Use Case Diagram and Relationships

Agenda

- Use Case Diagram
  - Actors
  - Use Cases
    - Specification of Use Cases

What are Use Case Diagrams

- The integration of business knowledge with the development specification is a requirement
- The development organization knows the specifications for developing a module, but does not know who will interact with these modules and for what purpose
- Use Case Diagrams depicts the human interaction with the system to give the context of who uses, which part of the system and for what purpose

- Use case diagrams are behavior diagrams used to describe a set of actions (use cases) that systems perform in collaboration with one or more external users of the system (actors)
- Use case diagrams are mainly used to gather requirements of a system and identify the internal and external factors influencing the system.

Actors

The use case diagram is composed of

- Actors
- Use cases and their specifications
- Relationships between Use cases

- Actors are entities that interface with the system
- They can be people or other systems
- Actors are depicted as stylized stick figures
- This stick figures are used as stereotypes to depict many models at the same time like process, thread, meta-class, power-type etc labeled with guillemets <<>>
- The roles the actors play in the system is important, not their real world identity

Actors

Actors can be classified as

- **human**: e.g. novice/trained user; system administrator
- **non-human**: e.g., fax, e-mail
- **primary**: ultimate user of the system
- **secondary**: ensures the correct functionality of the system
- **active**: initiates use cases
- **passive**: corresponding use case is initiated by the system

How can Actors be identified?

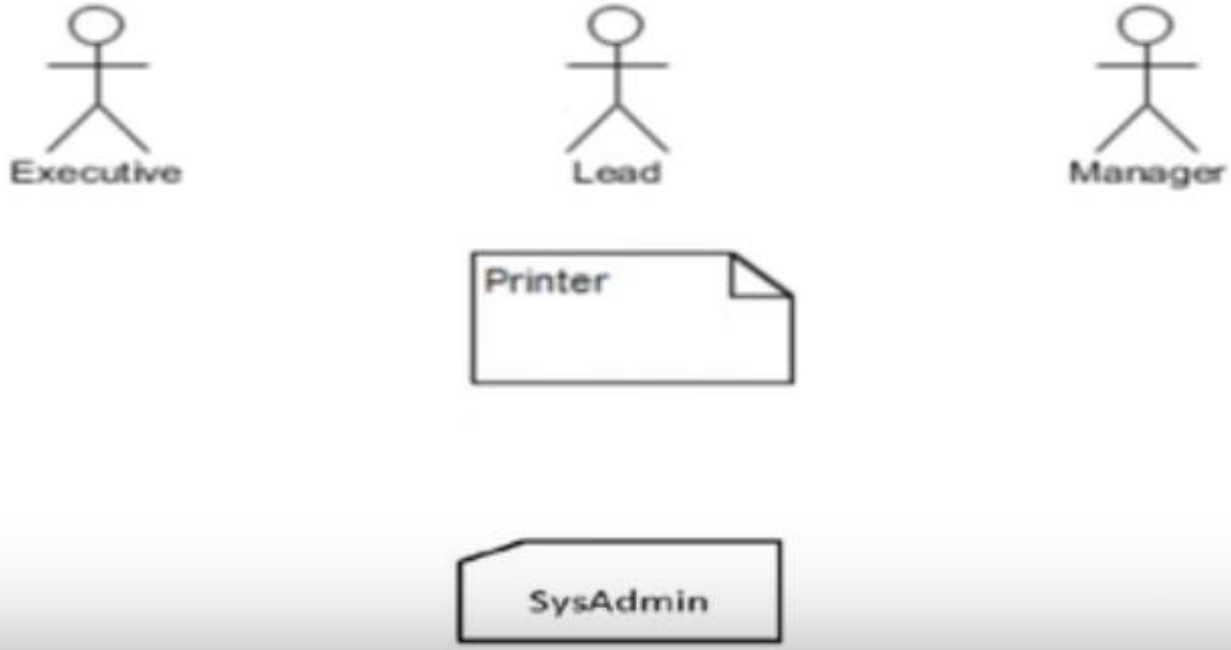
- Who uses the essential use cases?
- Who needs system support in order to fulfill the daily tasks?
- Who is responsible for system administration?
- What are the external devices/software systems the system has to communicate with?
- Who is interested in the results of the system?

Leave Management System - Nouns

Company	Attendance	Leave	Employees
Contributors	Lead	Executive	Manager
Leave Rules	Days	Year	Name
Type of Leave	Period	Absence	Holiday
PL	CL	EL	DL
SL	ML	LWP	UL
Pre-approval	Month	Service	Quarter
Medical Certificate	Parenthood Certificate	Disciplinary Action	Administration Function
Daily Attendance	Personal Details	Calender Year	Batch Task
Account	Balance	Designation	SysAdmin
Parent	Salary	Week	List
Privilege	Right	Login ID	Leave Status
Employee Code			

Actors in Leave Management System

- In the LMS, the human actors are Manager, Lead and Executive.
- The non human actors is the printer
- A Secondary actor is the SysAdmin



Use Case in Leave Management System

- Use cases represent what the actors want your system to do for them
- Each use case is a complete course of events to be executed in the system from a user's perspective
- Use cases can contain short descriptions course of events in the system from a user's perspective

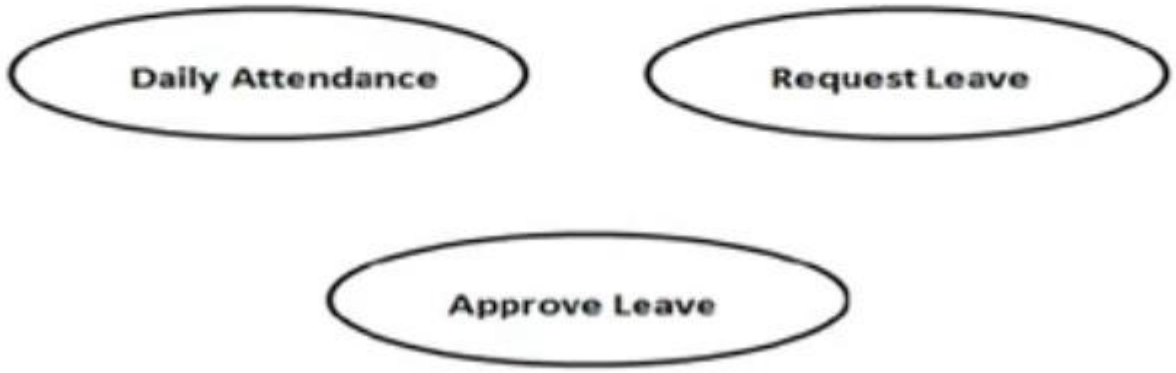
Leave Management System - Verbs

Wants	Manage	Work	Report
Approve	Regret	Credit	Join
Prorate	Cross	En-cash	Paid
Allow	Send	Need	Become
Enjoy	Avail	Proceeding	Employ
Consider	Deduct	Provide	Request
Cancel	Check	Export	Revoke
Debit	Adjust	Perform	Hire
Fire	Generate	Leave En-cashment	Can be Availed
Can be Clubbed	Can't be Availed	Can't be Carried forward	Can't be Clubbed
Can't be Continued	Accumulated Up	Proposed for	Join Back
Doesn't Draw	Can be Revoked	Leave Credited	

Many extracted verbs are in derived forms – so we extract the unique ste

Use Cases in Leave Management System

- Shown below are important use cases executed by actors of Leave Management System. Request Leave, Daily Attendance are use cases, that is, functionality which the all the three human actors need from the system. Similarly approve leave use case executed by only Lead and Manager



Specification of Use Case

Use Case Section	Comment
Use case Name	Start with a verb
Scope	The system under design
Level	User goal or sub function
Primary actor	Calls on the system to deliver its services
Stakeholders and interests	Who cares about the UC and what do they want ?
Preconditions	What must be true on start .
Success Guarantee	What must be true on successful completion
Main success scenario	Unconditional happy path scenario
Extensions	Alternate scenarios of success or failure
Special requirements	Related non functional requirements
Technology and data variations list	Varying I/O methods and data formats
Frequency of occurrence	Influences investigation, testing and timing of implementation
Miscellaneous	Such as open issues

Specification of a Use Case

- A use case can be specified in the following manner. Depicted below for Request Leave.
  - **Use Case Name:** Request Leave
  - **Use Case Purpose:** The Executive, Lead and Manager, all of them uses this functionality to request for leaves in the system.
  - **Use Case Pre-condition:** User has login id and password to enter the system
  - **Use Case Post-condition:** A new leave request is entered in the system for the user
  - **Failure Conditions:** User does not have valid credentials to enter the system, the user do not have sufficient leave balance
  - **Actors:** Lead, Executive, Manager
  - **Optimistic Flow:**
    - The Executive, Lead and Manager checks for the leave balance.
    - If leave balances are available, leave request added.

