Advanced Software Engineering

Use Case Modeling

Prof. K. Priyantha Hewagamage

Developing Use-Case Models of Systems

A *use case* is a typical sequence of actions that a user performs in order to complete a given task

- The objective of *use case analysis* is to model the system

  … from the point of view of how users interact with this system

  … when trying to achieve their objectives.

- A *use case model* consists of

  —a set of use cases

  —an optional description or diagram indicating how they are related

Use cases

- In general, a use case should cover the *full sequence of steps* from the beginning of a task until the end.
- A use case should describe the *user's interaction* with the system ...

  —not the computations the system performs.
- A use case should be written so as to be as *independent* as possible from any particular user interface design.
- A use case should only include actions in which the actor interacts with the computer.

main purposes to get through with the use case diagrams.

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.
- Show the interaction among the requirements and actors.

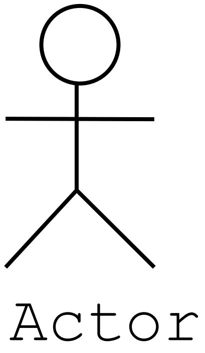
Actor

An actor is **behavioral classifier** which specifies **a role** played by an external entity that interacts with the subject,

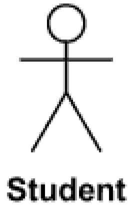
a human user of the designed system,

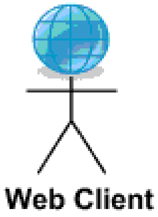
some other system

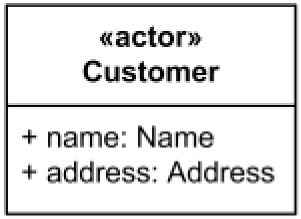
hardware using services of the subject



Examples






«actor»
Customer

+ name: Name
+ address: Address

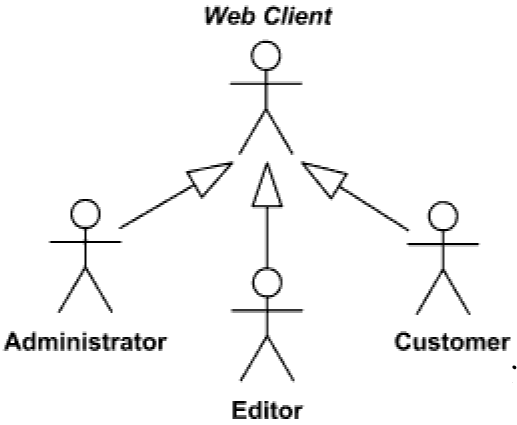
Actor …

An actor can only have binary associations to use cases, components, and classes

An actor has a unique name and an optional description

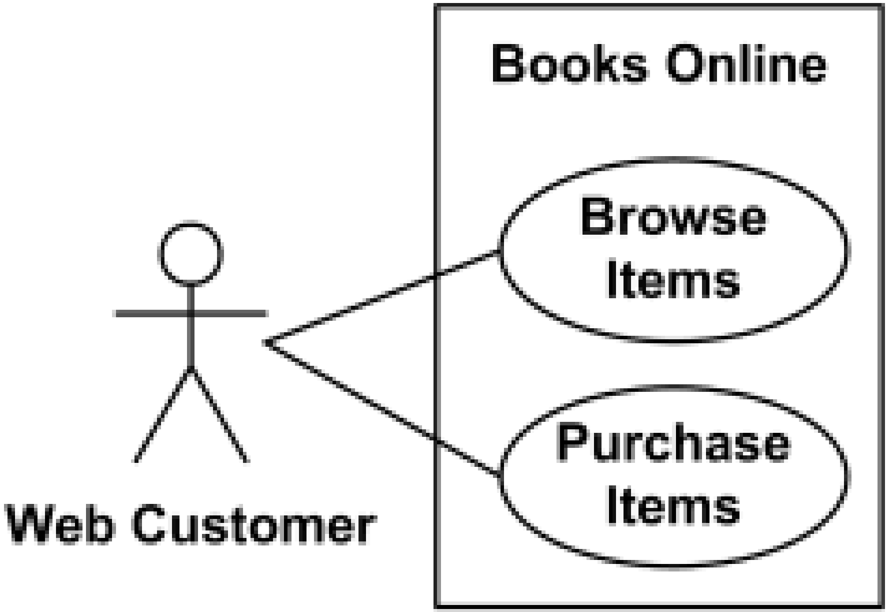
- **Passenger** [A person in the train]
- **GPS satellite** [An external system that provides the system with GPS coordinates]

can define **abstract or concrete actors** and specialize them using generalization relationship

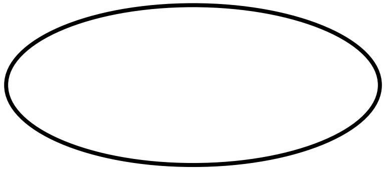


Subject (2.5)

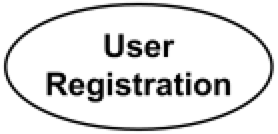
A subject of a use case defines and represents boundaries of a business, software system, physical system or device, subsystem, component or even single class in relation to the requirements gathering and analysis

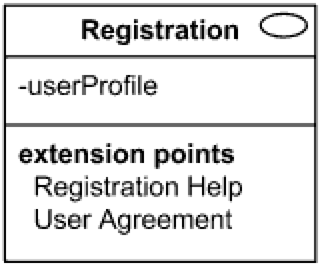


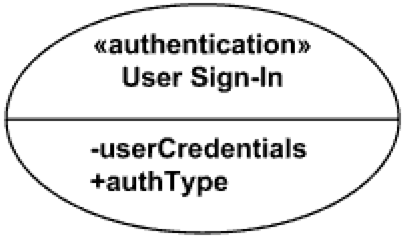
Use Case

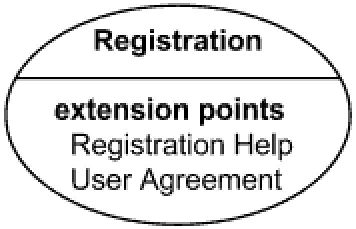


- A use case represents a class of functionality provided by the system
- Use cases can be described textually, with a focus on the event flow between actor and system
- The textual use case description consists of 6 parts:
  1. Unique name
  2. Participating actors
  3. Entry conditions
  4. Exit conditions
  5. Flow of events
  6. Special requirements.

