System Analysis and Design

(IT3120E)

Quang Nhat Nguyen

quang.nguyennhat@hust.edu.vn

Hanoi University of Science and Technology

School of Information and Communication Technology

Academic year 2022-2023

Content:

- Introduction of object-oriented system analysis and design
- Introduction of the modeling language UML
- Introduction of software development process
- Analysis of the environment and needs
- **Function analysis**
- Structure analysis
- Interaction analysis
- Behavior analysis
- Design of the system's overall architecture
- Class detail design
- User interface design
- Data design

Function analysis

- Goal of the function analysis
- Modeling business processes by activity diagrams
- Modeling functional requirements by use case diagrams
- Illustrative example exercise

Goal of the function analysis

- Give a preliminary look at the system's functionality
  - *Business processes*
  - *Functional requirements*
- Function analysis is only intended to show the major functions (the system’s requirements)
  - Without going further into the minor functions
  - Because the currently and commonly used method of analysis and design (taught in this course) is **object-oriented** (not function-oriented)

Modeling business processes by activity diagrams (1)

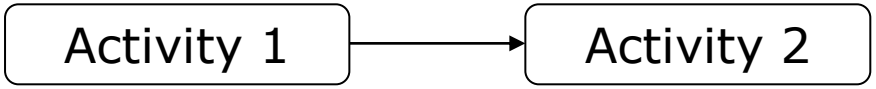
- Goal of modeling business processes
  - Usually, the project will be started with a **Preliminary Research** step to understand the **Business Environment** of the system to be built
  - In that environment, users, equipment, and computers work together according to certain **Business Processes**
  - Business Processes are often described using **Activity Diagrams**

Modeling business processes by activity diagrams (2)

- Goal of modeling business processes (2)
  - **Activity Diagram** is a diagram that describes activities following the flow from one activity to another
  - Activity Diagram may be used to describe:
    - A business process
    - Execution logic of a use case or a group of use cases
    - Execution logic of a usage scenario
    - Execution logic of a complex operation
  - Activity diagrams are UML models equivalent to block diagrams or data flow diagrams in the Function-oriented Analysis and Design method

Modeling business processes by activity diagrams (3)

- Activities and Transitions
  - An activity diagram is a directed graph, consisting of: **Nodes** are activities, **Edges** are transitions, **Start node** (bold black dot: ●), and **End node(s)** (bold black dot with border: ◉)
  - **Activity** is a job, can be handled by hand – example: form filling, or processed by computer – example: display information on the screen
    - Representation of an activity: Name of Activity
  - **Transition** is the transition from one activity to another
    - Representation of a transition:

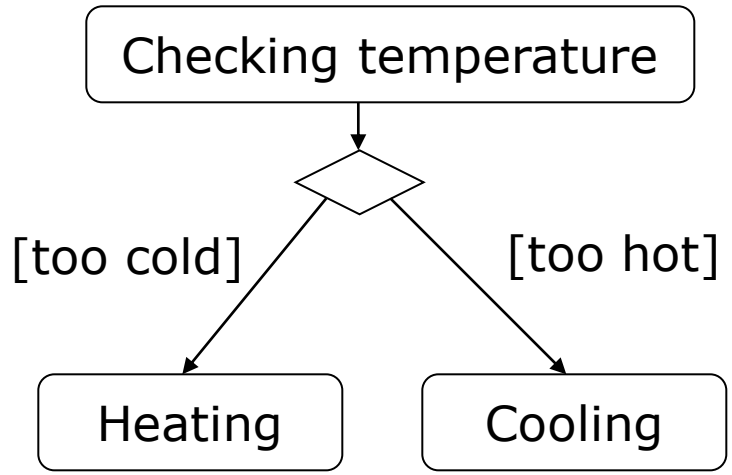


Modeling business processes by activity diagrams (4)

- Branching
  - **A guard** (<>) is a condition check, attached to a transition, to indicate that the transition is performed only when this condition is satisfied
  - To support branching, UML provides 2 kinds of guards:
    - A **decision** node: A single incoming flow and multiple outgoing flows (these outgoing flows must be **mutually-exclusive**);
    - A **merge** node: Multiple incoming flows and a single outgoing flow (this merge point is passed when an incoming flow occurs). **Merge node should not be used to synchronize concurrent flows!**

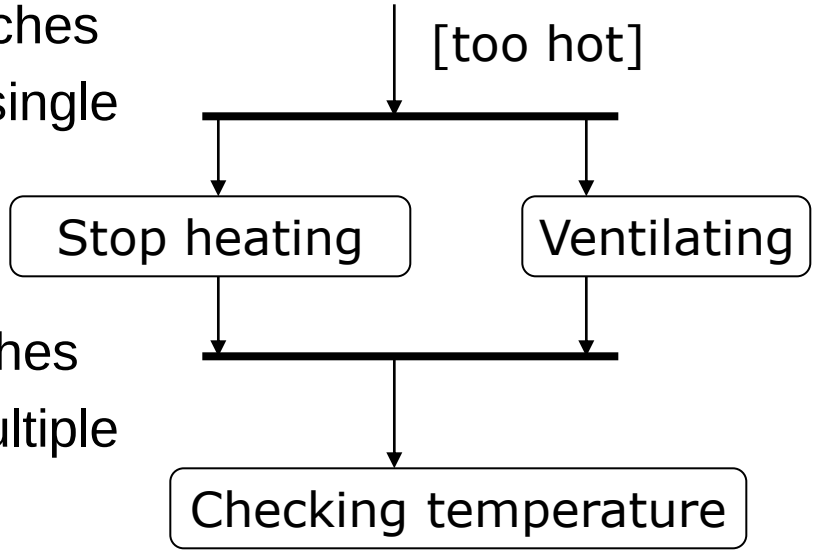
Modeling business processes by activity diagrams (5)

- Example of a decision node:



Modeling business processes by activity diagrams (6)

