Visual Modelling with UML

Unified Modelling Language

Aims

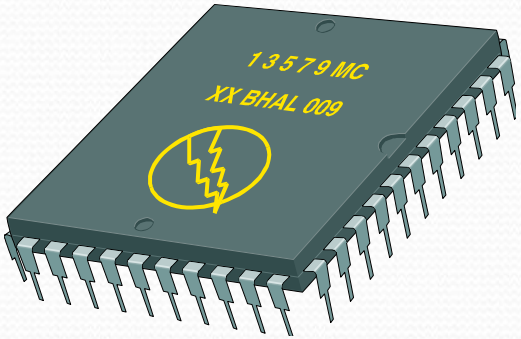
- Introduction to visual modelling using UML
- Modelling Requirements with Use Cases
- Capturing System Structure and Design with Class diagrams
- Capturing System Behaviour with Interaction diagrams

What is “Modelling”?

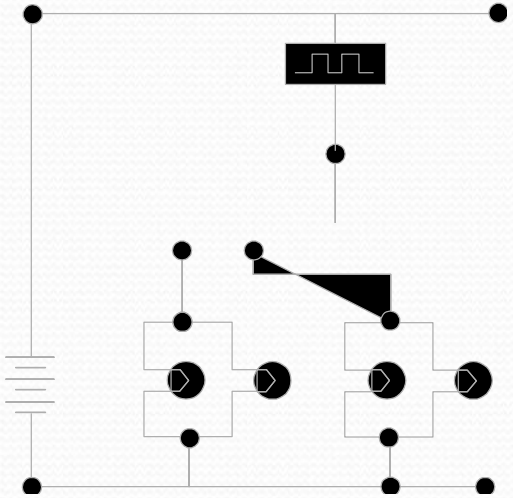
- Modelling
  - an abstraction to show the construction of something
  - a simplified representation of a phenomenon
  - an exhibition of relationships or connections between objects and actions
- Abstraction is performed
  - Lose some details so that an overview is produced

More than one model

- Who is viewing the model.
- Why they need to view it.



