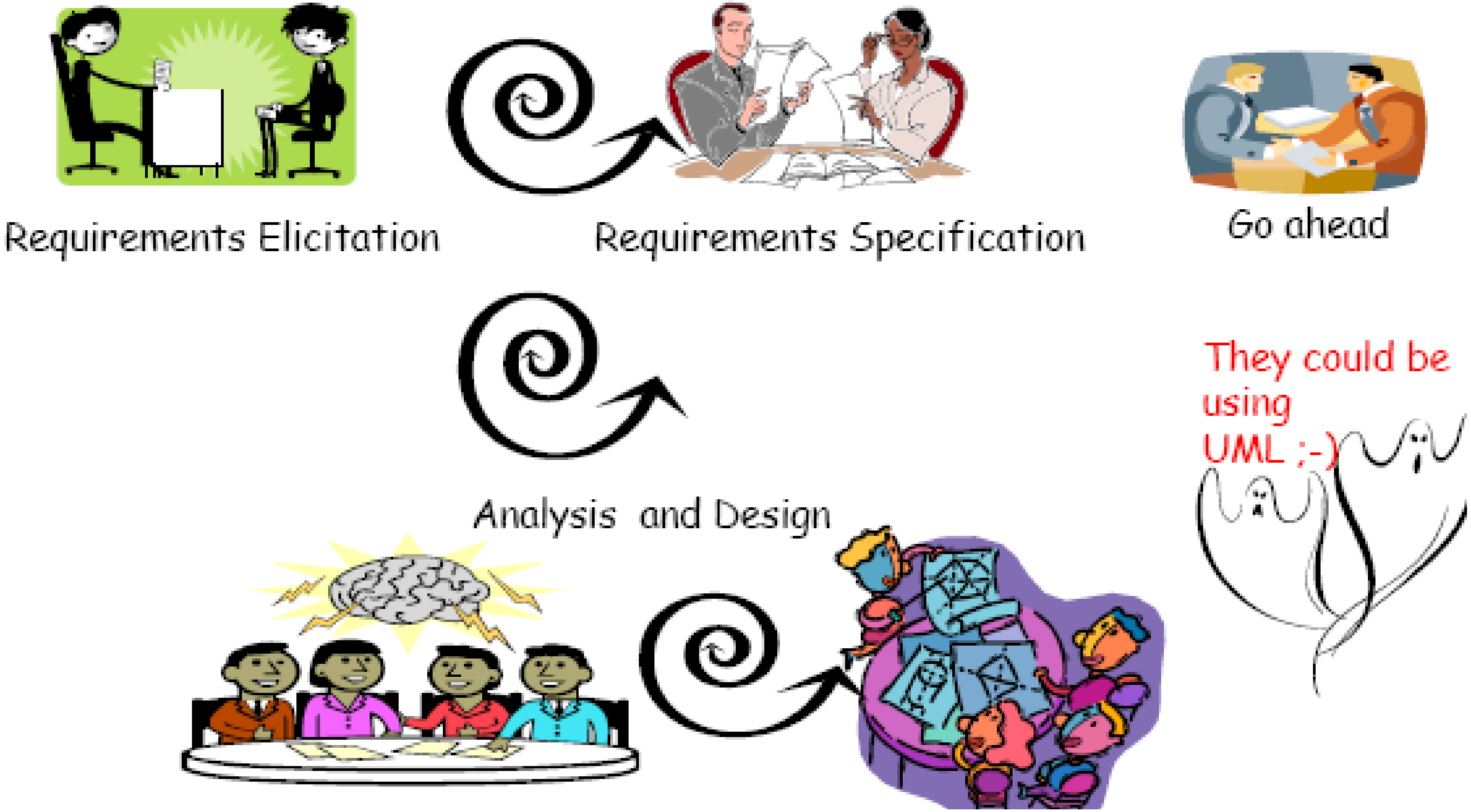
Unified Modelling Language (UML)

Dr Adel Taweel

Objectives

- To explain unified modelling language as object modelling tools.
- To describe
  - Different types of UML diagrams
  - UML modelling techniques/tools and their use

Covering…!!



Requirements Elicitation

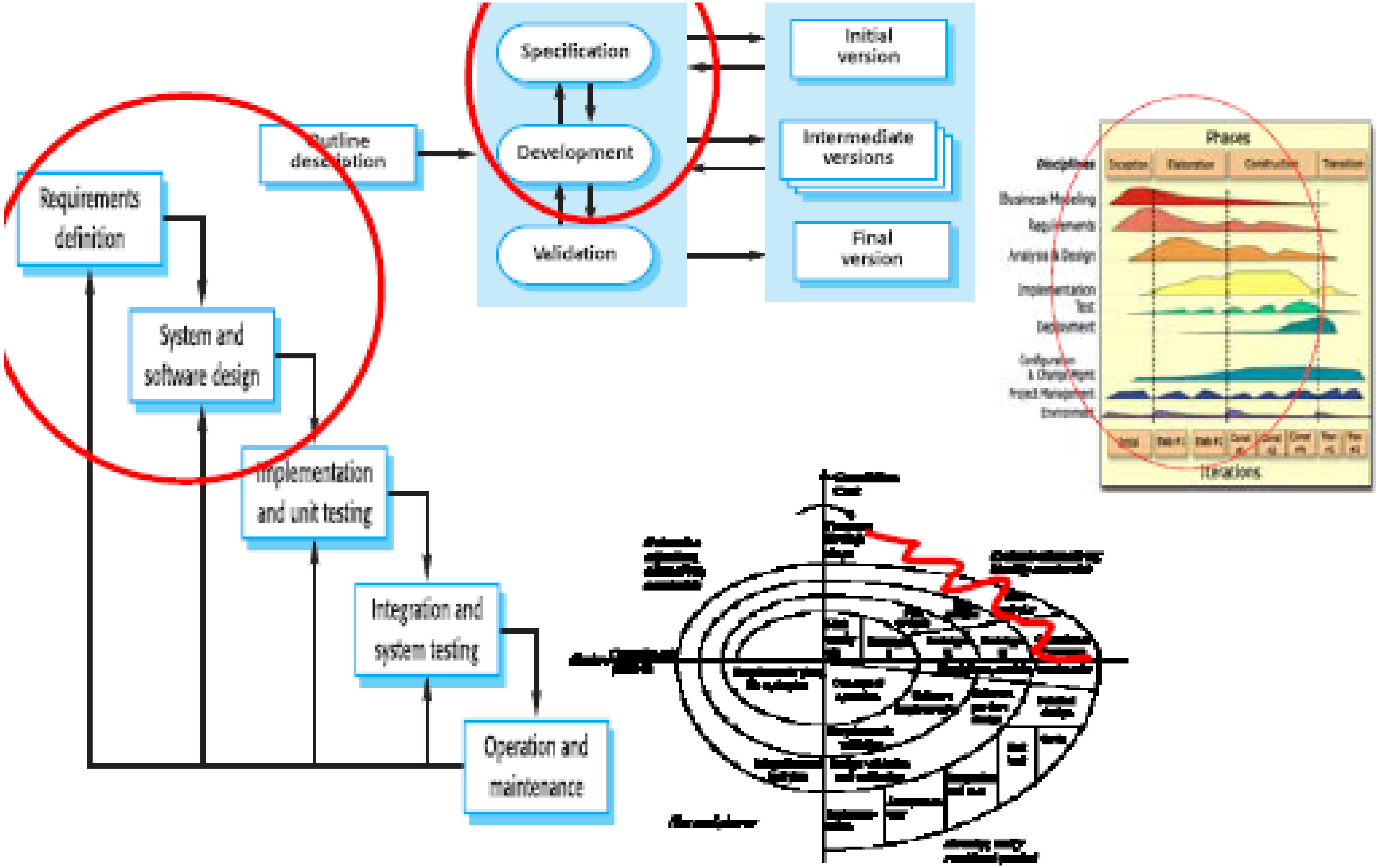
Requirements Specification

Go ahead

Analysis and Design

They could be using UML ;-)

A 'tool' for…


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance
Outline description
Specification
Development
Validation
Initial version
Intermediate versions
Final version
Phases
Disciplines
Inception
Elaboration
Construction
Transition
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Configuration & Change Mgmt
Project Management
Environment
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 …

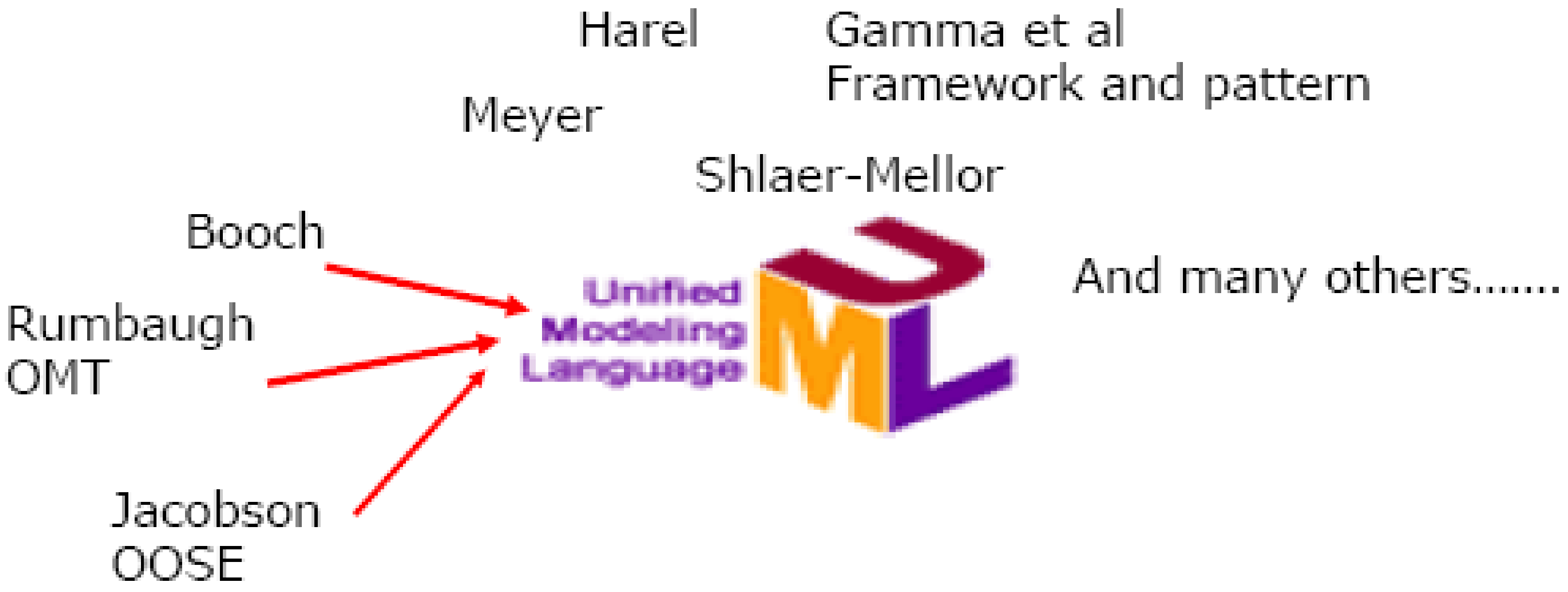
- Primarily used for visually modelling systems. Many system views are supported by different appropriate models

Language because …

- It offers a syntax through which to express modelled knowledge

UML Contributors

- http://www.uml.org/



Harel

Gamma et al
Framework and pattern

Meyer

Shlaer-Mellor

Booch

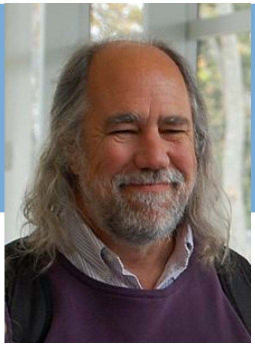
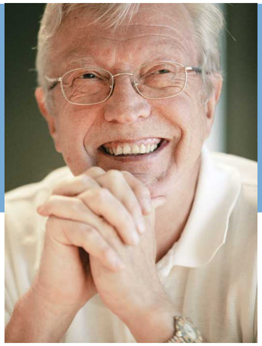
Rumbaugh
OMT

Jacobson
OOSE

And many others…….

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

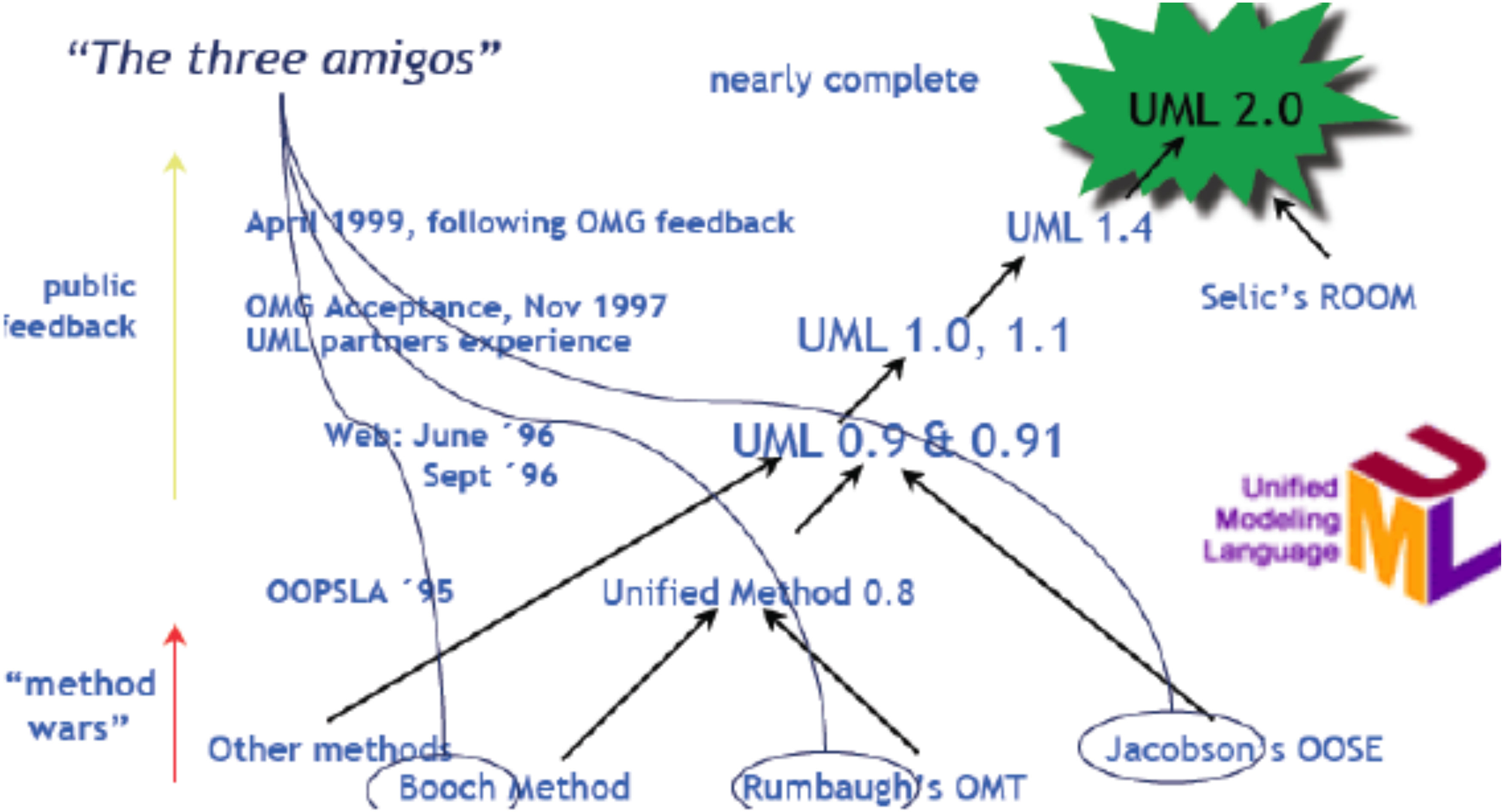
The Three Amigos!

