Introduction to Software Engineering Activity Diagram

Lecture 6

TOPICS COVERED

- Business Processes Vs. Business Rules
  - Business Rules representation techniques
  - Business Processes representation techniques
- Activity Diagrams
  - Advantages
  - Elements & Symbols
  - Examples
  - Linking activities to user interfaces
- Use-Case Models
  - Advantages
  - Elements & Symbols
  - Descriptions
  - System Boundary
  - Scenarios
  - Generalization Among Actors
  - Relationships between Use-Cases
    - «Include» .. Sharing behaviour
    - «Extend» Alternatives to main scenarios
    - Extending descriptions
  - Avoiding over-complex use case diagrams
  - Use-Case as a planning aid
  - Use-Case .. & Architecture
  - Use-Case .. & Testing

While rules generally focus on ‘what’, processes are commonly oriented towards ‘how’.

Business rules describe for example what needs to be done, what criteria must be applied and what policies must be enforced.

Business rules therefore focus on the value that needs to be created in/through a process.

Processes on the other hand describe in what order (how) activities should be executed to create the desired value or outcome. Processes versus rules

BUSINESS PROCESSES VS. BUSINESS RULES

Business processes define what is done in a business, by whom, in what order, needing which resources, and with what consequences.

Business rules constrain how a business is run.

- There is a clear distinction between the business processes and the constraints.
- Example, in a car rental company:
  - renting a car (business process);
  - the car allocated is the lowest mileage car that is available in the chosen group (business rule);

BUSINESS PROCESSES VS. BUSINESS RULES

Business processes are important to:

-get an understanding of what a business does,

-and to gain the domain expertise needed to develop software solutions in the business context.

Business rules need to be identified and recorded:

1. to provide quick and easy access whenever there is a change.

2. Business rules are the basis of decision making.

Traceability of business rules should be possible from the business needs to the software solution.

BUSINESS RULES REPRESENTATION TECHNIQUES

The language used in expressing business rules should be well defined and structured, so that verification can be made easier if not automatic:

1. Using natural language to make it easy for business people to understand it (.. *quite informal*).

2. Using UML and OCL;

OCL is **O**bject **C**onstraint **L**anguage, which is a formal language used to represent constraints in UML.

BUSINESS PROCESSES REPRESENTATION TECHNIQUES

Business processes are represented using **UML activity diagrams**.

- An activity diagram shows a process as a set of activities, showing their sequences, and where activities can be carried out in parallel.

- It can also be extended to show which person is responsible for which activity.

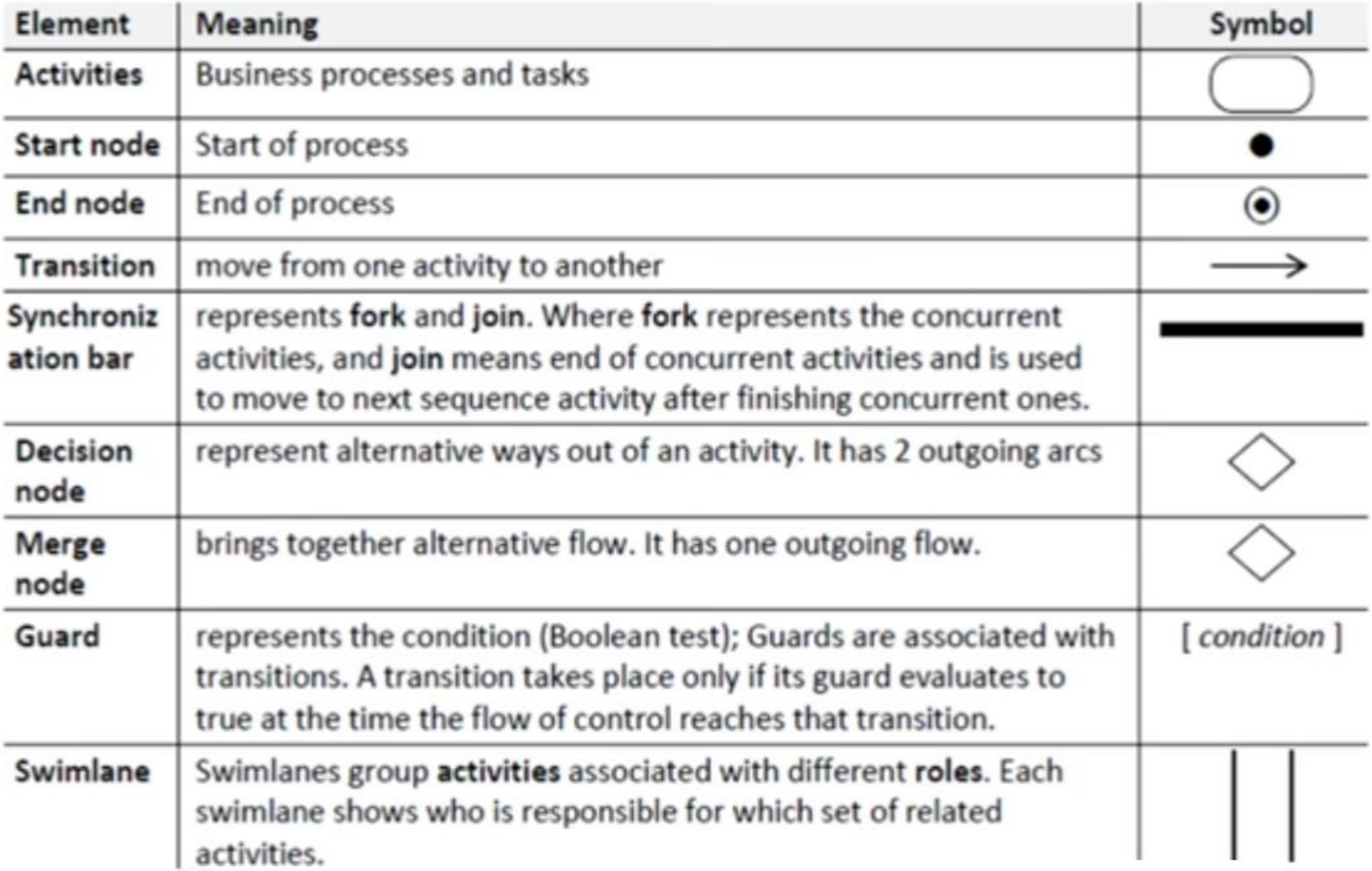
- An activity could be a task that a person or a computer might perform.

ACTIVITY DIAGRAMS .. ADVANTAGES

1. Help investigate the **workflow*** (flow of control) from one activity to another.
2. Help in understanding the basic behavior of a system, and understand the business situation.
3. Can be used to model concurrent systems.
4. Can record scenarios of use cases.
5. Can help identify the stages at which each role requires some interaction with the process. This is of particular benefit when we want to investigate the steps that people take in order to do their jobs.

