COMP433: Software Engineering

Unified Modelling Language (UML)

Prof. Adel Taweel
ataweel@birzeit.edu

Objectives

To explain unified modelling language as object modelling tools.

To describe

- Different types of UML diagrams
- UML modelling techniques/tools and their applied use

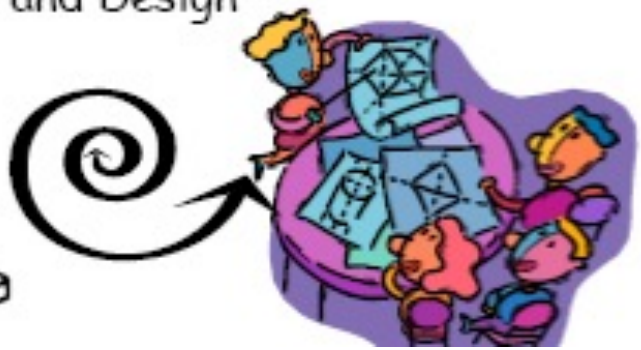
Covering…!!

Requirements Elicitation

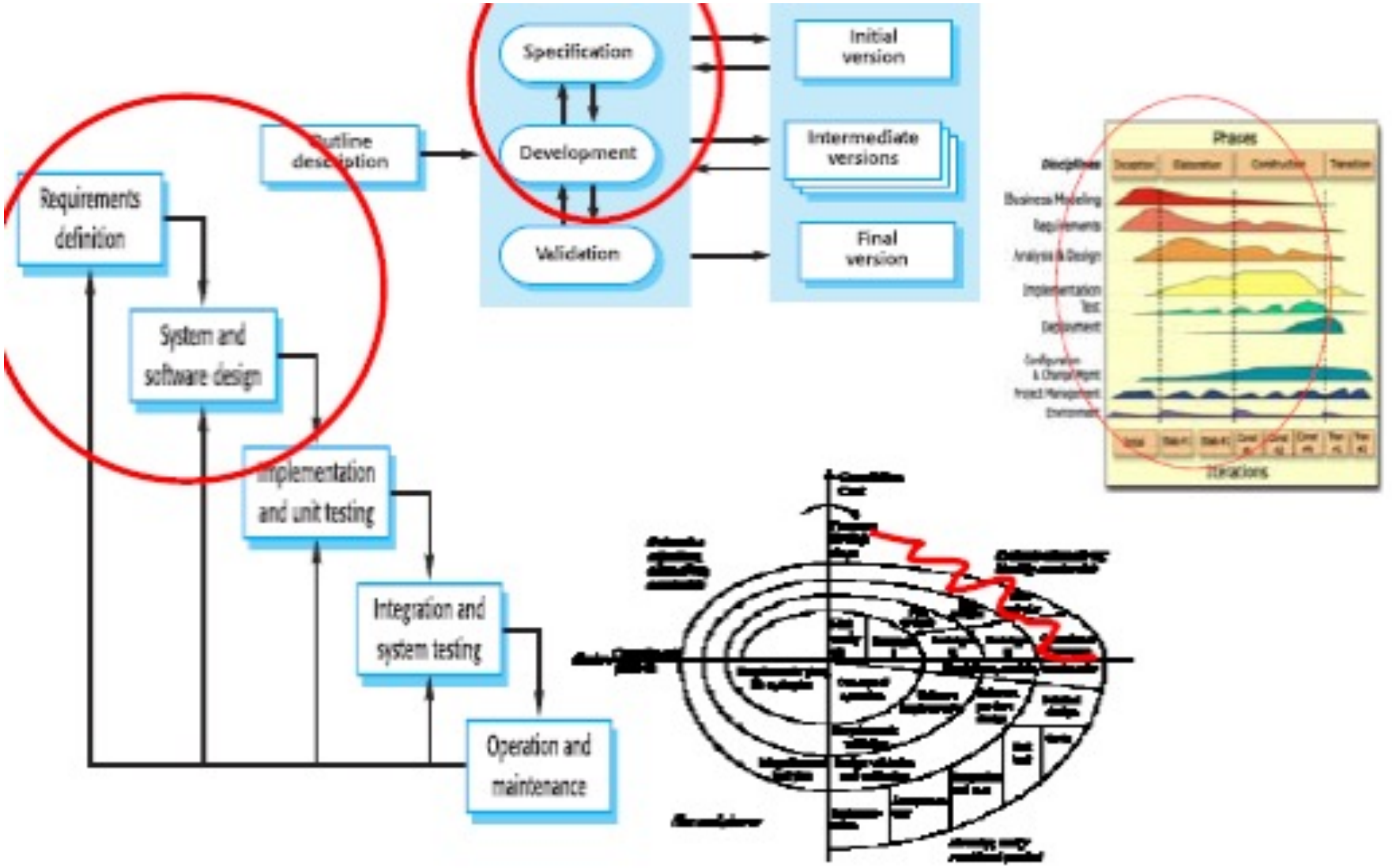
Requirements Specification

Go ahead

Analysis and Design



A ‘tool’ for…


Specification
Development
Validation
Initial version
Intermediate versions
Final version
Outline description
Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance
Phases
Iterations

The Unified Modelling Language

Several different notations for describing object oriented designs were proposed in the 1980s and 1990s.

The Unified Modelling Language is an integration of these notations.