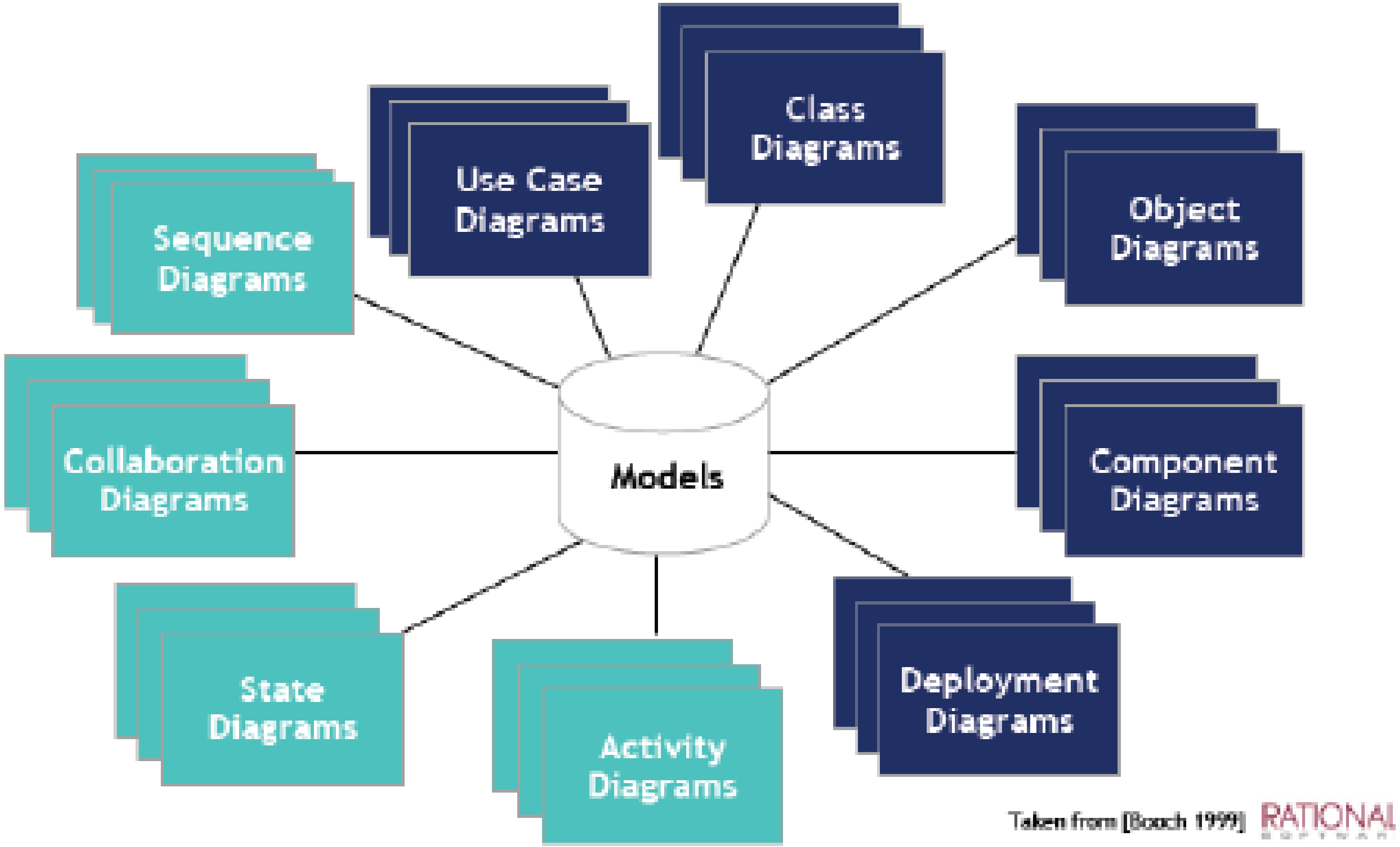
Grady Booch,
Ivar Jacobson,
and Jim Rumbaugh – historically and fondly known in the UML community as *The Three Amigos* – are often credited with the dominant contribution to the Unified Modeling Language

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
UML 1.0, 1.1
Selic's ROOM
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



Taken from [Booch 1999]

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) information

UML Diagrams

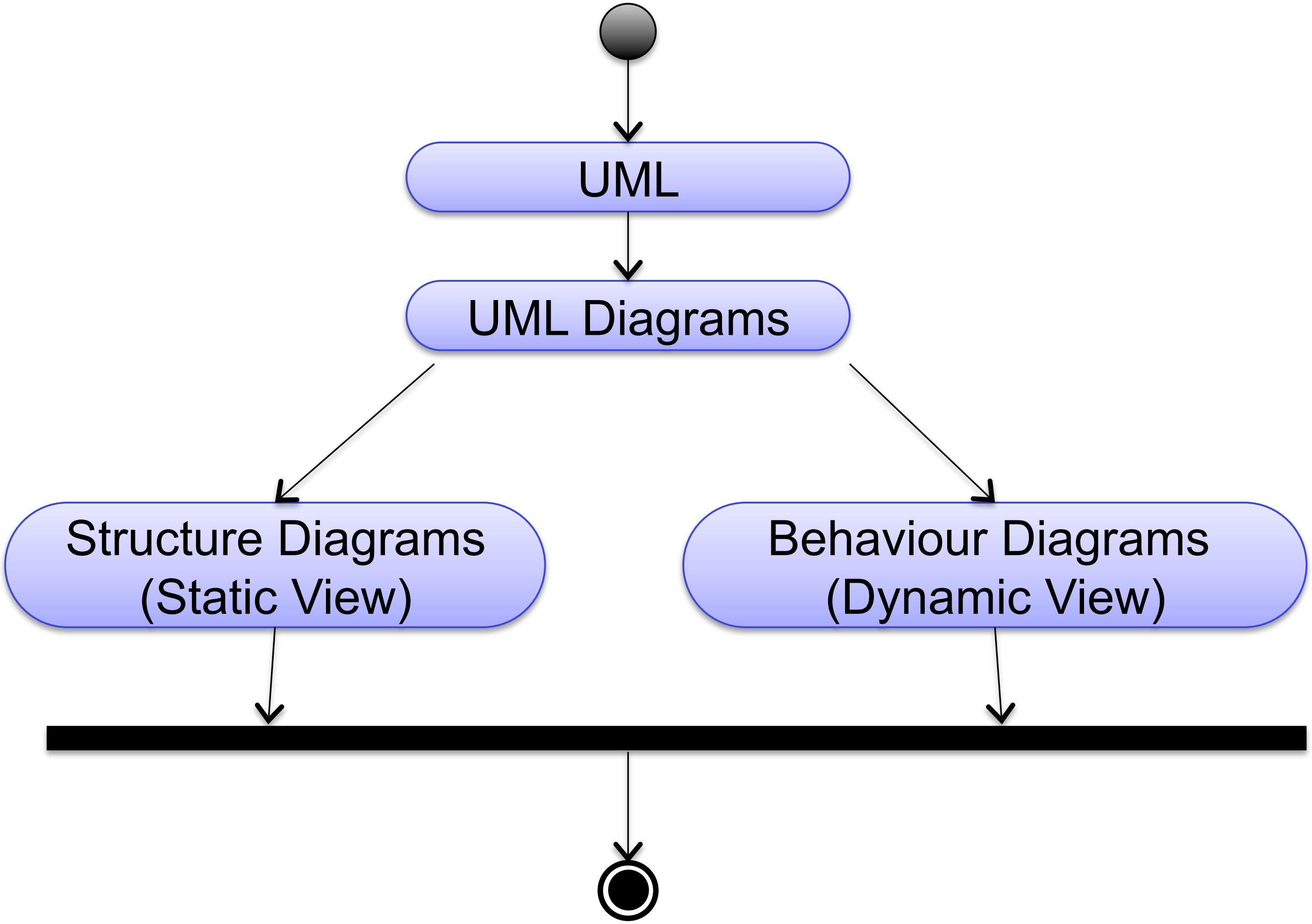
Diagram: a view into the model

In UML, there are nine 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!


UML
UML Diagrams
Structure Diagrams (Static View)
Behaviour Diagrams (Dynamic View)

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

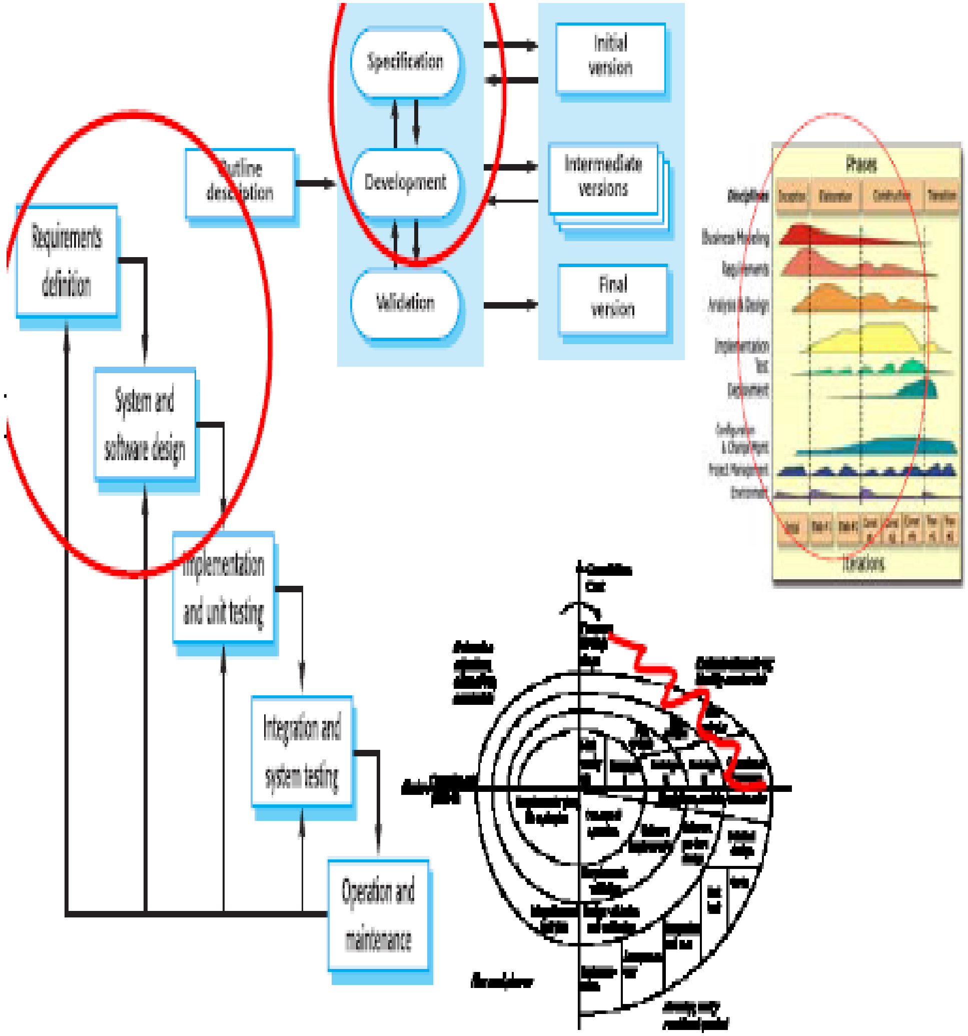
UML Diagrams vs Software life Cycle/Process models

Analysis:

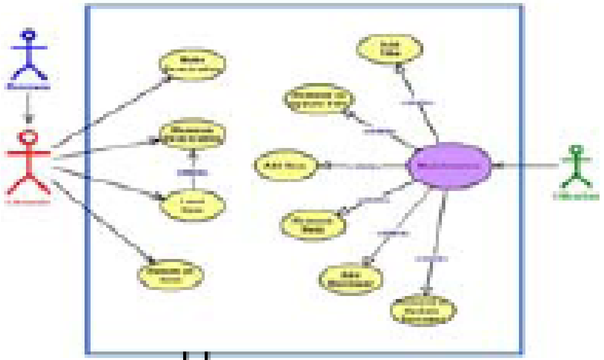
- Requirement Engineering:
  - Elicitation/discovery: User+system requirements->scenarios, interviews etc.
  - Requirement Analysis (of a Business/System)): [use case] + [Activity]
  - Specification: [Use case description]

Design:

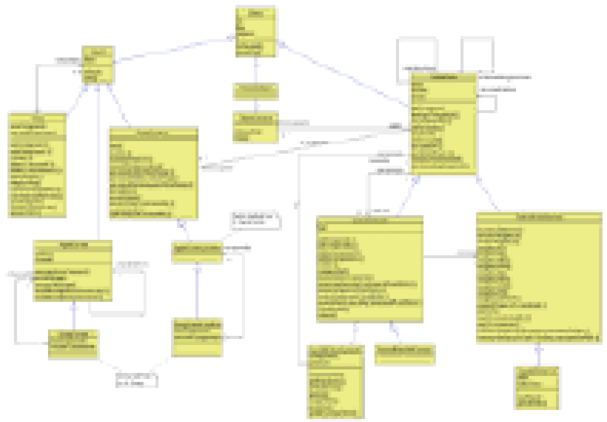
- System Analysis
  - Design options: [Component]
- System/object Design/Modelling
  - System Entities: [Class]+[object]
  - Interactions: [Sequence/ communication] + [State]
- System Design
  - Architecture/component view: [Component]
  - Execution view: [Deployment]



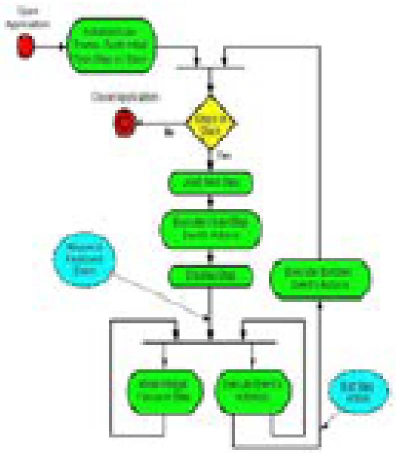
Examples of UML Diagrams



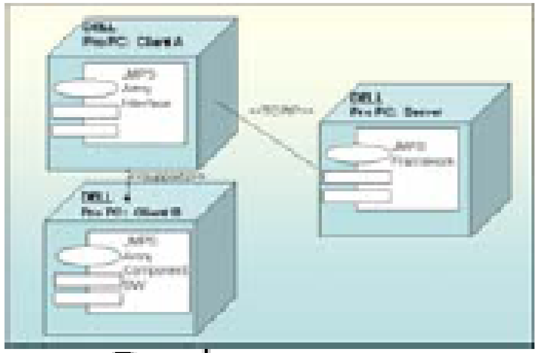
Use cases



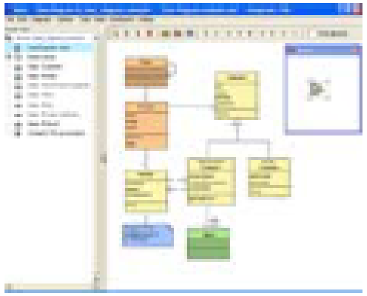
Class diagram



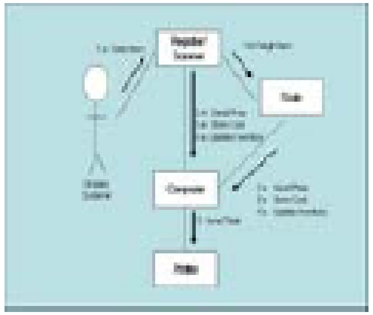
activity



Deployment



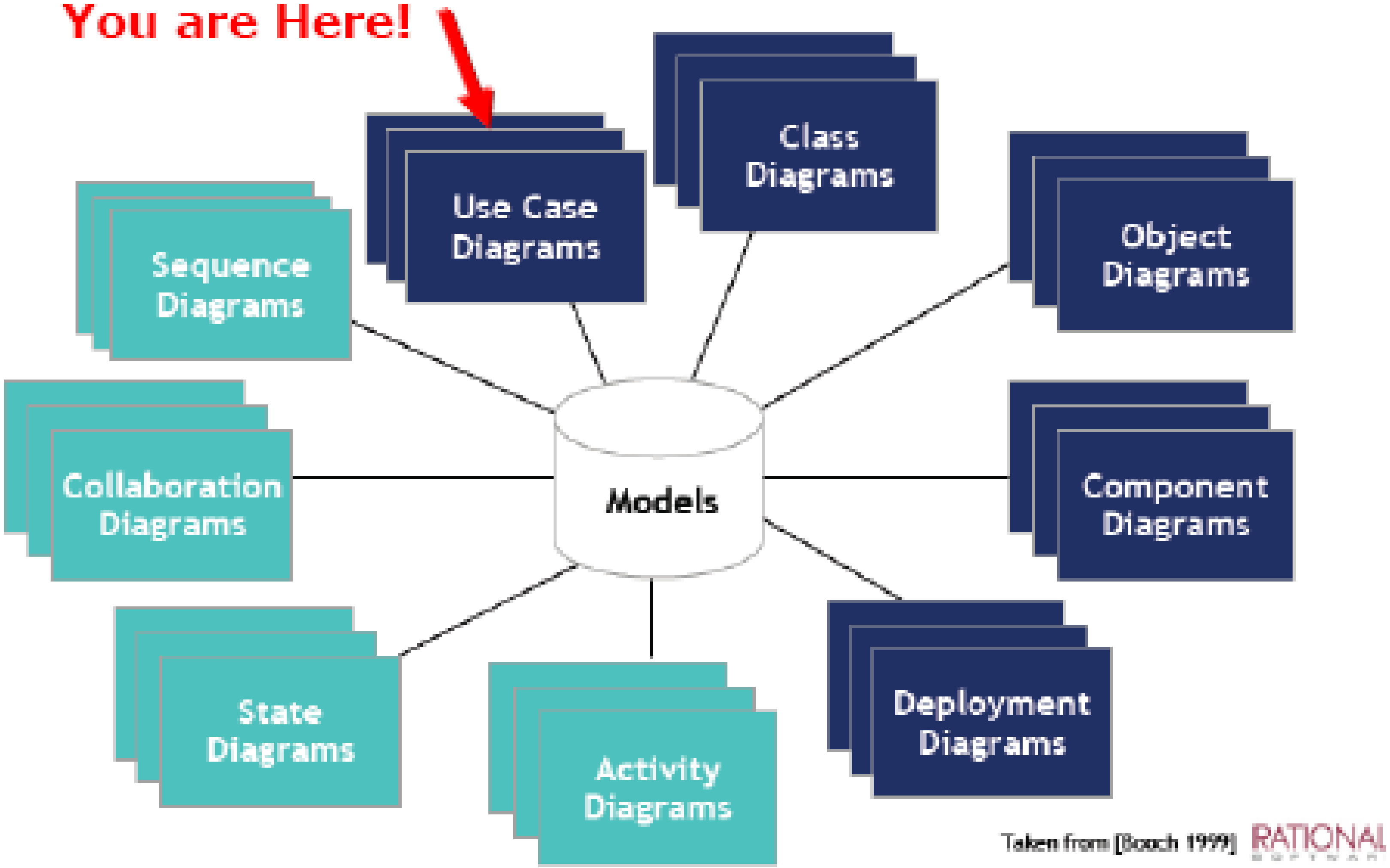
Sequence



Collaboration

Unified Modeling Language UML

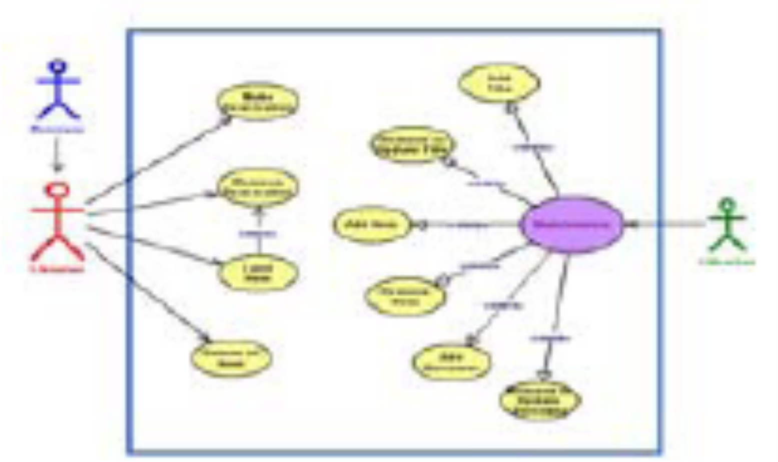
UML Diagrams



Taken from [Booch 1999]

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
Detailing a use case…



What is use case modelling?

