Software Engineering

Use Case Diagrams

Dr. Sayed AbdelGaber
Professor
Faculty of Computers and Information
Helwan University

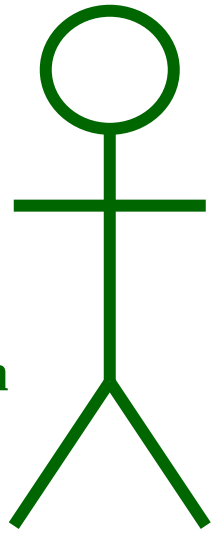
COMPARISON WITH DATA FLOW DIAGRAMS

- **Resemble:**
  - ✓ Model processes (requirements).
  - ✓ A temporal.

- **Different:**
  - ✓ Do not show flow of information.
  - ✓ Do not follow a strict hierarchical structure.
  - ✓ Do not model (explicitly) data repositories and sources.

ACTORS

- **Denoted by:**
  - ✓ stick figure
  - ✓ role name underneath



role-name

- **Represent:**
  - ✓ roles that people have when interacting with the system.
  - ✓ external systems or hardware that are essential to system operation.
  - ✓ other information systems or databases.

- **Examples:**
  - ✓ Employee, account holder, scheduling system, sales database

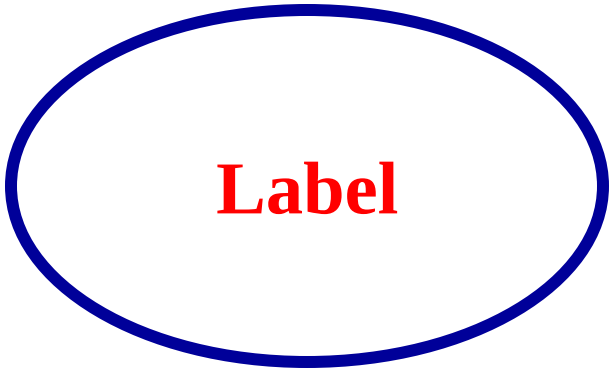
USE CASES

- **Denoted by:**
  - ✓ an ellipse
  - ✓ label *inside* or underneath

- **Represent:**
  - ✓ a discrete unit of system functionality.
  - ✓ '...a sequence of actions that a system performs to achieve an observable result of value to an actor'.

[Source: Bennet et al., 2001]

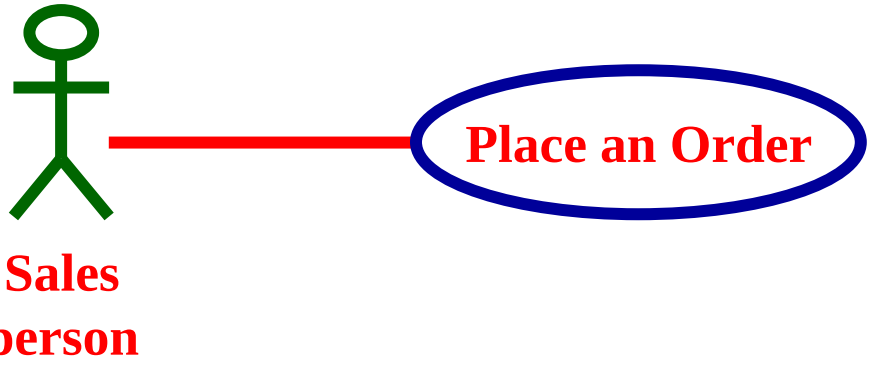
  - ✓ Activity from perspective of an actor.



- **Examples:**
  - ✓ Place an order, Rent a video, Check bank balance, ...

ASSOCIATIONS

- **Denoted by:**
  - ✓ Solid line between actor and (associated) use case.
- **Represents:**
  - ✓ Relationship between actors and use cases (communication).
  - ✓ Actor can be associated with more than one use case.
  - ✓ Use case can be associated with more than one actor.
- **Other types:**
  - ✓ Generalization
  - ✓ Stereotyped relationships



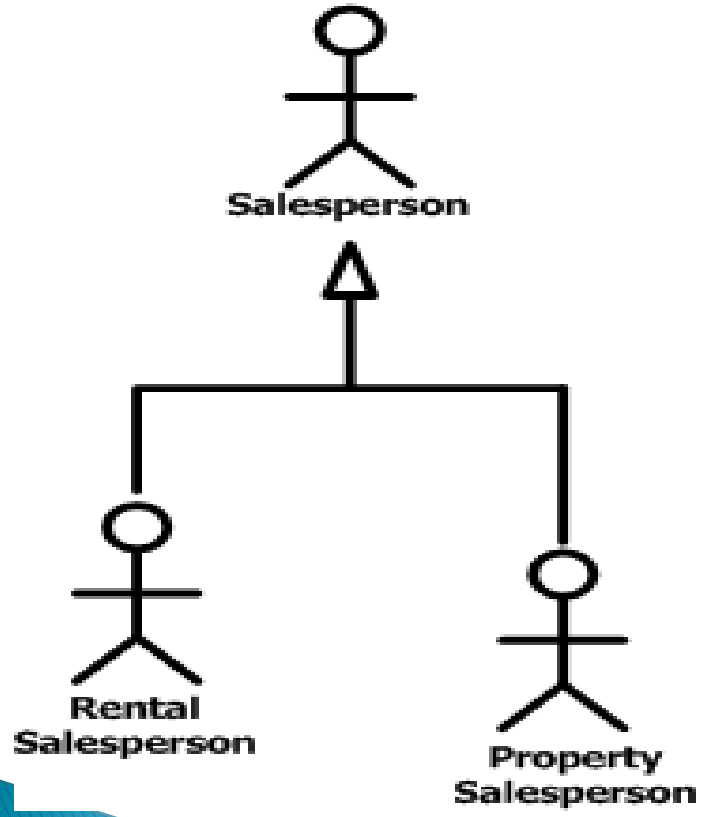
- **Can be annotated to show:**
  - ✓ direction (navigability)
  - ✓ multiplicity (cardinality/participation)