View for Customers

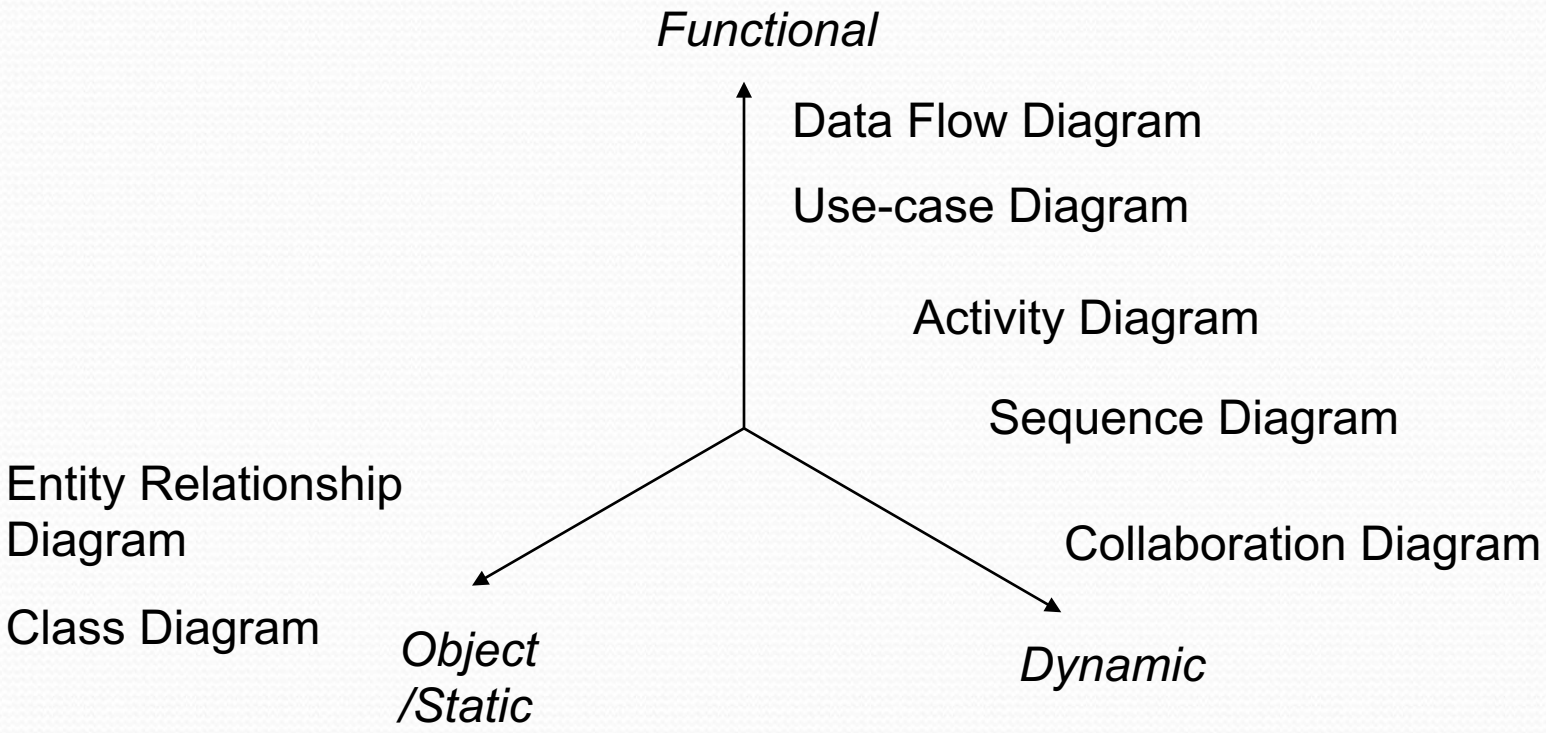


View for Designers

Why Model?

- Modeling achieves four aims:
  - Helps you to visualize a system as you want it to be.
  - Permits you to specify the structure and behavior of a system.
  - Gives you a template that guides you in constructing a system.
  - Documents the decisions you have made.
- You build models of complex systems because you cannot comprehend such a system in its entirety.
- You build models to better understand the system you are developing.

Views of Modeling



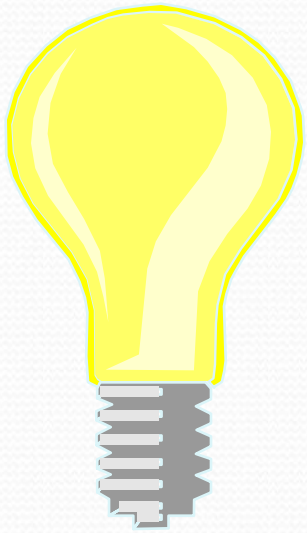
Three Viewpoints we can use

- A software system can be modeled from three viewpoints
  - **Object model** -- describes the static, structural, data aspects of the system
  - **Dynamic model** -- describes the temporal, behavioural, control aspects of the system.
  - **Functional model** -- describes the transformational, algorithmic aspects of the system.

- Note that the three models are not completely independent of each other

The UML Is a Language for Visualising

- Communicating conceptual models to others is prone to error unless everyone involved speaks the same language.
- There are things about a software system you can't understand unless you build models.
- An explicit model facilitates communication.

