# 31269:
# Business Requirements Modelling

## Week 7 Lecture:
## Object Oriented Modelling - Use Case Modelling

- **References**
  - Object Oriented Systems Analysis and Design Using UML, 4th Edition by Farmer, Ray, McRobb- Chapters 6 and A2

# Where to from here?

- Last Half: Structured Analysis
  - What info do we need; where to get it from?
  - How do we get the information we need?
  - How do we analyse the information?

- This Half: Object Oriented Analysis
  - Use Case Modelling (***This Lecture***)
  - Class Modelling
  - Interaction Modelling
  - State and Event Modelling

# Objectives

- Appreciate how Object Oriented (OO) modelling techniques can help to understand the working of business systems
- Discover why system specifications are important and how OO modeling can be used to specify systems and user requirements
- Use object oriented system analysis techniques to develop a system model (**Use Case Model**)

# Topics

- Structured Analysis
- Object Oriented Analysis
- Unified Modelling Language (UML)
- Use Case Modelling and Narrative
- Use Case Application Example

# Structured Analysis Focus

Identifying and Modelling:

- **Processes** (BPMN)
  - Actions & transformations
- **Data** (ERD)
  - Consumed and produced by the system

# Object Oriented Analysis Focus

Identifying and Modelling:

- **Objects**
  - Represent real-world things

- Objects contain
  - **Methods** or Processes
  - **Attributes** or Data

# Different Application Models

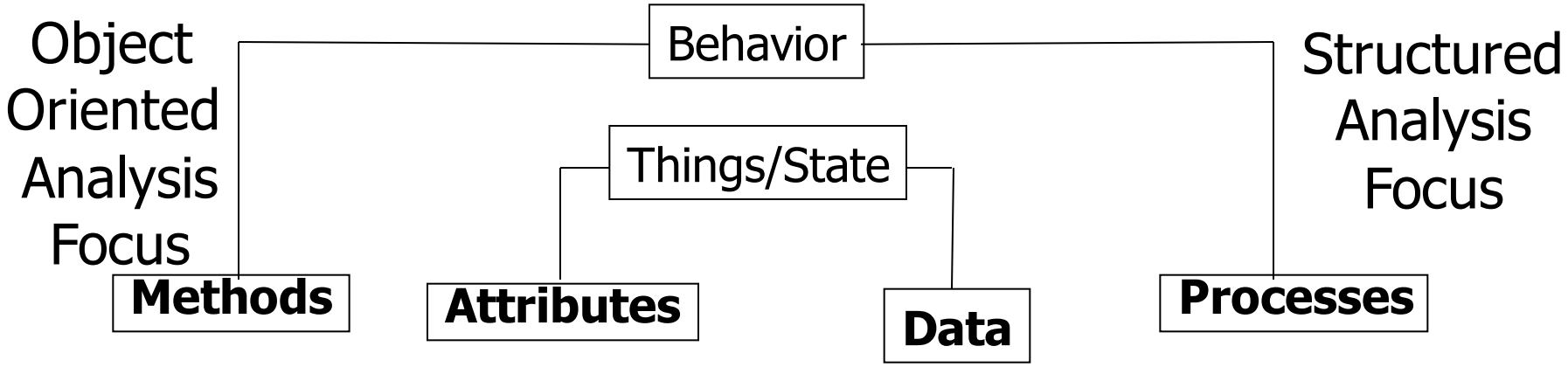
**Structured Approach**

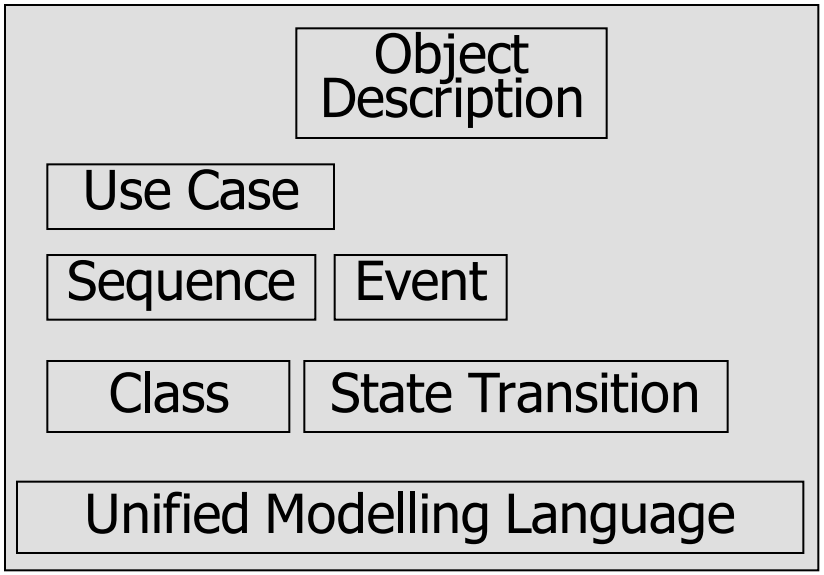
- Application is a **collection of processes** organised into a system
- Processes interact via flows of data
- Processes accept inputs and produce outputs

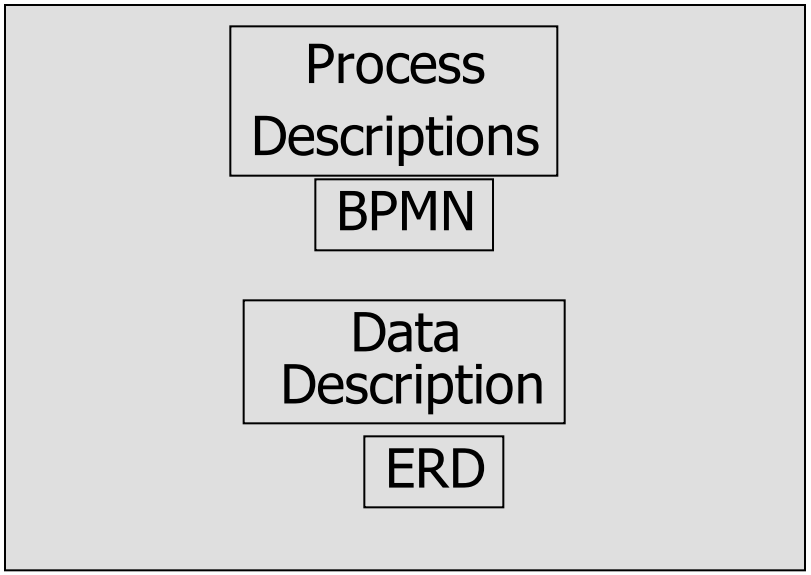
**O-O Approach**

- Application is a **collection of interacting objects**
- Objects interact with people and each other
- Objects send and respond to messages/methods