GENERALIZATION

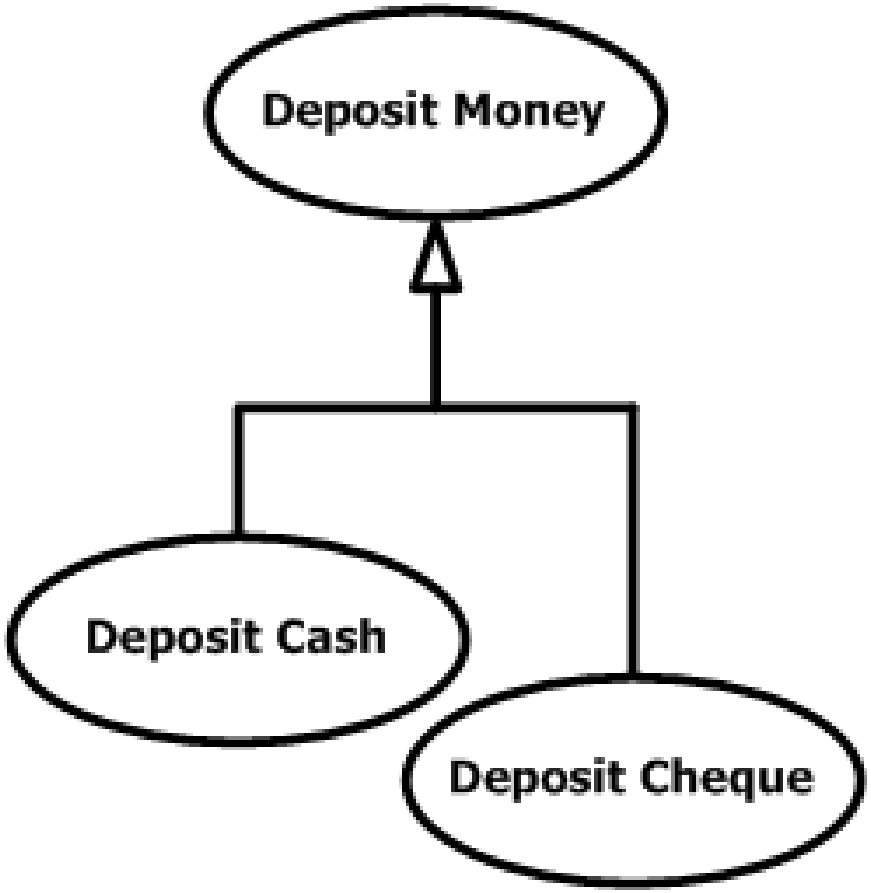
- **Actors and use cases can be organized into generalization hierarchies:**
  - ✓ **actors can be specializations of a (super) actor.**
  - ✓ **use cases can be specializations of a (super) use case.**
- **Simplifies diagram by minimizing associations.**
- **Use cases can have:**
  - ✓ **common actions**
  - ✓ **unique actions**
- **Similar concept to inheritance.**
- **Generalized use case can be an *abstract* use case.**

GENERALIZATION EXAMPLES

- Two specializations of sales person:

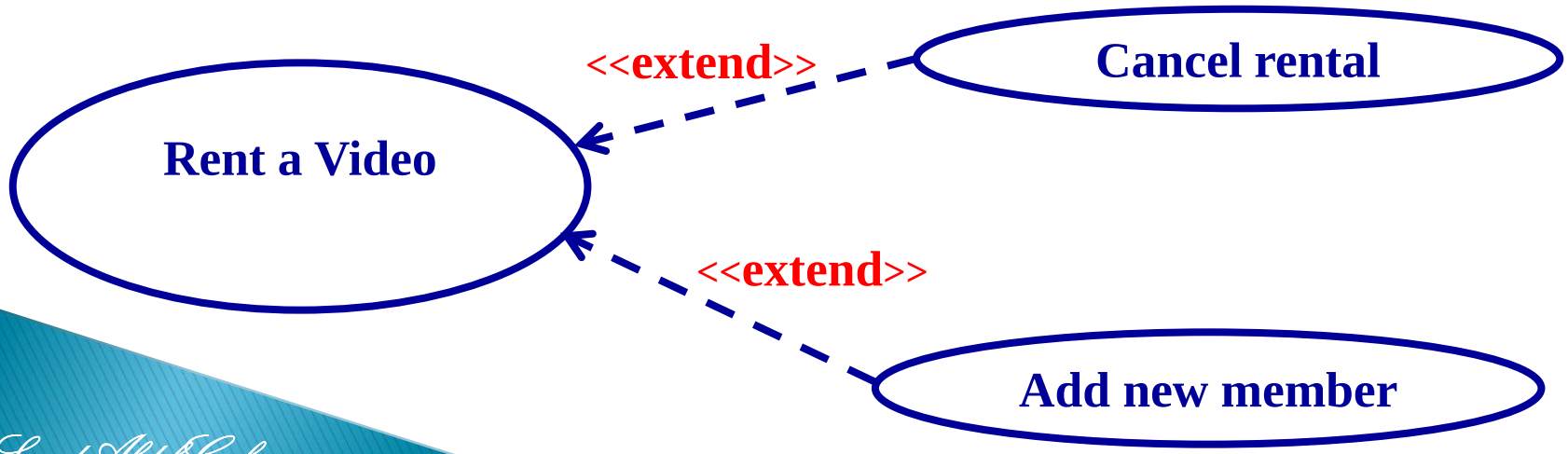


- Two specializations of deposit money:



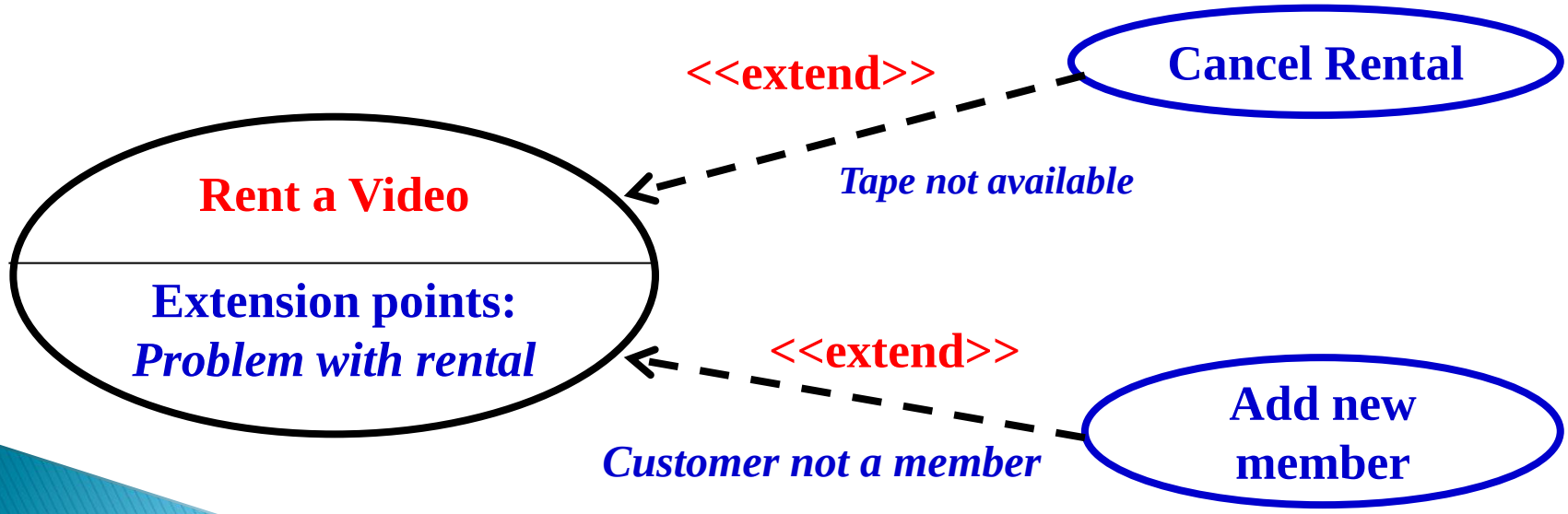
THE <<EXTEND>> STEREOTYPE

- Adds new or alternative functionality to an existing use case when required.
- Activity is optional unless conditions met.
- Denoted by:
  - ✓ a dashed arrow that points towards the base use case.
  - ✓ <<extend>> label above arrow.



EXTENSION POINTS

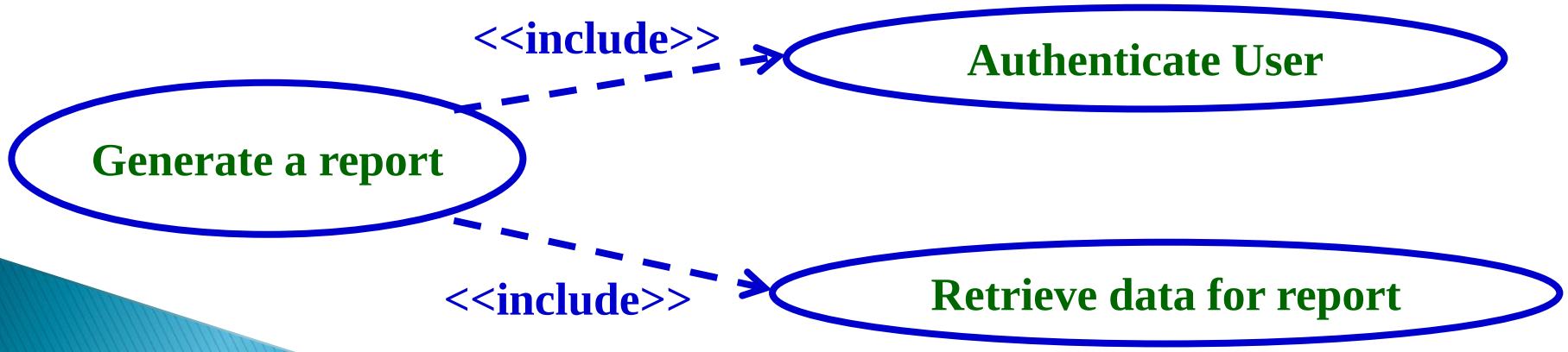
- Used to clarify the circumstances and conditions that trigger an extended use case.
- Extensions defined in base use case directly under the use case name.
- Conditions are placed next to the relevant <<extend>> association.



THE <<INCLUDE>> STEREOTYPE

- Represents an activity that must be performed as part of another use case.

- Denoted by:
  - ✓ a dashed arrow that points away from the base use case towards mandatory activity.
  - ✓ <<include>> label above arrow.



