




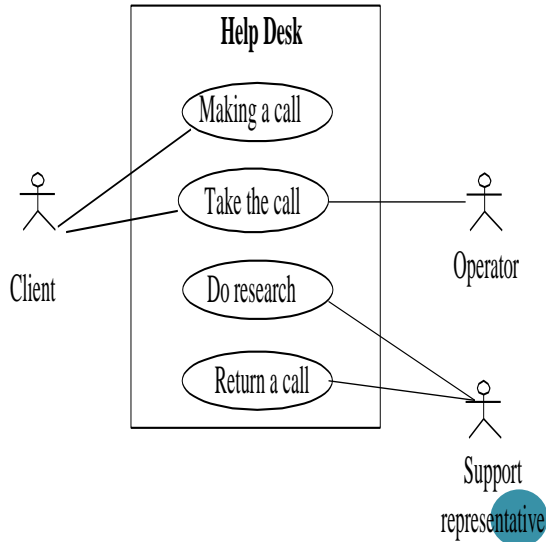
Usecase Diagram

USE-CASE DIAGRAM

- A Use Case is a description of a set of sequence of actions that a system performs that yields an observable result of value to a particular action
 - The description of a use case defines what happens in the system when the use case is performed
 - In essence, the use-case model defines the outside (actors) and inside (use case) of the system's behavior
 - An Actor is someone or something that must interact with the system. An Actor initiates the process(that is USECASE)
- 

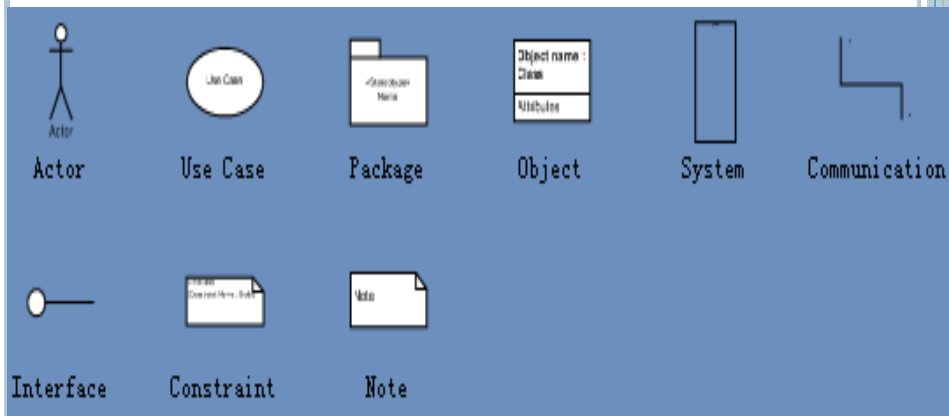
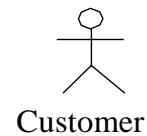
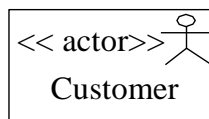
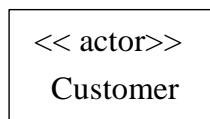
USE-CASE DIAGRAM (CON'T)

- A use-case diagram is a graph of actors, a set of use cases enclosed by a system boundary, communication (participation) associations between the actors and the use cases, and generalization among the use cases



ACTOR NOTATIONS

- The three representations of an actor are equivalent



USE-CASE DIAGRAM (CON'T)

- These relationships are shown in a use-case diagram:
 - *Communication*
 - *The communication relation of an actor in a use case is shown by connecting the actor symbol to the use-case symbol with a solid path*
 - *Uses*
 - *A uses relationship between use cases is shown by a generalization arrow from the use case*
 - *Extends*
 - *The extends relationship is used when you have one use case that is similar to another use case but does a bit more. Like a subclass*



INTRODUCTION

- A use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system.
- An effective use case diagram can help your team discuss and represent:
 - Scenarios in which your system or application interacts with people, organizations, or external systems
 - Goals that your system or application helps those entities (known as actors) achieve
 - The scope of your system






INTRODUCTION


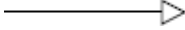
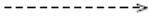
- UML use case diagrams are ideal for:
 - Representing the goals of system-user interactions
 - Defining and organizing functional requirements in a system
 - Specifying the context and requirements of a system
 - Modeling the basic flow of events in a use case



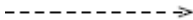
Notations

Construct	Description	Notation
Use-case	A sequence of transactions performed by a system that produces a measurable result for a particular actor	
Actor	A coherent set of roles that users play when interacting with these use cases	
System Boundary	The boundary between the physical system and the actors who interact with the physical system	<div>ApplicationName</div> 