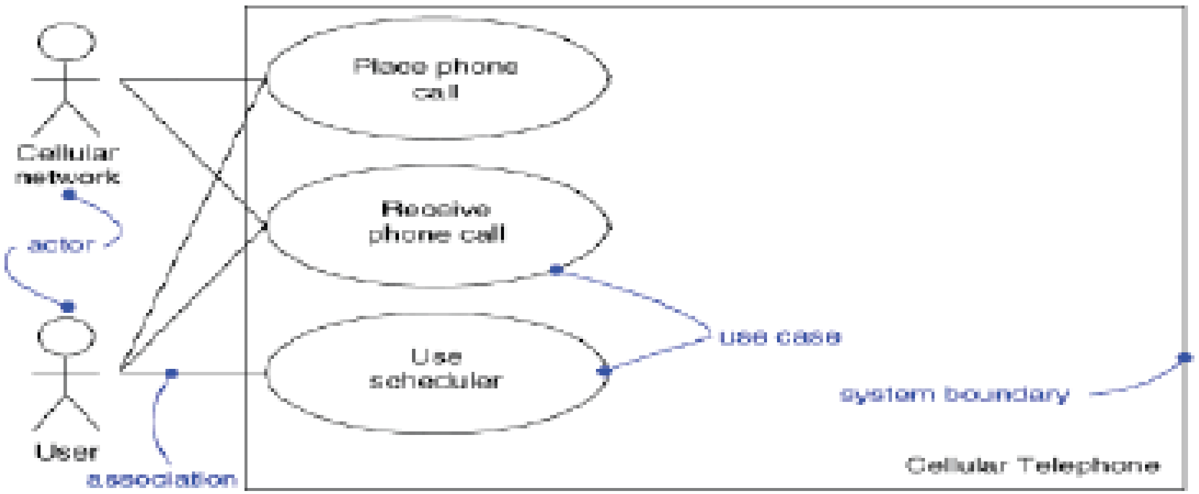
Basis for a user-oriented approach to system development

Identify the users of the system (actors)

Identify the tasks they must undertake with the system (use cases)

Relate users & tasks (relationship)… help identify boundary

Capture system functionality as seen by *users*



Taken from [Booch 1999]

Use cases?

Represent that an Actor has a **case** of (or for) **using** the system.

Use cases:

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

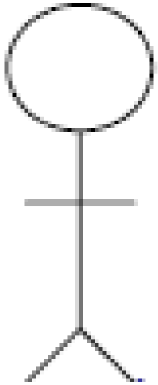
Use cases aid to:

- 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 some one (or a user) that makes use of the library for borrowing books

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

Exercise: Potential Actors!

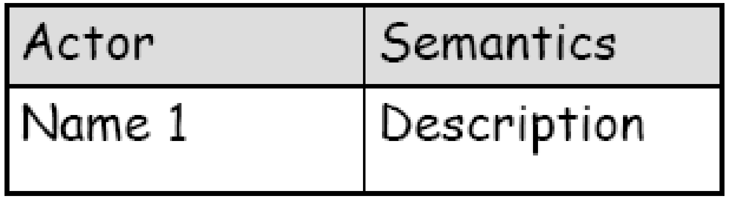
Actor	Semantics/Description
BookBorrower	This actor represents someone who is a member of the library, registered on the system, that can borrow books only
JournalBorrower	This actor represents someone who is a member of the library, registered on the system, that can borrow books and journals
BookBrowser	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

Exercise!

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

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



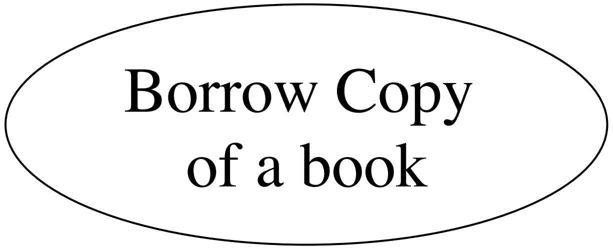
Actor	Semantics
Name 1	Description

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

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**



Borrow Copy of a book

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

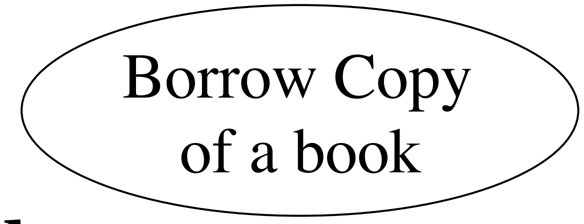
- Identify actors, based on system users (and/or stakeholders).
- Then start with the list of actors and consider
  - What they need from the system (i.e. what use cases that have value for each actor)
  - Any other interactions they expect to interact with the system (i.e. which use cases they might 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.

Example: Library System

Books and journals:

The library contains books and journals. It may have several copies of a given book. Some of the books are for short term loans only. All other books may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals. Members of the public, who are not members of the library, can use the library and browse/search for books, but cannot borrow books.

Borrowing/Returning/Renewing books:

The system must keep track of when books and journals are borrowed and returned, enforcing the rules described above.

Managing books:

The system must enable library staff/librarian to manage: add/update/catalog/remove existing and add new books and journals

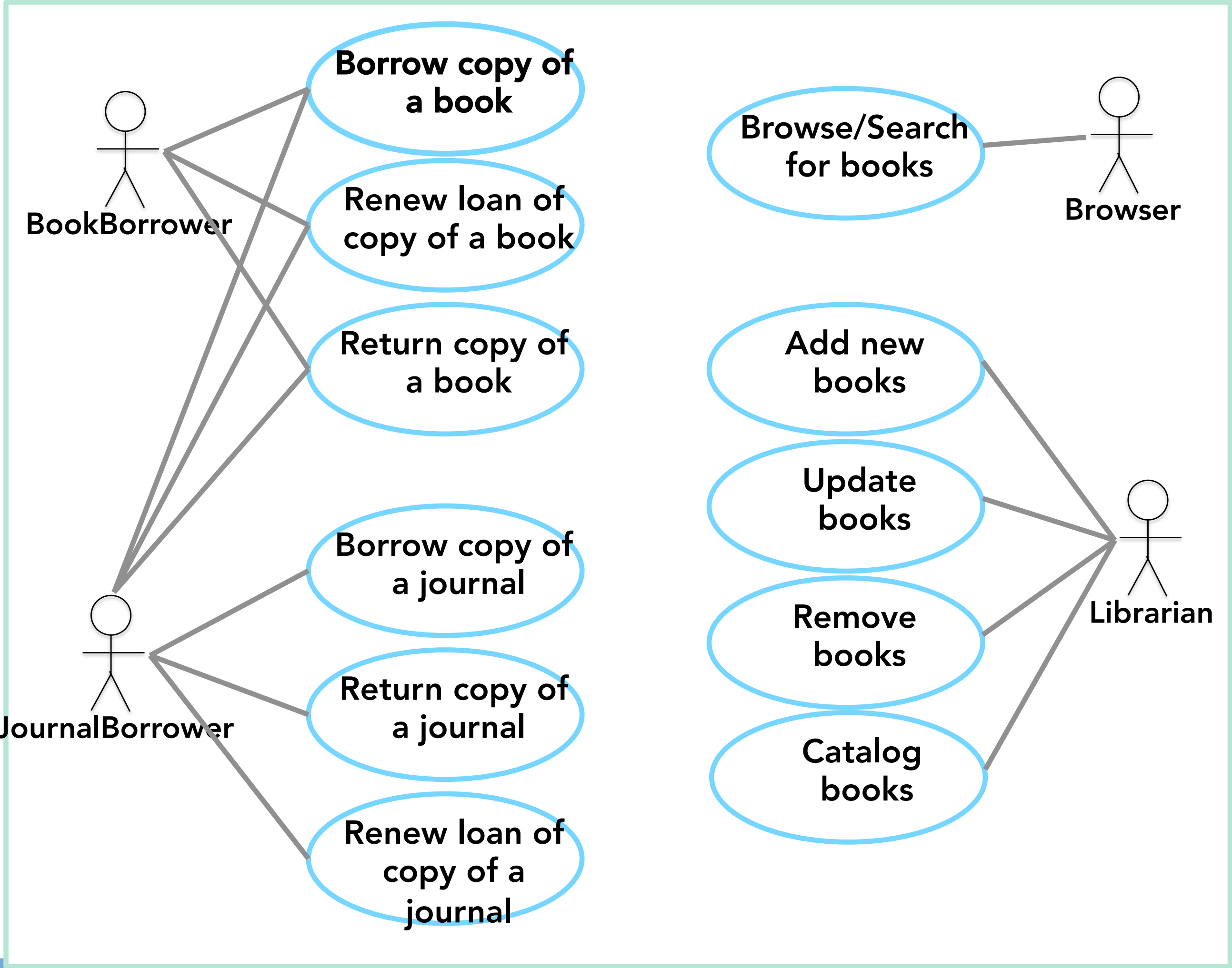
Example: Library System

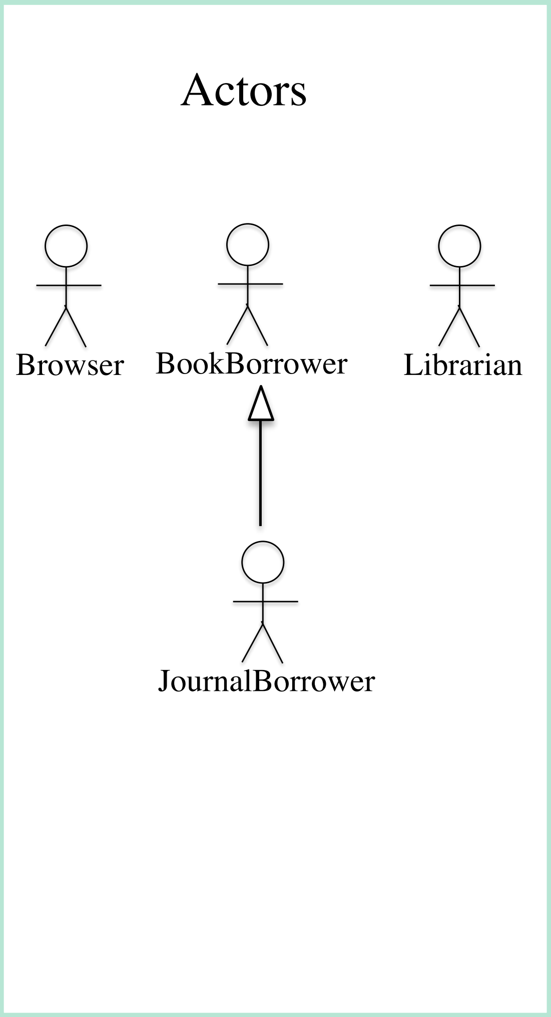
Books and journals:
The library contains books and journals. It may have several copies of a given book. Some of the books are for short term loans only. All other books may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals. Members of the public, who are not members of the library, can use the library and browse/search for books, but cannot borrow books.

Borrowing/Returning/Renewing books:
The system must keep track of when books and journals are borrowed and returned, enforcing the rules described above.

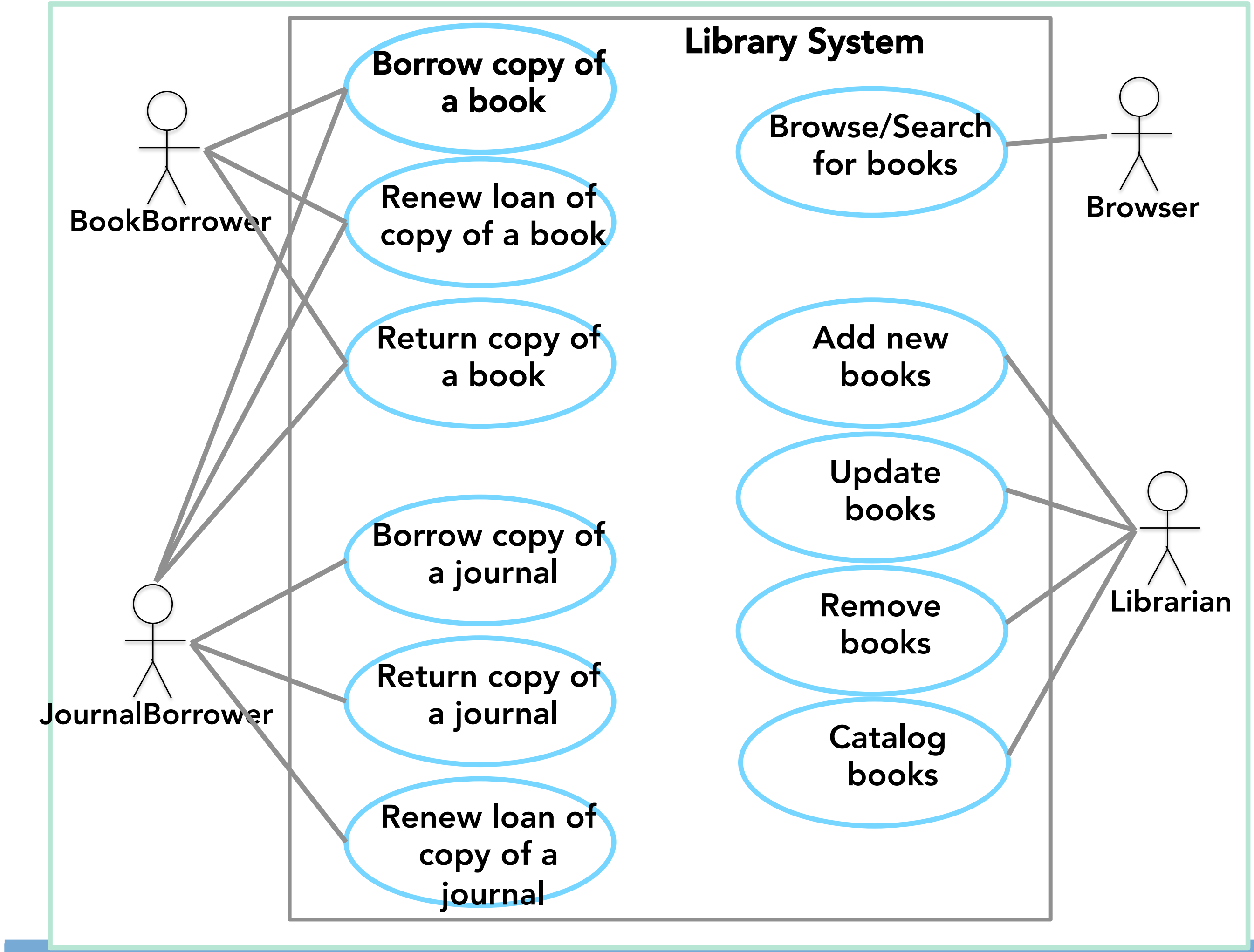
Managing books:
The system must enable library staff/librarian to manage: add/update/catalog/ remove existing and add new books and journals

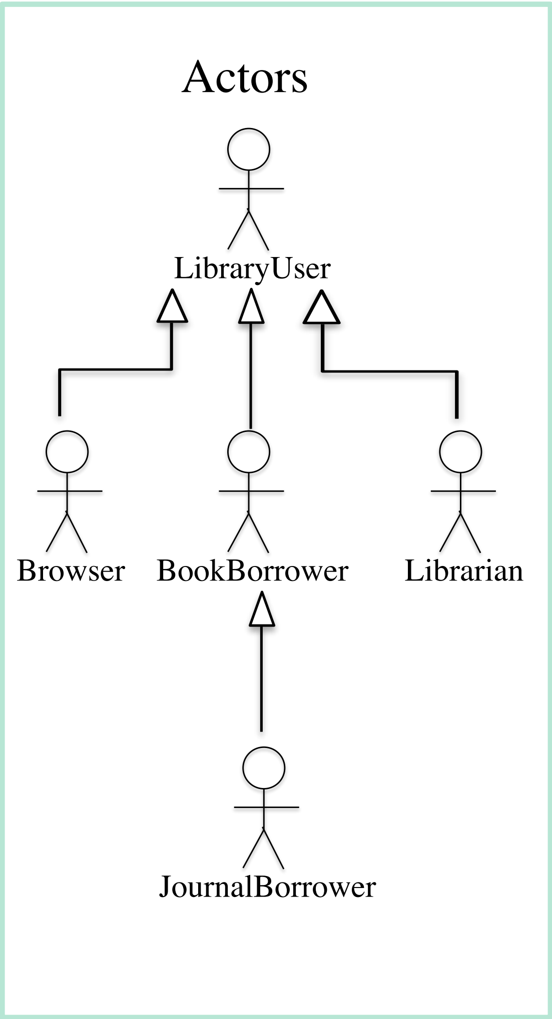
Possible Use Cases





Possible Use Cases- with system boundary



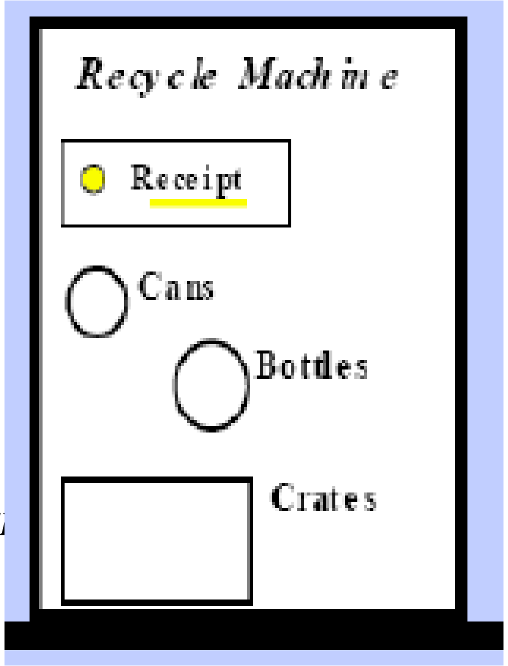


Recycle Machine Requirement Example

Multi-purpose recycling machine must:

- receive & check items for customers,
- print out receipt for items received,
- print total received items for operator,
- change system information,
- signal alarm when problems arise.