***Workflow** is defined as a sequence of activities that produces a result of observable value.

ACTIVITY DIAGRAMS .. ELEMENTS & SYMBOLS



Element	Meaning	Symbol
Activities	Business processes and tasks	▢
Start node	Start of process	●
End node	End of process	◉
Transition	move from one activity to another	→
Synchronization bar	represents **fork** and **join**. Where **fork** represents the concurrent activities, and **join** means end of concurrent activities and is used to move to next sequence activity after finishing concurrent ones.	▬
Decision node	represent alternative ways out of an activity. It has 2 outgoing arcs	◇
Merge node	brings together alternative flow. It has one outgoing flow.	◇
Guard	represents the condition (Boolean test); Guards are associated with transitions. A transition takes place only if its guard evaluates to true at the time the flow of control reaches that transition.	[*condition*]
Swimlane	Swimlanes group **activities** associated with different **roles**. Each swimlane shows who is responsible for which set of related activities.	| |

ACTIVITY DIAGRAMS .. EXAMPLE

Consider the lending process from a library :

"A typical lending library keeps a stock of books for the use of its members. Each member can look on the library shelves to select a copy to borrow. Then take out a number of books, up to a certain limit. After a given period of time, the library expects members to return the books that they have on loan.

When borrowing books, a member is expected to wait in queue, then to hand their chosen books to the librarian, who records each new loan before issuing the books to the member. After that the librarian will prepare for next member in the queue. When a book is on loan to a member, it is associated with that member: possession of the book passes from the library to the member for a defined period. The normal loan period for each book is two weeks. If the member fails to bring the book back on or before the due date, the library imposes a fine."

ACTIVITY DIAGRAMS .. EXAMPLE

What are the business processes and business rules in the above system?

ACTIVITY DIAGRAMS .. EXAMPLE

What are the business processes and business rules in the above system?

Business processes: find book on a shelf; wait in the queue; issue a book, and return a book.

Business rules:

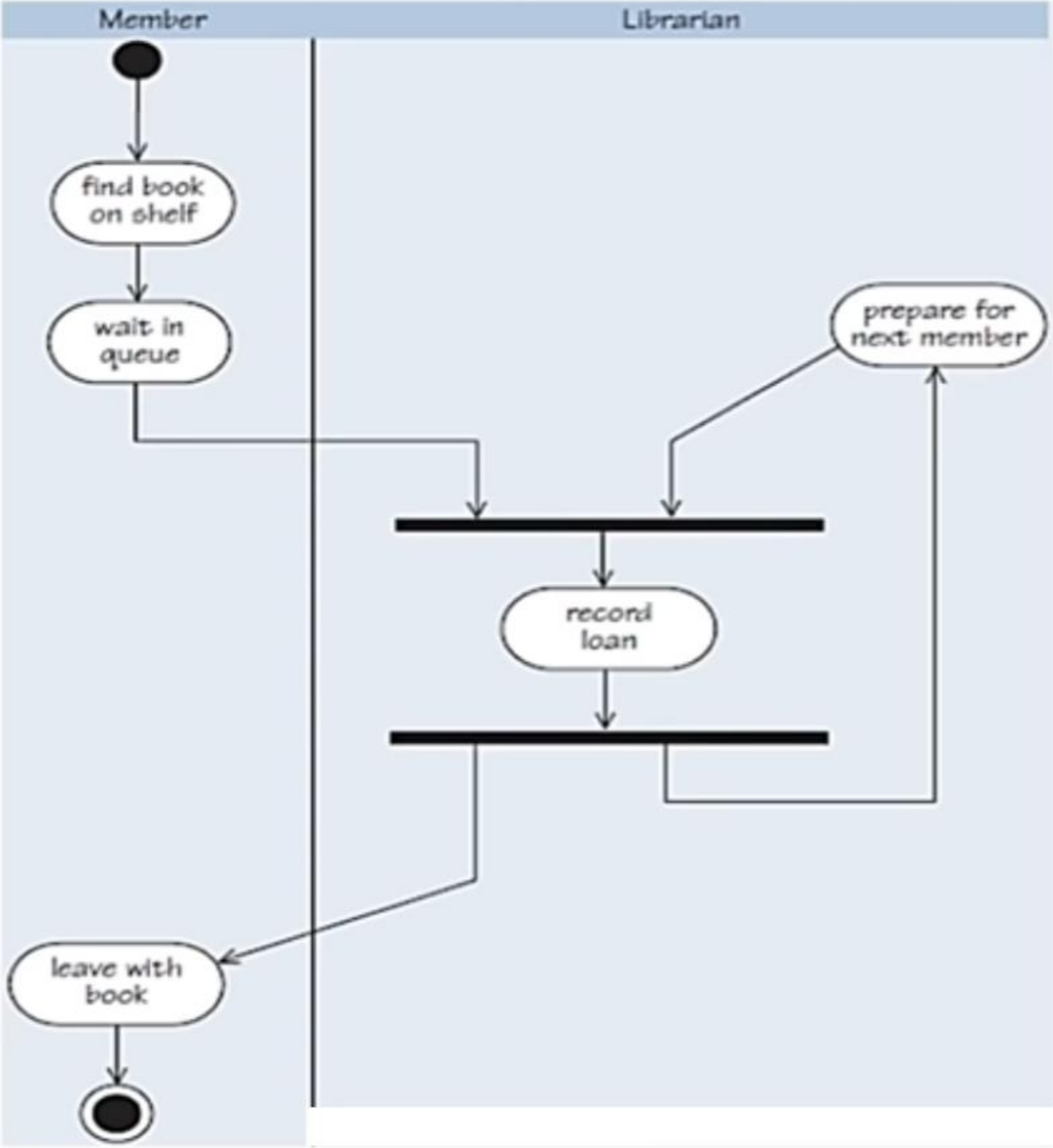
- There is a limit to the number of books a member can take out.

- A loan is for a period that is normally two weeks.

- Late returns incur a fine.

Draw an activity diagram for issuing a copy of a book.