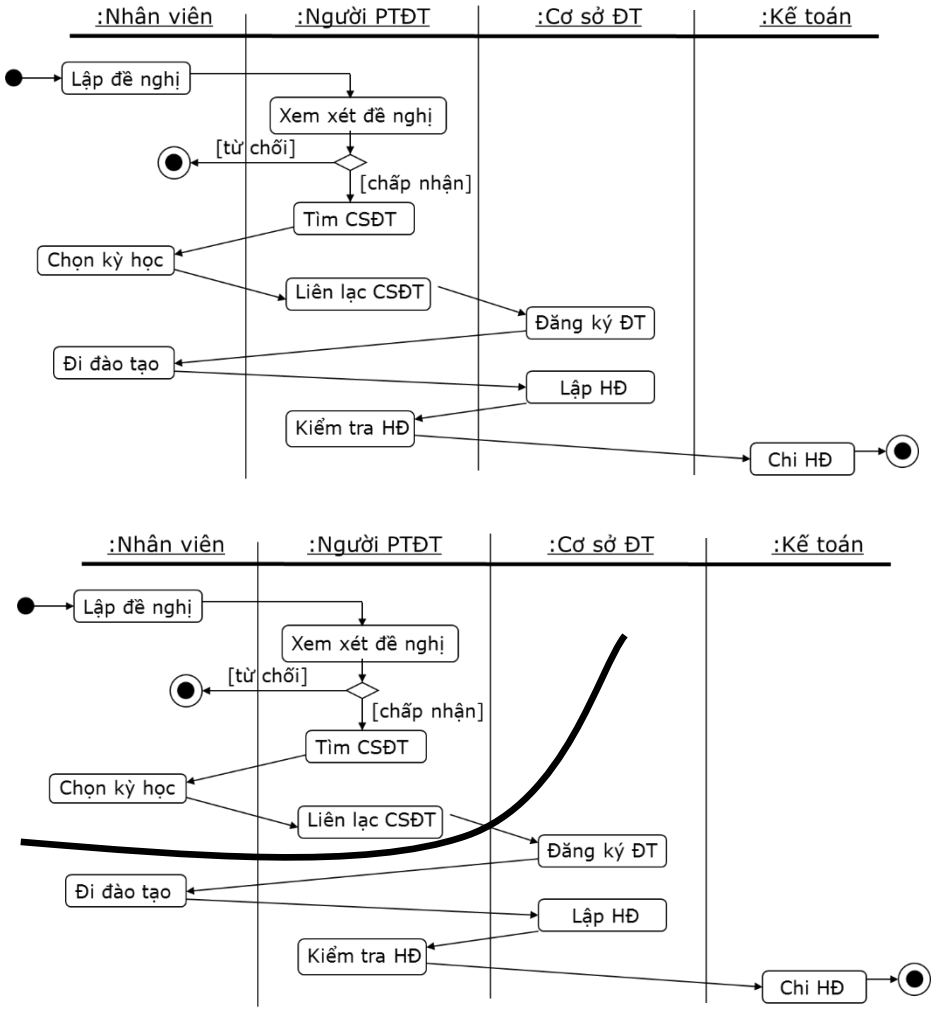
- Synchronization (of concurrent flows)
  - In an activity diagram, **synchronization bars** are used to open or close parallel execution branches:
    - **Opening** parallel execution branches by a synchronization bar to show a single incoming transition and multiple outgoing ones – called **a fork**
    - **Closing** parallel execution branches by a synchronization bar to show multiple incoming transitions and a single outgoing one – called **a join**



A join can be passed only when all the incoming execution branches complete

Modeling business processes by activity diagrams (7)

- Line-separating and partitioning
  - An activity diagram may be **line-separated**, like swim-lanes in a swimming pool. Each activity must be placed in a line, and each line shows the execution of one or more objects. Transitions may change lines freely.
  - An activity may be **partitioned**, each partition contains activities that have some characteristic in common. For example, a partition for the major usage scenario and several other partitions for minor ones.



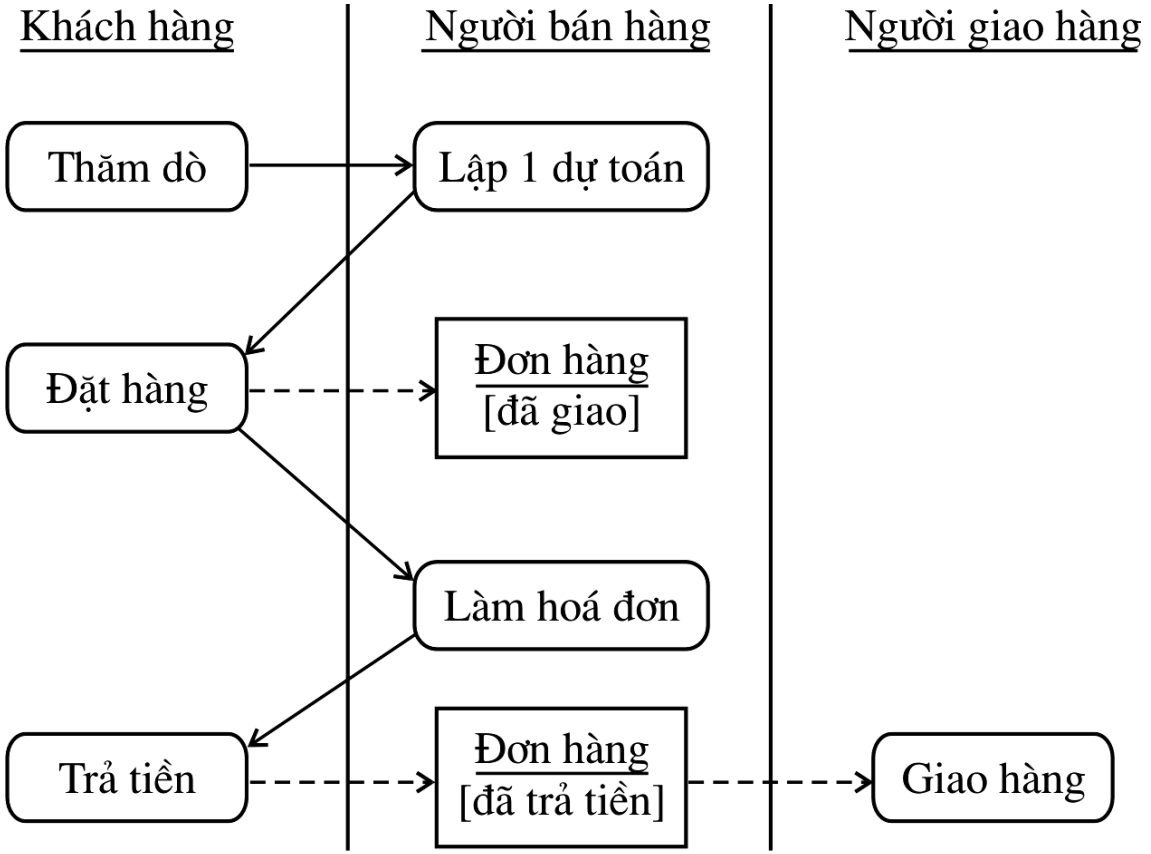
Modeling business processes by activity diagrams (8)