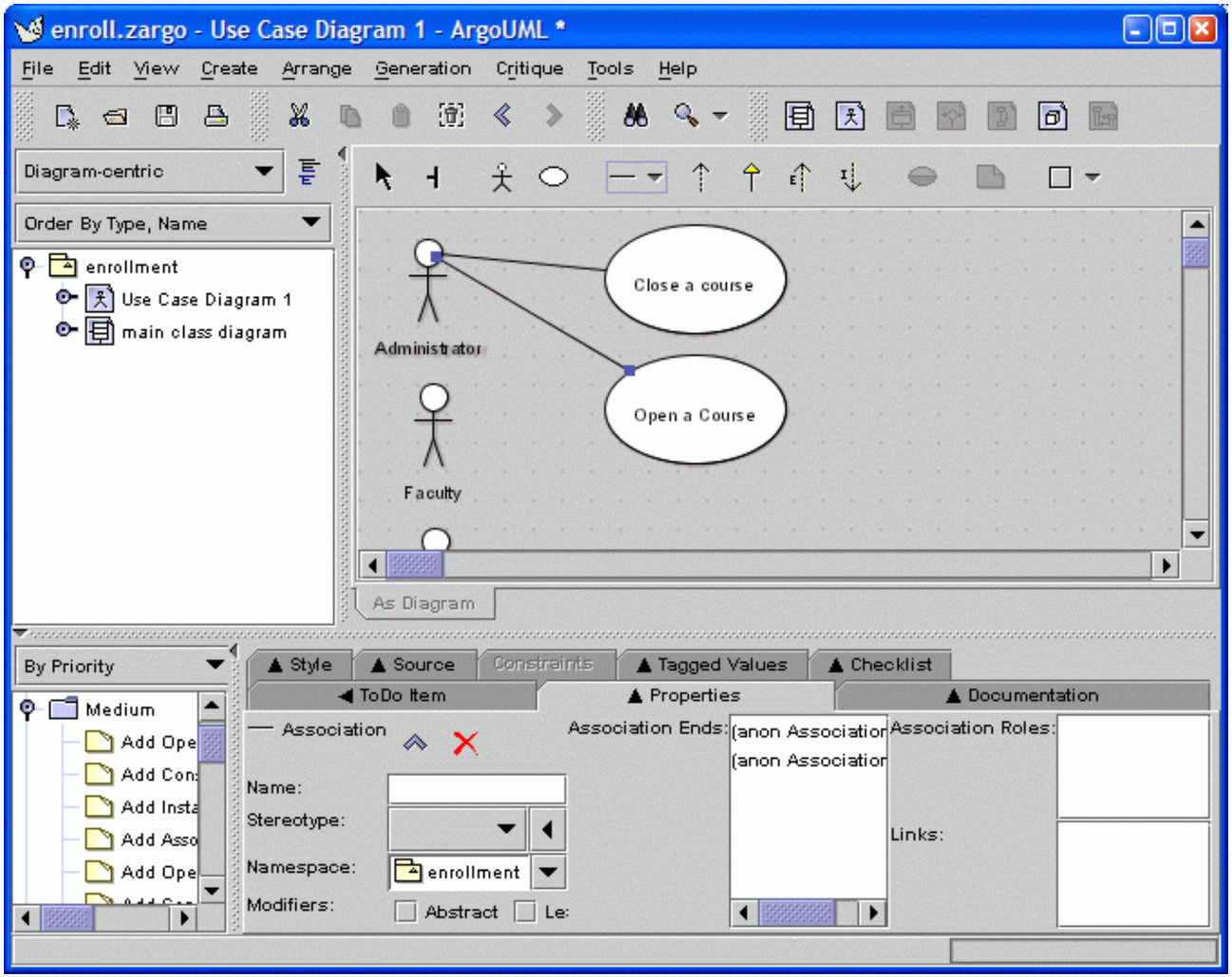


Today, UML has become the standard method for modelling software systems

Tools for drawing UML diagrams

- Several tools and templates for tools exist.
- AgroUML
- Visio
- Visual Paradigm

Note that we need to use UML 2 notations



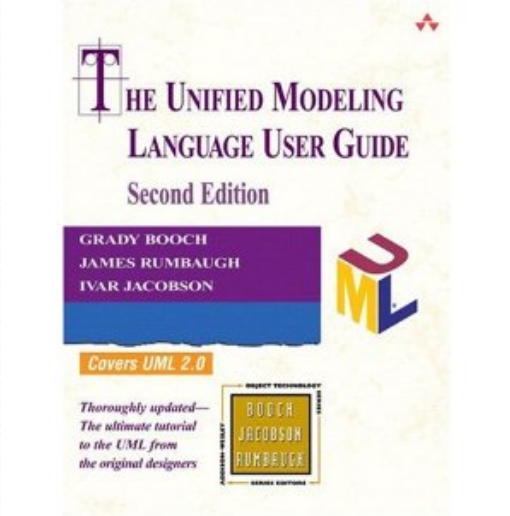
Remarks …

- There is no single correct model of a situation. A good model captures the crucial aspects of a problem and omits the others.
- The key objective is to achieve abstraction --- the selective examination of certain aspects of a problem.
- In this module, we will be looking at some basic techniques and apply them to small systems.