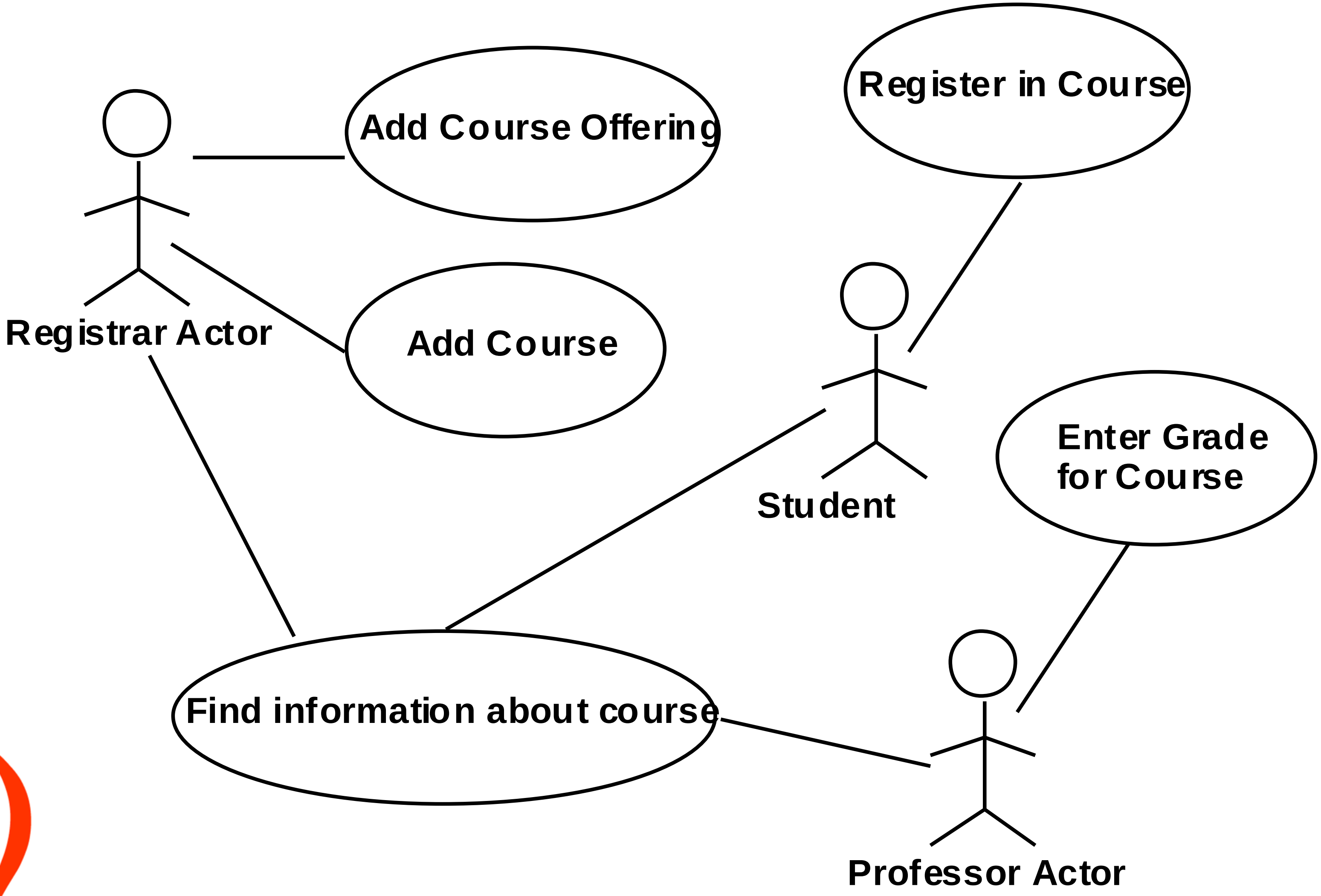


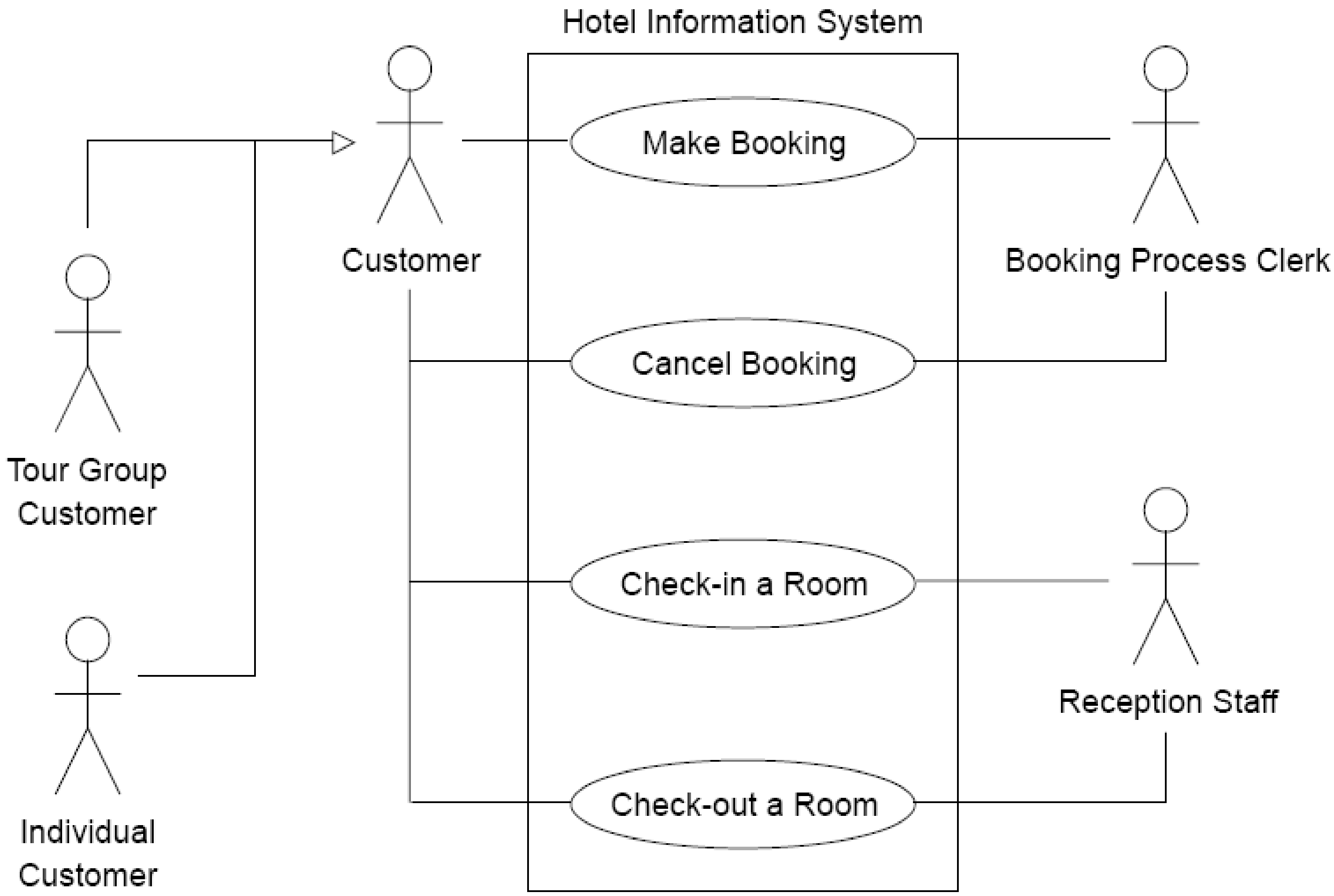






Use case diagrams

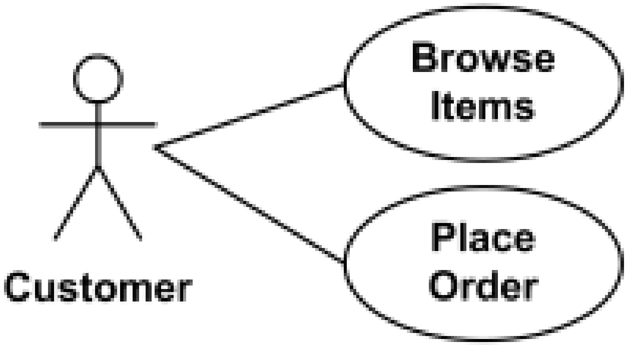


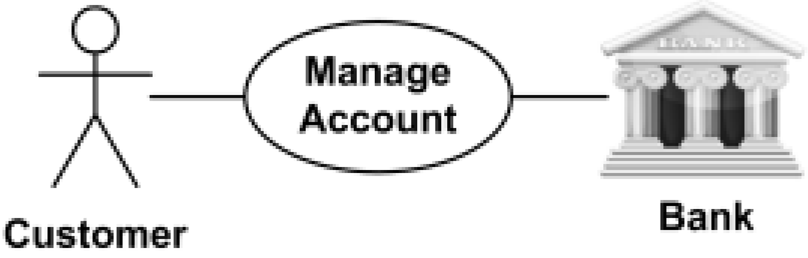

Hotel Information System
Make Booking
Cancel Booking
Check-in a Room
Check-out a Room
Customer
Tour Group Customer
Individual Customer
Booking Process Clerk
Reception Staff

Use Case Association

An **association** between an actor and a use case indicates that the actor and the use case somehow interact or communicate with each other.

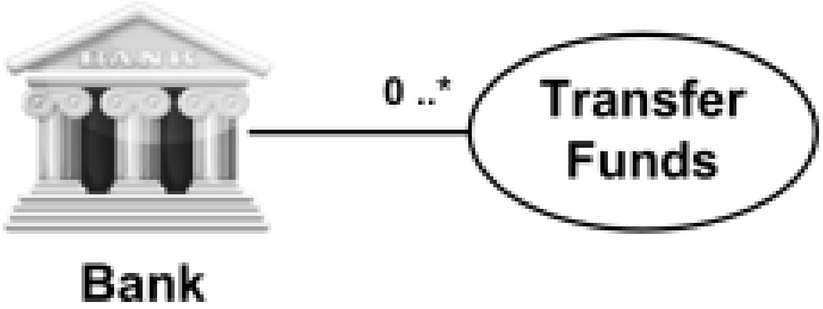
Only **binary associations** are allowed between actors and use cases.



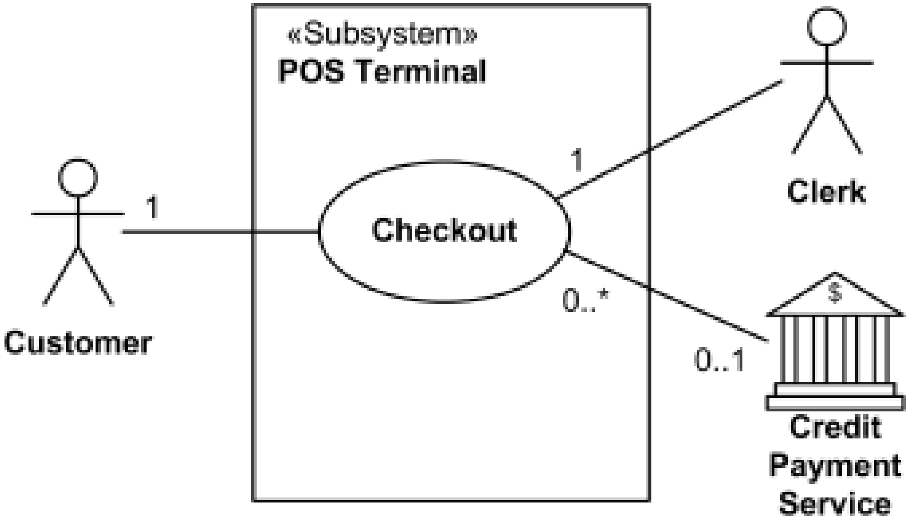


Multiplicity of Use Case or Actor (2.5)

Bank involves multiple transfer of funds

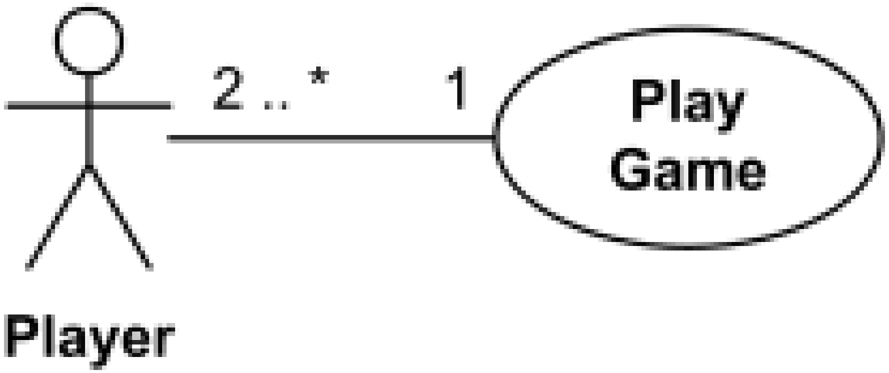


Checkout use case requires Customer actor, hence the 1 multiplicity of Customer. The use case may not need Credit Payment Service (for example, if payment is in cash), thus the 0..1 multipl



Two or more Player actors are involved in the Play Game use case.

Each Player participates in one Play Game.



Multiplicity of an actor could mean simultaneous or overleaping or mutually exclusive interaction

Use Case Associations

A use-case model consists of use cases and use case associations

A use case association is a relationship between use cases

important types of use case associations:

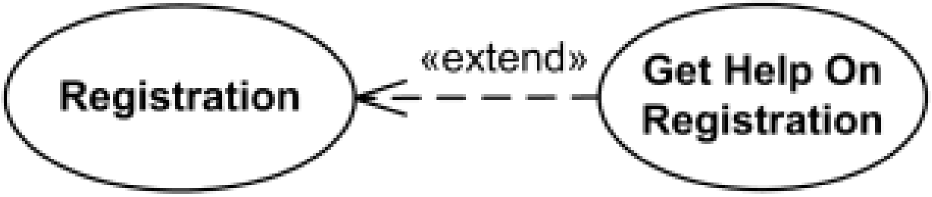
Include,

Extend,

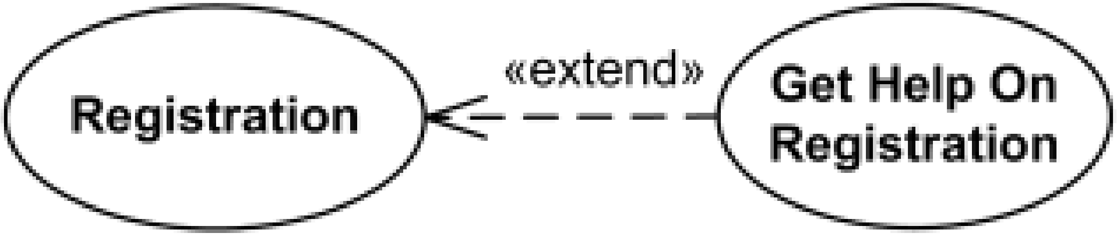
Generalization

Extensions of Use Cases

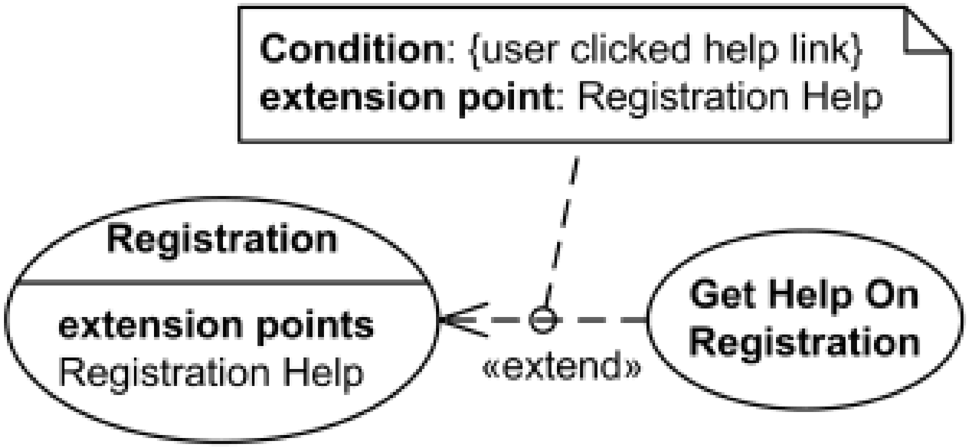
- Used to make *optional* interactions explicit or to handle *exceptional* cases.
- By creating separate use case extensions, the description of the basic use case remains simple.
- A use case extension must list all the steps from the beginning of the use case to the end
  - —Including the handling of the unusual situation.
- Please note UML style, stereotype and dash arrow



