
- Thành phần bao gồm
  - Các thành phần mã nguồn
  - Các thành phần mã nhị phần
  - Các thành phần thực hiện được

Component Diagram

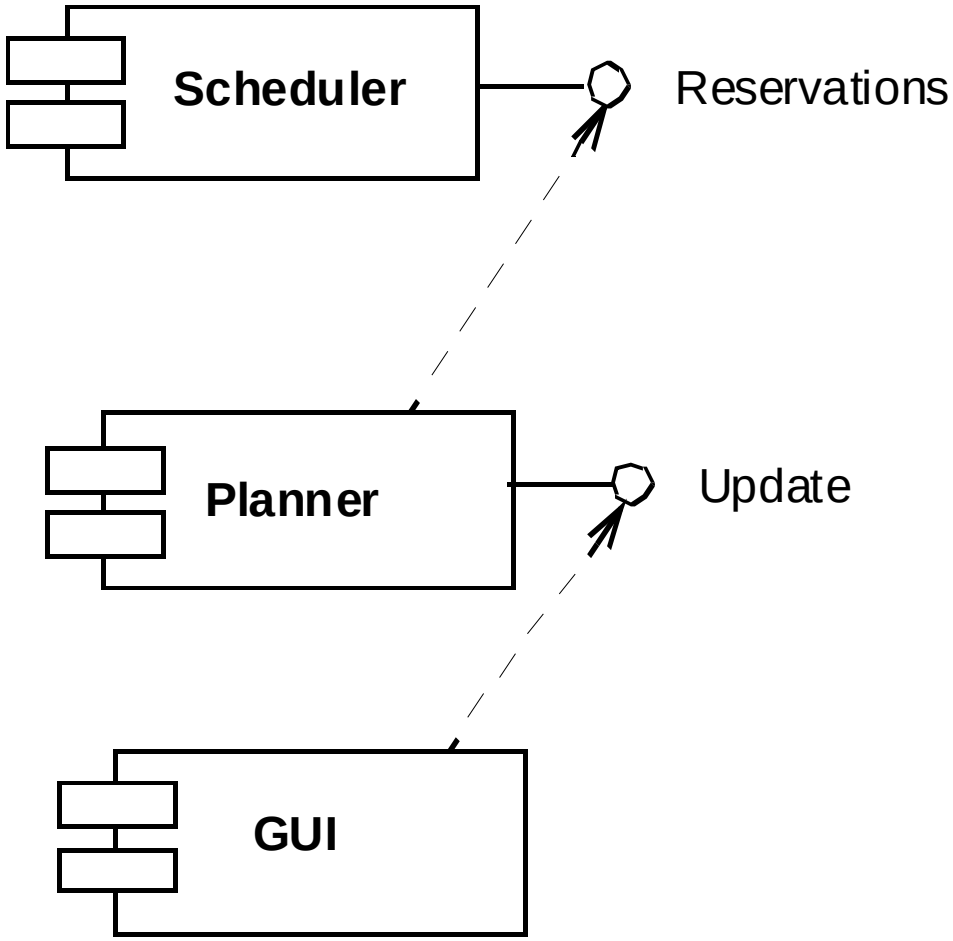
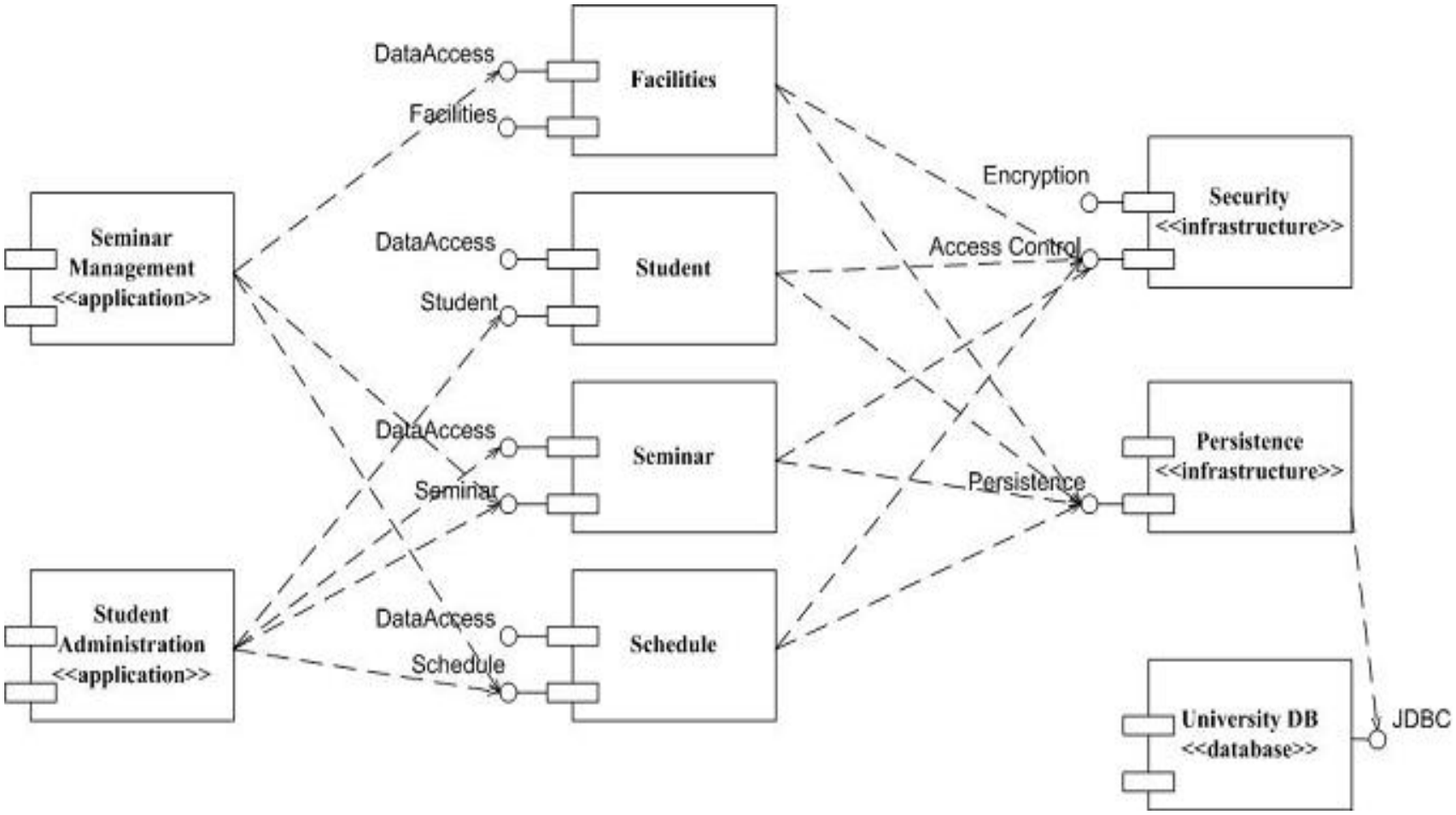
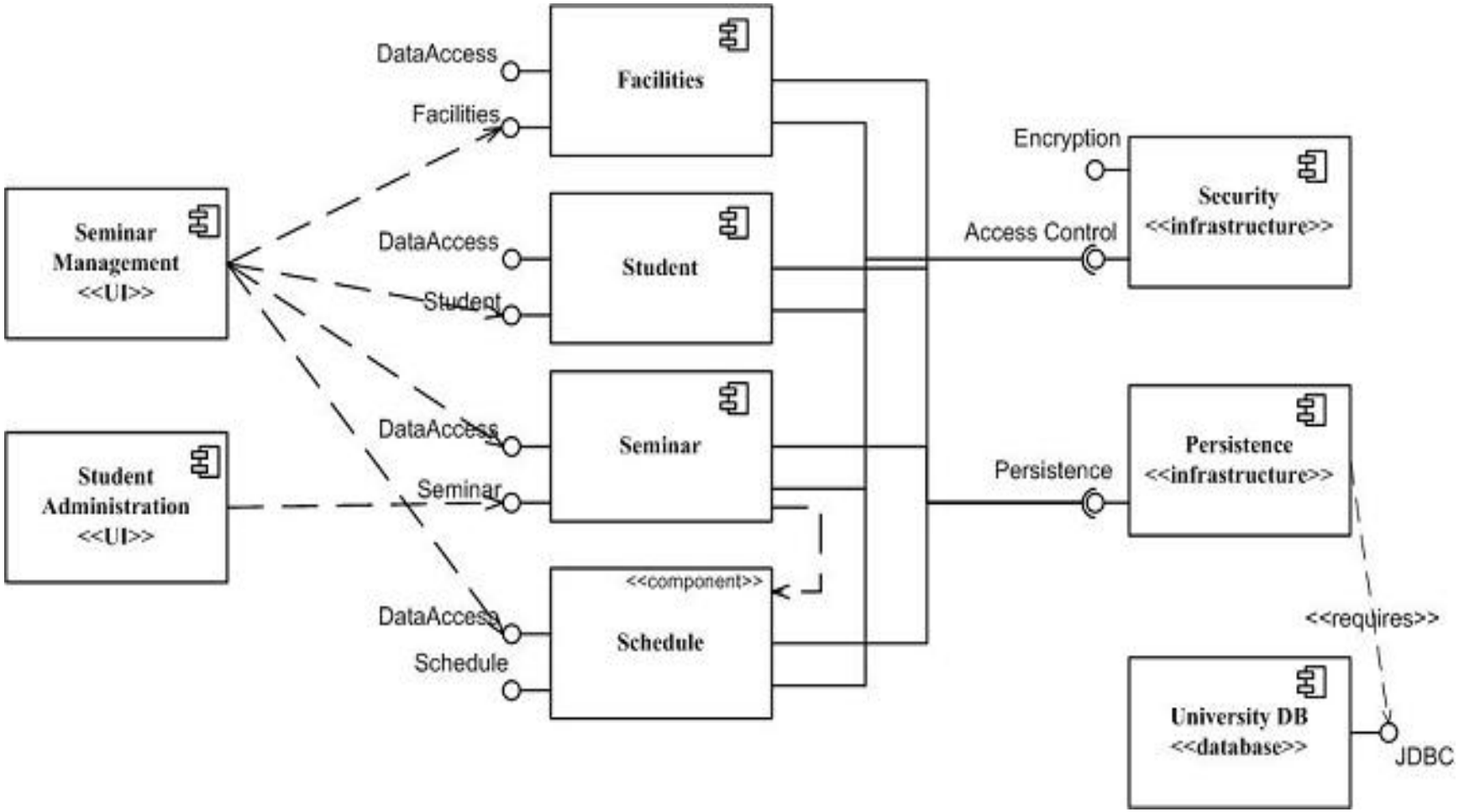


Fig. 3-81, *UML Notation Guide*

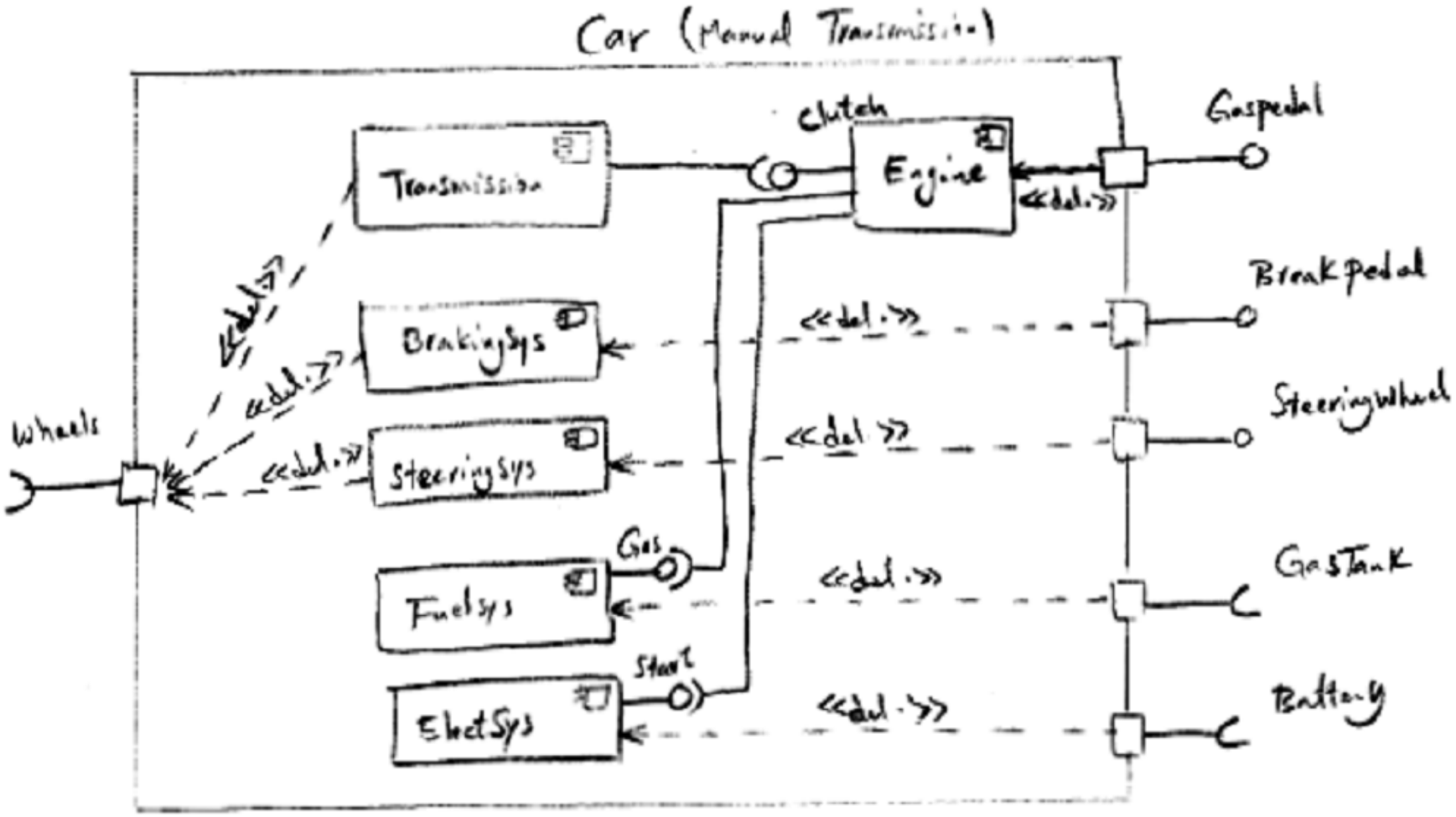
UML1.x component diagram



UML 2.x component diagram



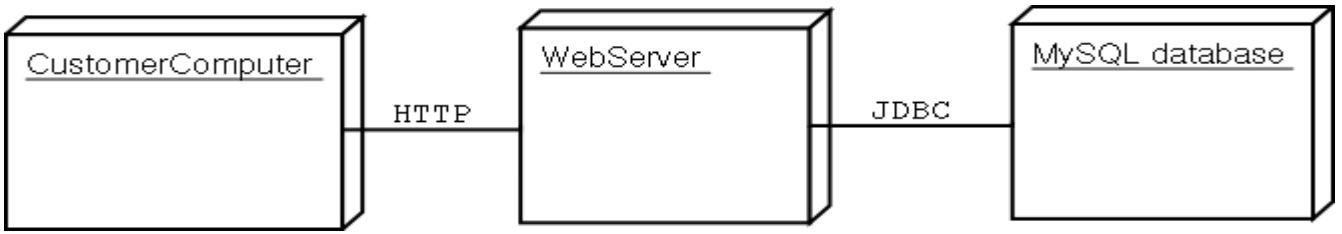
Biểu đồ thành phần của hệ thống truyền động ôtô



