# UML & Use Cases

Topic # 5

Chapter 6 – Craig Larman

## UML (Unified Modeling Language)

- The **Unified Modeling Language™** (**UML®**) is a standard visual modeling language intended to be used for
  - modeling business and similar processes,
  - analysis, design, and implementation of software-based systems
- UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems.
- UML is a standard modeling **language**, not a **software development process**.
- UML is intentionally **process independent** and could be applied in the context of different processes. Still, it is most suitable for use case driven, iterative and incremental development processes. An example of such process is **Rational Unified Process** (RUP).

## Version of UML (*Reading)

- The current version of the **Unified Modeling Language™** is **UML 2.5**, released in June 2015 [UML 2.5 Specification]. (*2.5.1)
- **UML® specification** (standard) is updated and managed by the **Object Management Group** (OMG™) OMG UML.
- The first versions of UML were created by "**Three Amigos**" - **Grady Booch** (creator of Booch method), **Ivar Jacobson** (Object-Oriented Software Engineering, OOSE), and **Jim Rumbaugh** (Object-Modeling Technique, OMT).

## UML History (*Reading)

- OO languages appear mid 70's to late 80's (cf. Budd: communication and complexity)
- Between '89 and '94, OO methods increased from 10 to 50.
- Unification of ideas began in mid 90's.
  - Rumbaugh joins Booch at Rational '94
  - v0.8 draft Unified Method '95
    - Jacobson joins Rational '95
  - UML v0.9 in June '96

  pre-UML
  - UML 1.0 offered to OMG in January '97
  - UML 1.1 offered to OMG in July '97
    - Maintenance through OMG RTF
  - UML 1.2 in June '98
  - UML 1.3 in fall '99
  - UML 1.5 http://www.omg.org/technology/documents/formal/uml.htm

  UML 1.x
  - UML 2.0 http://www.uml.org/
  - UML 2.5 underway

  UML 2.0
- Rational Rose http://www-306.ibm.com/software/rational/

## Unified Modeling Language (UML)

- An effort by IBM (Rational) – OMG to standardize OOA&D notation
- Combine the best of the best from
  - Data Modeling (Entity Relationship Diagrams); Business Modeling (work flow); Object Modeling
  - Component Modeling (development and reuse - middleware, COTS/GOTS/OSS/…:)
- Offers vocabulary and rules for communication
- *Not* a process but a language

## UML is for Visual Modeling

*A picture is worth a thousand words!*

- standard graphical notations: Semi-formal
- for modeling enterprise info. systems, distributed Web-based applications, real time embedded systems, …


***Specifying & Documenting*** models that are precise, unambiguous, complete
- UML symbols are based on well-defined syntax and semantics.
- analysis, architecture/design, implementation, testing decisions.

***Construction:*** mapping between a UML model and OOPL (Object Oriented Programming Language)

Use Cases
Ivar Jacobson 1994
Use Case Model

***"What will the System do?"***

## Use Case Analysis

- What is an Actor?
- A user or outside system that interacts with the system being designed in order to obtain some value from that interaction
- Use Cases describe scenarios that define the interaction between users of the system (the actor) and the system itself.

## Use Cases

- **What is a scenario?**
  - A scenario is a specific sequence of actions and interactions between the system and actors. It is also called a **use case instance.**
  - It is one particular story of using a system or one path through the use case. For e.g. the scenario of successfully purchasing items with cash.
- **What is a Use Case?**
  - A scenario-based technique in the UML
  - A formal way of representing system functionality, the requirements of the system from the user's perspective. (Illustrates the activities that are performed by the user of the system)
  - representing how a system interacts with its environment
- **Use Case diagram:** It shows a set of use cases and actors and their relationships.

## Components of Use Case Diagram

- Actors
- Use Case
- Relationship
- Boundary

## Actors


**Actors**

The people or systems that provide or receive information from the system;
Could be human beings, other systems, timers and clocks or hardware devices.

Actors that stimulate the system and are the initiators of events are called primary actors (active)
Actors that only receive stimuli from the system are called secondary actors (passive)

Notation

## Actors

**Actors**

- Who/what will be interested in the system?
- Who/what will want to change the data in the system?
- Who/what will want to interface with the system?
- Who/what will want information from the system?

