Chương 3

Phân tích hệ thống

(system analysis)

- Những vấn đề trong phân tích hệ thống
- Thu thập yêu cầu từ người sử dụng
- Phân tích yêu cầu
- Xác định tính năng hệ thống

Mục tiêu của phân tích hệ thống

- Khách hàng và nhà phát triển gặp nhau để thảo luận về yêu cầu của hệ thống phần mềm cần xây dựng
- Nhà phát triển tìm hiểu, phân tích và kiểm chứng lại (validate) yêu cầu và biểu diễn nó bằng mô hình phân tích
- Mô hình phân tích đặc tả toàn bộ nội dung : chức năng, dữ liệu nhập/xuất, các hoạt động của hệ thống cần phát triển

Mục tiêu của phân tích hệ thống (tt)

- Xây dựng các từ điển dữ liệu định nghĩa các khái niệm đặc thù của hệ thống, ý nghĩa, cấu trúc,…
- Thống nhất với khách hàng về mô hình và tính năng của hệ thống

Phân tích hệ thống

- Phân tích hệ thống là bước đầu tiên rất quan trọng cho dự án phát triển phần mềm

- Công việc phân tích hệ thống bao gồm
  - Thu thập yêu cầu và quy trình nghiệp vụ hiện tại
  - Phân tích và xác lập các quy trình sẽ được phát triển/thay thế bằng máy tính
  - Xác thực các yêu cầu/tính năng của hệ thống

Phân tích hệ thống (tt)

- Kết quả của việc phân tích hệ thống là các tài liệu đặc tả tính năng hệ thống. Các tài liệu này thông thường ở dạng các sơ đồ, biểu đồ,..
- Kết quả này dùng cho việc xác thực các tính năng của hệ thống với khách hàng
- Kết quả này là đầu vào của quá trình tiếp theo là thiết kế hệ thống.
- Tùy thuộc vào công nghệ phát triển mà sử dụng các phương pháp phân tích phù hợp : cấu trúc hay OOP

Những vấn đề trong phân tích hệ thống

- Cách biệt về chuyên môn của lĩnh vực cần phân tích
- Sự hiểu biết của những người end user về quy trình làm việc và khả năng ứng dụng phần mềm cho công việc của họ
- Những vấn đề về điều kiện hạ tầng hỗ trợ hoạt động của hệ thống

Những vấn đề trong phân tích hệ thống (tt)

- Tính sẵn sàng thông tin của các hệ thống đang có sẽ tương tác với hệ thống cần xây dựng
- Định hướng ứng dụng lâu dài chưa có/ chưa rõ ràng
- Công cụ/ngôn ngữ sử dụng để đặc tả hệ thống / kết quả phân tích

Quy trình phân tích hệ thống

- Các bước chính
  - Thu thập thông tin hệ thống hiện tại
  - Thu thập yêu cầu
  - Phân tích yêu cầu
  - Xác lập tính năng hệ thống
  - Xác thực tính năng hệ thống

Tìm hiểu và xây dựng lại hiện trạng của hệ thống

- Các quy trình hoạt động/nghiệp vụ
- Phương thức và ý nghĩa của các quá trình xử lý
- Dữ liệu của hệ thống
- Điều kiện hạ tầng: thiết bị, con người

Quy trình phân tích hệ thống

- Các bước chính
  - Thu thập thông tin hệ thống hiện tại
  - Thu thập yêu cầu
  - Phân tích yêu cầu
  - Xác lập tính năng hệ thống
  - Xác thực tính năng hệ thống

Xác định các yêu cầu

- Các yêu cầu về chức năng của hệ thống
- Các yêu cầu về môi trường vận hành: thiết bị, con người

Quy trình phân tích hệ thống

- Các bước chính
  - Thu thập thông tin hệ thống hiện tại
  - Thu thập yêu cầu
  - Phân tích yêu cầu
  - Xác lập tính năng hệ thống
  - Xác thực tính năng hệ thống

Phân tích các yêu cầu

- Phân tích các yêu cầu theo quy trình sử lý
- Bổ sung các quy trình cho phù hợp với máy tính
- Yều cầu bổ sung các thông tin

Quy trình phân tích hệ thống

- Các bước chính
  - Thu thập thông tin hệ thống hiện tại
  - Thu thập yêu cầu
  - Phân tích yêu cầu
  - Xác lập tính năng hệ thống
  - Xác thực tính năng hệ thống

Xác lập tính năng của hệ thống

•Xác lập các chức năng mà hệ thống sẽ bao gồm
•Xác lập các điều kiện và môi trường hoạt động

Quy trình phân tích hệ thống

- Các bước chính
  - Thu thập thông tin hệ thống hiện tại
  - Thu thập yêu cầu
  - Phân tích yêu cầu
  - Xác lập tính năng hệ thống
  - Xác thực tính năng hệ thống

Xác thực tính năng hệ thống

- Xác thực với người dùng về tính hợp lý và đầy đủ của các tính năng
- Xác thực các quy trình nghiệp vụ
- Xác thực các ràng buộc

Quy trình phân tích hệ thống

- Các bước chính
  - Thu thập thông tin hệ thống hiện tại
  - Thu thập yêu cầu
  - Phân tích yêu cầu
  - Xác lập tính năng hệ thống
  - Xác thực tính năng hệ thống

Phương pháp & Công Cụ

Phương pháp cấu trúc

Các bước được thực hiện đồng thời và xen kẽ nhau
Thường sử dụng lược đồ: DFD, ERD, STD

Phương pháp OOP

Sử dụng UML: lược đồ Use case, Class

Phân tích hệ thống theo hướng phát triển kỹ thuật lập trình cấu trúc

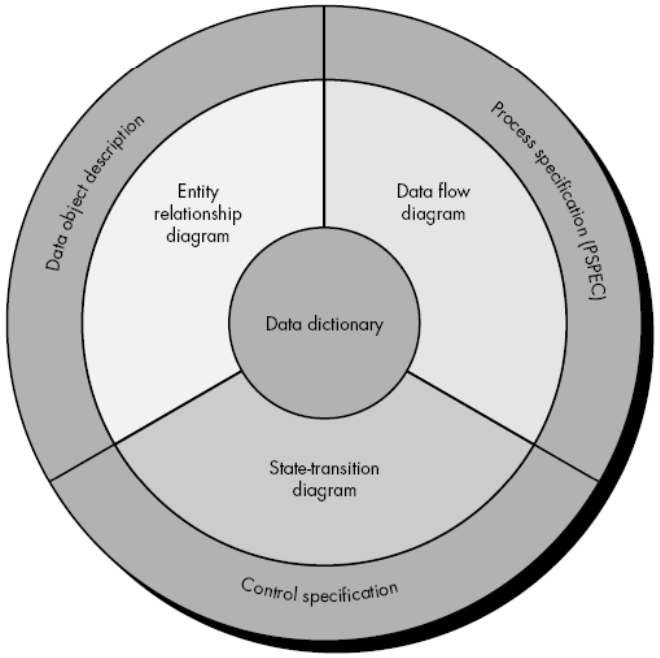
Tiếp cận của phương pháp phát triển cổ điển cho bước phân tích hệ thống

Các lược đồ DFD, STD, ERD

CÁC YẾU TỐ CĂN BẢN CỦA MÔ HÌNH

Objective:

- Describe what the customer requires
- Establish a basis for the creation of a software design
- Define a set of requirements that can be validated once the software is built

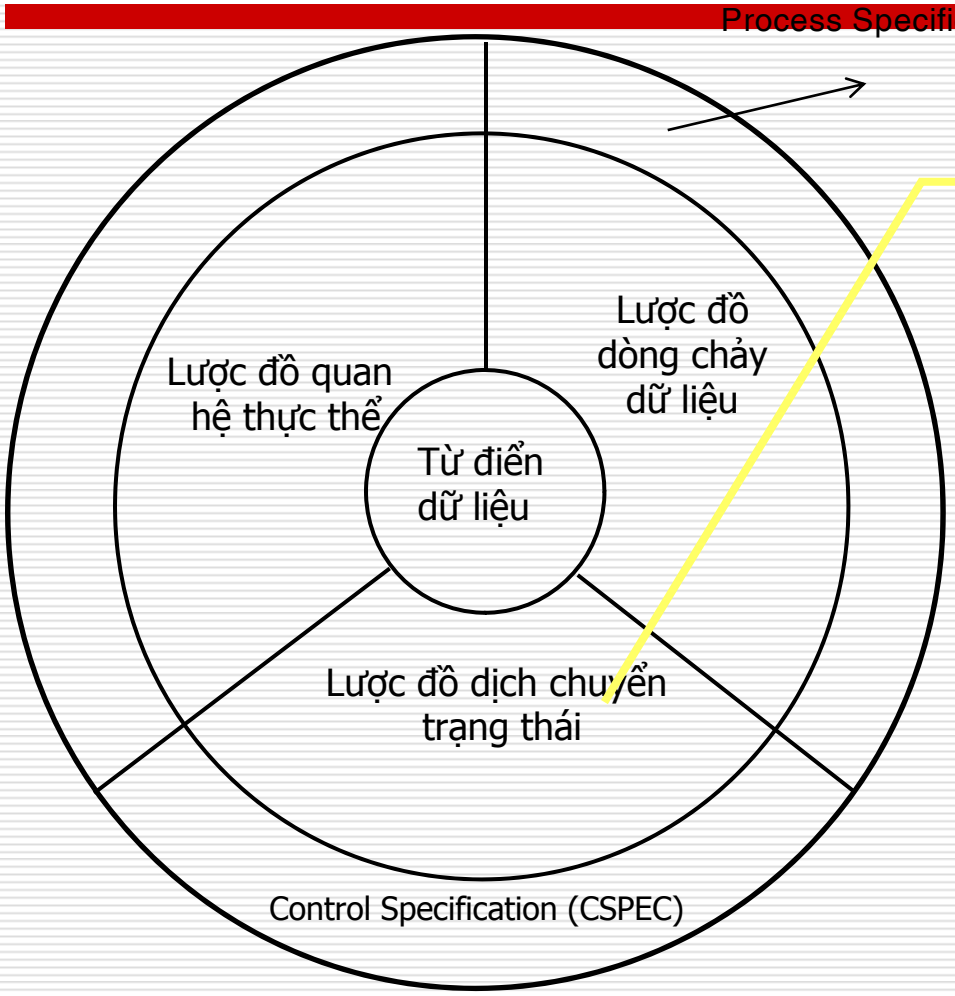


CÁC YẾU TỐ CĂN BẢN CỦA MÔ HÌNH

Lược đồ DFD

- **Mô hình chức năng và dòng thông tin: DFD, PSPEC**
- **Mô tả dòng thông tin di chuyển (flow) xuyên qua các hệ thống thiên về phần mềm.**
- **Diễn tả các tương tác xuất nhập dữ liệu với con người và các hệ thống khác**
- **Lưu đồ dòng chảy dữ liệu DFD (Data Flow Diagram) cung cấp 4 ký hiệu cơ bản để mô hình sự di chuyển của dòng thông tin**
- **Mở rộng của Ward & Mellor; Hatley & Pirbhai cho realtime**

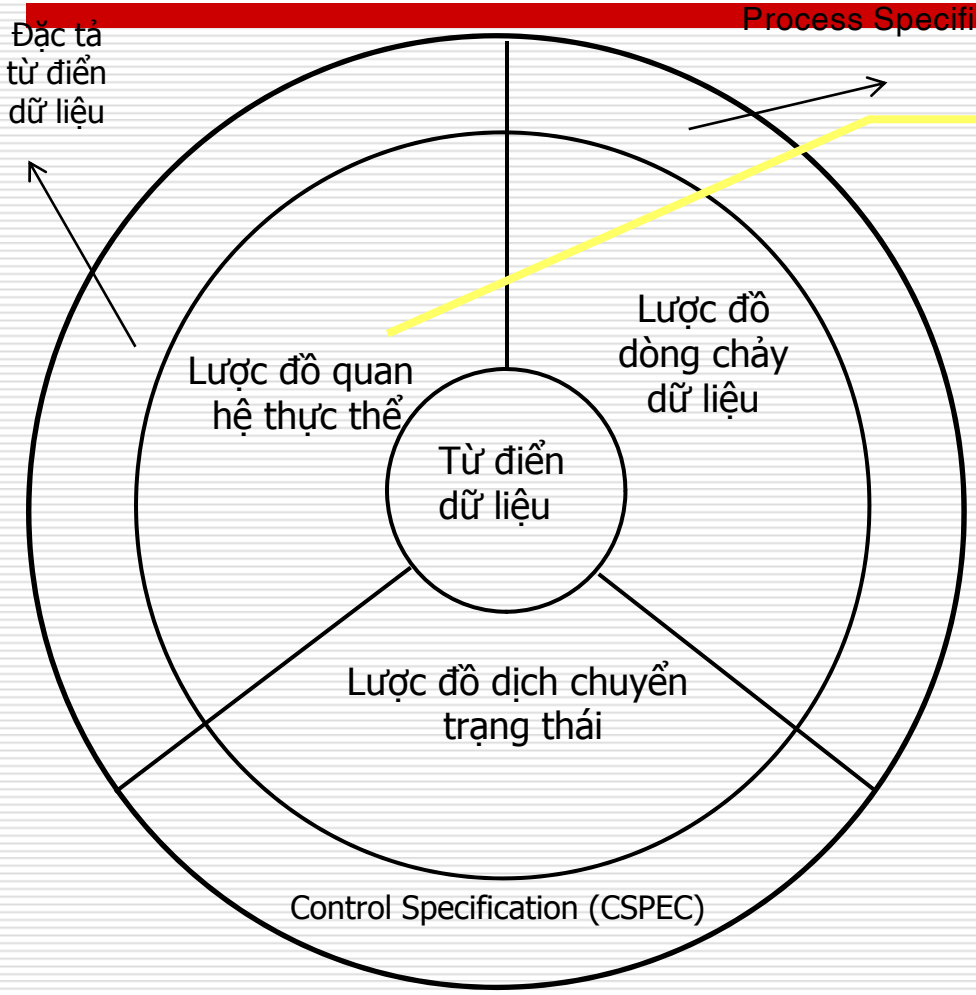
CÁC YẾU TỐ CĂN BẢN CỦA MÔ HÌNH



Lược đồ STD

- **Mô hình hành vi của hệ thống**
- **Lược đồ dịch chuyển trạng thái (STD) thể hiện**
  - **Các trạng thái khác nhau của hệ thống**
  - **Sự dịch chuyển giữa các trạng thái đó**
- **Mô tả chi tiết hơn điều kiện xảy ra của các hành vi**
- **Cung cấp một hình ảnh động về hệ thống**

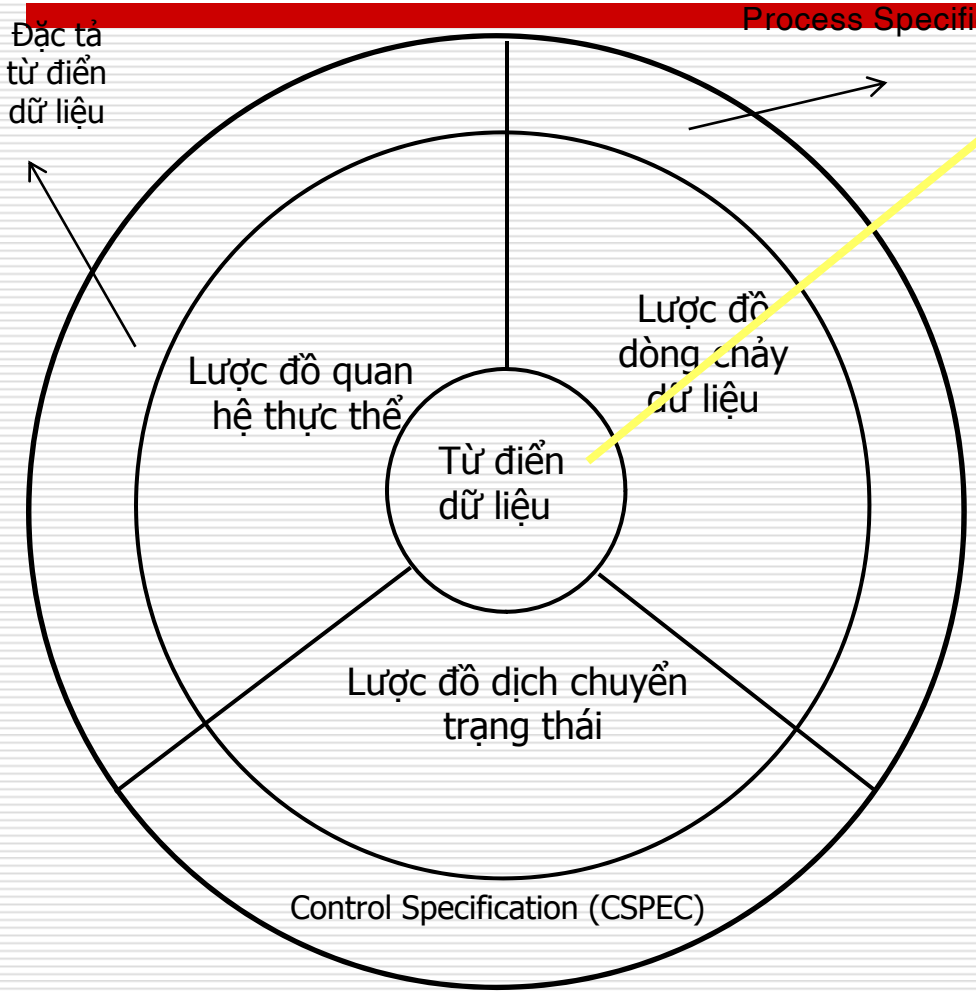
CÁC YẾU TỐ CĂN BẢN CỦA MÔ HÌNH



Lược đồ ERD

- **Đặc tả các thông tin về dữ liệu của hệ thống**
- **Cấu trúc dữ liệu**
- **Các quan hệ và ràng buộc dữ liệu**