Notations

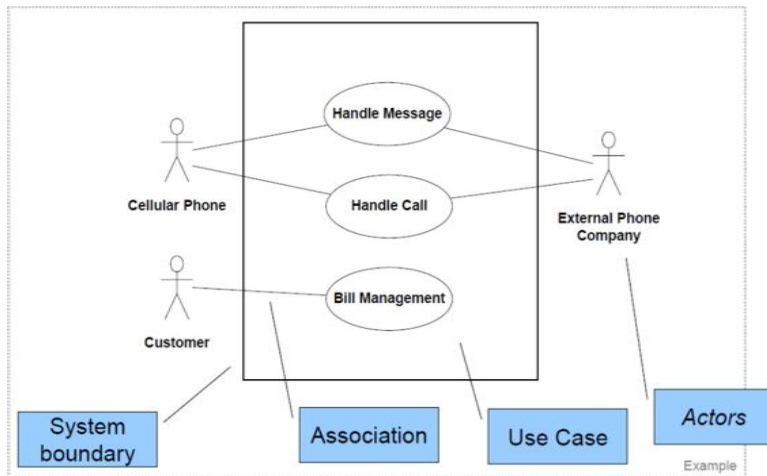
Construct	Description	Notation
Association	The participation of an actor in a use case, i.e. an instance of an actor and instances of a use case communicating with each other	
Generalization	A taxonomic relationship between a general use case and a more specific use case. The arrow head points to the general use case	
Extend	A relationship between an <i>extension use case</i> and a <i>base use case</i> , specifying how the behavior of the extension use case can be inserted into the behavior defined for the base use case. The arrow head points to the base use case	<p><<extend>></p> 

Notations

Construct	Description	Notation
Include	A relationship between a <i>base use case</i> and an <i>inclusion use case</i> , specifying how the behavior for the inclusion use case is inserted into the behavior defined for the base use case. The arrow head points to the inclusion use case	<p><<include>></p> 



Example Use-Case Diagram



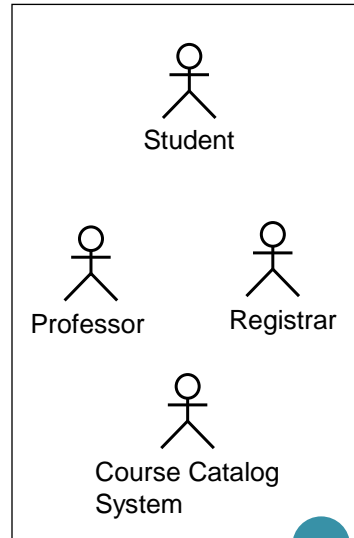
A standard form of use case diagram is defined in the Unified Modeling Language.

How to create use case diagram

1. List main system functions (use cases) in a column:
 - think of business events demanding system's response
 - users' goals/needs to be accomplished via the system
 - Create, Read, Update, Delete (CRUD) data tasks
 - Naming use cases – user's needs usually can be translated in data tasks
2. Draw ovals around the function labels
3. Draw system boundary
4. Draw actors and connect them with use cases (if more intuitive, this can be done as step 2)
5. Specify include and extend relationships between use cases (yes, at the end - not before, as this may pull you into process thinking, which does not apply in UC diagramming).

Name the actor

- Actor names should clearly convey the actor's role
- Good actor names describe their responsibilities



Describe the actor

Name

Student

Brief description A person who signs up for a course

Relationships with use cases

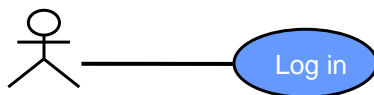


Find use cases (cont.)

- What are the goals of each actor?
 - Why does the actor want to use the system?
 - Will the actor create, store, change, remove, or read data in the system? If so, why?
 - Will the actor need to inform the system about external events or changes?
 - Will the actor need to be informed about certain occurrences in the system?
- Does the system supply the business with all of the correct behavior?

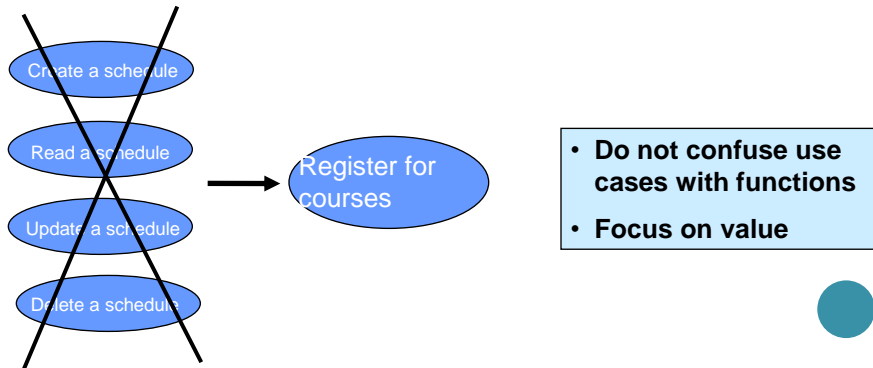
Is Log in a use case?

- By UML definition, log in is not a use case, because it does not produce results of value to an actor.
- However, in many cases, there is a need to capture log in separately because it:
 - Captures more and more complex behaviors (security, compliance, customer experience)
 - Is included in other use cases
- Recommendation: Make an exception and capture log in as a separate use case.