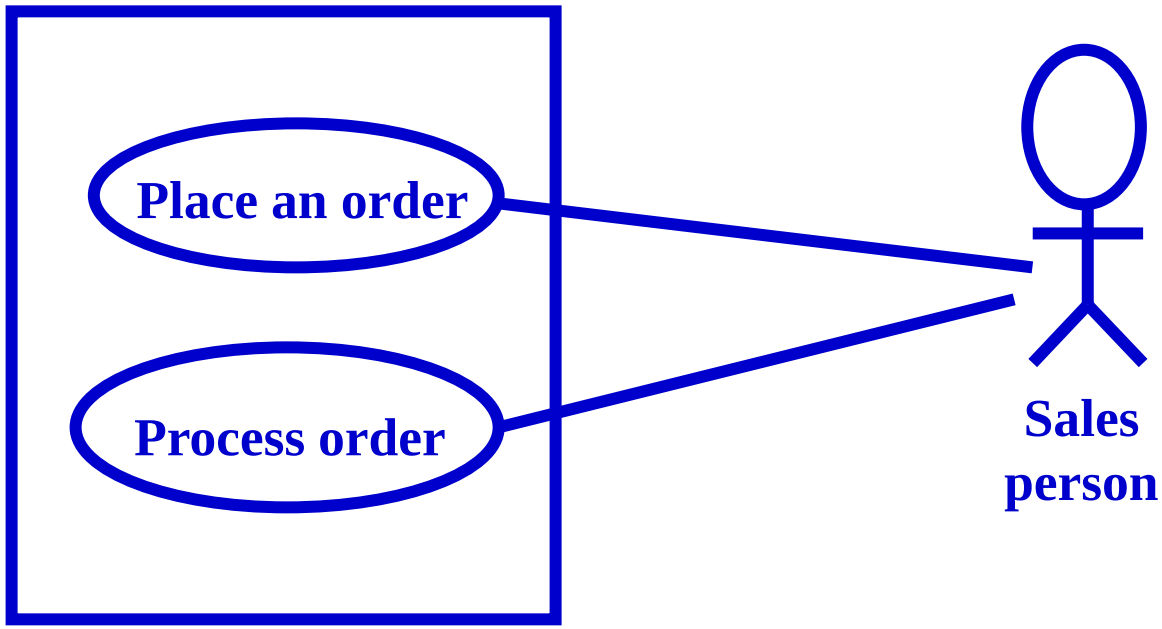
Documenting Use Cases

Each use case is a list of scenarios, and each scenario is a sequence of steps. Each scenario of each use case will also have its own page, listing in text from the:

- **Actor who initiates the use case**
- **Preconditions for the use case**
- **Steps in the scenario**
- **Post-conditions when the scenario is complete**
- **Actor who benefits from the use case**

SYSTEM BOUNDARY

- Represents boundary between the system and actors that interact with the system.
- Can organize related activities to simplify diagrams.
- Optional.
- Denoted by a rectangle box enclosing use cases.



APPROACHES

- **Top-down:**
  - ✓ Find actors → find use cases → detail use cases
- **Bottom-up:**
  - ✓ create scenario → generalize scenario → organize use case model
- **Organize collection of use case diagrams into a use case model:**
  - ✓ Almost parent/child structure
  - ✓ Wrap individual diagrams in *packages*

TOP-DOWN APPROACH

1. **Find actors and use cases.**
   - ✓ **Who enters information and/or receives information?**
   - ✓ **What other systems will interact with system?**
2. **Priorities use cases.**
3. **Develop each use case (starting with priority ones).**
4. **Structure the use case model.**

[Source: Bennet et al., 2001]

EXAMPLE A : HR ONLINE SYSTEM DESCRIPTION

- **Actors :**
  - ✓ employee, employee account db, healthcare plan system, insurance plan system

secondary actor

- **Preconditions:**
  - ✓ Employee has logged on to the system and selected ‘update benefits’ option
- **Basic course:**
  - ✓ System retrieves employee account from employee account db
  - ✓ System asks employee to select medical plan type; include Update Medical Plan.
  - ✓ System asks employee to select dental plan type; include Update Dental Plan.
  - ✓ …
- **Alternative courses:**
  - ✓ If health plan is not available in the employee’s area the employee is informed and asked to select another plan...

ORGANISING USE CASES

How to organize collections of use cases or use case diagrams:

System boundary and hyperlinking:

- ✓ **Use boundary box to model context of system (context model).**
- ✓ **Hyperlink to detailed views of use cases presented in context model.**
- ✓ **Similar to DFD context and level-1 diagrams.**

Note that a *use case model* is composed of a collection of use case diagrams.

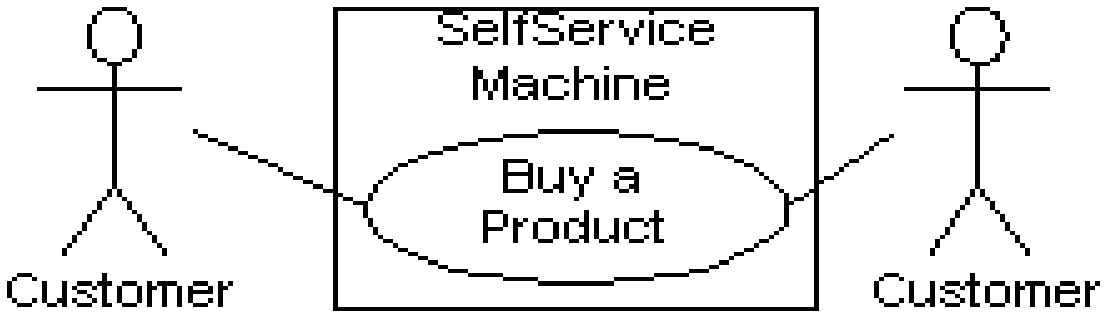
USE CASE DIAGRAM - EXAMPLE

Self-service machine

In this example we're going to model out use case diagrams for a self-service machine.

The main functions of self-service machine is to allow a customer to buy a product(s) from the machine (candy, chocolate, juice...). Every user that you asked for a set of scenarios happening during usage of machine, can tell you that main use case can be labeled as "Buy a product". Let's examine every possible scenario in this use case. These scenarios would be revealed through conversations with users.