- Object creation and Object flow
  - In an activity diagram, we want to specify that some activity creates an object, or modifies the object's state, or takes as input an object for processing => We draw additional objects (with their states, if necessary) in the activity diagram, and connect those objects to the related activities with dashed arrows (called object flows). **An object flow** from one activity to an object and then continue to another activity can be seen as a control flow (a transition) between the two activities.
  - Example: The following figure shows a sales process, where an object (*"Đơn hàng"*) has been created and changed state through the activities.

Modeling business processes by activity diagrams (9)

- Object creation and Object flow (2)

A sales process:

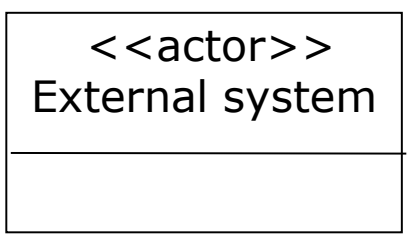


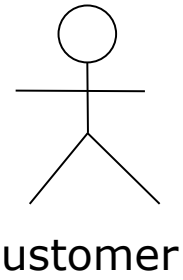
Modeling functional requirements by use case diagrams (1)

- Goal of modeling functional requirements
  - Define major system functions expected by users (called functional requirements)
  - Describe the analysis results with UML use case diagrams
- The order of work is as follows:
  - Modeling the work environment of the system by actors
  - Modeling the requirements by use cases
  - Determining relations (actors, use cases)
  - Creating use case diagrams

Modeling functional requirements by use case diagrams (2)

- Modeling the work environment of the system by actors
  - **Actors** is the role of one or more **users** or **external (i.e., outside the system to be built) objects** that interact with the system
  - Types of actors:
    - *Users* vs. *external systems/objects*
    - *Primary* vs. *secondary* (a.k.a. *supporting*)
- Representation of actors:





Modeling functional requirements by use case diagrams (3)

- Modeling the requirements by use cases
  - Definition of use case:
    - **A use case** is a representation of a set of system actions that provide a result to an actor
  - Characteristics of a use case:
    - A use cases must be associated with one or more actors
    - A use case must lead to a specific result
    - A use case must be a set of actions

Modeling functional requirements by use case diagrams (4)

- Modeling the requirements by use cases
  - Representation of a use case:

  - Specification of a use case:
    - In natural language
      - Name of the use case
      - Description of the usage purpose
      - (Primary, secondary) actors
      - Trigger (event)
      - Pre-conditions
      - Post-condition
      - Main (a.k.a. normal) flow
      - Alternative flows
    - Supplemented by an activity diagram or a sequence diagram

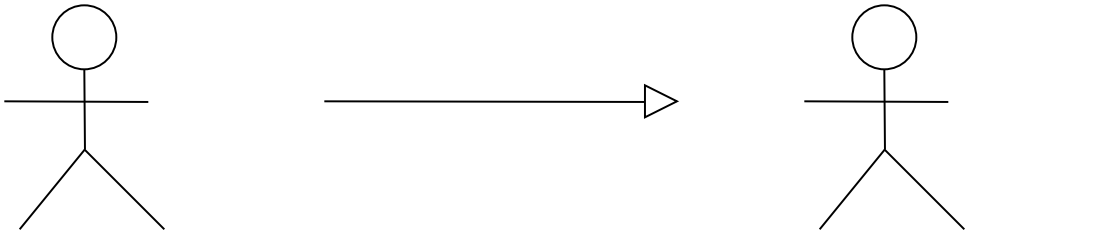
Modeling functional requirements by use case diagrams (5)

- Determining relations

There are different types of relations:

a) Generalization between actors

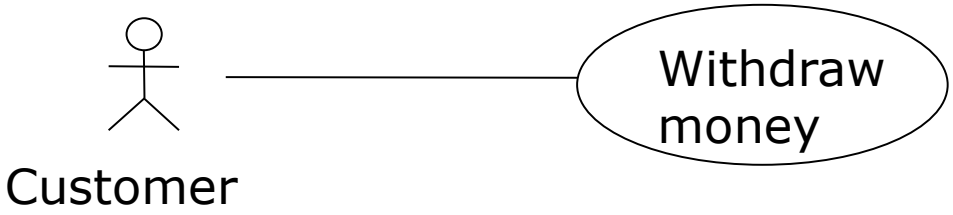
- Actor A is a **generalization** of actor B, if B inherits all the characteristics of A
- Representation:



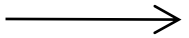
Modeling functional requirements by use case diagrams (6)

b) Communication between an actor and a use case

- If the actor and the use case communicate with each other
- Representation:



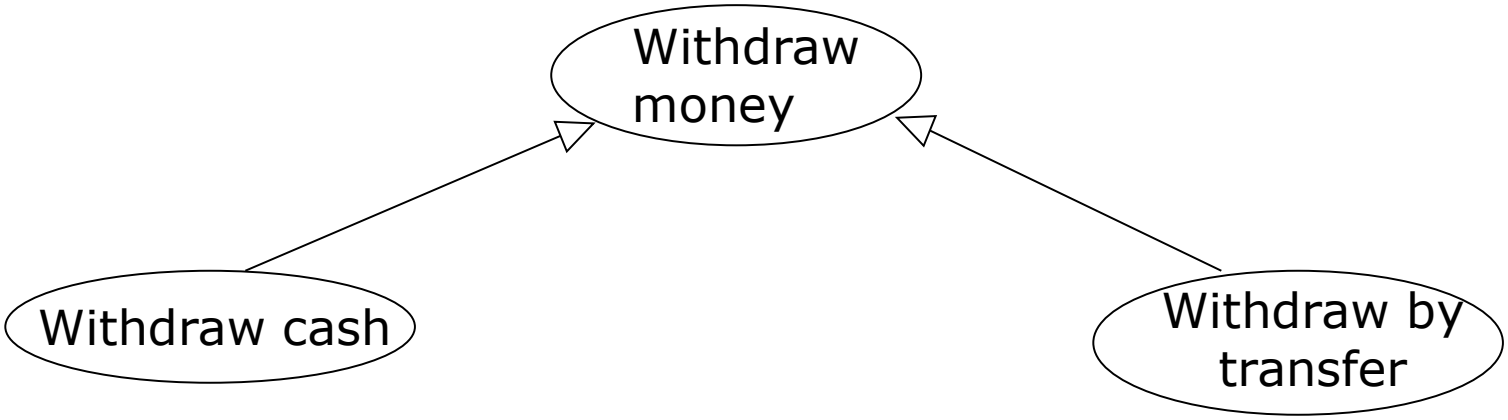
If the communication is one-way, use the arrow



Modeling functional requirements by use case diagrams (7)

c) Generalization between use cases

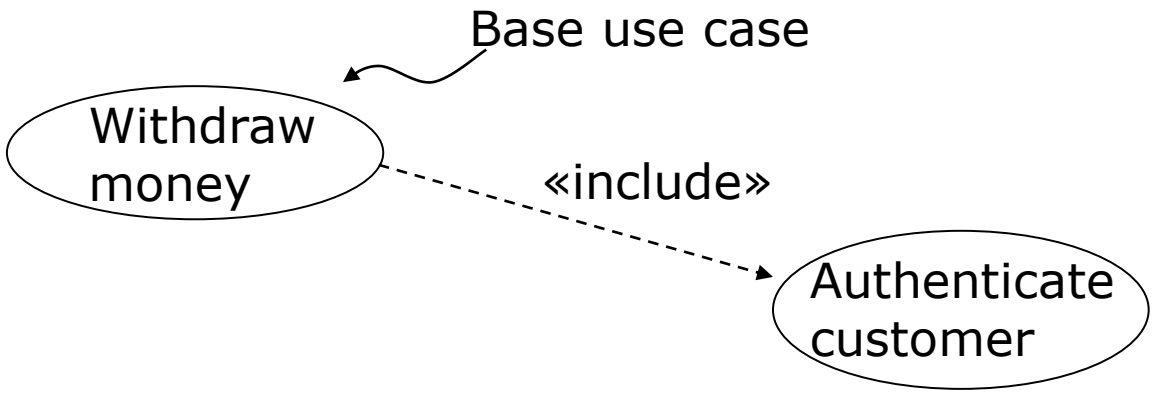
- Use case X is a **generalization** of use case Y, if Y inherits all the characteristics of X (can modify and add)
- Representation:



Modeling functional requirements by use case diagrams (8)

d) Inclusion between use cases

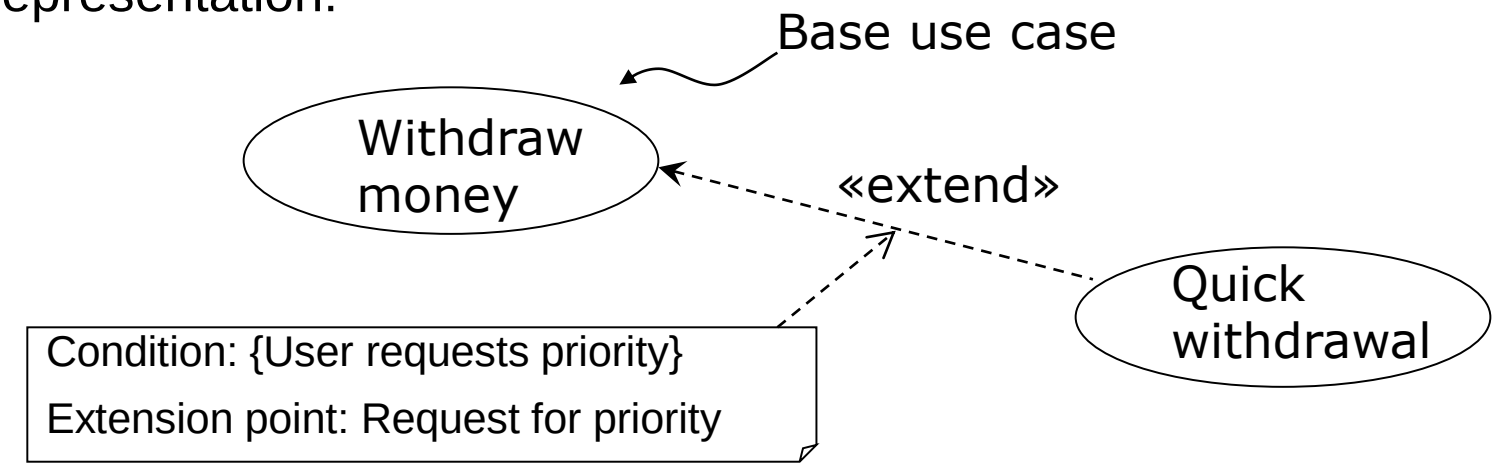
- If use case X **always include** the content of use case Y into the content of X at a position (i.e., inclusion point) specified in the specification of X
- X (the use case at the beginning of the arrow) is called the *base use case*
- Representation:



Modeling functional requirements by use case diagrams (9)

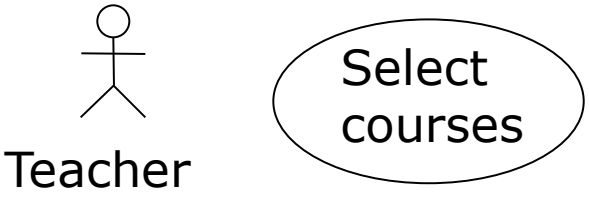
e) Extension between use cases

- If use case X **conditionally include** the content of use case Y into the content of X at a position (i.e., extension point) specified in the specification of X
- X (the use case at the end of the arrow) is called the *base use case*
- Representation:



Modeling functional requirements by use case diagrams (10)

- Creating use case diagrams
  - Nodes are actors and use cases:



  - Edges can be either of 4 types:
    - **Generalization** (between actors or use cases):
    - **Communication between** actor and use case:
    - **Inclusion** between use cases: <<include>>
    - **Extension** between use cases: <<extend>>

Illustrative example exercise (1)

(A) Modeling business processes

Đầu bài: Để nâng cấp hệ thống thông tin của mình, một công ty muốn, trong bước đầu, vạch ra một quy trình đào tạo (QTĐT) cho nhân viên của mình, để sau đó tin học hoá một phần quy trình này

1) QTĐT bắt đầu khi có một Nhân viên (NV) gửi đến Người phụ trách đào tạo (PTĐT) một đề nghị được đi đào tạo. Người PTĐT xem xét đề nghị này và đưa ra trả lời đồng ý hay không đồng ý.