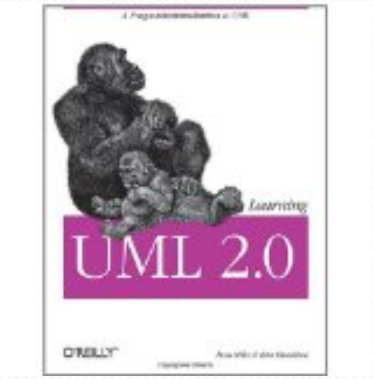
References

Many books in the library:

- The Unified Modelling Language User Guide, Addison, 2005
- Learning UML 2.0, O'Reilly, 2006 – electronic resource



Check the UML resource pages on the Object Management group site: www.omg.org



Note that we shall be using the

UML 2.0 notation

Use Case Diagrams

Introduction

After this lecture, you should be able to:

- Explain how use cases fit into a requirements management process and the software development lifecycle
- Define actor, use case, and use-case model
- List the benefits of use cases

The Use Case

- Use Cases define the behaviour performed by system
  - Shows how system behaves from **the users' point of view**
- Captures the users' functional requirements of a system
  - Use cases are developed according to users' needs
  - A useful technique for requirements gathering & specification
- Describes WHAT the system will do for the user - not HOW the system will do it
  - Should not include implementation details

Use cases involve a shift in thinking

From a focus on the function of a system

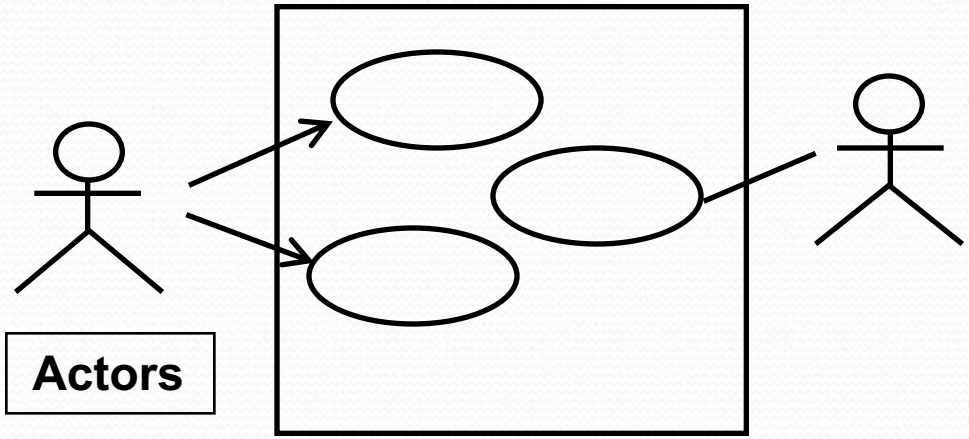
To a focus on the value a system must deliver for its stakeholders

What is a use-case model?

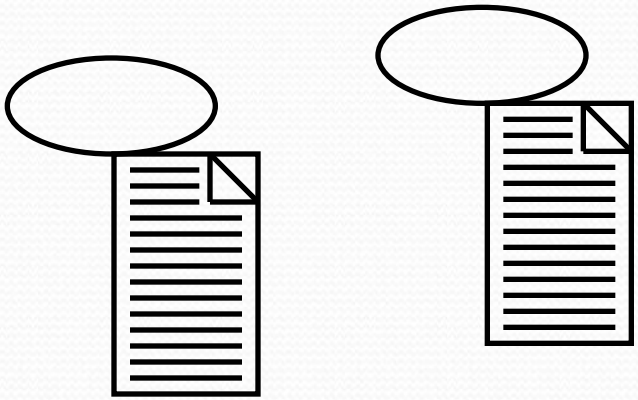
- Describes the functional requirements of a system in terms of use cases
- Links stakeholder needs to software requirements
- Serves as a planning tool
- Consists of **actors** and **use cases**

Relevant Requirements Artifacts

Use-Case Diagram Model

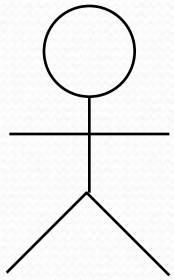


Use-Case Specifications



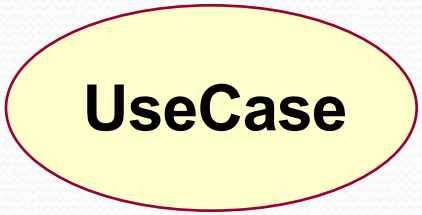
Major Concepts in Use-Case Modeling

- An actor represents anything that interacts with the system.