THE "BUY A PRODUCT" USE CASE



Buy a product	
Actor initiates the use case	customer
Preconditions	hungry or thirsty customer
Steps in the scenario	First of all the customer inserts money into the machine, selects one or more products, and machine presents a selected product(s) to the customer.
Postcondition	product from the machine
Actor who benefits from the use case	customer

But if we think a little, others scenarios immediately come to mind. It's possible that the self-service machine is out of one or more products, or the machine hasn't the exact amount of money to return to customer.

-Are we supposed to handle with these scenarios?

If we are, then let's return at the moment when the customer inserts money into machine and enters his or her selection. After this imagine that machine is out of brand. In this case it's preferable to present a message to the customer that machine is out of brand and allow him or her to make another selection or return money back. If incorrect-amount-of-money scenario has happened, then self-service machine is supposed to return original amount of money to the customer. The precondition here is the customer, and postcondition is either product from the machine or the returned money.

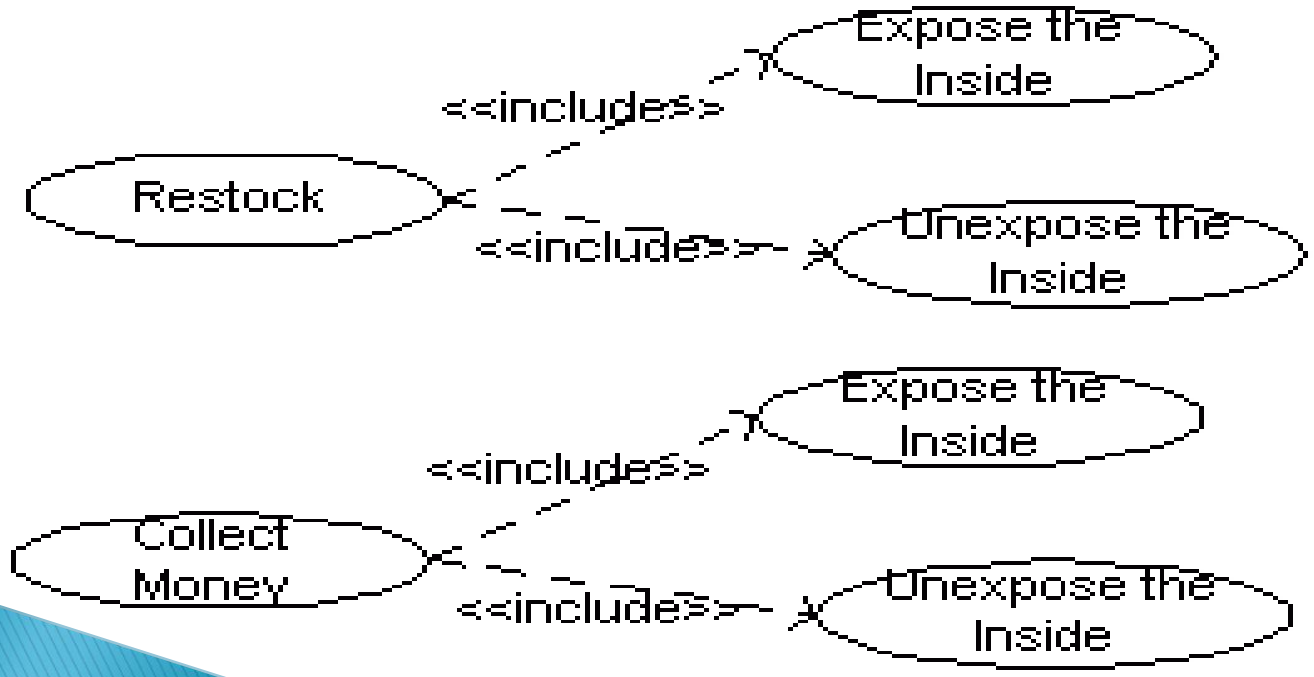
THE "RESTOCK" USE CASE

Restock	
Actor initiates the use case	supplier
Preconditions	the passage of the interval
Steps in the scenario	supplier must do every time interval (say, one or two weeks) are: Supplier unsecured the machine, opens the front of the machine, and fills each brand's compartment to capacity. (The supplier may fill each brand according to consuming of article). Then he/ she close the front of the machine and secure it.
Postcondition	supplier has a new set of potential sales
Actor who benefits from the use case	supplier