Relationship among Use Cases

- Relationship among Use-Cases
  - Includes
  - Extends
  - Generalization

- Use-Cases share various kinds of relationships
- A relationship between two Use-Cases is basically a dependency between the two Use-Cases
- Defining the relationship between two Use-Cases is the decision of the modeler of the use case diagram
- We discuss the following three relationships among Use-Cases
  - <<include>>
  - <<extend>>
  - Generalization

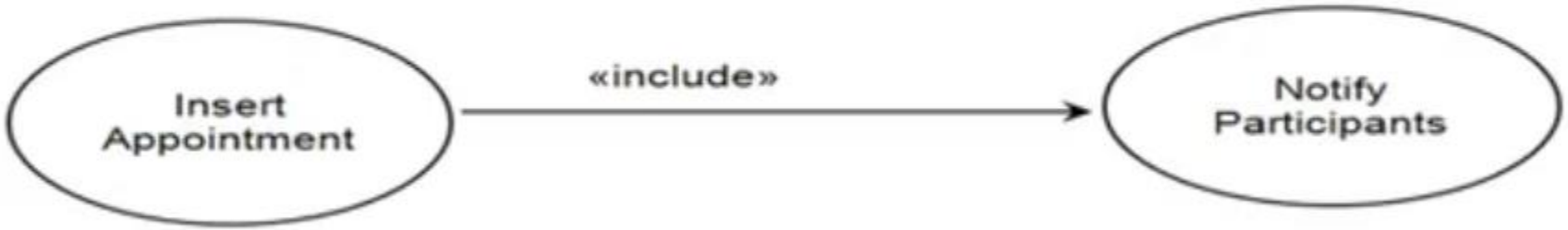
Relationship among Use Cases << include >>

- The <<**includes**>> relationship involves one Use-Case including the behavior of another Use-Case in its sequence of events and actions
- Thus, the includes relationship explores the issue of reuse by factoring out the commonality across Use-Cases
- <<**includes**>> relationship:

  - the behavior of B is included into A
  - the included use case B is necessary to ensure the functionality of the base use case A

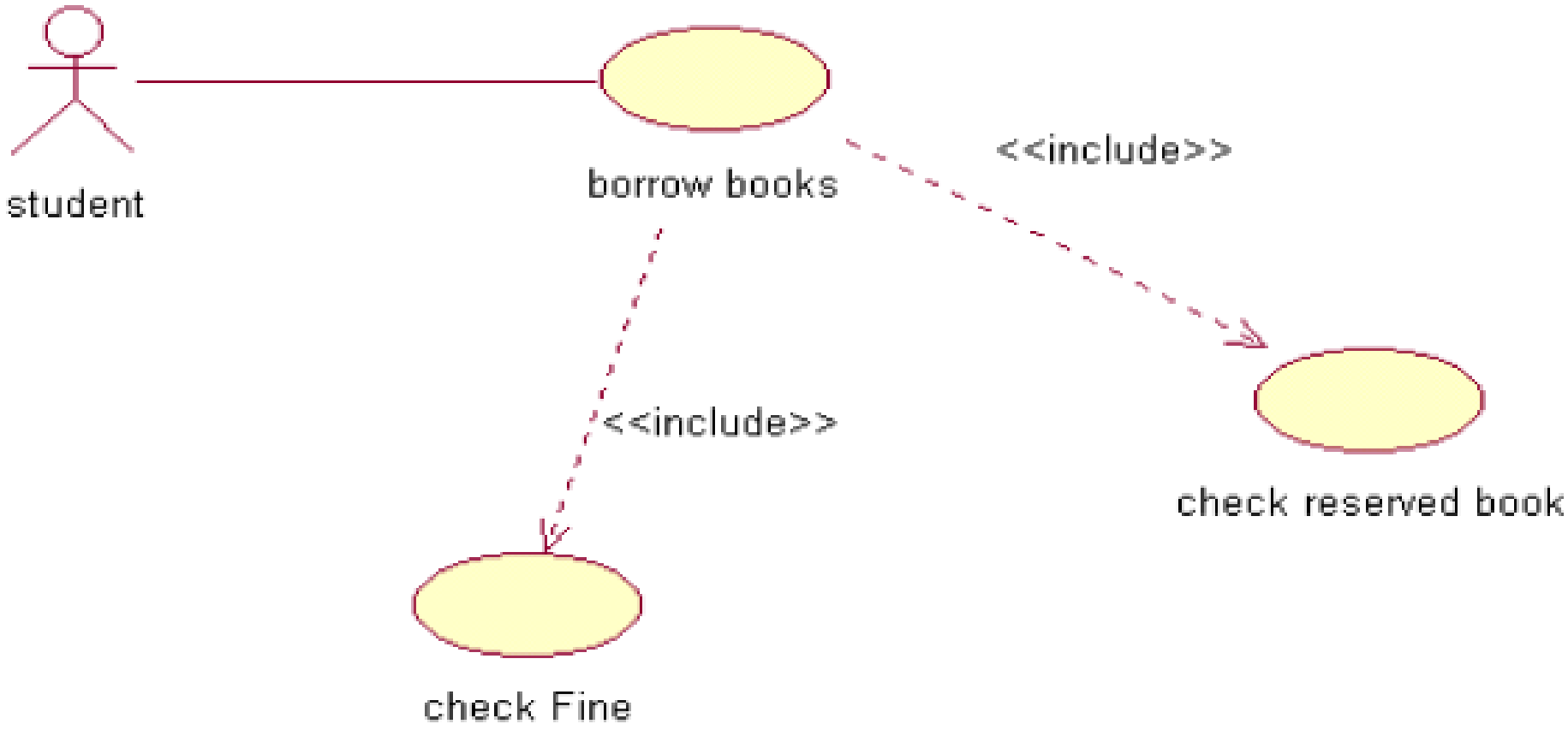
Relationship among Use Cases << include >>

- In a calendar, if an appointment is inserted by *Insert Appointment*, the participants will be notified of appointment by included Use-Case *Notify Participants*
- The included Use-Case *Notify Participants* is necessary to ensure the functionality of the base Use-Case *Insert Appointment*



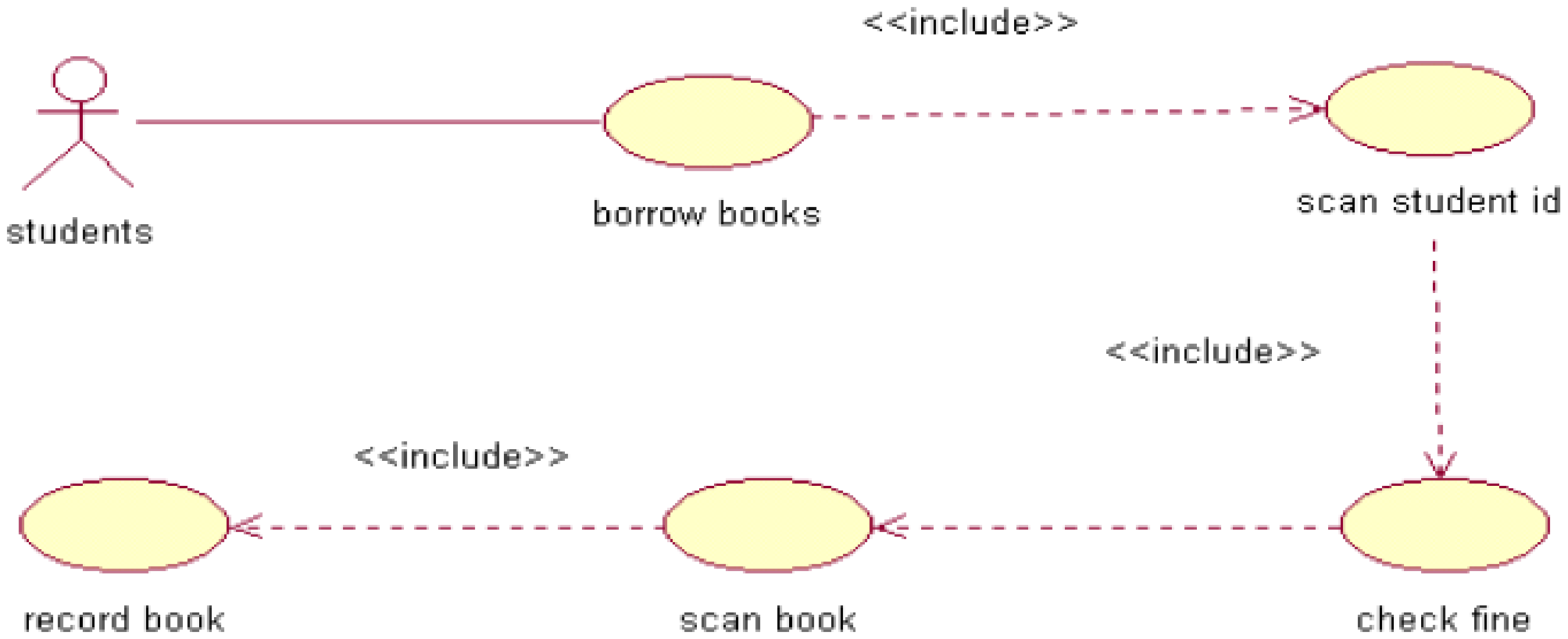
- The Use-Case *Notify Participants* may be included in other Use-Cases – for example, in *Cancel Appointment*