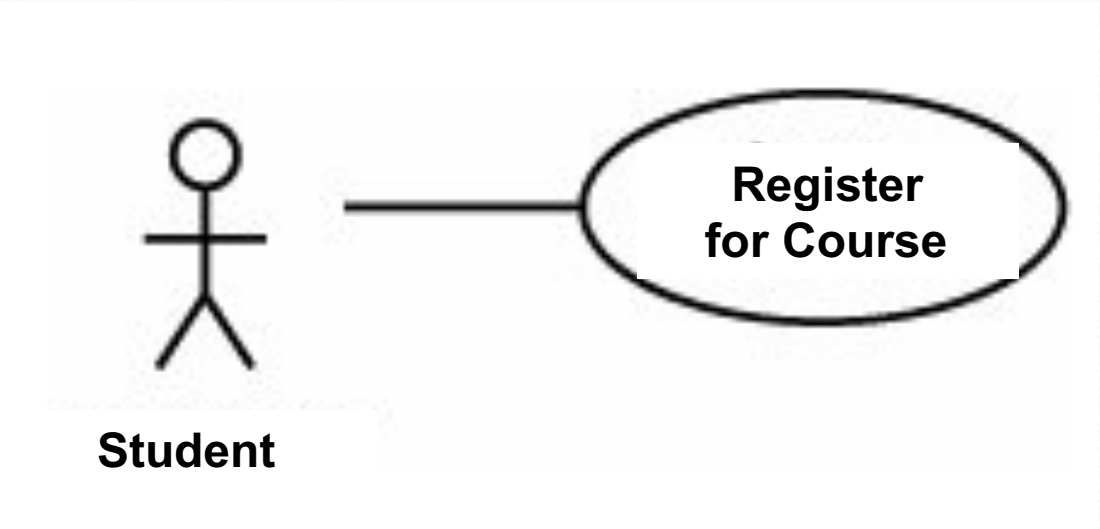
Actor

- A use case is a sequence of actions a system performs that yields an observable result of value to a particular actor.



Example - **Course Registration System**

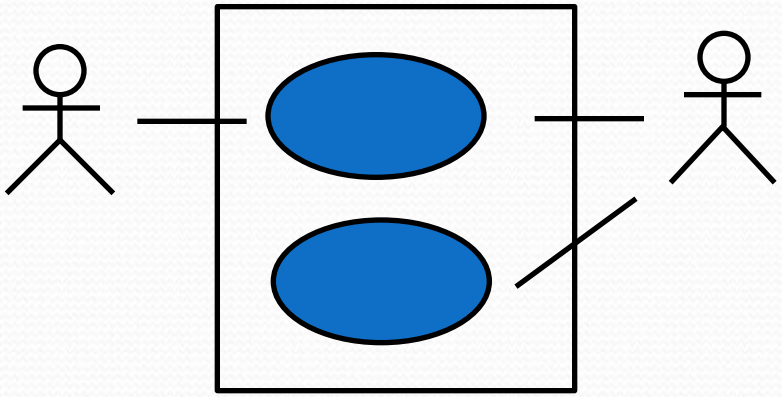
Requirement A.1: A student must be able to register for courses using the course registration system.



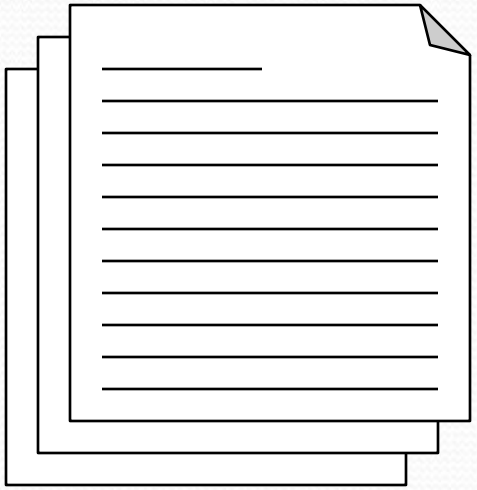
A use-case model is comprised of:

Capture a use-case model

Use-case diagrams
(visual representation)