ACTIVITY DIAGRAMS .. EXAMPLE



ACTIVITY DIAGRAMS .. SWIM-LANES

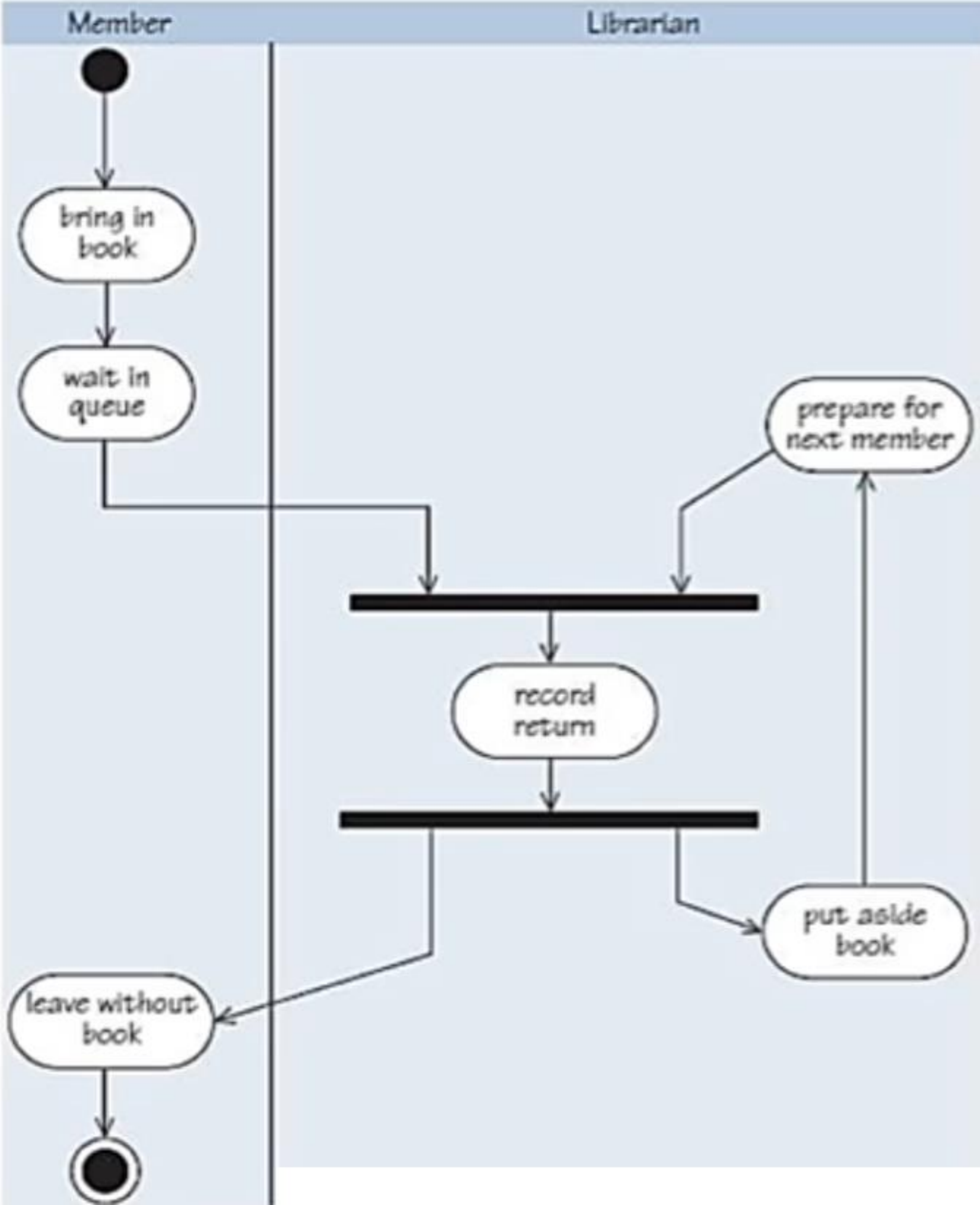
Swim-lanes group activities associated with different roles.

- The swim-lanes show the role that is responsible for each activity.
- Activity diagrams represent the sequence of activities.
- When you are modelling a workflow that involves more than one role, it is possible to identify which role is responsible for a particular activity.
- An activity diagram can help identify the stages at which each role requires some interaction with the process.

In example 5 The left-hand swim-lane contains the activities that a library member performs when borrowing a book, while the right-hand swim-lane shows the library's self-service system workflow for the same process ('issue copy of book').

ACTIVITY DIAGRAMS .. EXAMPLE

An activity diagram of returning a copy of a book to the library.

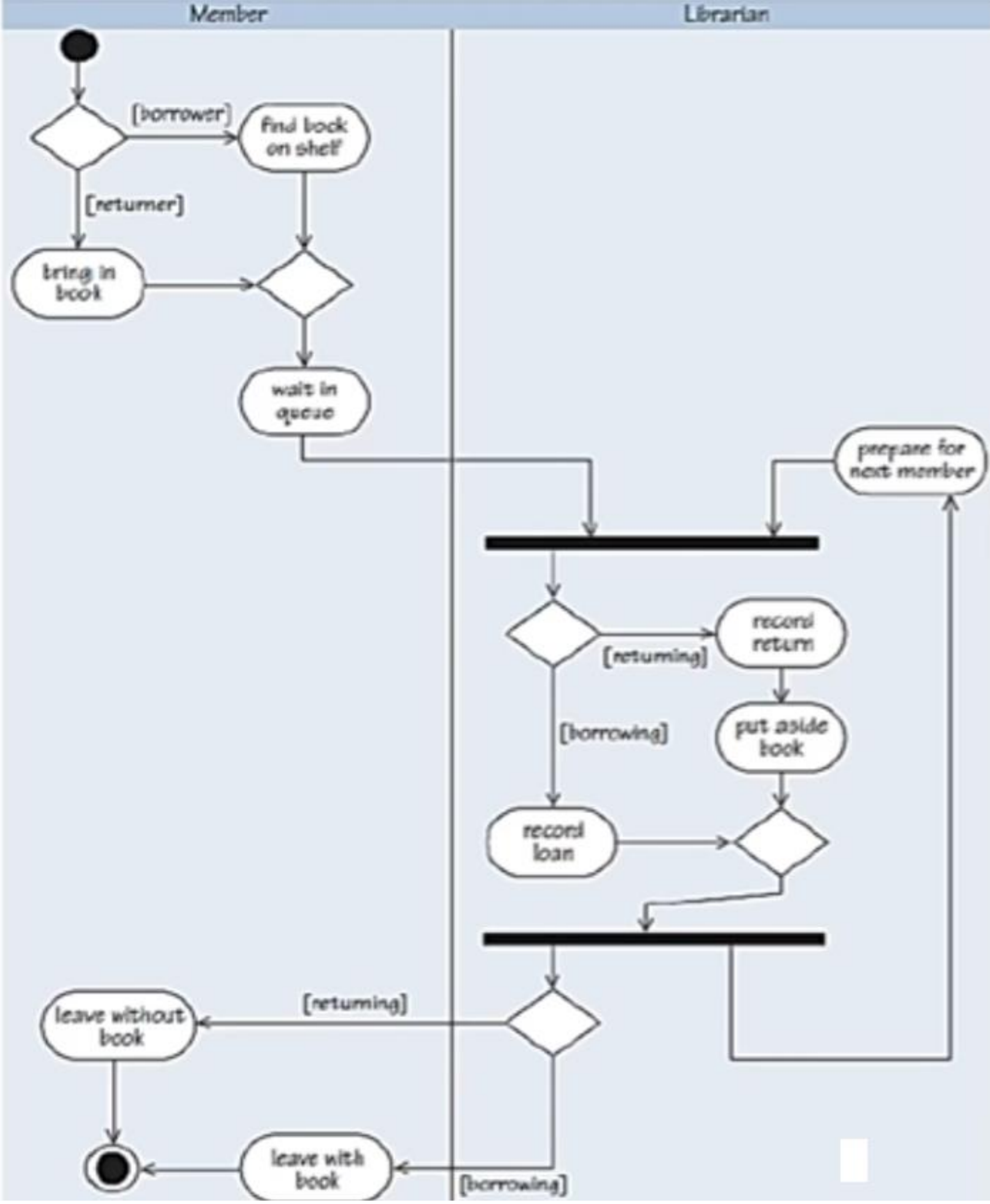


ACTIVITY DIAGRAMS .. EXAMPLE

Use examples 5 and 6 to produce a single activity diagram that models the borrowing and returning of books, based on the following assumptions:

1. You do not need to consider fines for overdue books.
2. The librarian places returned books onto a trolley (which will be used to carry the books back to the library's shelves at certain times during the day – an activity which need not be included at this stage).
3. There is just one queue to deal with members.