Relationship among Use Cases << include >>



Relationship among Use Cases << include >>

- Example of Bad Use Case



Relationship among Use Cases << include >> in LMS

- *Validate Leave* Use-Case is included in the Use-Case *Request Leave*
- The include Use-Cases *Validate Leave* is necessary to complete the functionality of the base *Request Leave*

- The Use-Case *Validate Leave* will be included in *Approve Leave* too – and so on

Relationship among Use Cases << extend >>

- The <<**extend**>> relationship among the Use-Cases is used to show optional system behavior
- An optional system behavior is extended only under certain conditions, known as **Extension Points**
  - <<**extend**>> relationship:

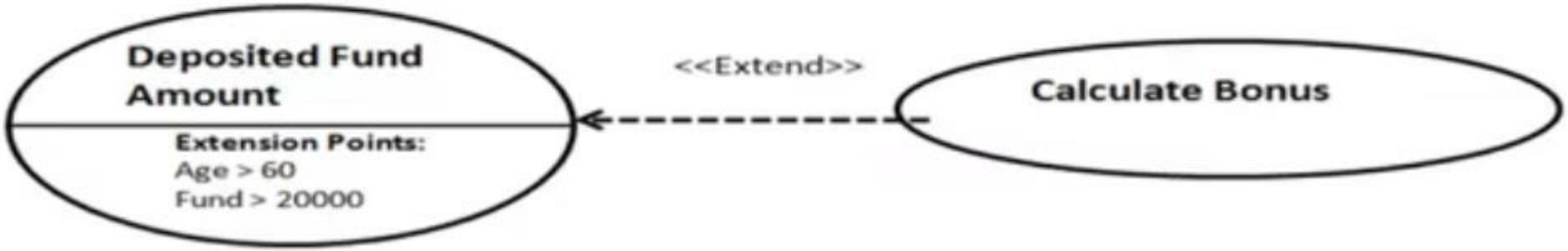
***base* use case;**
is self-contained;
controls, if B is executed or not

***extending* use case;**
is self-contained

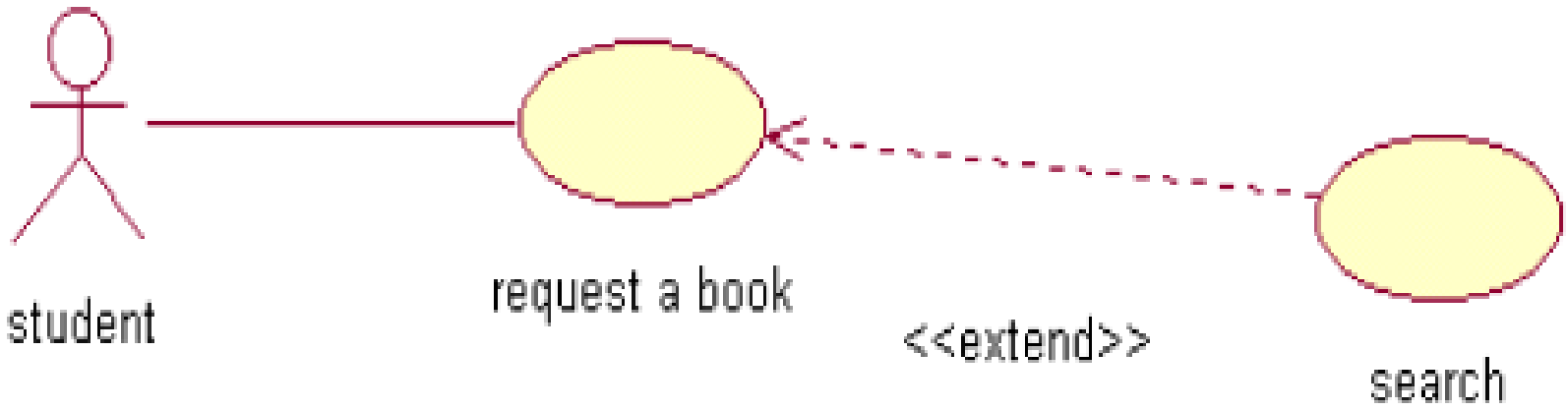
- the behavior of B may be incorporated into A
- the extending Use-Case B may be (but need not be) activated by the base Use-Case A
- **extension points** specify the location where the extending Use-Cases extends the base Use-Case
- the **condition** under which the extending Use-Case is incorporated has to be specified
- more than one extension point can be specified for each Use-Case
- the names of extension points have to be unique
- the names of extension points need not be equal with the names of the extending Use-Cases

Relationship among Use Cases << extend >>

- In a savings bank account, bonus is provided only if the deposited fund is above 20,000 or the depositor is above the age of 60 years
- The behavior of *Calculate Bonus* may be incorporated into *Deposited Fund amount*
- At **extension point**: Age above 60 years, Deposit > 20,000

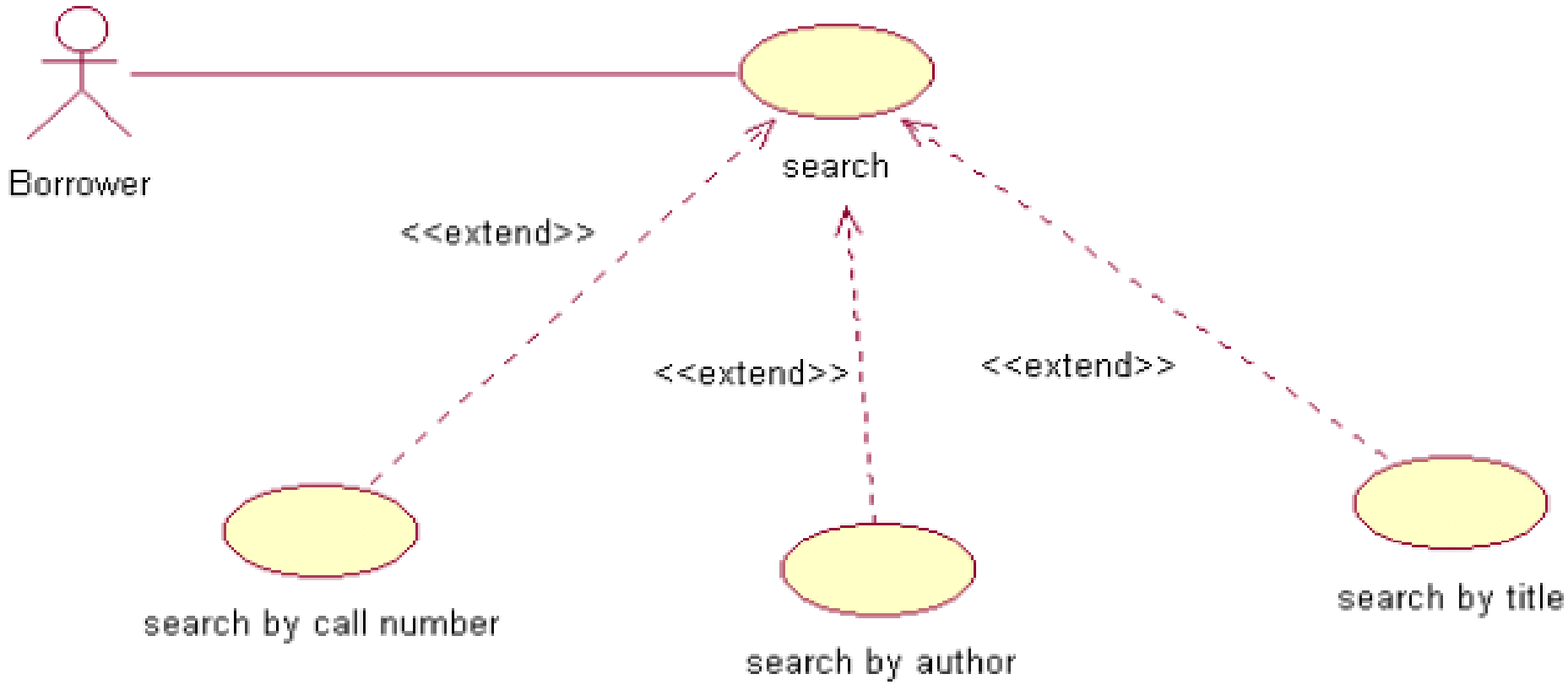


Relationship among Use Cases << extend >>



Relationship among Use Cases << extend >>

- Example of Bad Use Case

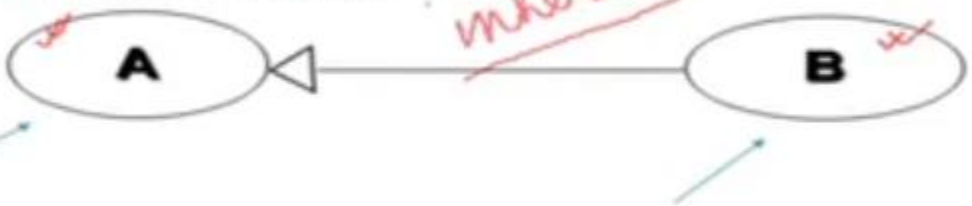


Relationship among Use Cases << extend >> in LMS

- The behavior of *Check Report* may be incorporated into *Approve Leave*
- The extending Use-Case may be (but need not be) activated by the base Use-Case *Approve Leave* at extension points: Medical Leave, Maternity Leave
- Extending Use-Case (*Check Report*) extends the base Use-Case (*Approve Leave*)