CÁC YẾU TỐ CĂN BẢN CỦA MÔ HÌNH

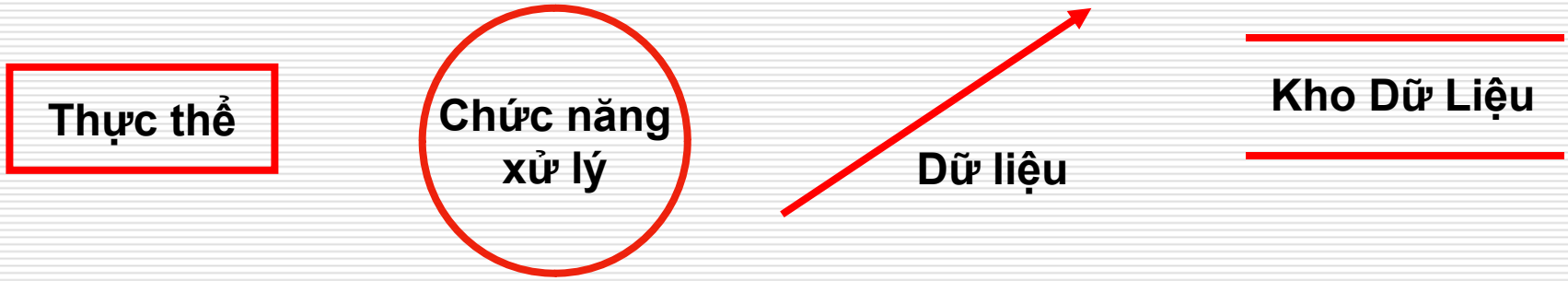


Từ điển dữ liệu

- **Làm rõ các khái niệm và thuật ngữ trong hệ thống**
- **Nếu lên ý nghĩa và phạm vi sử dụng của các khái niệm này**
- **Xác định các cấu trúc thông tin cần thiết**

LƯỢC ĐỒ DÒNG CHẢY DỮ LIỆU (DFD)

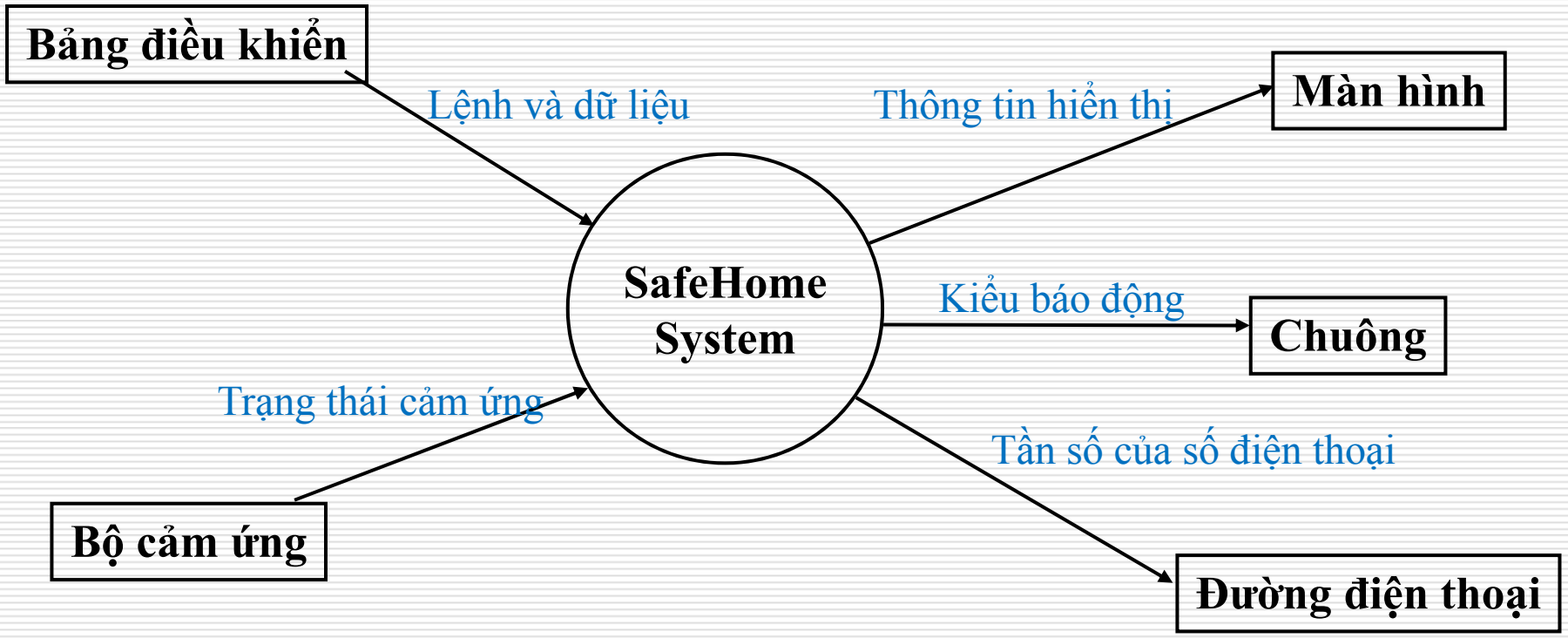
- Được xây dựng từ 4 phần tử chính
  - Thực thể: tạo ra hoặc tiêu thụ thông tin, nằm bên ngoài biên giới của phạm vi thông tin hệ thống
  - Chức năng xử lý: thực hiện chức năng nào đó, tiêu thụ và tạo ra thông tin, nằm bên trong phạm vi thông tin hệ thống
  - Thông tin hay dữ liệu
  - Kho dữ liệu: lưu trữ dữ liệu mà được sử dụng bởi nhiều chức năng xử lý



LƯỢC ĐỒ DÒNG CHẢY DỮ LIỆU (t.t)

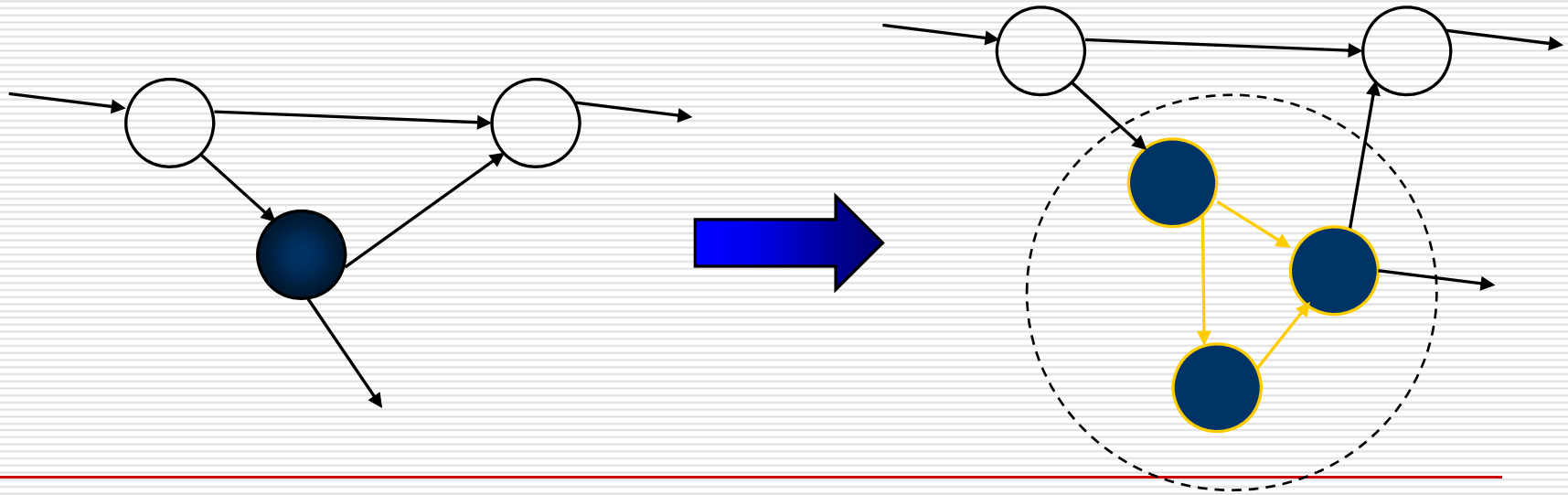
- DFD được xây dựng qua nhiều mức khác nhau: mức 0, 1, 2…
- DFD mức 0 còn được gọi là “Fundamental System Model” hay “Context Model” , đại diện cho toàn bộ hệ thống với một hình tròn duy nhất với các đường input và output data
- DFD mức sau chi tiết hơn mức trước

LƯỢC ĐỒ DÒNG CHẢY DỮ LIỆU (t.t)

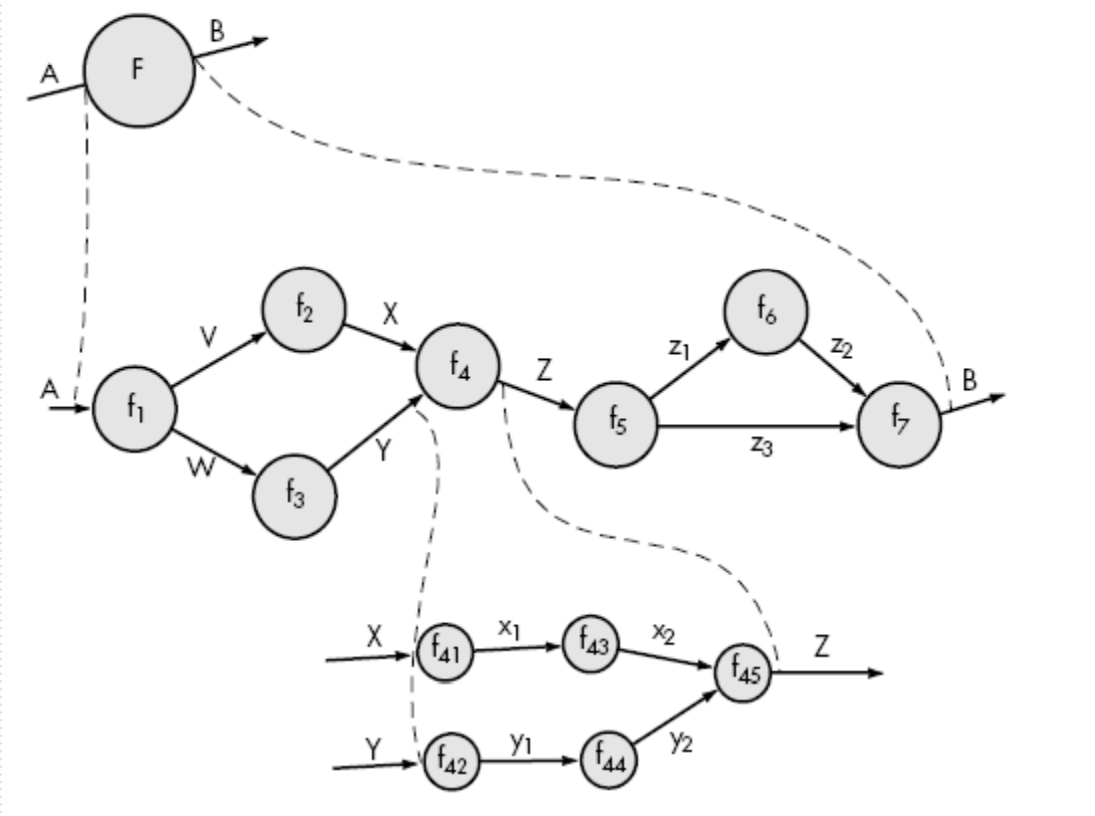


LƯỢC ĐỒ DÒNG CHẢY DỮ LIỆU (t.t)

- Process Specification (PSPEC) bổ sung cho DFD
- Tính liên tục của dòng dữ liệu



LƯỢC ĐỒ DÒNG CHẢY DỮ LIỆU (t.t)

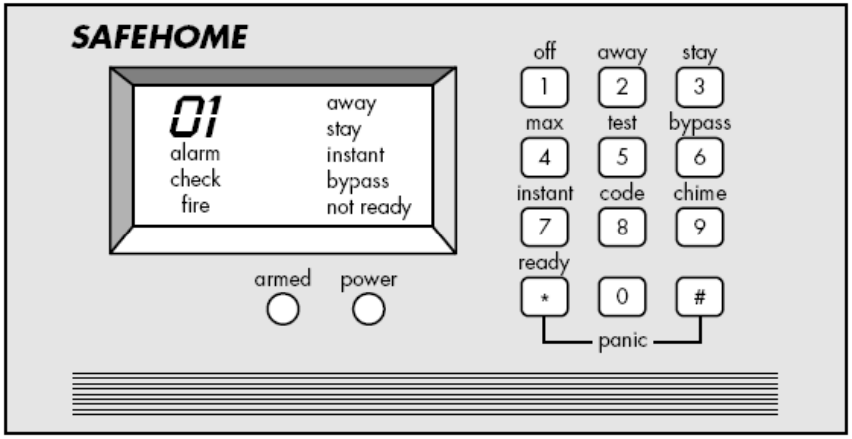


Kỹ thuật phân tích hệ thống

- Tiếp xúc, phỏng vấn các người dùng trong hệ thống thu thập các thông tin về nghiệp vụ của người dùng
- Thiết lập đoạn văn miêu tả chức năng (processing narrative) cho hệ thống cần xây dựng
- Xây dựng DFD ở các mức khác nhau
  - Thiết lập sơ đồ ngữ cảnh (DFD mức 0)
  - Phân hoạch DFD vào các mức cao hơn
  - Sử dụng phương pháp duyệt văn phạm.
  - Luôn luôn tuân theo tính liên tục của dòng dữ liệu
- Viết PSPEC cho các chức năng của DFD mức cao nhất

Xây dựng DFD – Ví dụ

- SafeHome object control panel might be
  - mounted on wall
  - size approximately 9 5 inches
  - contains standard 12-key pad and special keys
  - contains LCD display of the form shown in sketch [not presented here]
  - all customer interaction occurs through keys
  - used to enable and disable the system
  - software provides interaction guidance, echoes, and the like
  - connected to all sensors



Xây dựng DFD – Ví dụ

- When installed, SafeHome software enables the homeowner to configure the security system, monitors all sensors connected to the security system, and interacts with the homeowner through a keypad and function keys contained in the SafeHome control panel

- During installation, the SafeHome control panel is used to "program" and configure the system. Each sensor is assigned a number and type, a master password is programmed for arming and disarming the system, and telephone number(s) are input for dialing when a sensor event occurs.

Xây dựng DFD – Ví dụ

- When a sensor event is recognized, the software invokes an audible alarm attached to the system.

  After a delay time that is specified by the homeowner during system configuration activities, the software dials a telephone number of a monitoring service, provides information about the location, reporting the nature of the event that has been detected.