Extension Points in Use Cases

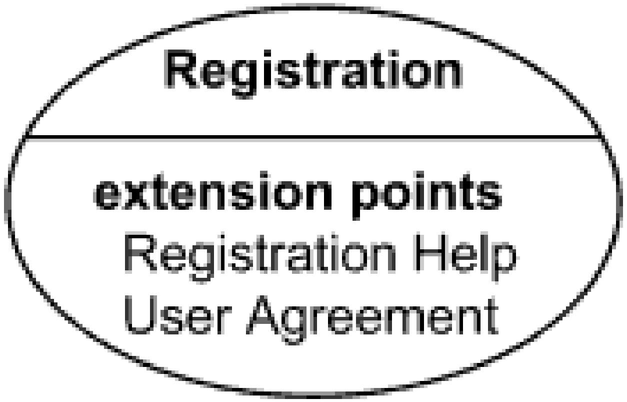


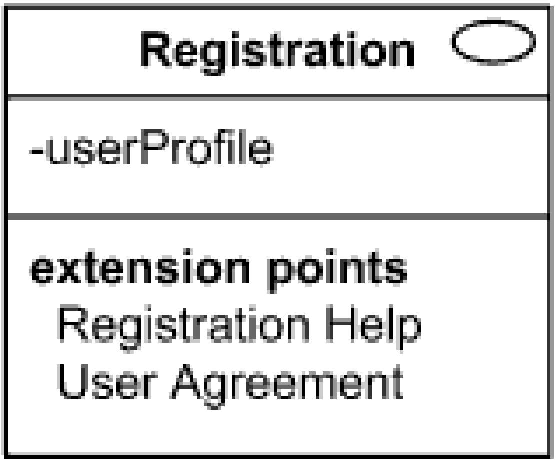
Registration use case is complete and meaningful on its own.
It could be extended with optional **Get Help On Registration** use case.



The **condition** of the extend relationship as well as the references to the extension points are optionally shown in a comment note attached to the corresponding extend relationship.

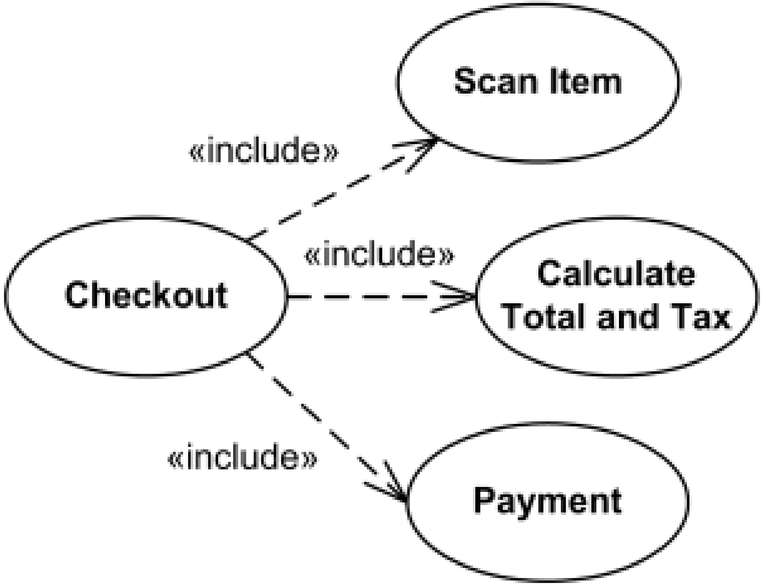
More than one extension point



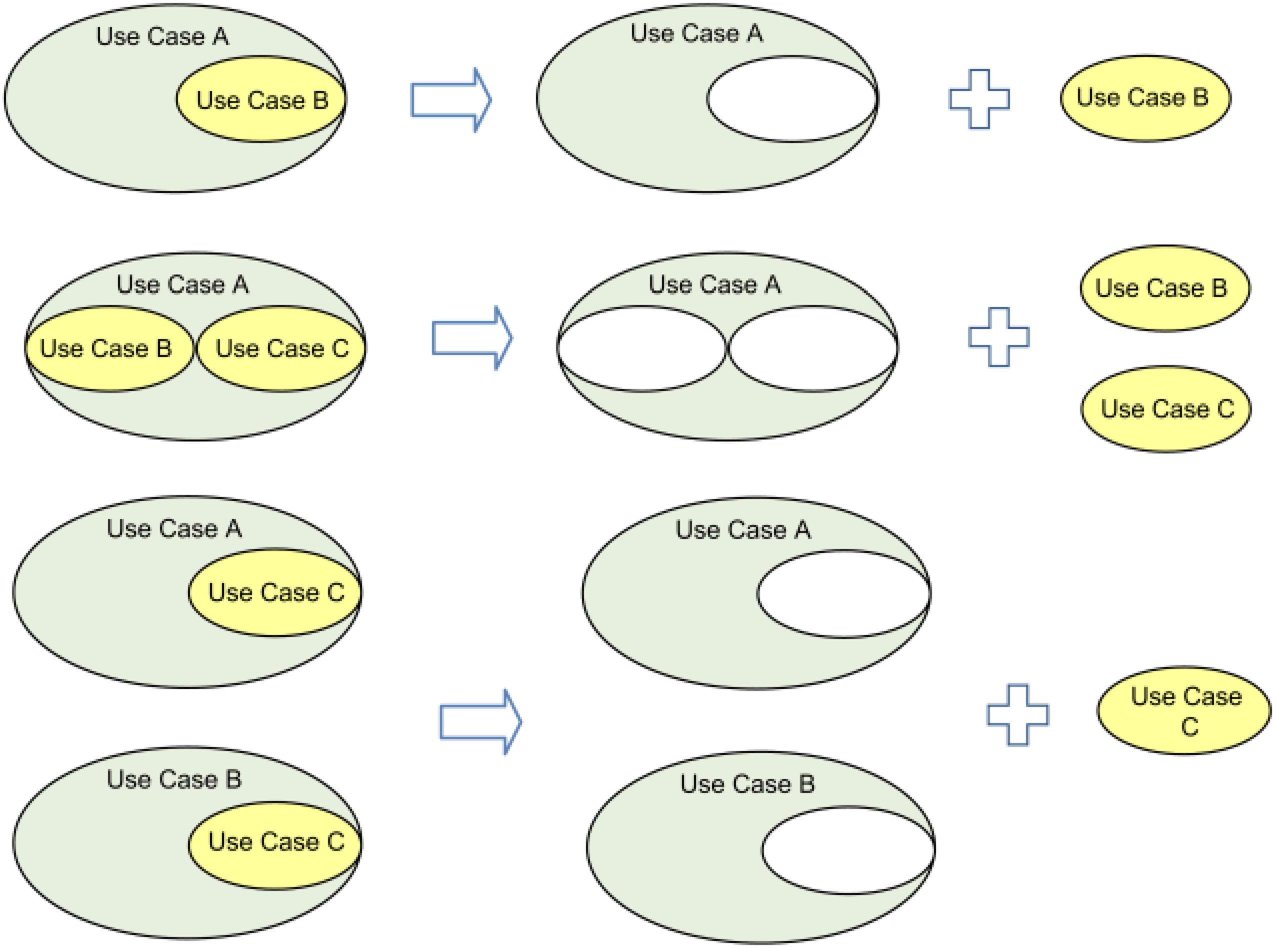


Inclusions of Use Cases

- Allow one to express *similarity* between several different use cases.
- Are included in other use cases
  - —Even very different use cases can share sequence of actions.
  - —Enable you to avoid repeating details in multiple use cases.
- Represent the performing of a *lower-level task* with a lower-level goal.



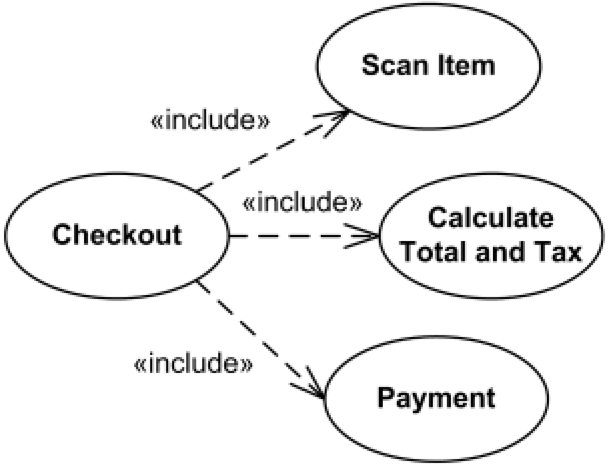
Possible inclusions

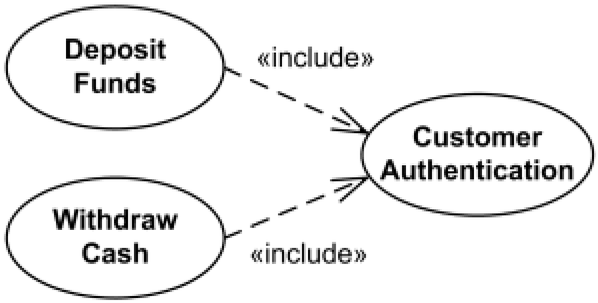


<<Inclusion>>

Objectives of Inclusion relationship

- to simplify large use case by splitting it into several use cases,
- to extract common parts of the behaviors of two or more use cases.