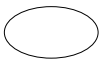
## Four Types of Actors

- **Primary business actor** (Primary Actor )
  - The stakeholder that primarily benefits from the execution of the use case.
  - e.g. the employee receiving the paycheck
- **Primary system actor (Primary Actor)**
  - The stakeholder that directly interfaces with the system to initiate or trigger the business or system event.
  - e.g. the bank teller entering deposit information
- **External server actor** (Supporting actor)
  - The stakeholder that responds to a request from the use case.
  - e.g. the credit bureau authorizing a credit card charge
- **External receiver actor (off stage actor)**
  - The stakeholder that is not the primary or supporting actor but receives something of value from the use case.
  - e.g. the govt tax agency, * marketing

## Offstage actors

- An actors who has interest in the behavior of the use case but its not supporting or primary. For e.g. govt tax agency.
- Identifying them ensures that all necessary interests are identified and satisfied.
- Offstage actor interests are easy to miss unless these actors are explicitly named.

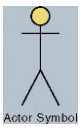
## Components of Use Case Model

- Use Case
  - Define the functionality that is handled by the system.
  - Each use case specifies a complete functionality (from its initiation by an actor until it has performed the requested functionality).
  - Describes the interactions between various actors and the system.
- Notation 

## Basic Use-Case Symbols

- **Use case** – subset of the overall system functionality
  - Represented graphically by a horizontal ellipse with the name of the use case appearing above, below, or inside the ellipse.


- **Actor** – anything that needs to interact with the system to exchange information.
  - Could be a human, an organization, another information system, an external device, or even time.
- **Temporal event** – a system event triggered by time.
  - The actor is time.




## Use case Diagrams & Scenarios

- describe what a system does from the standpoint of an external observer. The emphasis is on *what* a system does rather than *how*.
- Use case diagrams are closely connected to scenarios. A **scenario** is an example of what happens when someone interacts with the system.

Here is a scenario for a medical clinic.

- A *patient calls the clinic to make an appointment for a yearly checkup. The receptionist finds the nearest empty time slot in the appointment book and schedules the appointment for that time slot. "*
- Remember: A **use case** is a summary of for a single task or goal.

## Use Case: Example Scenario

- The picture below is a **Make Appointment** use case for the medical clinic.
- The actor is a **Patient**. The connection between actor and use case is a **communication association** (or **communication** for short).
- Actors are stick figures. Use cases are ovals. Communications are lines that link actors to use cases.

## Use Case Diagram – Guidelines & Caution

1. Use cases should ideally begin with a verb – i.e generate report. Use cases should NOT be open ended – i.e Register (instead should be named as Register New User)
2. Avoid showing communication between actors
3. Actors should be named as singular. i.e student and NOT students. NO names should be used – i.e John, Sam, etc.
4. Do NOT show behaviour in a use case diagram; instead only depict only system functionality.
5. Use case diagram does not show sequence – unlike DFDs

## Use Case - Relationships and its Types

- Relationships
  - Represent communication between actor and use case
- 4 types of relationships
  - Association relationship
  - **Generalization** relationship
    - Generalization relationship between actors
    - Generalization relationship between use cases
  - **Include** relationship between use cases
  - **Extend** relationship between use cases

## Use Case - Relationships and its Types

- Association relationship: Represent communication between actor and use case
- Often referred to as a communicate association
- use just a line to represent
- Notation

## Use Case - Relationships and its Types

- Generalization:
- The child use case inherits the behaviour and meaning of the parent use case.
- The child may add to or override the behaviour of its parent.
- Not

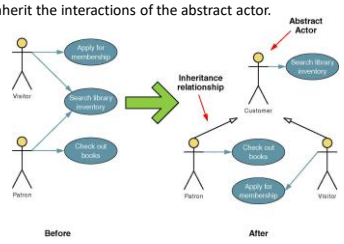

## Use Case - Relationships and its Types

- Generalization relationship between actors
  - actor generalization refers to the relationship which can exist between two actors and which shows that one actor (descendant) inherits the role and properties of another actor (ancestor).
- Generalization relationship between use cases
  - use case generalization refers to the relationship which can exist between two use cases and which shows that one use case (child) inherits the structure, behavior, and relationships of another use case (parent).