# Comparison of Structured and OO Approach



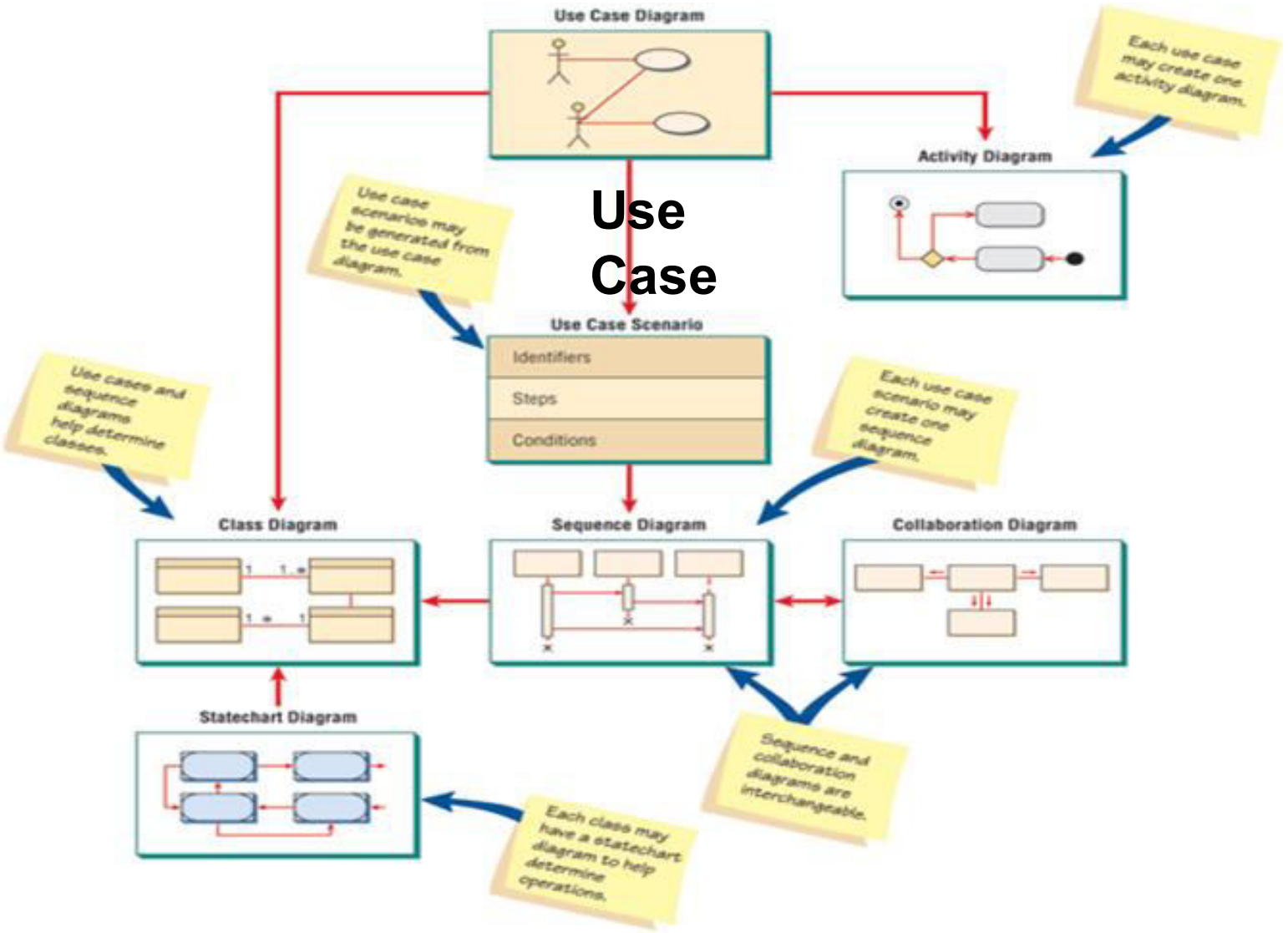



# Object Oriented Modeling Language

- **The Unified Modeling Language (UML)**
- UML is a standard language for specifying, visualizing, constructing, and documenting the artefacts of software systems.
- UML 2.X is an Industry standard developed to support Object-Oriented analysis and design
- UML is a pictorial language used to make software blue prints.
- UML diagrams are not only made for developers but also for business users, common people and anybody interested to understand the system.

# OO Modelling Using UML – Different models used in OO



# Object Oriented Modeling Tools

- Visual Paradigm
- IBM Rational Modeler

  (hhttp://www-1.ibm.com/software/awdtools/modeler/)
- MS Visio
- Sparx Enterprise Architect

# Use Case Model and Narratives: Development Steps involved

User Stories (last lecture)

Use Case Diagram

Use Case Scenarios or Narratives

User Interface or Wireframe or Screenshot

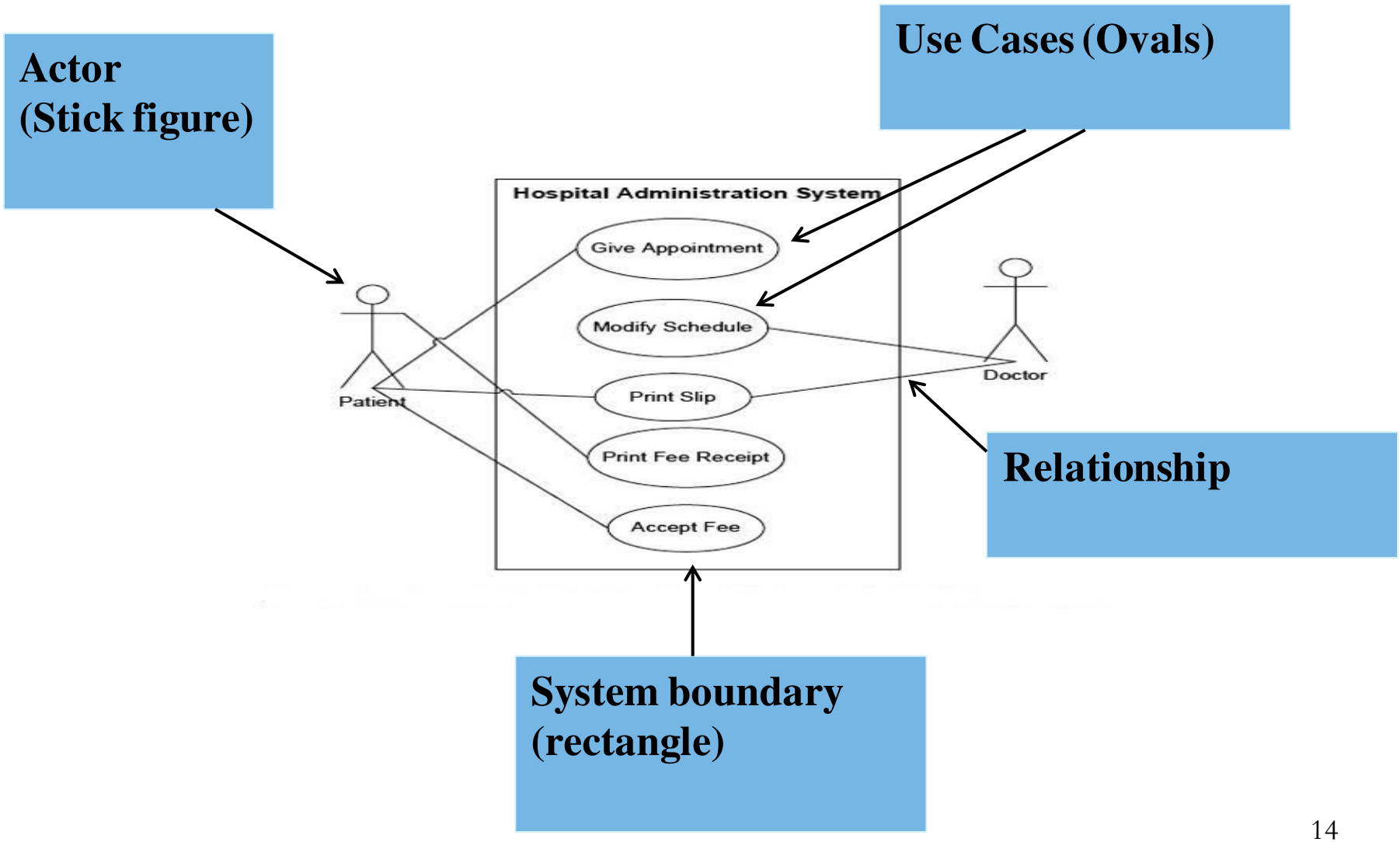
- ***A User Story can be detailed in terms of 1 or many Use Cases***

# Use Case Model/Diagram

A use case diagram contains four components:

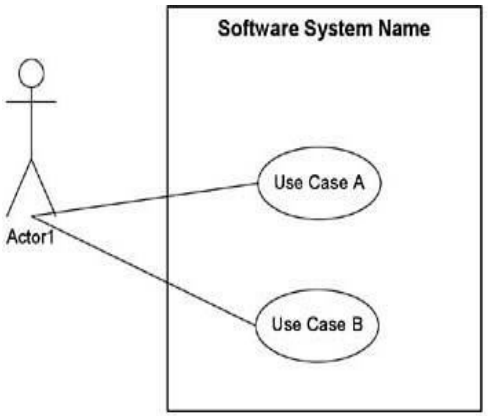
- ***System boundary***, which defines the system of interest (scope) in relation to the world around it.
- ***Actors***, could be a person, organization, or external system that plays a role in one or more interactions with your system.
- ***Use cases***, is a generalized description of a set of interactions between the system and one or more actors. The system **functionalities** are captured in use cases.
- ***Relationships*** between and among the actors and the use cases.