Reference: Anthony Finkelstein, UCL



Example: General Scenario

Counting returning items is started by Customer when they want to return cans, bottles or crates. With each item that the Customer places in the recycling machine, the system will increase the received number of items from the Customer as well as the daily total of this particular type.

When a Customer has deposited all their items, they will press a receipt button to get a receipt on which returned items have been printed, as well as the total return sum. Operator can change machine information and print reports of returned items.

Example: Instance Scenario

1. After the party Sarah goes to the recycling machine with three crates containing 5 cans and 3 bottles. Sarah deposits the cans and the bottles in the machine. Sara presses the “print receipt” button, the machine prints receipt containing number of bottles and cans and the total number of deposited items.

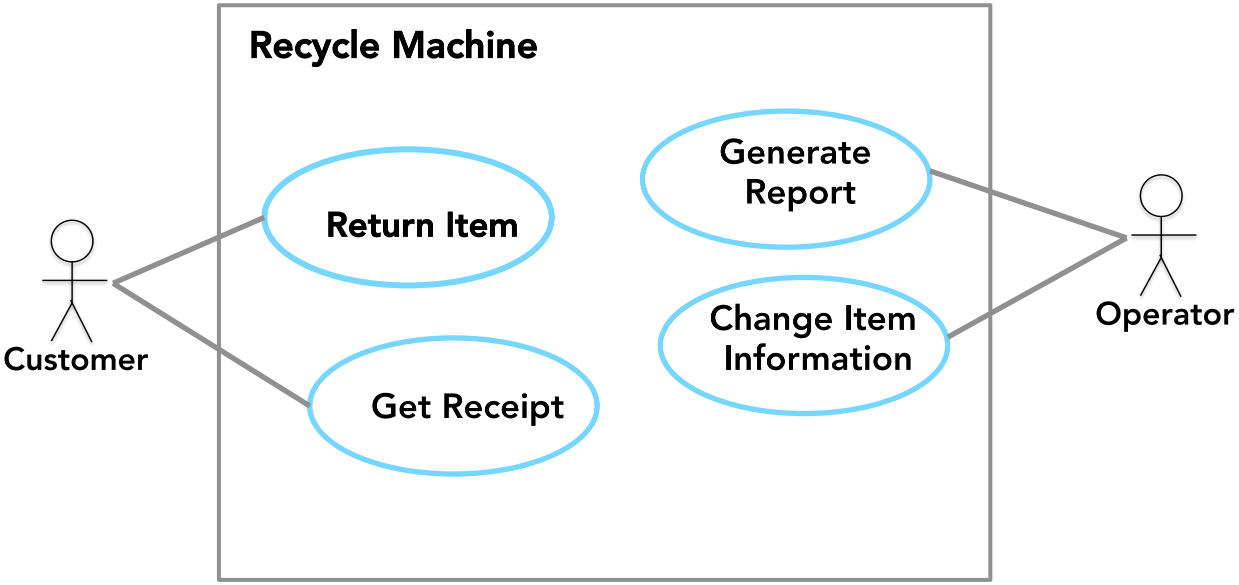
2. At the end of day, Adam, the operator, checks the recycling machine. Adams opens the machine with a key, and presses “print stats” button inside the machine. The machine print a report, that shows, the daily total of deposited items, and the grand total of deposited items for the current month, year and since its start.

Example: General Description

Counting returning items is started by Customer when they want to return cans, bottles or crates. With each item that the Customer places in the recycling machine, the system will increase the received number of items from the Customer as well as the daily total of this particular type.

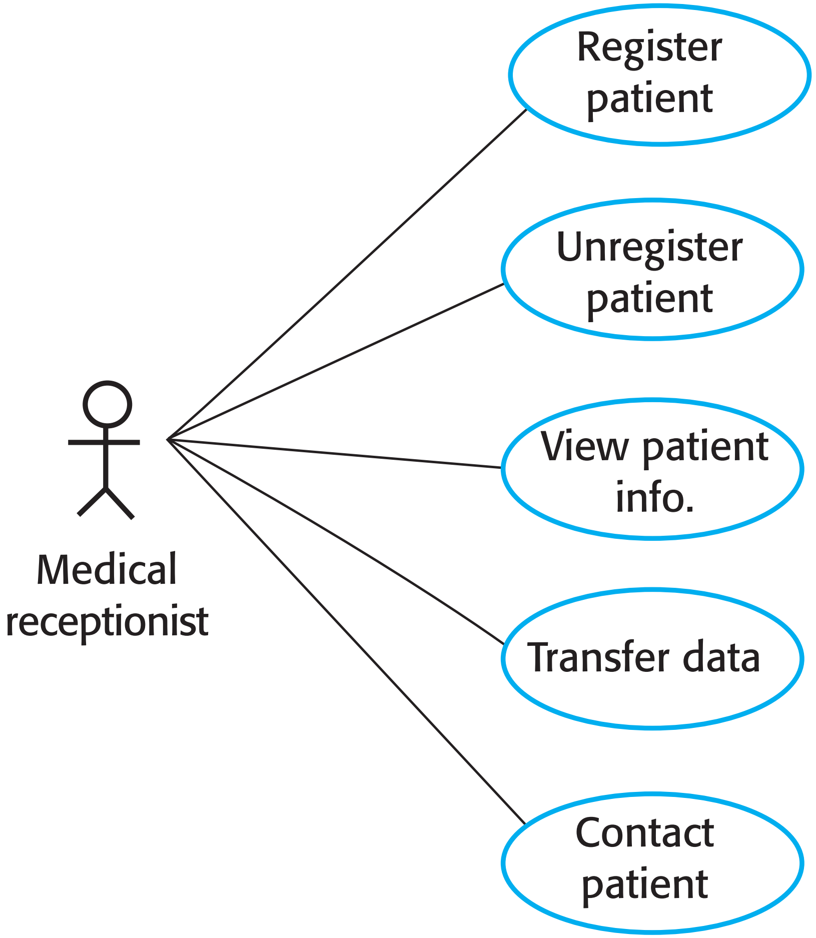
When a Customer has deposited all their items, they will press a receipt button to get a receipt on which returned items have been printed, as well as the total return sum. Operator can change item information and print reports of returned items.

Use case Diagram



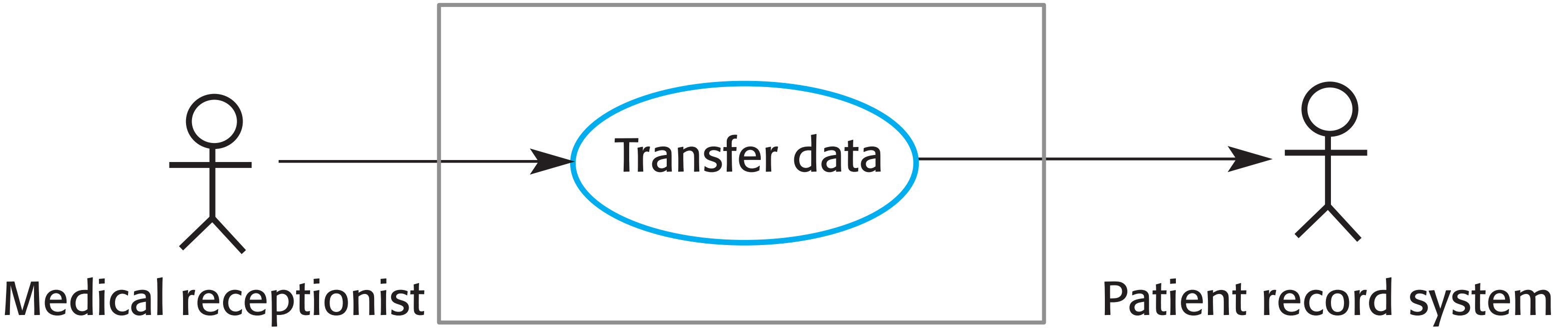
Reference: Anthony Finkelstein, UCL

Actor-based analysis -Use cases in the MHC-PMS Actor: 'Medical Receptionist'

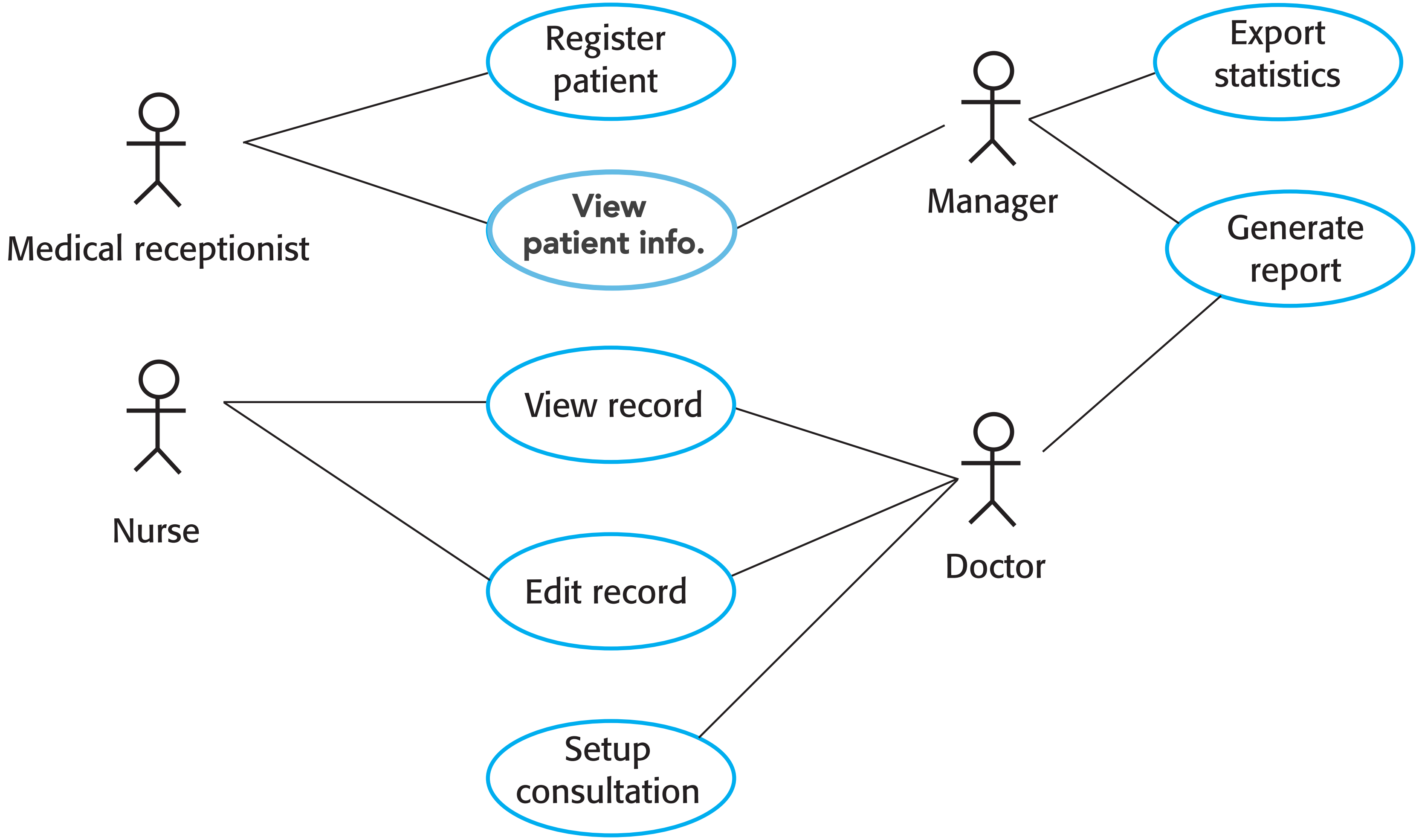


Example: PMS

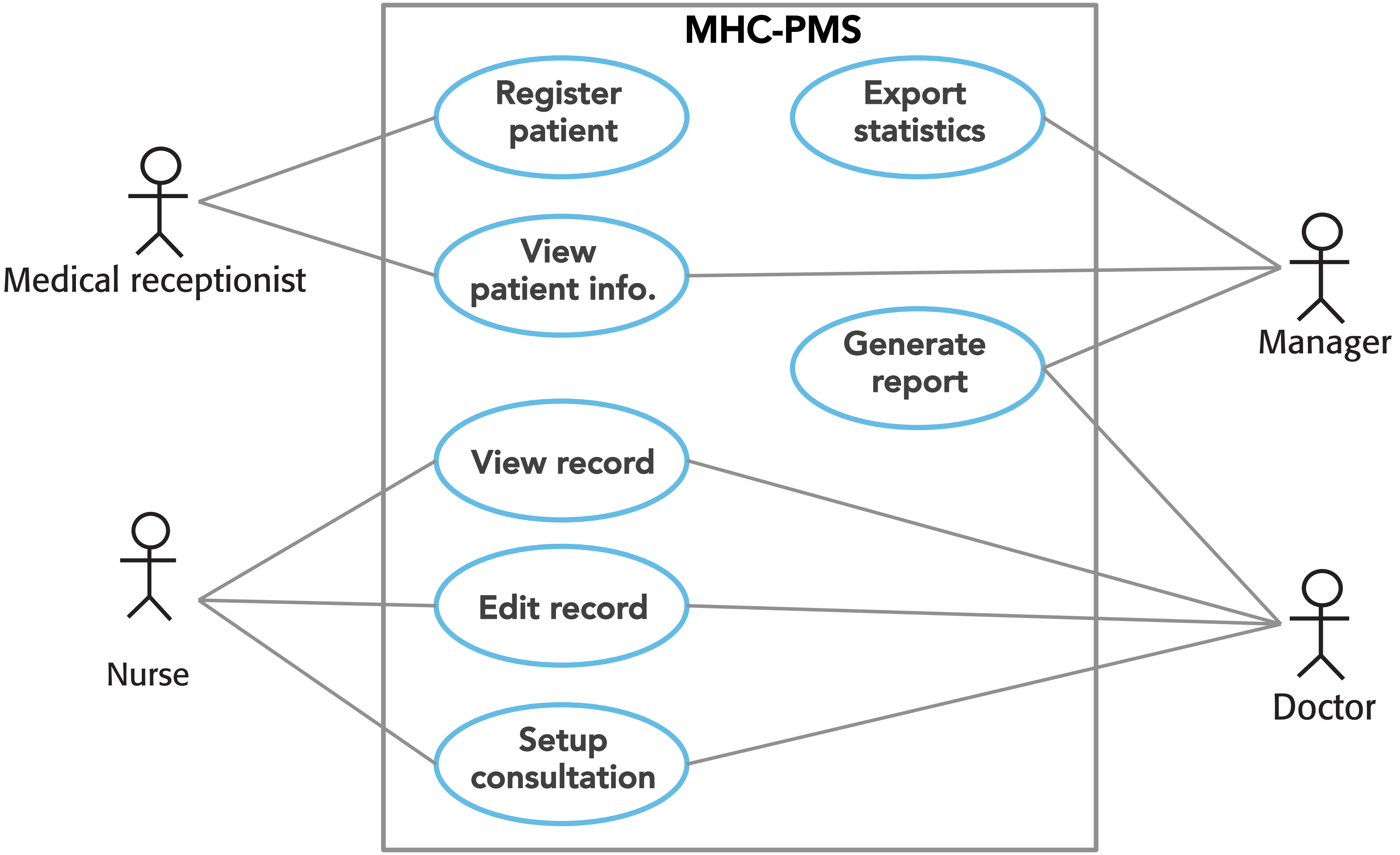
Transfer-data use case in the MHC-PMS



Use cases for the MHC-PMS- for different actors



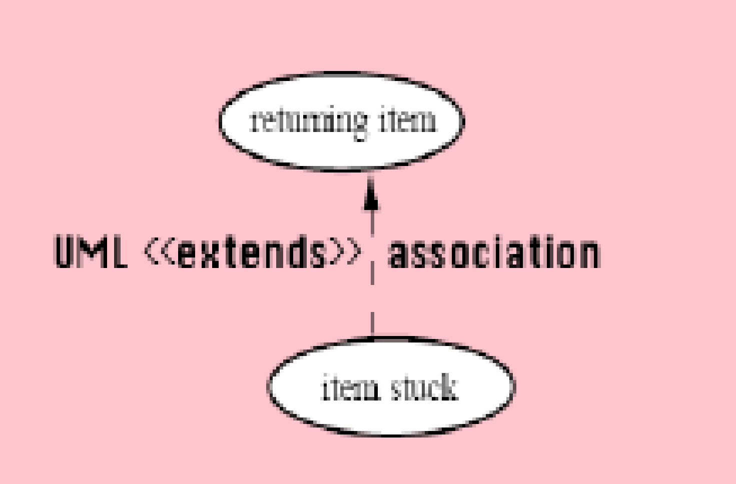
Use cases for the MHC-PMS-System Boundary