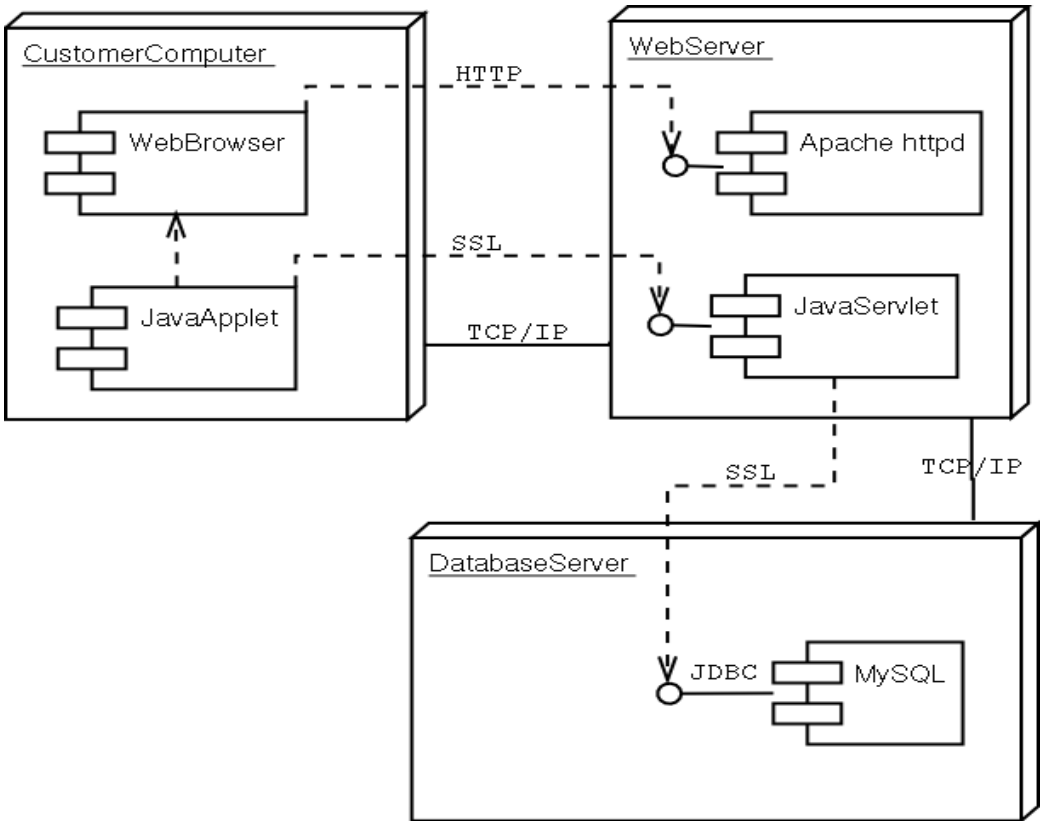
Biểu đồ triển khai

- Biểu diễn quan hệ vật lý giữa thành phần phần cứng và phần mềm như khi hệ thống đang thực hiện
- Biểu đồ triển khai có thể được sử dụng để chỉ ra thành phần nào có thể chạy trong node nào

Ví dụ về biểu đồ triển khai



Ví dụ về biểu đồ triển khai



Biểu đồ hoạt động
Activity diagrams

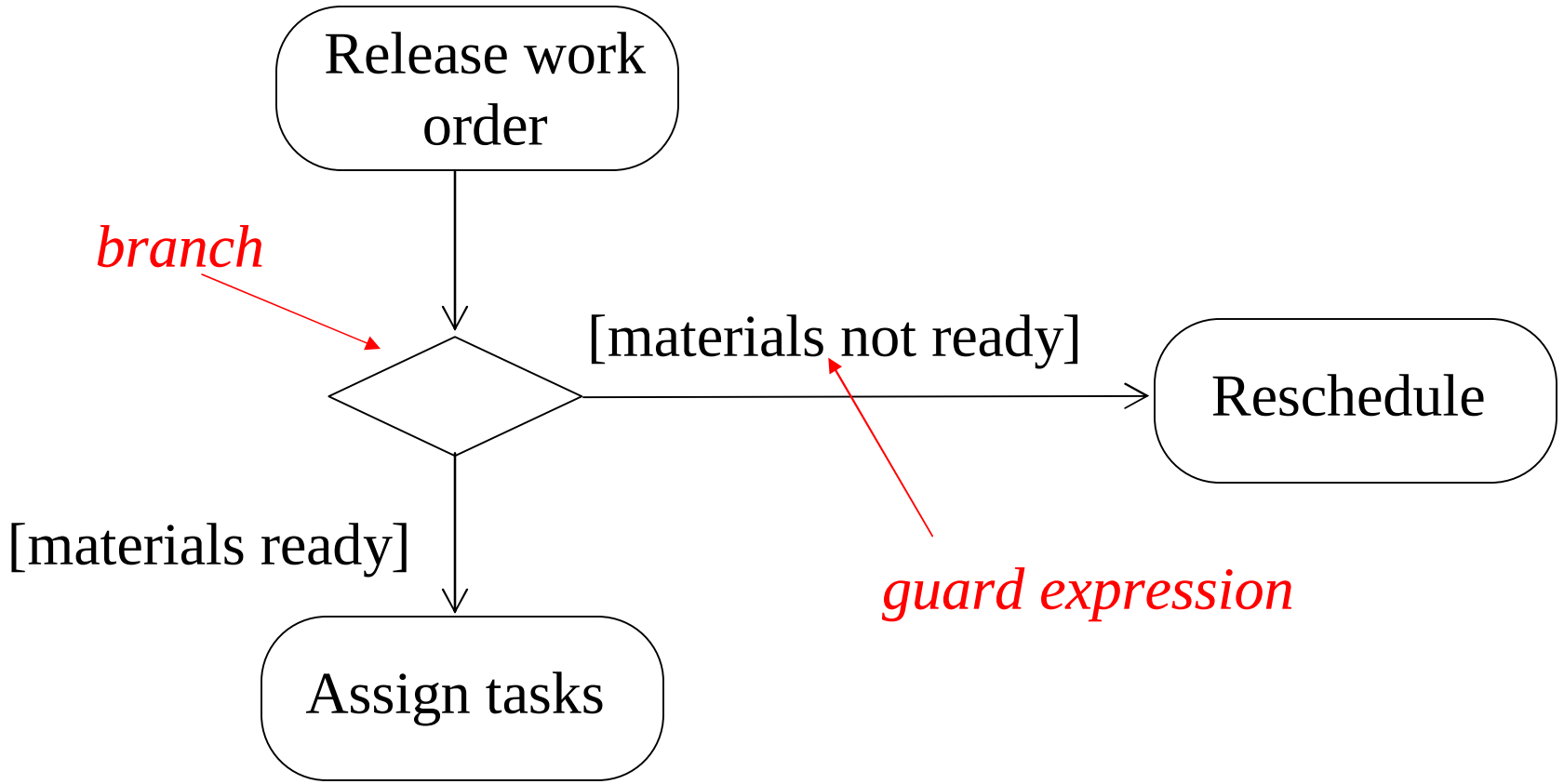
Biểu đồ hoạt động là gì?

- Biểu đồ hoạt động là phương tiện để mô tả
  - luồng nghiệp vụ (business processes),
  - luồng công việc trong ca sử dụng và
  - luồng công việc cho các phương thức phức tạp
- Biểu đồ hoạt động bao gồm *activities*, *states* và *transitions*
  - Một hoạt động là đặc tả của hành vi được biểu diễn bởi một luồng các hành động
  - Một trạng thái là điểm mà các sự kiện cần đạt tới trước khi hoạt động tiếp tục
  - Một chuyển tiếp là việc chuyển đổi giữa các hoạt động hoặc trạng thái

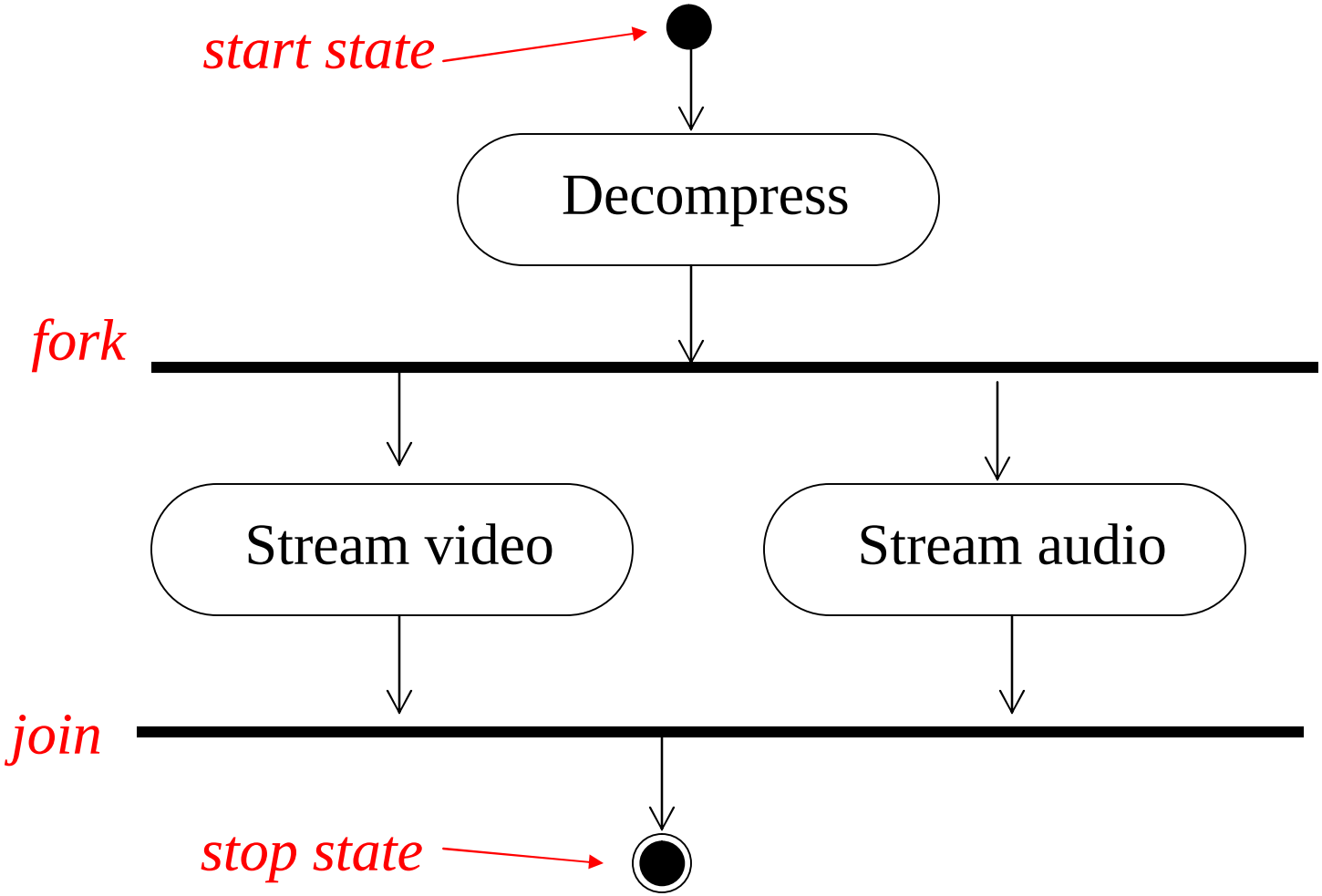
Biểu đồ hoạt động

- Điểm quyết định là điểm trong một luồng công việc mà ở đó việc chuyển tiếp từ một trạng thái hoặc một hoạt động phân theo các nhánh khác nhau tùy theo điều kiện
- Một chuyển tiếp xuất hiện khi tất cả các hành động của một hoạt động hoàn thành hoặc khi một sự kiện kích hoạt việc thoát khỏi nó từ một trạng thái hoặc sự kiện khác
- Làn bơi là một biểu diễn để chỉ ra một hoạt động diễn ra ở đâu trong một hệ thống phức tạp

Điểm quyết định (phân nhánh)