2) Nếu đồng ý, Người PTĐT tìm trong danh mục cơ sở đào tạo (CSĐT) một nơi có các lớp đào tạo thích hợp, thông báo nội dung đào tạo lại cho NV đã xin đi đào tạo, cùng với một danh sách các kỳ học sẽ mở tới đây. Khi NV đã chọn kỳ học, Người PTĐT gửi một đăng ký cho NV đó tới cơ sở đào tạo.

Illustrative example exercise (2)

(A) Modeling business processes

Đầu bài *(…tiếp theo)*

3) Nếu sau khi đăng ký hoặc sau khi gửi đề nghị đào tạo, mà NV không thể tham dự được, NV phải báo sớm cho Người PTĐT để huỷ đăng ký hay huỷ đề nghị đào tạo.

4) Sau khi đào tạo xong, NV phải nộp lại cho Người PTĐT một giấy xác nhận sự có mặt và giấy nhận xét kết quả học tập.

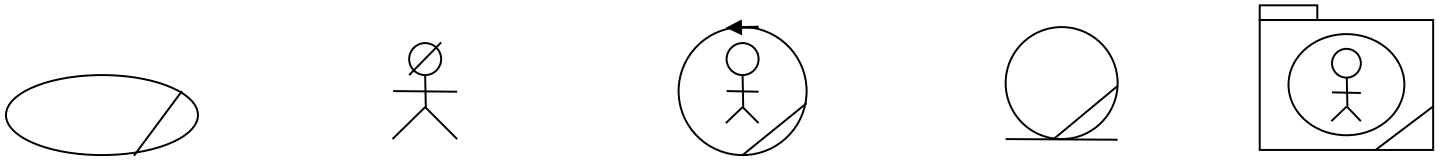
5) Người PTĐT kiểm tra lại hoá đơn mà CSĐT gửi tới, trước khi chuyển cho kế toán trả tiền.

Illustrative example exercise (3)

(A) Modeling business processes

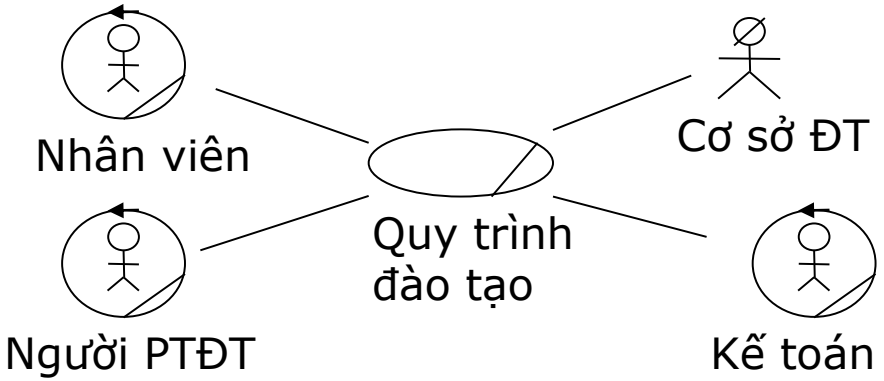
Bước 1: *MHH quy trình trên bằng một biểu đồ ca sử dụng (mức đỉnh)*

- Để MHH nghiệp vụ (NgV), ta dùng các biểu tượng sau, do Jacobson đề xuất (được dùng trong RUP và trong Rational Rose 2000):