Relationship among Use Cases
Generalization among Use Cases– Concept of Hierarchy

- Generalization works the same way with Use-Cases as it does with classes
- The child Use-Case inherits the behavior and meaning of the parent Use-Case
- Generalization helps us to depict the hierarchy present between Use-Cases
- Generalization relationship:



***base* use case;**
is self-contained

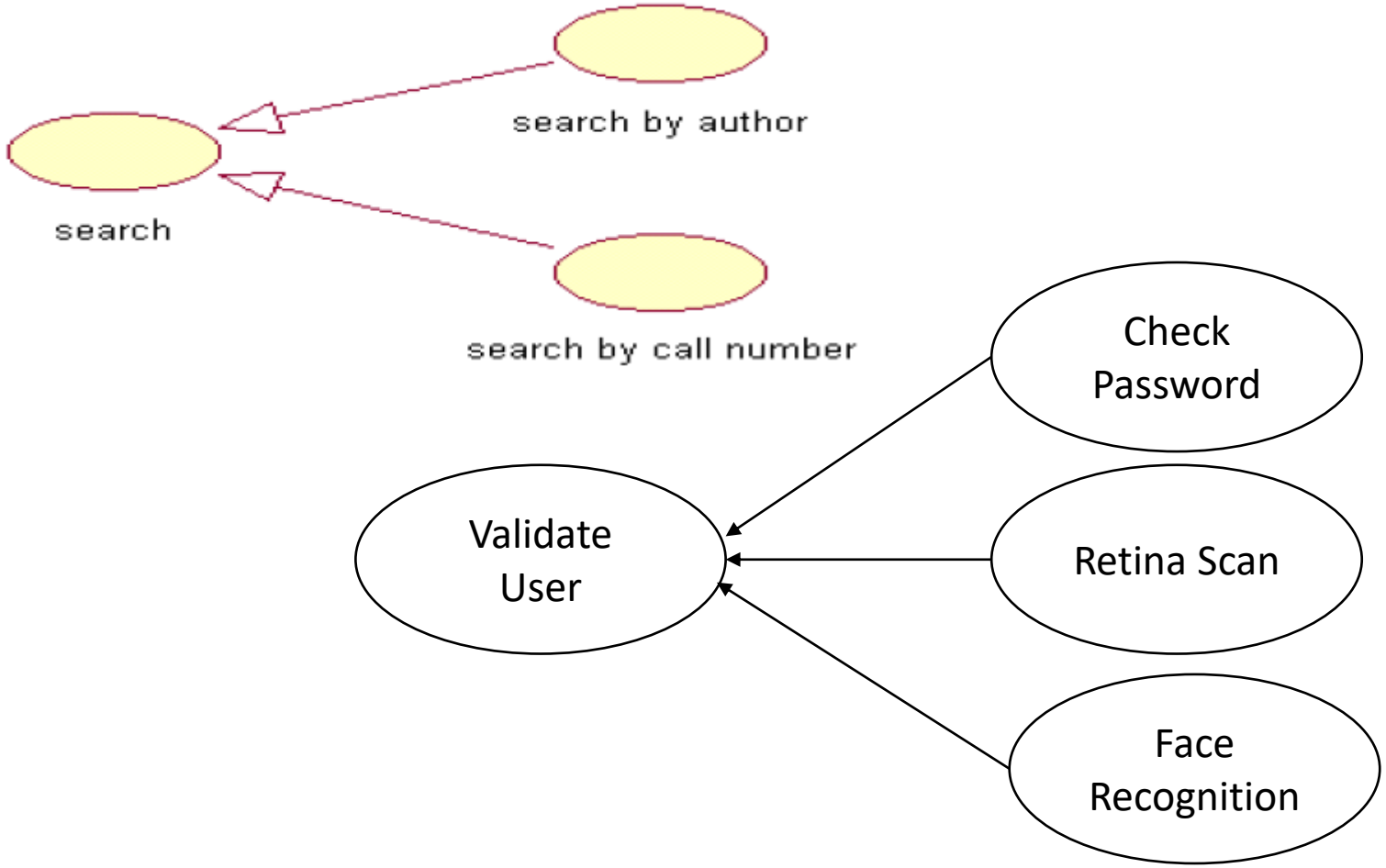
***sub* use case;**
needs A (gets base functionality from A);
controls, what is executed from A
and what gets changed

- Similar to the generalization relationship between classes
- B **inherits** the behavior of A and is allowed to **override** and **extend** it
- B **inherits** all relationships of A
- Modeling of **abstract Use-Cases** is also possible (abstract)

Relationship among Use Cases
Generalization among Use Cases– Concept of Hierarchy

- *Authentication by Fingerprint* Use-Case will inherit the base Use-Case *Authentication*, to include the basic algorithm, but it adds on features like finger print matching

Relationship among Use Cases
Generalization among Use Cases– Concept of Hierarchy

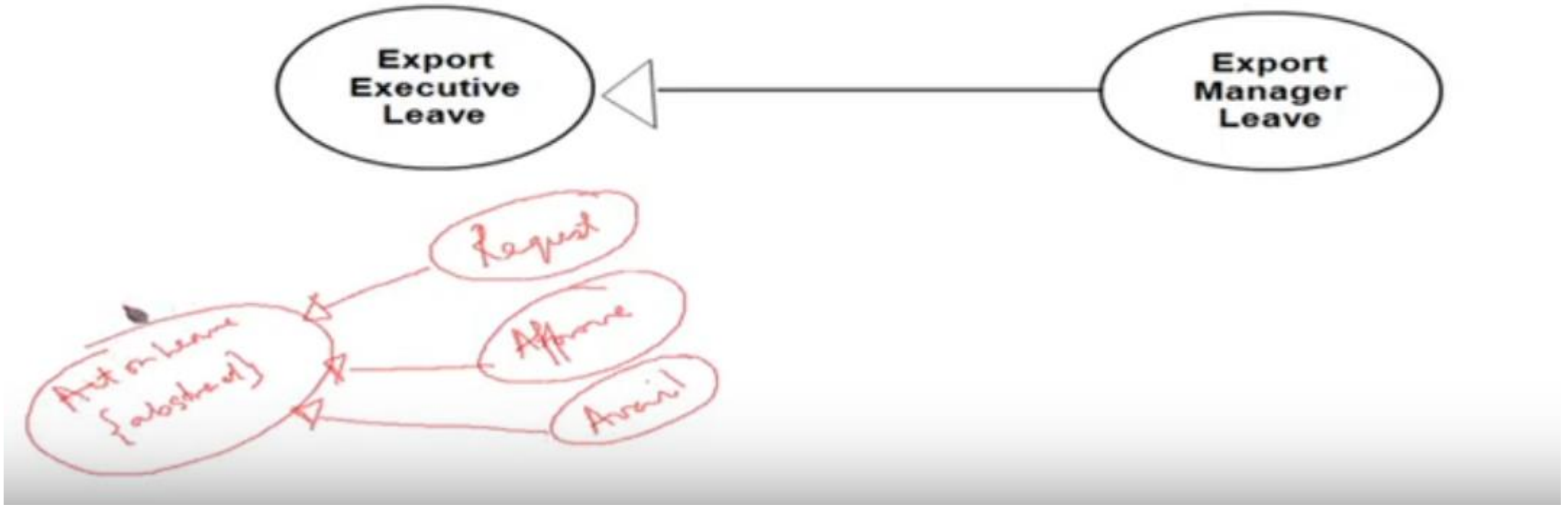


Relationship among Use Cases
Generalization among Use Cases– Concept of Hierarchy