CRUD Use Cases

- A CRUD use case is a Create, Read, Update, or Delete use case
- Remove CRUD use cases if they are data- management use cases that do not provide results that are of value to actors



Name the use case

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

Register for courses

Select a course to teach

Guideline: Conduct a survey to learn whether customers, business representatives, analysts, and developers all understand the names and descriptions of the use cases

Describe a use case (text description)

Name

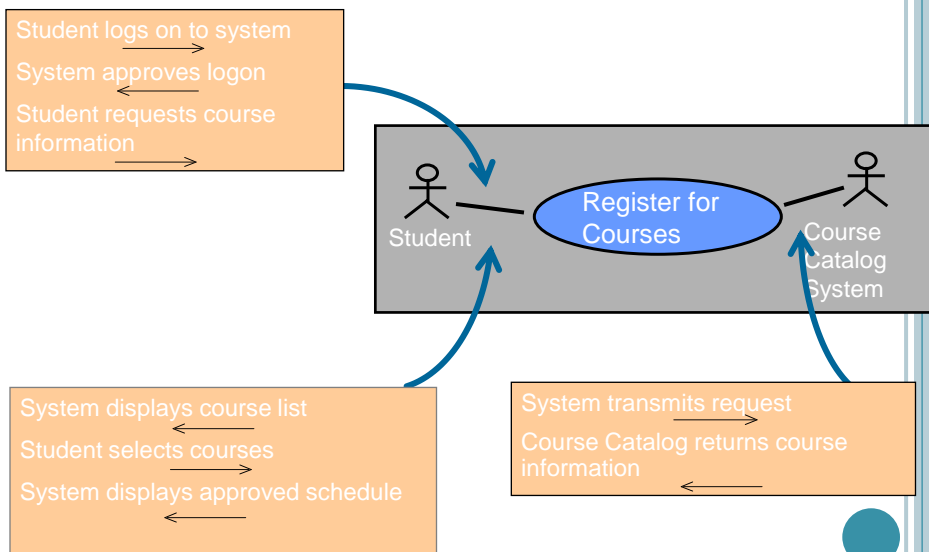
Register for Courses

Brief description The student selects the courses they wish to attend to the next semester. A schedule of primary and alternate courses is produced.

Relationships with actors



Each communicates-association is a whole dialog



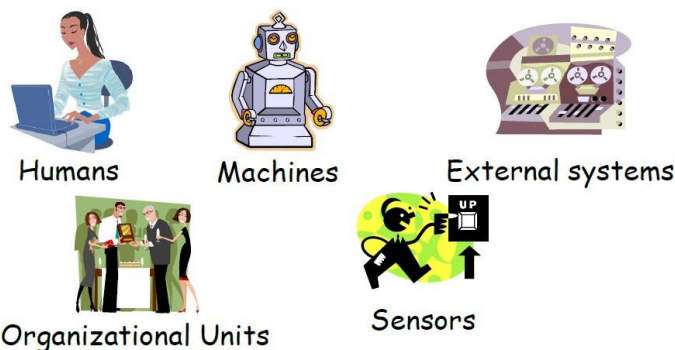
What is an Actor?

- Include all user roles that interact with the system
- Include system components only if they are responsible for initiating/triggering a use case.
 - For example, a timer that triggers sending of an e-mail reminder
- **primary** - a user whose goals are fulfilled by the system
 - importance: define user goals
- **supporting** - provides a service (e.g., info) to the system
 - importance: clarify external interfaces and protocols
- **offstage** - has an interest in the behavior but is not primary or supporting, e.g., government
 - importance: ensure all interests (even subtle) are identified and satisfied

Finding Actors

External objects that produce/consume data:

- Must serve as sources and destinations for data
- Must be external to the system



Finding Actors

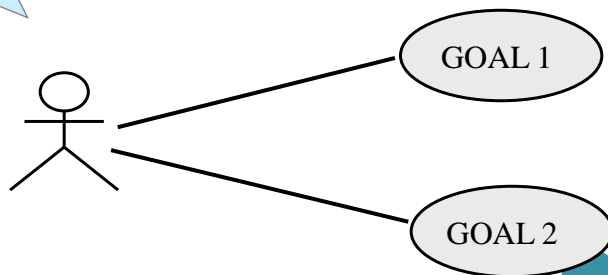
Ask the following questions:

- Who are the system's primary users?
- Who requires system support for daily tasks?
- Who are the system's secondary users?
- What hardware does the system handle?
- Which other (if any) systems interact with the system in question?
- Do any entities interacting with the system perform multiple roles as actors?
- Which other entities (human or otherwise) might have an interest in the system's output?



Find Use Cases

What goal am I trying to achieve by using the system?



Rules while drawing use-case

- The name of an actor or a use case must be meaningful and relevant to the system.
- Interaction of an actor with the use case must be defined clearly and in an understandable way.
- Annotations must be used wherever they are required.
- If a use case or an actor has multiple relationships, then only significant interactions must be displayed.



Use-Case Diagrams: Example

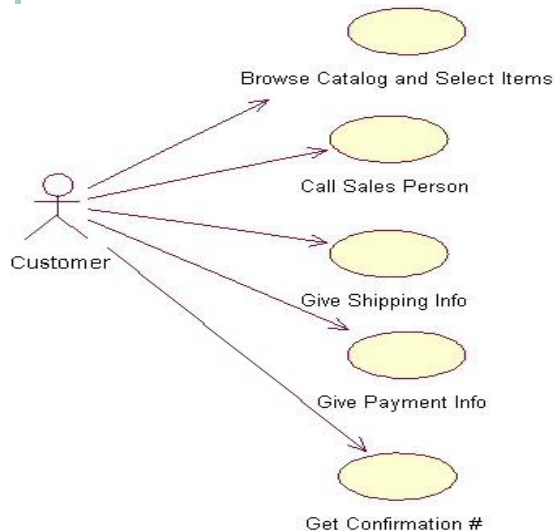
I. Begin with a Use Case!