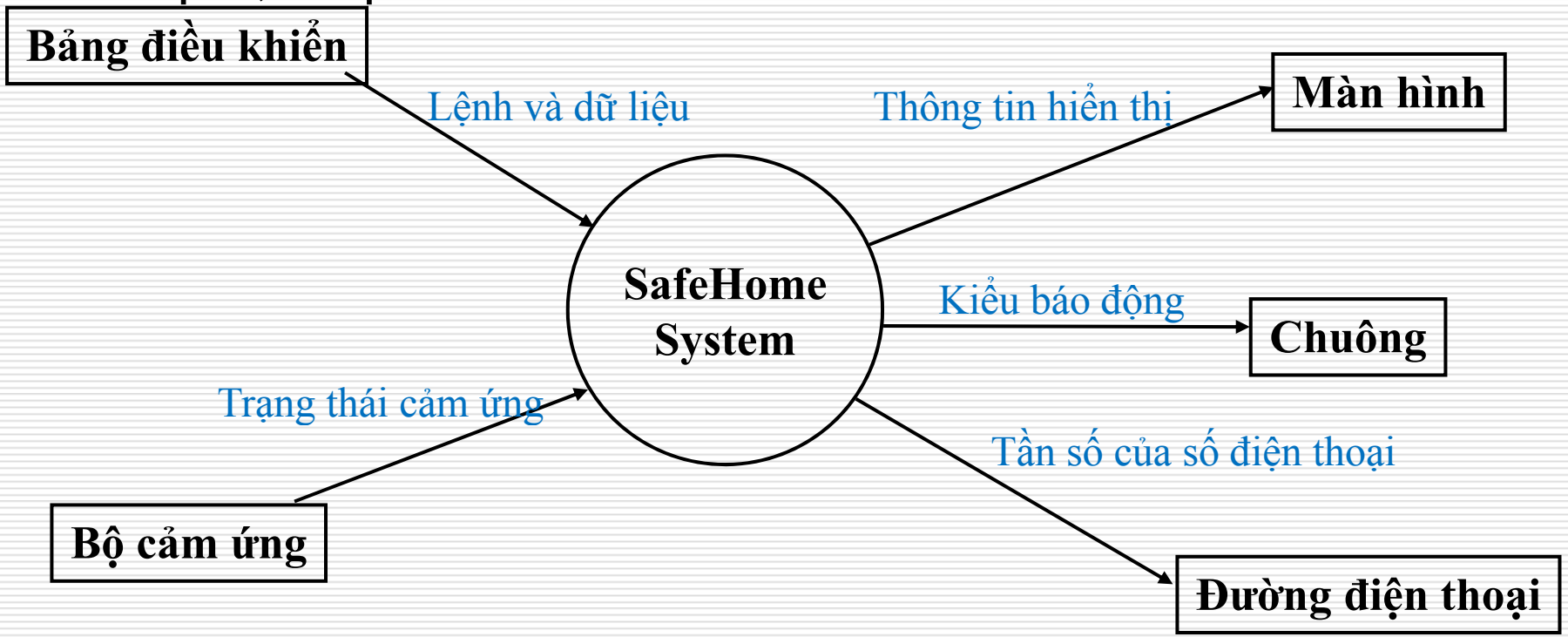
  (The telephone number will be redialed every 20 seconds until telephone connection is obtained.)

Xây dựng DFD – Ví dụ

- All interaction with SafeHome is managed by a user-interaction subsystem that reads input provided through the keypad and function keys, displays prompting messages on the LCD display, displays system status information on the LCD display.

Xây dựng DFD – Ví dụ

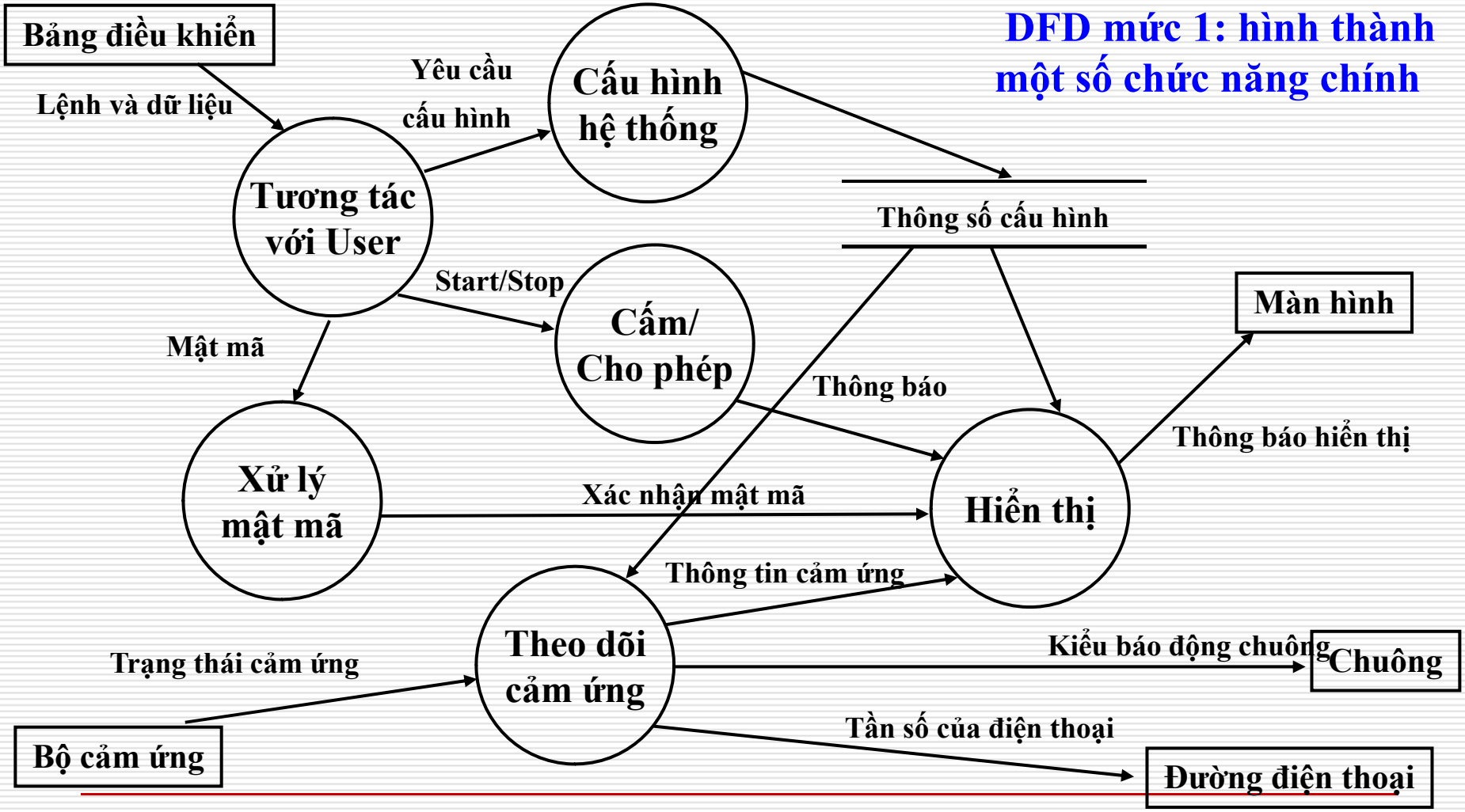
- Phần mềm SafeHome: Thiết lập đoạn văn miêu tả xử lý
- DFD mức ngữ cảnh: nhận diện các thực thể và dữ liệu input, output



Xây dựng DFD – Ví dụ

- SafeHome software *enables* the homeowner to *configure* the security system when it is *installed*, *monitors* all sensors *connected* to the security system, and *interacts* with the homeowner through a keypad and function keys *contained* in the SafeHome control panel

Xây dựng DFD – Ví dụ



PSPEC

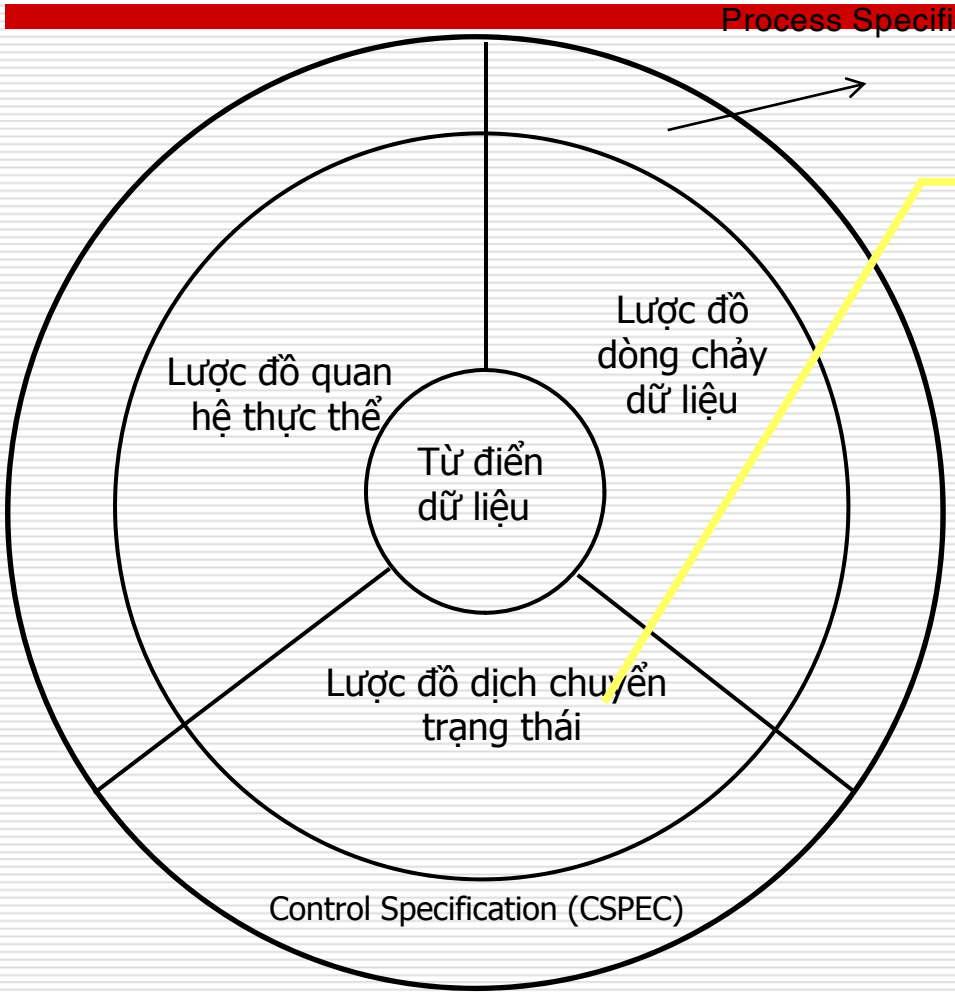
- The process specification (PSPEC) is used to describe all flow model processes that appear at the final level of refinement.

- The content of the process specification can include narrative text, a *program design language (PDL) description of the process* algorithm, mathematical equations, tables, diagrams, or charts

PSPEC

- By providing a PSPEC to accompany each bubble in the flow model, the software engineer creates a "minispec“ that can serve as a first step in the creation of the Software Requirements Specification and as a guide for design of the software component that will implement the process.

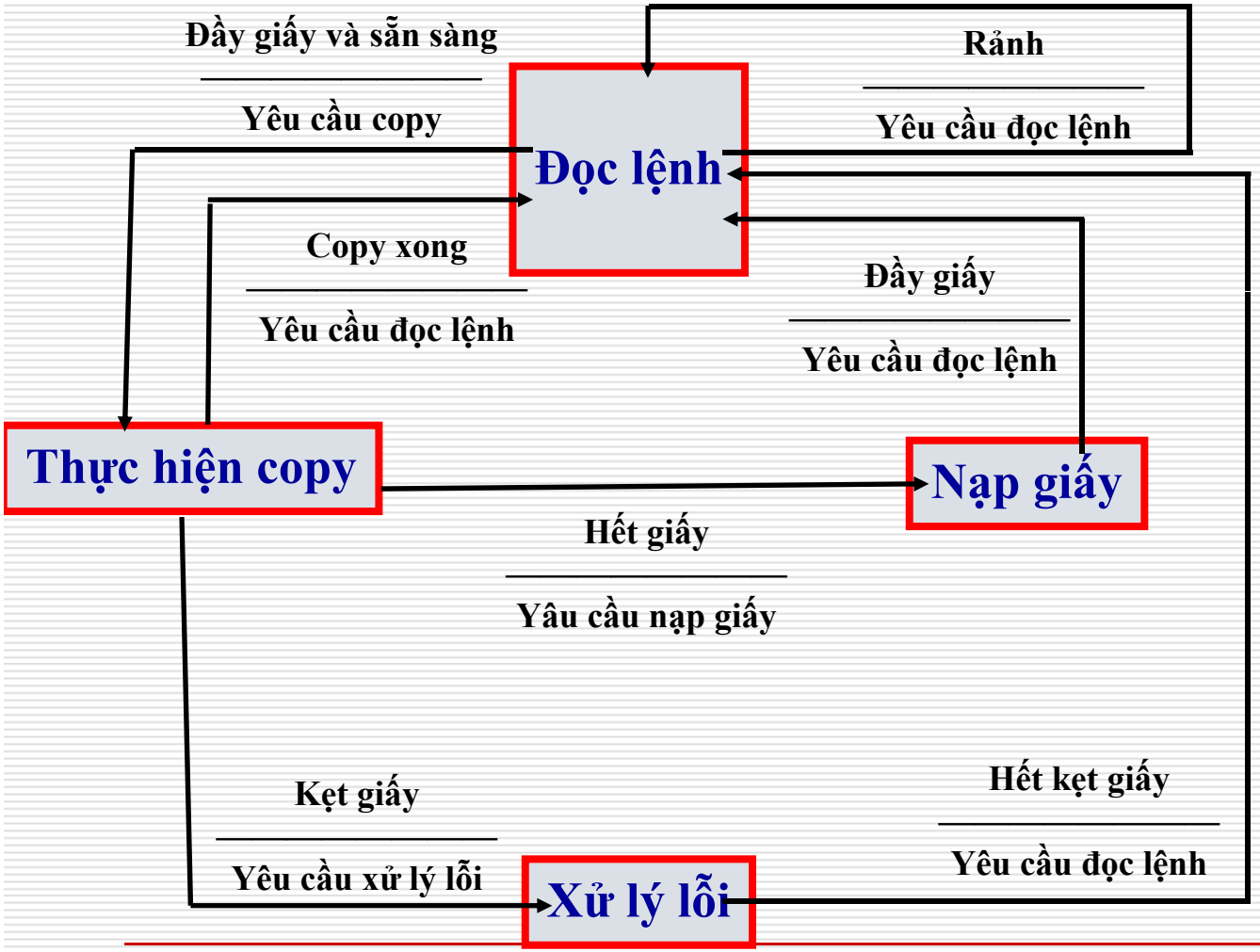
Mô hình hành vi STD



Lược đồ STD

- Mô hình hành vi của hệ thống
- Lược đồ dịch chuyển trạng thái (STD) thể hiện
  - Các trạng thái khác nhau của hệ thống
  - Sự dịch chuyển giữa các trạng thái đó
- Mô tả chi tiết hơn điều kiện xảy ra của các hành vi
- Cung cấp một hình ảnh động về hệ thống

Mô hình hành vi STD – Ví dụ



A state is any observable mode of behavior

Mô hình hành vi hệ thống máy photocopy

Tự điển dữ liệu

- Nhiều phần tử được tạo ra trong mô hình phân tích: dữ liệu, chức năng, điều khiển …

- Phải có một cách thức quản lý các phần tử đó sao cho hiệu quả → **Từ điển dữ liệu**

Tự điển dữ liệu

- Định nghĩa: Từ điển dữ liệu là một danh sách có tổ chức của tất cả các phần tử dữ liệu cần thiết cho hệ thống. Các phần tử được định nghĩa chính xác và chặt chẽ sao cho cả phân tích viên và khách hàng cùng chia sẻ một suy nghĩ về chúng.

- Từ điển dữ liệu thường được hiện thực như là một phần của công cụ CASE.

- Mỗi phần tử bao gồm những thông tin: tên, bí danh, được dùng ở đâu/như thế nào, đặc tả nội dung và thông tin phụ trợ

Tự điển dữ liệu

- The notation used to develop a content description is noted in the following table

Data Construct	Notation	Meaning
	=	is composed of
Sequence	+	and
Selection	[|]	either-or
Repetition	$\{\ \}^n$	*n* repetitions of
	( )	optional data
	* ... *	delimits comments

Tự điển dữ liệu

- The notation enables a software engineer to represent composite data in one of the three fundamental ways that it can be constructed:
  - As a sequence of data items.
  - As a selection from among a set of data items
  - As a repeated grouping of data items.