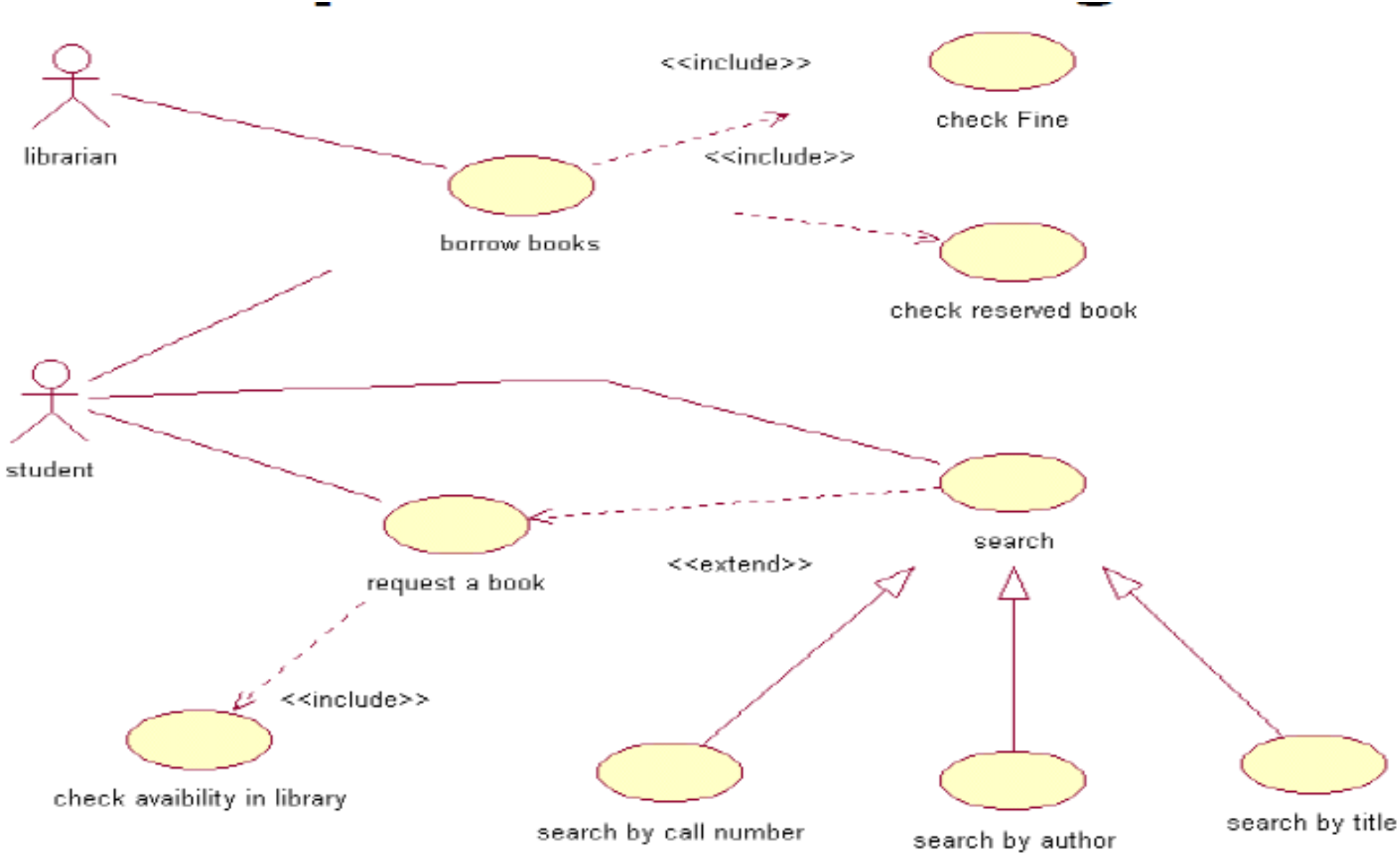
- In LMS, *Export Manager Leave* Use-Case inherits the behavior of the parent Use-Case *Export Executive Leave*, that is, it contains the list of leaves but along with it, it contains additional work responsibilities of each of the executives. Hence a specialized Use-Case

Relationship among Use Cases
Generalization among Use Cases– Concept of Hierarchy

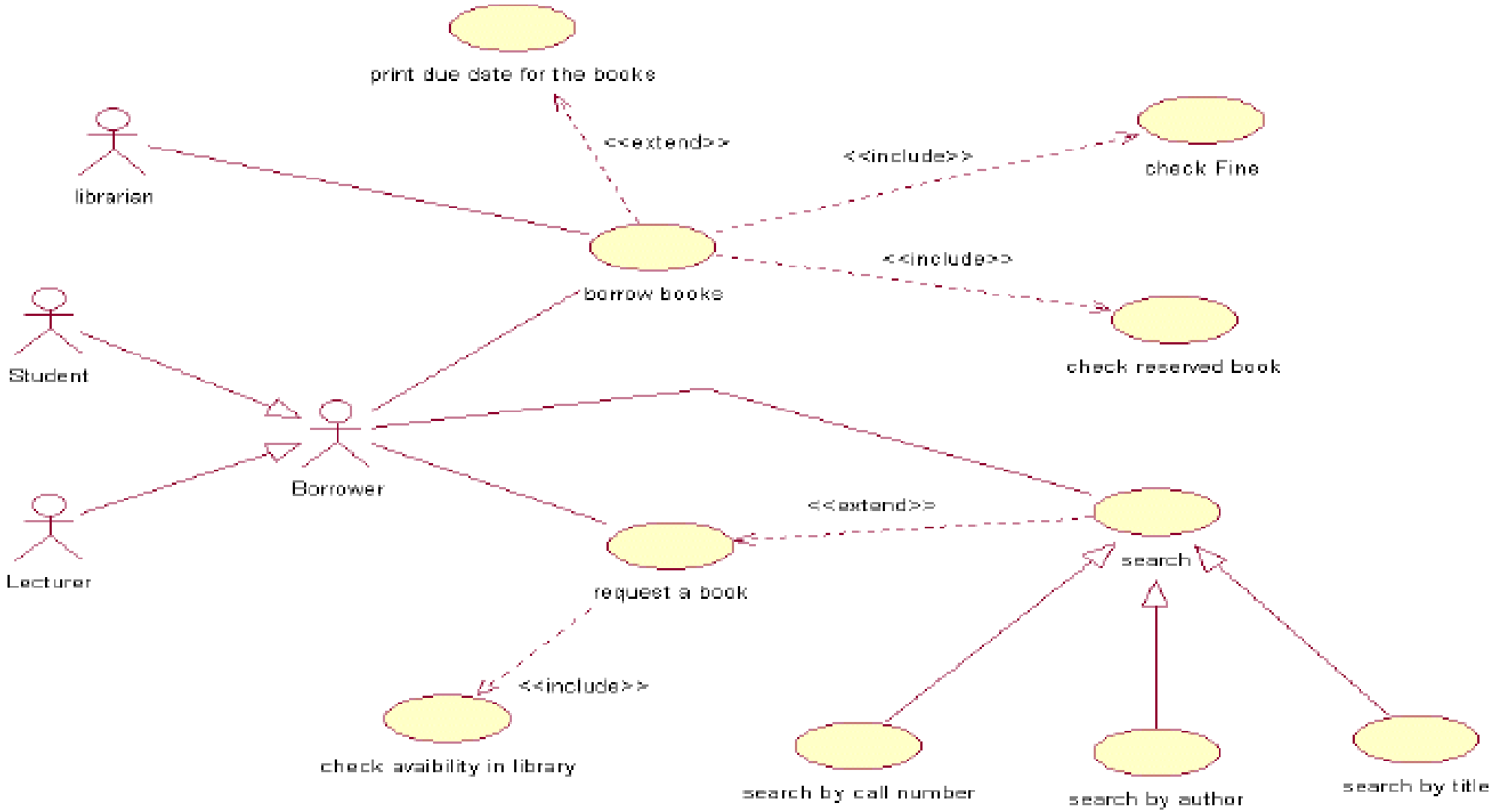
- In LMS, *Export Manager Leave* Use-Case inherits the behavior of the parent Use-Case *Export Executive Leave*, that is, it contains the list of leaves but along with it, it contains additional work responsibilities of each of the executives. Hence a specialized Use-Case