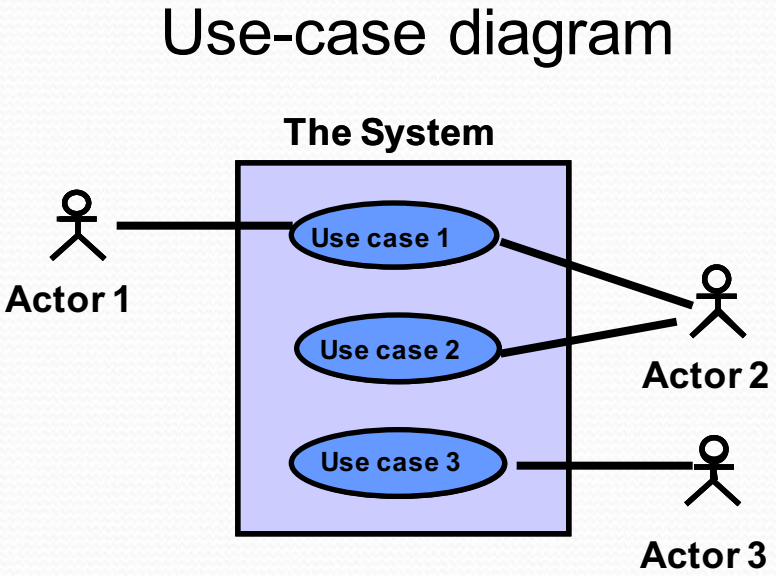


Use-case specifications
(text representation)



Use-case diagram

- Shows a set of use cases and actors and their relationships
- Defines clear boundaries of a system
- Identifies who or what interacts with the system
- Summarizes the behavior of the system



Process of writing use cases

Find actors

Student

Find use cases

Register for Course

Brief description: This use case allows a Student to register for course offerings in the current semester. A Schedule of core and option courses is produced.

Outline a use case

Register for Course Outline

-Flow of events

-Step by step

Detail a use case

Register for Course Use-Case Specification

-Detailed Flow of Events

-Special Requirements

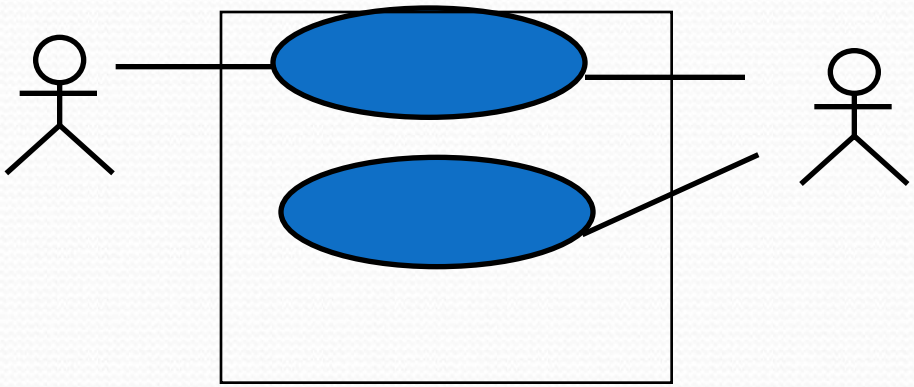
-Pre/Postconditions

Actor

- An actor represents a role that a human, hardware device, or another system can play in relation to the system
- An actor is external to the system
- A complete set of actors describes all of the ways in which outside users communicate with the system
- Actor names should clearly convey the actor's role
- Good actor names describe their responsibilities
- Avoid "User"