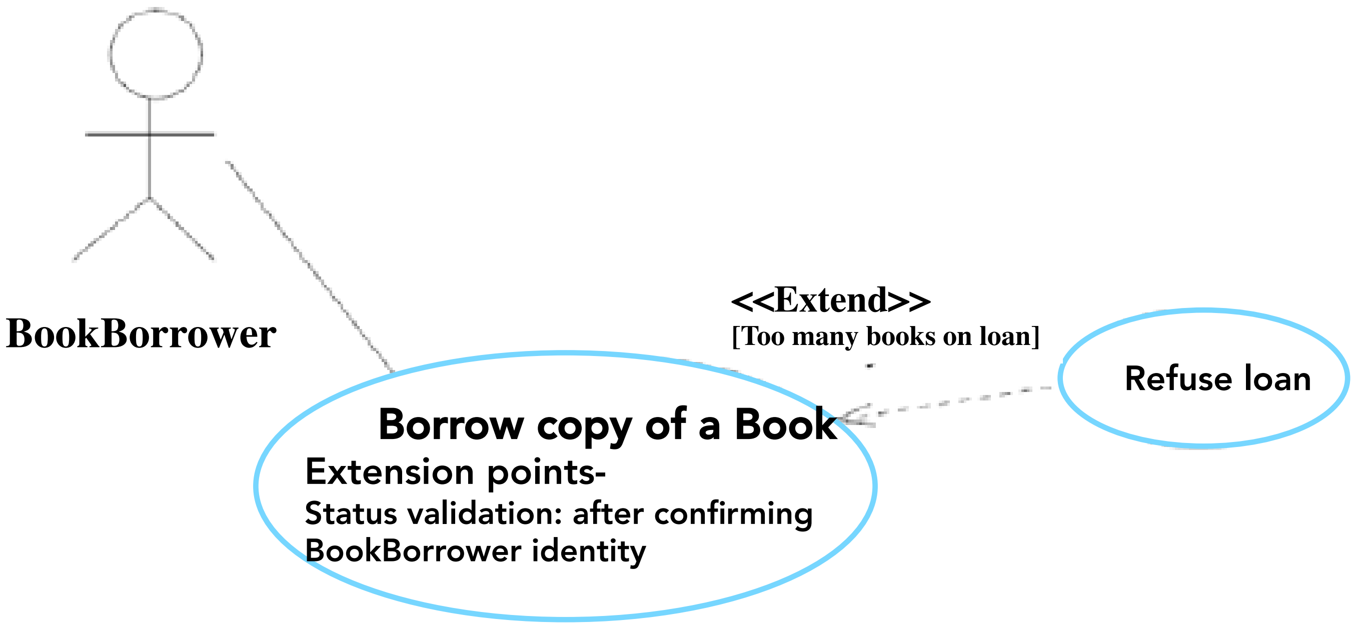
Extensions

Extensions provide opportunities for :

optional parts
alternative complex cases
separate sub-cases
insertion of use cases



Refinement - <<extend>>


BookBorrower
<<Extend>>
[Too many books on loan]
Refuse loan
Borrow copy of a Book
Extension points-
Status validation: after confirming BookBorrower identity

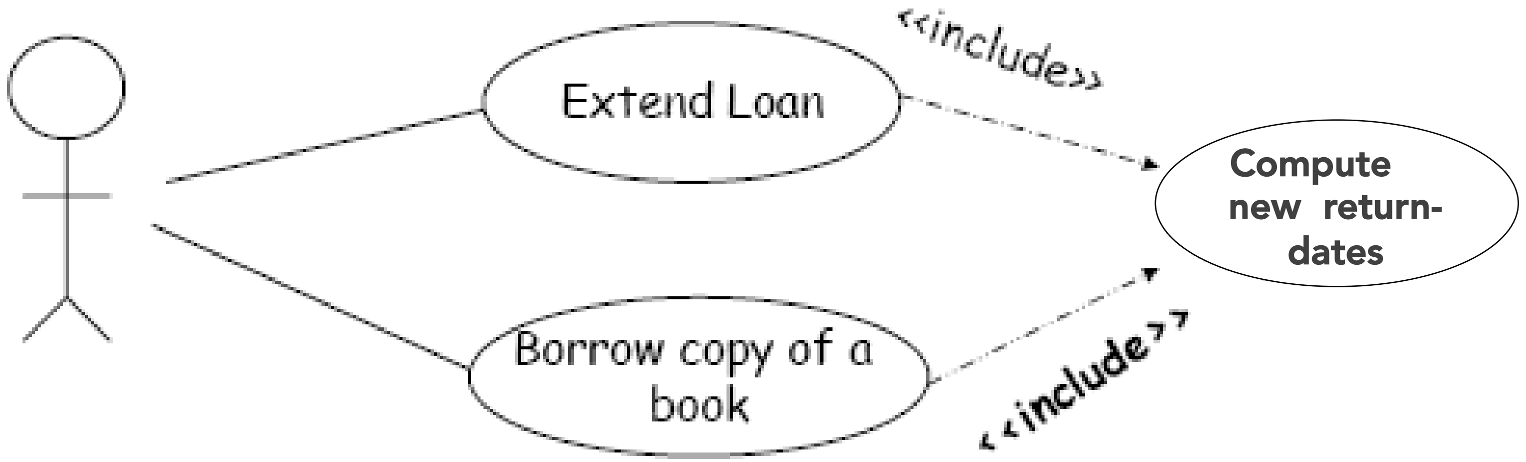
Use <<include>>

Use <<include>>

- How the system can reuse pre-existing component
- To show common functionality between use cases
- To develop the fact that project from existing components!

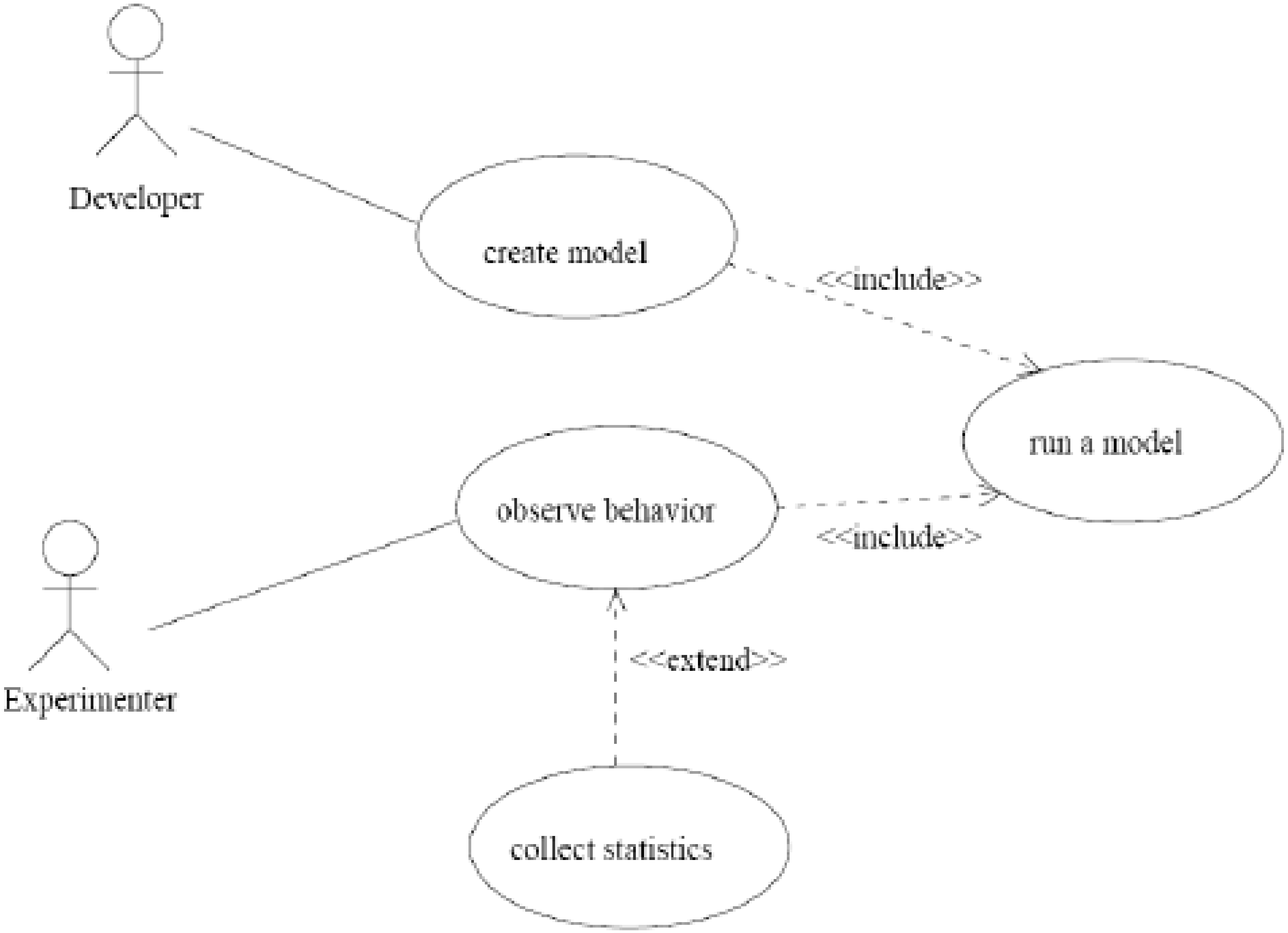
Note: <<include>> and <<extend>>: are UML stereotypes used to attach additional classification

Refinement - <<include>>



<<include>>

<<include>>


Developer
create model
<<include>>
run a model
observe behavior
<<include>>
Experimenter
<<extend>>
collect statistics