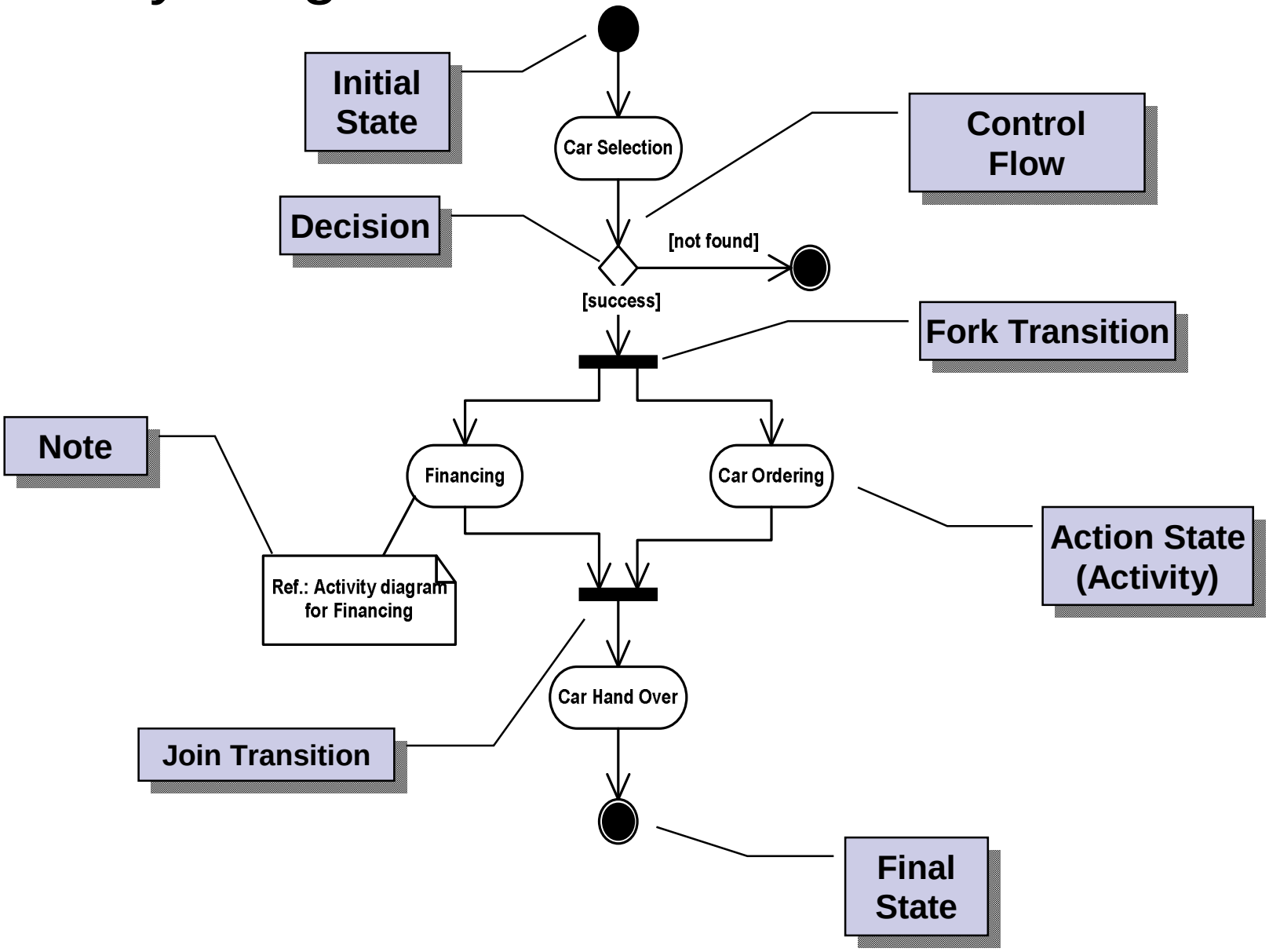
Phân nhánh và kết nối



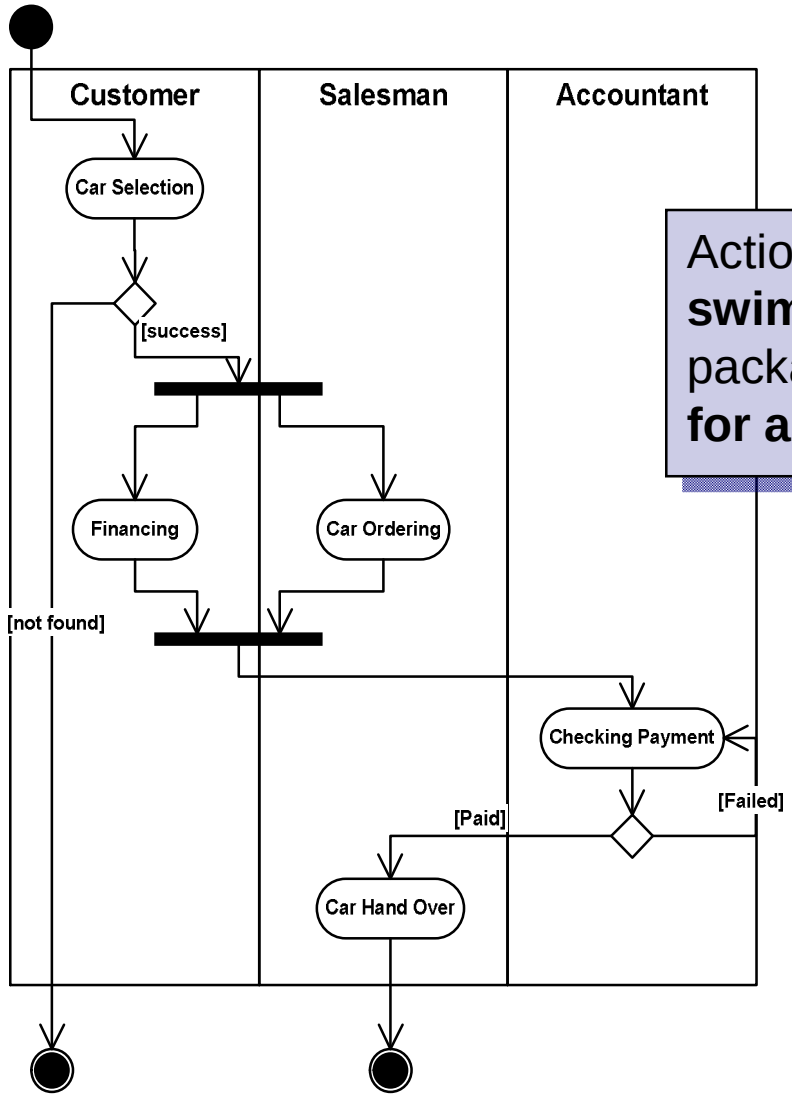
Làn bơi (Swimlanes)

- Sử dụng để mô hình hóa luồng công việc trong tiến trình nghiệp vụ
- Chỉ ra ai có trách nhiệm thực hiện từng hoạt động
- Để phân hoạch các trạng thái hoạt động vào nhóm
- Mỗi hoạt động thuộc về một làn bơi
- Chuyển tiếp có thể được vẽ từ làn bơi này đến làn bơi khác
- Mỗi làn bơi có thể được cài đặt bởi một hay nhiều lớp

Activity Diagram: Car Sale Process



Swimlanes: Packages of Responsibilities

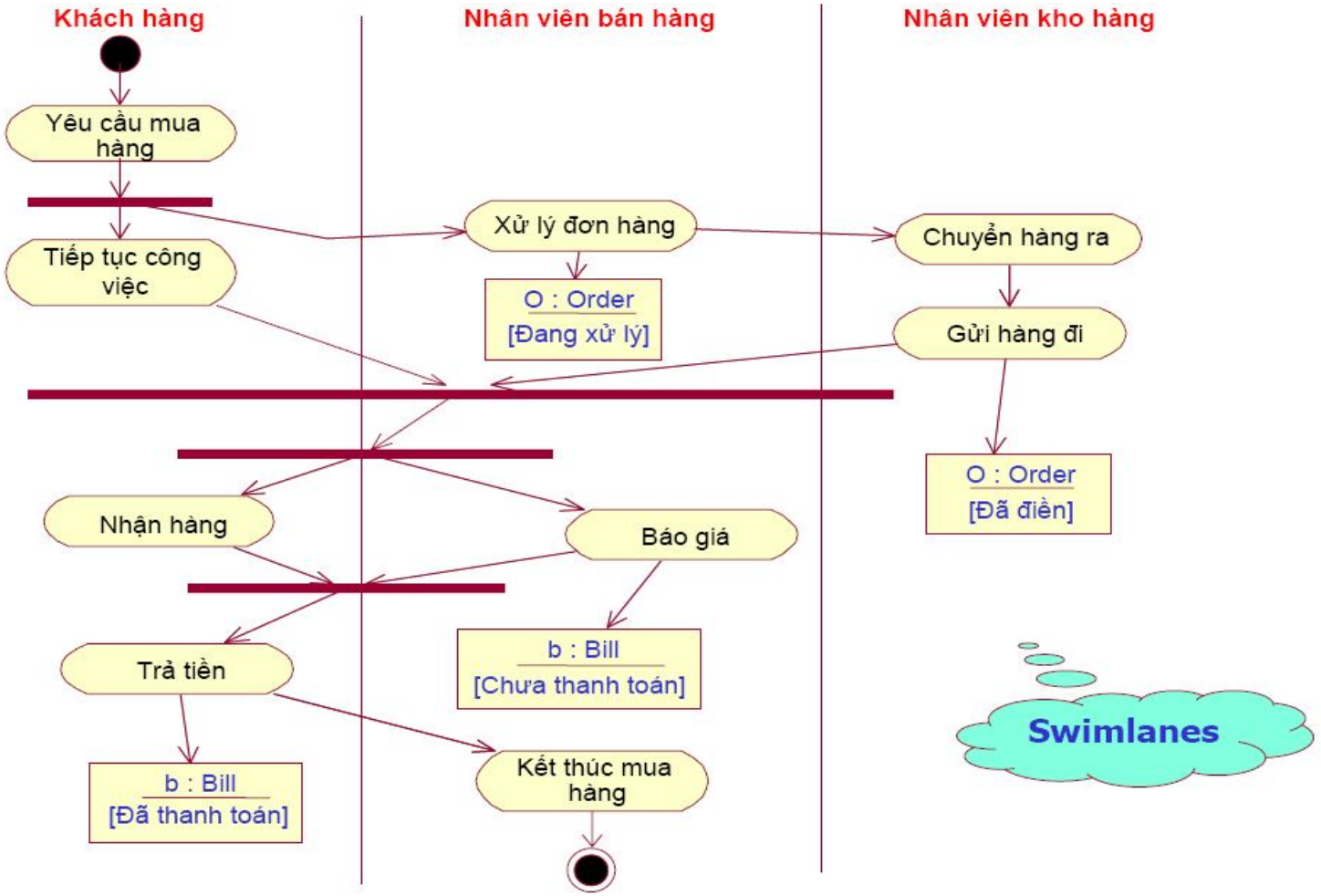


Actions may be organized into **swimlanes**. Swimlanes are a kind of package for organizing **responsibility for activities** provided by workers.

Bài tập

- Vẽ biểu đồ hoạt động của luồng nghiệp vụ mua hàng
- Vẽ biểu đồ hoạt động của luồng nghiệp vụ mua sách trên mạng

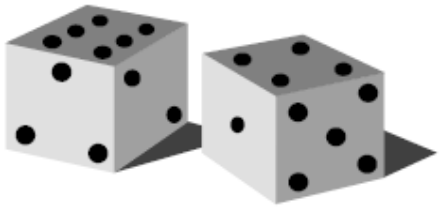
Biểu đồ hoạt động



Object oriented design:
A case study

Trò chơi xúc xắc

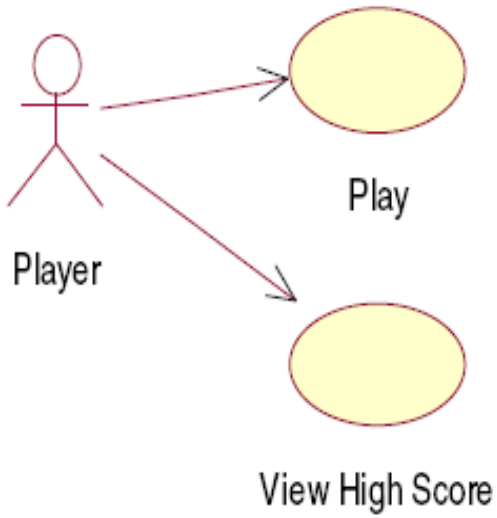
- Người chơi tung 10x2 xúc xắc
- Nếu tổng điểm xuất hiện là 7 thì ghi được 10 điểm
- Số điểm tung một ván được ghi vào bảng điểm



Phân tích yêu cầu

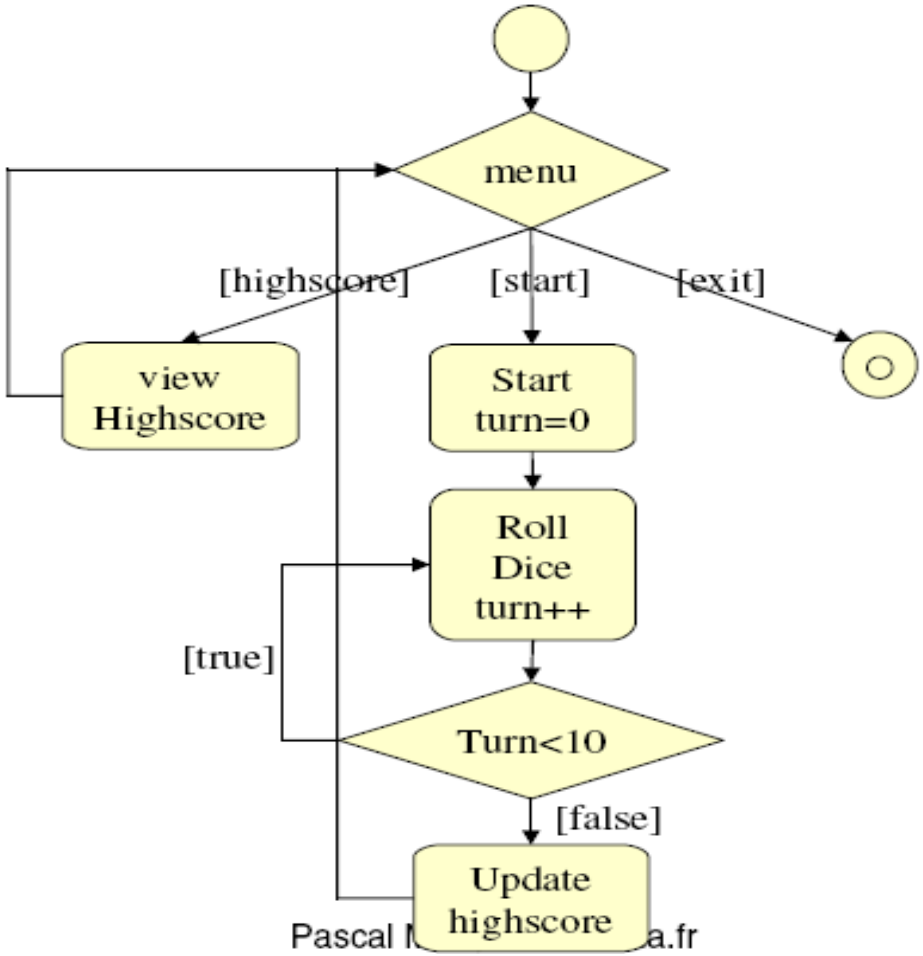
- Xác định các tác nhân
- Xác định các trường hợp sử dụng
- Các chức năng ngoài

Use case thứ nhất

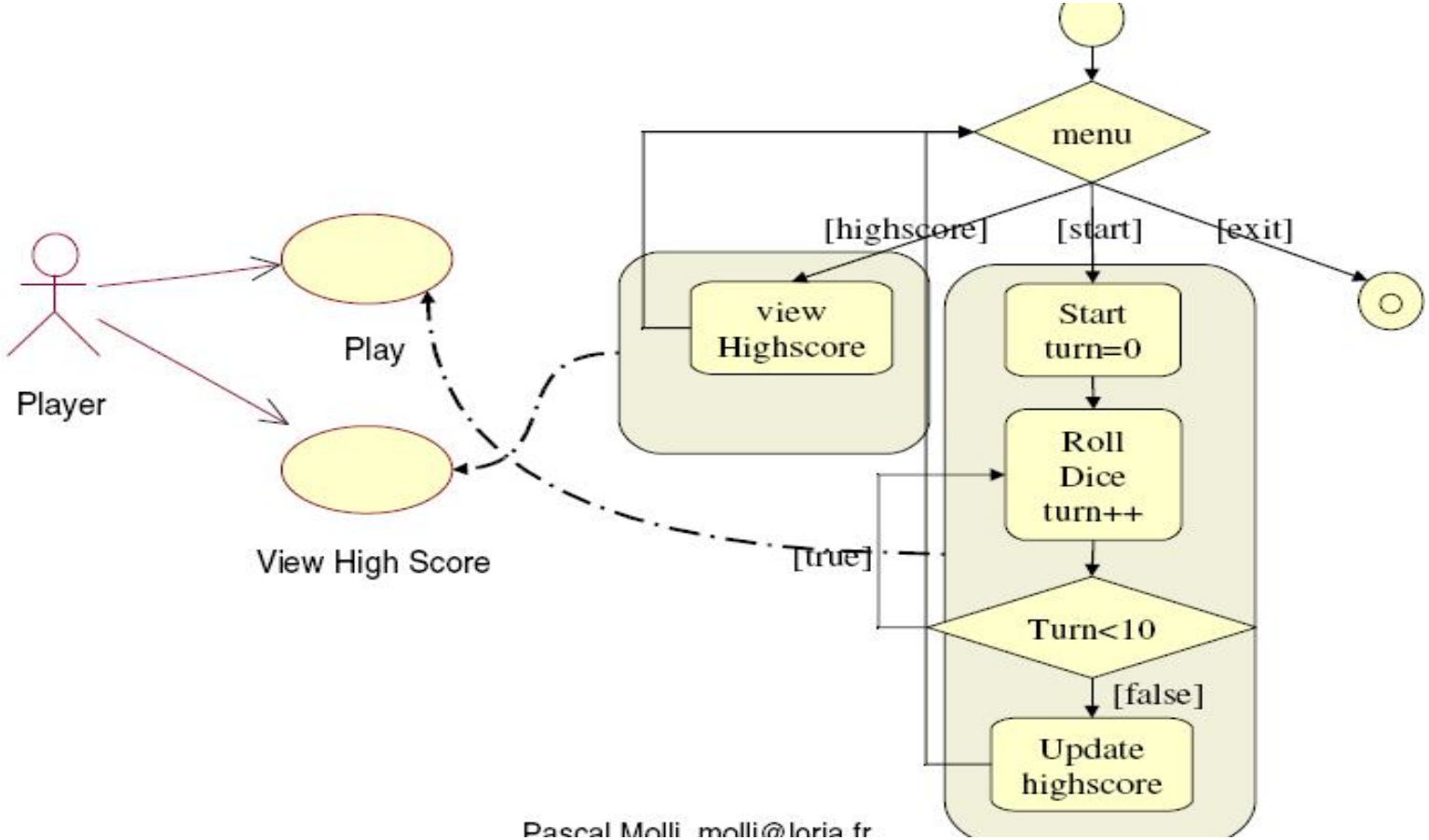


- Play
  - Tác nhân: Người chơi
  - Mô tả: Người chơi tung 10x xúc xắc, mỗi lần xuất hiện 7 nodes, ghi được 10 điểm
- View high score
  - Tác nhân: Người chơi
  - Mô tả: Người chơi xem điểm ghi được