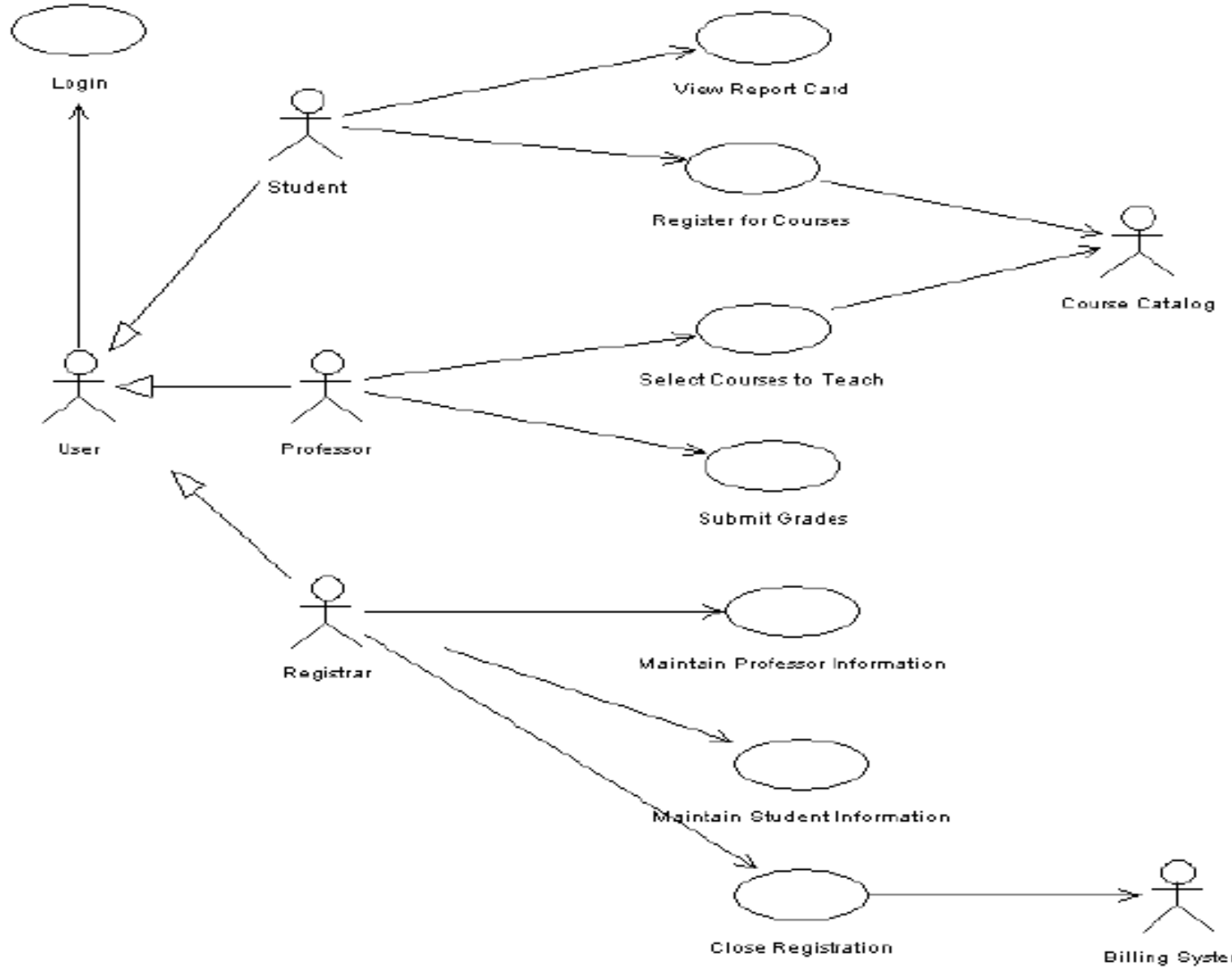
Relationship among Use Cases - Generalization among Actors Concept of Hierarchy



Relationship among Use Cases - Generalization among Actors Concept of Hierarchy



Relationship among Use Cases - Generalization among Actors Concept of Hierarchy



Relationship among Use Cases
Generalization among Actors in LMS
Concept of Hierarchy

- There can be generalization among actors, which can be captured in the use case diagrams
- Generalization among actors, specify that use cases executed by the base actor is inherited by the derived actor
- The derived actor can execute extra use cases

Generalization Among Actors

Summary

- Relationships among Use-Cases are classified and discussed
- Relationships among Use-Cases are identified for LMS

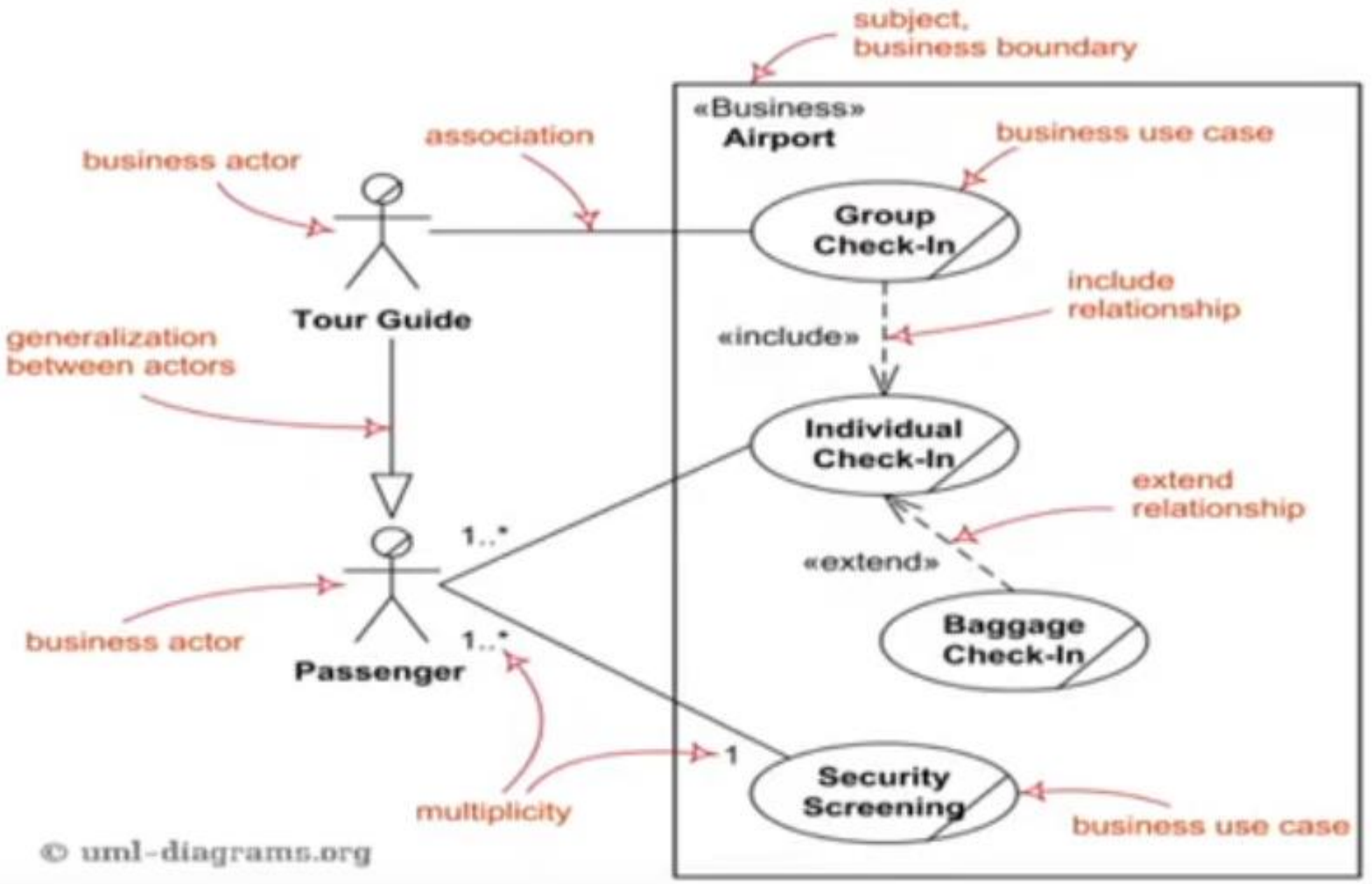
Next

- Exploring the Use-Case Diagrams for Leave Management System (LMS)

For the Leave Management System we shall

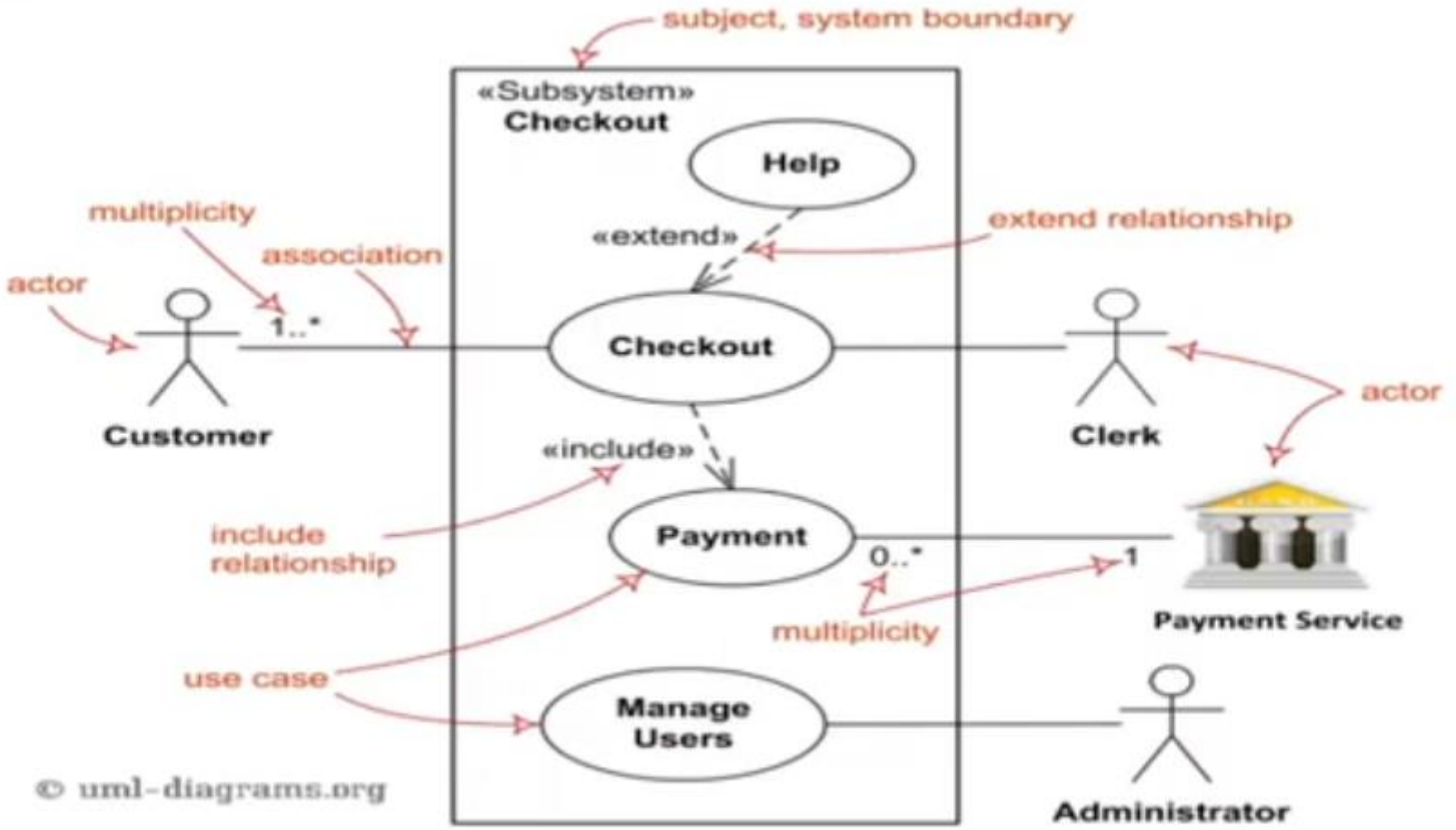
- Identify Actors
- Identify Use-Cases
- Identify the Relationships among Use-Cases
- Build Use-Case Diagram for LMS