It describes notations for a number of different models that may be produced during OO analysis and design.

It is now a de facto standard for OO modelling.

UML

Unified Modelling Language

Visualising and documenting analysis and design effort.

Unified because it …

Combines main preceding OO methods (Booch by *Grady Booch, OMT by Jim Rumbaugh and OOSE by Ivar Jacobson)*

Modelling because it is …

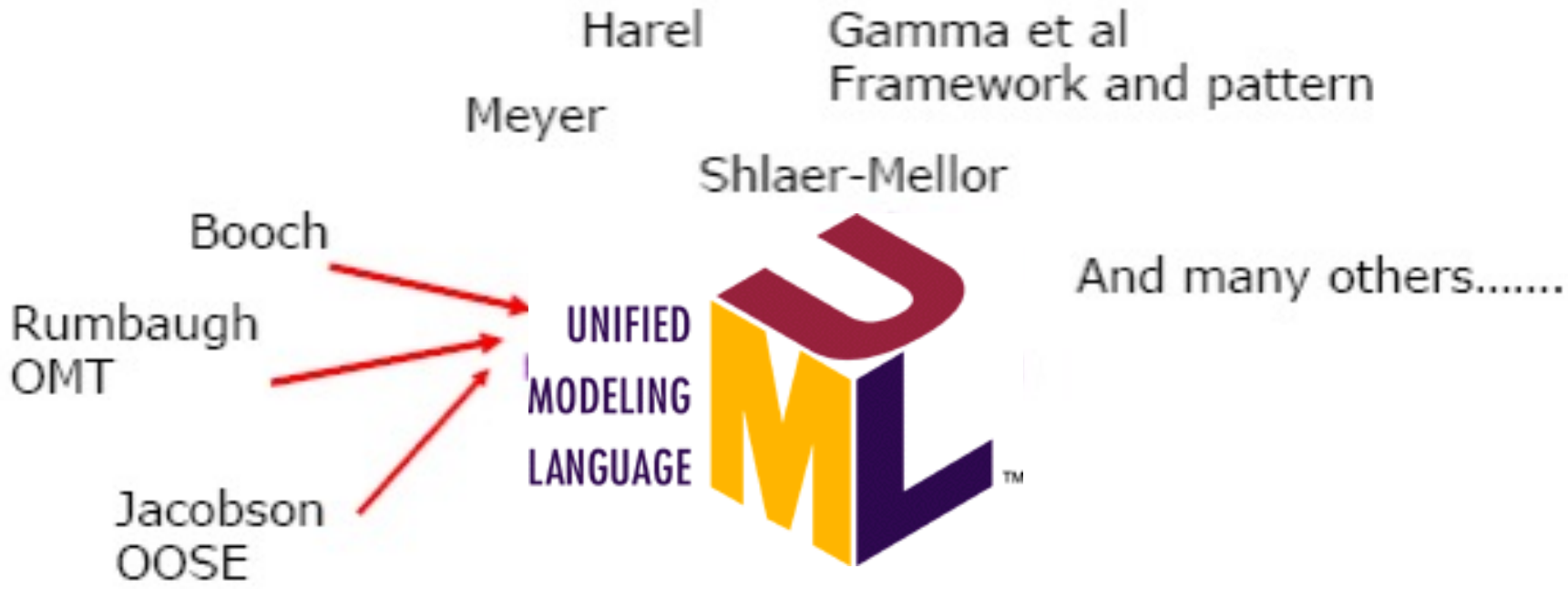
Primarily used for visually modelling systems. Many system views are supported by different appropriate models