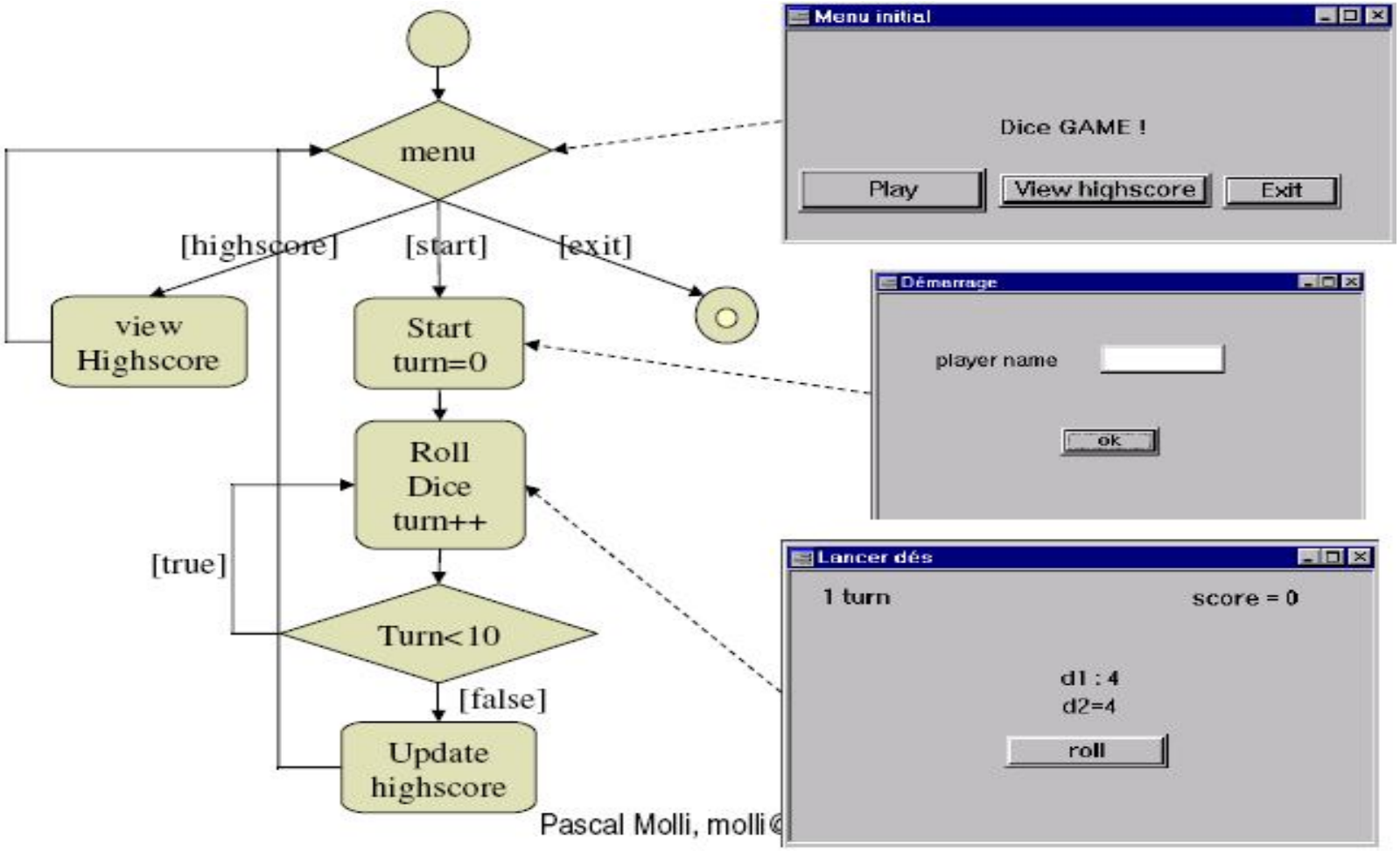
Biểu đồ hoạt động



Biểu đồ hoạt động



Biểu đồ hoạt động



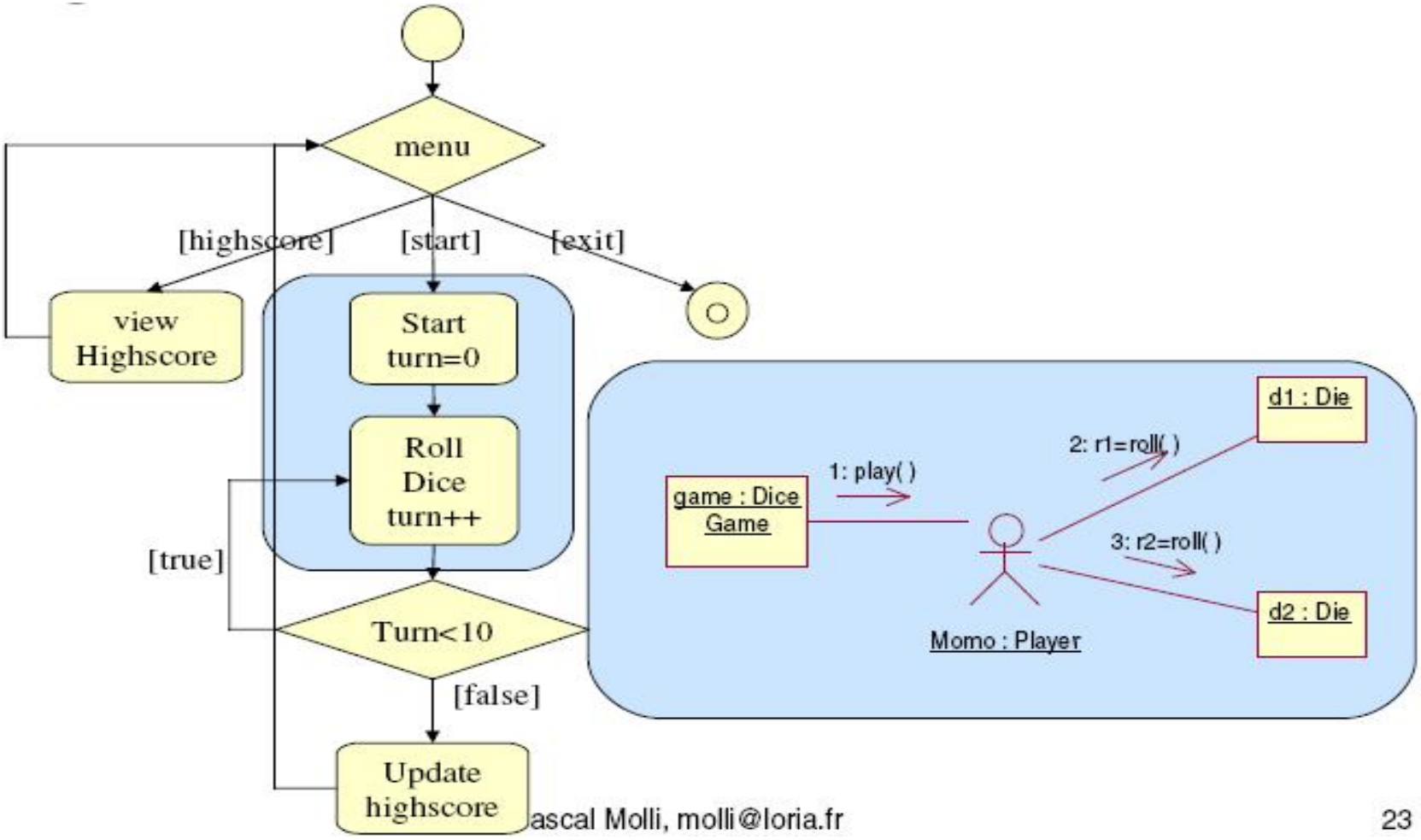
Phân tích

- Mô hình hóa thế giới thực
- Độc lập với thế giới thực
- Xác định các lớp đối tượng của thế giới thực
- Mô hình hóa phần động của hệ thống: Biểu đồ cộng tác

Biểu đồ cộng tác

- Xác định các đối tượng
- Quan hệ giữa các đối tượng
- Thông điệp và thứ tự gửi thông điệp giữa các đối tượng

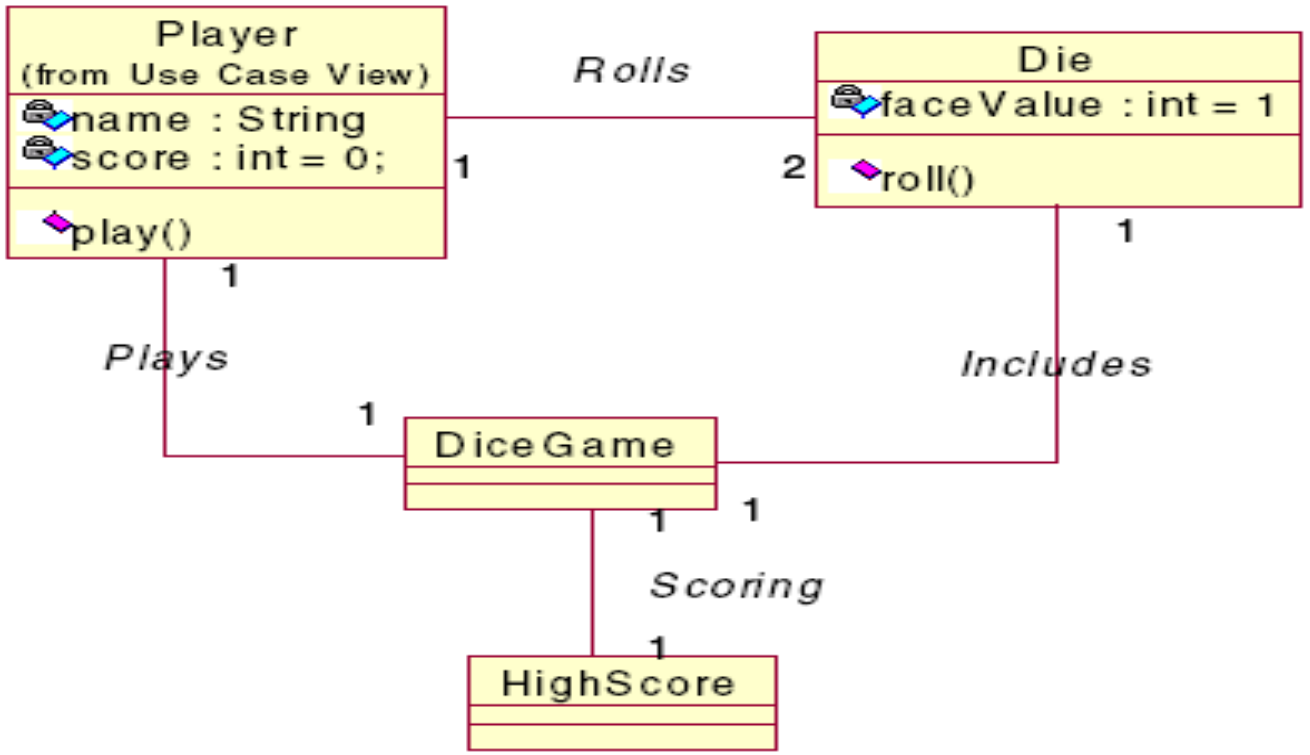
Biểu đồ cộng tác



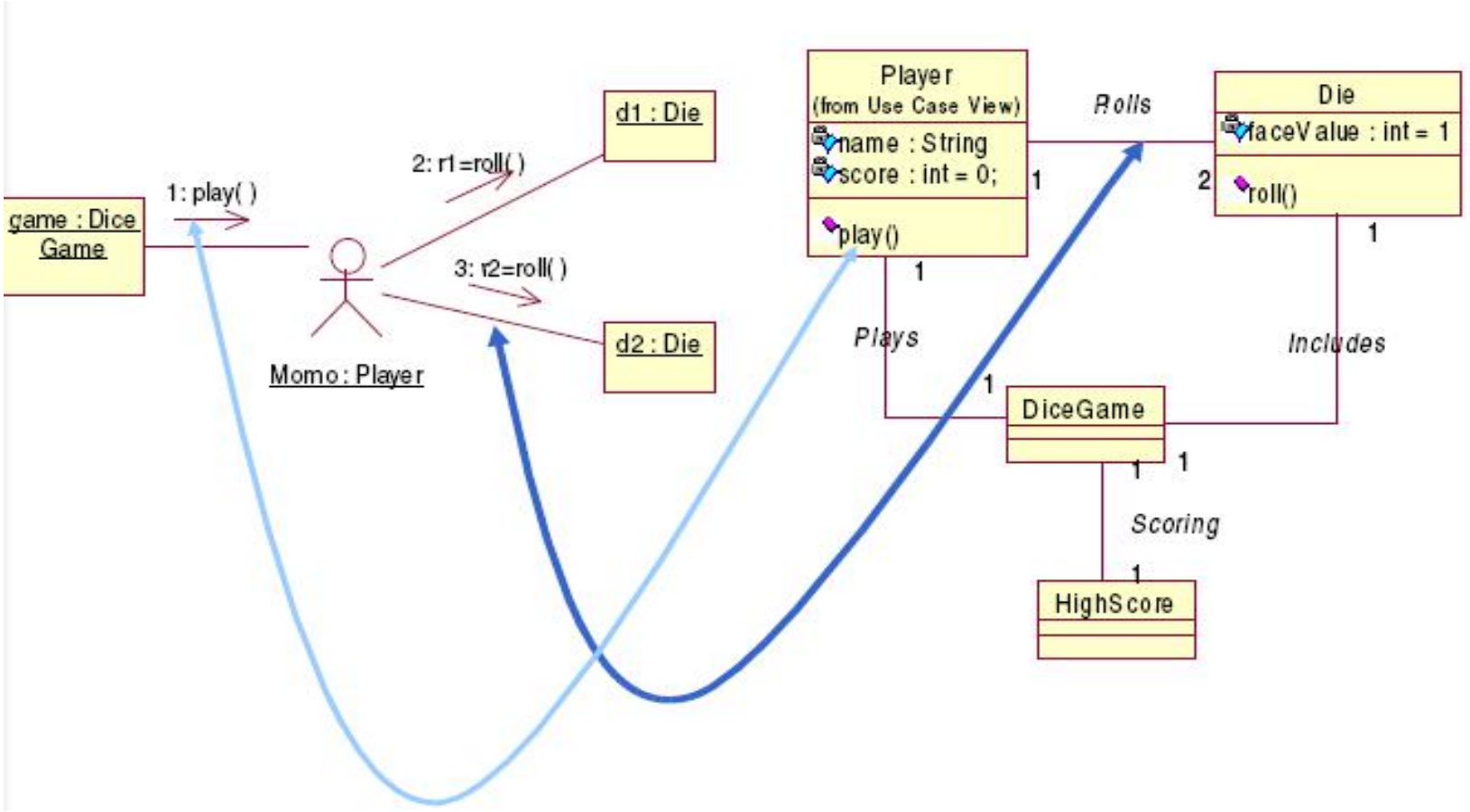
Biểu đồ lớp

- Xác định các lớp
- Xác định quan hệ giữa các lớp
- Xác định thuộc tính
- Xác định phương thức