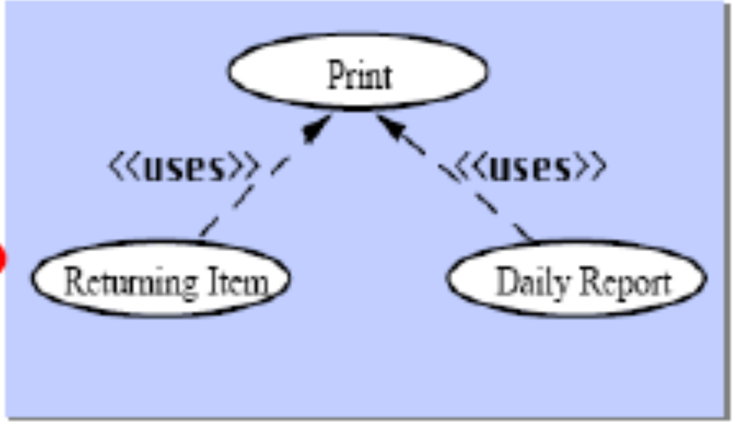
Refinement

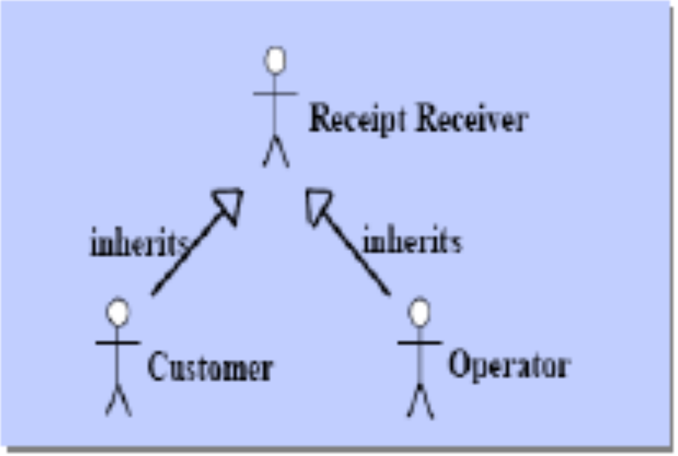
Abstract use case

Concrete use case

Abstract actors

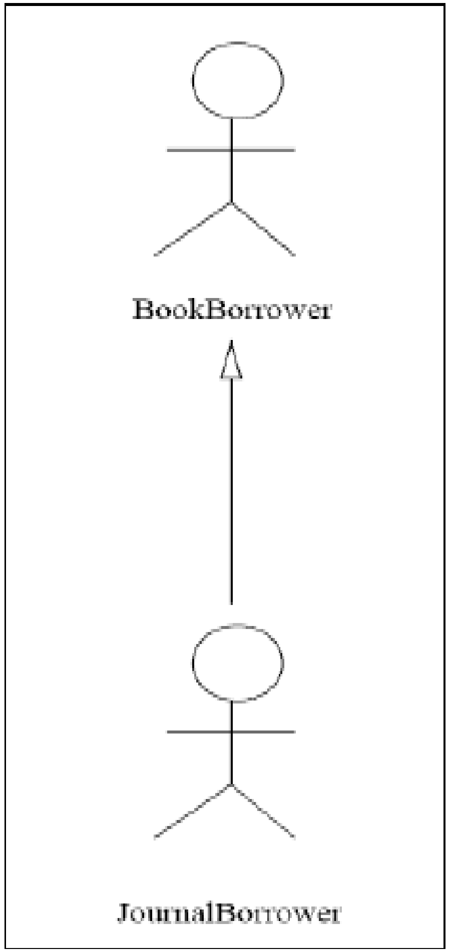
Concrete actors



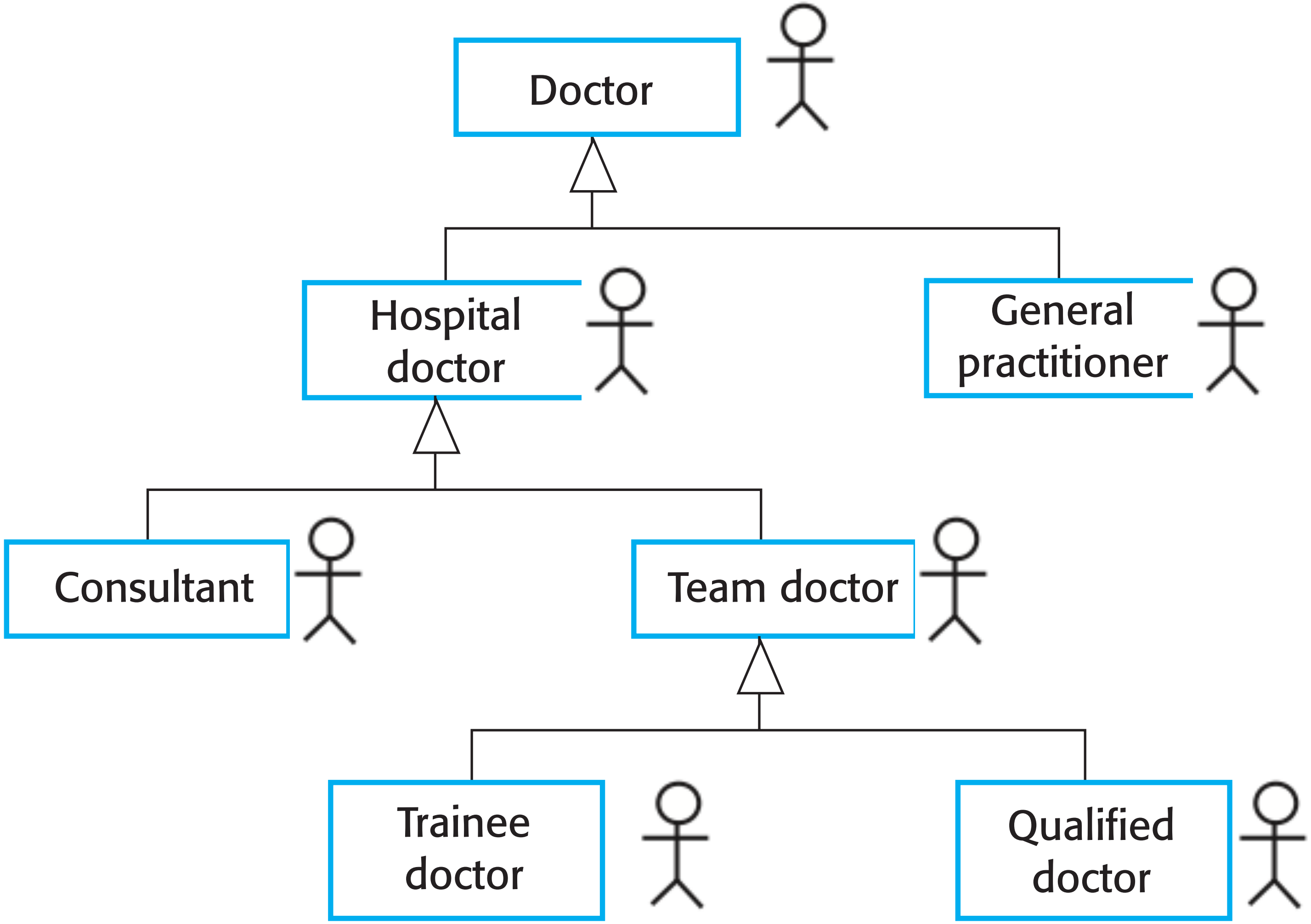


Generalization

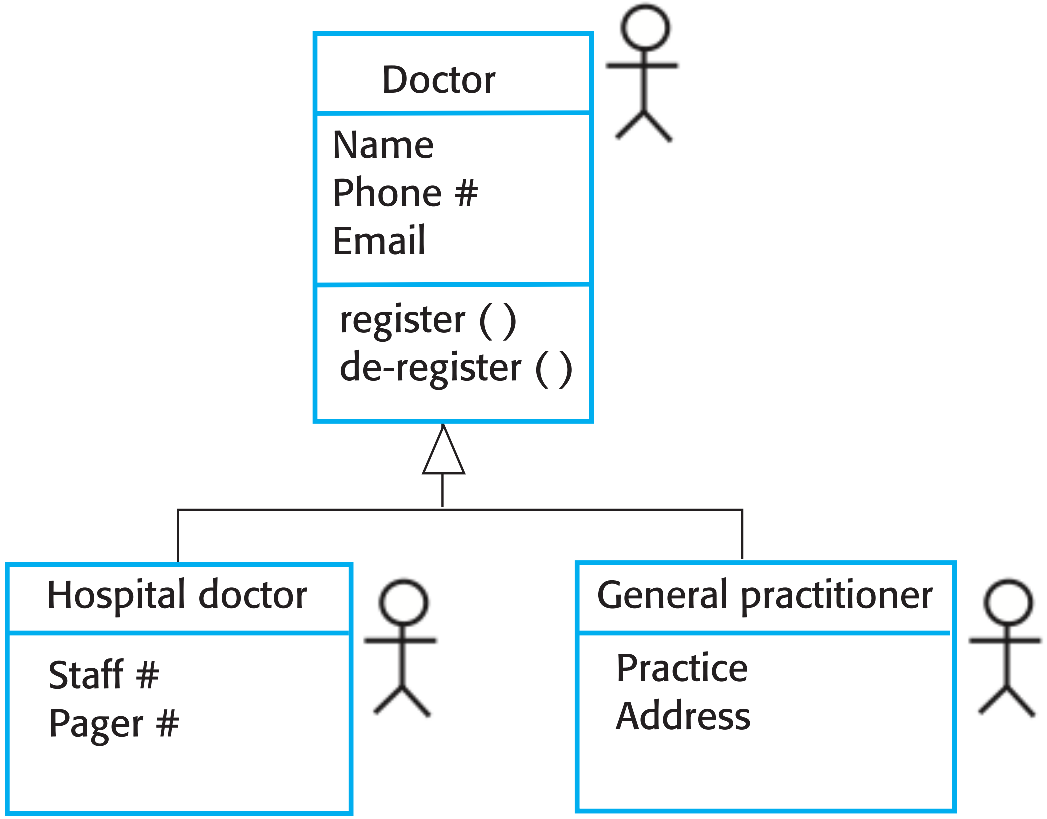
Journal borrower is a book borrower



A generalization hierarchy



A generalization hierarchy: Details



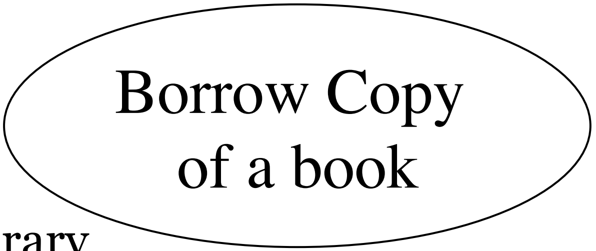
Detailing a use case

Writing a specification for the use case

Good Practice

- **Preconditions**: the system state before the case begin (i.e., facts, things that must be true)
- **Flow of events**; the steps in the use case (i.e. actions…)
- **Postconditions**: the system state after the case has been completed

Detailing a use case: Example


Borrow Copy of a book

Borrow copy of a book

Precondition

1. the BookBorrower is a member of the library
2. the BookBorrower has not got more than the permitted number of books on loan

Flow of events

1. the use case starts when the BookBorrower attempts to borrow a book
2. the librarian checks if it is ok to borrow a book
3. If Yes...... (Normal path of action)
   3.1...
   3.2...
   If No.... (an alternative path of action)
   3.1...
   3.2

Post-conditions

1. the system has updated the number of books the BookBorrower has on loan

Use Case Description: Example 1

Library System: Borrow Copy of a Book	
Actors	BookBorrower, Librarian
Description	A BookBorrower may borrow a copy of a book from the library. A book must exist in the library and available to borrow and will be issued by Librarian. The status of the copy of the book will change to <on-loan> and the loan period of the copy book will be decided by the type of the book: ShortLoan: 2 days, MediumLoan: 2 weeks, LongLoan: 3 months.
Pre-conditions	1. the BookBorrower is a member of the library 2. the BookBorrower has not already borrowed more than the permitted number of books on loan
Sequence/Flow of Events	1. the BookBorrower asks librarian to borrow a book 2. the librarian checks if BookBorrow is allowed to borrow a book 3. If Yes……, if No…. (indicates an alternative path of action)
Data	Book information, Borrow information, Book status information
Stimulus/Trigger	User command issued by Librarian on behalf of BookBorrower
Post-conditions/ Response	1. the system has updated the number of books the BookBorrower has on loan, if successful 2. Copy of book loan status updated – to <on-loan>, if successful
Comments	The librarian must have appropriate security permissions to access BookBorrow information

Use Case Description: Example 2

Medical System: TRANSFER PATIENT DATA	
Actors	Medical receptionist, patient records system (PRS)
Description	A receptionist may transfer data from the MHC-PMS to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient's diagnosis and treatment.
Pre-conditions	1. Patient is a member of the clinic 2. Patient information is access-able
Sequence/Flow of Events	1. the Medical receptionist select patient records to transfer 2. Medical receptionist transfers selected patient records to authority 3. If successful……, if not successful…. (indicates an alternative path of action)
Data	Patient's personal information, treatment summary
Stimulus/Trigger	User command issued by medical receptionist
Post-conditions/ Response	Confirmation that PRS has been updated
Comments	The receptionist must have appropriate security permissions to access the patient information and the PRS.

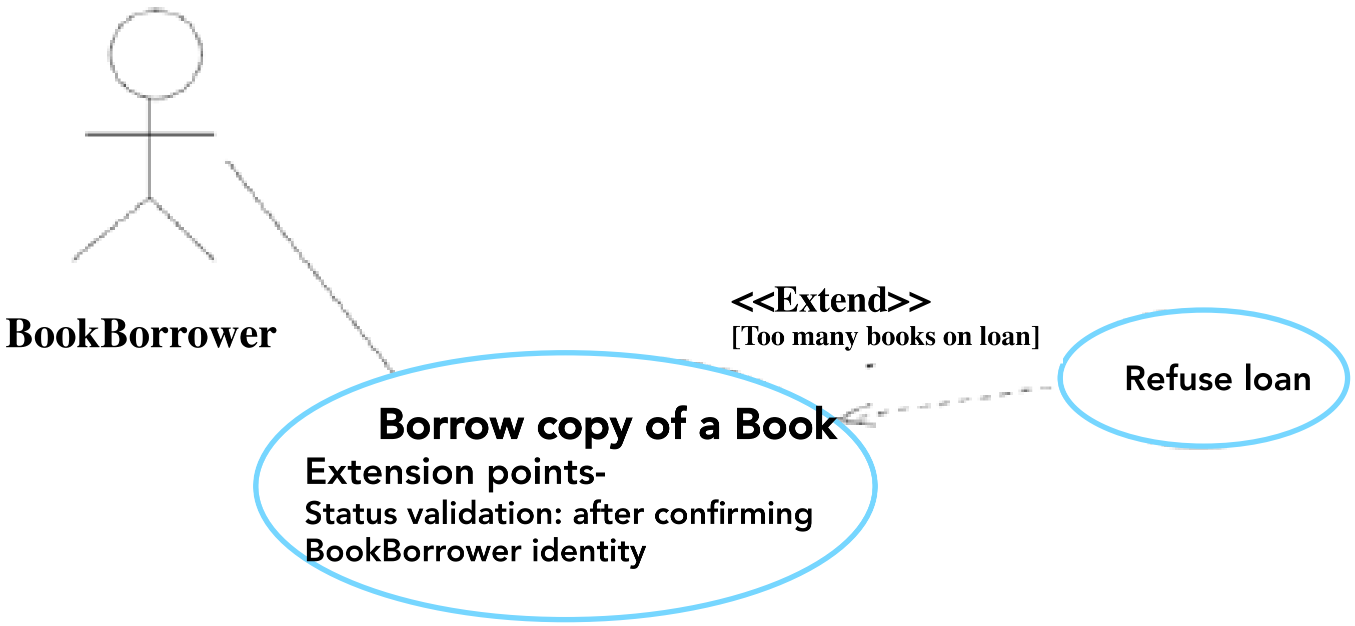
Scenarios

Each time an actor interacts with a system, the triggered use cases instantiate a scenario

Each case corresponds to a specific path through a use case with no branching

Scenarios are typically documented as text along side the use case and activity diagrams

Write the scenarios for this Use Case diagram



Example: Borrow copy of a book

Scenario 1

BookBorrower Joe borrows the library's only copy of "Using UML", when he has no other book on loan. The system is updated accordingly.

Scenario 2

BookBorrower Ann tries to borrow the library's second copy of "Software Engineering", but is refused because she has six books out on loan, which is her maximum allowance.

Scenario Example: Borrow copy of a book

Normal (~successful outcome)

BookBorrower Joe borrows from the library a copy of “Using UML”. Joe has no other books on loan, takes the copy of the book to the the librarian, who checks Joe’ allowance, scans the copy’s barcode and issues the book to Joe. The system is updated accordingly.

Alternative (~successful outcome)

BookBorrower Joe borrows from the library a copy of “Using UML”. Joe has no other books on loan, Joe takes the copy of the book to auto-librarian, auto-librarian scans Joe’s library ID and the barcode on the copy of the book. It checks Joe’s borrowing allowance, and it automatically issues the book to Joe. The system is updated accordingly.

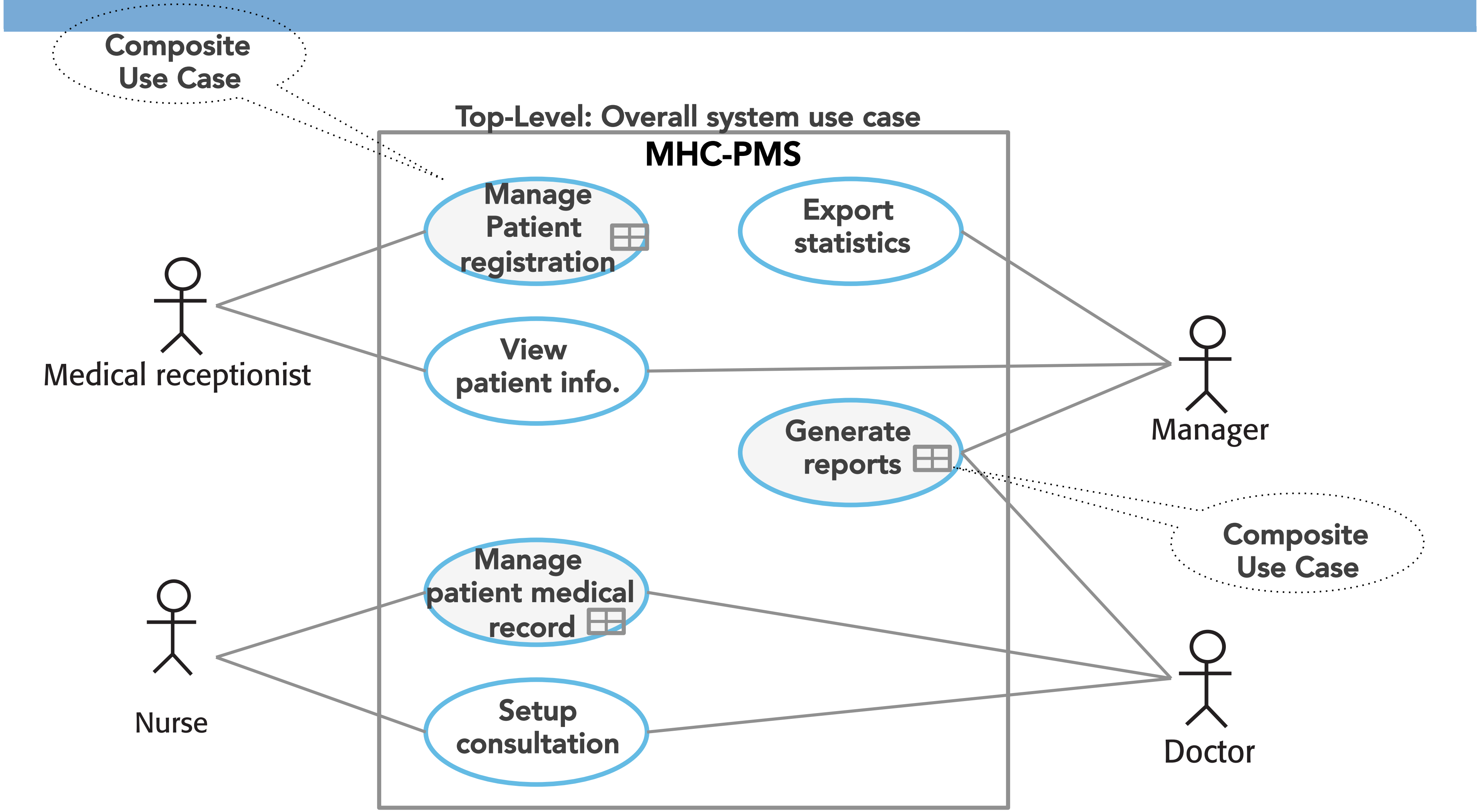
Error (~unsuccessful outcome)

BookBorrower Joe borrows from the library a copy of “Using UML”. Joe takes the book to the the librarian, who checks Joe’ allowance, scans the copy’s barcode, but Joe is refused because he has six books out on loan, which is his maximum allowance.

Error (~unsuccessful outcome)

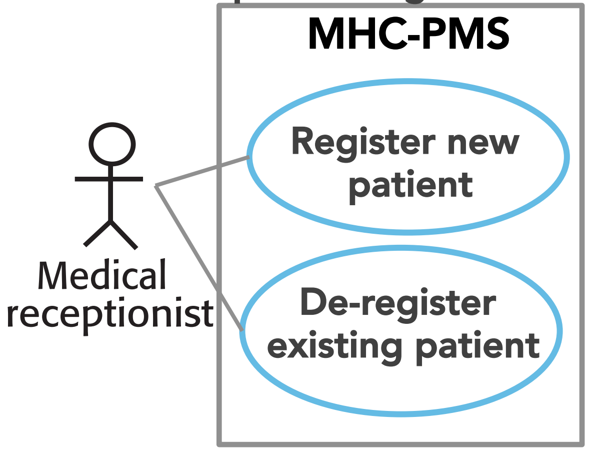
BookBorrower Joe borrows from the library a copy of “Using UML”. Joe takes the book to the the librarian, who checks Joe’ allowance, scans the copy’s barcode, but barcode is damaged, the copy was not issued.