Tự điển dữ liệu – Ví dụ

- Ví dụ phần tử dữ liệu số điện thoại

Tên: Số điện thoại

Bí danh: Không

Được dùng ở đâu/như thế nào:

output của Thiết lập điều kiện báo động

input của Quay số

Đặc tả nội dung:

số điện thoại = [mở rộng địa phương | số bên ngoài]

mở rộng địa phương = [2001 | 2002 … | 2009]

số bên ngoài = 9 + [số địa phương | số đường dài]

số địa phương = tiền tố + <chuỗi 4 ký số>

số đường dài = (1) + mã vùng + số địa phương

tiền tố = [795 | 799 | 874 | 877]

Review – Phân tích hệ thống theo cấu trúc

- Phân tích yêu cầu theo pp cổ điển bao gồm:
  - Mô hình chức năng và dòng thông tin (DFD),
  - Mô hình dữ liệu (ERD)
  - và Mô hình hành vi (STD)
- Lược đồ DFD cơ bản có 4 ký hiệu và nó được mở rộng để biểu diễn được các hệ thống thời gian thực
- Xây dựng DFD mức 0 rồi đến các mức cao hơn; chú ý bảo toàn tính liên tục của dòng dữ liệu
- Từ điển dữ liệu giúp quản lý và tra cứu các phần tử dữ liệu

Phân tích hệ thống theo hướng phát triển kỹ thuật lập trình OOP (OO Paradigm)

Tiếp cận của phương pháp phát triển OOP cho bước phân tích hệ thống

AI / Vị trí như thế nào / Làm Gì / Khi nào

Các lược đồ

- Lược đồ Use-case: thu thập yêu cầu – mô hình nghiệp vụ
- Lược đồ lớp: phân tích hệ yêu cầu – mô hình phân tích

Mô hình nghiệp vụ - Thu thập yêu cầu

- Quan điểm thu thập/phân tích yêu cầu của mô hình nghiệp vụ: hệ thống gồm có AI/Làm những gì/Khi nào
- Lược đồ Use-case :
  - Actor & Use-case
  - Các mối quan hệ : Actor – Actor ; Actor-Use-case, - Use-case-Usace
- Actor xác định một bộ vai trò mà người hoặc vật sẽ đóng vai khi tương tác với hệ thống phần mềm
  - Actor nằm ngoài phạm vi của hệ thống
  - Chỉ quan tâm các thông điệp mà actor gửi hay nhận
  - Không quan tâm cấu trúc bên trong của actor
- Phân loại actor
  - Chủ yếu / Thứ yếu
  - Tích cực / Thụ động

Mô hình nghiệp vụ - Thu thập yêu cầu

- A primary actor uses the system's primary functions (e.g. a bank cashier);
- A secondary actor uses the system's secondary functions (e.g. a bank manager, system administrator);
- An active actor initiates a use-case;
- A passive actor only participates in one or more use-cases.

Nhận diện các ACTOR

- Trả lời một số câu hỏi như
  - Ai là người sử dụng chức năng chính của hệ thống ?
  - Ai cần sự hỗ trợ từ hệ thống để thực hiện công việc thường nhật của họ ?
  - Ai phải thực hiện công việc bảo dưỡng, quản trị và giữ cho hệ thống hoạt động ?
  - Hệ thống sẽ kiểm soát thiết bị phần cứng nào ?
  - Hệ thống đang xây dựng cần tương tác với những hệ thống khác hay không ?
  - Ai hoặc vật thể nào quan tâm đến hay chịu ảnh hưởng bởi kết quả mà hệ thống phần mềm tạo ra ?

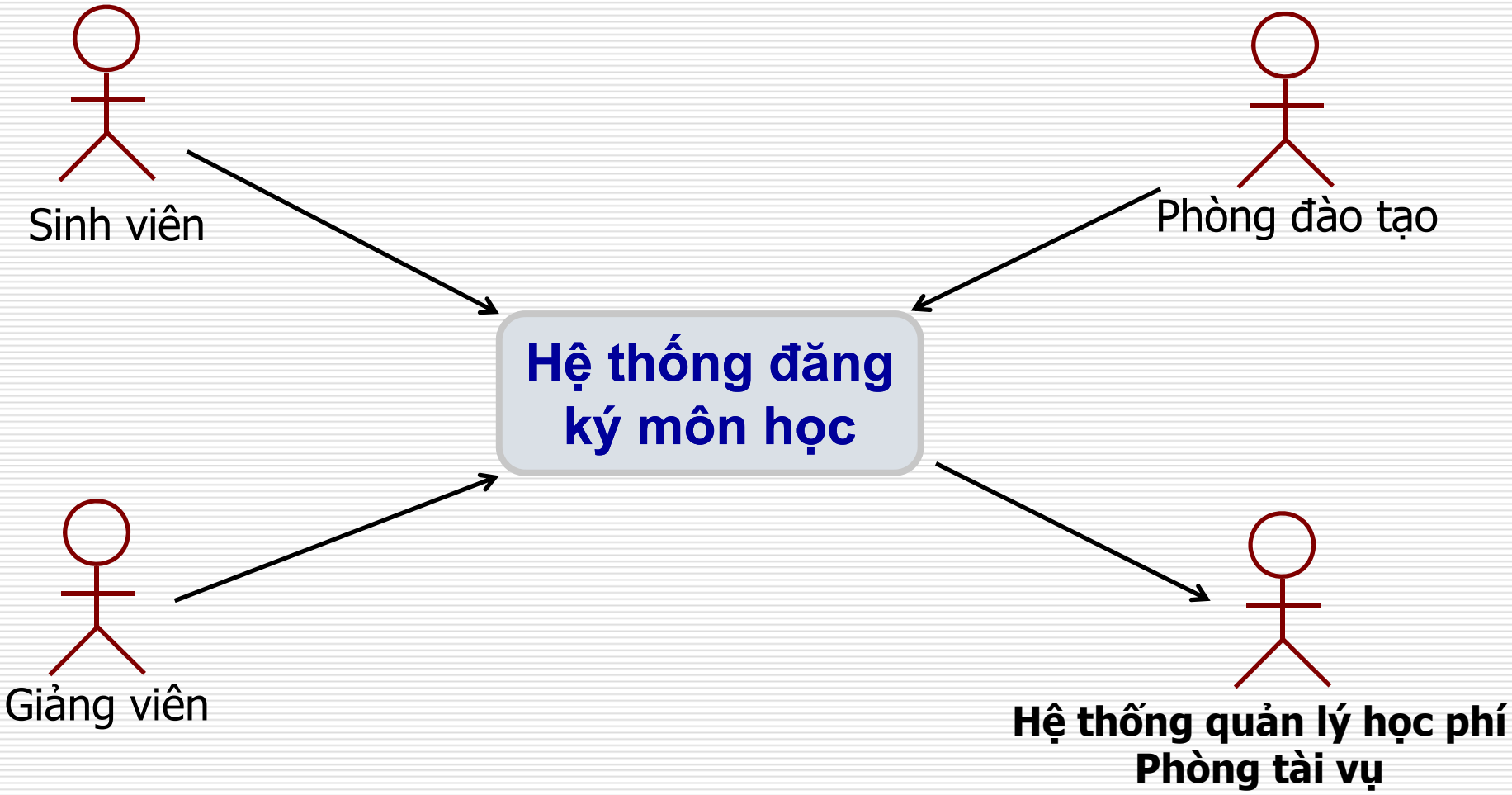
Biểu diễn ACTOR trong UML

- Actor được biểu diễn bằng ký hiệu hình người

Tên Actor

- Actor được xem là một lớp (class) có stereotype là <<actor>>
- Giữa các actor có thể có quan hệ tổng quá hoá
  - Ví dụ: Sinh viên, giảng viên và khách đều là độc giả của hệ thống quản lý thư viện: độc giả là actor tổng quát hóa của 2 actor sinh viên và giảng viên
- Ví dụ: một hệ thống đăng ký môn học trong trường đại học

Các Actor trong hệ thống đăng ký môn học

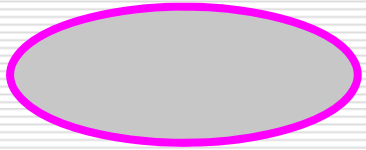


Khái niệm Use-case

- Use-case biểu diễn một chức năng của hệ thống phần mềm
- Use-case được biểu diễn bằng một chuỗi các thông điệp trao đổi bên trong hệ thống và một hoặc một số thông điệp trao đổi với actor
- Một số quy ước
  - Use-case luôn luôn được bắt đầu bằng thông điệp đến từ actor
  - Use-case phải hoàn tất: chuỗi thông điệp phải kết thúc bằng kết quả cụ thể.
- Lỗi thường gặp: chia nhỏ use-case trở thành những chức năng vụn vặt

Khái niệm Use-case

- Use-case được biểu diễn bằng hình ellipse:



Tên Use-case

- Điểm mở rộng là một vị trí trong use-case mà tại đó có thể chèn chuỗi sự kiện của một use-case khác
- Use-case có thể chứa điều kiện rẽ nhánh, xử lý lỗi, ngoại lệ...
- Minh dụ của use-case là kịch bản (scenario): miêu tả cụ thể trình tự các sự kiện xảy ra khi Use-case được thực hiện.

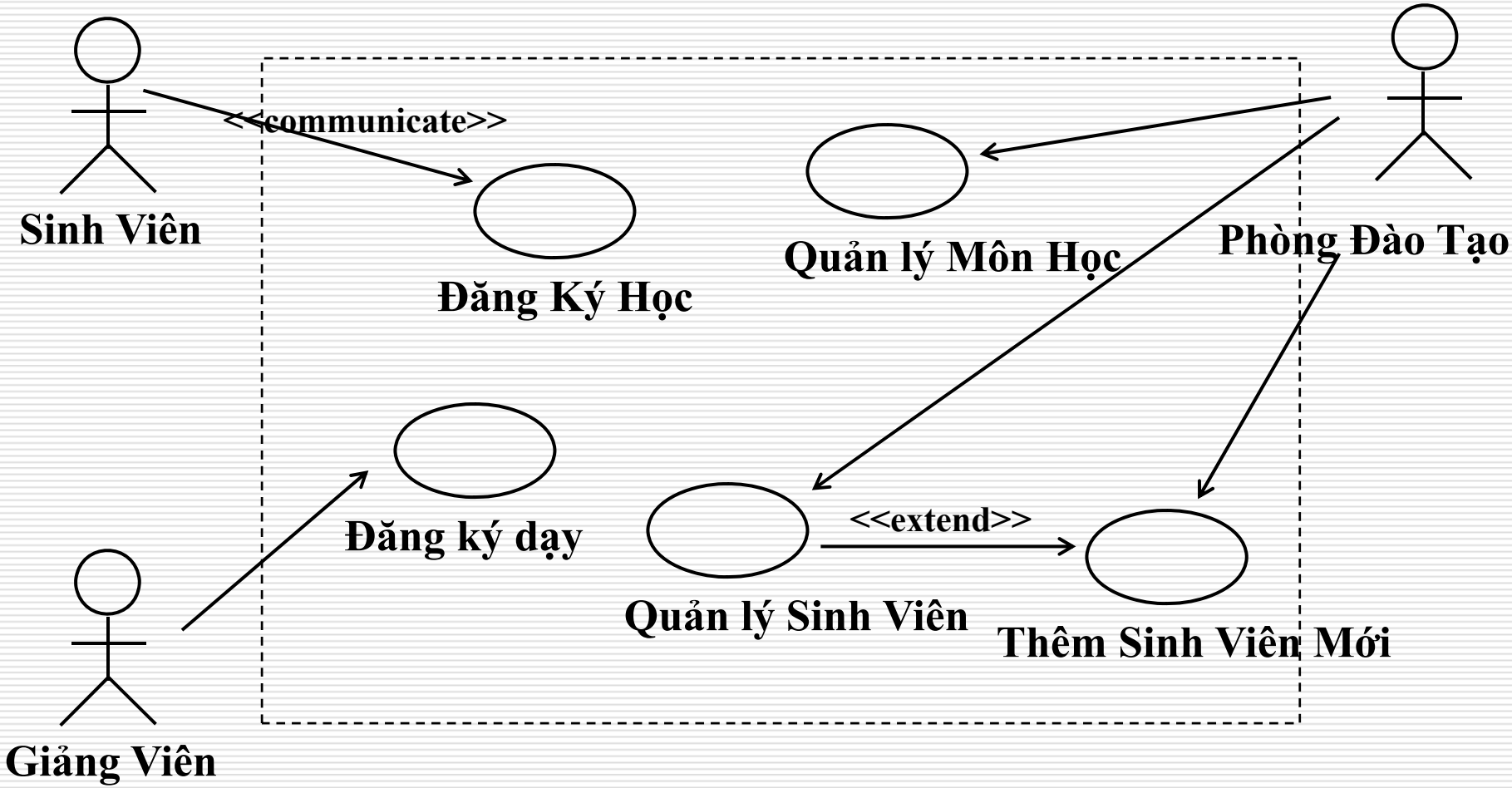
Tìm kiếm Use-case

- Trả lời một số câu hỏi như
  - Actor yêu cầu chức năng gì của hệ thống ?
  - Actor cần phải đọc, tạo, xoá, sửa đổi hoặc lưu trữ thông tin nào đó của hệ thống không ?
  - Actor cần thiết phải được cảnh báo về những sự kiện trong hệ thống, hay actor cần phải báo hiệu cho hệ thống về vấn đề nào đó không ?
  - Hệ thống có thể hỗ trợ một số công việc thường nhật của actor nào đó hay không ?
- Một số câu hỏi khác cần chú ý
  - Hệ thống cần dữ liệu input/ouput nào ? Dữ liệu đó đến từ đâu ?
  - Những khó khăn nào liên quan đến hiện thực của hệ thống hiện tại (chẳng hạn hệ thống quản lý bằng giấy tờ nên được thay thế bằng hệ thống quản lý trên máy tính) ?

Các quan hệ của Use-case

- Sau khi xác định các Actor và Use-case thì các quan hệ sẽ được thiết lập để hoàn chỉnh lược đồ Use-case
- Giữa use-case và actor thường có quan hệ liên kết
  - Use-case được Actor nào kích hoạt
- Giữa các use-case cũng có quan hệ liên kết hoặc tổng quát hoá
  - Quan hệ liên kết: extent , include
  - Quan hệ tổng quát hóa
- Ví dụ: một hệ thống đăng ký môn học theo tín chỉ trong trường đại học

Ví dụ một Use-case đơn giản



Quan hệ liên kết

- Quan hệ liên kết chỉ ra một quan hệ có ý nghĩa giữa hai bên
  - Trong thực tế: hành khách với lái xe, sinh viên với giáo viên, giảng viên với môn học …
- Một số tính chất liên quan
  - Tên của liên kết
  - Một chiều hay 2 chiều
  - Bậc: số lượng thực thể tham gia vào liên kết tại mỗi bên
- UML biểu diễn liên kết như là một đoạn thẳng (hai chiều) hoặc mũi tên (một chiều)

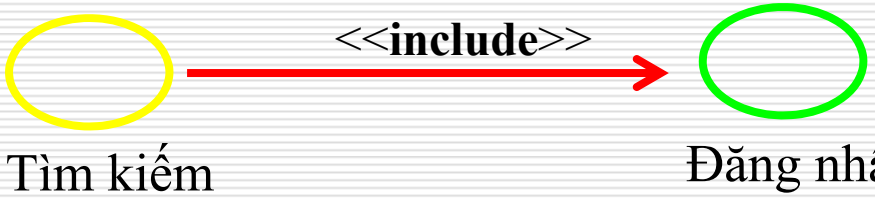
- Có thể áp dụng stereotype:
  - <<include>>
  - <<extend>>
  - <<communicate>>
  - ...