# Use Case Model: Example

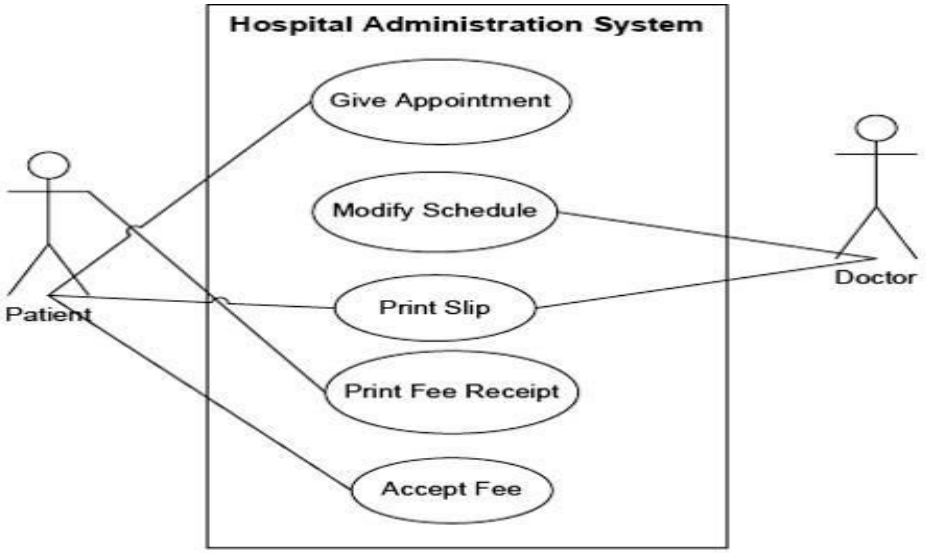


# System Boundary

- The rectangle around the use cases is called *the system boundary*, it indicates the scope of your system - the use cases inside the rectangle represent the functionality that you intend to implement.

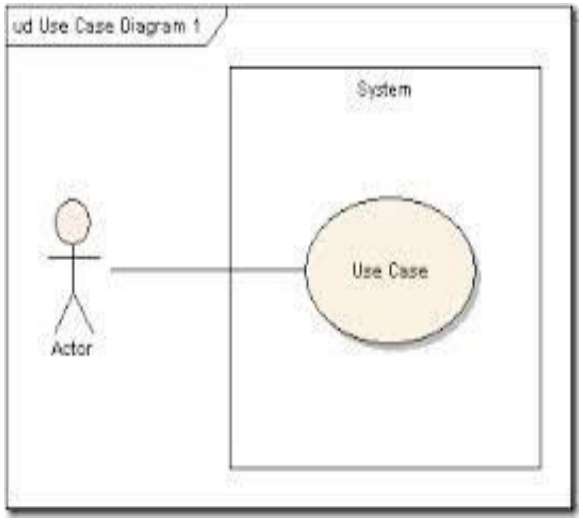


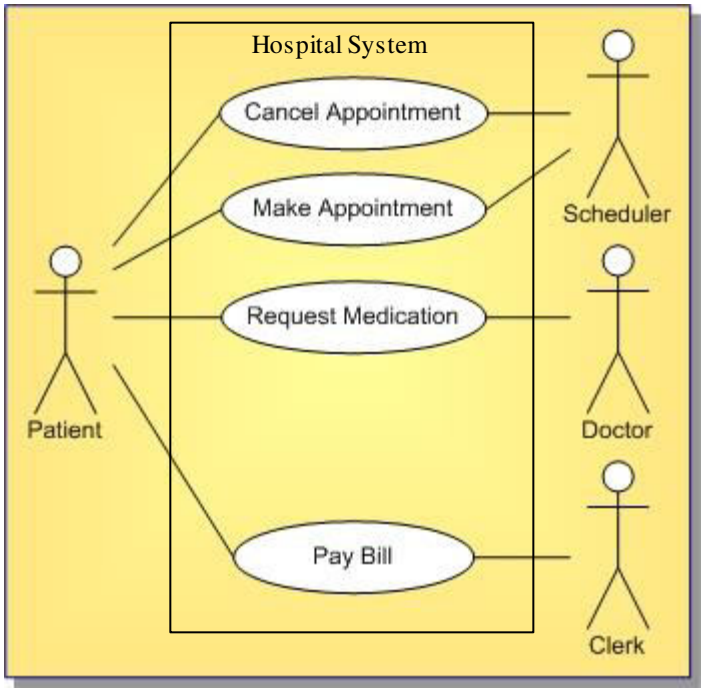

*The System Boundary*



# Actor

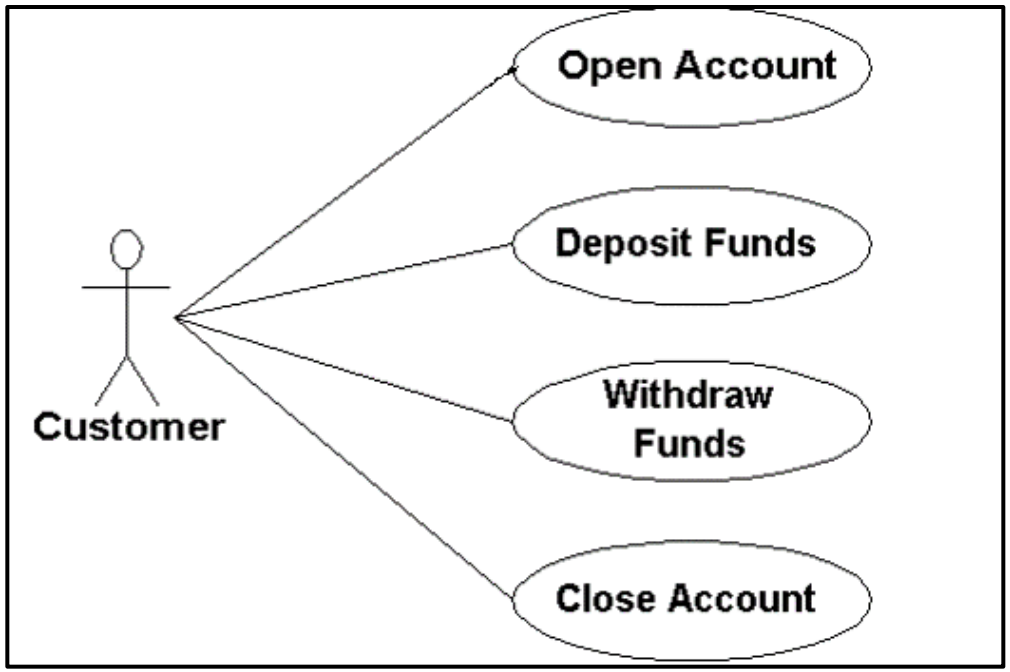
- An actor is a person, organization, or external system that plays a role in one or more interactions with your system