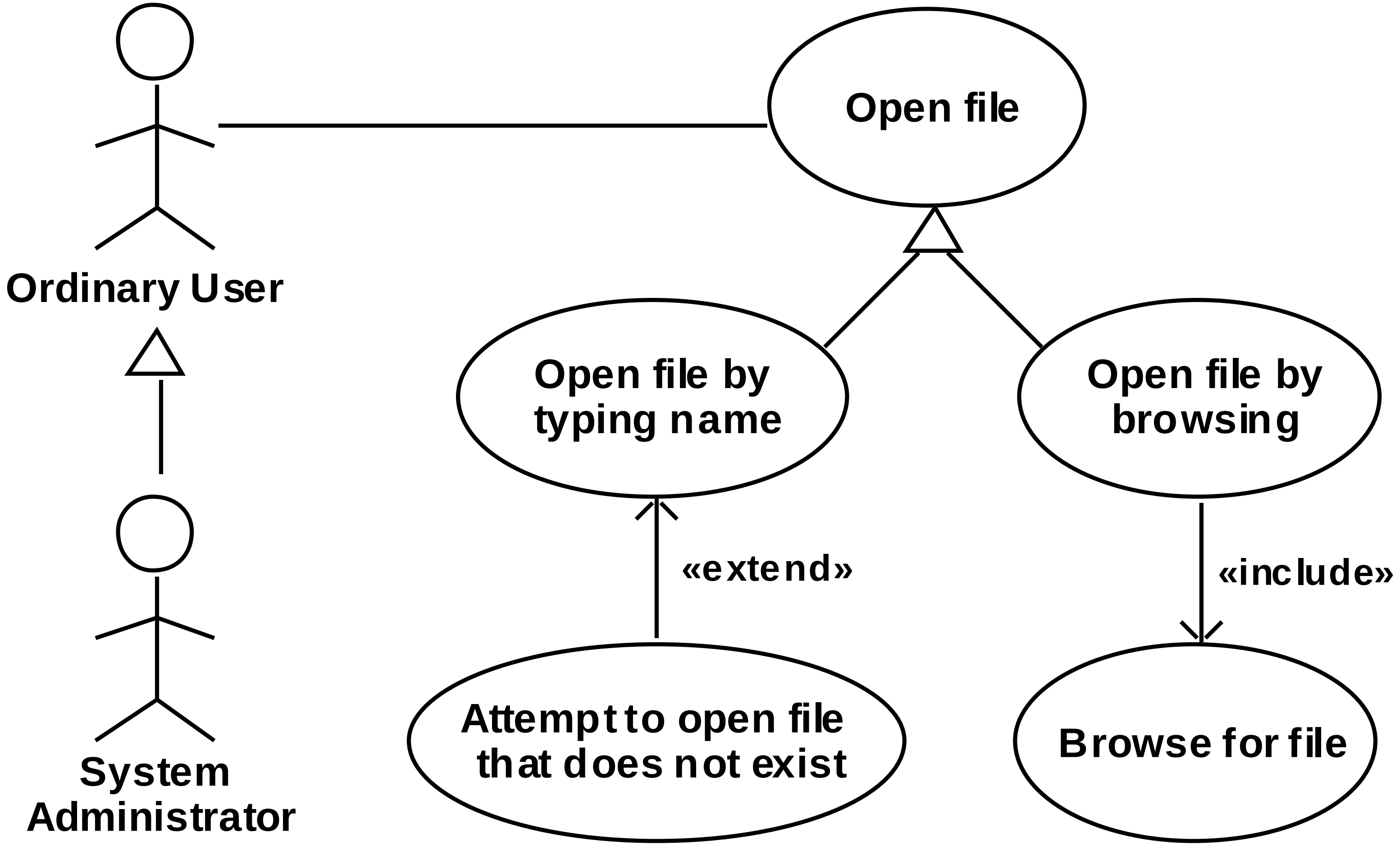




Generalizations of Use Cases

- Much like super classes in a class diagram.
- A generalized use case represents *several similar* use cases.
- One or more specializations provides details of the similar use cases.

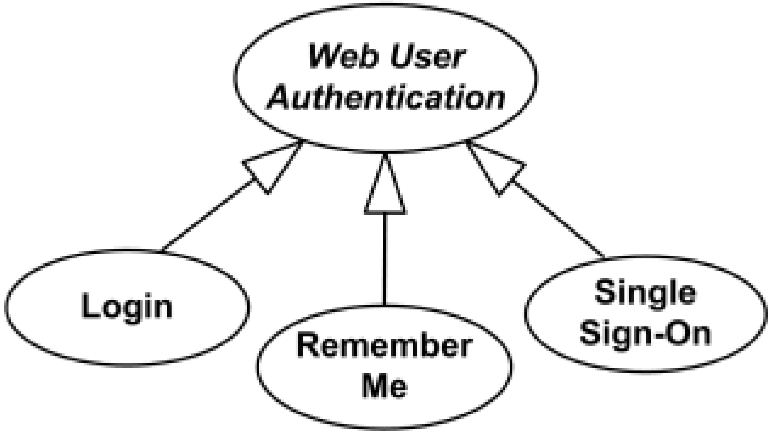
Example of generalization, extension and inclusion

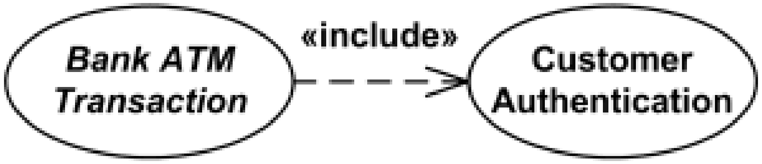


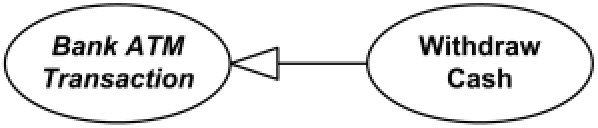
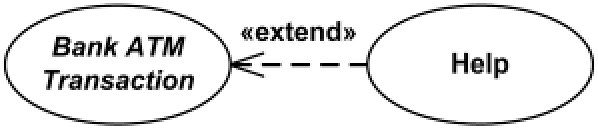
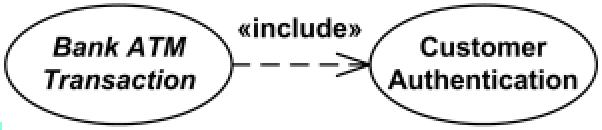
Abstract Use Case

UML 2.5 – no specific definition like 1.5 , good to have all use cases specific (concrete)

Possible abstract use cases (use case name appear in italics)





Generalization	Extend	Include
		 Bank ATM Transaction «include» Customer Authentication
Base use case could be abstract use case (incomplete) or concrete (complete).	Base use case is complete (concrete) by itself, defined independently.	Base use case is incomplete (abstract use case).
Specialized use case is required, not optional, if base use case is abstract.	Extending use case is optional, supplementary.	Included use case required, not optional.
No explicit location to use specialization.	Has at least one explicit extension location.	No explicit inclusion location but is included at some location.
No explicit condition to use specialization.	Could have optional extension condition.	No explicit inclusion condition.

Guidelines for Drawing Use Case Diagrams

Define Subject (system Boundaries)

Identify the actors and the use cases

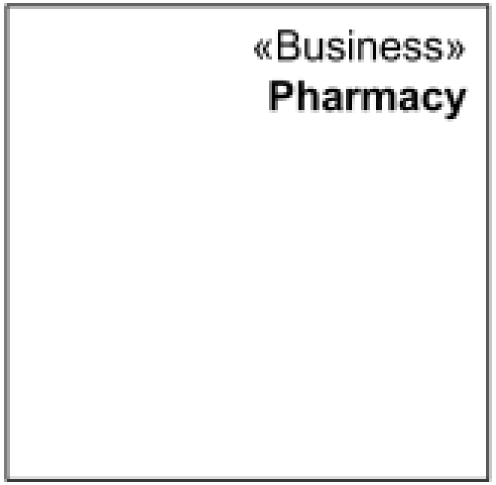
Prioritize the use cases

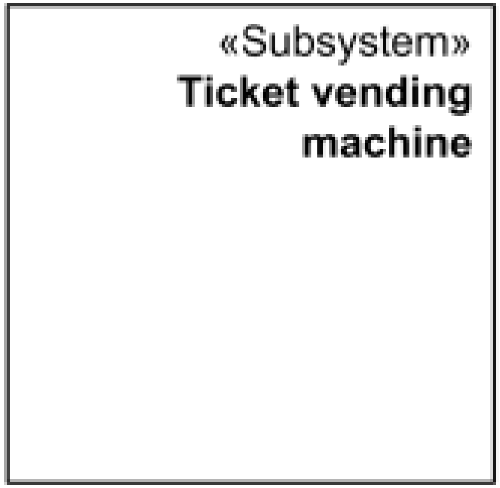
Develop each use case, starting with the priority ones, writing a description for each

Add structure to the use case model: generalization, include and extend relationships and sub-systems

Define Subject

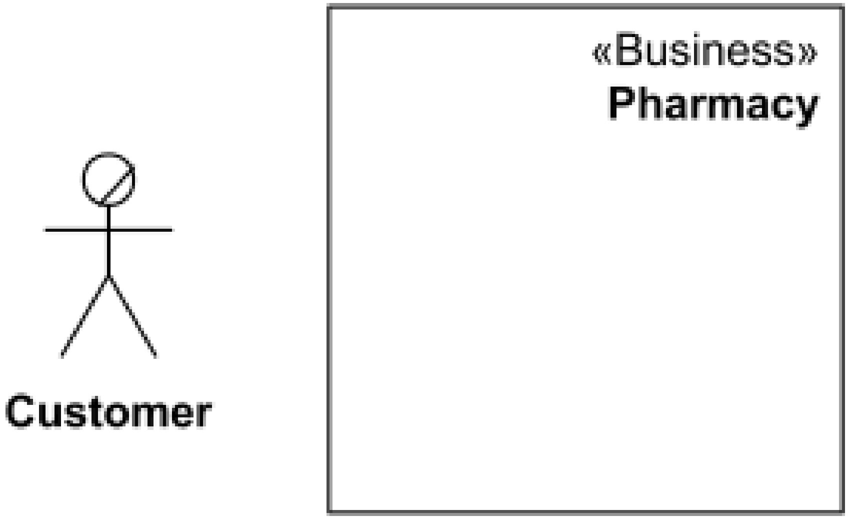
Subject is a business, software system, subsystem, component, device, etc. that we are designing or just trying to understand how it is working

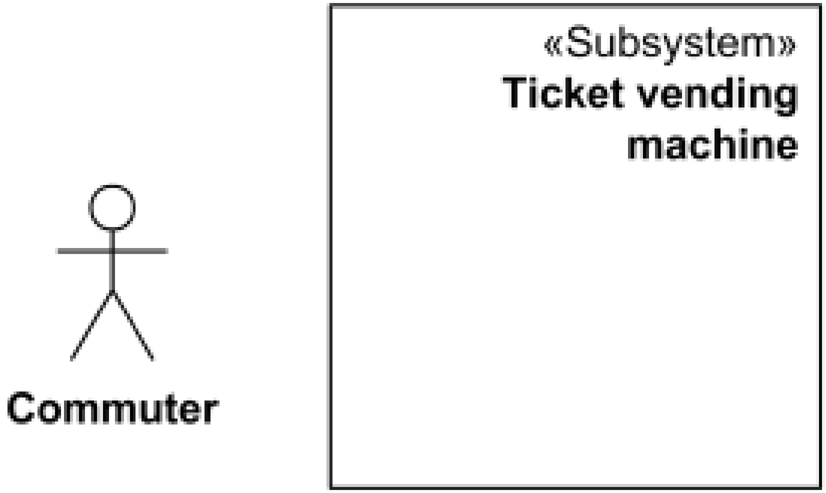




Identify / Define Actors

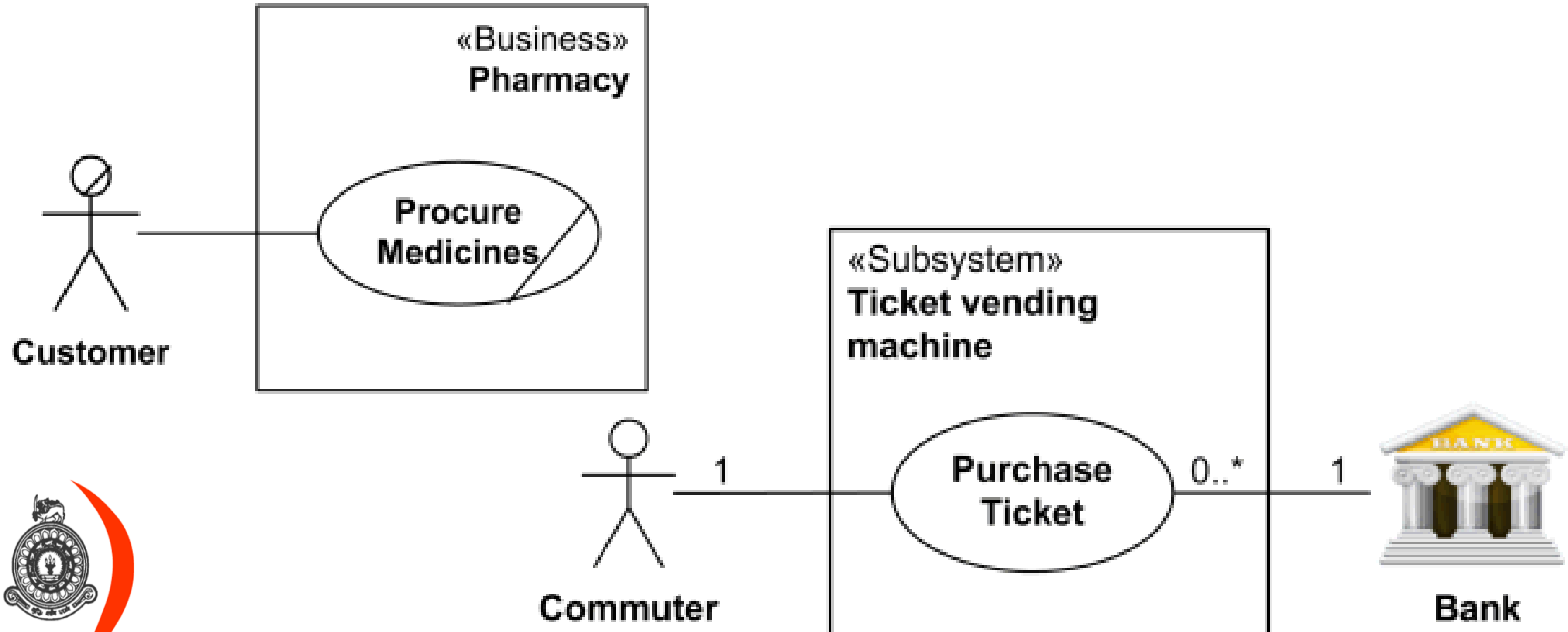
Actor is an external entity, which could be a human user of the designed system, or some other system or device using our system.