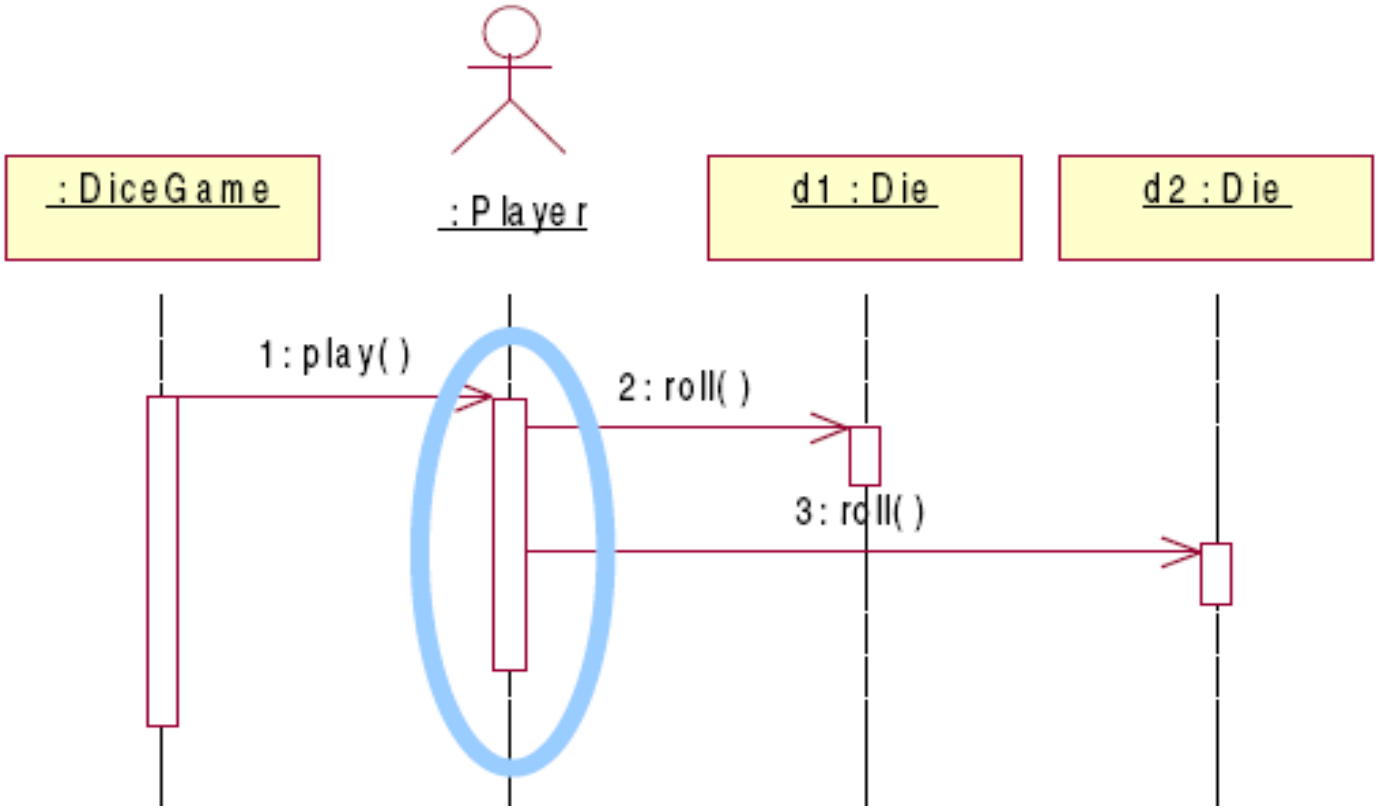
Biểu đồ lớp


Player
(from Use Case View)
name : String
score : int = 0;
play()
Rolls
1
2
Die
faceValue : int = 1
roll()
1
1
Plays
1
Includes
DiceGame
1
1
Scoring
1
HighScore

Biểu đồ lớp


game : Dice Game
1: play( )
Momo : Player
2: r1=roll( )
3: r2=roll( )
d1 : Die
d2 : Die
Player
(from Use Case View)
name : String
score : int = 0;
play()
Rolls
Die
faceValue : int = 1
roll()
Plays
Includes
DiceGame
Scoring
HighScore
1
2

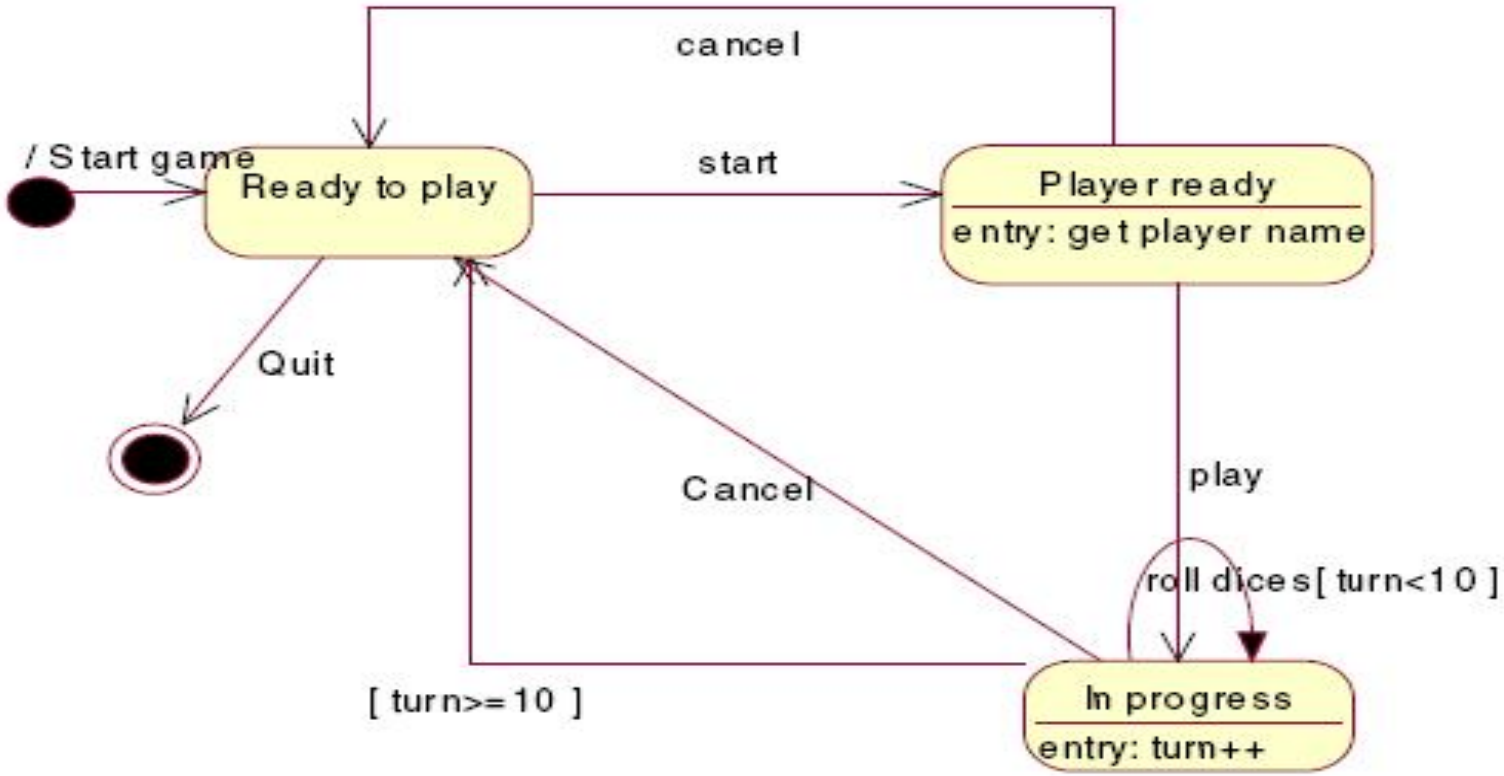
Biểu đồ tuần tự



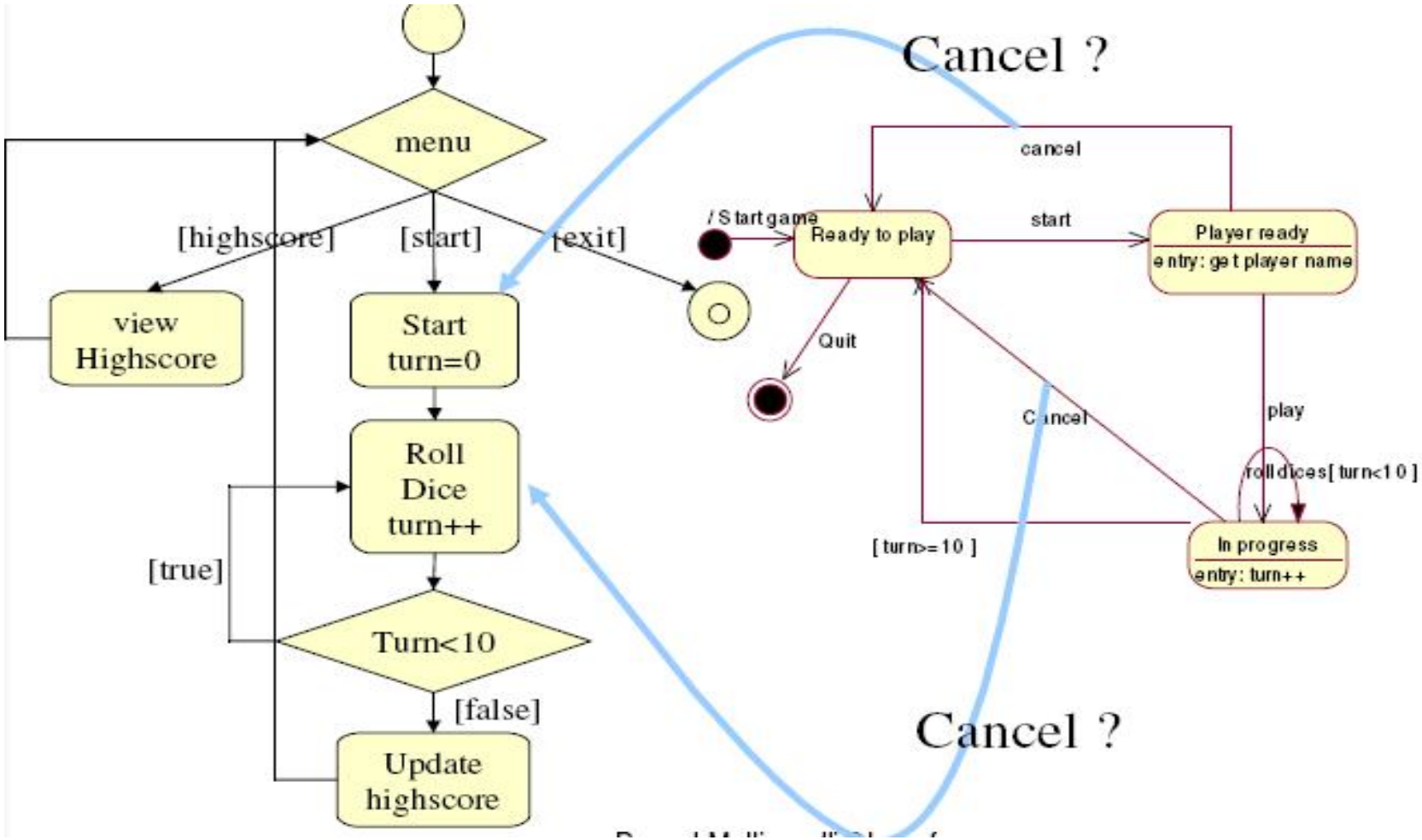
Biểu đồ trạng thái

- Xác định các trạng thái của đối tượng
- Xác định sự chuyển trạng thái trong đối tượng