Use cases contain software requirements

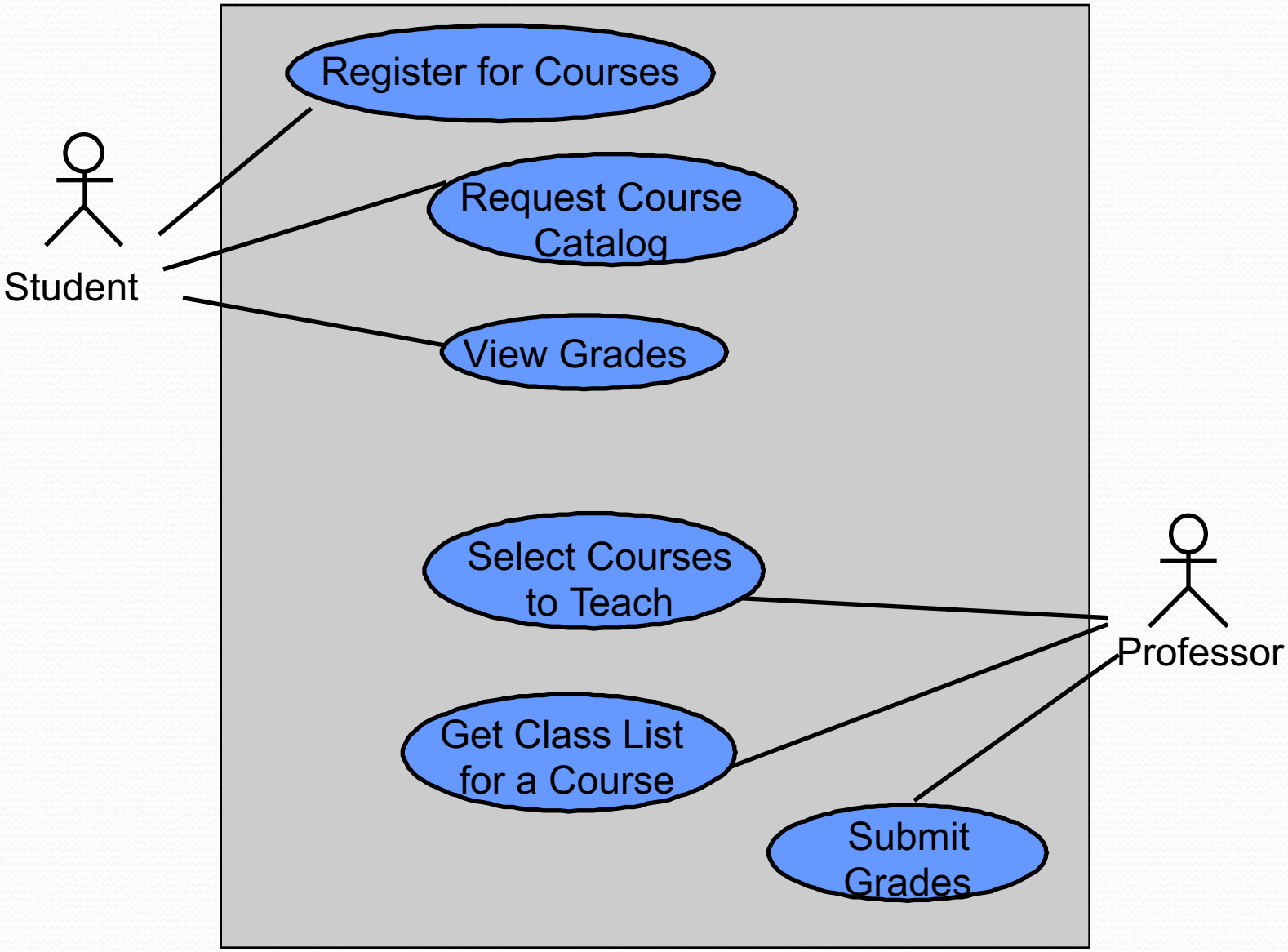
- Each use case
  - Models a dialog between the system and actors
  - Is a coherent unit of functionality performed by a system, which yields an observable result
  - Describes sequences of actions that the system takes to deliver something of value to an actor
  - Describes what the system should do, rather than how it should do it. the specification of a set of actions
  - Can be small or large, but always achieves a specific goal for the user



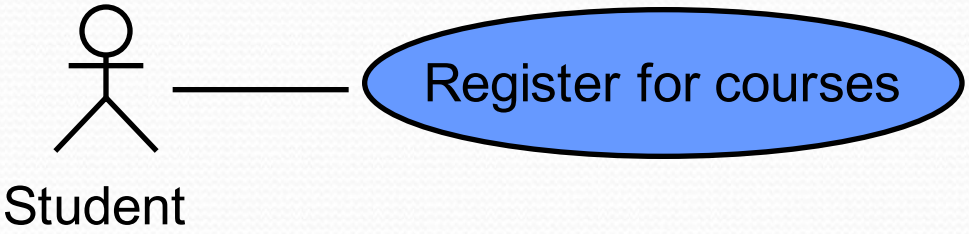
Name the use case

- A use case name should:
  - Be unique, intuitive, and self-explanatory
  - Define clearly and unambiguously the observable result of value gained from the use case
  - Be from the perspective of the actor that triggers the use case
  - Describe the behavior that the use case supports
  - Start with a verb and use a simple verb-noun combination

Use-case diagram example



Describe a use case (text description)

Name	Register for Courses
Brief description	The student selects the courses they wish to attend in the next semester. A schedule of core and option courses is produced.
Relationships with actors	

Use Case ‘Simple’ Description of a requirement.

Use Case No:	***Use Case Name:***	***Rating MoSCoW***
Purpose:		
Main actor:	***Secondary Actors:***	
Description: •		