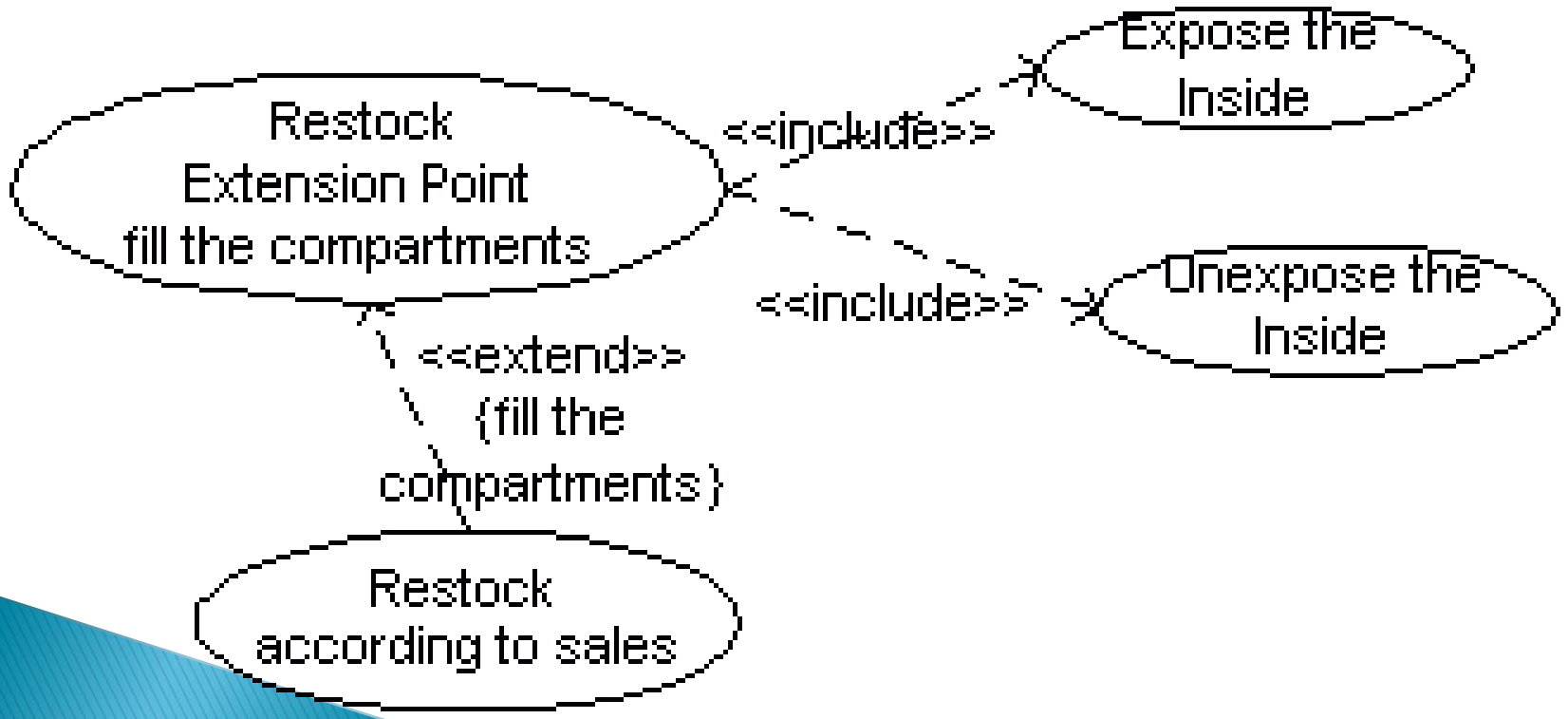
THE "COLLECT MONEY" USE CASE

Collect Money	
Actor initiates the use case	collector
Preconditions	the passage of the interval
Steps in the scenario	collector must do are same as the steps of supplier, but the collector don't deal with products, he/ she deals with money. When the time interval has passed this person collects the necessary amount of money from the machine.
Postcondition	the money in hands of the collector
Actor who benefits from the use case	collector

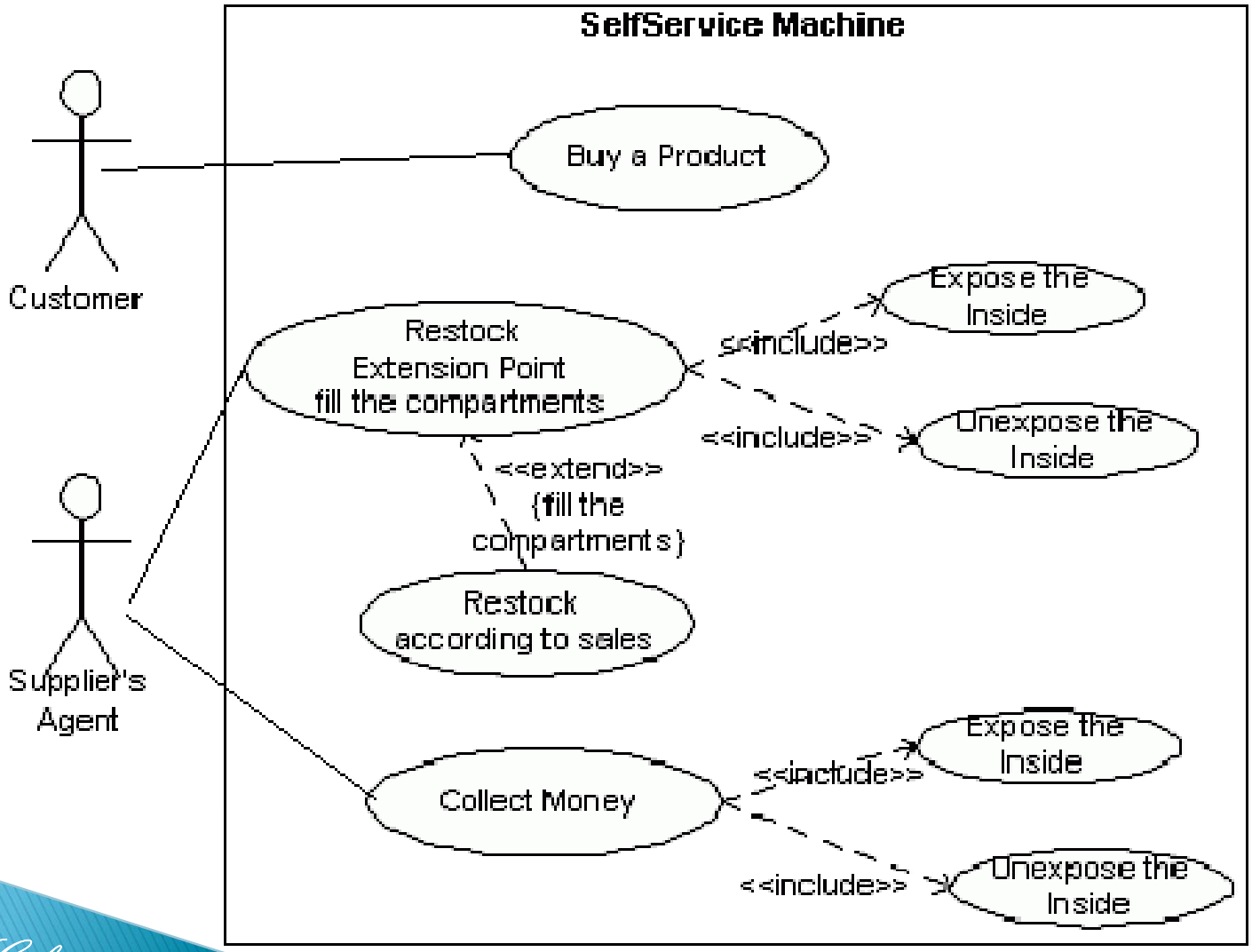
The steps of unsecuring the machine, front opening, closing and securing the machine are the same that supplier and collector must do. This is a good place to include a use case. Let's combine the "unsecure" and "pull open" steps into use case called "Expose the inside" and the "close machine" and "secure" with use case "Unexpose the inside". By including a use case Restock and Collect use cases may look as this:



The Restock use case could be basis of another use case: "Restock according to sales". Here supplier may fill up brands with new products according of sale of those products. This is an extension of a use case. After inclusion and extension the Restock use case can be:



SELF-SERVICE MACHINE



CASE STUDY

Courseware System Description

For this case study, the task is of constructing the design elements for a system that can be used to manage courses and classes for an organization that specializes in providing training. The name of the system is Courseware System. The organization offers courses in a variety of areas such as learning management techniques and understanding different software languages and technologies. Each course is made up of a set of topics. Tutors in the organization are assigned courses to teach according to the area that they specialize in and their availability. The organization publishes and maintains a calendar of the different courses and the assigns tutors every year. There is a group of course administrators in the organization who manage the courses including course content, assigning courses to tutors, and defining the course schedule. The training organization aims to use the Courseware System to get a better control and visibility to the course management and to also streamline the process of generating and managing schedules for different courses.

COURSEWARE OVERVIEW

The following terms and entities are specific to the system:

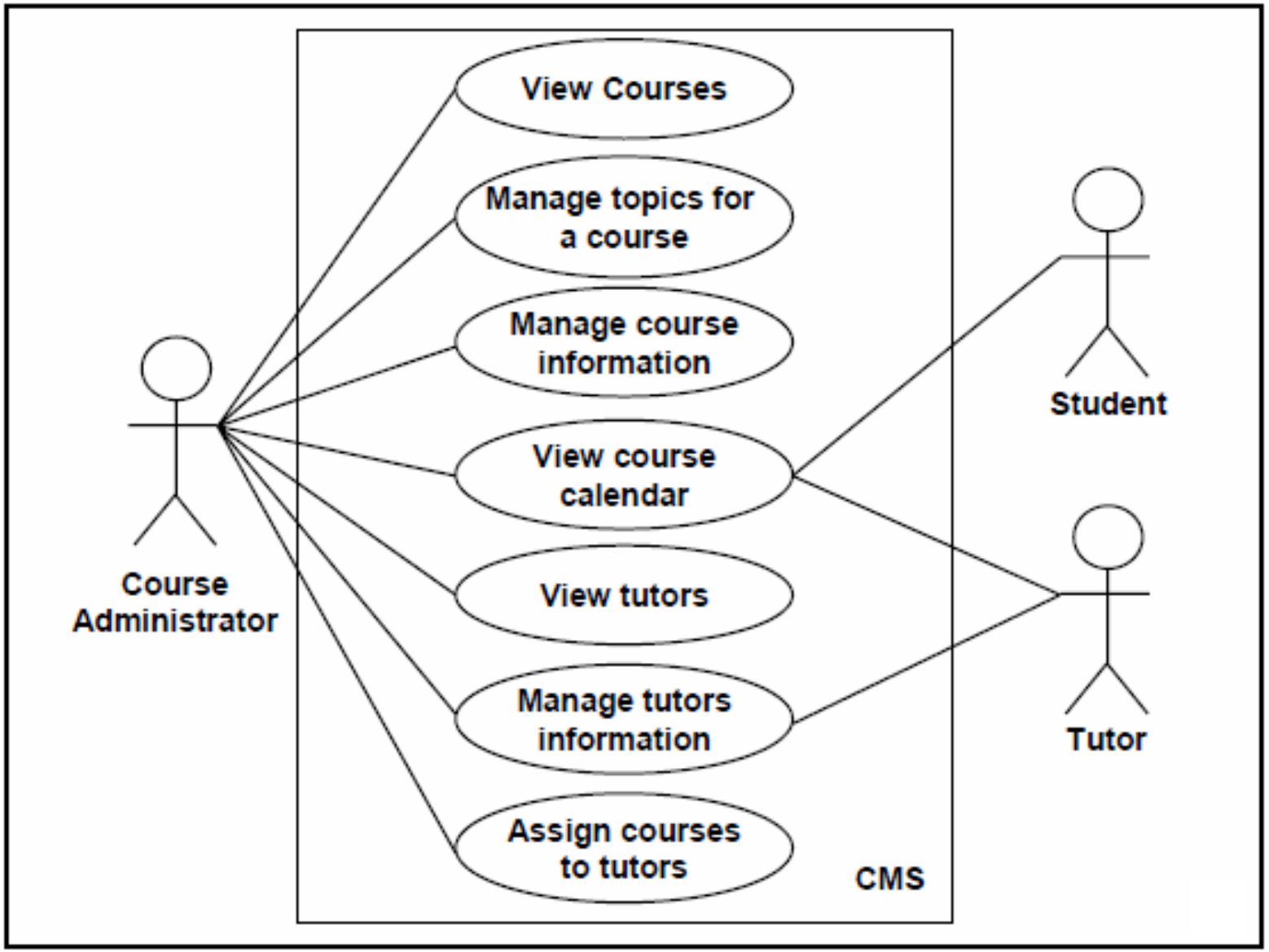
- **Courses and Topics that make up courses**
- **Tutors that teach courses**
- **Course Administrators who manage the assignment of courses to tutors**
- **Calendars and Course Schedules that are generated as a result of the work performed by the course administrators**
- **Students who refer to Calendars and Course Schedules to decide which courses they wish to take up for study**