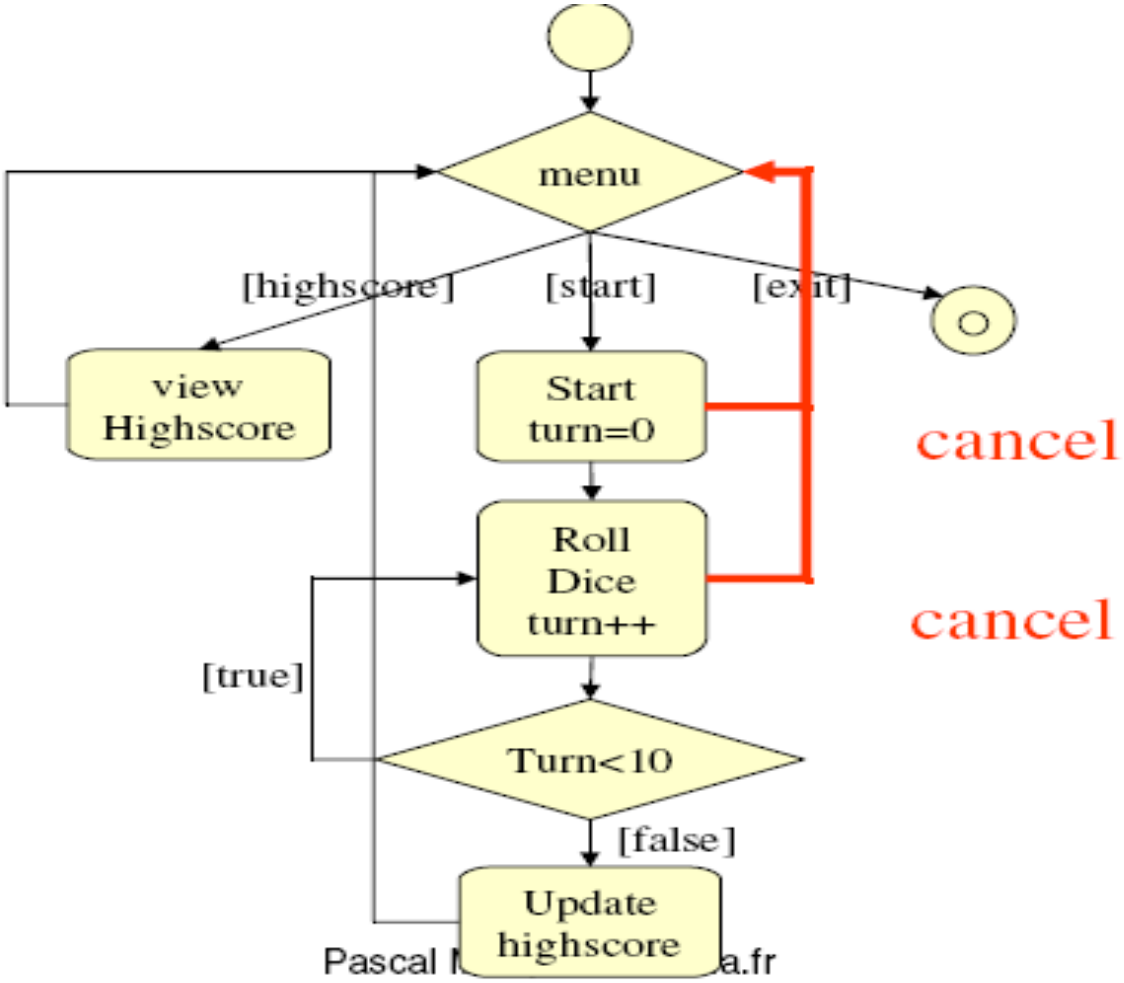
Biểu đồ trạng thái



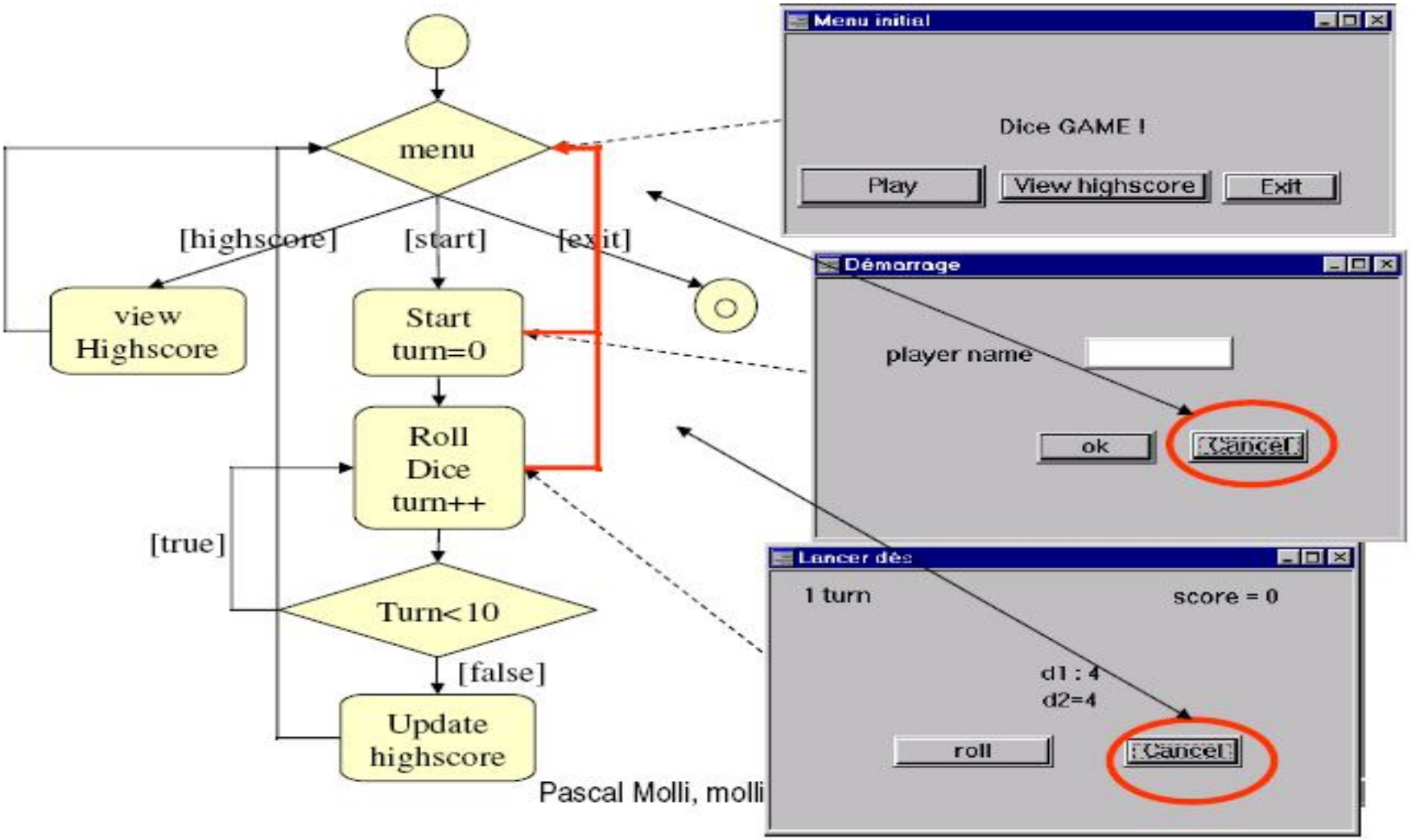
Biểu đồ trạng thái



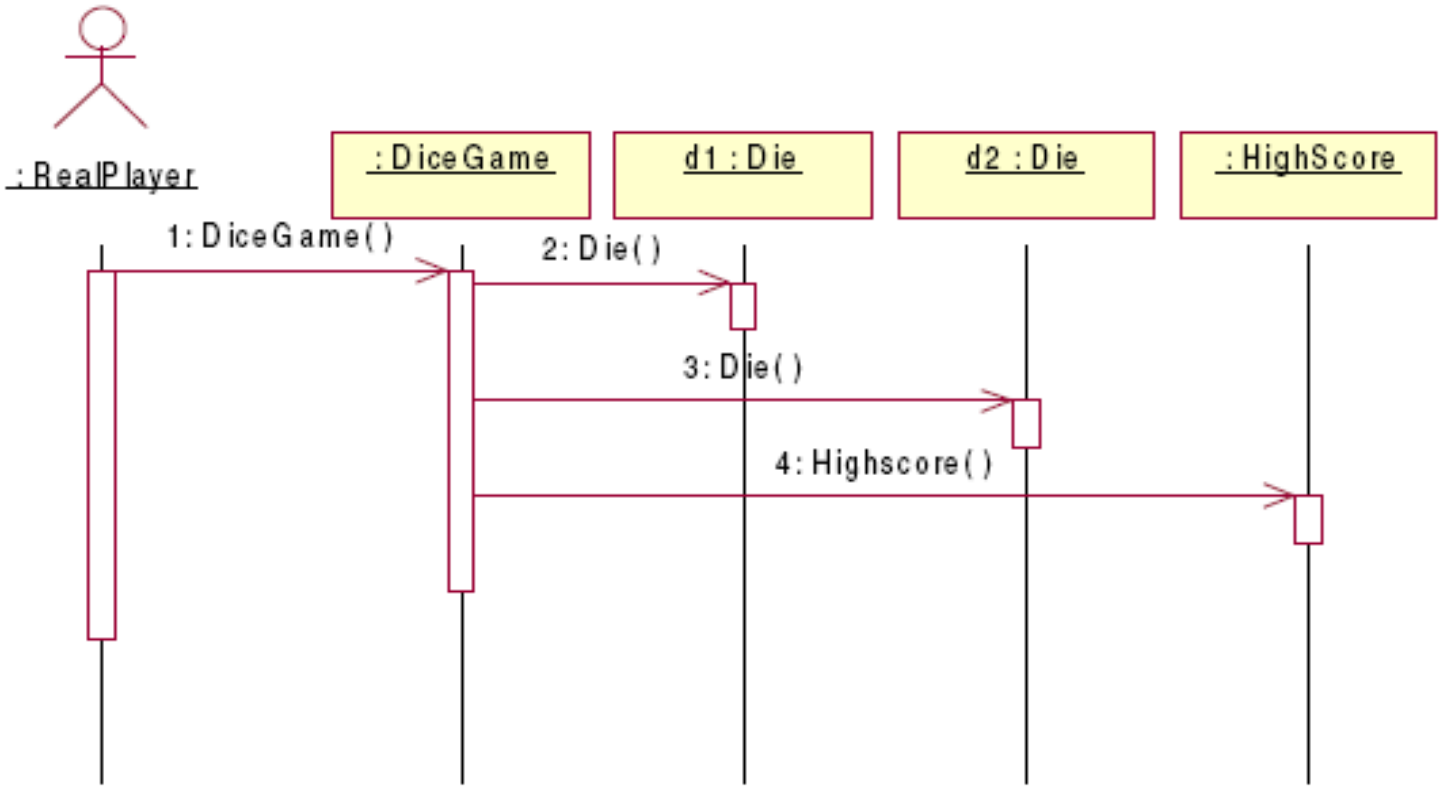
Thay đổi biểu đồ



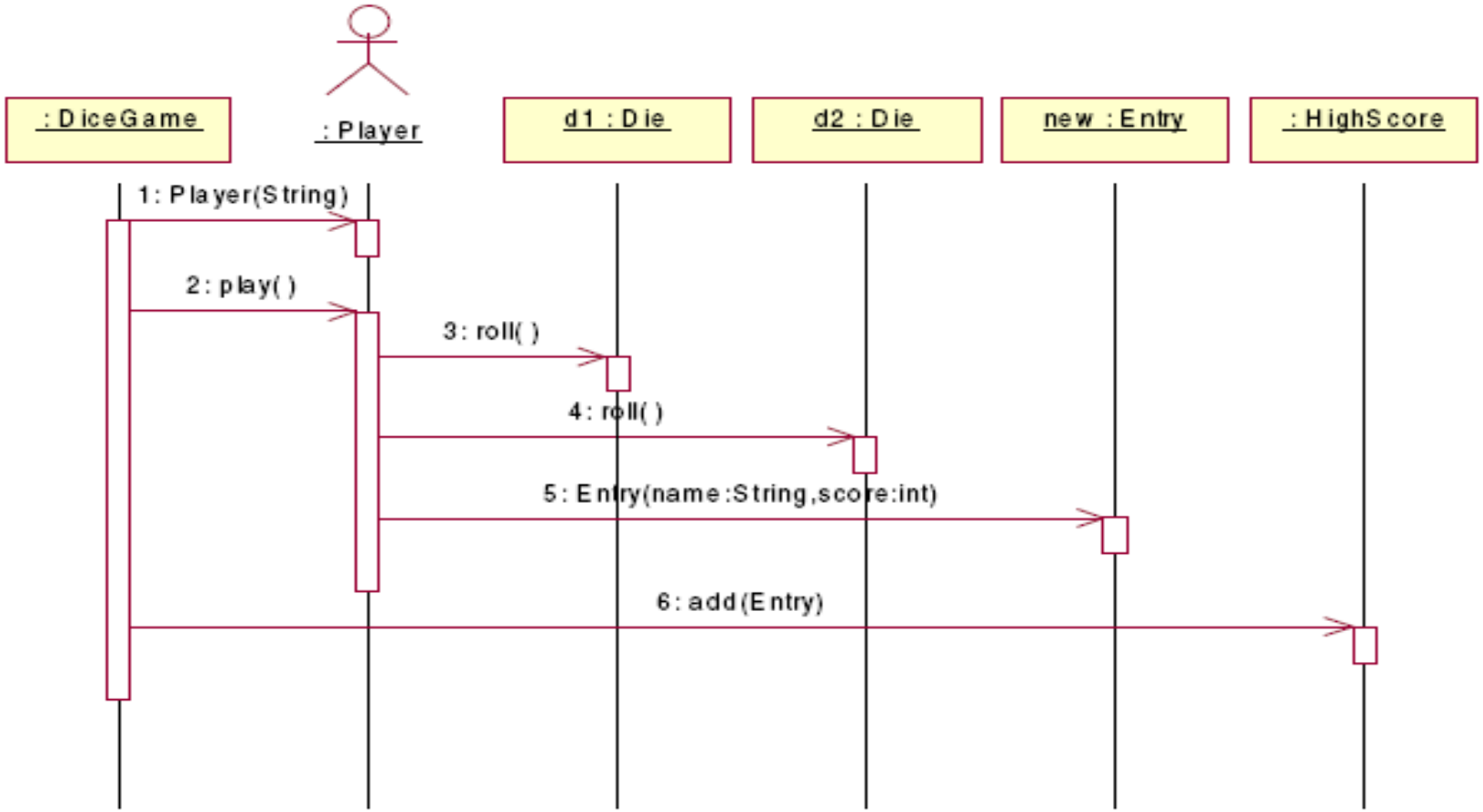
Thay đổi biểu đồ



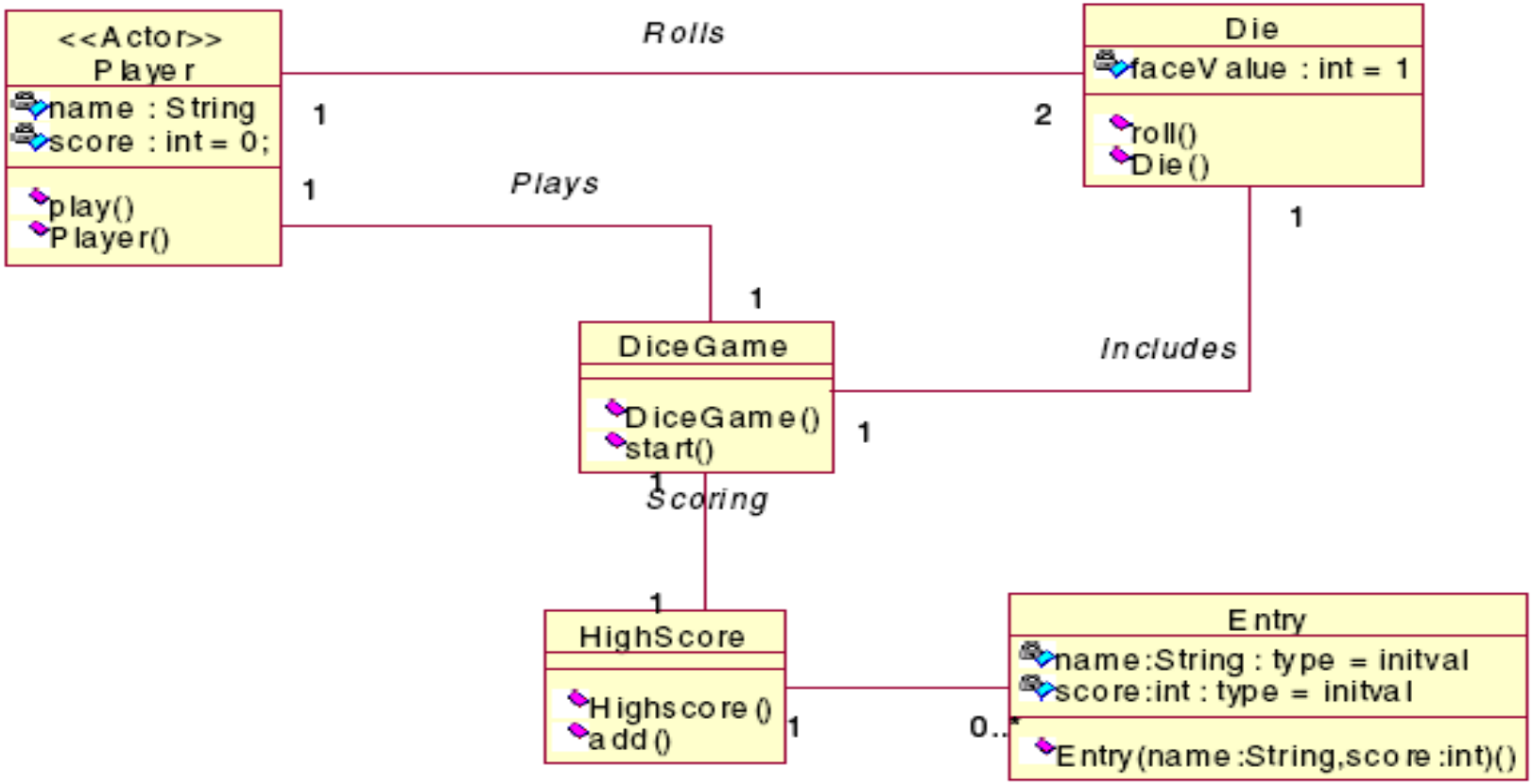
Biểu đồ tuần tự



Biểu đồ tuần tự



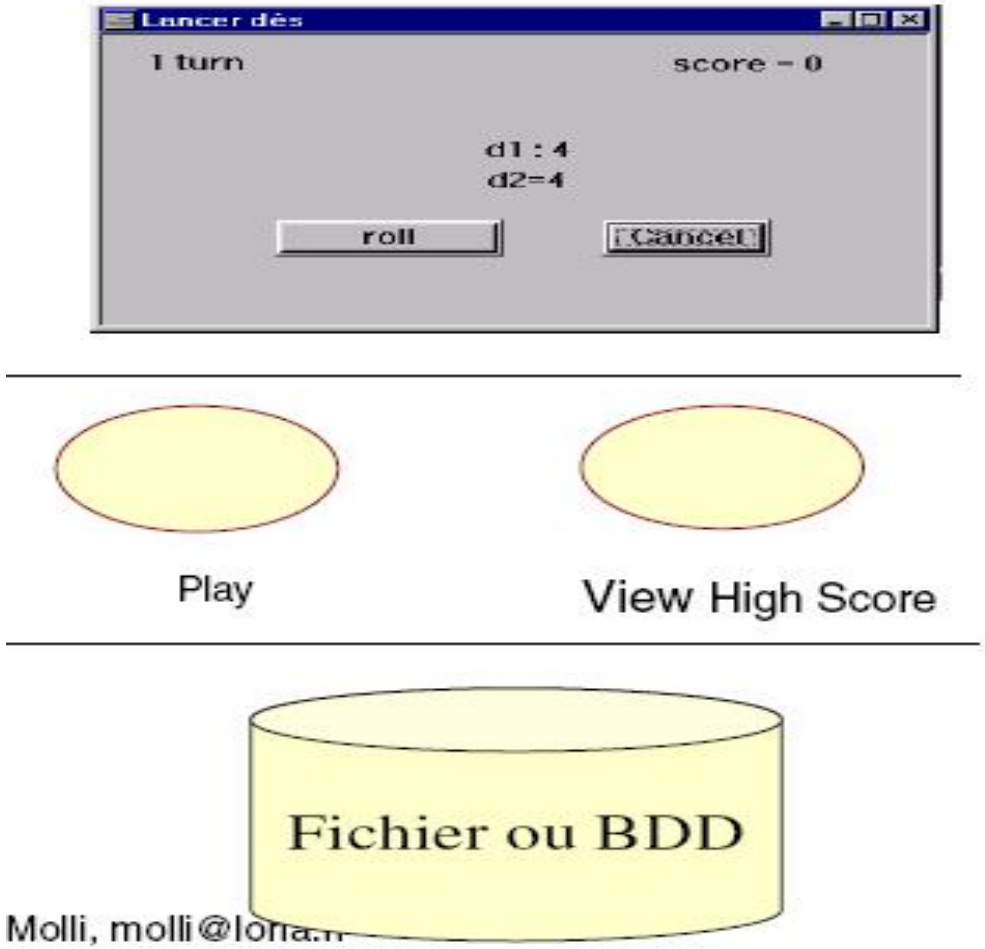
Biểu đồ lớp


<<Actor>>
Player
name : String
score : int = 0;
play()
Player()
Rolls
1
2
Die
faceValue : int = 1
roll()