ACTIVITY DIAGRAMS .. EXAMPLE



ACTIVITY DIAGRAMS ..
LINKING ACTIVITIES TO USER INTERFACES

Before prototyping, the developer needs identify the appropriate interfaces from the users' requirements and determine where and when they will be needed.

- One way to achieve this goal is to use an activity diagram.

- To the users, the interface is the software system: an unacceptable interface can lead to failure.

- The user interface is the link between what the users want and what the developers produce in response

ACTIVITY DIAGRAMS ..
LINKING ACTIVITIES TO USER INTERFACES

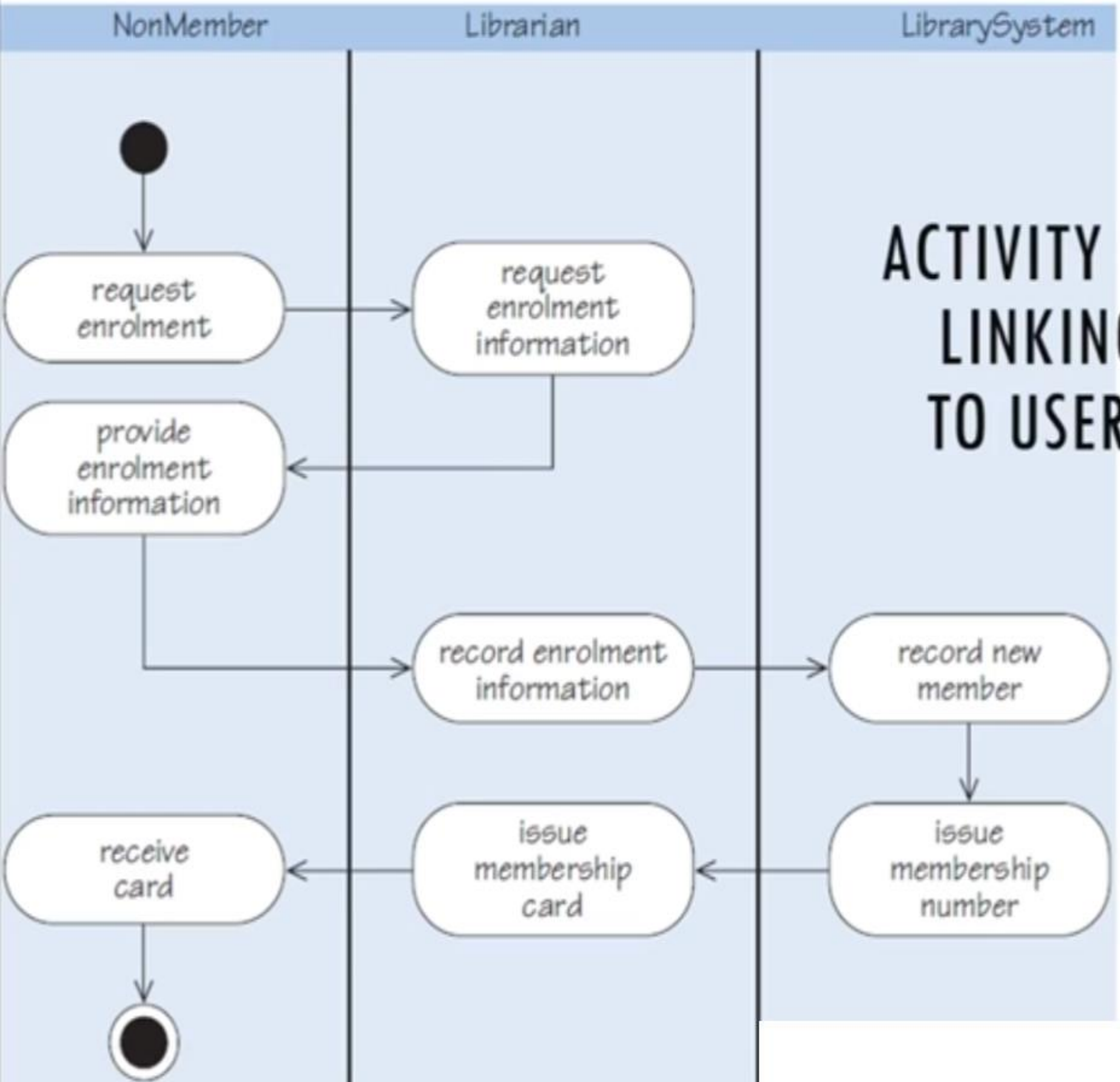
Before prototyping, the developer needs identify the appropriate interfaces from the users' requirements and determine where and when they will be needed.

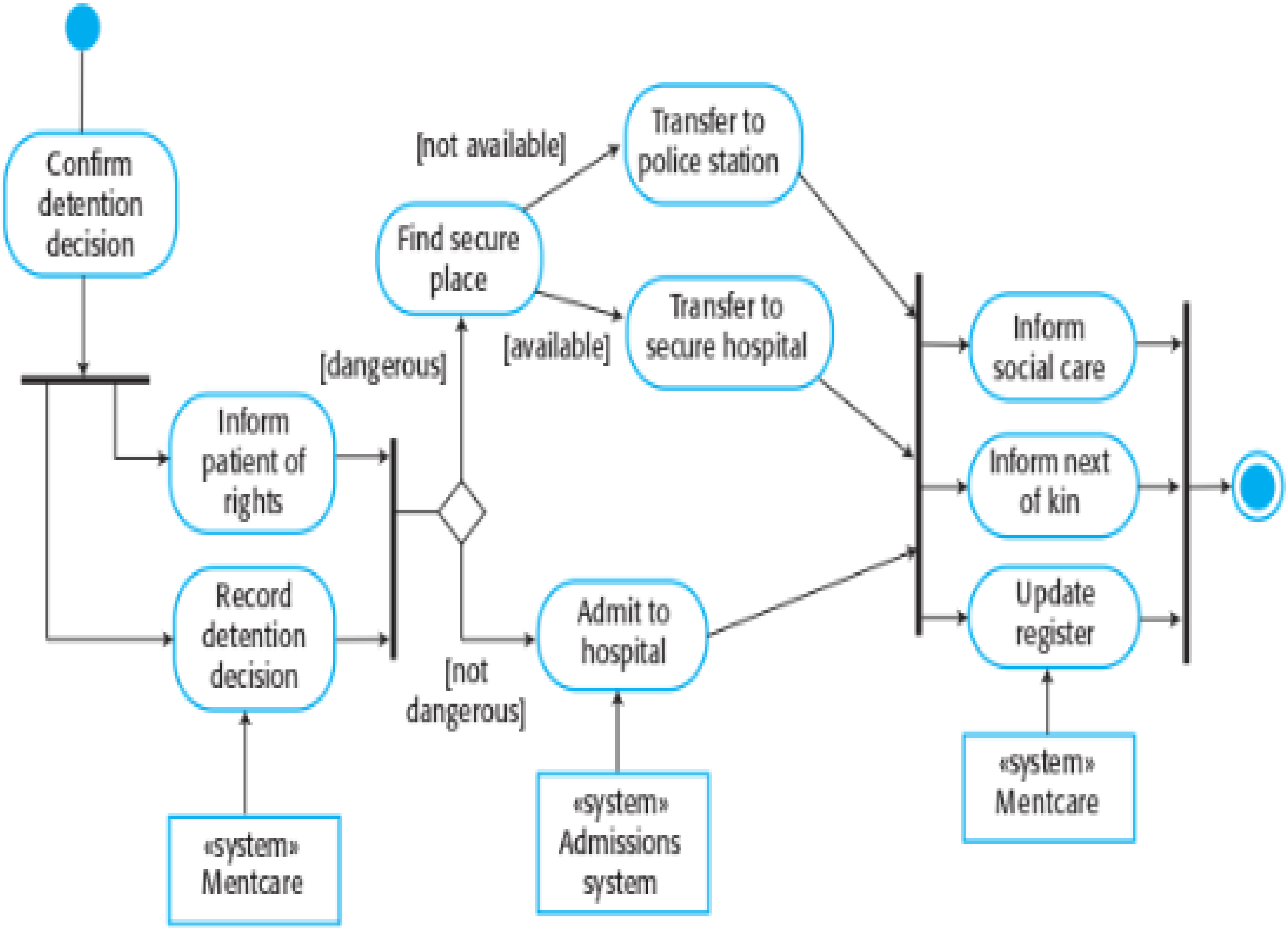
- One way to achieve this goal is to use an activity diagram.