Use Case Diagram for Travel Guide Application



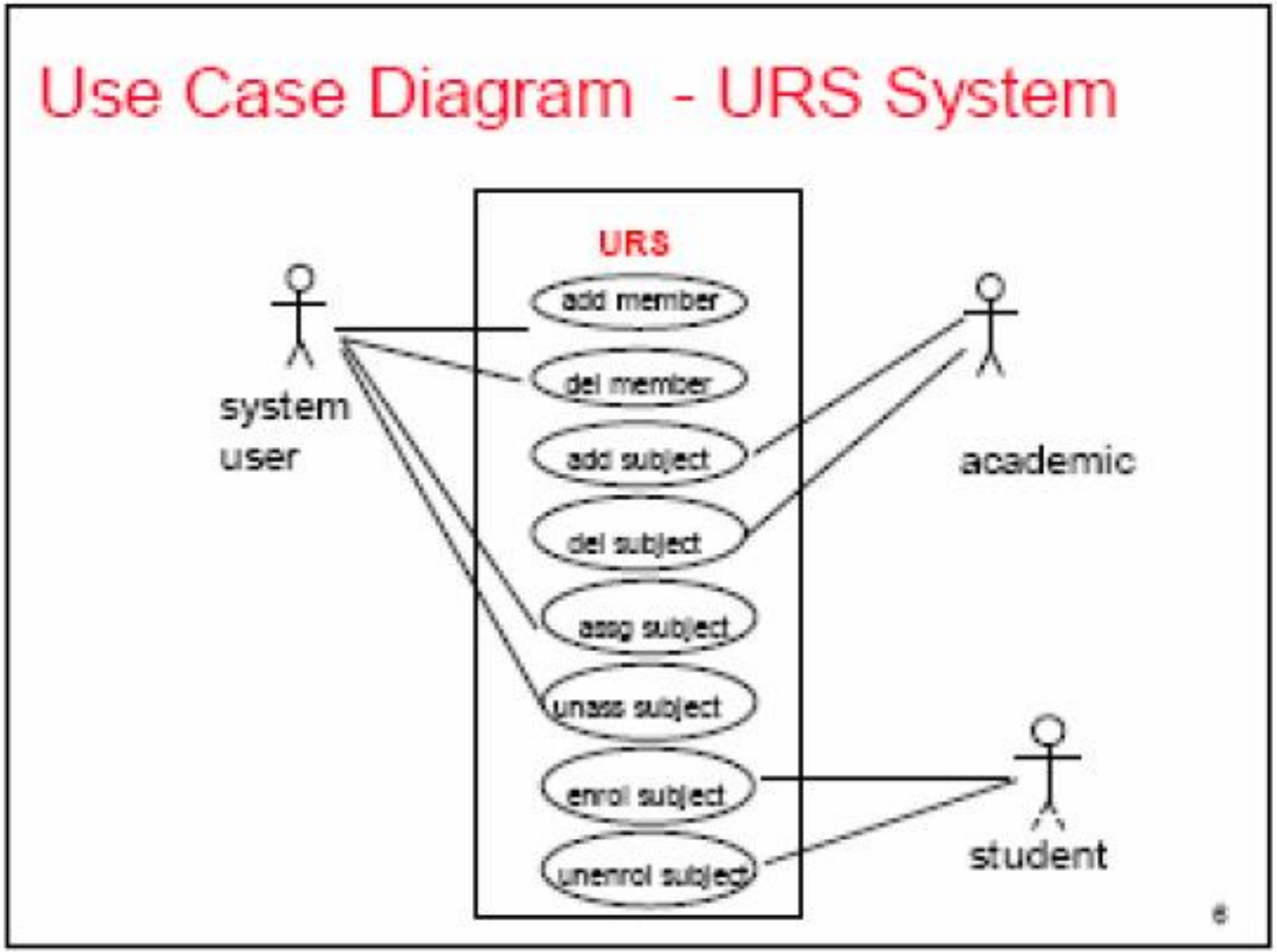
Source: *UML 2.5 Diagrams Overview*: http://www.uml-diagrams.org/uml-25-diagrams.html (10-Aug-16)

Use Case Diagram for Travel Guide Application

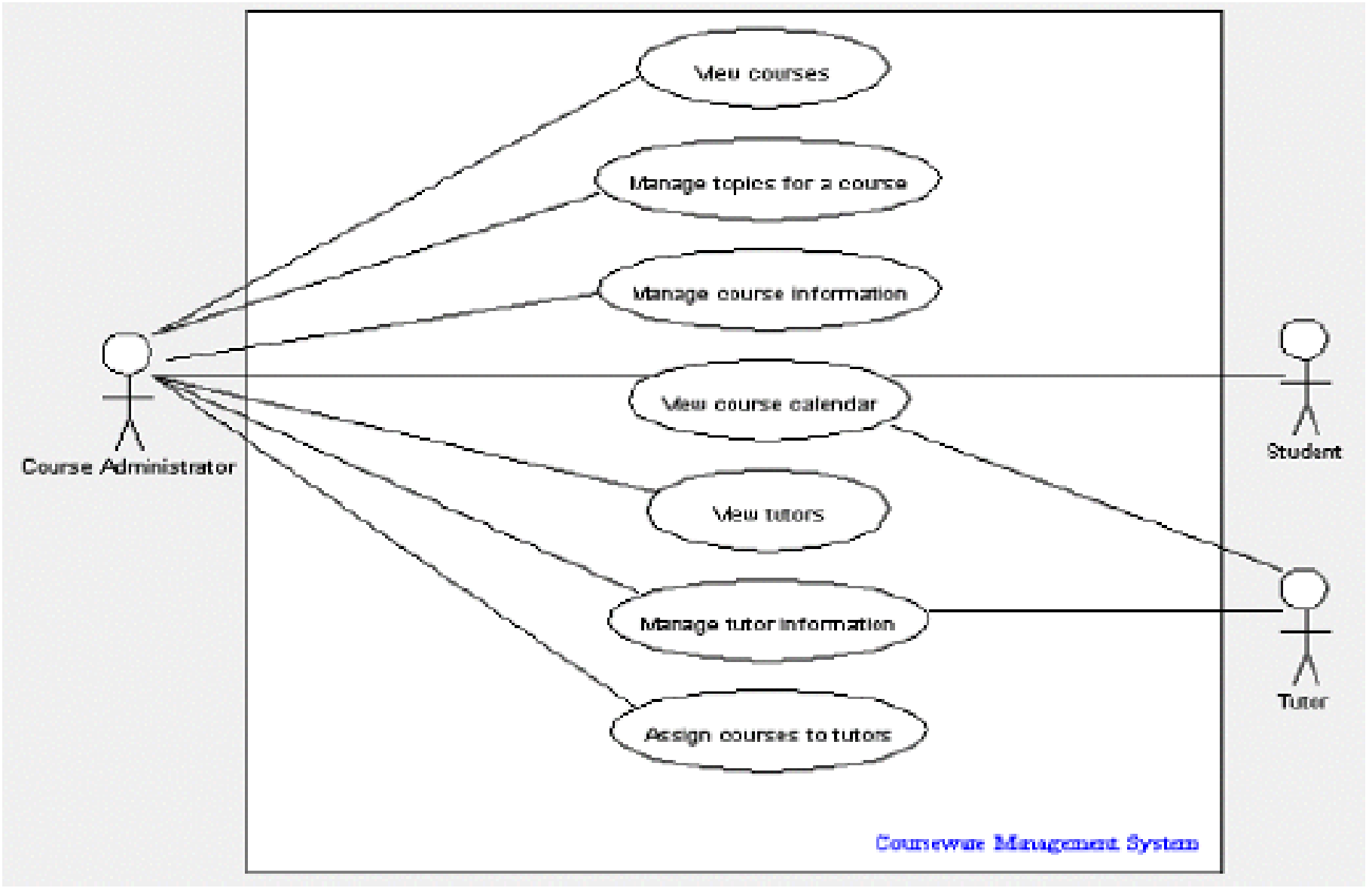


Source: *UML 2.5 Diagrams Overview*: http://www.uml-diagrams.org/uml-25-diagrams.html (10-Aug-16)