Die()
1
Plays
1
1
DiceGame
DiceGame()
start()
Includes
1
1
Scoring
1
HighScore
Highscore()
add()
1
0..*
Entry
name:String : type = initval
score:int : type = initval
Entry(name:String,score:int)()

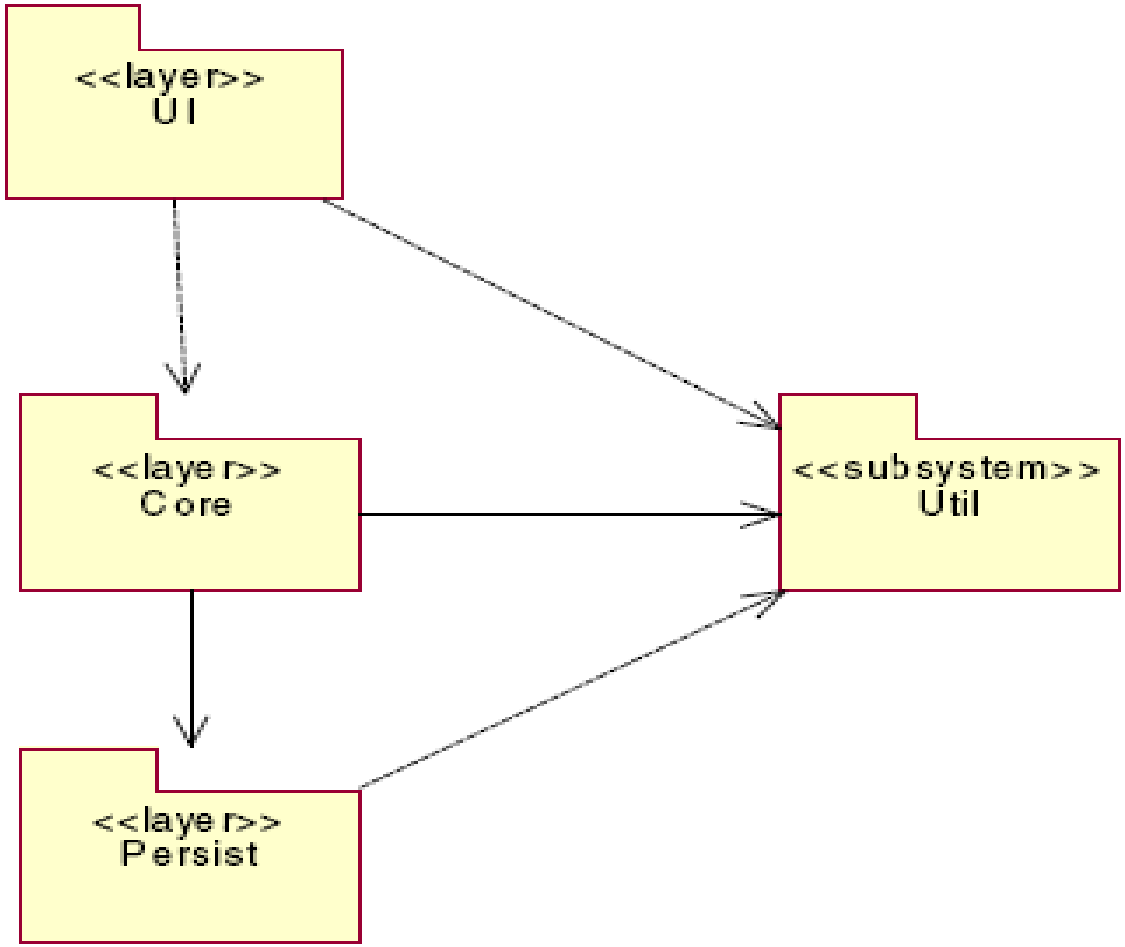
Thiết kế

- Chú ý việc thực thi phần mềm
- Định nghĩa kiến trúc logic
- Định nghĩa kiến trúc vật lý

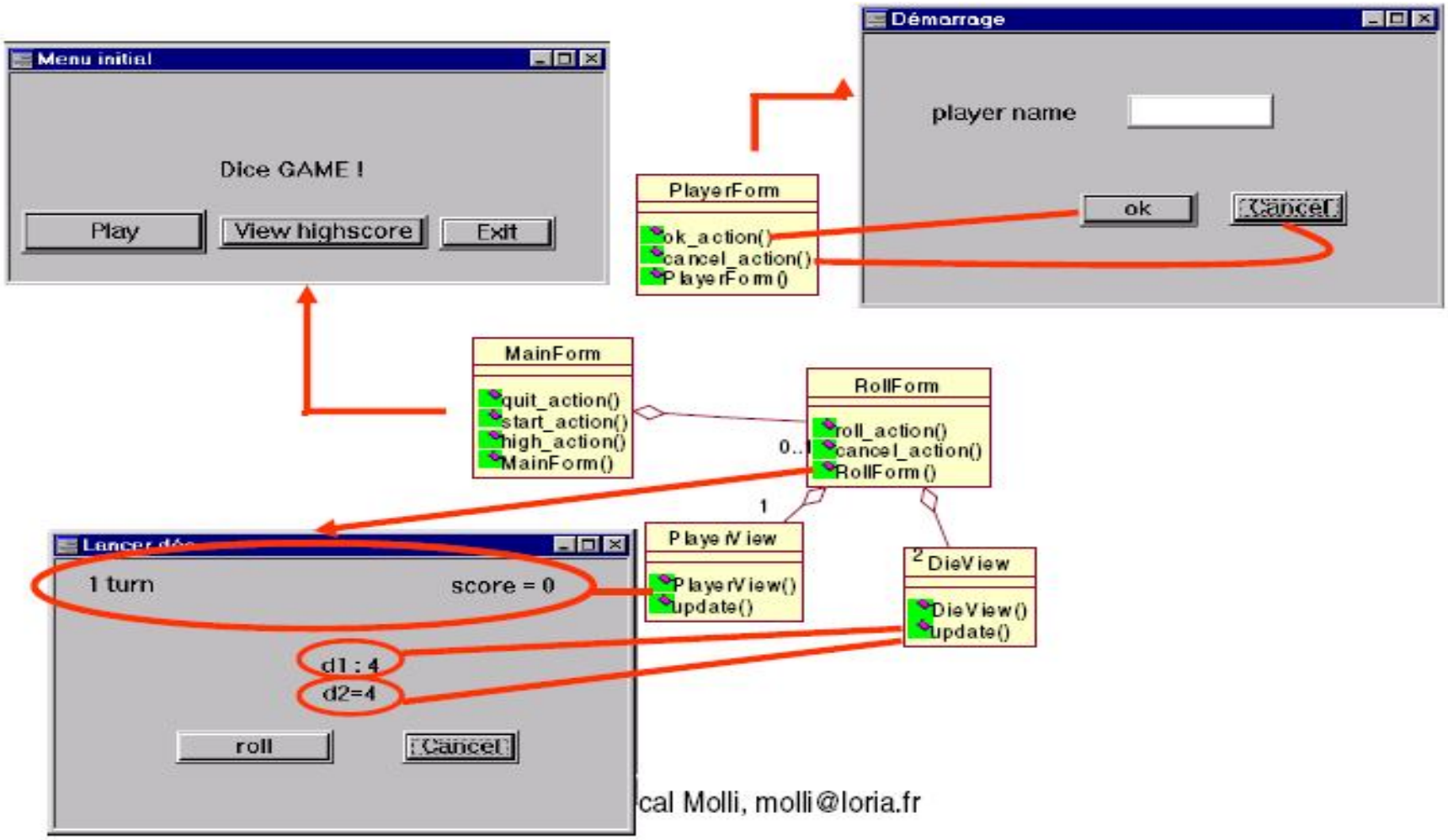
Thiết kế


Lancer dés
1 turn
score = 0
d1 : 4
d2=4
roll
Cancel
Play
View High Score
Fichier ou BDD