- A user placing an order with a sales company might follow these steps :
 1. Browse catalog and select items.
 2. Call sales representative.
 3. Supply shipping information.
 4. Supply payment information.
 5. Receive conformation number from salesperson.

II. Then translate Use Case sequence into Diagram



Use-Case Diagrams: Example



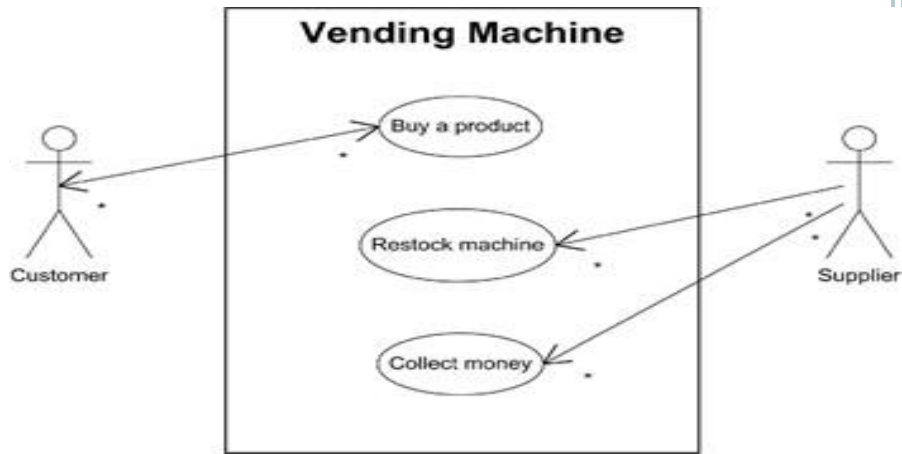
- The salesperson could also be included in this use case diagram because the salesperson is also interacting with the ordering system.

Use-Case Diagram Case Study

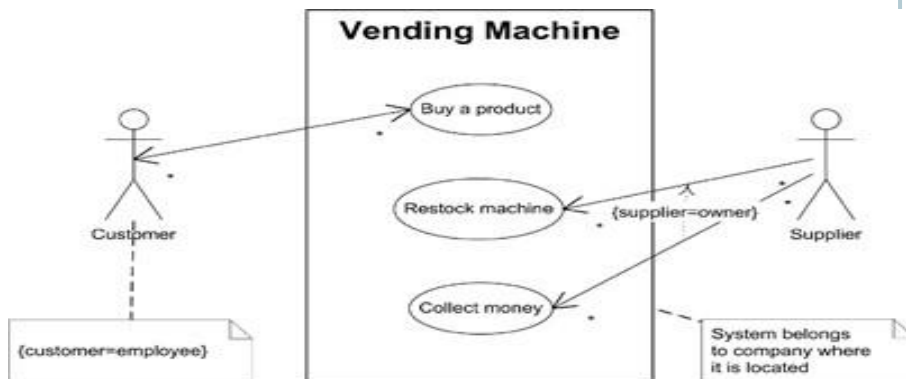
Vending Machine

- After client interview the following system scenarios were identified:
 - A customer buys a product
 - The supplier restocks the machine
 - The supplier collects money from the machine
- On the basis of these scenarios, the following three actors can be identified:
- Customer; Supplier; Collector (in this case Collector=Supplier)

Use-Case Diagram Case Study



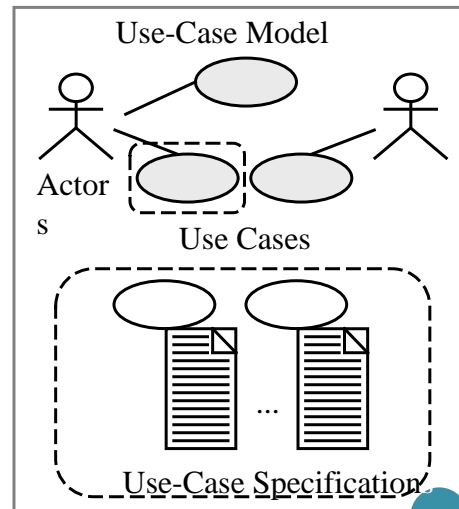
Use-Case Diagram Case Study



Introducing annotations (notes) and constraints.

Use-Case Specifications

- Name
- Brief description
- Flow of Events
- Relationships
- Activity diagrams
- Use-Case diagrams
- Special requirements
- Pre-conditions
- Post-conditions
- Other diagrams



A Scenario Is a Use-Case Instance



Scenario 1

Log on to system.
 Approve log on.
 Enter subject in search.
 Get course list.
 Display course list.
 Select courses.
 Confirm availability.
 Display final schedule.

Scenario 2

Log on to system.
 Approve log on.
 Enter subject in search.
Invalid subject.
Re-enter subject.
 Get course list.
 Display course list.
 Select courses.
 Confirm availability.
 Display final schedule.