Define Use Case

Each use case specifies a unit of complete and useful functionality that the subject provides to the actor(s). Use case should reflect user needs and goals, and should be initiated by an actor.

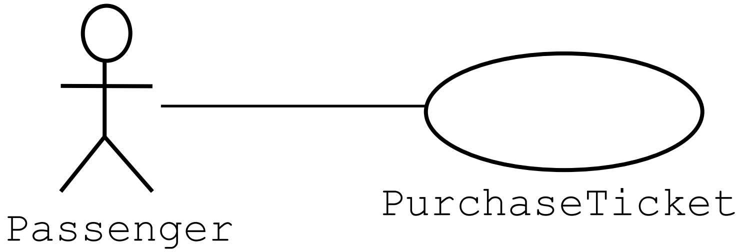


Describe the use case behavior

Use case behaviors may be described

- in a natural language text (**Use Case Description**), which is current common practice, or
- by using UML **behavior diagrams** for specific behaviors such as
  - —activity,
  - —state machine,
  - —interaction.

Textual Use Case Description Example



1. Name: **Purchase ticket**

2. Participating actor: **Passenger**

3. Entry condition:

Passenger stands in front of ticket distributor

Passenger has sufficient money to purchase ticket

4. Exit condition:

Passenger has ticket

5. Flow of events:

1. Passenger selects the number of zones to be traveled
2. Ticket Distributor displays the amount due
3. Passenger inserts money, at least the amount due
4. Ticket Distributor returns change
5. Ticket Distributor issues ticket

6. Special requirements: None.

How to describe a single use case

A. **Name**: Give a short, descriptive name to the use case.

B. **Actors**: List the actors who can perform this use case.

C. **Goals**: Explain what the actor or actors are trying to achieve.

D. **Preconditions**: State of the system before the use case.

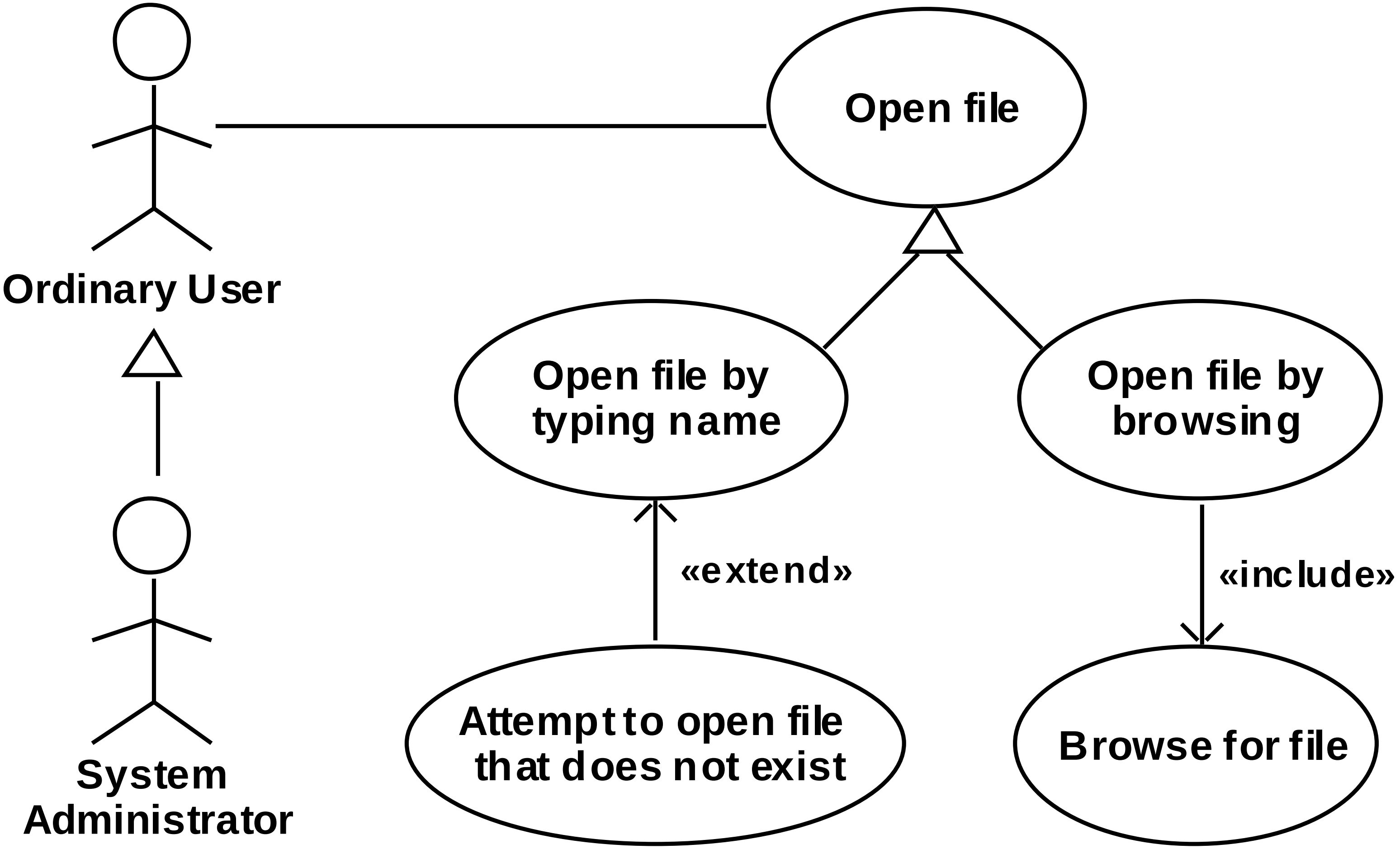
E. **Description**: Give a short informal description.

F. **Related use cases.**

G. **Steps**: Describe each step using a 2-column format.

H. **Postconditions**: State of the system.

Example of generalization, extension and inclusion



Example description of a use case

Use case: **Open file**

Related use cases:
Generalization of:

- Open file by typing name
- Open file by browsing

Steps:

Actor actions	**System responses**
1. Choose ‘Open…’ command	2. File open dialog appears
3. Specify filename	
4. Confirm selection	5. Dialog disappears

Example (continued)

Use case: **Open file by typing name**

Related use cases:
Specialization of: Open file

Steps:

Actor actions	**System responses**
1. Choose ‘Open…’ command	2. File open dialog appears
3a. Select text field	
3b. Type file name	
4. Click ‘Open’	5. Dialog disappears

Example (continued)

Use case: **Open file by browsing**

Related use cases:
Specialization of: Open file
Includes: Browse for file

Steps:

Actor actions	**System responses**
1. Choose ‘Open…’ command	2. File open dialog appears
3. Browse for file	
4. Confirm selection	5. Dialog disappears

Example (continued)

Use case: **Attempt to open file that does not exist**

Related use cases:
Extension of: Open file by typing name

Actor actions	**System responses**
1. Choose ‘Open…’ command	2. File open dialog appears
3a. Select text field	
3b. Type file name	
4. Click ‘Open’	5. System indicates that file does not exist
6. Correct the file name	
7. Click ‘Open’	8 Dialog disappears

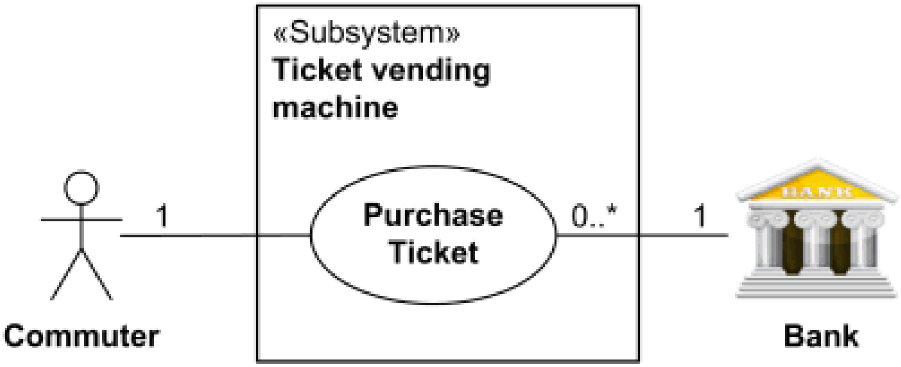
Example (continued)

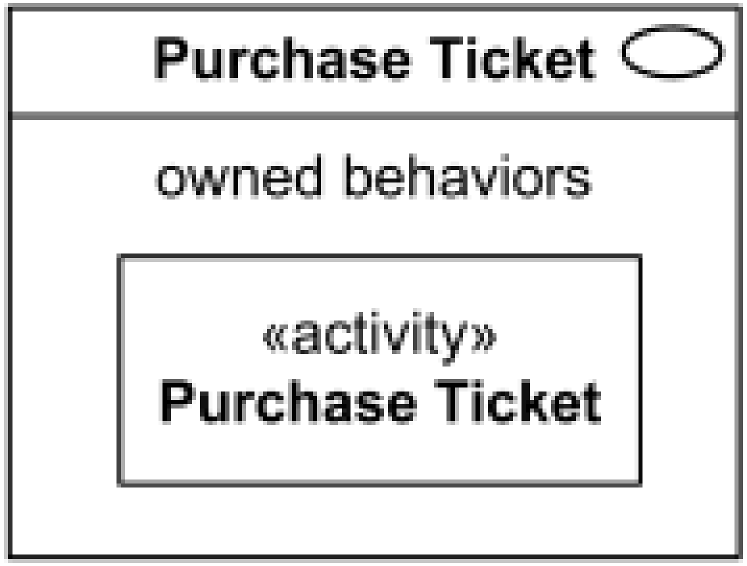
Use case: **Browse for file (inclusion)**