- To the users, the interface is the software system: an unacceptable interface can lead to failure.

- The user interface is the link between what the users want and what the developers produce in response

ACTIVITY DIAGRAMS .. LINKING ACTIVITIES TO USER INTERFACES




Confirm detention decision
Inform patient of rights
Record detention decision
«system» Mentcare
[dangerous]
Find secure place
[not available]
Transfer to police station
[available]
Transfer to secure hospital
[not dangerous]
Admit to hospital
«system» Admissions system
Inform social care
Inform next of kin
Update register
«system» Mentcare

USE-CASE MODELS

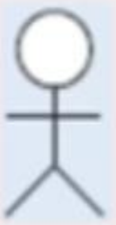
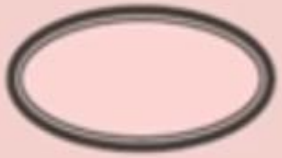
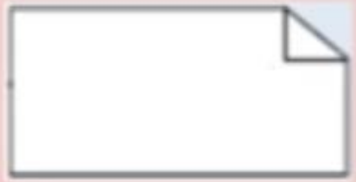
.. introduces a means of defining what a proposed system should do from a **user's perspective.**

- This can provide the basis for a contract between the customer and the developer
- Your aim is to construct a software system that will meet the needs of its users. Hence you must identify 'who does what'.
- To do this, you explore the problem description, any domain models and the initial set of user requirements to determine the people involved, the work that they do and the events that trigger some work to be done.
- You are likely to identify a variety of people, some of whom may be playing a number of different roles and may be associated with different business events.
- You should ask yourself who the actual users will be and what tasks they must perform with the aid of your software system.

USE-CASE MODELS .. ADVANTAGES

- Capturing and eliciting requirements;
- Representing requirements;
- Planning iterations of development;
- Validating software systems.
- Defining the scope of a system, as they represent the interaction of a system with its environment.
- Acting as a discussion tool between developer and user, and offer a common language for agreeing on the functions of the software system.

USE-CASE MODELS .. ELEMENTS & SYMBOLS

The actors, represented by **stick figures**;	
The use cases, represented by **ovals**, and are used to represent tasks.;	
The relationships between actors and use cases, represented by **lines**.	———
Note symbol: is used to clarify an aspect or a task, it can be used with any UML diagram.	
A **dashed line** is used to attach the note to the model element to which it refers.	- - - - - - - - - - - -

USE-CASE MODELS .. DESCRIPTIONS

Each use case description should contain the following parts in minimum:

- **Use case** identifier and name

- **Initiator**: name of the actor who initiates the process

- **Pre-condition(s)**: a condition that must hold before this use case can be carried out..

- **Post-condition(s)**: a condition that must hold after the use case has been completed.

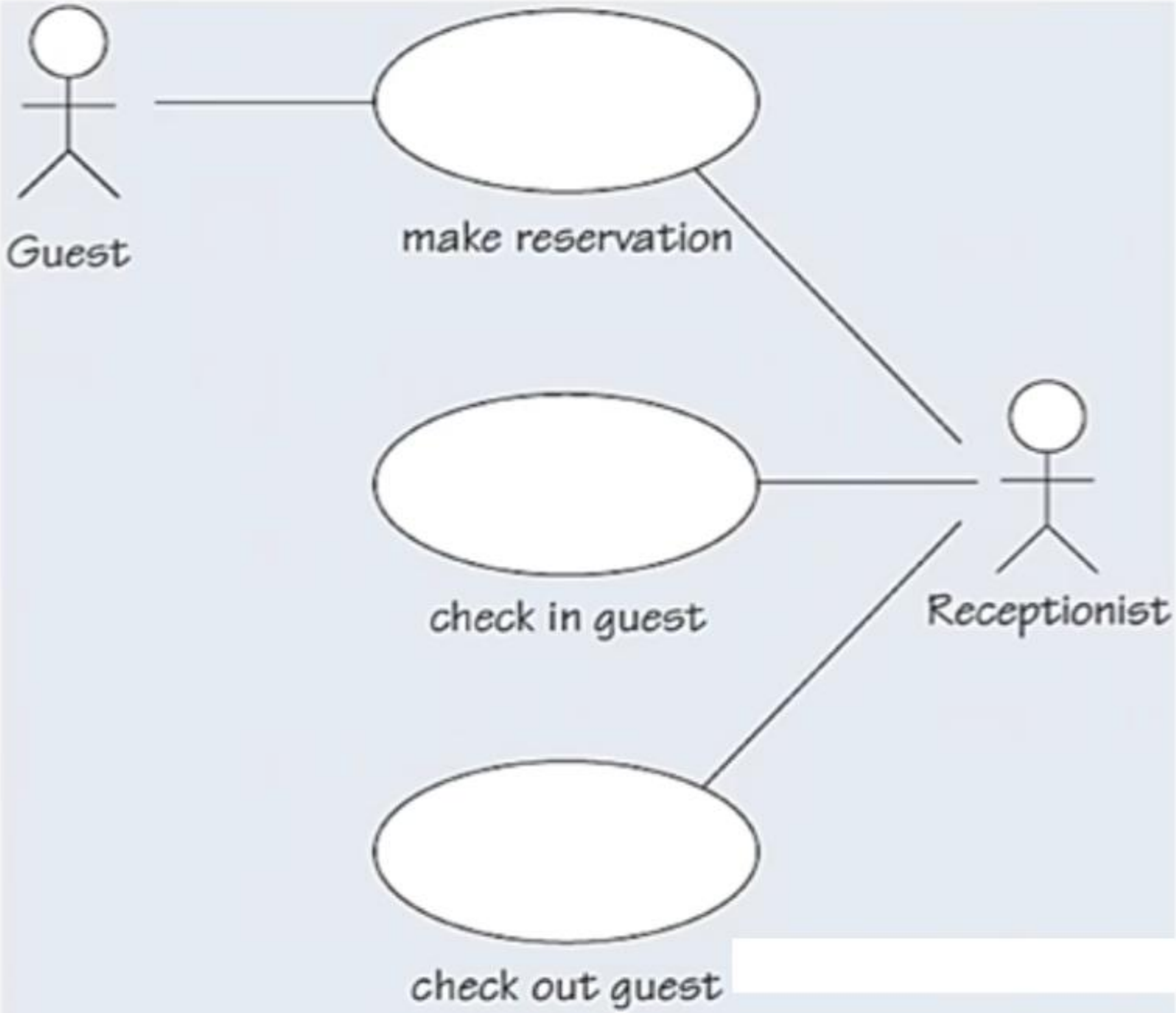
- **Main Success Scenario**: a single sequence of steps that describe the main success scenario. You can number the steps. A scenario is an instance of a use case.