Quy trình NgV Tác nhân NgV Nhân viên NgV Thực thể NgV Gói NgV

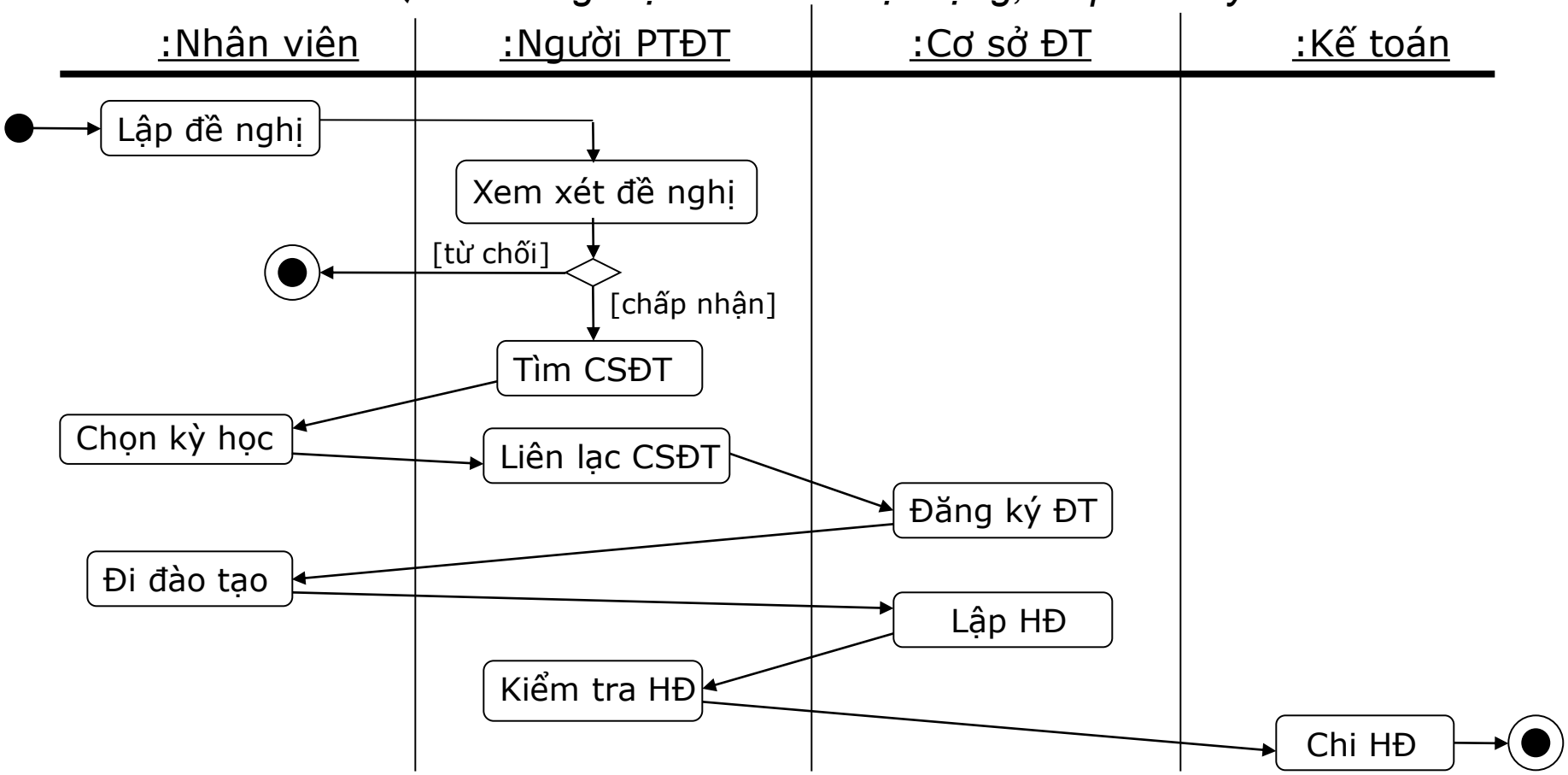
- Biểu đồ ca sử dụng NgV (mức đỉnh):



Illustrative example exercise (4)

(A) Modeling business processes

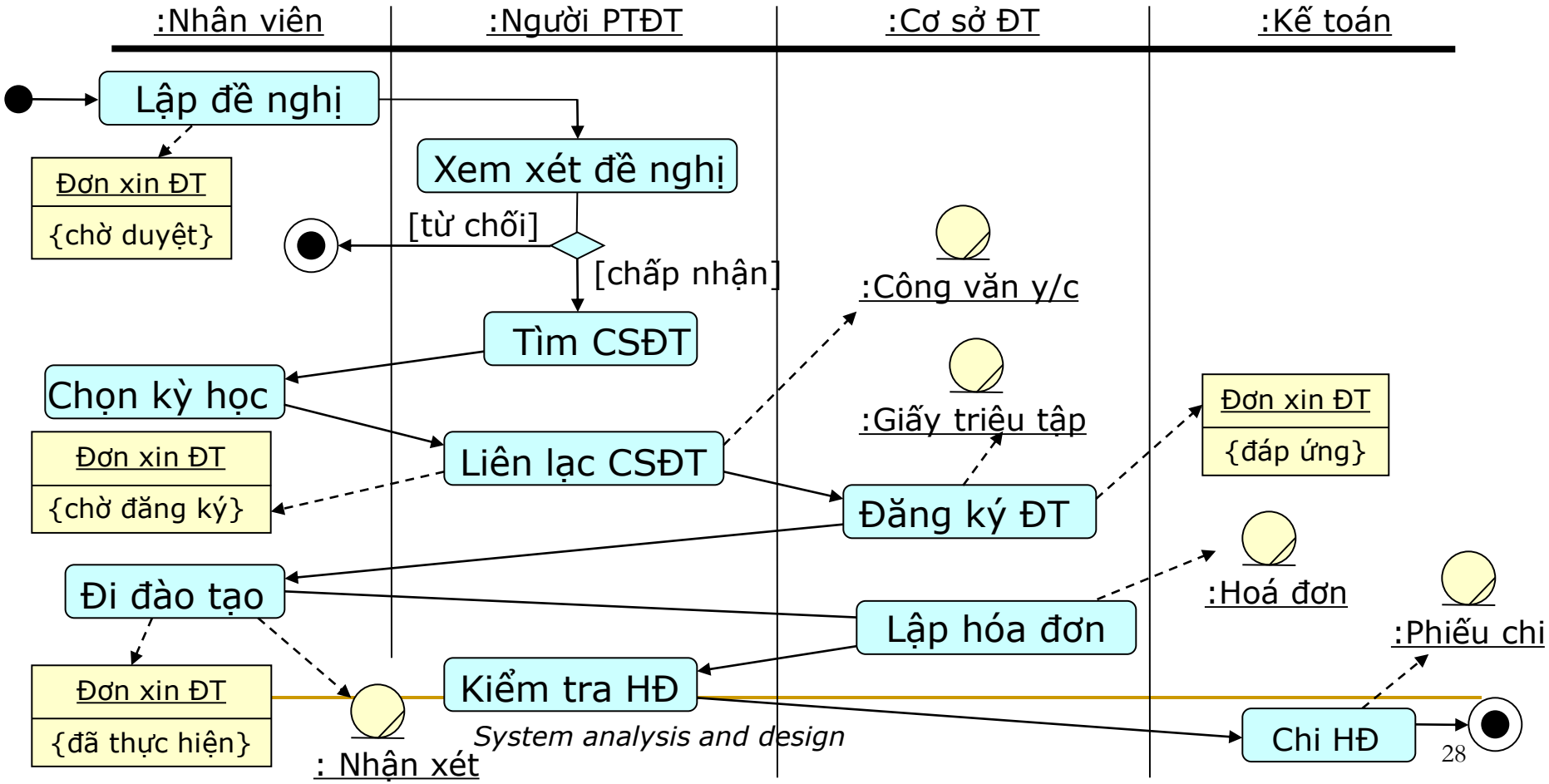
Bước 2: *Mô tả QTĐT bằng một biểu đồ hoạt động, có phân tuyến*



Illustrative example exercise (5)