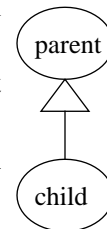
Linking Use Cases

- *Association* relationships
- *Generalization* relationships
 - One element (child) "is based on" another element (parent)
- *Include* relationships
 - One use case (base) includes the functionality of another (inclusion case)
 - Supports re-use of functionality
- *Extend* relationships
 - One use case (extension) extends the behavior of another (base)



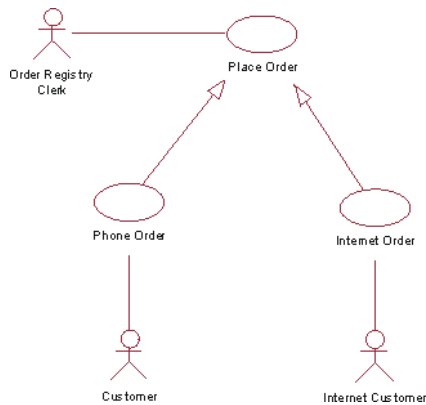
Generalization

- Generalization Relationship
 - Represented by a line and a hollow arrow from child to parent
 - Inheritance :A use case generalization shows that one use case is simply a special kind of another.
 - The child use case inherits the behavior and meaning of the parent use case.
 - The child may add to or override the behavior of its parent
 - A clinic may have a use case such as Pay Bill which would be a parent use case and Bill Insurance which is the child.
 - A child can be substituted for its parent whenever necessary.

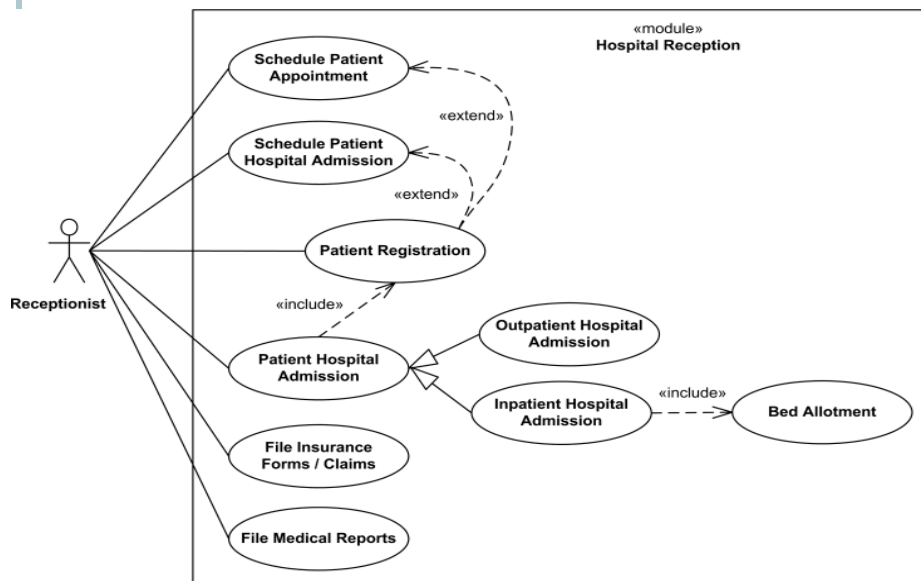


Generalization Example

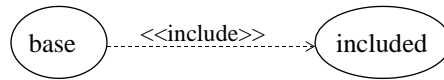
- The actor Order Registry Clerk can instantiate the general use case Place Order.
- Place Order can also be specialized by the use cases Phone Order or Internet Order.



Generalization Example



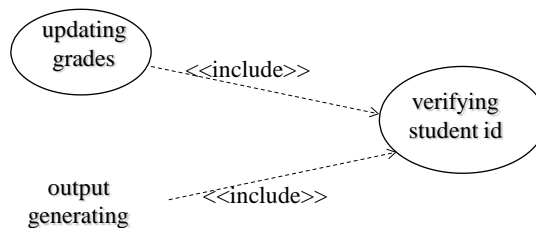
Include



- The base use case explicitly incorporates the behavior of another use case at a location specified in the base.
- The included use case never stands alone. It only occurs as a part of some larger base that includes it.

More about Include

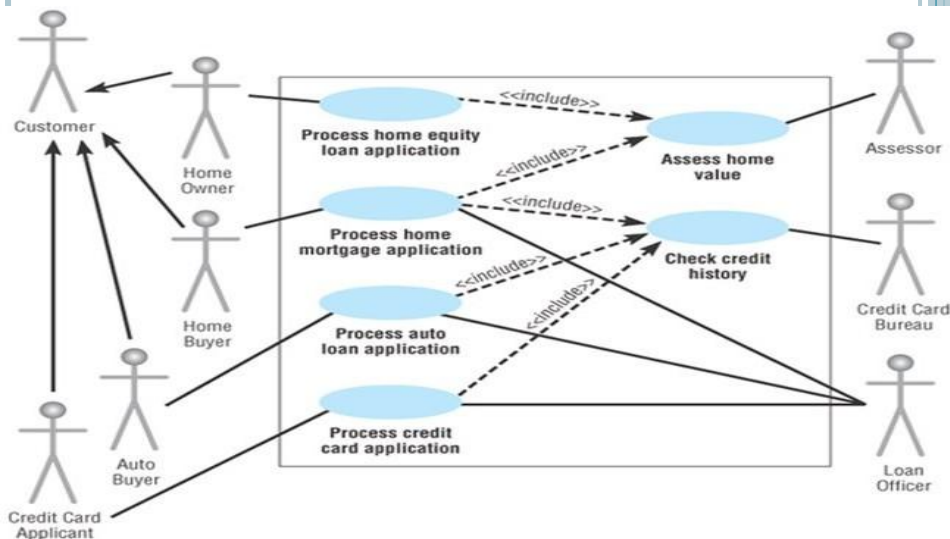
- Enables us to avoid describing the same flow of events several times by putting the common behavior in a use case of its own.