Language because …

It offers a syntax through which to express modelled knowledge

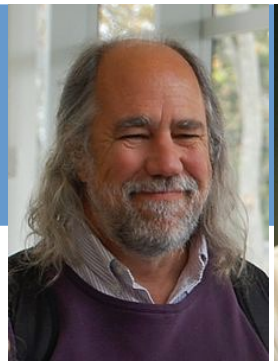
UML Contributors

- http://www.uml.org/



Major three (submission to OMG Jan 97, Acceptance Nov 97...)
http://www.omg.org/

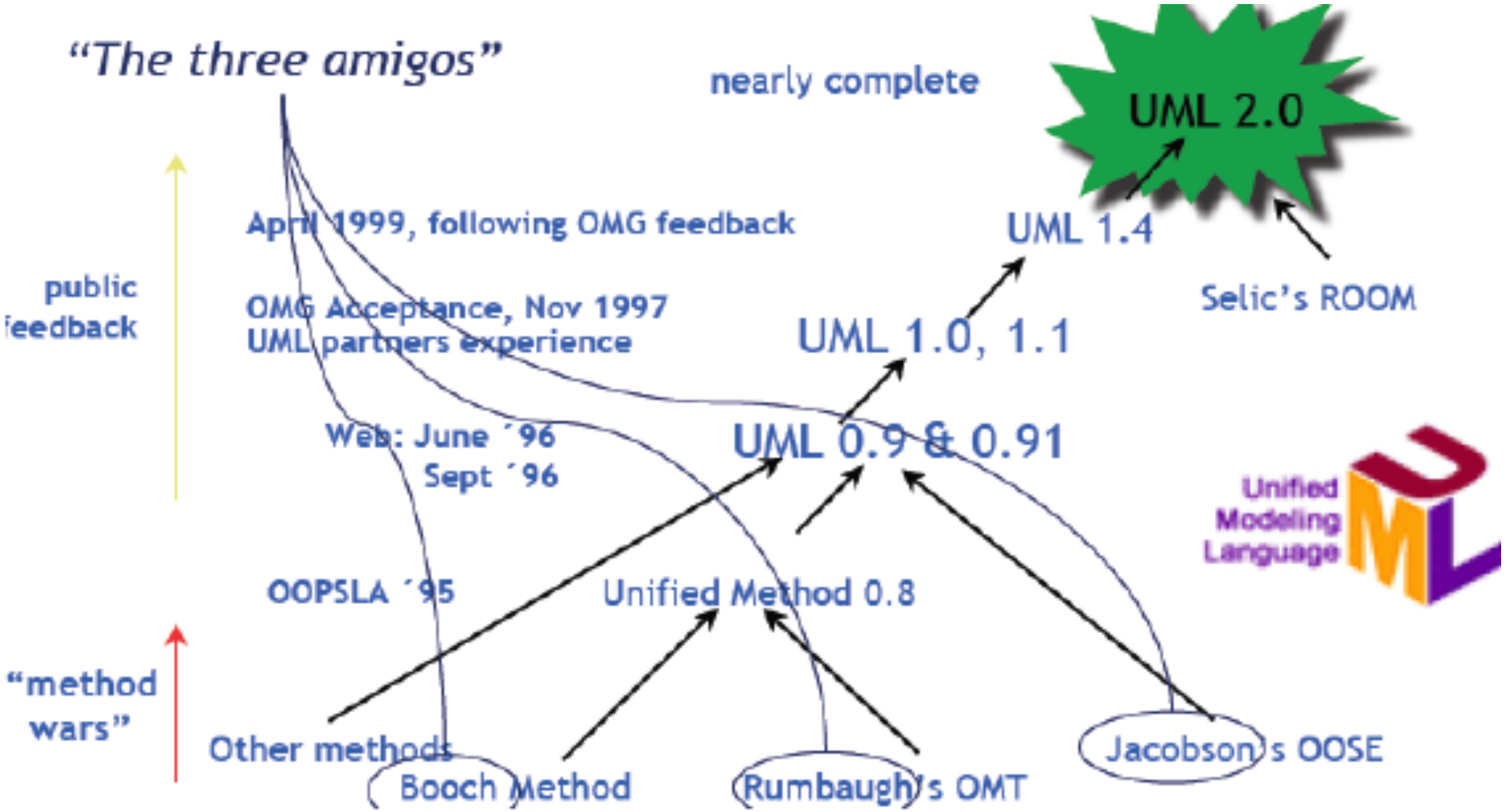
The Three Amigos!



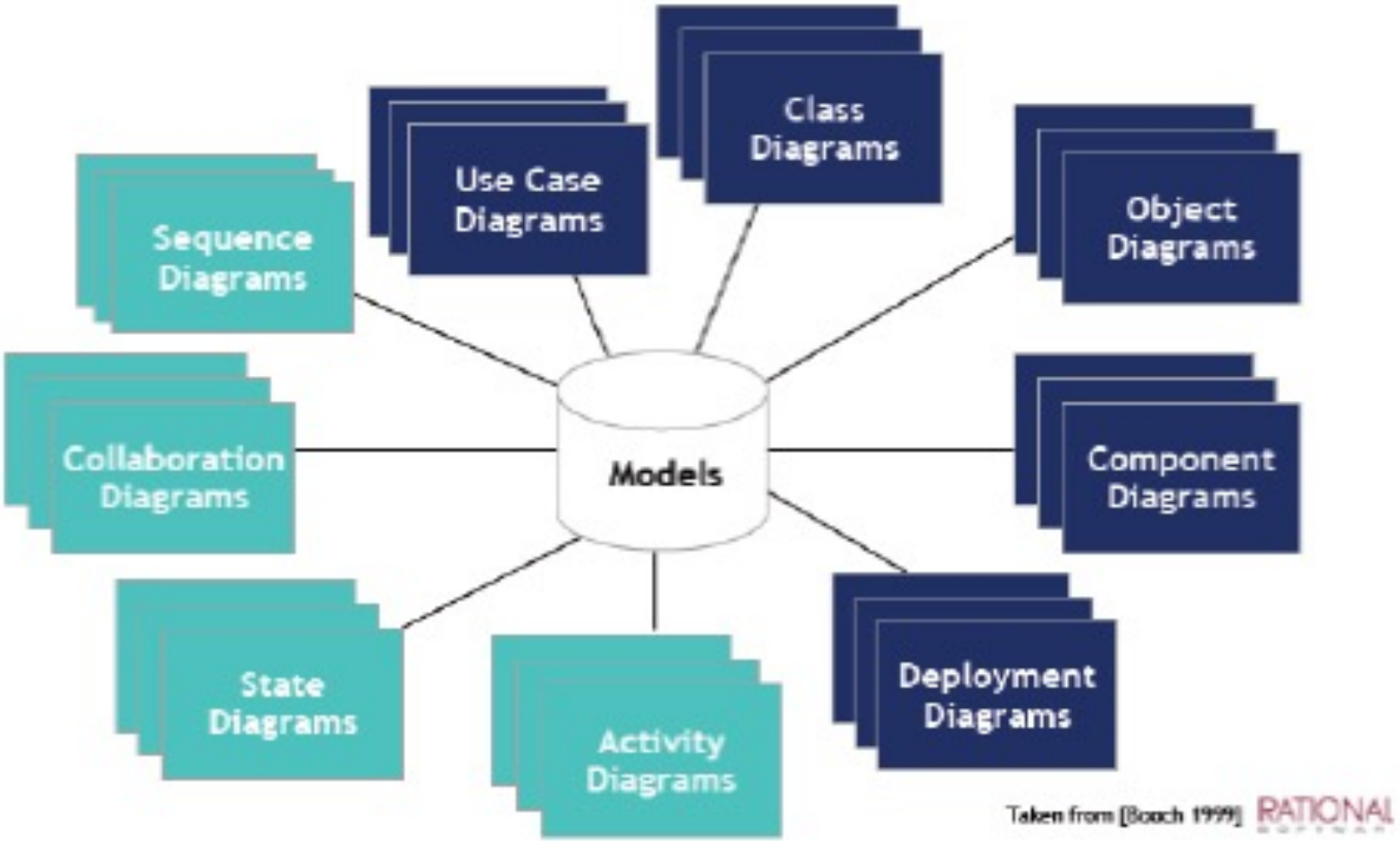
Grady Booch,
Ivar Jacobson,
and Jim Rumbaugh – historically and fondly known in the UML community as *The Three Amigos* – are often credited with the dominant contribution to the Unified Modeling Language