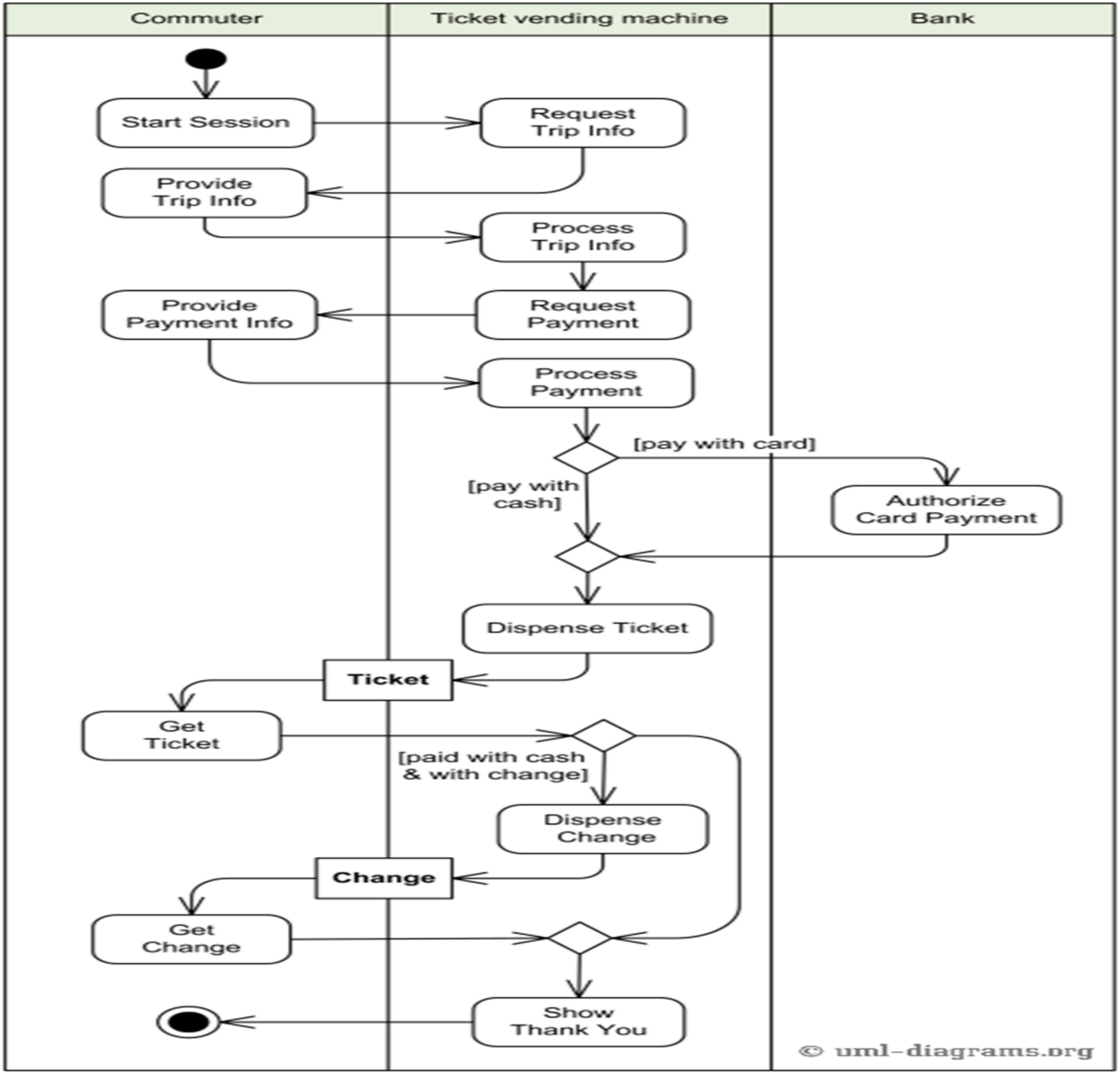
Steps:

Actor actions	**System responses**
1. If the desired file is not displayed, select a directory	2. Contents of directory is displayed
3. Repeat step 1 until the desired file is displayed	
4. Select a file	

Describing the behavior of Use case (2.5)






Commuter
Ticket vending machine
Bank
Start Session
Request Trip Info
Provide Trip Info
Process Trip Info
Provide Payment Info
Request Payment
Process Payment
[pay with card]
[pay with cash]
Authorize Card Payment
Dispense Ticket
Ticket
Get Ticket
[paid with cash & with change]
Dispense Change
Change
Get Change
Show Thank You
© uml-diagrams.org

Scenarios

A <u>scenario</u> is an *instance* of a use case

- It expresses a *specific occurrence* of the use case
  - —a specific actor ...
  - —at a specific time ...
  - —with specific data.

Scenarios

What is a scenario?

- It is a single logical path in a use case

A use case may have one or more scenarios

- Test cases are developed based on all scenarios in a use case

Each scenario has a single point (why?)

- From every decision point a scenario branch into two or more new paths

Good source to write “User Guide”

Hint: the complexity of a use case

How to identify scenarios?

Analyze the given textual description

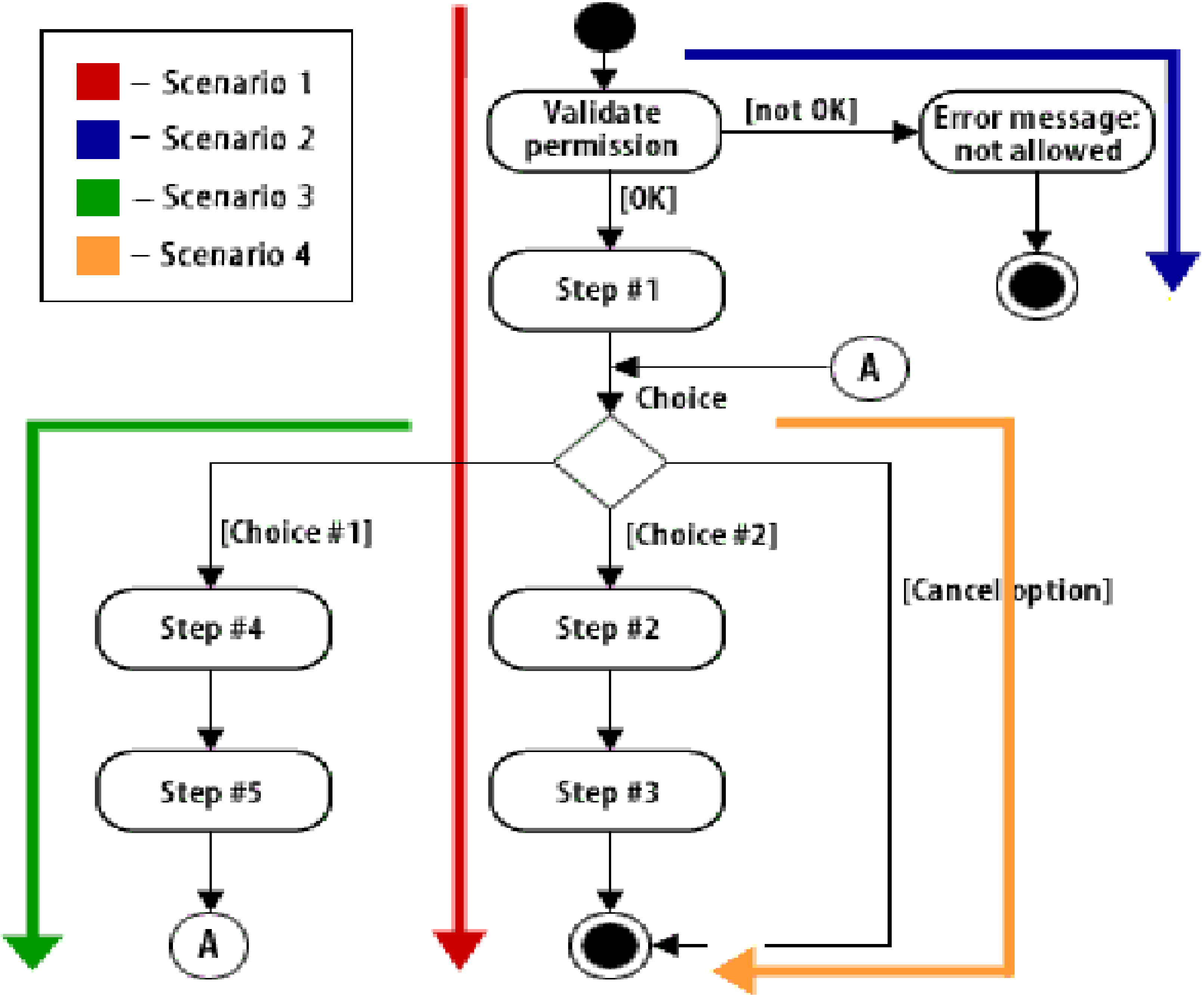
Classify events flow into main and exceptional

Present and discuss with users

Avoid Redundant Scenarios

For templates

http://sce.uhcl.edu/helm/RationalUnifiedProcess/process/templates.htm


🟥 – Scenario 1
🟦 – Scenario 2
🟩 – Scenario 3
🟧 – Scenario 4
Validate permission
[not OK]
Error message: not allowed
[OK]
Step #1
A
Choice
[Choice #1]
[Choice #2]
[Cancel option]
Step #4
Step #2
Step #5
Step #3
A

How to derive test cases from scenarios?

```
Repeat {

}Until end of all activities
```

The modeling processes: Choosing use cases on which to focus

- Often one use case (or a very small number) can be identified as *central* to the system
  - —The entire system can be built around this particular use case
- There are other reasons for focusing on particular use cases:
  - —Some use cases will represent a high *risk* because for some reason their implementation is problematic
  - —Some use cases will have high political or commercial value

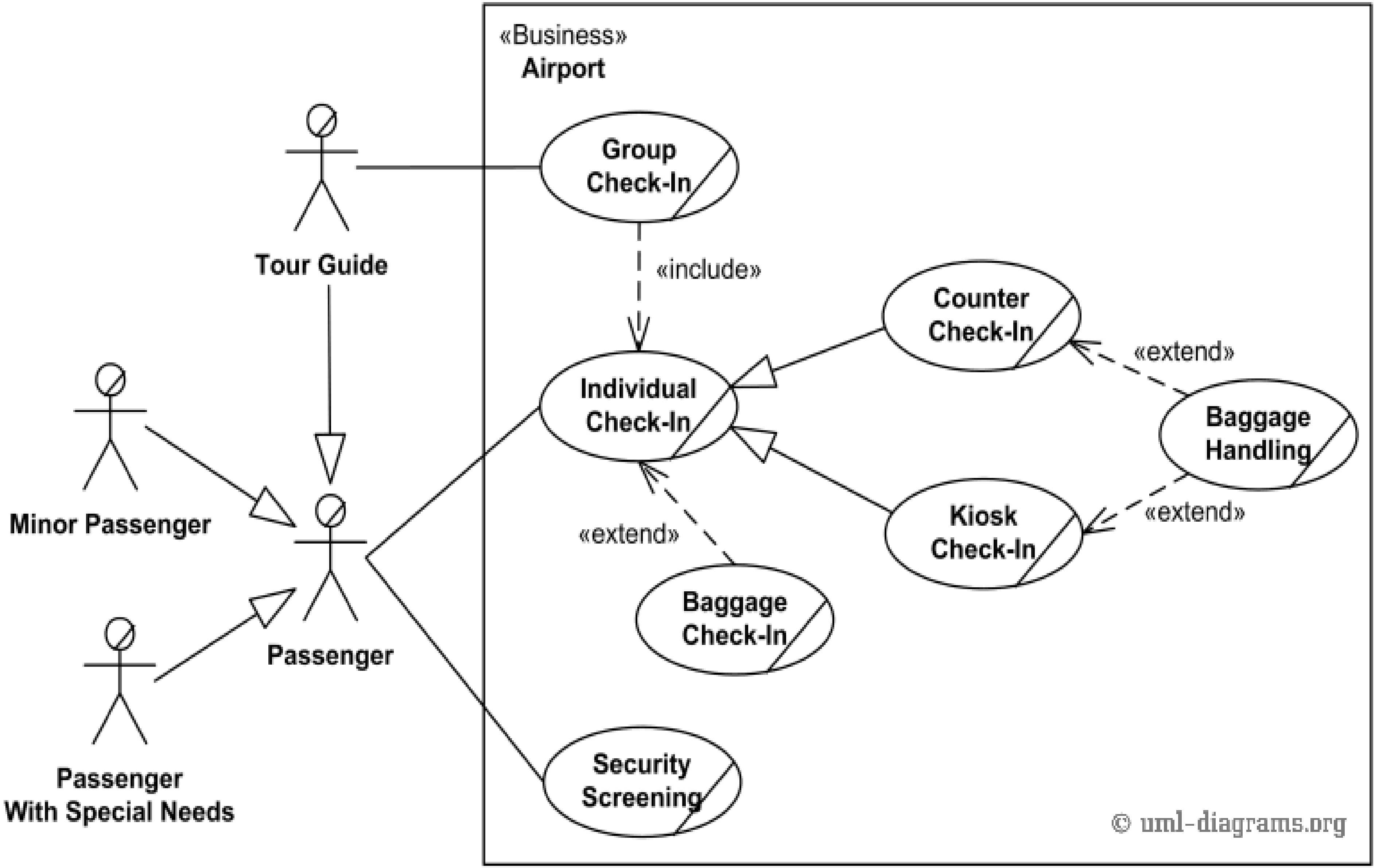
The benefits of basing software development on use cases