# Use Cases

- A use case is a generalized description of a **set of interactions** between the system and one or more actors. A use case describes ***a sequence of actions*** performed by a system for a specific goal. The system **functionalities** are captured in use cases.
- A use case is drawn as a horizontal ellipse/oval in a use case diagram. Use case name ***begins with a verb.***



# Relationship

- ***The relationships*** between and among the actors and the use cases. Links an Actor to a Use Case
- There are several types of relationships that may appear on a use case diagram:

  **An association between an actor and a use case**

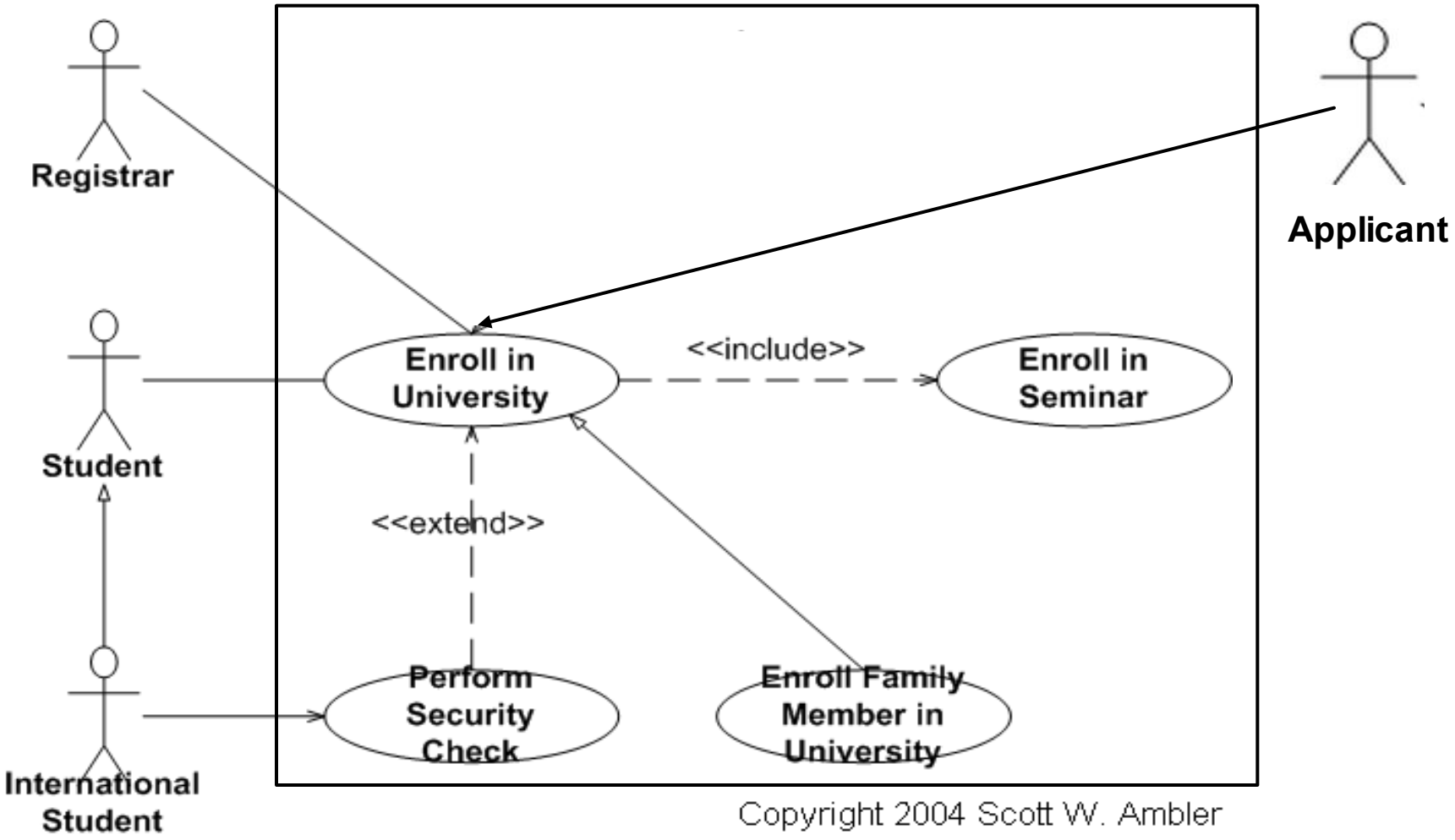
  **An association between two use cases (includes and extends)**

  **A generalization between two actors**

  **A generalization between two use cases**

- **Associations** are depicted as lines connecting two modeling elements with an optional open-headed arrowhead on one end of the line indicating the direction of the initial invocation of the relationship.
- **Generalizations** are depicted as a close-headed arrow with the arrow pointing towards the more general modeling element.

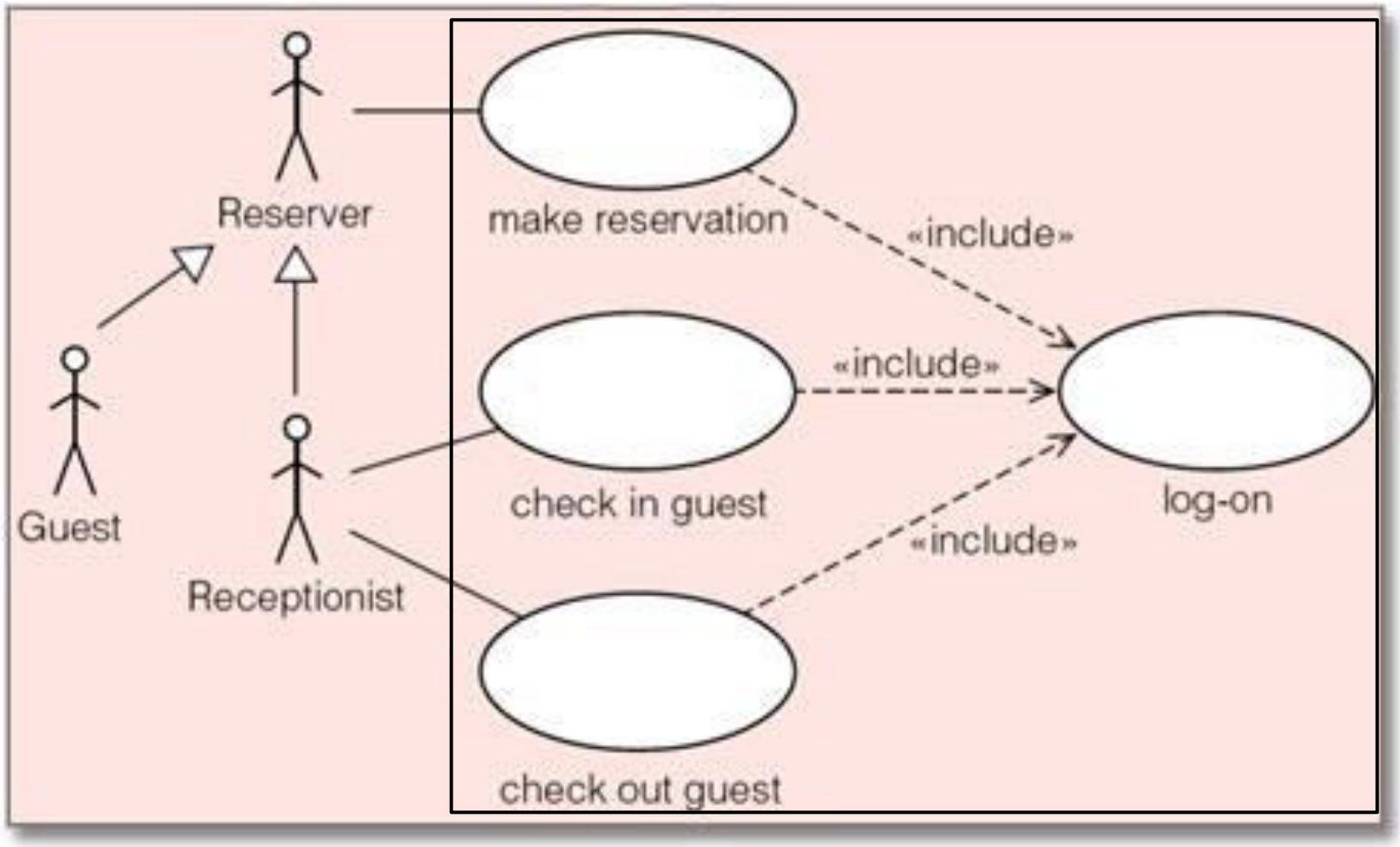
# Example of Enrolment System Use Case showing various relationships