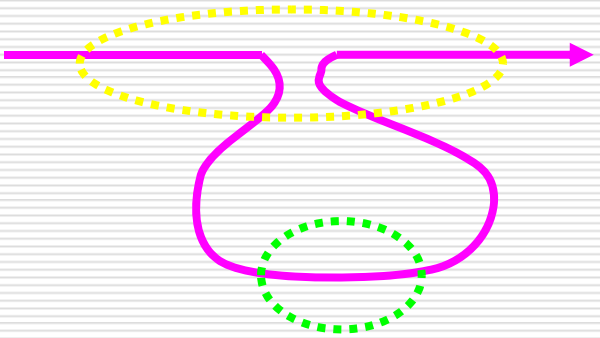
Quan hệ liên kết giữa Actor và Use-case

- Liên kết là quan hệ duy hất giữa actor & Use-case
- Liên kế có thể là 2 chiều hay 1 chiều
  - actor kích hoạt use-case và nhận kết quả về: liên kết 2 chiều
  - actor kích hoạt use-case, không quan tâm kết quả về: liên kết 1 chiều
- Quan hệ liên kết phổ biến giữa Actor & Use-case là giao tiếp: *stereotype* là <<communicate>> dùng liên kết giữa actor và use case mà nó kích hoạt

Quan hệ gộp giữa 2 Use-case

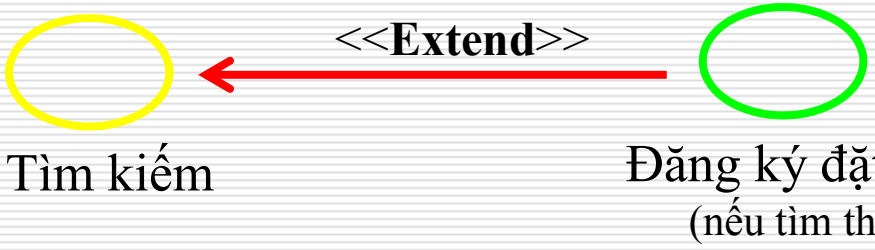
- Dùng để liên kết 2 Use-case: có stereotype là <<include>>
- Trong Use-case nguồn có điểm mở rộng mà tại đó bắt buộc phải chèn Use-case đích vào.
  - Tại điểm mở rộng, diễn tiến của use-case nguồn tạm thời ngừng lại để chuyển sang diễn tiến của use-case đích
  - Khi kết thúc use-case đích, diễn tiến của use-case nguồn lại tiếp tục

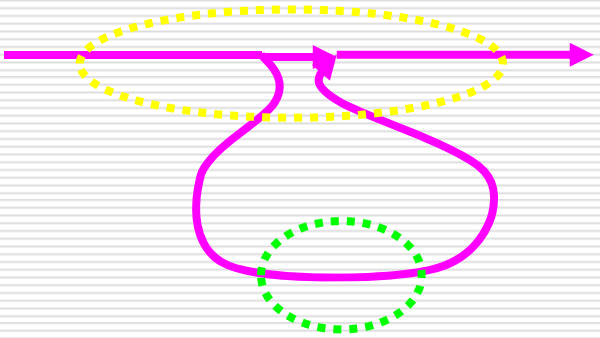




Quan hệ mở rộng giữa 2 Use-case

- Dùng để liên kết 2 Use-case: có stereotype là <<extend>>
- Trong use-case nguồn có một điểm mở rộng mà tại đó có thể (hoặc không) chèn use-case đích vào tùy thuộc vào điều kiện rẽ nhánh hoặc tương tác từ actor
  - Tại điểm mở rộng, nếu được mở rộng thì diễn tiến của use-case nguồn tạm thời ngừng lại để chuyển sang diễn tiến của use-case đích
  - Khi kết thúc use-case đích, diễn tiến của use-case nguồn lại tiếp tục





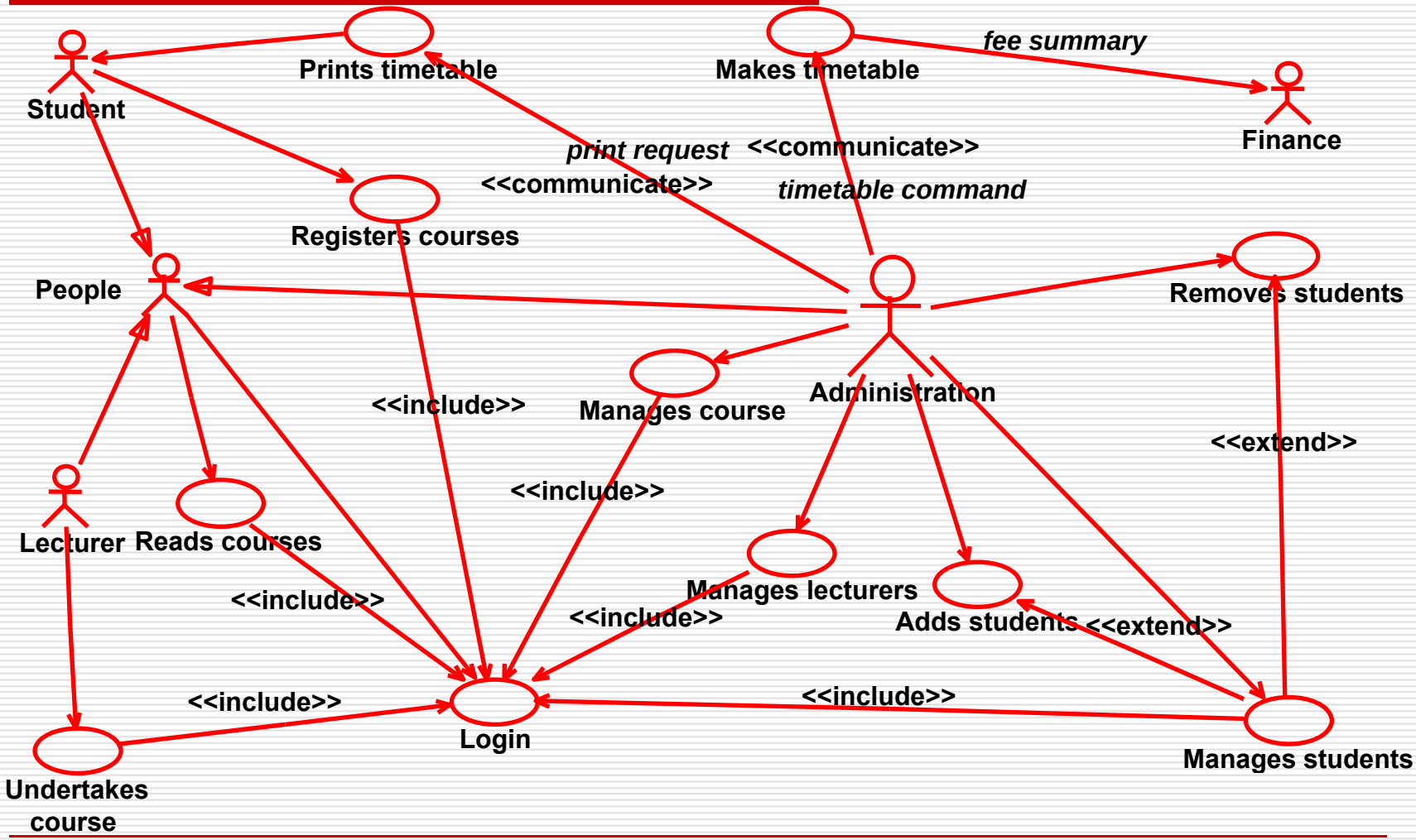
Quan hệ tổng quát hóa

- Là mối quan hệ giữa các đối tượng cùng nhóm tạo nên một đối tượng mang những tính chất chung của các đối tượng kia
- Quan hệ thống quát hóa giữa các Actor: nhiều actor có chung một số vai trò -> hình thành actor tổng quát hóa mang vai trò chung đó.
  - Ví dụ: sinh viên, giáo viên đều có chung use-case login và đều là user của hệ thống -> tạo nên actor user là tổng quát hóa của actor sinh viên và actor giáo viên
- Quan hệ thống quát hóa giữa các Use-case: khi có nhiều use-case là trường hợp cụ thể một use-case trừu tượng
  - Vì dụ: Use-case login của sinh viên , giáo viên có thể được thực hiện theo cơ chế khác nhau nhưng cùng mang chung ý nghĩa là đăng nhập → là các trường hợp cụ thể của Use-case trừu trương LOGIN

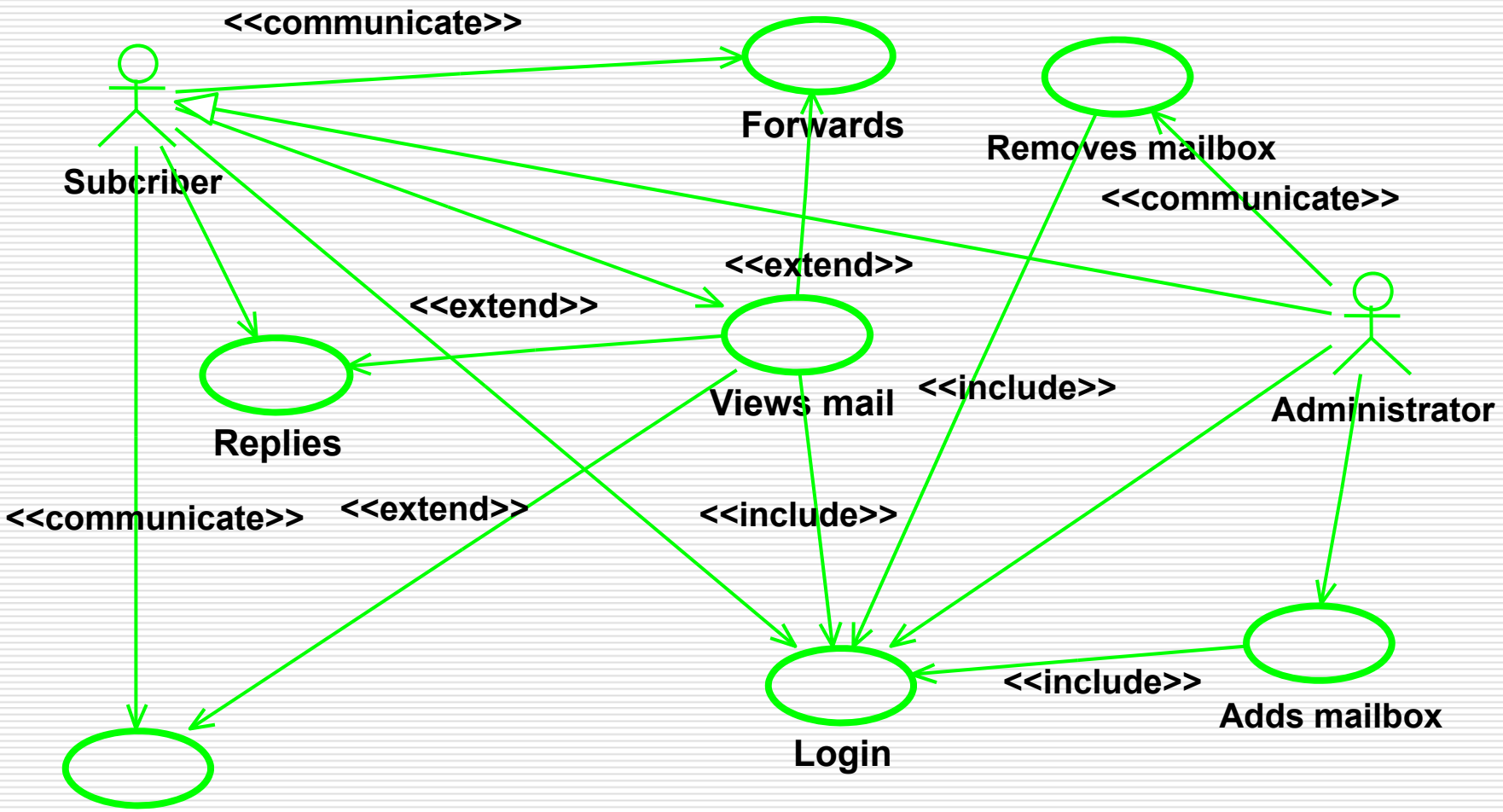
Xây dựng mô hình Use-case

- Các yêu cầu của phần mềm được mô tả trong mô hình use-case
- Mô hình use-case bao gồm lược đồ use-case và (có thể) một số package (gom một số use-case thành một bộ chức năng con của hệ thống)
- Phương pháp thực hiện:
  - Xác định các actor và use-case của hệ thống
  - Xác lập các quan hệ giữa các đối tượng này
    - Quan hệ liên kết giữa actor và use-case: một chiều hoặc hai chiều, thường có stereotype là <<communicate>>
    - Quan hệ mở rộng hay gộp giữa 2 use-case: quan hệ liên kết với stereotype <<extend>> hay <<include>>
    - Quan hệ tổng quát hoá (generalization) giữa các actor: nhiều actor có vai trò của một actor trừu tượng
    - Quan hệ tổng quát hoá giữa các use-case: nhiều use-case là trường hợp cụ thể của một use-case trừu tượng
  - Trình bày thành lược đồ use-case theo chuẩn UML
  - Có thể xác định các package

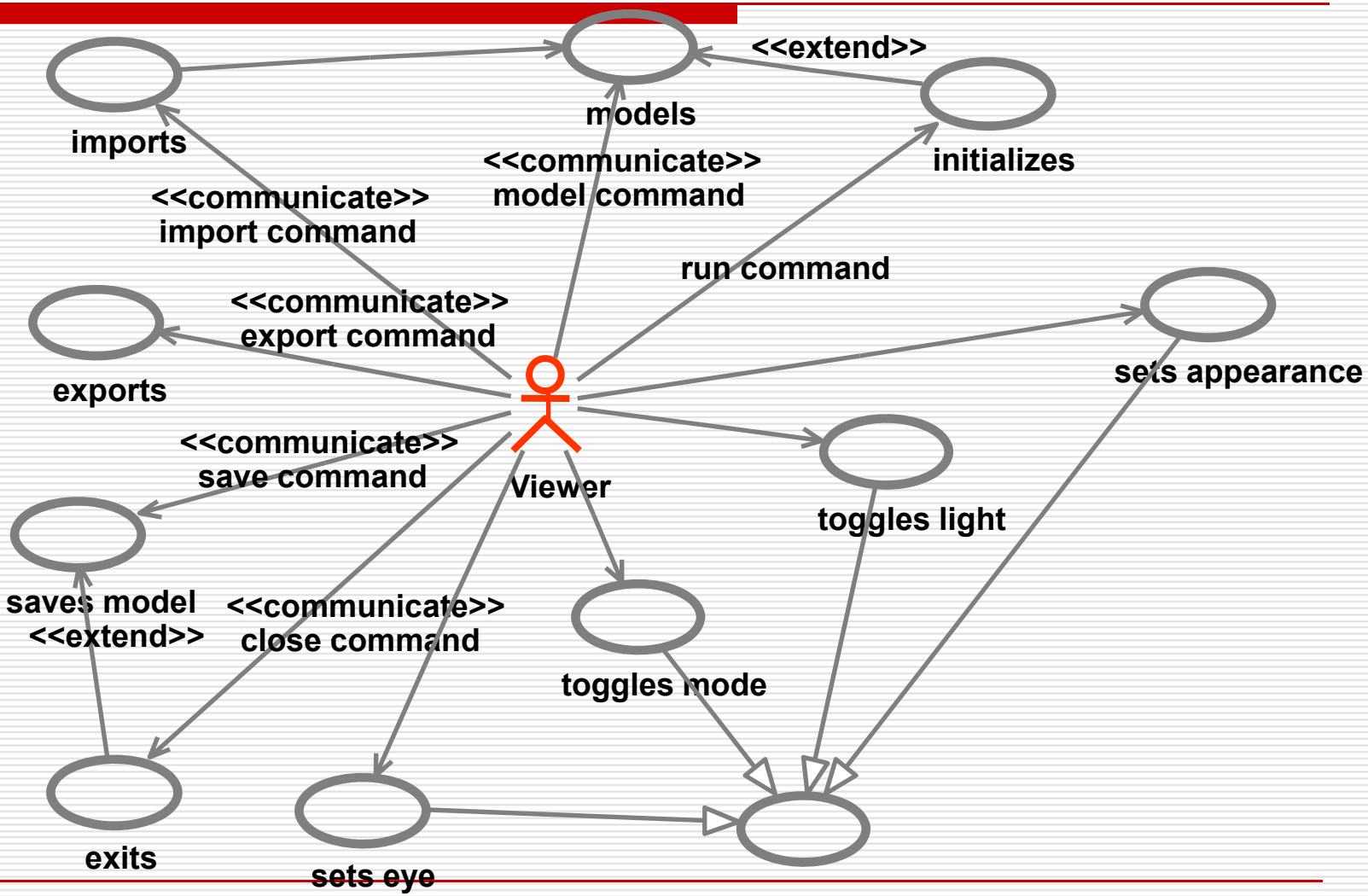
Xây dựng mô hình Use-case – VÍ DỤ



Xây dựng mô hình Use-case – VÍ DỤ



Xây dựng mô hình Use-case – VÍ DỤ



Mô hình phân tích – Phân tích yêu cầu

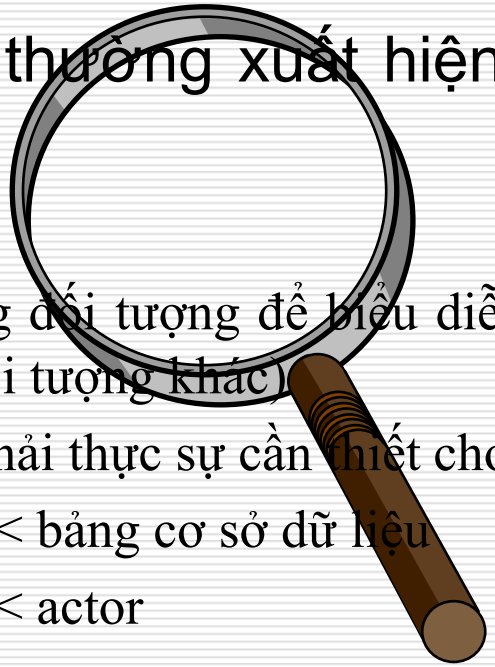
- **Mô hình nghiệp vụ** biểu diễn các chức năng phần mềm cần xây dựng dưới dạng các use-case
- **Mô hình phân tích** sẽ tìm kiếm các đối tượng "sống" trong ngữ cảnh của phần mềm
- Các đối tượng sẽ tương tác với nhau để tạo nên các chức năng mô tả bởi use-case
- Lược đồ Class phân tích diễn tả cấu trúc, mối quan hệ giữa các đối tượng/lớp trong hệ thống
- Chưa quan tâm đến hành vi cụ thể và nhiệm vụ chi tiết của chúng trong ngữ cảnh của hệ thống
- Nguyên tắc: mô hình phân tích phải độc lập với o/s, ngôn ngữ lập trình, công cụ phát triển