- They can help to define the *scope* of the system
- They are often used to *plan* the development process
- They are used to both develop and validate the requirements
- They can form the basis for the definition of testcases
- They can be used to structure user manuals

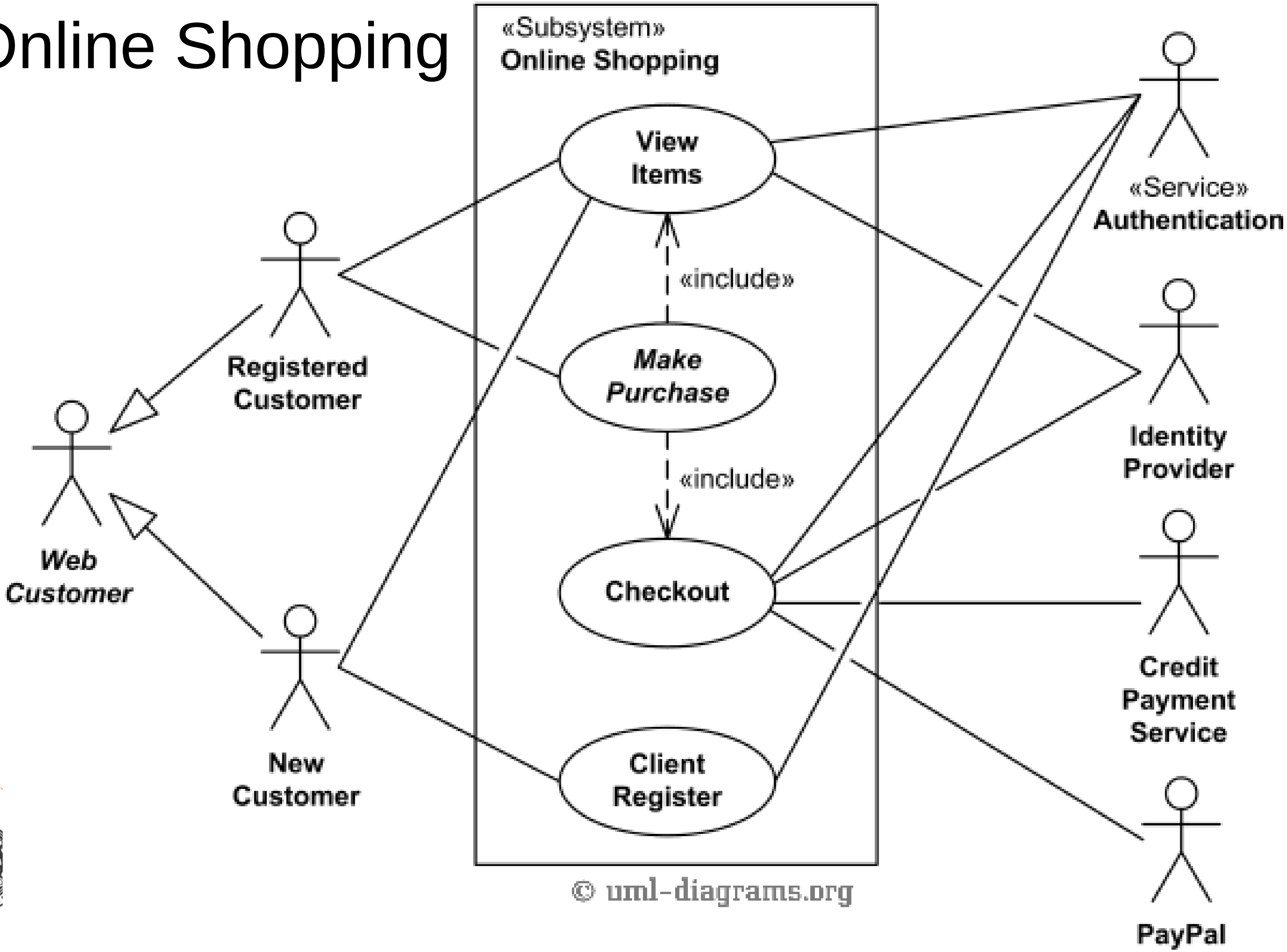
Use case limitations

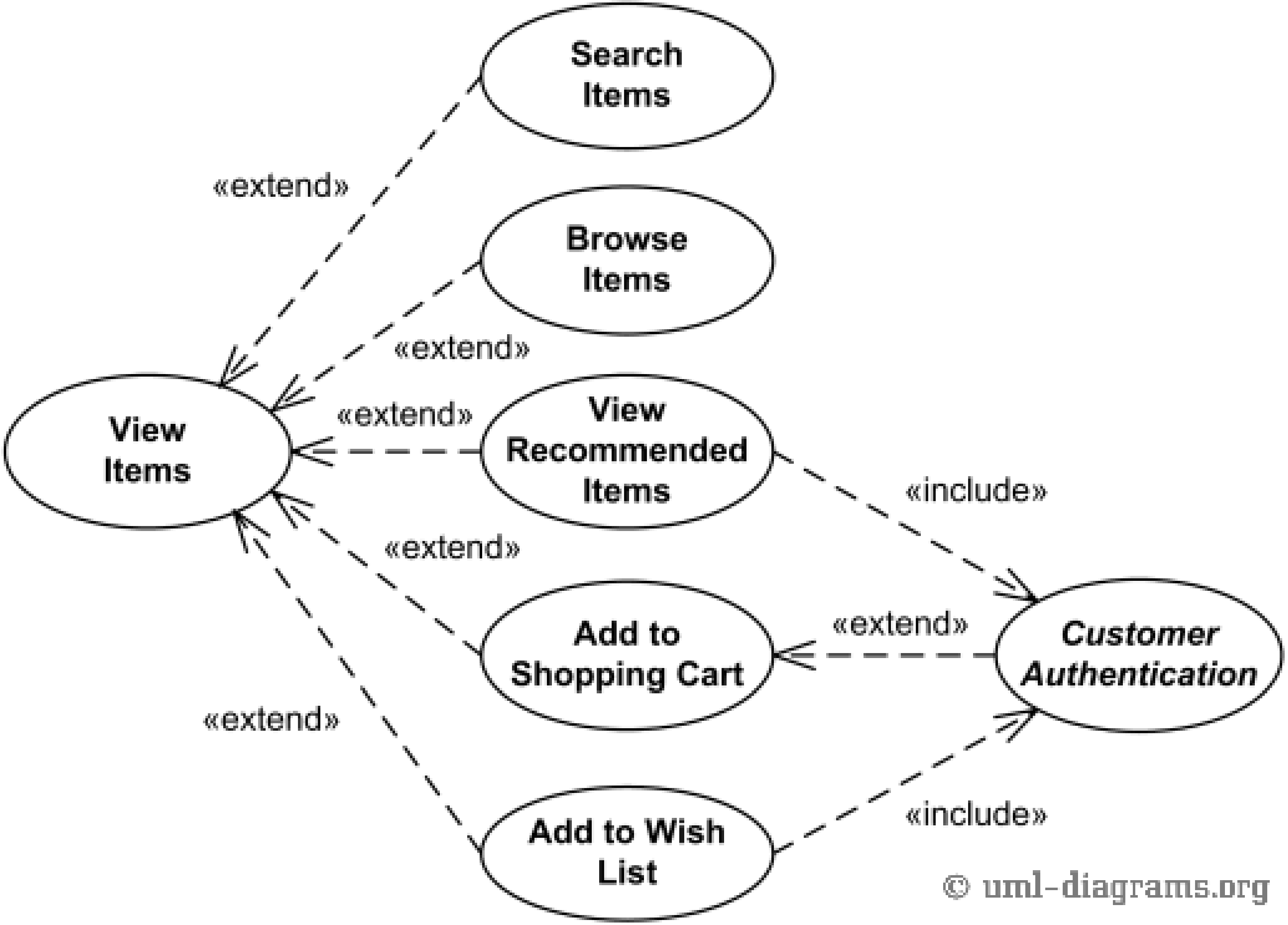
- Use cases are not enough. E.g. non-functional requirements not documented in use cases
- NOT object-oriented. Each use case captures functional abstraction based on the functional decomposition (i.e. not in line with object-oriented paradigm)
- Do not have flow? (comparing the DFD) Relationship is not enough show the flow
- Difficult to document relevant business or legal rules in use cases
- Is use case a specialization of class diagram (as described in 2.5)?