Grady Booch Ivar Jacobson James Rumbaugh

UML History


"The three amigos"
nearly complete
UML 2.0
April 1999, following OMG feedback
UML 1.4
public feedback
OMG Acceptance, Nov 1997
UML partners experience
Selic's ROOM
UML 1.0, 1.1
Web: June '96
Sept '96
UML 0.9 & 0.91
Unified Modeling Language
OOPSLA '95
Unified Method 0.8
"method wars"
Other methods
Booch Method
Rumbaugh's OMT
Jacobson's OOSE

UML Diagrams



Models

The language of the designer

(real-world) Representations of the system to-be-built or as built

A complete description of a system from a particular perspective

Tools for communication with various stakeholders

Allow reasoning about some characteristics of a system

Often captures both structural and behavioural (e.g., interaction) aspects of the system

UML Diagrams

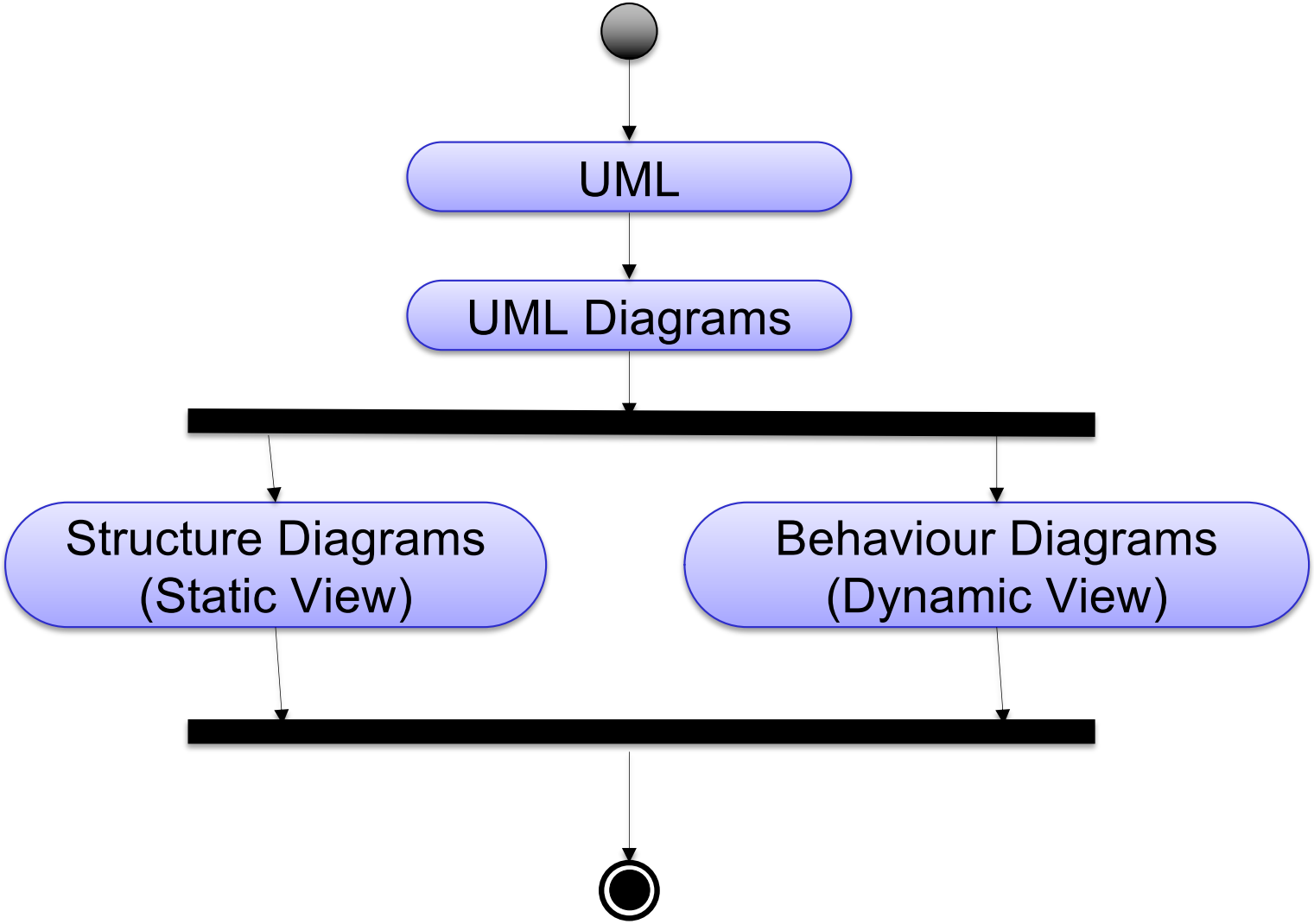
Diagram: a view into the model

In UML, there are more than fourteen modelling diagrams, but nine are considered standard diagrams:

Structure diagrams [Static view]: use-case, class, object, component, deployment

Behaviour/Interaction diagrams [Dynamic view]: activity, sequence, communication/collaboration, state

Model of UML Diagrams!



Summary of UML Diagrams (1): Structure

Use Case Diagram
Shows use cases, actors, and their interrelationships

Class Diagram
Shows a collection of static model elements such as classes and types, their contents, and their relationships

Component Diagram
Depicts the components that compose an application, system, or enterprise. The components, their interrelationships, interactions, and their public interfaces are depicted

Deployment Diagram
Shows the execution architecture of systems. This includes nodes, either hardware or software execution environments, as well as the middleware connecting them

Object Diagram
Depicts objects and their relationships at a point in time, typically a special case of either a class diagram or a communication diagram

Summary of UML Diagrams (2): Dynamic

Activity Diagram

Depicts high-level business processes, including data flow, or to model the logic of complex logic within a system

Sequence Diagram

Models the sequential logic, in effect the time ordering of messages between classes (or classifiers)

Communication/Collaboration Diagram

Shows instances of classes, their interrelationships, and the message flow between them. Communication diagrams typically focus on the structural organization of objects that send and receive messages. Formerly called a Collaboration Diagram

State (Machine) Diagrams – Behavioral and Protocol

Describes the states an object or interaction may be in, as well as the transitions between states. Formerly referred to as a state chart diagram, or a state-transition diagram. A behavioral state machine examines the behavior of a class; a protocol state machine illustrates the dependencies among the different interfaces of a class

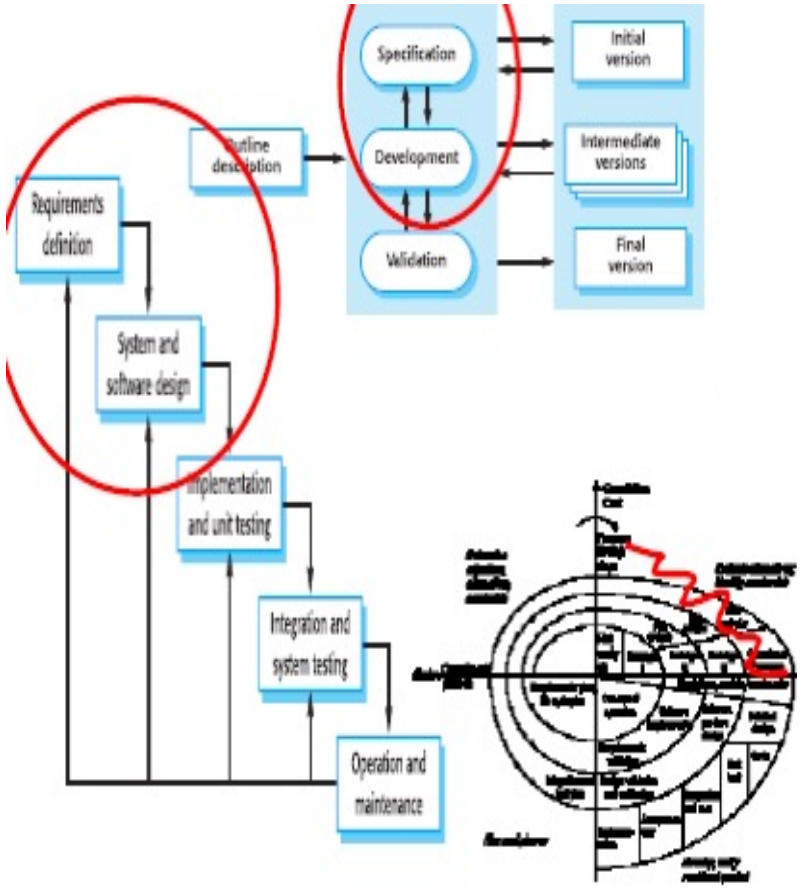
UML Diagrams vs Software life Cycle/Process models

Analysis:

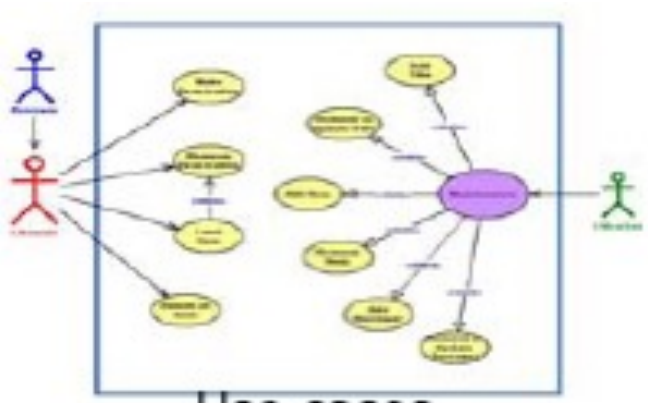
- Requirement Engineering:
  - Elicitation/discovery: User+system requirements->scenarios, interviews etc.
  - Requirement Analysis (of a Business/System)): [use case] + [Activity]
  - Specification: [Use case description]

Design:

- System Analysis
  - Design options: [Component]
- System/object Design/Modelling
  - System Entities: [Class]+[object]
  - Interactions: [Sequence/communication] + [State]
- System Design
  - Architecture/component view: [Component]
  - Execution view: [Deployment]