Multi-level use cases

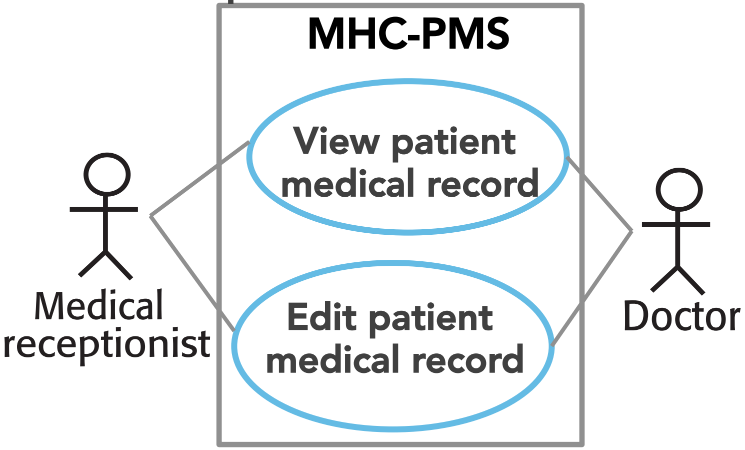


Multi-level Use cases

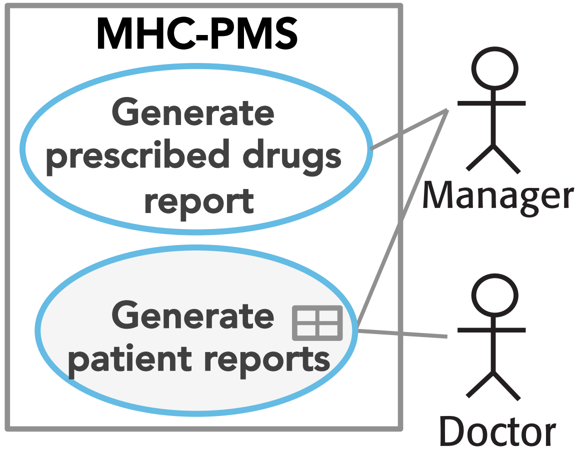
1st Level; use case: Manage patient registration



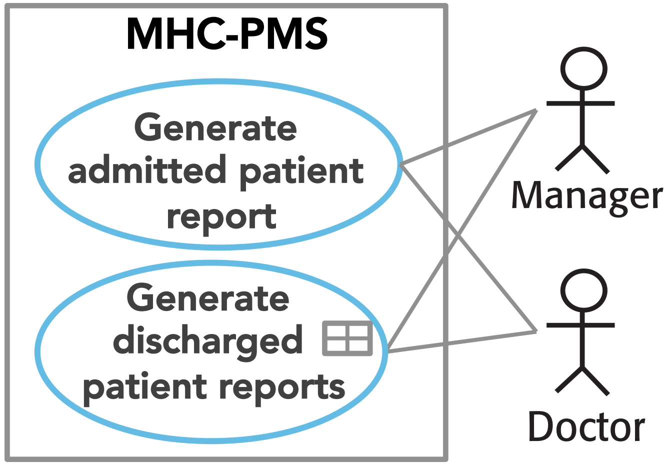
1st Level; use case: Manage patient medical record



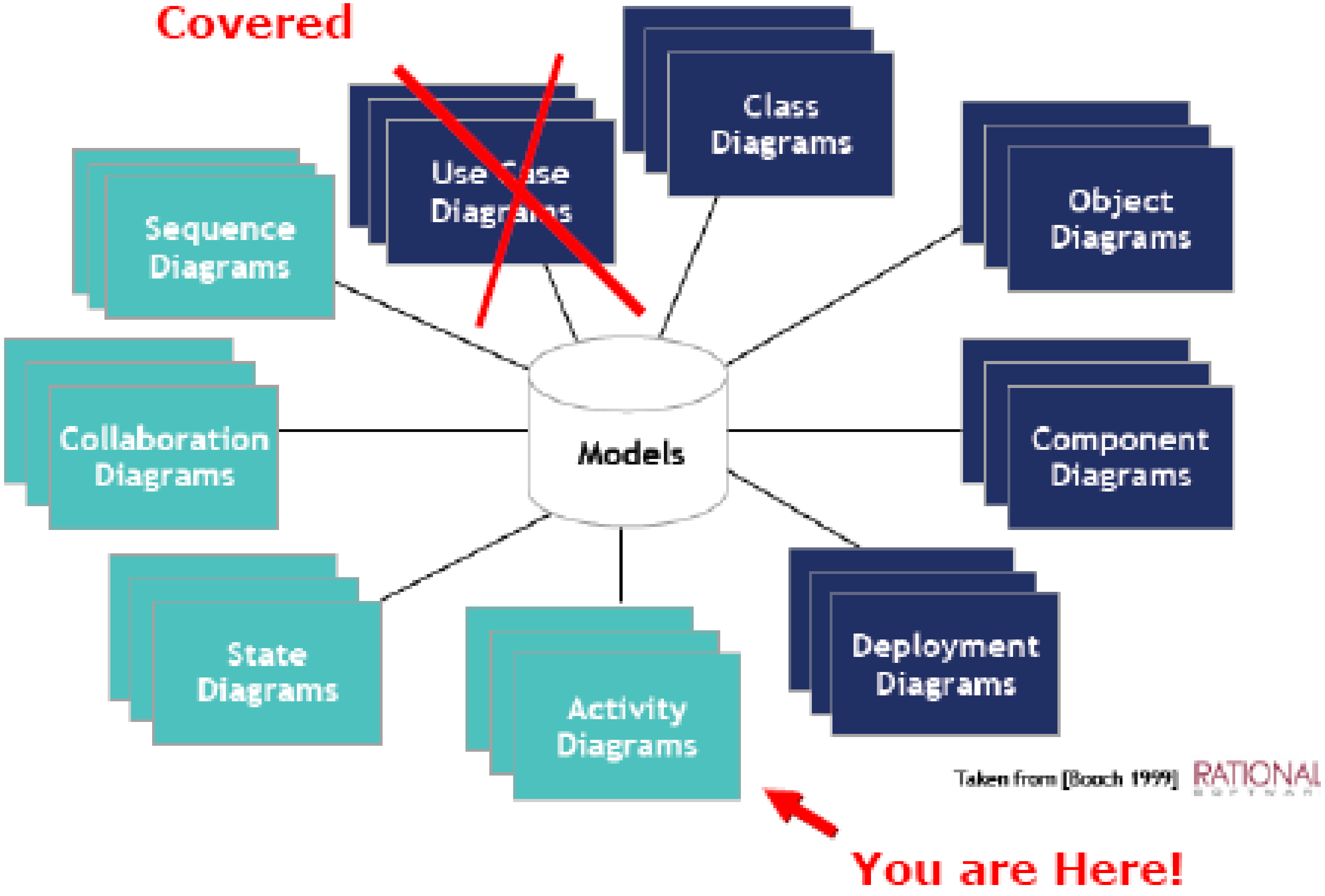
1st Level; use case: Generate reports



2nd Level; use case: Generate patient reports


MHC-PMS
Generate admitted patient report
Generate discharged patient reports
Manager
Doctor

UML Diagrams



Activity Diagram

Activity diagrams helps to represent Workflows and business processes

They model the **behaviours** (activities) of the system

They show the dependencies and coordination between activities within a system

the activity flow should not get “stuck”

they can be used during the requirements elicitation process …

-to help identify how use cases interact to achieve business processes

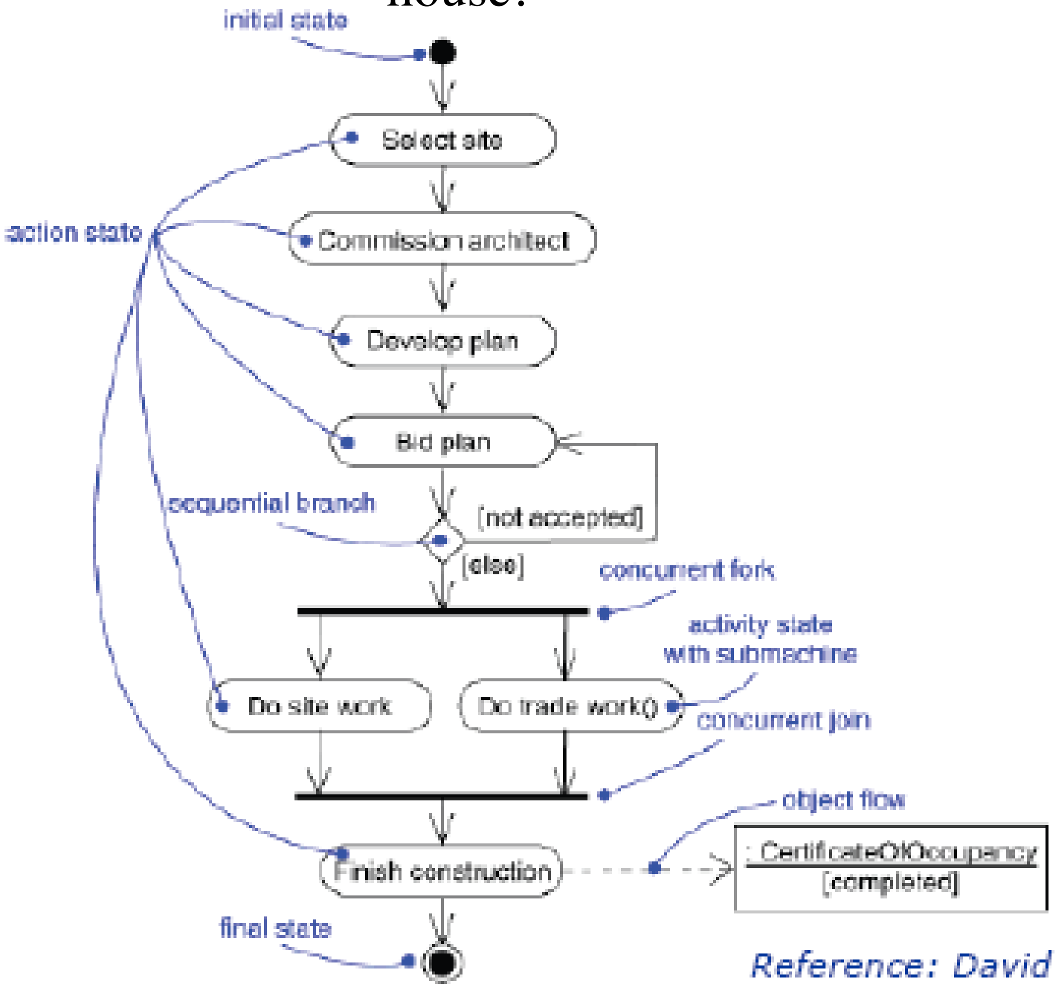
-to help in identifying use cases of a system and operations involved in the realization of a use case

But, generally, they can be attached to any model element to model its **dynamic behaviour**

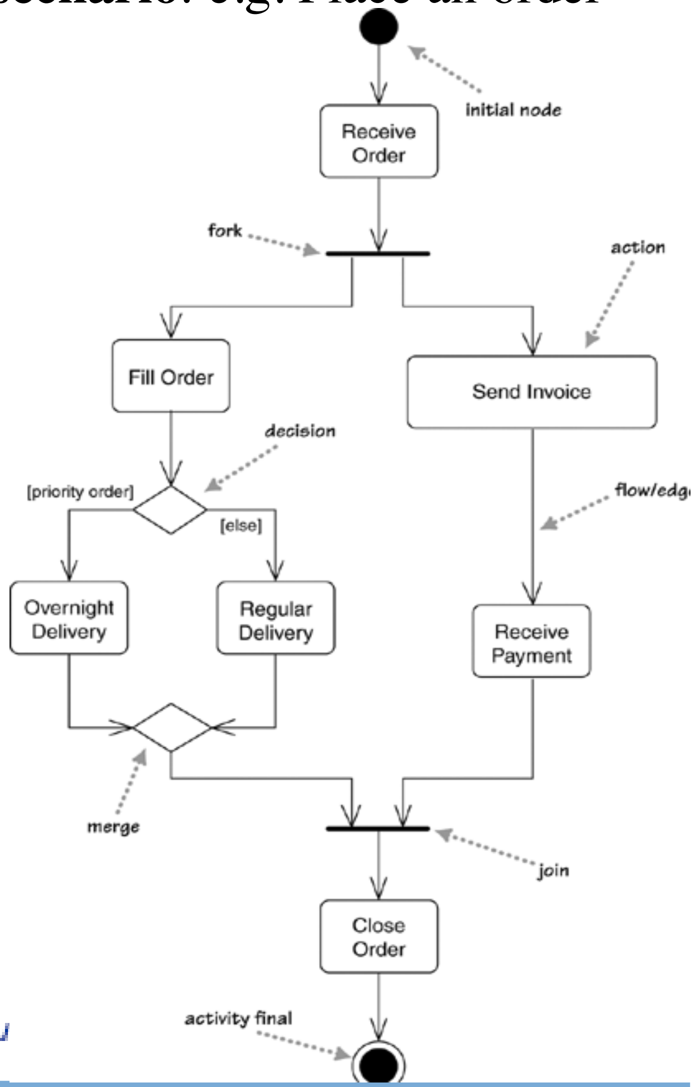
Activity Diagram: Example

Capture the **Overall Process**: e.g. Build a house!

Business **scenario**: e.g. Place an order



Reference: David Rosenblu

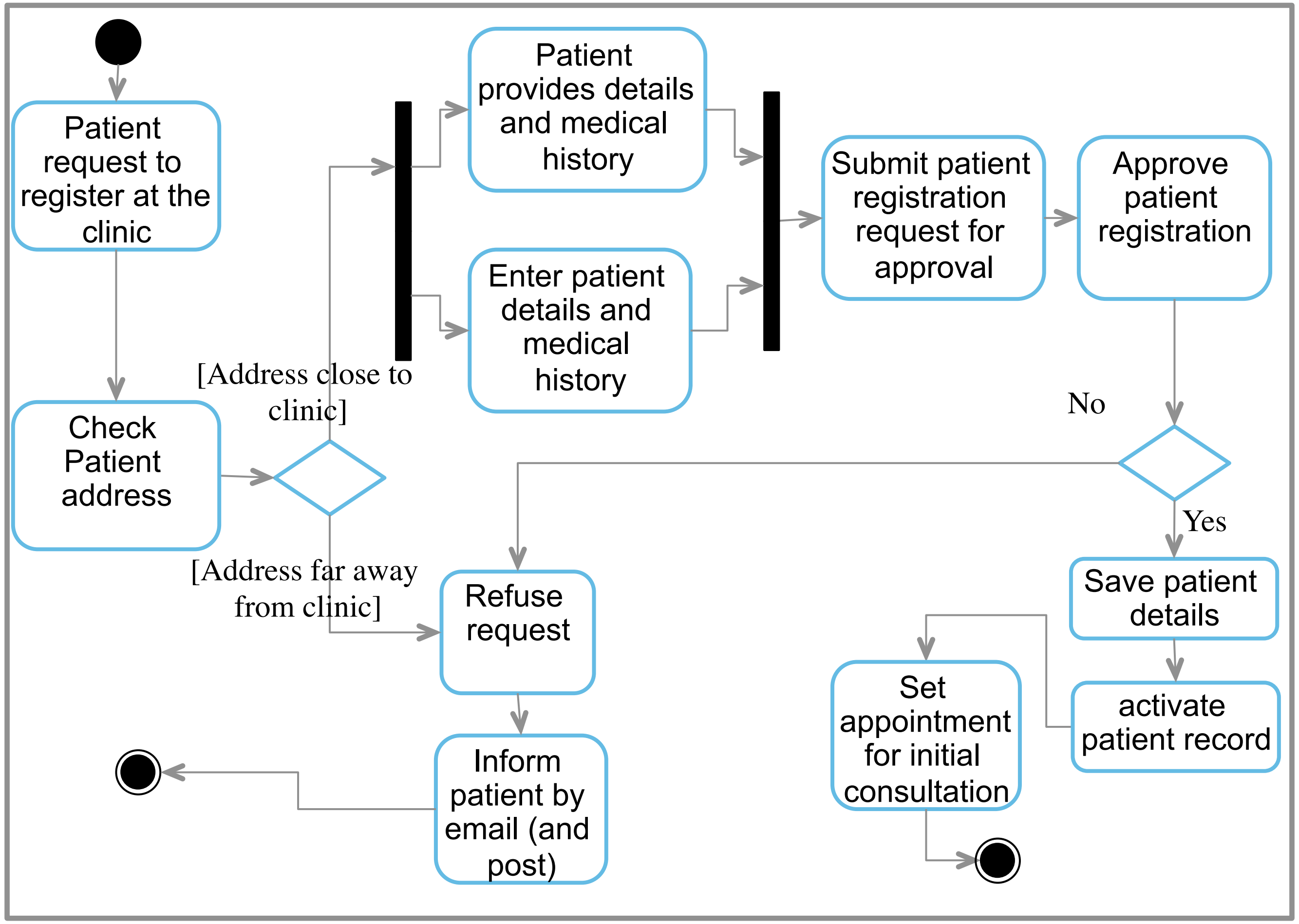


Activity Diagram: Examples

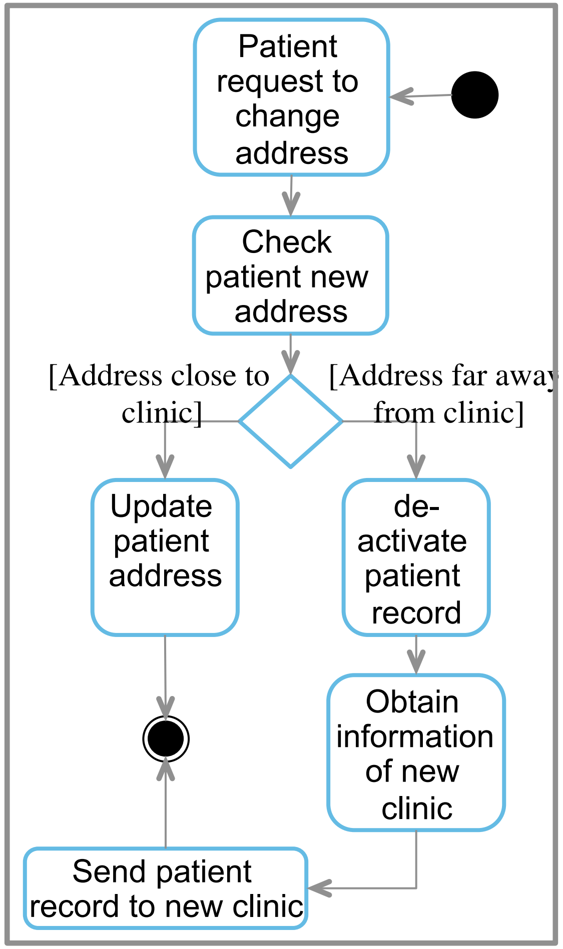
Capture behaviour of use cases

use case: register patient – **business process view**

use case: Change patient address (or de-register patient)