(A) Modeling business processes

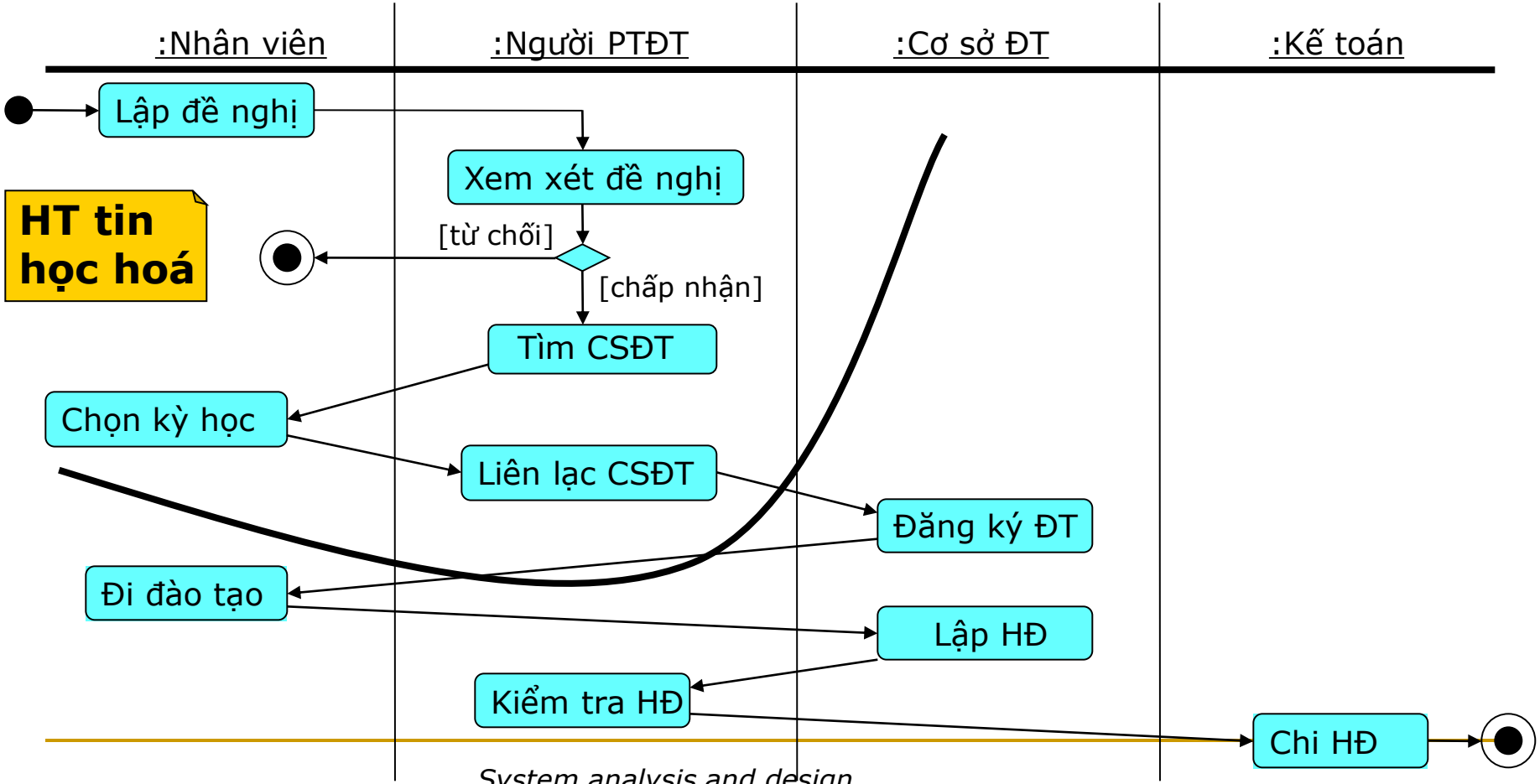
Bước 2 (tiếp): *Thêm vào biểu đồ hoạt động các đối tượng NgV đã được sản sinh cùng với trạng thái của chúng nếu có*



Illustrative example exercise (6)

(B) Modeling functional requirements

Bước 3: *Từ mô hình nghiệp vụ, xác định phạm vi của Hệ thống (HT) tin học hóa*



Illustrative example exercise (7)

(B) Modeling functional requirements

Bước 4: *Xác định các yêu cầu của Hệ thống tin học hoá (Hệ thống YCĐT)*

- Hệ thống YCĐT cho phép bắt đầu một yêu cầu đào tạo và theo dõi nó cho đến khi hoàn tất việc đăng ký đào tào cho một nhân viên
- Hệ thống YCĐT phải tin học hoá các hoạt động nghiệp vụ sau:
  - Soạn thảo một đề nghị đào tạo (Nhân viên);
  - Xem xét một đề nghị (Người PTĐT);
  - Tìm cơ sở đào tạo (Người PTĐT);
  - Chọn lớp và kỳ học (Nhân viên);
  - Đăng ký đào tạo (Người PTĐT).
- Muốn đáp ứng các yêu cầu trên, Hệ thống phải quản lý một danh mục CSĐT mà Nhân viên có thể đọc và Người PTĐT có thể viết vào đó và tổ chức nó theo chủ đề
- *Lưu ý*: Nhân viên có thể huỷ Đăng ký đào tạo, hoặc huỷ Đề nghị đào tạo