## Use Case Inheritance Relationship

**Inheritance** – a use case relationship in which the common behavior of two actors initiating the same use case is extrapolated and assigned to a new *abstract* actor to reduce redundancy.

- Other actors can inherit the interactions of the abstract actor.
- Depicted as an arrowheaded line beginning at one actor and pointing to the abstract actor whose interactions the first actor inherits.

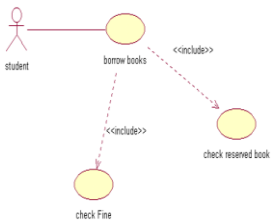


## Use Case - Relationships and its Types

- **Include**
  - Specifies that the source use case explicitly incorporates the behavior of another use case at a location specified by the source
  - The include relationship adds additional functionality not specified in the base use case.
  - <<include>> is used to include common behaviour from an included use case into a base use case
  - Notation


## <<include>>

- An include relationship connects a base use case (i.e. borrow books) to an inclusion use case (i.e. check Fine).
- An include relationship specifies how behaviour in the inclusion use case is used by the base use case.

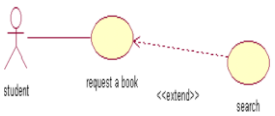



## Use Case - Relationships and its Types

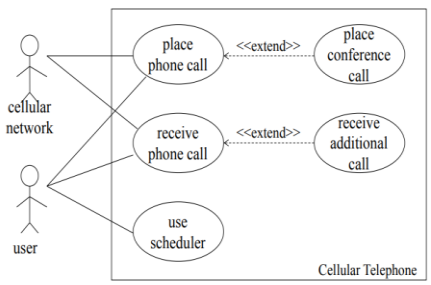
- **Extend**
  - Specifies that the target use case extends the behavior of the source.
  - The extend relationships shows optional functionality or system behaviour.
  - <<extend>> is used to include optional behaviour from an extending use case in an extended use case.
  - Notation <<extend>>

## <<extend>>

- The extend relationship is in between Request a book and Search.
- If the student desires, he/she can search the book through the system.
- However, the student may only Request a book through the system without searching the book if the student knows the call number.



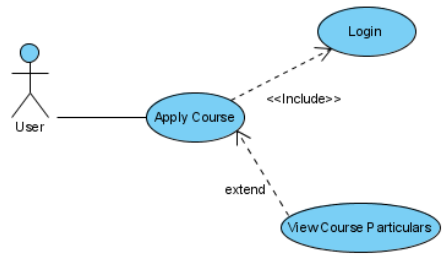

## Cellular telephone




## Include versus Extends

- Include Relationship
  - Represents the inclusion of the functionality of one use case within another
  - Arrow is drawn from the base use case to the used use case
  - Write << include >> above arrowhead line
- Extend relationship
  - Represents the extension of the use case to include optional functionality
  - Arrow is drawn from the extension use case to the base use case
  - Write << extend >> above arrowhead line

## Example – Include and Extend

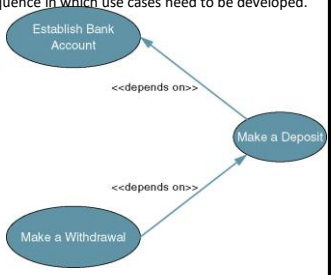



The use of include and extend is discouraged simply because they add unnecessary complexity to a Use Case diagram.

## Use Case Depends On Relationship

**Depends On** – a use case relationship that specifies which other use cases must be performed before the current use case.

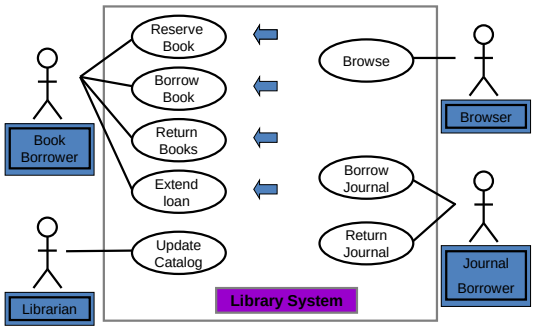
- Can help determine sequence in which use cases need to be developed.
- Depicted as an arrowheaded line beginning at one use case and pointing to a use case it is dependent on.
- Each depends on relationship line is labeled "<<depends on>>."