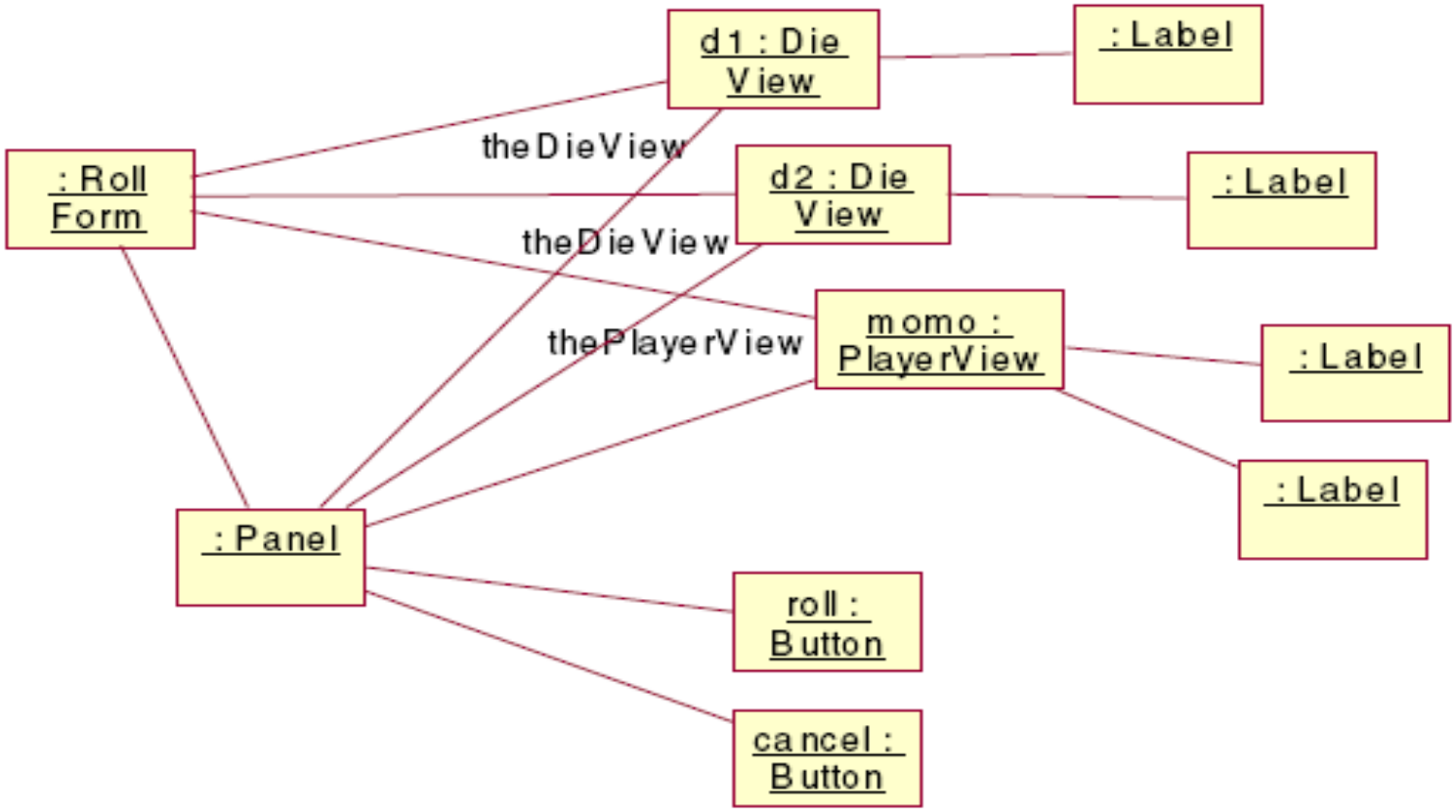
Kiến trúc theo tầng



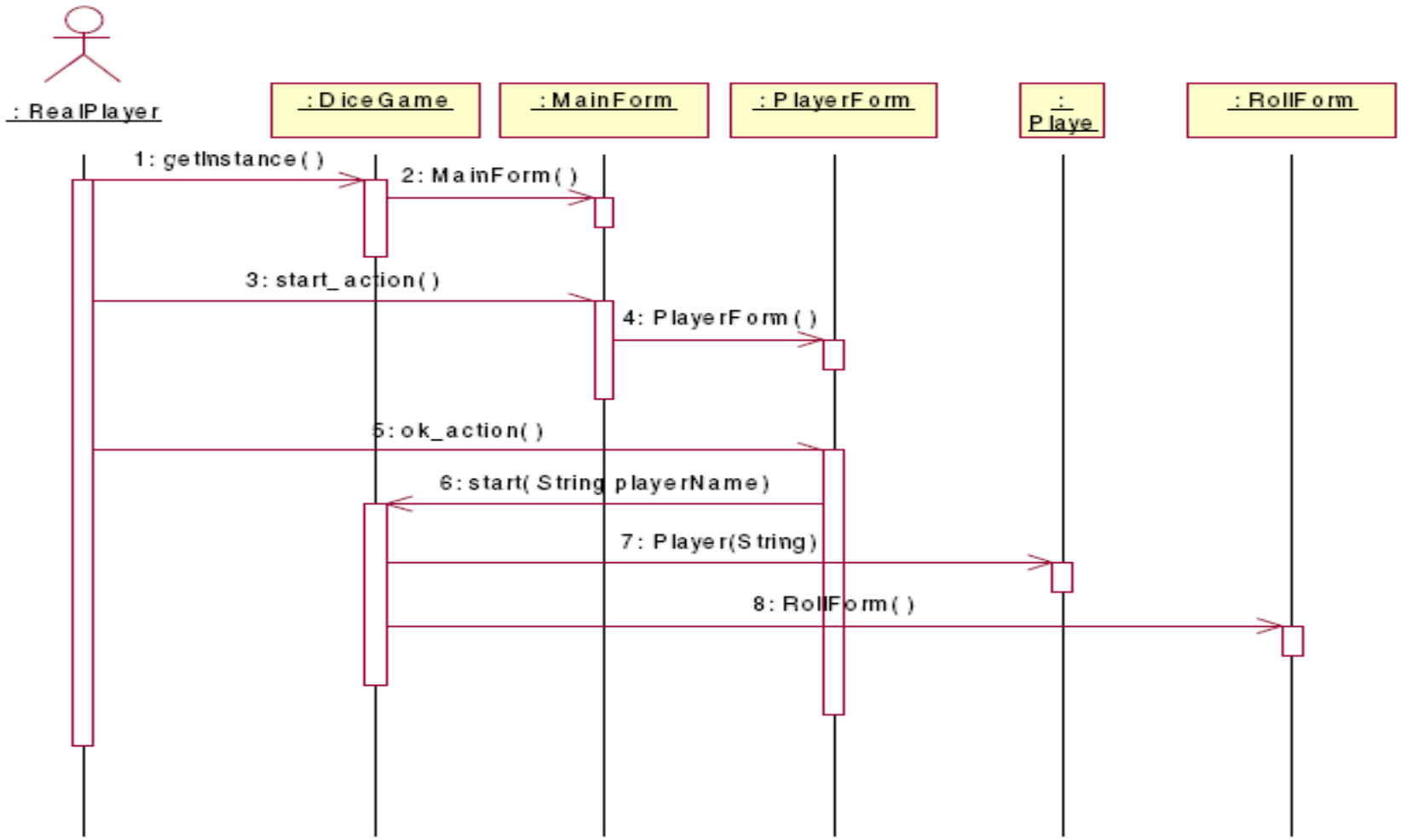
Ánh xạ các lớp UI, UI



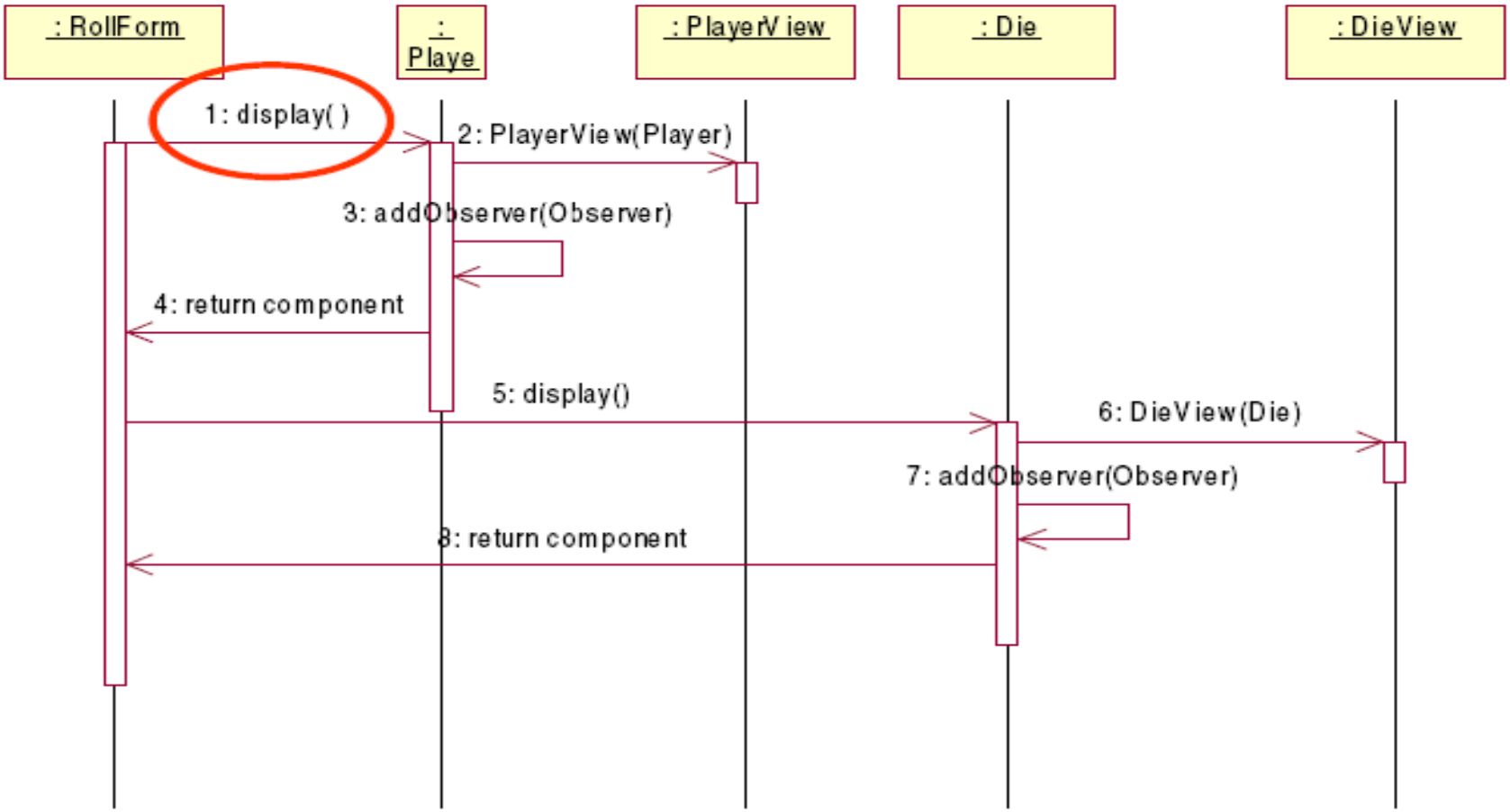
Biểu đồ đối tượng



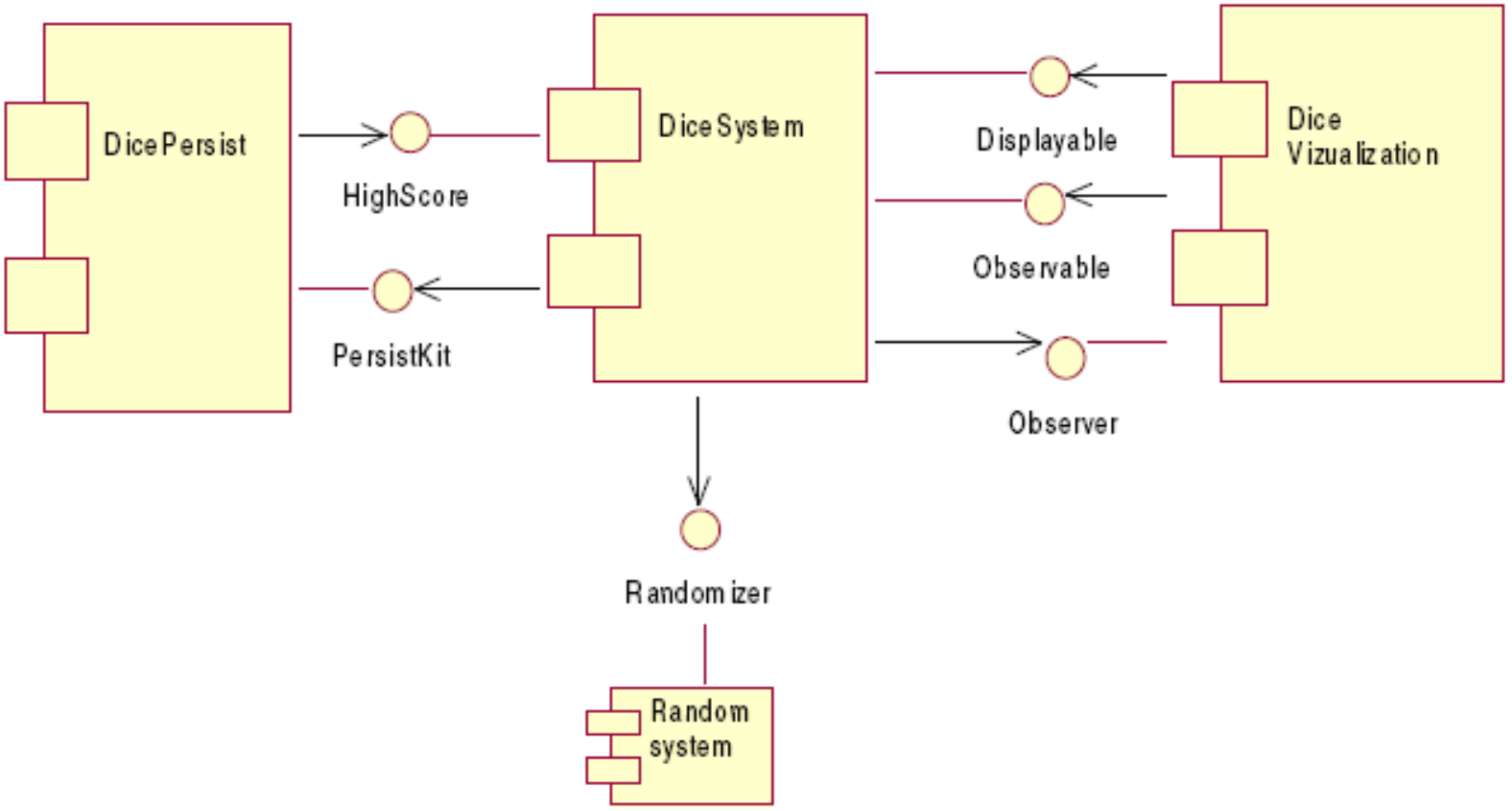
Biểu đồ tuần tự



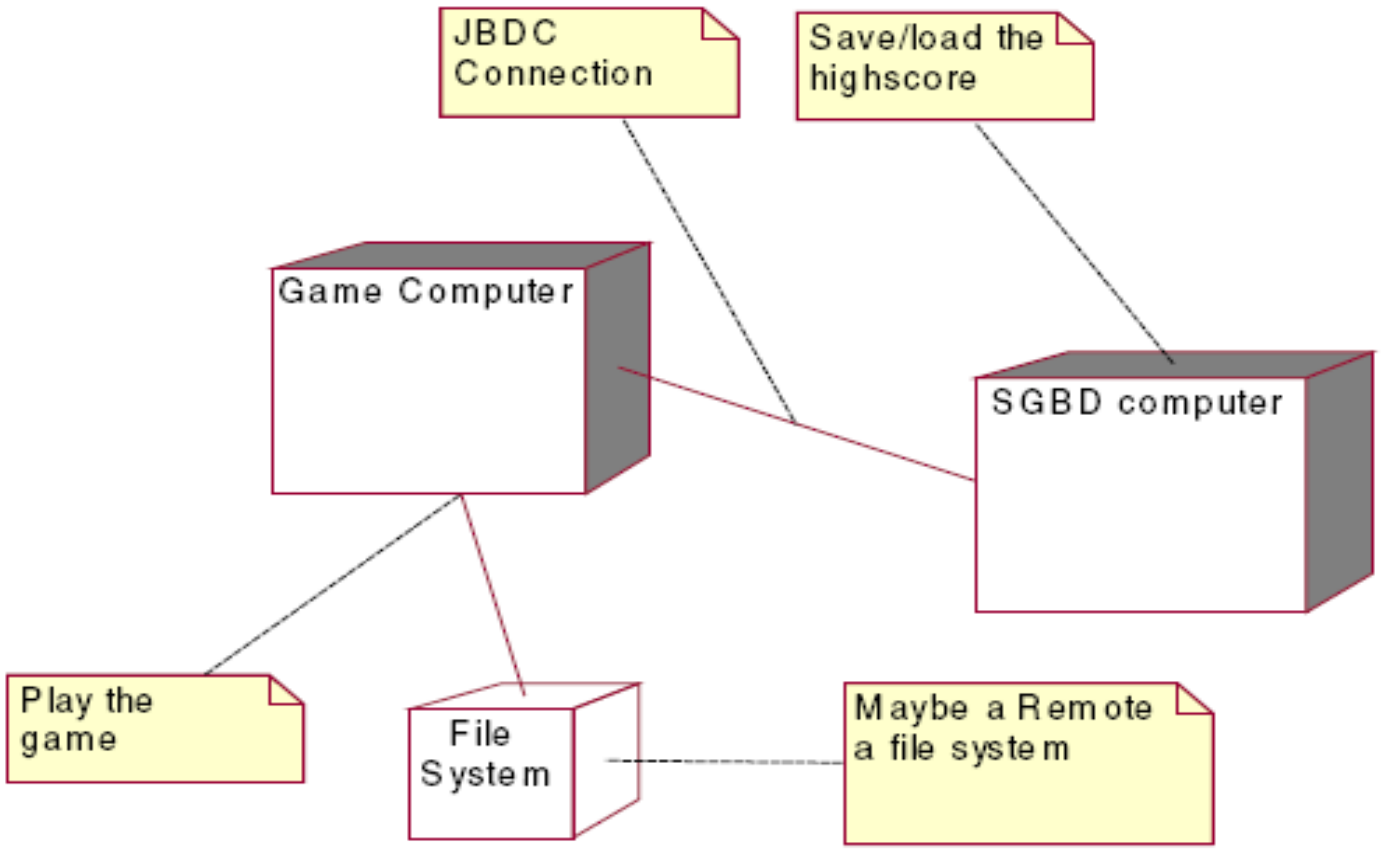
Biểu đồ tuần tự



Biểu đồ thành phần



Biểu đồ cài đặt



Reference

- Slides of Prof. Pascal Molli, Loria, France
- Object oriented Design, JEDI,
  Sun Microsystem