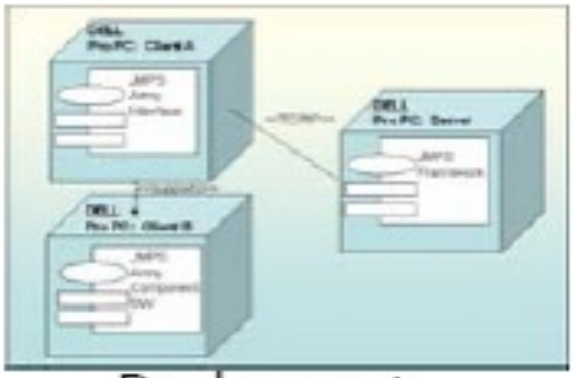
Examples of UML Diagrams



Use cases

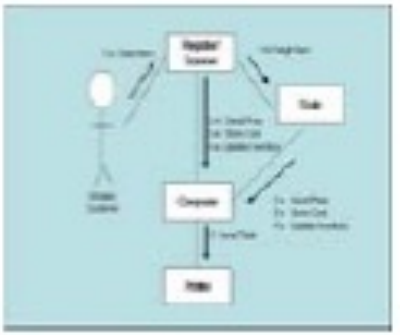
Class diagram

activity



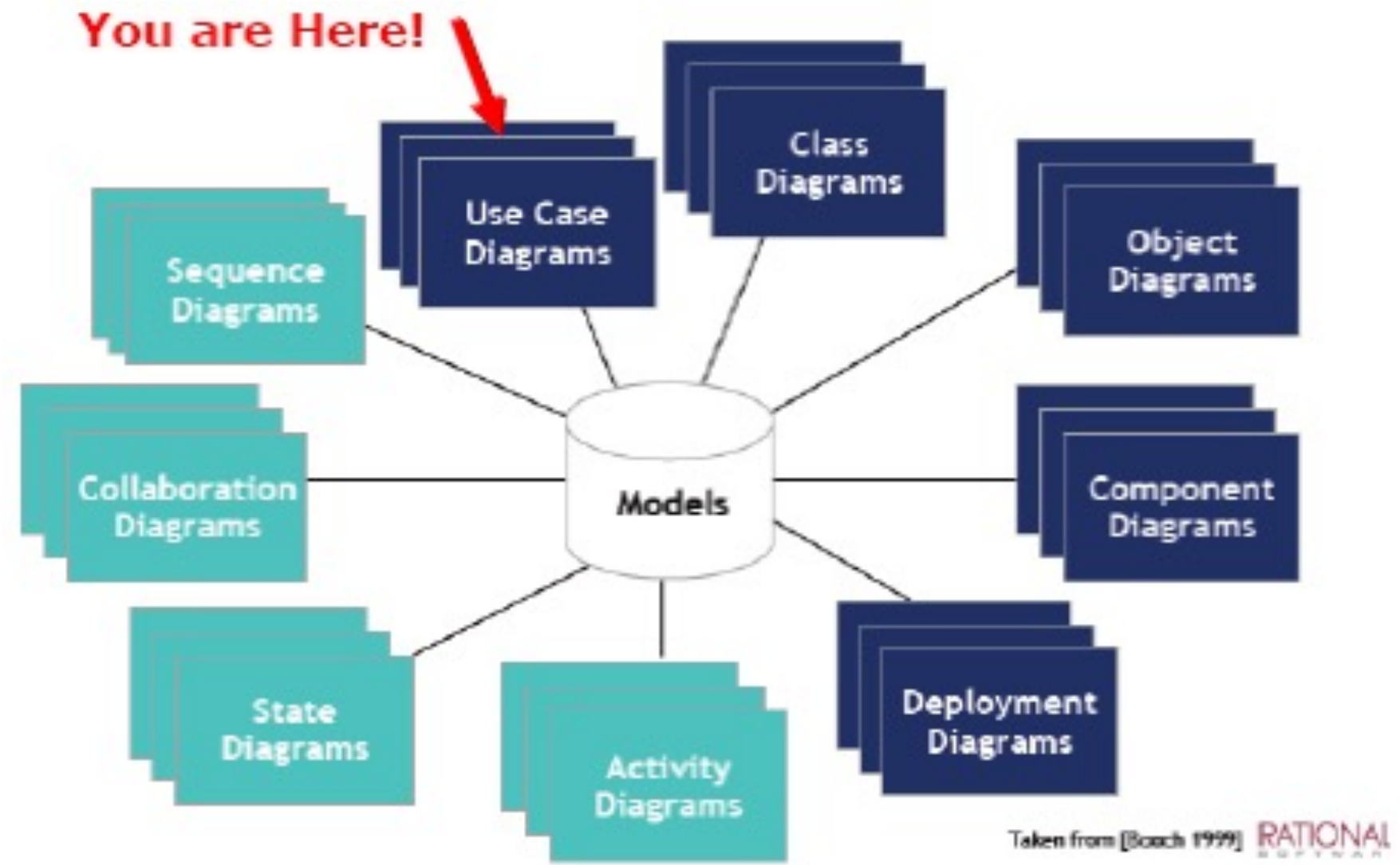
Deployment

Sequence



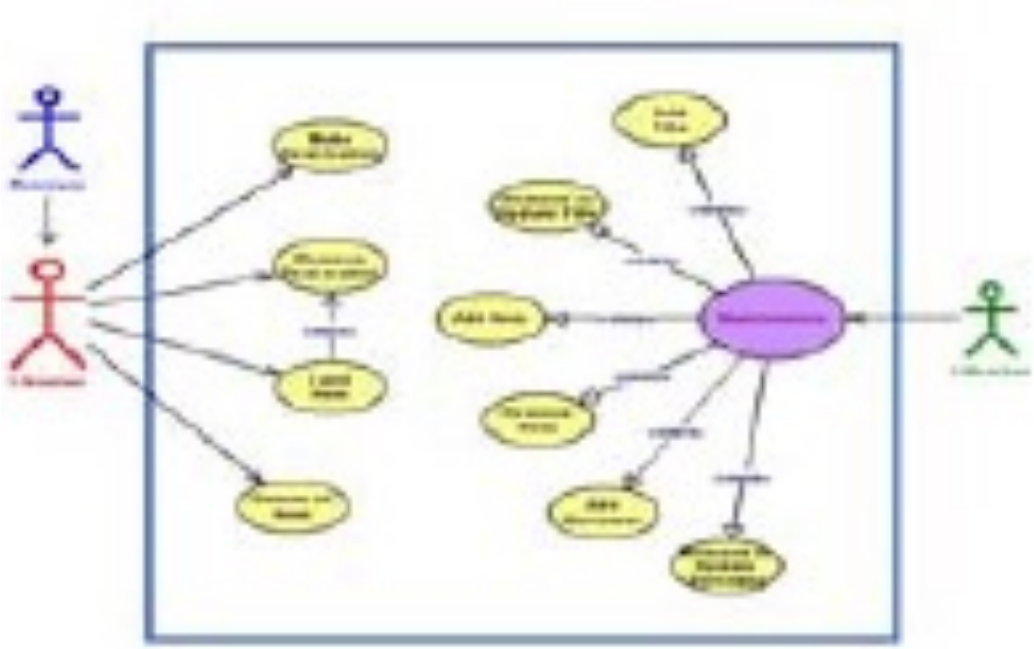
Collaboration