## Use Case - Boundary

- Boundary
  - A boundary rectangle is placed around the perimeter of the system to show how the actors communicate with the system.
  - A system boundary of a use case diagram defines the limits of the system.

## Use Diagram for a Library System




## Use Cases

- Here is a scenario for a purchasing items.
- "Customer arrives at checkout with items to purchase in cash. Cashier records the items and takes cash payment. On completion, customer leaves with items. "
- We want to write a use case for this scenario.
- Remember: A **use case** is a summary of scenarios for a single task or goal.

## Steps involved in creating use cases

1. Identify the actors
2. Identify the use cases
3. Draw the system boundary
4. Place the use cases on the diagram
5. Add relationships

## Identifying Actors

- As we read the scenario, define those people or systems that are going to interact with the scenario.
- Customer arrives at checkout with items to purchase in cash. Cashier records the items and takes cash payment. On completion, customer leaves with items.

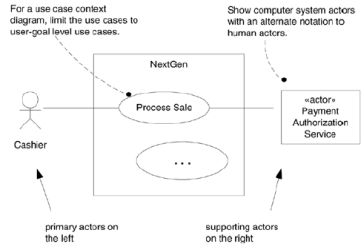
## Questions for Identifying People Actors

- Who is interested in the scenario/system?
- Where in the organization is the scenario/system be used?
- Who will benefit from the use of the scenario/system?
- Who will supply the scenario/system with this information, use this information, and remove this information?
- Does one person play several different roles?
- Do several people play the same role?

## Questions for Identifying Other Actors

- What other entity is interested in the scenario/system?
- What other entity will supply the scenario/system with this information, use this information, and remove this information?
- Does the system use an external resource?
- Does the system interact with a legacy system?

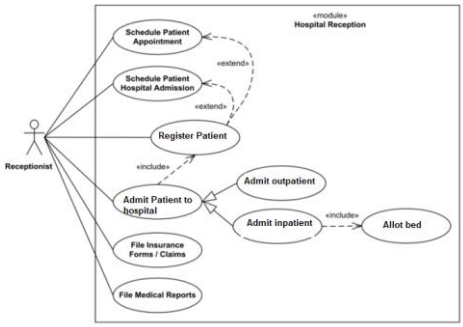
## Notation Suggestions




## Examples

## Exercise

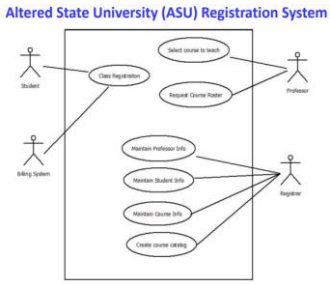
- Draw a use case diagram for the hospital reception system. In this system, receptionist can schedule patient appointment and patient hospital admission after the patient registration. Admitting patient requires registration. Both types of patients i.e. outpatient and inpatient can be admitted in the hospital. Inpatient patients are allotted bed. Receptionist also checks the insurance and claim forms and put them in file. Patient medical report is also filed by the receptionist.




## Exercise: Your turn

Altered State University (ASU) Registration System

1. Professors indicate which courses they will teach on-line.
2. A course catalog can be printed
3. Allow students to select on-line four courses for upcoming semester.
4. No course may have more than 10 students or less than 3 students.
5. When the registration is completed, the system sends information to the billing system.
6. Professors can obtain course rosters on-line.
7. Students can add or drop classes on-line.
8. The information about the course registration is sent to billing system.
9. The information about courses, professors, student and catalog is maintained by the registrar.




## READING

Chapter 6 – Applying UML and Pattern by Craig Larman 3rd Edition

# END OF TOPIC 5

-COMING UP!!!!!!
-Use case Narration
-Domain Models
-Class Diagrams

47