Xây dựng mô hình phân tích

- Mô hình phân tích được diễn đạt trong UML bằng lược đồ lớp phân tích (Class diagram)
- Các công việc xây dựng lược đồ phân tích bao gồm
  - Tìm kiếm các đối tượng / lớp trong hệ thống
    - Đối tượng / lớp thực thể
    - Đối tượng / lớp biên
    - Đối tượng / lớp control
  - Xác định các thuộc tính của đối tượng / lớp
  - Xác định các tác vụ của đối tượng / lớp
  - ả hận diện các lớp trừu tượng qua mối quan hệ tổng quát hóa
  - Xác lập các mối quan hệ giữa các lớp:
    - Tổng quát hoá (generalization)
    - Liên kết (association)
    - Bao gộp (aggregation)
- Biểu diễn thành lược đồ lớp phân tích

Nhập diện đối tượng / lớp

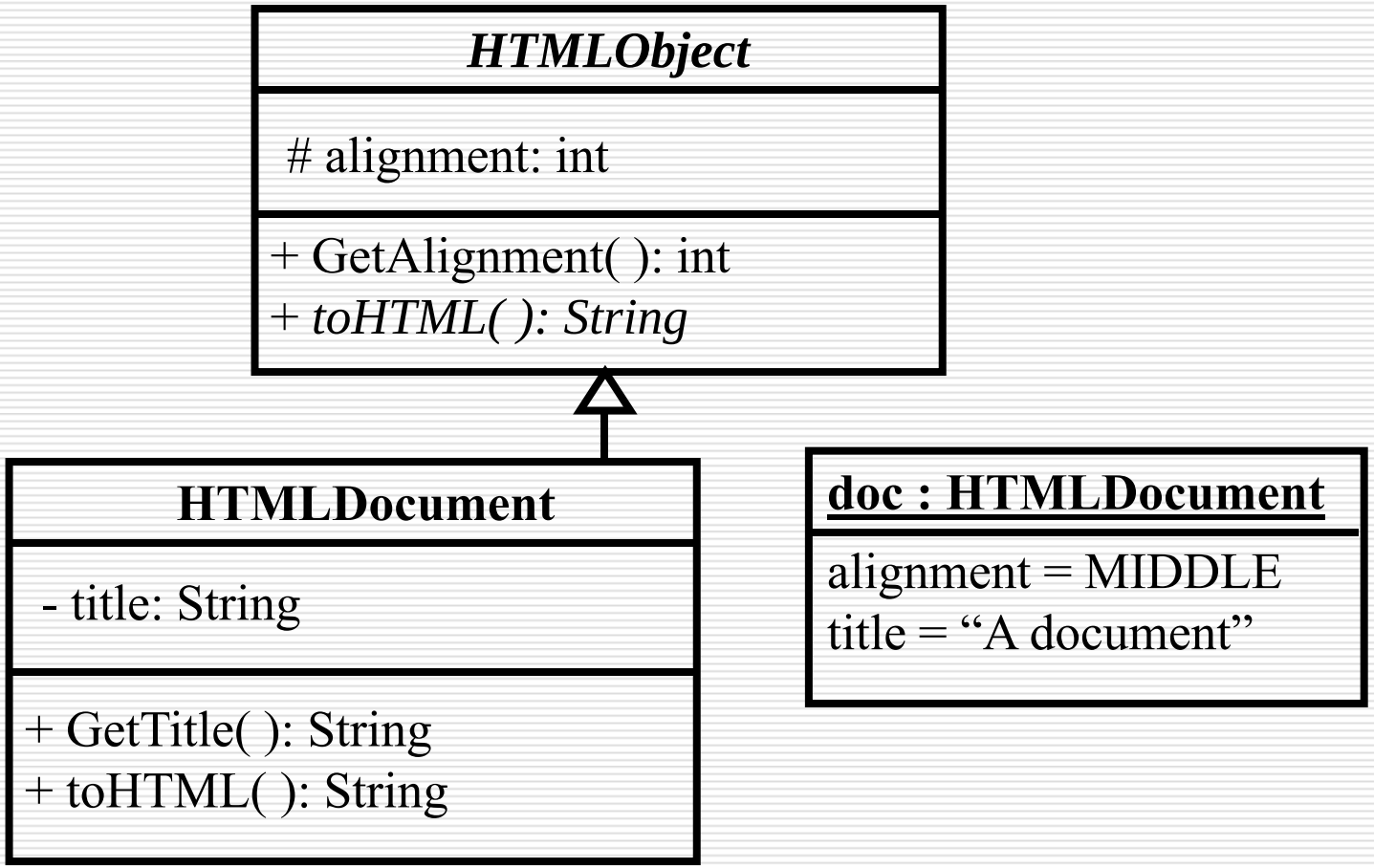
- Dựa vào đặc tả của từng use-case để tìm kiếm các đối tượng
- Các đối tượng thường xuất hiện trong các danh từ hay nhóm danh từ
- Một số lưu ý
  - Không nên dùng đối tượng để biểu diễn một dữ liệu đơn (nên xem là thuộc tính của đối tượng khác)
  - Đối tượng/lớp phải thực sự cần thiết cho sự hoạt động của hệ thống
  - Đối tượng/lớp >< bảng cơ sở dữ liệu
  - Đối tượng/lớp >< actor



Nhận diện và biểu diễn đối tượng / lớp

- Phân loại đối tượng/lớp
  - Đối tượng thực thể (entity): biểu diễn các thông tin thiết yếu của hệ thống, có thể được lưu trong cơ sở dữ liệu
  - Đối tượng biên (boundary): thực hiện chức năng giao tiếp với actor
  - Đối tượng điều khiển (control): điều khiển các đối tượng khác
- Trong UML, lớp được biểu diễn bằng một hình chữ nhật gồm 3 phần: tên, các thuộc tính và các tác vụ
- Có thể áp dụng stereotype cho lớp: <<entity>>, <<boundary>>, <<control>>...
- Đối tượng cũng được biểu diễn bằng hình chữ nhật, thông thường gồm 2 phần: tên đối tượng + tên lớp (được gạch chân), giá trị các thuộc tính (trạng thái của đối tượng)

Biểu diễn lớp / đối tượng



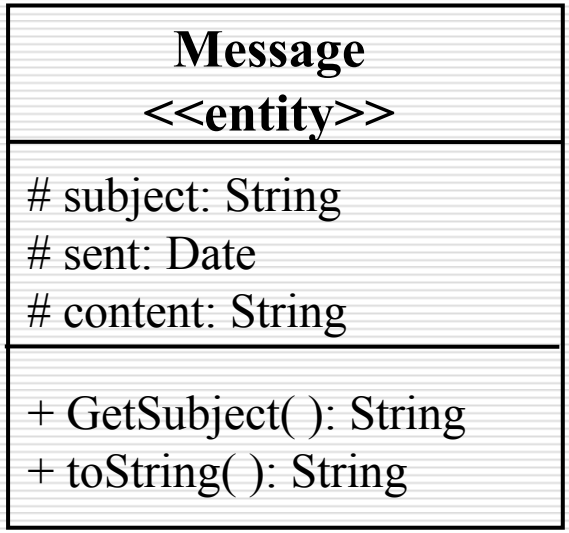
Đối tượng / lớp thực thể

- Biểu diễn cho các thực thể xuất hiện một cách tự nhiên trong hệ thống
- Thông tin về các đối tượng thực thể có thể phải được lưu trữ lâu dài (database, file...)
- Trong UML, được gán stereotype <<entity>>
- Dễ nhận diện các thuộc tính của chúng

Ví dụ:

• Đối với hệ thống đăng ký môn học hệ tín chỉ qua WEB, nhận diện các đối tượng thực thể như: thông tin SV, thông tin GV, nhóm lớp học, đăng ký nhóm, sổ tay sinh viên …

• Đối với hệ thống mail, nhận diện các đối tượng thực thể như: hộp thư, thông điệp mail…



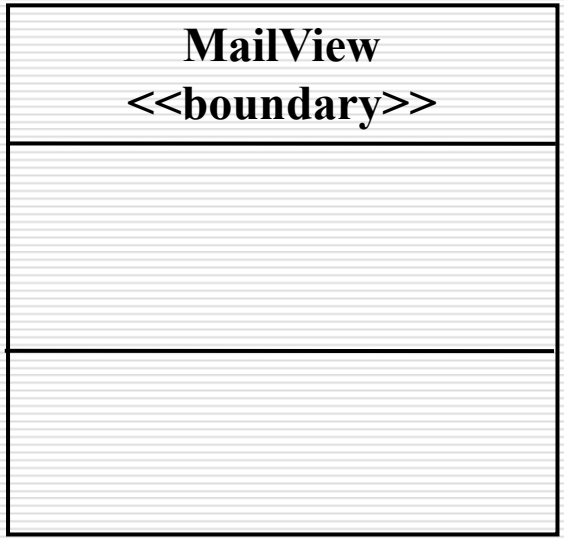
Đối tượng / lớp biên

- Thực hiện chức năng giao tiếp với actor
- Thường chứa các phần tử hoặc điều khiển giao diện người dùng (nút nhấn, hộp danh sách, tuỳ chọn, menu...)
- Trong UML, được gán stereotype <<boundary>>
- Khó nhận biết các thuộc tính và tác vụ trong mô hình phân tích

Ví dụ:

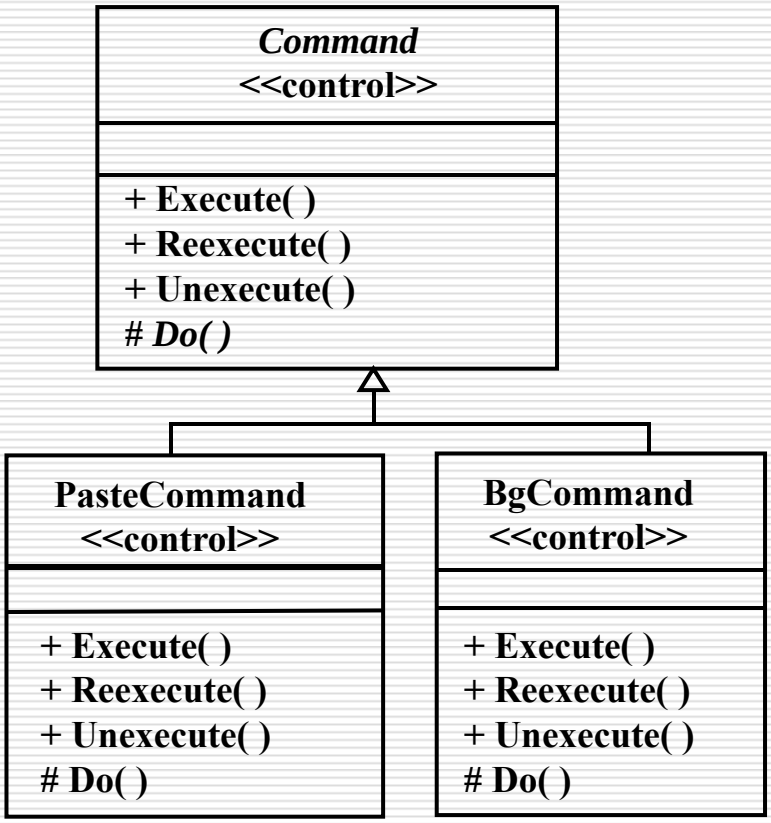
•Đối với hệ thống đăng ký môn học hệ tín chỉ qua WEB, nhận diện các đối tượng biên như: RegisterForm, StudentForm…

• Đối với hệ thống mail, nhận diện các đối tượng biên như: MailView, MailCompose...



Đối tượng / lớp điều khiển

- Có nhiệm vụ điều khiển các lớp khác hoặc
- Những lớp không phải là lớp thực thể và lớp biên
- Trong UML, được gán stereotype <<control>>
- Lớp biên thường có quan hệ liên kết hoặc phụ thuộc với các lớp khác
- Ví dụ:
- Đối tượng biểu diễn một số lệnh thông thường như cắt, dán, thay đổi thông số nhìn trong hiển thị đồ hoạ …



Nhận diện các thuộc tính

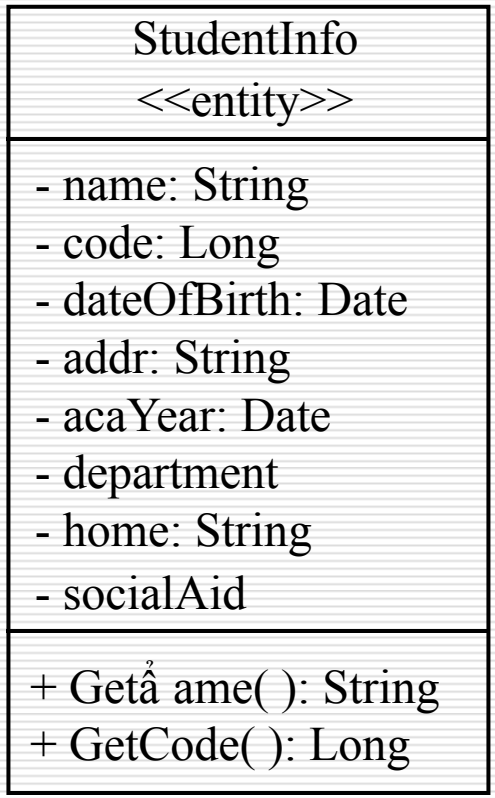
- Dựa vào đặc tả của từng use-case, tìm kiếm các danh từ hoặc nhóm danh từ liên quan đến đối tượng đang xét
- Trả lời câu hỏi: những thành phần nào cấu thành đối tượng đang xét ?
  - Lưu ý: cùng một đối tượng trong các ngữ cảnh khác nhau chúng ta có thể tìm được các thuộc tính khác nhau
- Nên xác định (tuy nhiên không bắt buộc) trong mô hình phân tích
  - Kiểu của thuộc tính: một số kiểu cơ bản
  - Bậc của thuộc tính: số ít hoặc số nhiều
  - Visibility của thuộc tính: mức độ cho phép truy xuất thuộc tính từ bên ngoài
- UML: thuộc tính được miêu tả tường minh hoặc thông qua quan hệ với các lớp khác

Xác định mức độ truy cập của thuộc tính

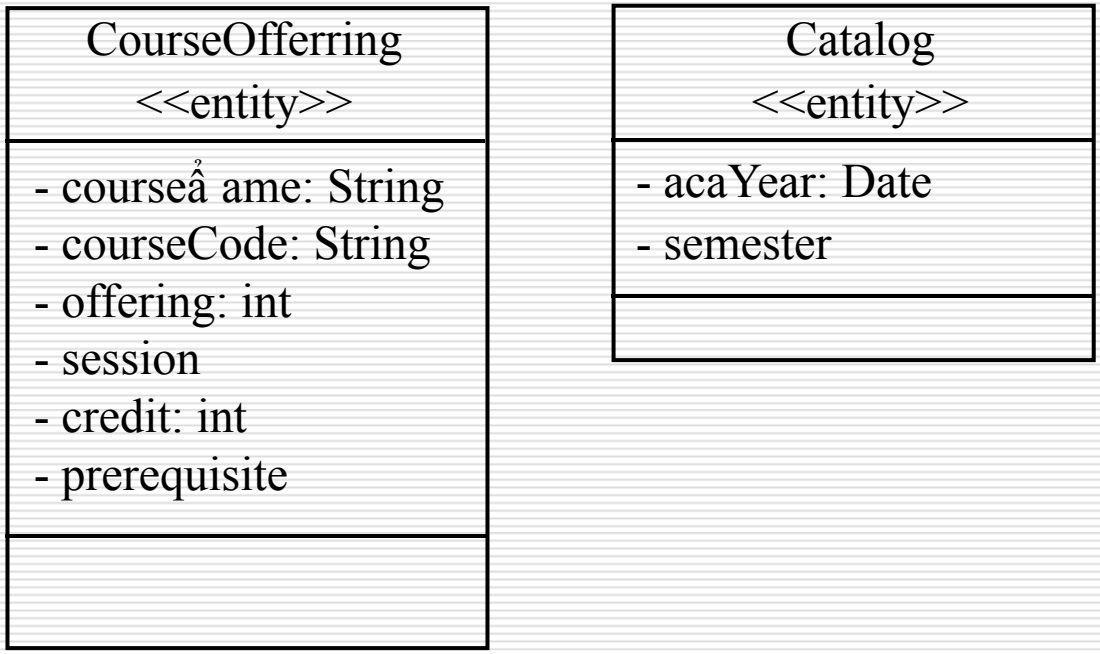
- Mức độ truy cập và phạm vi mà thuộc tính đó có thể được tham khảo đến trực tiếp
- UML định nghĩa 3 mức độ truy xuất thuộc tính (visibility)
  - public (+): có thể truy xuất thuộc tính từ tất cả các vị trí khác nhau
  - protected (#): bản thân lớp đang xét và các lớp con của nó có thể truy xuất thuộc tính
  - private (-): chỉ có lớp đang xét có thể truy xuất thuộc tính
- Thông thường nên đặt mức độ truy xuất thuộc tính là **private** hoặc **protected** (cho các lớp cơ sở), không nên là public.
- Thuộc tính nên được truy xuất thông qua tác vụ **get/set**