# Included Use Case

- **'Includes' or 'Uses'**
  - Links a use case to another use case that describes common behaviour
  - A *base* use case *includes* an *included* use case if an action step in the base use case calls out the included use cases' name
  - The included use case describes a lower-level goal than the base use case
  - Each of the included, more detailed use cases is a step that the actor or actors might have to perform to achieve the overall goal of the including use case. The arrow should point at the more detailed, included use case.

# Example of ‘include’ use case

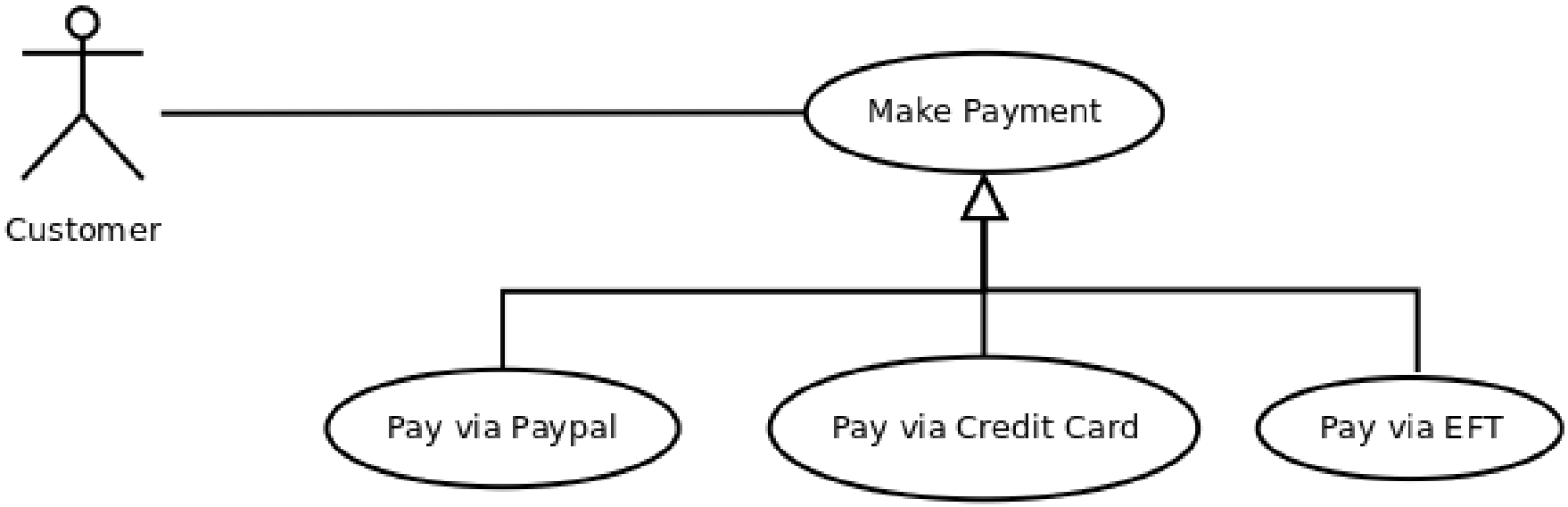



http://vcampus.uom.ac.mu/systemmodelling/concept%20of%20modelling/6.10.extendinclude.htm

# Generalised Use Case

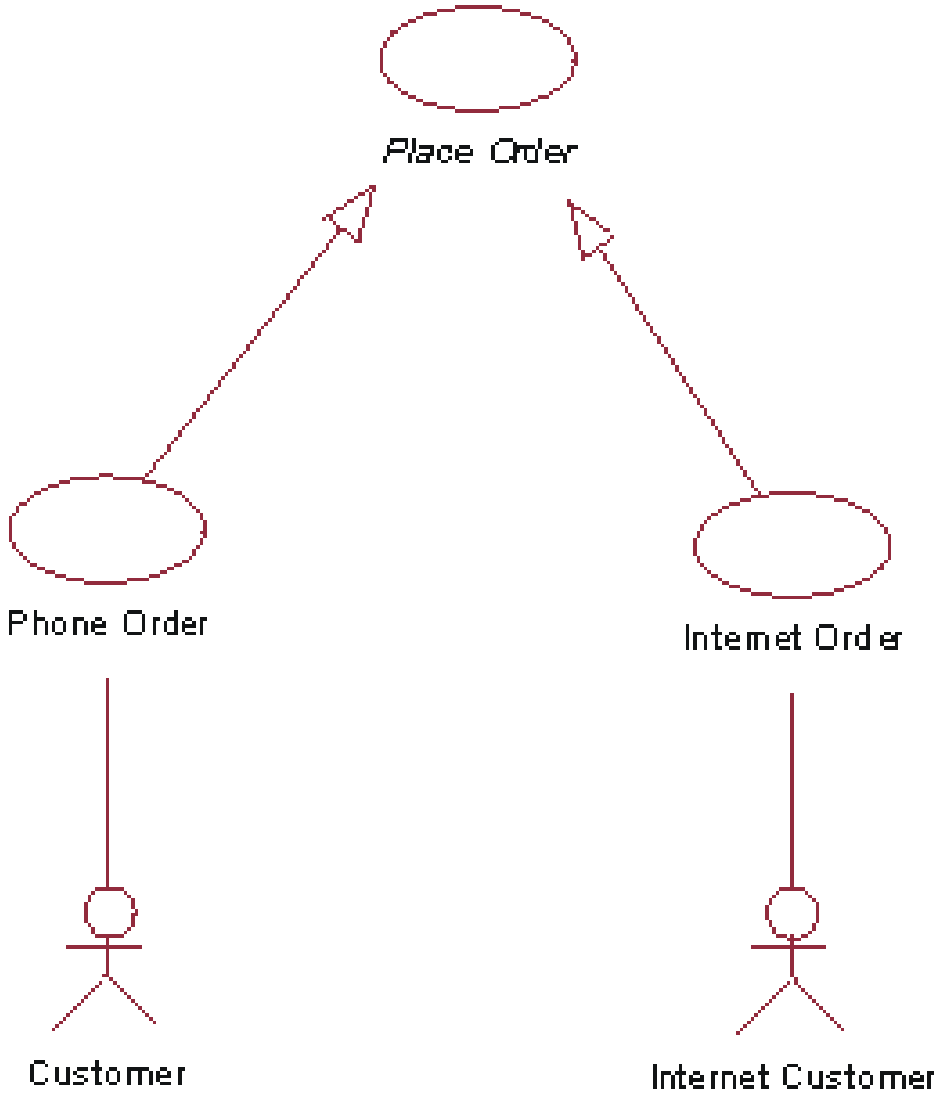
- **‘Generalise’ or ‘Inherits’**
  - The general use case generalises the specific one
  - Defines a link between a general and more specific use case
  - The (specialising) child should be of a “similar species” to that of the (general) parent.
  - A generalisation relationship between use case implies that the child use case contains all the attributes, sequences of behaviour, and extension points defined in the parent use case, and participates in all the relationships of the parent use case.