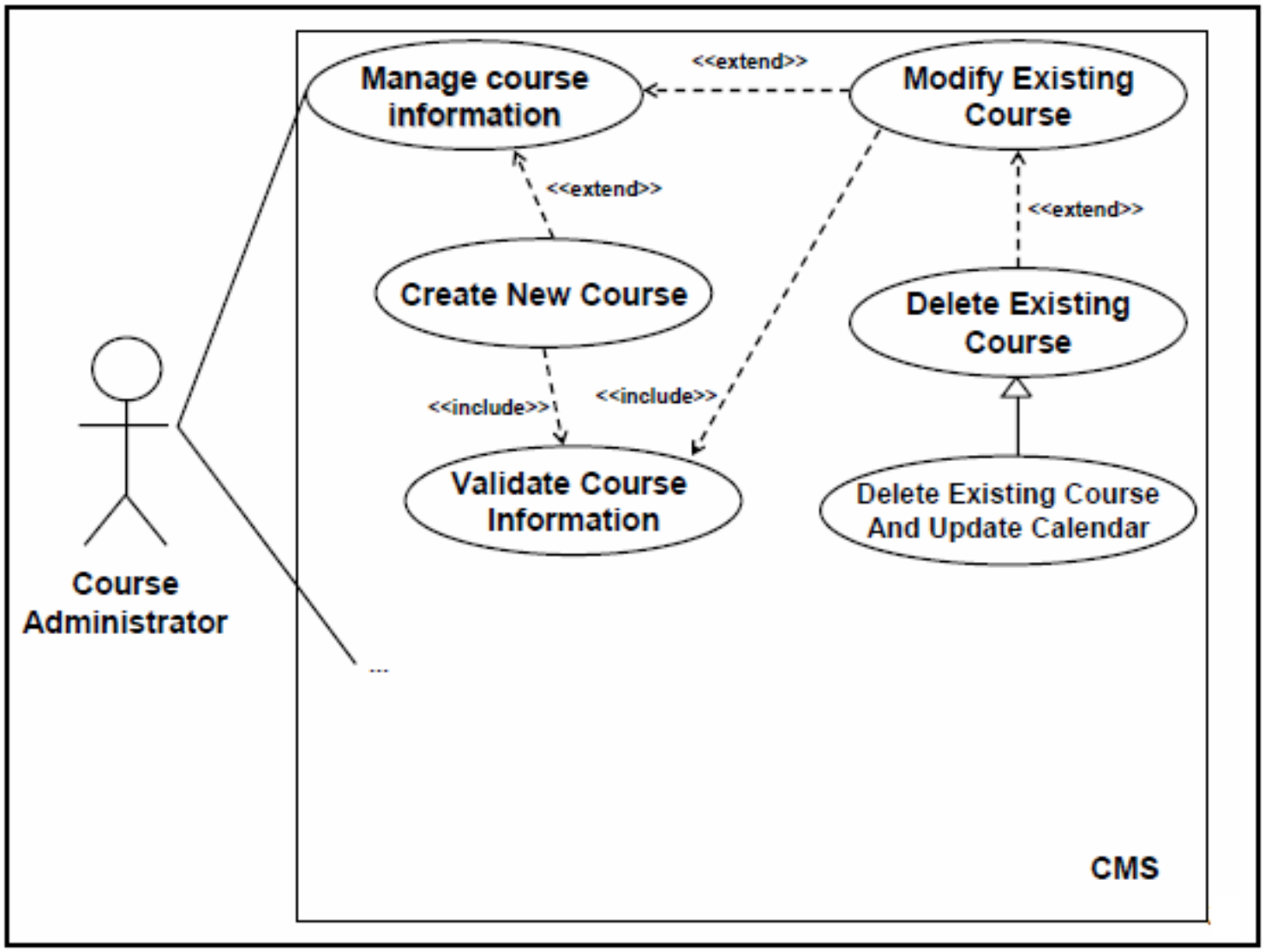
COURSEWARE ACTORS AND USE CASES

- **Actors:**

  Tutor, Student, Course Administrator (main actor)

- **Use Cases (primary business: secondary user)**
  - ✓ **Manage courses: View courses, Manage topics for a course, and Manage course information**
  - ✓ **Manage tutors: View course calendar, View tutors, Manage tutor information, and Assign courses to tutors**


View Courses
Manage topics for a course
Manage course information
View course calendar
View tutors
Manage tutors information
Assign courses to tutors
CMS
Course Administrator
Student
Tutor


Manage course information
<<extend>>
Modify Existing Course
<<extend>>
<<extend>>
Create New Course
Delete Existing Course
<<include>>
<<include>>
Validate Course Information
Delete Existing Course And Update Calendar
Course Administrator
...
CMS

QUESTIONS FOR THE WEEK

- **Describe the purpose of *use case diagrams*. In the context of systems development, how can they be utilised?**
- **Explain the difference between the <<include>> and <<extend>> stereotyped associations. Use examples to clarify your answers if required.**
- **What do *extension points* denote?**
- **How can the notion of *generalisation* be incorporated in use case diagrams?**
- **Define the following *elements* of use case diagrams. Clarify each of your answers with your own examples:**
  - ✓ **Actor**
  - ✓ **Use case**
  - ✓ **Association**
  - ✓ **Boundary**

Questions