Ví dụ về nhận diện các thuộc tính

- Hệ thống đăng ký môn học hệ tín chỉ qua WEB - Nhận diện các thuộc tính cho các đối tượng: StudentInfo, LecturerInfo
- Chú ý các mức độ truy cập của các thuộc tính
- Các tác vụ phát sinh trong khi nhận diện các thuộc tính → như Get/Set



Ví dụ về nhận diện các thuộc tính



- Hệ thống đăng ký môn học hệ tín chỉ qua WEB
- Nhận diện các thuộc tính cho các đối tượng:

CourseOffering,

Catalog

Nhận diện các tác vụ

- Dựa vào đặc tả của từng use-case, tìm kiếm các *động từ* hoặc *nhóm động từ* liên quan đến đối tượng đang xét
- Chú ý xem đối tượng được tạo ra và bị huỷ bỏ đi như thế nào ? Trong thời gian đó nó gửi/nhận thông điệp ra sao ?
- Các đối tượng biên có các tác vụ nhận lệnh từ actor.
- Xem xét mức độ truy xuất của tác vụ tương tự như đối với các thuộc tính; các tác vụ thường có visibility là **+** hoặc **#**
- Một số tác vụ không xuất hiện một cách tự nhiên trong mô hình phân tích → mô hình thiết kế sẽ nghiên cứu kỹ trách nhiệm và hành vi của từng đối tượng

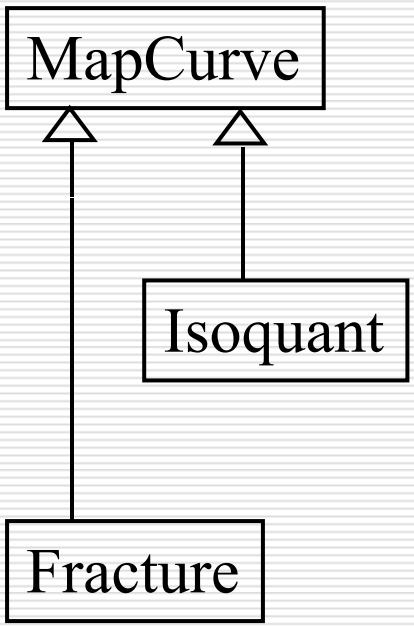
Nhận diện lớp cơ sở

- Lớp cơ sở (base class) được nhận diện sau khi đã nhận diện các lớp cụ thể
- Sự xuất hiện của lớp cơ sở làm cho mô hình phân tích có tính dùng lại cao (reusability) và dễ mở rộng (scalability)
- UML hỗ trợ quan hệ tổng quát hoá (generalization)
- Lớp cơ sở trừu tượng (không thể cụ thể hoá tạo ra đối tượng) có tên in nghiêng
- Lớp cơ sở được hình thành bằng cách xác lập các quan hệ tổng quát hóa của các lớp cụ thể có chung một số thuộc tính và/hay một số tác vụ

Nhận diện lớp cơ sở (tt)

- Đối với các đối tượng/lớp thực thể, tìm các thuộc tính chung để hình thành lớp cơ sở
- Ví dụ
  - Trong hệ thống quản lý thư viện qua WEB: các đối tượng Book, Magazine có một số thuộc tính chung hình thành lớp LibraryItem
  - Đối với hệ thống đăng ký môn học tín chỉ qua WEB: lớp PeopleInfo là lớp cơ sở của StudentInfo và LecturerInfo
  - Chương trình vẽ bề mặt địa hình: lớp MapCurve là lớp cơ sở của đường đồng mức Isoquant và đứt gãy Fracture
- Giữa lớp cơ sở và các lớp cụ thể có mối quan hệ tổng quát hóa có thể biểu diễn được trong UML

Biểu diễn lớp cơ sở và quan hệ tổng quát hóa

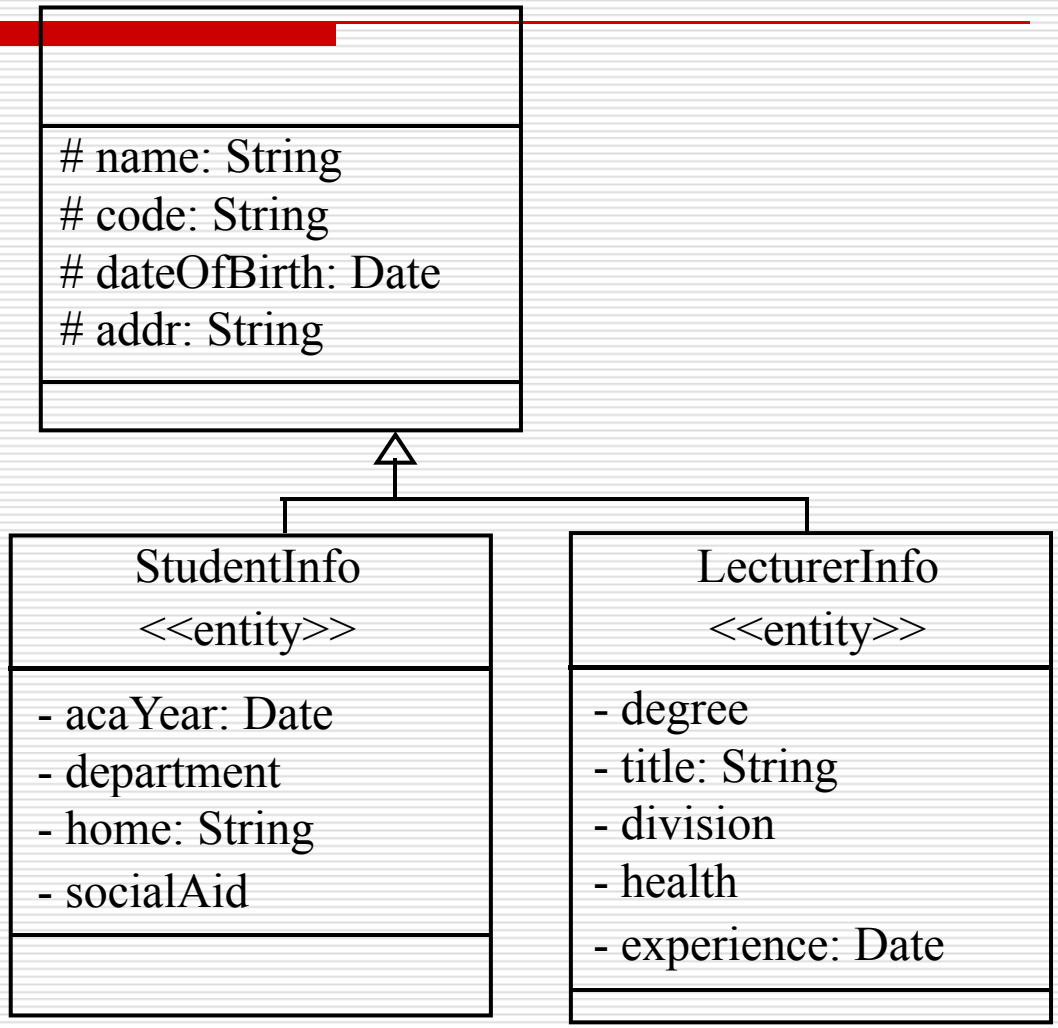


- UML định nghĩa quan hệ tổng quát hoá giữa một lớp tổng quá hơn với một lớp cụ thể hơn: lớp cụ thể hơn có tất cả thuộc tính, tác vụ và quan hệ của lớp kia + những thuộc tính/tác vụ riêng của nó
- Ký hiệu: mũi tên có đầu là một tam giác nhỏ
- Lớp tổng quát hơn nằm về phía mũi tên

Ví dụ về nhận diện lớp cơ sở

Ví dụ:

- Trong hệ thống đăng ký môn học tín chỉ qua WEB, lớp PeopleInfo là tổng quát hoá của StudentInfo và LecturerInfo



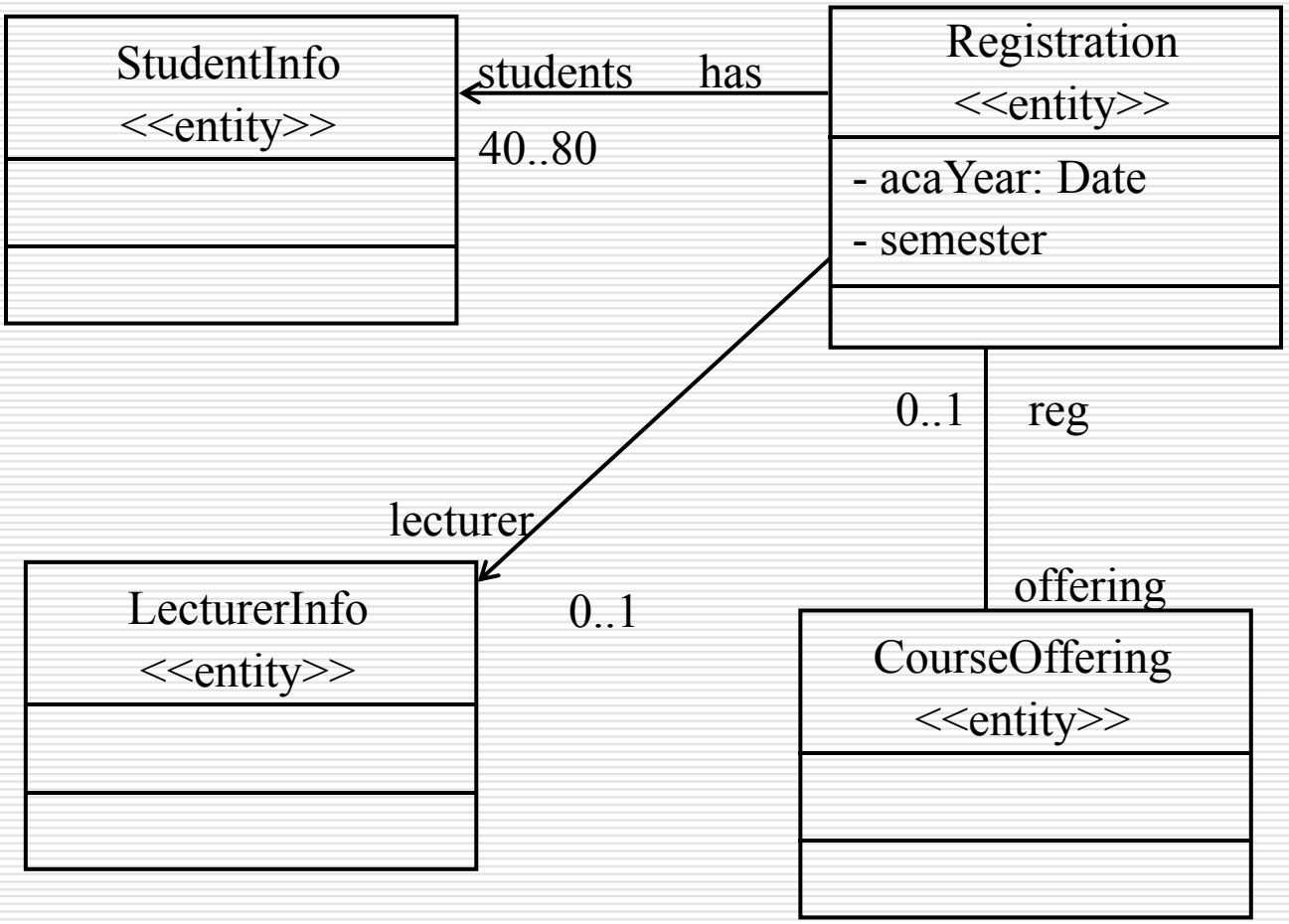
Nhận diện các mối quan hệ

- Sau khi xác định các lớp/đối tượng kể cả các đối tượng cơ sở, các quan hệ giữa các lớp cần được xác lập
- Trong mô hình phân tích các đối tượng/lớp có quan hệ với nhau
- Một số quan hệ mà UML hỗ trợ
  - Tổng quát hoá (generalization)
  - Liên kết (association)
  - Bao gộp (aggregation)
- Các quan hệ khác được áp dụng cho mô hình thiết kế
  - Phụ thuộc (dependency)
  - Cụ thể hoá (realization)

Quan hệ liên kết

- Quan hệ liên kết là mối quan hệ giữa 2 đối tượng/lớp
- Về ý nghĩa và ký hiệu giống như quan hệ liên kết trong mô hình nghiệp vụ
- Áp dụng cho 2 lớp có mối tương quan mang ý nghĩa nhất định
- Chú ý ghi rõ (nếu có thể được)
  - Bậc và tên vai trò của mỗi lớp trong quan hệ
  - Tên của chính quan hệ liên kết
- Dựa vào mô hình nhgiệp vụ xác định các mối quan hệ liên kết

Ví dụ - mối quan hệ liên kết



Ví dụ:

Lớp Registration liên kết với lớp StudentInfo LecturerInfo và CourseOffering

Quan hệ bao gộp

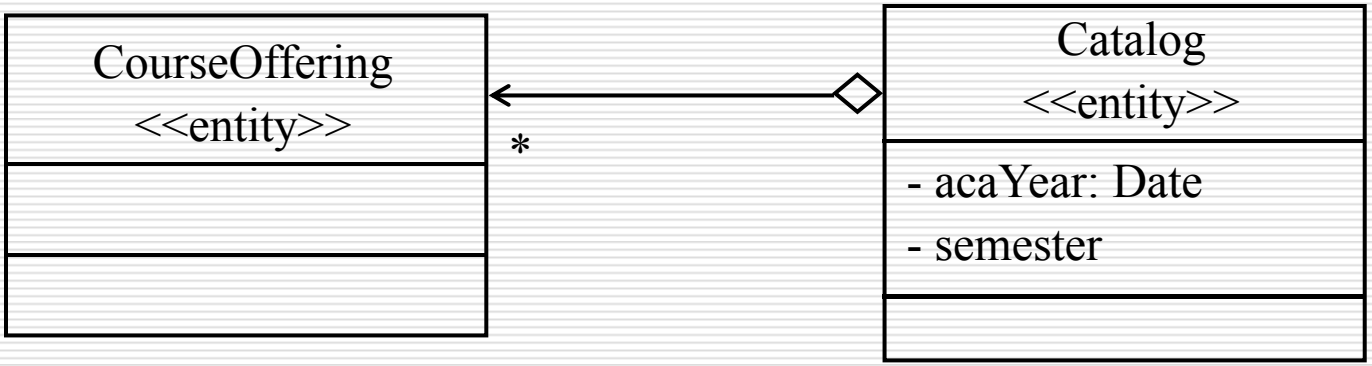
- UML định nghĩa quan hệ bao gộp là trường hợp đặc biệt của quan hệ liên kết, khi mà một đầu nối liên kết trở thành đầu nối bao gộp (aggregation)
- Lớp ở đầu nối bao gộp sẽ bao hàm lớp kia
- Ký hiệu của đầu nối bao gộp là một hình thoi tô hoặc không tô đen
- Có hai dạng bao gộp
  - Chia xẻ (shared): chia xẻ giữa các bao gộp khác nhau
  - Hoàn toàn (composite): sở hữu đầy đủ
- Xác lập các mối quan hệ bao gộm và biểu diễn chúng lên lược đồ lớp

Quan hệ bao gộp

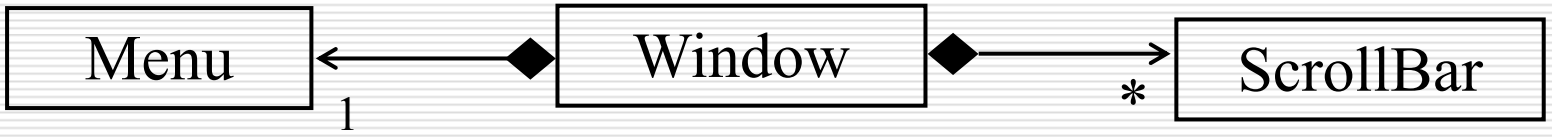
- Composite aggregation
  - Specifies that the existence of the part is dependent on the whole.
  - Is specified with a filled diamond.
- Shared aggregation
  - The part may be simultaneously in many composite instances
  - Shared aggregation is specified with a hollow diamond.