UML Diagrams



Use Cases

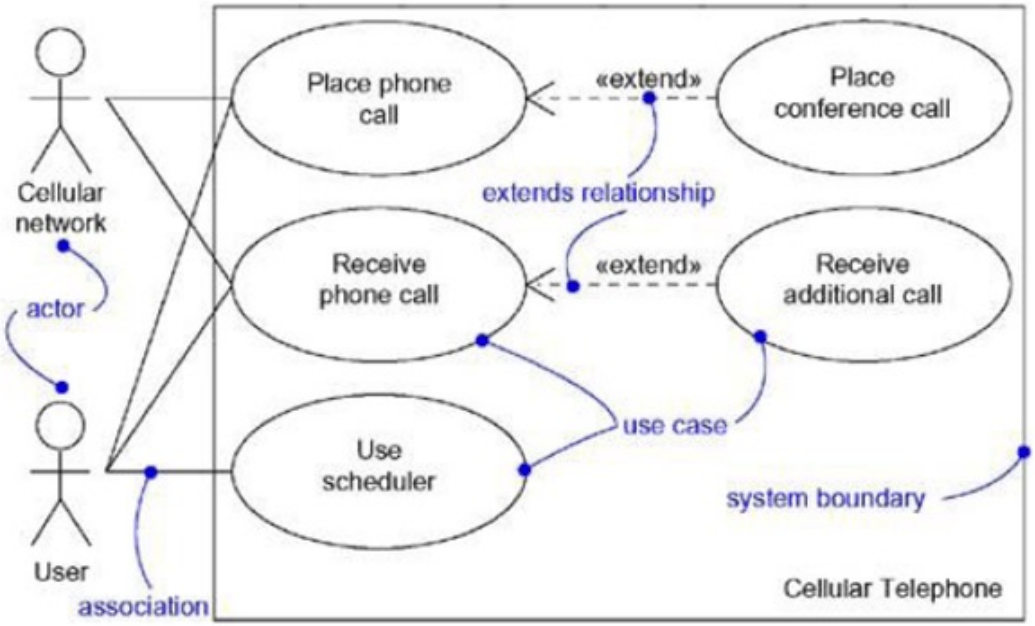
What is use case modelling?
What are actors?
How to find actors?
What are use cases?
How to find use cases?
How to construct a use case diagram?
Detailing a use case…



What is use case modelling?