- **Goal**: a short description of the goal of the use case;

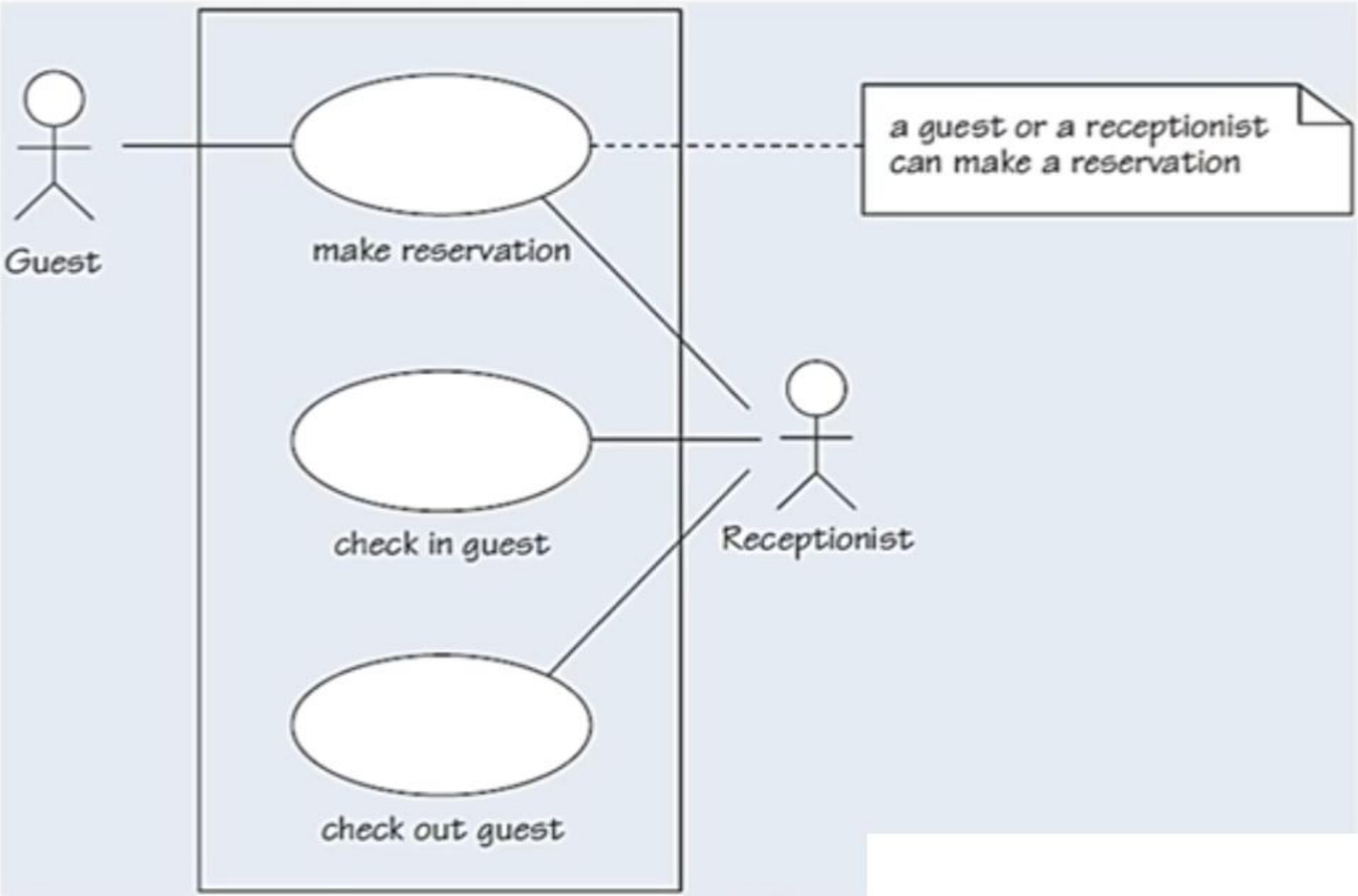
Exceptions to the normal behavior for a use case are common, especially where actors decide to cancel a use case without completing it.

USE-CASE DIAGRAMS .. EXAMPLE 1

A simple use case diagram for the main tasks in a hotel system: make reservation, check-in, and check-out, where the reservation can be done by the receptionist or a guest (by Phone/Web).



USE-CASE DIAGRAMS .. SYSTEM BOUNDARY



USE-CASE DESCRIPTIONS

A <u>**scenario**</u> describes a sequence of interactions between the system and some actors.

- In Each use case there is a set of possible scenarios. Where the main scenario is the successful scenario where nothing goes wrong and the use case is achieved.

For example: there are two scenarios for making reservation in a hotel:

Main Success Scenario:

- The guest wants to reserve a double room at the Hotel for 14 July. A double room is available for that date, and the reservation is done.

Unsuccessful Scenario:

- The guest wants to reserve a single room at the Hotel for the first week of August. There is no single room that is free for seven days in August, but there is one room available for four days and another one for the following three days. The system presents that option to the guest, who rejects it.

USE-CASE DESCRIPTIONS

What pre and post condition(s) you can obtain from the below description of a hotel check-in process?

Upon arrival, each guest provides the reservation number for his or her reservation to the hotel's receptionist, who enters it into the software system. The software system reveals the details of that reservation so that each guest can confirm them. The software system allocates an appropriate room to that guest and opens a bill for the duration of the stay. The receptionist issues a key for the room.

Precondition(s): There must be a reservation for the guest, and there must be at least one room available (of the desired type), and the guest must be able to pay for the room.

Post-condition(s): The guest will have been allocated to a room for the period identified in the reservation, the room will have been identified as being in use for a specific period, a bill will have been opened for the duration of the stay, and a key will have been issued.