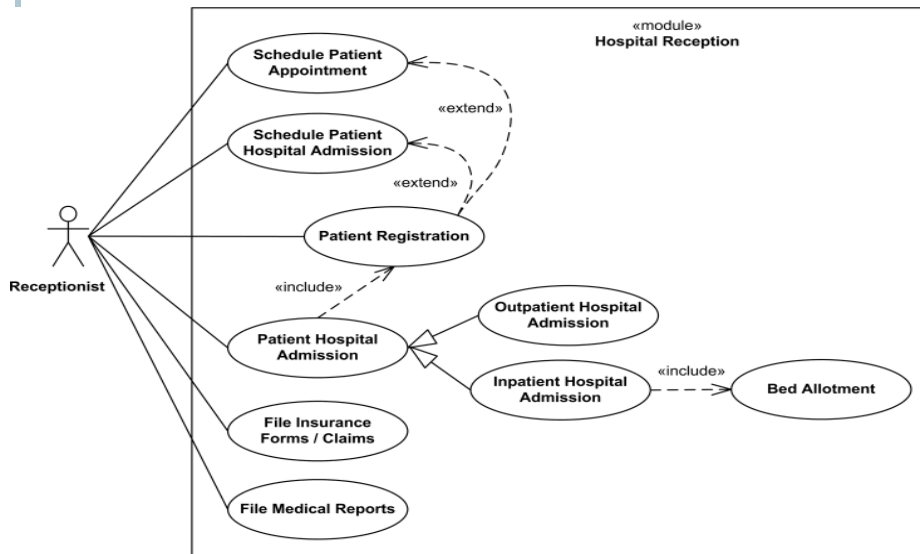
Include relationship

- Include relationship – a standard case linked to a **mandatory** use case.
- Example: to Authorize Car Loan (standard use case), a clerk must run Check Client's Credit History (include use case).
- The standard UC includes the mandatory UC (use the verb to figure direction arrow).
- Standard use case can NOT execute without the include case → **tight coupling**.

Reading use case diagram with Include relationship



Include Example



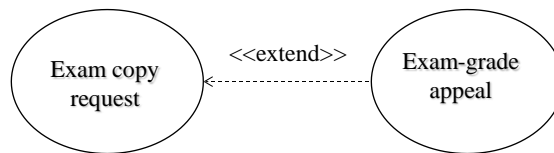
Extend



- The base use case implicitly incorporates the behavior of another use case at certain points called extension points.
- The base use case may stand alone, but under certain conditions its behavior may be extended by the behavior of another use case.

More about Extend

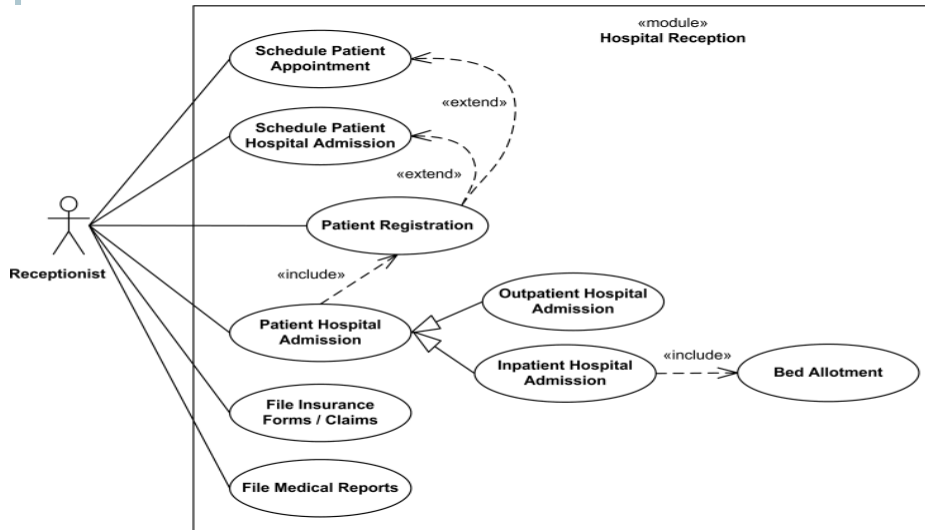
- Enables to model optional behavior or branching under conditions.