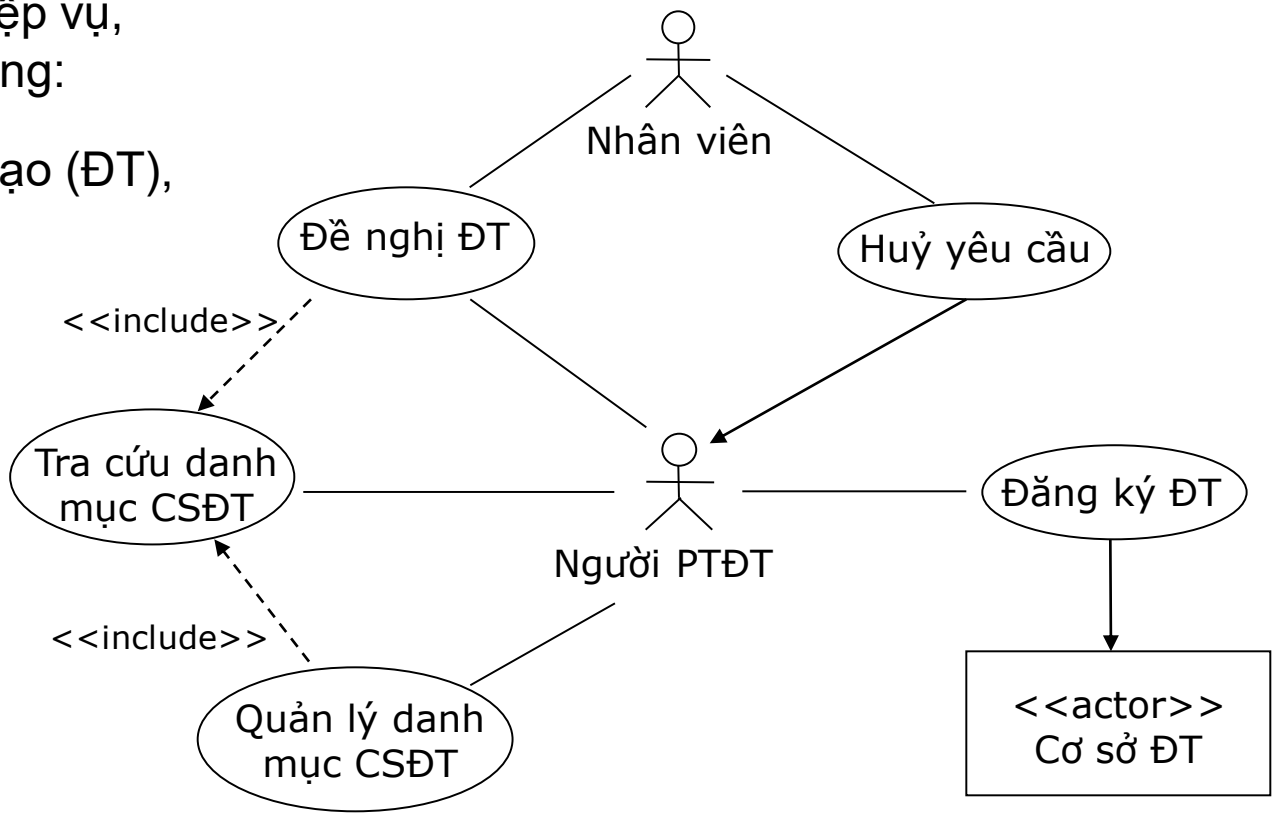
Illustrative example exercise (8)

(B) Modeling functional requirements

Bước 5: *Lập biểu đồ ca sử dụng cho Hệ thống YCĐT*

Từ các hoạt động nghiệp vụ, phát hiện các ca sử dụng:

- Đề nghị được đào tạo (ĐT),
- Đăng ký ĐT,
- Huỷ yêu cầu ĐT,
- Tra cứu danh mục CSĐT
- Quản lý danh mục CSĐT



Illustrative example exercise (9)

(B) Modeling functional requirements

Bước 6: *Mô tả chi tiết (đặc tả) ca sử dụng 'Quản lý danh mục CSĐT'*

I. Mô tả chung

- **Tên:** Quản lý danh mục CSĐT **Loại:** Chi tiết
- **Nội dung tóm tắt:** Người PTĐT có trách nhiệm cập nhật thường xuyên danh mục CSĐT, trong đó chứa các thông tin cần thiết về các CSĐT. Phần lớn các chỉnh sửa đều bắt nguồn từ các CSĐT.
- **Tác nhân:** Người PTĐT.
- **Ngày lập:** 20/11/2015 **Ngày cập nhật:** 01/02/2016
- **Phiên bản:** 3.0 **Người lập:** Lê Anh (leanh@company.com)

Illustrative example exercise (10)

(B) Modeling functional requirements

Bước 6 (tiếp): *Mô tả chi tiết (đặc tả) ca sử dụng 'Quản lý danh mục CSĐT'*

II. Mô tả các kịch bản

- Điều kiện tiên quyết (Pre-conditions): Người PTĐT đã được giao trách nhiệm
- Kịch bản chính (Normal flow):

1. Ca sử dụng bắt đầu khi một CSĐT báo cho Người PTĐT các thay đổi của họ	
2. Người PTĐT có thể bổ sung một CSĐT mới, loại bỏ một CSĐT cũ, hay cập nhật thông tin của một CSĐT đang tồn tại. Khi bổ sung hay cập nhật, Người PTĐT có thể điều chỉnh lịch học của các lớp đào tạo	3. Hệ thống thông báo cho các người dùng đang kết nối hãy coi chừng dùng phải một phiên bản cũ của danh mục CSĐT Khi loại bỏ một CSĐT cũ, hệ thống sẽ cho: Người PTĐT biết các NV đã đăng ký vào các lớp bị loại bỏ, và các NV đó biết các đăng ký đã bị huỷ bỏ
4. Khi cần, Người PTĐT có thể bổ sung thêm một loại hình đào tạo mới	
5. Người PTĐT xác nhận các điều chỉnh đã thực hiện	6. Hệ thống thông báo cho các người dùng đang kết nối là đã có một phiên bản mới của danh mục CSĐT

Illustrative example exercise (11)

(B) Modeling functional requirements

- Các kịch bản ngoại lệ (Alternative flows) có thể:
  - *A1: Thông tin không đầy đủ.*

    Kịch bản A1 khởi động ở điểm 2 của kịch bản chính.
  - *A2: Khi các thông tin liên quan tới một CSĐT mới là không đầy đủ (vd: thiếu ngày cho các lớp đào tạo), thì CSĐT đó được đưa vào danh mục CSĐT nhưng chưa cho đăng ký, và CSĐT đó phải được cập nhật sau.*

    Kịch bản chính tiếp tục từ điểm 2
- Các ràng buộc (constraints):
  - Xung đột: Ca sử dụng mỗi lần chạy chỉ làm việc với một Người PTĐT
  - Sẵn dùng: Danh mục CSĐT có thể truy cập được trong khoảng thời gian từ 09:00 thứ hai đến 17:00 thứ sáu. Các cập nhật cần hạn chế tối đa trong khoảng thời gian đó.