USE-CASE DESCRIPTIONS

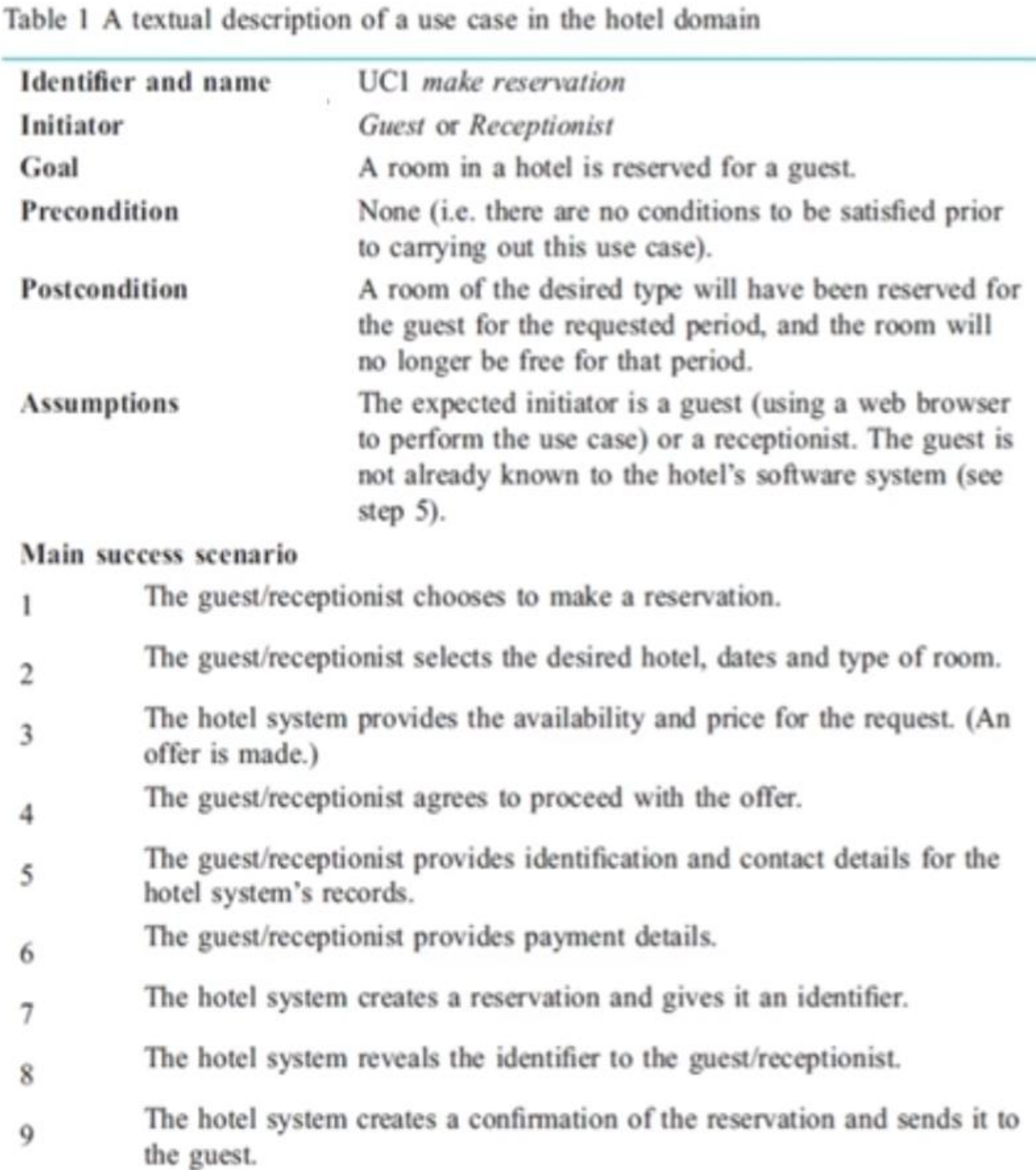


Table 1 A textual description of a use case in the hotel domain

Identifier and name	UC1 *make reservation*
Initiator	*Guest* or *Receptionist*
Goal	A room in a hotel is reserved for a guest.
Precondition	None (i.e. there are no conditions to be satisfied prior to carrying out this use case).
Postcondition	A room of the desired type will have been reserved for the guest for the requested period, and the room will no longer be free for that period.
Assumptions	The expected initiator is a guest (using a web browser to perform the use case) or a receptionist. The guest is not already known to the hotel's software system (see step 5).

Main success scenario

1 The guest/receptionist chooses to make a reservation.

2 The guest/receptionist selects the desired hotel, dates and type of room.

3 The hotel system provides the availability and price for the request. (An offer is made.)

4 The guest/receptionist agrees to proceed with the offer.

5 The guest/receptionist provides identification and contact details for the hotel system's records.

6 The guest/receptionist provides payment details.

7 The hotel system creates a reservation and gives it an identifier.

8 The hotel system reveals the identifier to the guest/receptionist.

9 The hotel system creates a confirmation of the reservation and sends it to the guest.

USE-CASE SCENARIOS

For each use case there is a set of possible scenarios. A scenario is an instance of a use case. A scenario describes a sequence of interactions between the system and some actors. Here are two examples of scenarios.

- A member of a lending library wishes to borrow a book, and is allowed to do that as long as they have no outstanding loans.
- Another member wishes to borrow a book, but has exceeded the quota for the number of books that can be borrowed.

In each scenario the member wishes to borrow a book, but both the circumstances and outcomes of events are different in each instance.

A use case includes a complex set of requirements that the system must meet in order to cope with every eventuality.

The main success scenario shows the steps normally followed to achieve the stated goal of the use case. But there can be other scenarios for the same use case, each one having different outcomes depending upon circumstances.

USE-CASE MODELS ..
GENERALIZATION AMONG ACTORS

UML provides a notation to use generalisation between actors.