# Example1 of Generalise or Inherit




http://stackoverflow.com/questions/15133595/use-case-generalization-versus-extension

# Example 2 of Generalise or Inherit

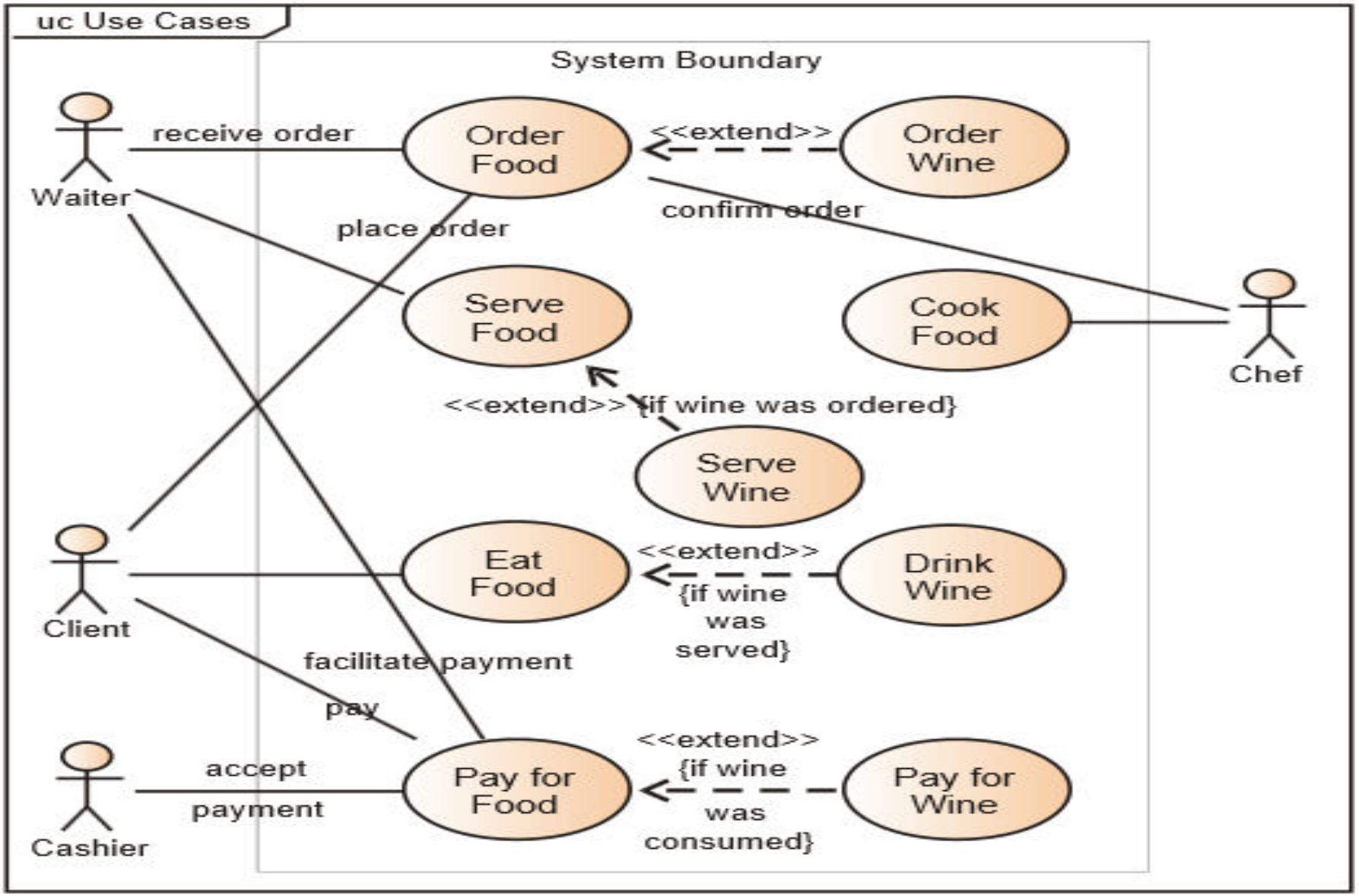



http://students.mimuw.edu.pl/~zbyszek/posi/ibm/RUP_Eval/process/modguide/md_ucgen.htm

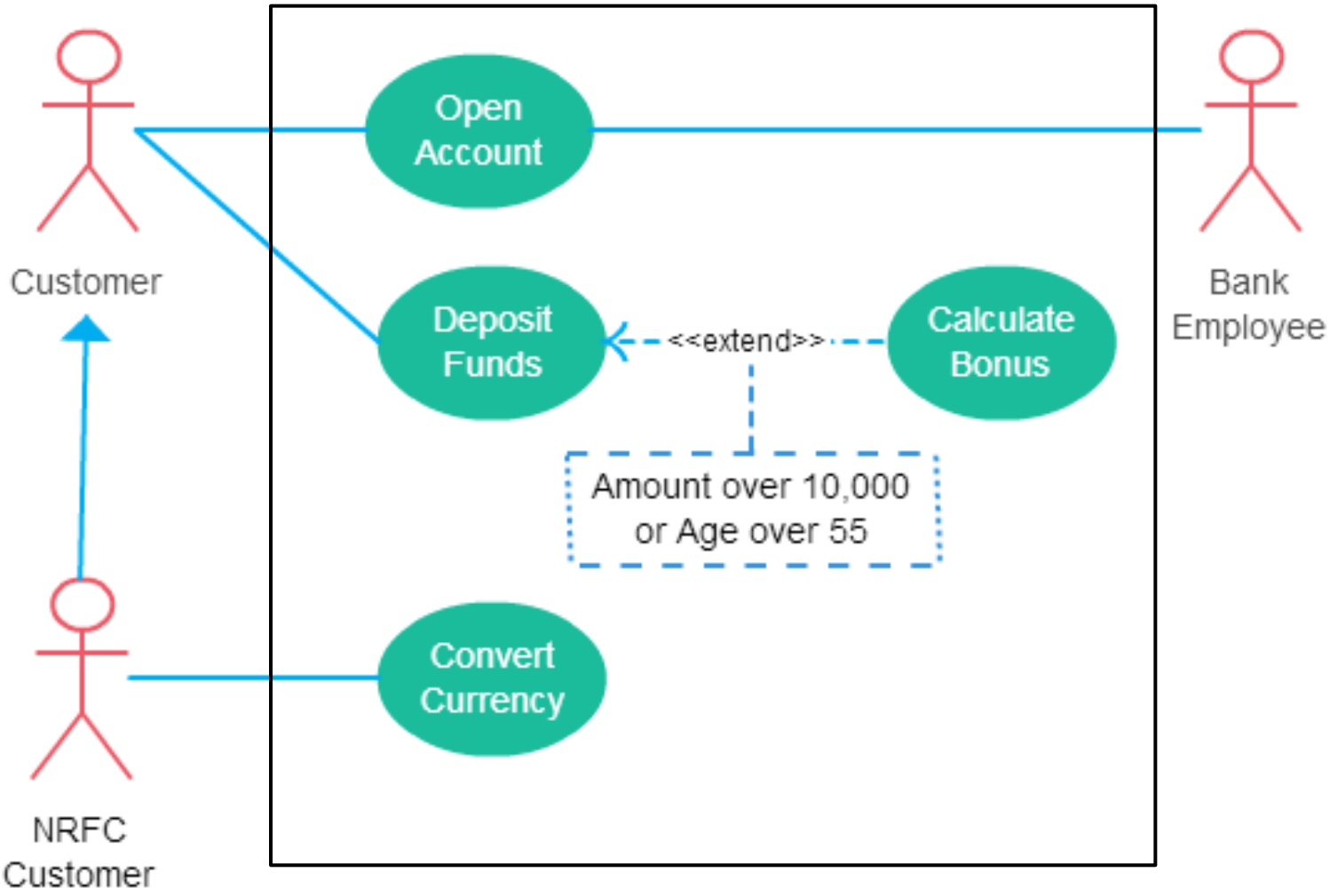
# Extension use case

- **'Extends'**
  - An ***extending*** or ***extension*** use case ***extends*** a base use case by naming the base use case and defining the circumstances under which it **interrupts the base use case**
  - The extending use case specifies some internal condition in the base use case and a triggering condition. Behavior runs through the base use case until the condition occurs, at which point it continues in the extending use case. When the extending use case finishes, the behavior picks up in the base use case where it left off.
  - Links a use case to another use case describing a **variation** from standard behavior
  - Use when the system takes a **different behavior**
  - The Extensions section in use case narrative defines **alternative paths** through the use case. Often, extensions are used for error handling; but extensions are also used to describe successful but **secondary paths**