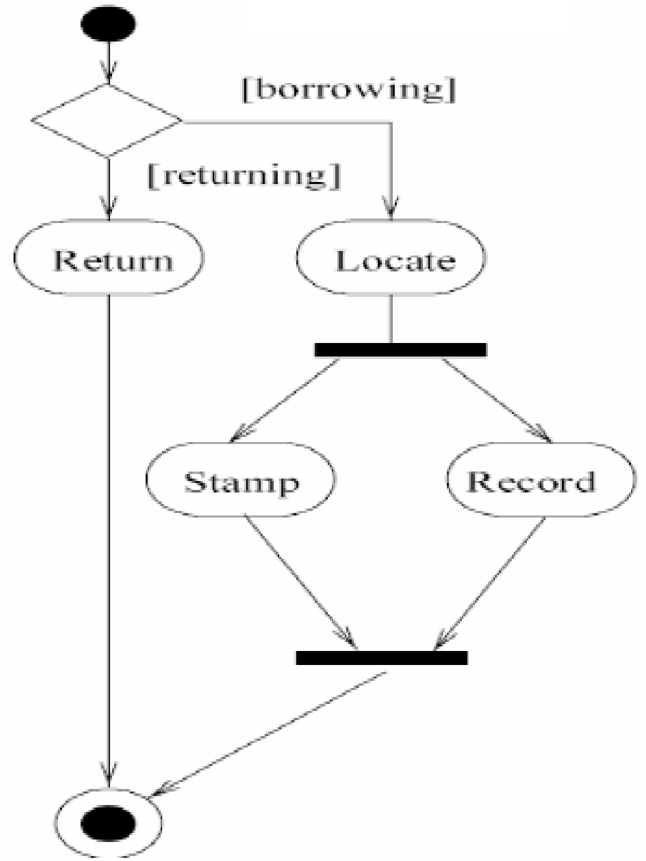
Patient request to register at the clinic
Check Patient address
[Address close to clinic]
[Address far away from clinic]
Patient provides details and medical history
Enter patient details and medical history
Submit patient registration request for approval
Approve patient registration
No
Yes
Refuse request
Inform patient by email (and post)
Save patient details
activate patient record
Set appointment for initial consultation



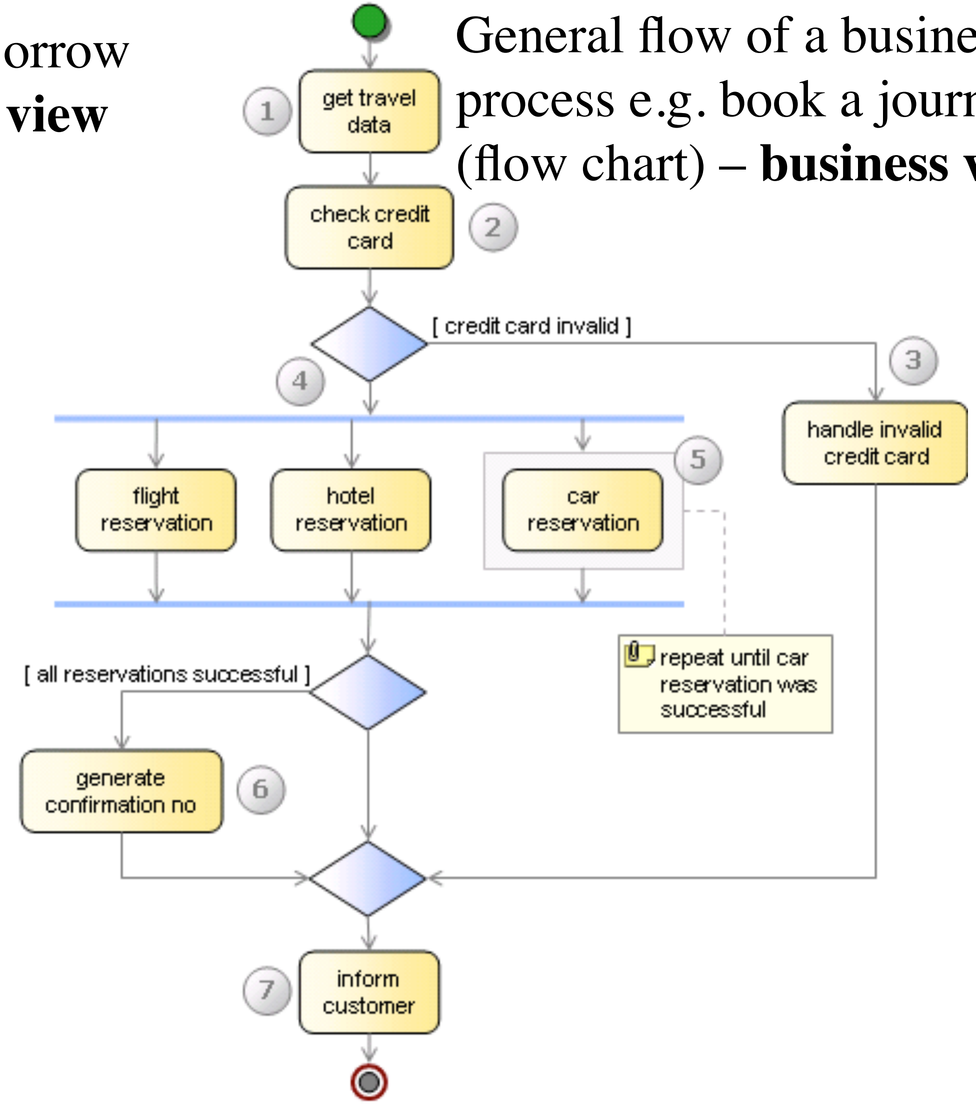
Patient request to change address
Check patient new address
[Address close to clinic]
[Address far away from clinic]
Update patient address
de-activate patient record
Obtain information of new clinic
Send patient record to new clinic

Activity Diagram: Examples

Specific flow of a use case: e.g. “Borrow copy of a book”-**business process view**

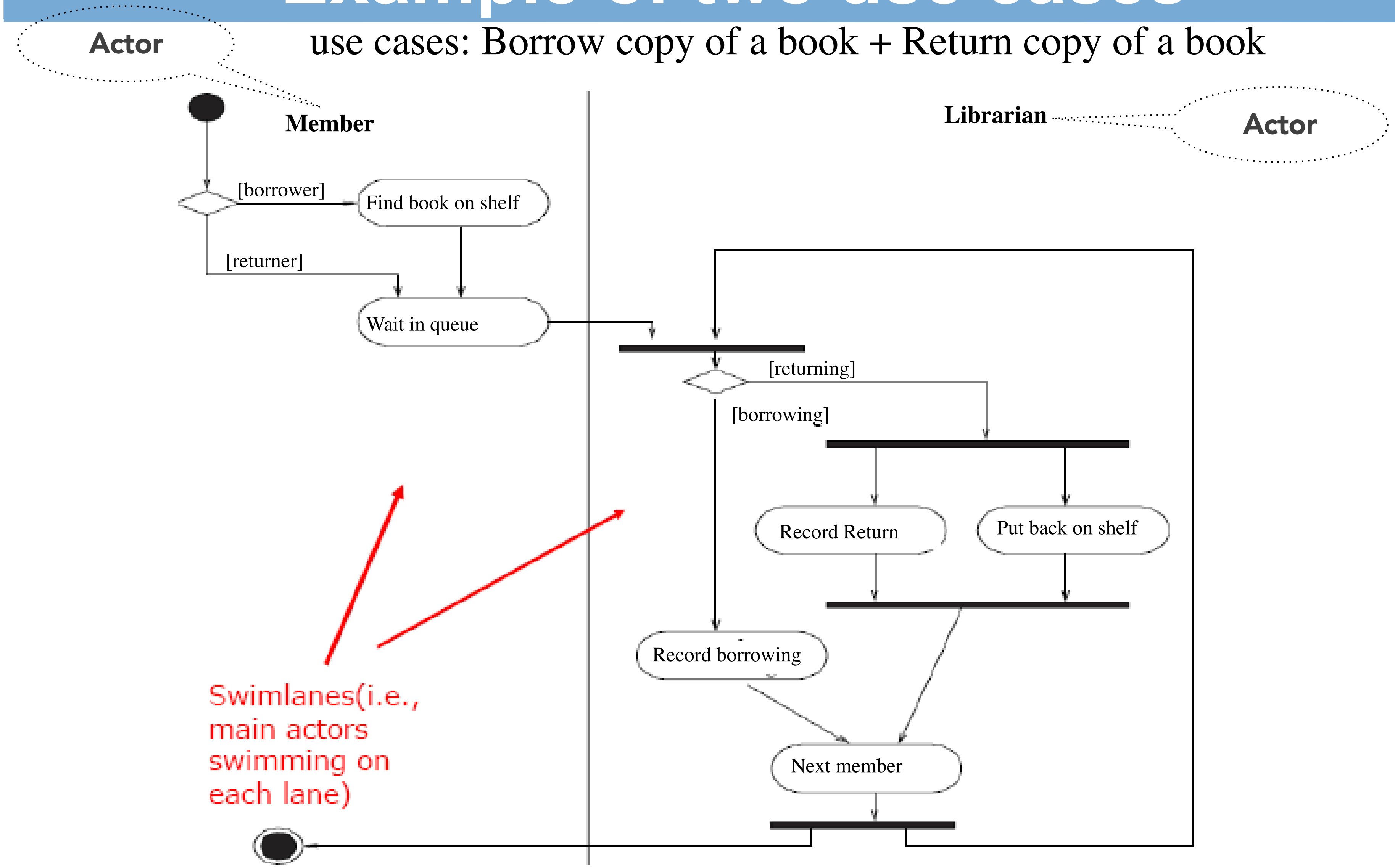


General flow of a business process e.g. book a journey (flow chart) – **business view**



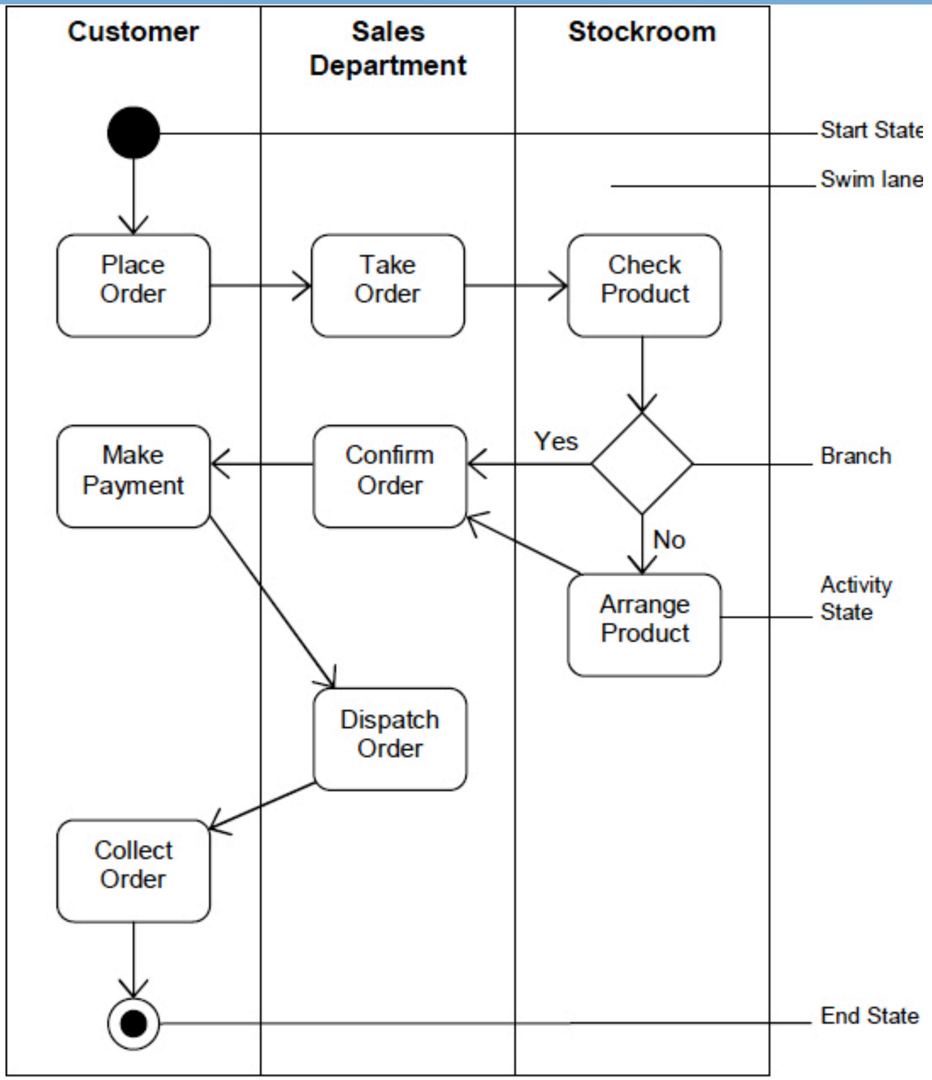
Activity Diagram-swim lanes: Example of two use cases

use cases: Borrow copy of a book + Return copy of a book



Activity Diagram-swim lanes : Example

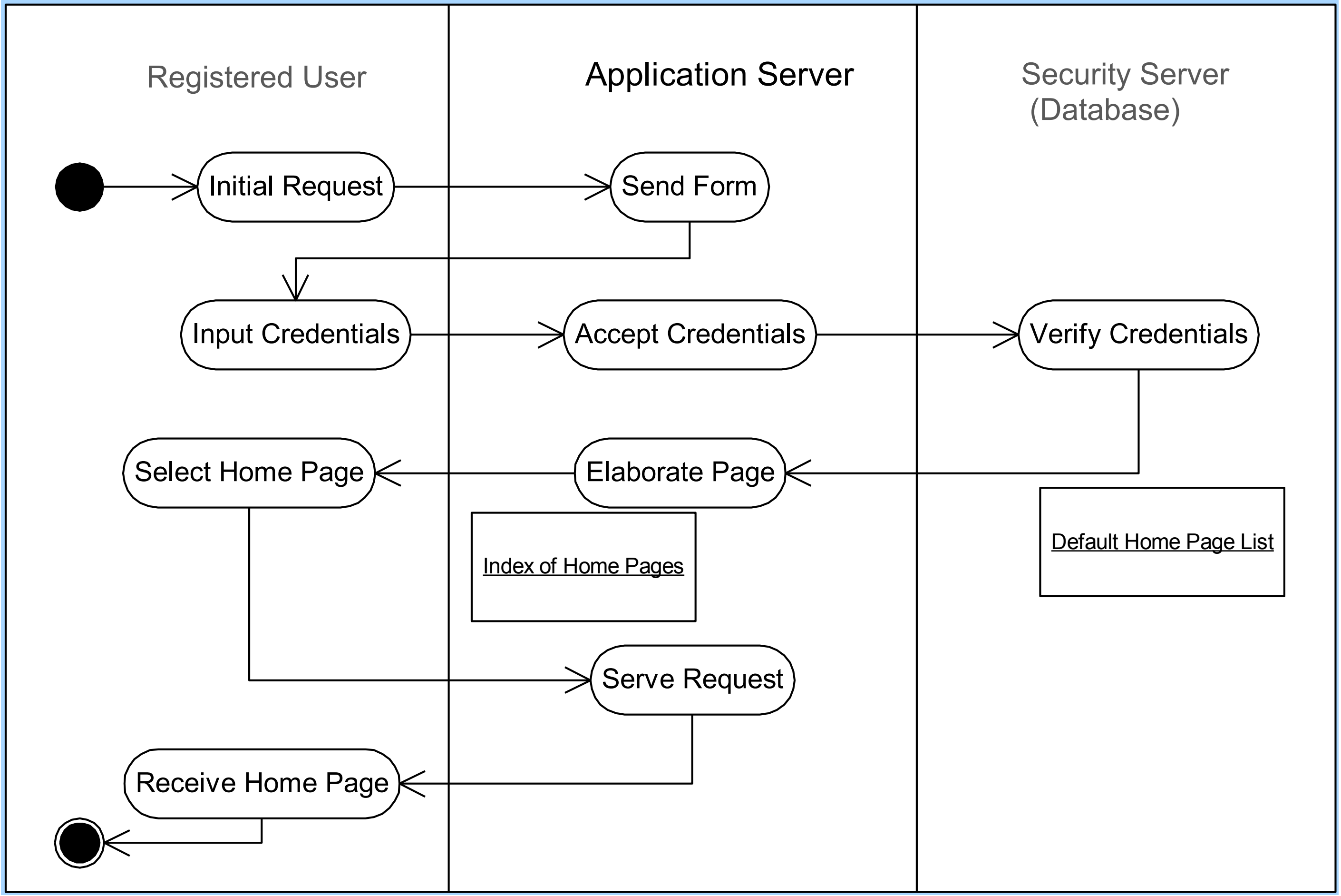
General Scenario or business process: e.g. Sell a product – **Business view**



Activity Diagram for Product Sale

Activity Diagram: swim lanes- Example

Use case: login to a company website (**successful normal scenario**):
System view


Registered User
Application Server
Security Server (Database)
Initial Request
Send Form
Input Credentials
Accept Credentials
Verify Credentials
Select Home Page
Elaborate Page
Index of Home Pages
Default Home Page List
Serve Request
Receive Home Page