Basis for a user-oriented approach to system development

- Identify the **users** of the system (**actors**)
- Identify the **tasks** they must undertake with the system (**use cases**)
- Relate users & tasks (**relationship or association**)… help identify system boundary
- Capture system functionality as seen by *users*



Booch 1999

Use cases?

Represent that an Actor has a **case** of (or for) **using** the system. The tasks that must provided by the system to the user (Actor) to undertake.

Use cases:

- Built in early stages of development
- Developed by analysts & domain experts during requirements analysis

Use cases aid to:

- Specify the context of a system
- Plan iterations of development
- Validate a system's architecture
- Drive implementation & generate test cases

How to identify Actors?

Observe direct users of the system- could be users or systems
- What roles do they play?
- Who provides information to the system?
- Who receives information from the system?

Actors could be:
- Principal
- Secondary (External hardware, other systems, …)

Describe each actor clearly and precisely (semantics)
- Short name: always a **Noun**
- Description: describe what is their role and how they interact with the system

Example:

BookBorrower: This actor represents someone (or a user) that makes use of the library for borrowing books [Principal actor]

SystemTimer: This actor represents a system-event that triggers regularly (automatically) checking expired loans [Secondary Actor]

Exercise!

Assume you have a requirements documents for a Patient Medical System (PMS): identify KEY actors that interact with the system

For each actor, write down the name and provide a brief textual description (i.e., describing the semantics of the actor)

Actor	Semantics
Name 1	Description

Exercise: Potential Actors!

Actor	Semantics/Description
Doctor	This actor represents someone who is a member of the PMS, registered on the system, that can view and edit patient records only
Nurse	This actor represents someone who is a member of the PMS, registered on the system, that can view and edit patient records
Receptions	This actor represents someone who is a member of the PMS, registered on the system, that can view, Edit and create patient records
Patient	This actor represents someone who their information is registered on the system, but can not view, edit their records
IT Staff	This actor represents someone who can maintain patient records
Lab Staff …	This actor represents someone who can edit patient records to enter lab tests only

Exercise!

Assume you have a requirements document for a library system: identify all actors that interact with the system

For each actor, write down the name and provide a brief textual description (i.e., describing the semantics of the actor)

Actor	Semantics
Name 1	Description

Exercise: Potential Actors!

Actor	Semantics/Description
BookBorrower	This actor represents someone who is a member of the library, registered on the system, that can borrow books only
JournalBorrower	This actor represents someone who is a member of the library, registered on the system, that can borrow books and journals
BookBrowser	This actor represents someone who can search for books or journals (but may not be a member of the library and cannot borrow books or journals)
BookClassifier	This actor represents someone who classifies/catalogs new books and registers them in the systems
BookReturnRegistrar	This actor represents someone who receives returned books and registers them in the system
BookLendRegistrar	This actor represents someone who lends (or renews borrowing) books and registers them in the system.
BookShelver	This actor represents someone who shelves books and register book shelving status in the system

Librarian

What are use cases?

Things actors do with the system

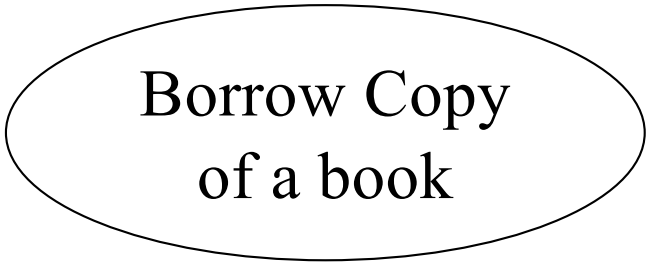
A **task** which an actor needs to perform with the help of a system (e.g., Borrow a book)

An **interaction** with another specific kind of a system

Describe the behaviour of the system from a **user's standpoint**

A role an actor takes in using the system.

Represented by **ellipses**



How to find Use Cases?

- Scenario-based analysis

  Write system processes (or services) as scenarios. Identify interactions with the system, each interaction is a potential use case!

- Actor-based analysis

  Identify actors, based on system users (and/or stakeholders). Then start with the list of actors and consider

  What they need from the system (i.e. what use cases that have value for each actor)

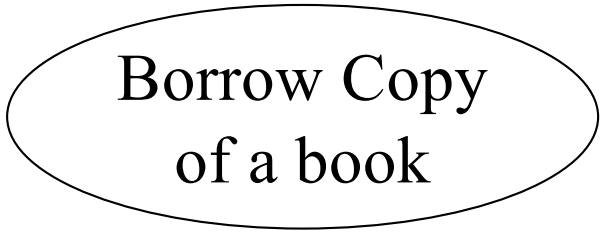
  Any other interactions they expect to interact with the system (i.e. which use cases they might take part in for someone's else benefit)

How do you know what is a use case?

Estimate frequency of use, examine differences between use cases, distinguish between "normal" and "alternative" course of events & create new uses when necessary

Describing use cases

Semantics should be described fully!
Always start a *use case* with a **verb**!



Example:

Use case: **Borrow copy of a book**

A book borrower presents a book. The system checks that the potential borrower is a member of the library & that s/he does not have the maximum number of books.