# Example of extends use case



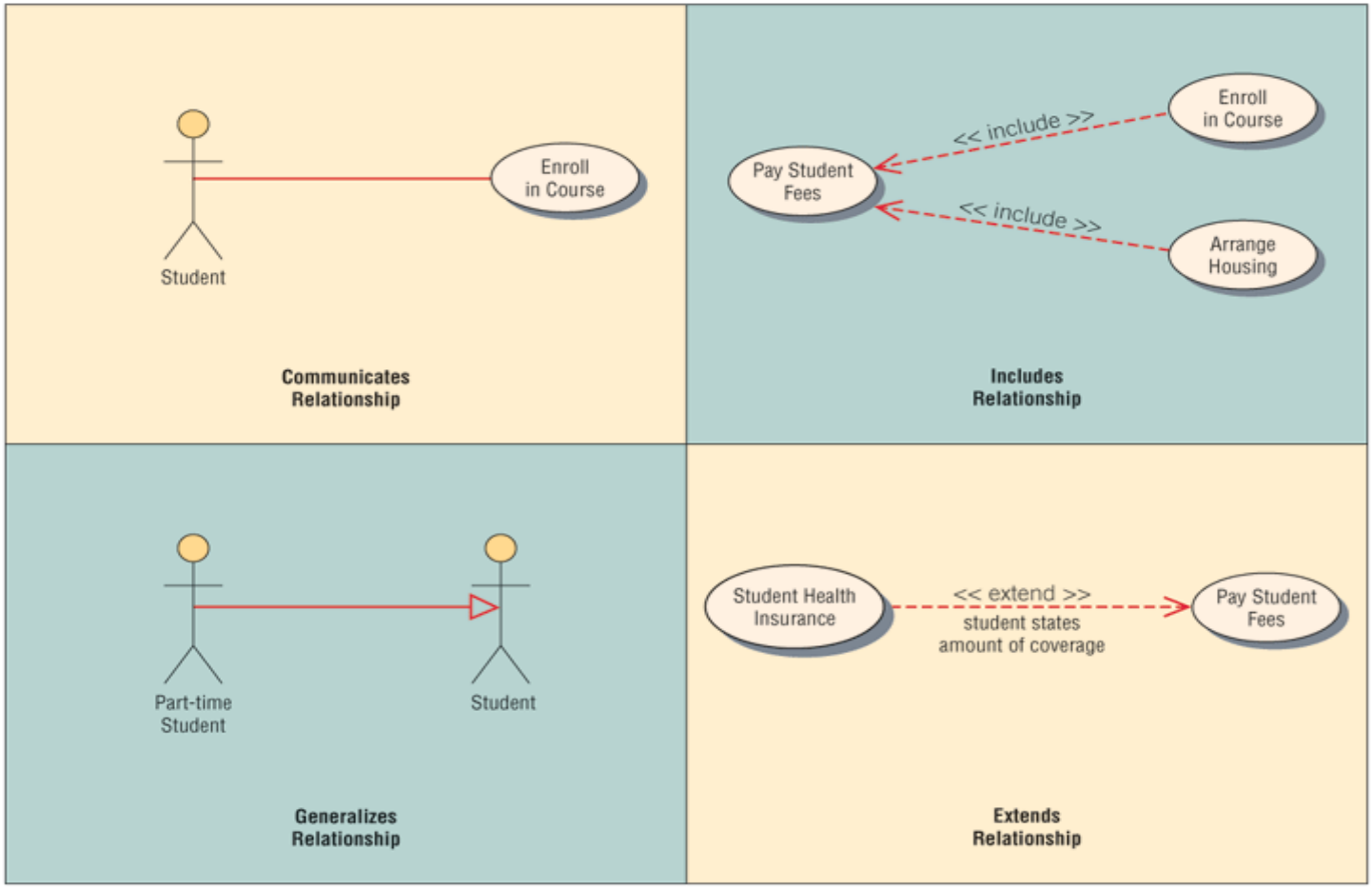
# Another example of extends Use Case




http://creately.com/blog/diagrams/use-case-diagram-relationships/

# Use Case Diagram: Relationships

**Figure 18.7**
Four types of UML behavioral relationships and the arrows and lines used to represent the relationships.



# Putting it all together – Steps in development

User Stories (last week)