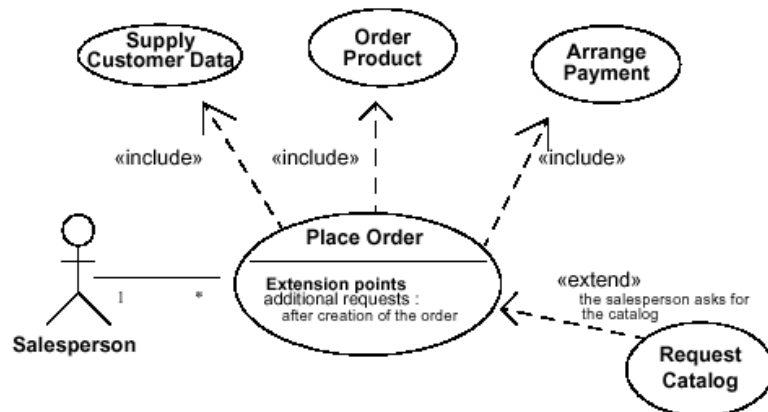
Extend relationship

- Extend relationship – linking an optional use case to a standard use case.
- Example: Register Course (standard use case) may have Register for Special Class (extend use case) – class for non-standard students, in unusual time, with special topics, requiring extra fees...).
- The optional UC extends the standard UC
- Standard use case can execute without the extend case → loose coupling.

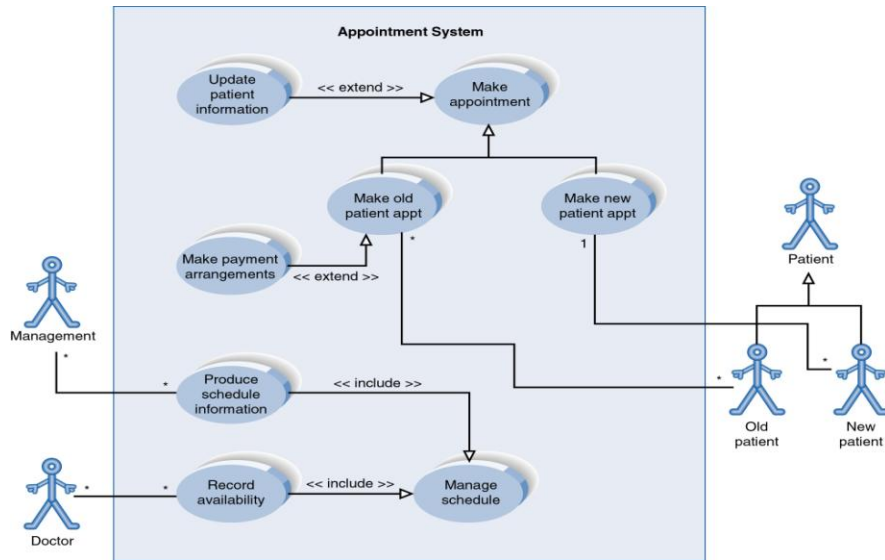
Extend Example



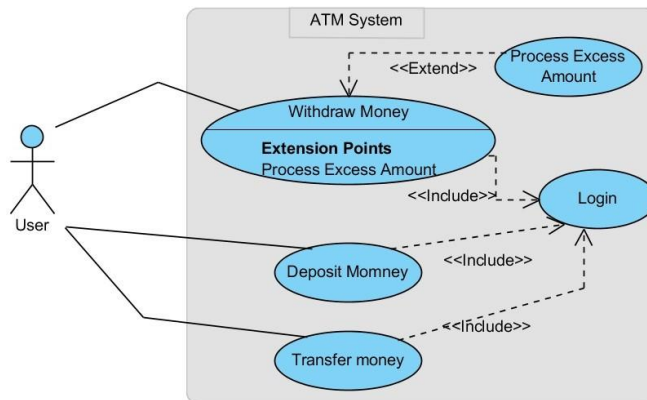
Example Use Case Diagram



Example of Relationships



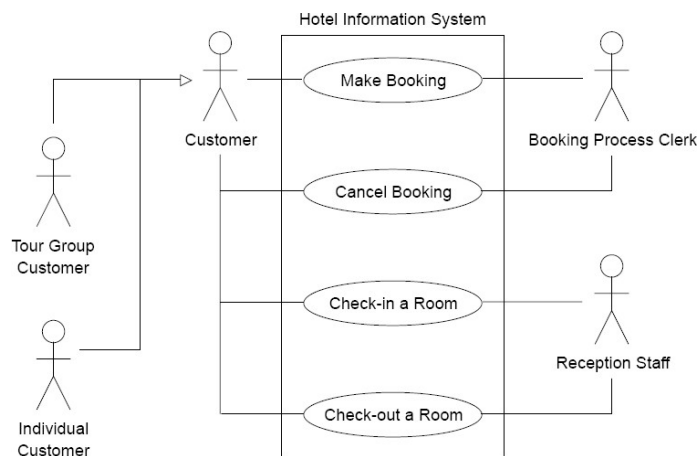
Use-Case Scope



Use-Case Diagram: Example

- Let us consider a simple hotel information system for two types of customers : Tour customer and Individual Customers
- Tour Group customers are those who have made reservations through a tour operator in advance, while Individual customers make their reservations directly with the hotel
- Both types of customers can book, cancel, check-in and check-out of a room by phone or via the Internet