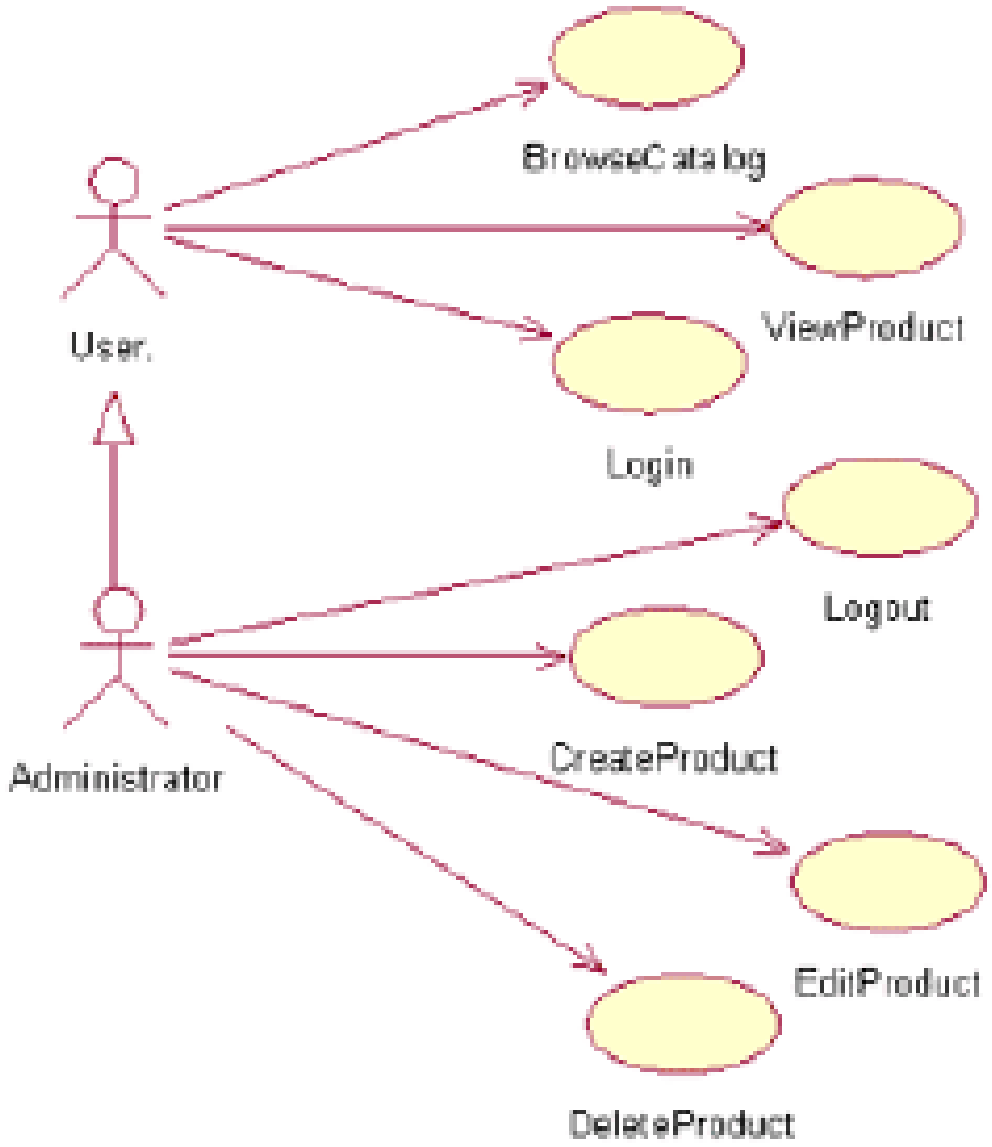
Use Case Diagram for University Record System



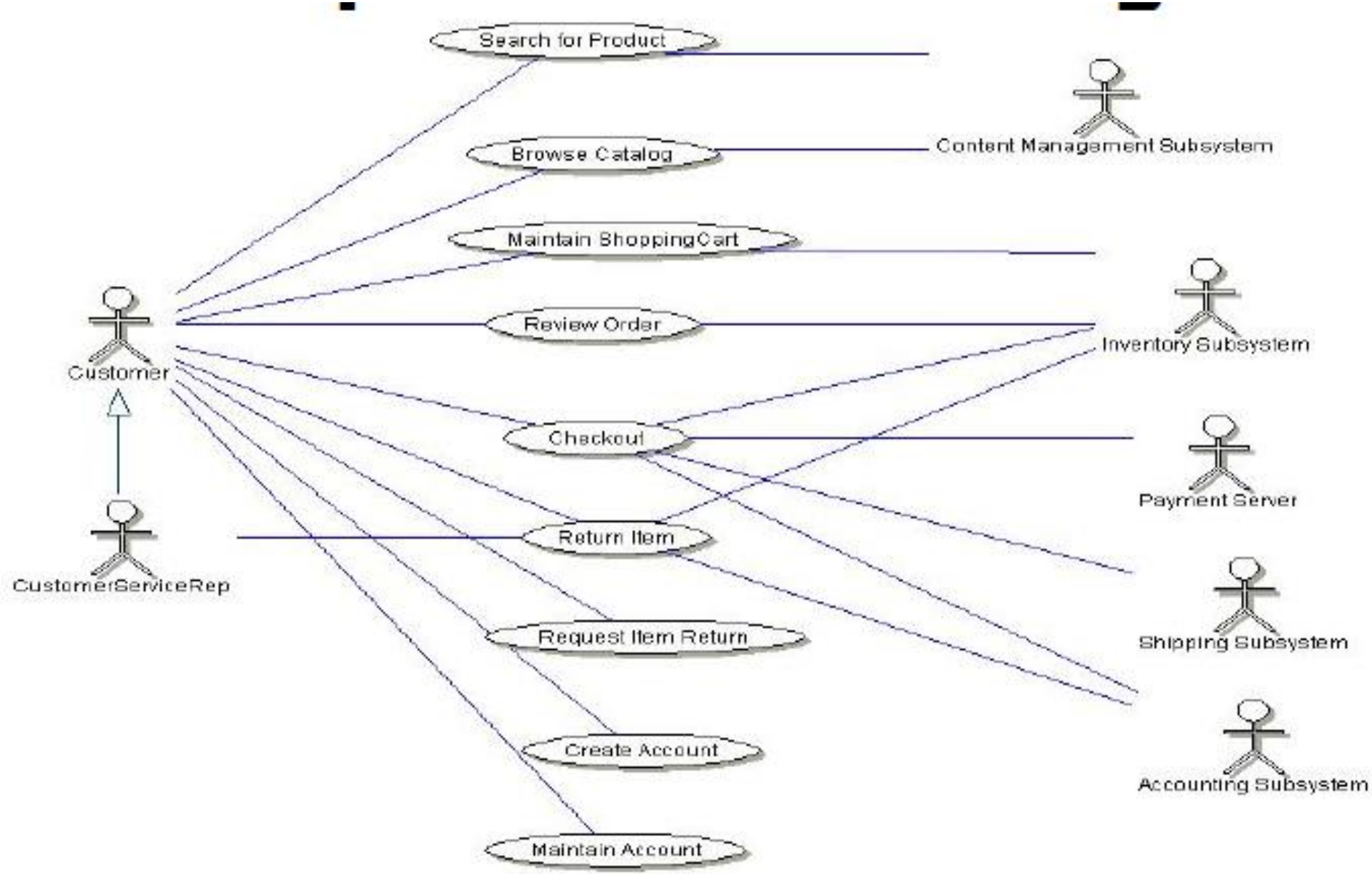
Use Case Diagram for Course Management System



Use Case Diagram for Catalog System



Use Case Diagram for Content Management System



Summary

- Use case diagrams help to integrate business knowledge with the development specification
- Use Case diagram is used to model the various behaviors (usecases) of a system, and the external elements using and executing them (actors)

References

- Source: NPTEL **Object-Oriented Analysis and Design, IIT Kharagpur**
- Prof. Partha Pratim Das Prof. Samiran Chattopadhyay Prof. Kausik Datta

- https://nptel.ac.in/courses/106105153