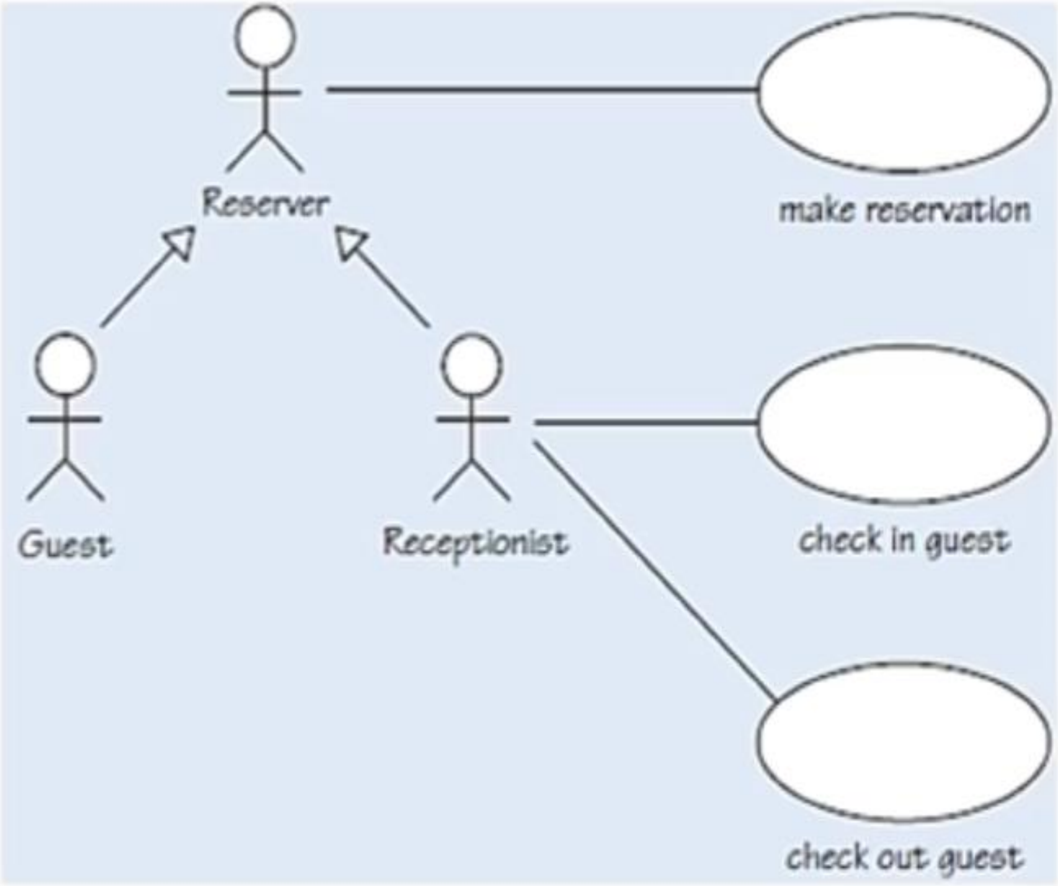
- When two actors share the same behaviour (interacting with the same use cases) and one of them has some extra behaviour, then the common behaviour can be associated with a generalised actor and the more specific behaviour with the specialised actor.

For example, in the hotel system both the receptionist and the guest are allowed to make reservation.

- Guest can make reservation through online system;

- the receptionist can do reservation on behalf of guest when a guest do reservation through phone or a fax.

USE-CASE MODELS .. GENERALIZATION AMONG ACTORS



We introduce a new actor called Reserver associated with the make reservation use case.

- The actors Guest and Receptionist specializes Reserver.
- Guest can do the same things that a Reserver can, but may do other things too.
- By using the open-headed arrow from Guest to Reserver, you are saying that a Guest can do the same things that a Reserver can, but may do something else that a Reserver does not. Reserver is a generalised actor and Guest a specialised one.

USE-CASE MODELS ..
RELATIONSHIPS BETWEEN USE-CASES

Use cases can be related to one another. There are two very common and important forms of relationship between use cases: **inclusion** and **extension.**

Inclusion: This is when two or more use cases have a common area of functionality that can be factored out as a distinct use case.

The new use case can be used by each of the original use cases, so avoiding duplication.

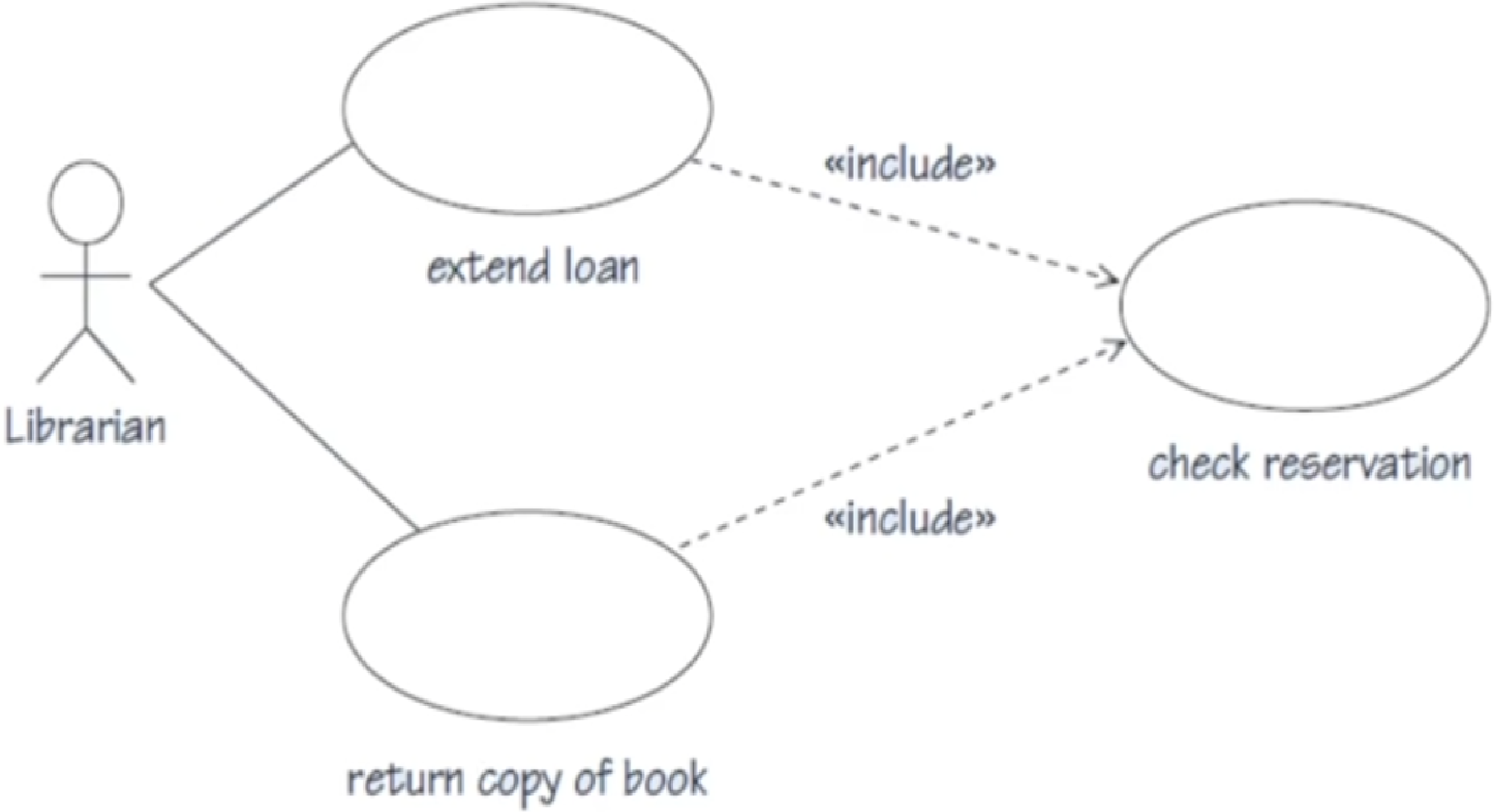
Extension: This is when a use case has a main success scenario but also alternative scenarios which demand a variation on the original use case – different or additional actions.

Each variation can be separated out as a use case that is distinct from but related to the original use case.

An extension is conditional while an inclusion is not.

In UML we use «include» and «extend» **stereotypes** for representing inclusion and extension.

USE-CASE MODELS .. «INCLUDE» .. EXAMPLE



Librarian
extend loan
«include»
check reservation
«include»
return copy of book