Use Case Diagram

Use Case Scenarios or Narrative

User Interface or Wireframe or Screenshot

- ***A User Story can be detailed in terms of 1 or many Use Cases***

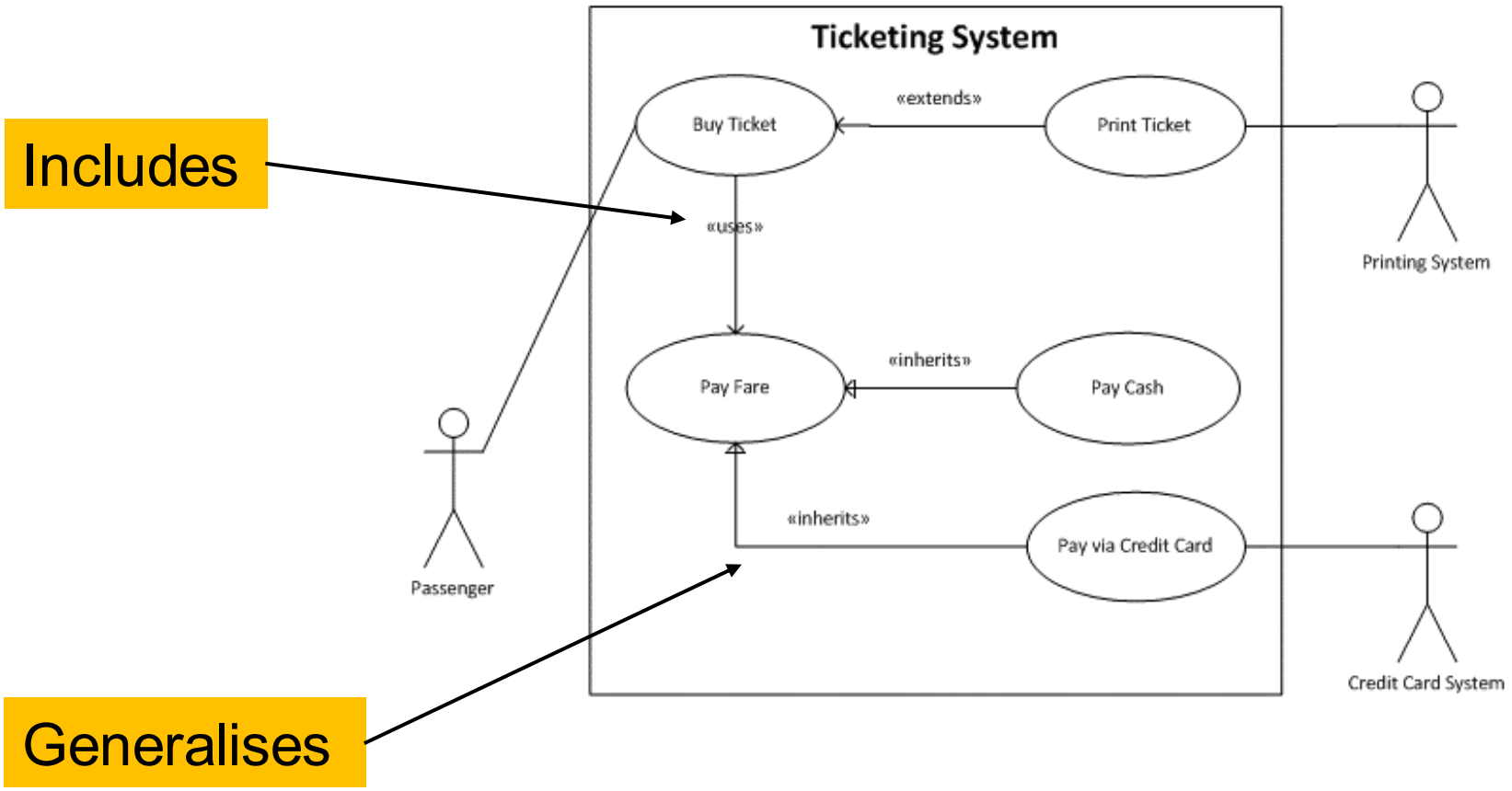
# User Stories

- User Story
  - ***As a*** *passenger,* ***I want*** *to* buy a ticket via Online Ticketing System **so** ***that*** *I can travel from one city to another city in Australia.*

- *Remember -> A User Story can be detailed in terms of 1 or many Use Cases. Or many user stories can be detailed in one use case. The relationship depends on a number of factors.*

- *Based on this User Story you have further conversations with the user or proxy user and identify the additional details.*

# Use Case Diagram



# Use Case Narratives/Scenarios

- Use Case Narrative Template (refer to a separate document available on uts online and understand each section).
  **MUST READ EACH SECTION.**

- Detail “Buy Ticket” Scenario or Narrative. Refer to a separate document on uts online to see the narrative for Buy Ticket use case.

# Use Case Narrative Template

## Use Case Template

**Use Case:** The name of your use case, keep it short and sweet

| | |
|---|---|
| **Use Case ID** | ID to represent your use case |
| **User Story** | User Story related to this use case |
| **Goal** | |
| **Priority** | High, Medium or Low |
| **Actors** | Primary and Secondary actors |
| **Pre-conditions** | A list of conditions, if any, that must be met before this use case can start. |
| **Post-conditions** | A list of conditions, if any, that that will be true after the use case finishes successfully. |
| **Trigger** | The event that triggers this use case |
| **Main Flow** | Detailed and step by step **description of user actions and system responses.** Also termed as ***main success scenario*** refers to the primary successful path. The main/basic flow should be the events of the use case when everything is perfect; there are no errors, no exceptions. This is the "happy day scenario". The secondary/alternative successful paths achieving the same goal of this use will be handled in the "Alternate Flows" section. |
| **Exceptions** | It is used to specify error conditions, a **list of things that could go wrong** at any step in the main flow are captured here. |

| | |
|---|---|
| **Alternate Flow 1** | The variation of main flow that still achieves the goal of the main/base use case. The paths that are the result of an alternate way to work. Alternate flows are used to describe **successful but secondary paths** to achieve the same goal of the base use case. Should re-join the steps in the main flow of the base use case. |
| **Trigger** | What event triggers this use case |
| **Step** | Specify steps that in the alternate flow. Last step in this alternate flow should re-join some step in the main flow and continue from there. |
| **Post conditions** | |
| **Exceptions** | Things that could go wrong at any step in the Alternate Flow are described here. |

| | |
|---|---|
| **Alternate Flow 2** | |
| **Trigger** | |
| **Step** | |
| **Post conditions** | |
| **Exceptions** | |

# Summary of Template

- Use Case Narrative Template (Must refer the template)
  - Header: Everything before main flow
  - Main Flow: **User actions and system responses**
  - Footer: Everything after main flow
  - Alternatives: Alternate path to successful outcomes; successful but secondary paths.
  - Exceptions: Main flow fails. A list of things that could go wrong in the main flow, unsuccessful path.

# Use Case Narratives/Scenarios for “Buy Ticket”

- Now that we have seen the Use Case Narrative Template, let’s detail “Buy Ticket” Scenario or Narrative.
- Refer to a separate document on uts online to see the narrative for Buy Ticket use case.

# Difference between User Stories and Use Cases (homework)

This slide to be read at home by students

- With so many engineering teams making the paradigm shift from waterfall to Agile Software Development, people often get caught up in having a pure Agile process which would include the use of User Stories. So what's all of the hoopla with User Stories? What are they, how are they different from use cases, do I need them, and where do they fit in the process?
  - **What is a User Story?** Simply put, written from the context of the user as a simple statement about their feature need.
  - **How is a User Story different than a Use Case?** While a use case is highly structured and tells a story, the User Story sets the stage by stating the need. A User Story is the **prelude** to the use case by **stating the need before the use case describes the need in detail**.
  - **How does the User Story fit into the process?** User Stories are great as an activity in collecting and prioritizing the high level features. Getting this initial feedback from the customer is a simple way of trying to get all of their needs identified and prioritized. **The User Stories will then morph themselves into the business requirements and use cases.**
- Can I use Use Cases in agile development? Yes, but keep the use cases lean with less features so that they can be iterated and adapted with each release.
  *http://www.gatherspace.com/static/use_case_example.html#12*

# Difference between User Stories and Use Cases (homework)

This slide to be read at home by students

- **Purpose:** The purpose of the use case is to document an agreement between the customer and the development team. User stories, on the other hand, are written to facilitate release and iteration planning, and to serve as placeholders for conversations about the users' detailed needs.
- **Scope:** Both are sized to deliver business value, but stories are kept smaller in scope because we place constraints on their size (such as "no story can be expected to take more than 10 days of development work") so that they can be used in scheduling work. A use case almost always covers a much larger scope than a story.
- **Completeness**: The text on a story card plus acceptance tests "are basically the same thing as a use case." This means that the story corresponds to the use case's main success scenario, and that the story's tests correspond to the extensions of the use case.
- **Longevity:** Use cases are often permanent artifacts that continue to exist as long as the product is under active development or maintenance. User stories, on the other hand, are not intended to outlive the iteration in which they're added to the software. While it's possible to archive story cards, many teams simply rip them up.
  *http://www.mountaingoatsoftware.com/articles/advantages-of-user-stories-for-requirements*

# Assignment 2 – Released Now

- Object Oriented Requirements Analysis and Specification Report – 18 Marks
- Same Case Study as Assignment 1: Customer Onboarding System (COS) for Epic Video
- Functional and non-functional requirements
  - Functional requirements using:
    - User Story Map
    - User Stories and Use Cases (narratives)
    - Sequence Diagram
  - Data Requirements using:
    - Class Diagram
    - State Transition Diagram
  - Non-functional requirements:
    - User Interface requirements using wireframes
    - Security requirements
    - Performance requirements

# Assignment 2 Template: Template adapted in this subject

# Conclusion

- This Week's Workshops
  - **Quiz 5 – Software Requirements Specification (3 marks)**
  - Tasks – Use Case Modelling

---

- Next Week's Lecture
  - Object Oriented Models with UML- Class Modelling
- Next Week's Workshop:
  - **Quiz 6 – Use Case Modelling (3 marks)**
  - Tasks – Class Modelling

**Reminder: Assignment 2 released this week**