Quan hệ bao gộp – ví dụ

- Đối với hệ thống đăng ký môn học tín chỉ qua WEB, lớp Catalog bao gộp lớp CourseOffering

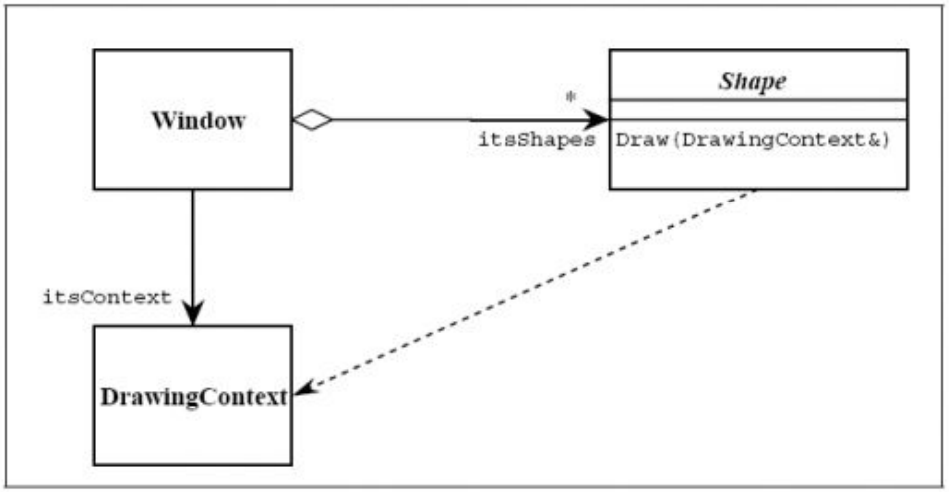


- Cửa sổ giao diện bao gộp hoàn toàn thanh cuộn và menu



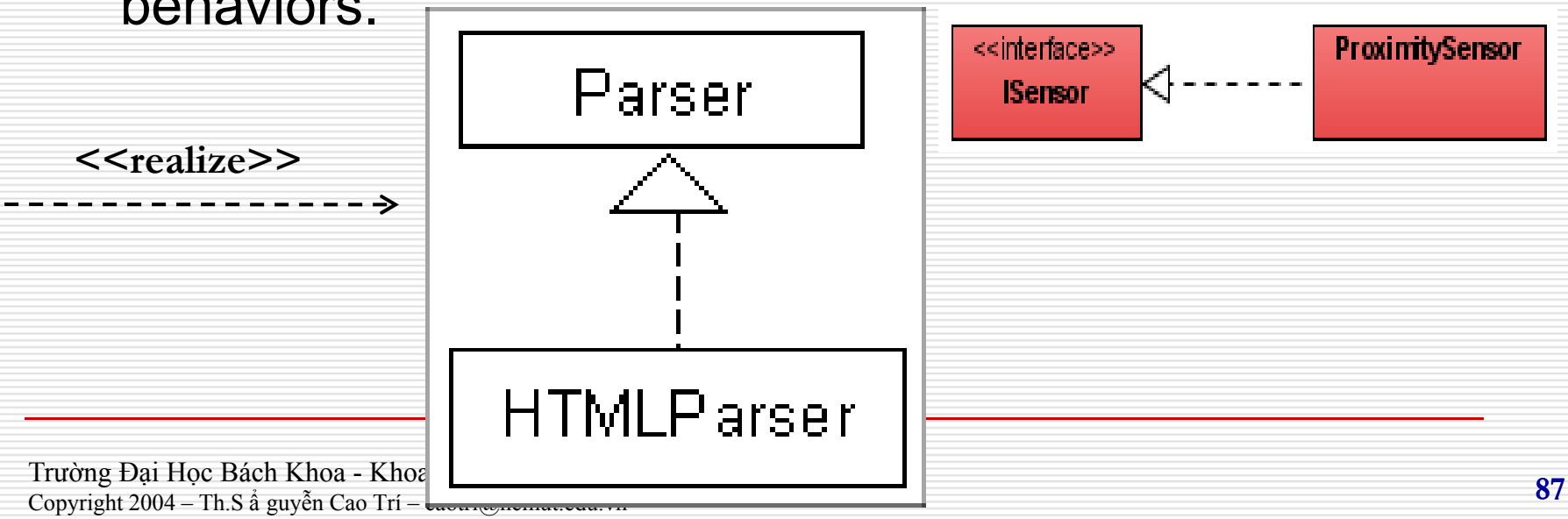
Quan hệ phụ thuộc

- A dependency relationship indicates that a change to one class may cause a change in the other class.



Quan hệ cụ thể hóa

- A realization relationship exists between two classes when one of them must realize, or implement, the behavior specified by the other.
- For example, a realization relationship connects an interface to a subsystem. The interface specifies the behaviors, and the subsystem implements the behaviors.

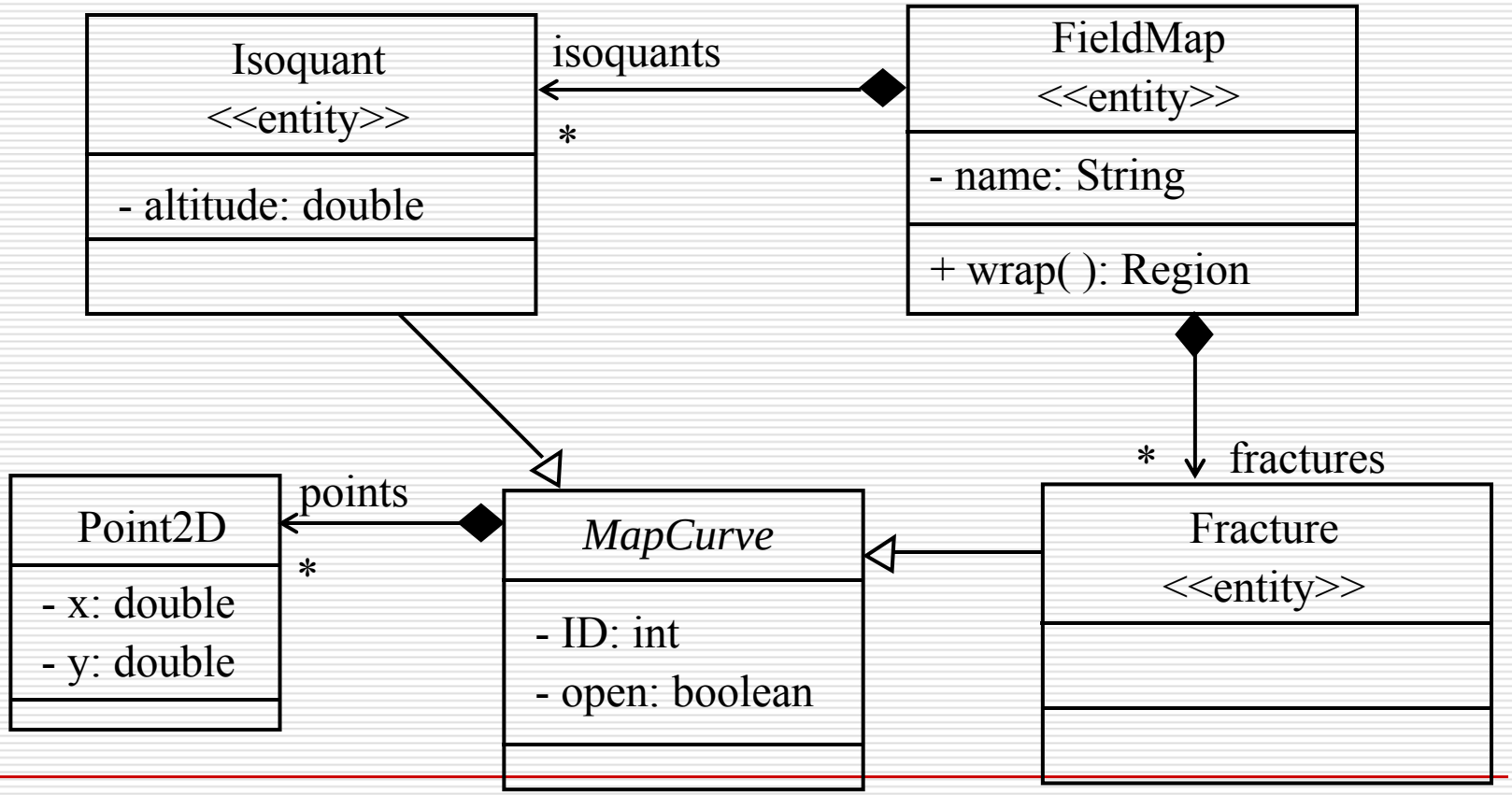


Xây dựng lược đồ lớp

- Lược đồ lớp (class diagram) biểu diễn cấu trúc của một số lớp và quan hệ giữa chúng → mô tả khía cạnh tĩnh (static) của hệ thống
- Hệ thống phức tạp có nhiều lớp → cần xây dựng nhiều lược đồ lớp, mỗi lược đồ mô tả một phần của hệ thống
- Lược đồ lớp được bổ sung và hoàn thiện trong mô hình thiết kế (thêm một số lớp, chi tiết các thuộc tính và tác vụ, làm rõ các quan hệ)
- Lược đồ lớp được xây dựng qua các bước
  - Xác định các lớp
  - Xác định thuộc tính và tác vụ của các lớp
  - Xác định các lớp cơ sở và quan hệ tổng quát hoá
  - Xác định các quan hệ liên kết và bao gộp

Ví dụ một lược đồ lớp phân tích

- một lược đồ lớp của chương trình hiển thị bề mặt địa hình

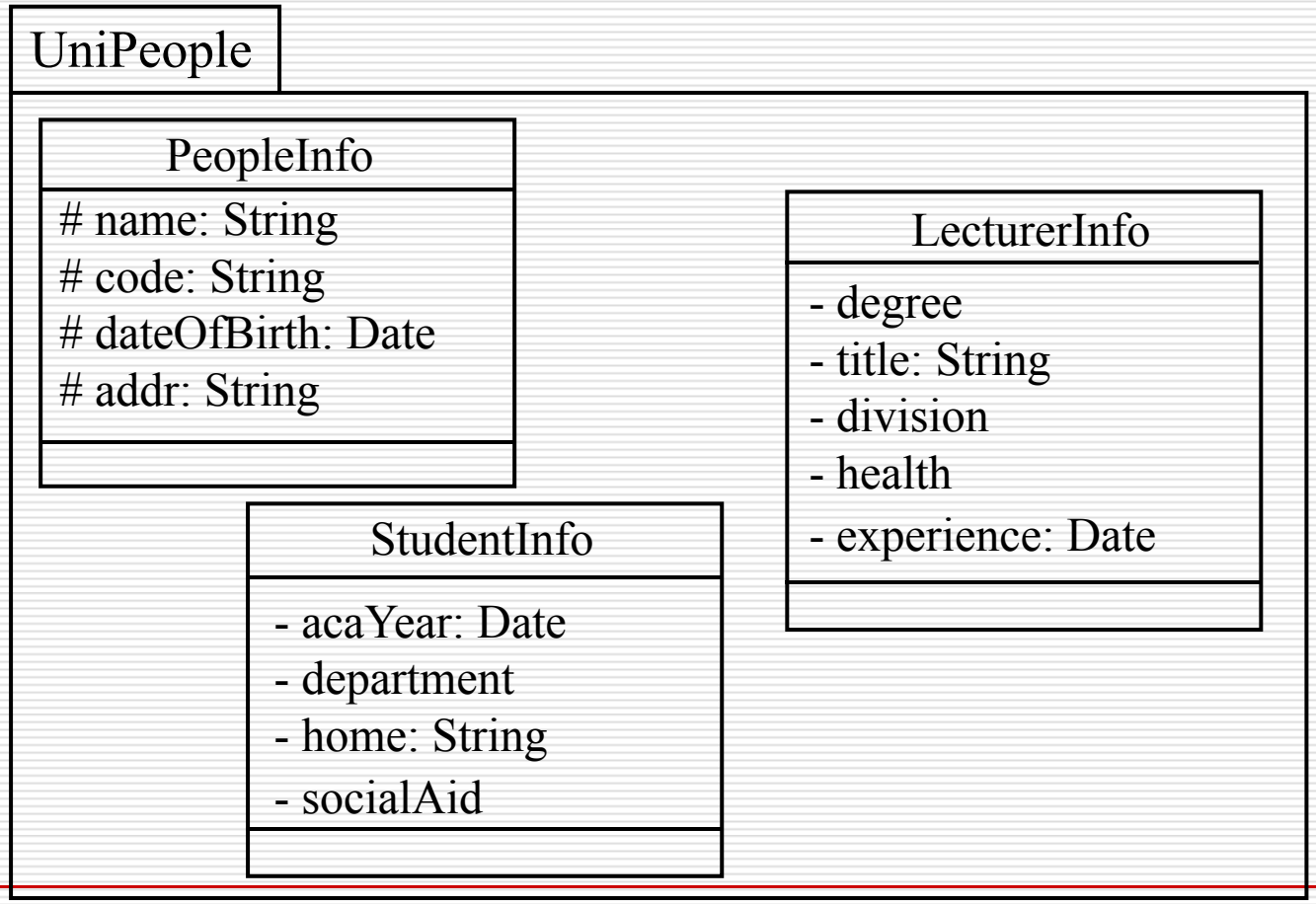


Thiết lập PACKAGE

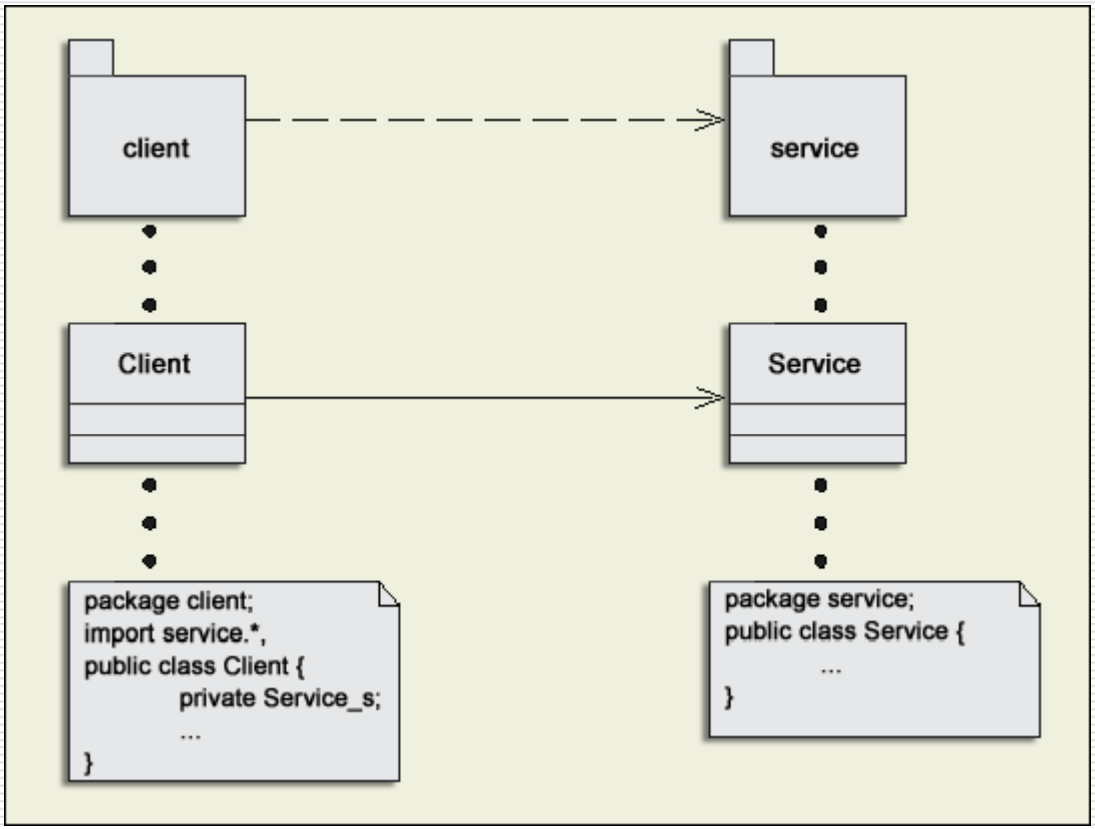
- Package là một cơ chế để tổ chức các phần tử vào các nhóm có liên hệ về ngữ nghĩa với nhau
- Package có thể import các phần tử từ một package khác
- Có thể chỉ ra quan hệ giữa các package
  - Phụ thuộc
  - Tổng quát hoá
- Mức độ truy xuất của package
  - Private: chỉ nó và các package import nó có thể truy xuất nội dung
  - Protected: giống như private nhưng cho phép thêm các package dẫn xuất
  - Public: các package khác có thể truy xuất nội dung
  - Implementation: không cho phép import, có thể áp dụng cho các phần tử bên trong package
- UML cho phép biểu diễn các PACKAGE và các mối quan hệ PACKAGE

Ví dụ về một PACKAGE

- package UniPeople chứa các lớp liên quan đến thông tin con người



Ví dụ về PACKAGE



Tổng kết

- Phân tích hệ thống cho OOP theo UML chia làm 2 bước:
  - Thu thập yêu cầu bằng mô hình nghiệp vụ
  - Phân tích và xác định tính năng hệ thống bằng mô hình phân tích
- Mô hình phân tích nhận diện các đối tượng/lớp: thực thể, biên, điều khiển
- Nhận diện các thuộc tính và một số tác vụ, tuy nhiên chưa làm rõ hành vi của chúng (→ mô hình thiết kế)
- UML hỗ trợ một số phần tử: lớp, đối tượng, lược đồ lớp, package

Tài liệu tham khảo

- Chapter 10, 11, 12, 20, 21: Software Engineering – A Practioner's Approach