Use-Case Diagram: Example

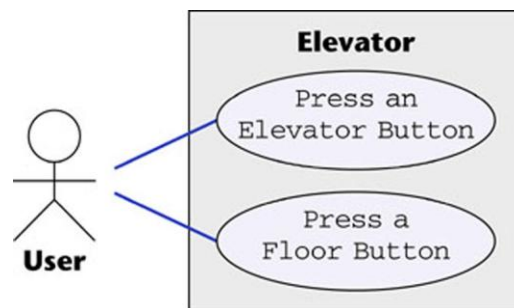


The Elevator Problem Case Study

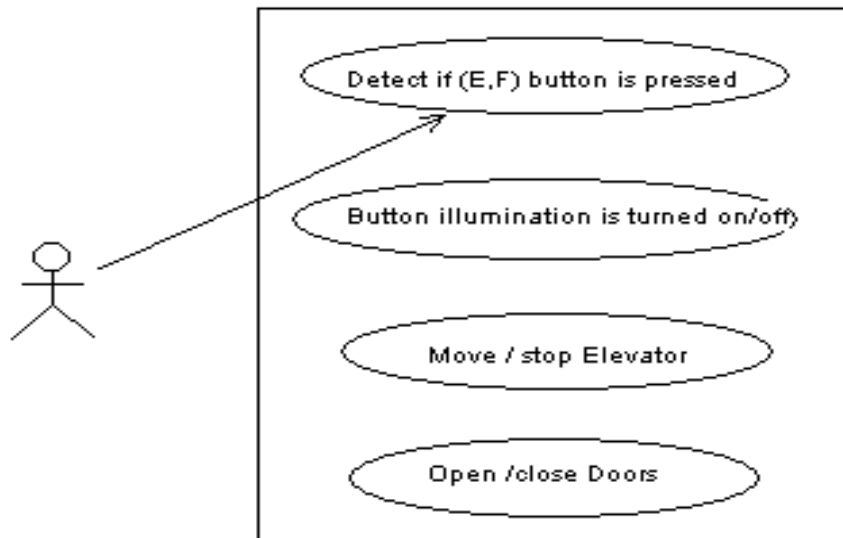
- A product is to be installed to control n elevators in a building with m floors. The problem concerns the logic required to move elevators between floors according to the following constraints:
 1. Each elevator has a set of m buttons, one for each floor. These illuminate when pressed and cause the elevator to visit the corresponding floor. The illumination is cancelled when the corresponding floor is visited by the elevator
 2. Each floor, except the first and the top floor, has two buttons, one to request an up-elevator, one to request a down-elevator. These buttons illuminate when pressed. The illumination is cancelled when an elevator visits the floor, then moves in the desired direction
 3. If an elevator has no requests, it remains at its current floor with its doors closed

Use Cases

- For the elevator problem, there are only two possible use cases
 - Press an Elevator Button, and
 - Press a Floor



Use Cases Elevator



Normal Scenario: Elevator Problem

1. User A presses the Up floor button at floor 3 to request an elevator. User A wishes to go to floor 7.
2. The Up floor button is turned on.
3. An elevator arrives at floor 3. It contains User B, who has entered the elevator at floor 1 and pressed the elevator button for floor 9.
4. The Up floor button is turned off.
5. The elevator doors open.
6. The timer starts.
User A enters the elevator.
7. User A presses the elevator button for floor 7.
8. The elevator button for floor 7 is turned on.
9. The elevator doors close after a timeout.
10. The elevator travels to floor 7.
11. The elevator button for floor 7 is turned off.
12. The elevator doors open to allow User A to exit from the elevator.
13. The timer starts.
User A exits from the elevator.
14. The elevator doors close after a timeout.
15. The elevator proceeds to floor 9 with User B.

Exception Scenario: Elevator Problem

1. User A presses the Up floor button at floor 3 to request an elevator. User A wishes to go to floor 1.
2. The Up floor button is turned on.
3. An elevator arrives at floor 3. It contains User B, who has entered the elevator at floor 1 and pressed the elevator button for floor 9.
4. The Up floor button is turned off.
5. The elevator doors open.
6. The timer starts.
User A enters the elevator.
7. User A presses the elevator button for floor 1.
8. The elevator button for floor 1 is turned on.
9. The elevator doors close after a timeout.
10. The elevator travels to floor 9.
11. The elevator button for floor 9 is turned off.
12. The elevator doors open to allow User B to exit from the elevator.
13. The timer starts.
User B exits from the elevator.
14. The elevator doors close after a timeout.
15. The elevator proceeds to floor 1 with User A.

Use cases & Usage Scenarios

- A collection of user scenarios that describe the thread of usage of a system
- Each scenario is described from the point-of-view of an “actor”
- An actor is a person or device that interacts with the software
- Each scenario answers the following questions
 - Who is the primary actor, the secondary actor (s)?
 - What are the actor’s goals?
 - What preconditions should exist before the story begins?
 - What main tasks or functions are performed by the actor?
 - What extensions might be considered as the story is described?
 - What variations in the actor’s interaction are possible?
 - What information

Use cases & Usage Scenarios

- What system information will the actor acquire, produce, or change?
- Will the actor have to inform the system about changes in the external environment?
- Does the actor wish to be informed about unexpected changes?
- Scenarios are created by user researchers to help communicate with the design team.
- User stories are created by project/product managers to define the requirements prior to a sprint in agile development.
- Scenarios are stories that capture the goals, motivations, and tasks of a persona in a given system.