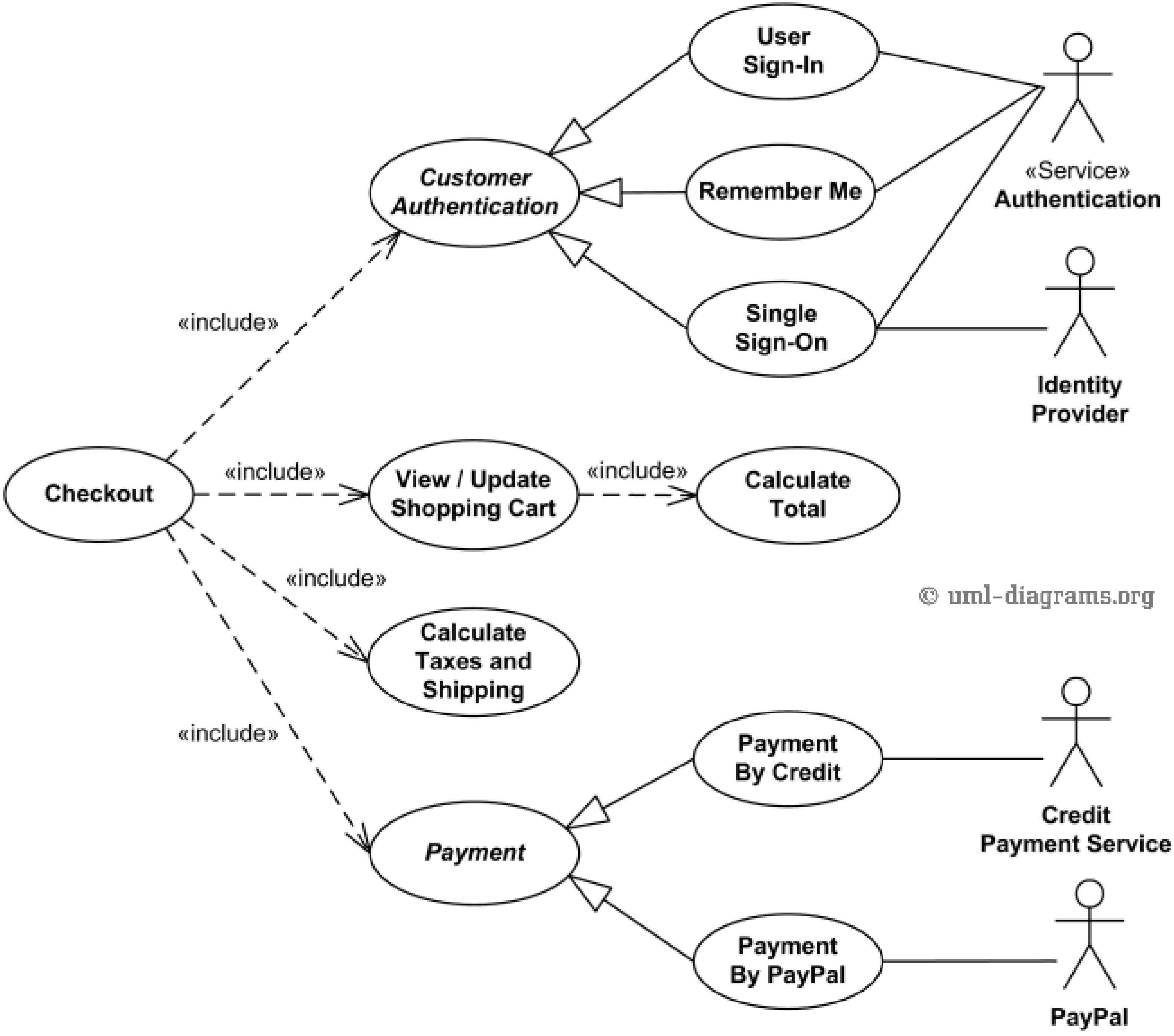
Examples: Airport Checking



Online Shopping

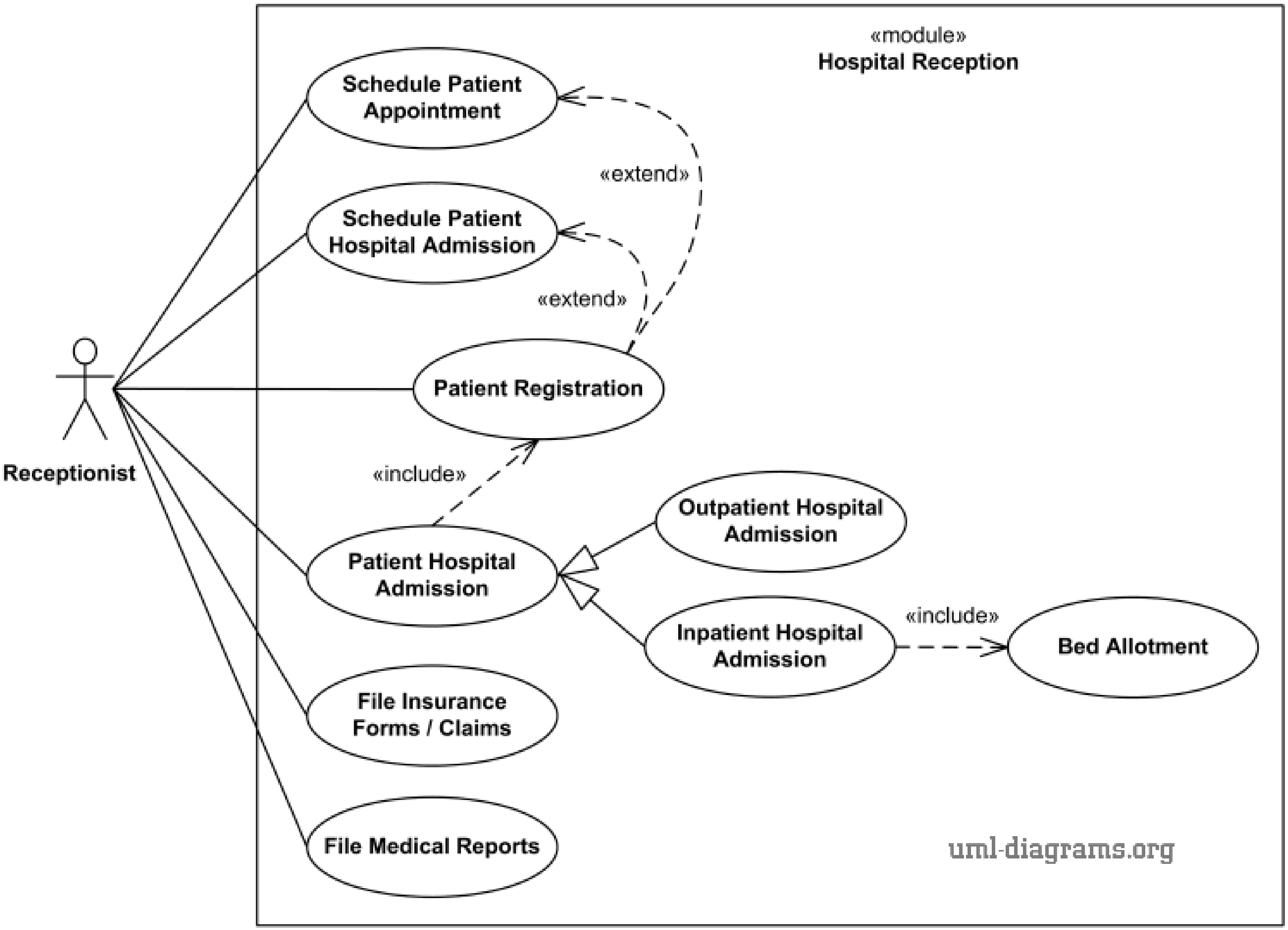



Search Items
«extend»
Browse Items
«extend»
View Items
«extend»
View Recommended Items
«include»
«extend»
Add to Shopping Cart
«extend»
Customer Authentication
«extend»
Add to Wish List
«include»
© uml-diagrams.org

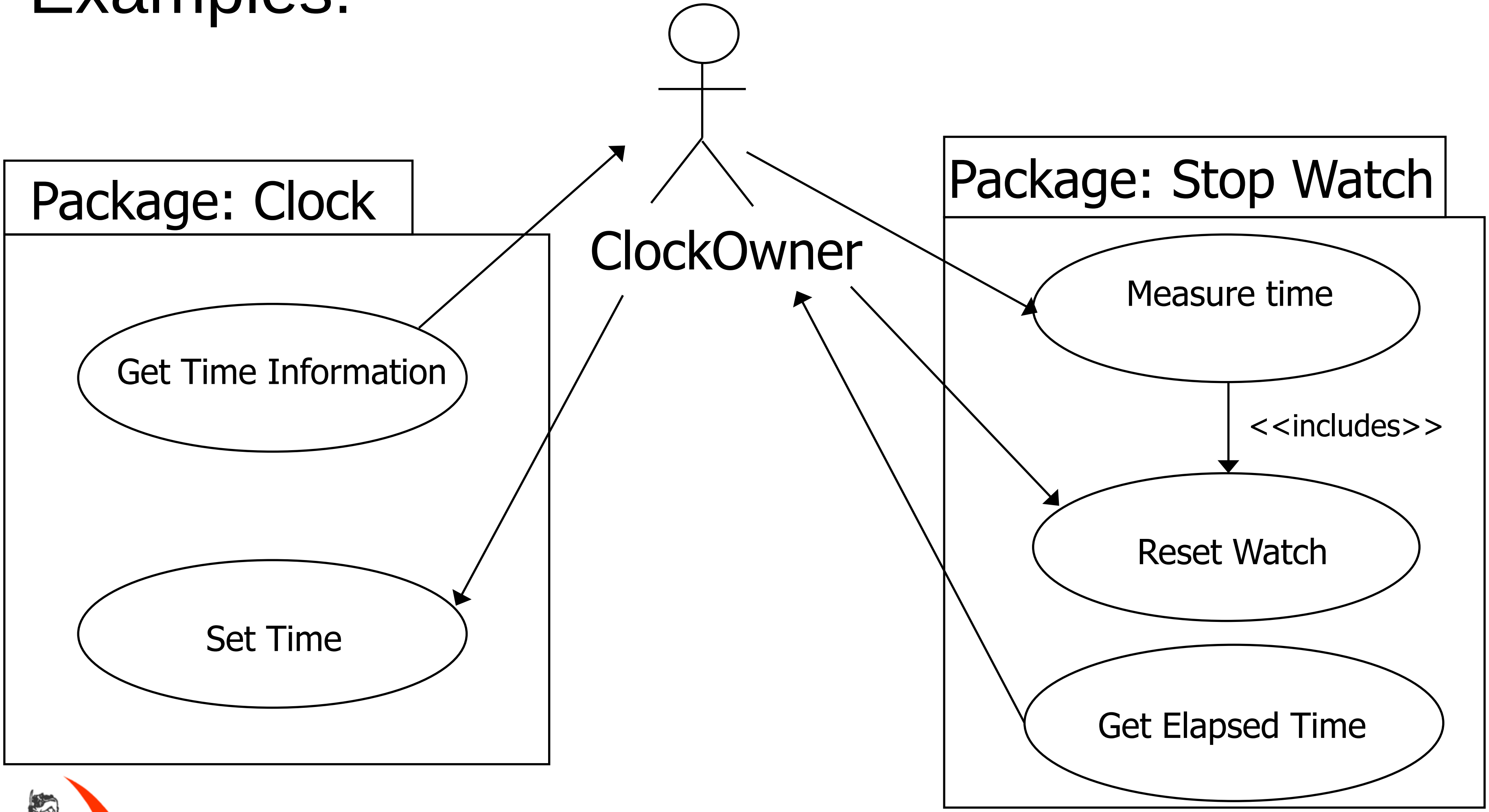


© uml-diagrams.org

Hospital Management


«module»
Hospital Reception
Schedule Patient Appointment
Schedule Patient Hospital Admission
«extend»
«extend»
Patient Registration
Receptionist
«include»
Outpatient Hospital Admission
Patient Hospital Admission
Inpatient Hospital Admission
«include»
Bed Allotment
File Insurance Forms / Claims
File Medical Reports
uml-diagrams.org

Examples:


ClockOwner
Package: Clock
Get Time Information
Set Time
Package: Stop Watch
Measure time
<<includes>>
Reset Watch
Get Elapsed Time

Use Case Description

UseCase Name	Measure time
Summary	This use case is used to time.
Actor(s)	● Clock owner
Pre-Conditions	• Stop watch is not running
Begins When	• Begins when the user hits the start button
Description	• This use case is used to time. It is performed with a stop watch object. If the stop watch is not in Zero state, the clock is automatically reset, when the start operation is called. When the stop operation is called the elapesed time between the two operations is captured in the stop watch object state.
Ends When	• Ends when the user hits the stop button
Exceptions	• Stop watch is already running
Post-Conditions	• .Elapsed time is captured
Traceability	•