A Use Case Primer

- Use cases basics
- Benefits of use cases
- Steps of building use case model
- Use cases, storyboarding, and user interface design

What is a Software Requirement?

It is a software capability that

- is needed by the user to solve a problem to achieve an objective, and
- must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documentation

Requirements Types

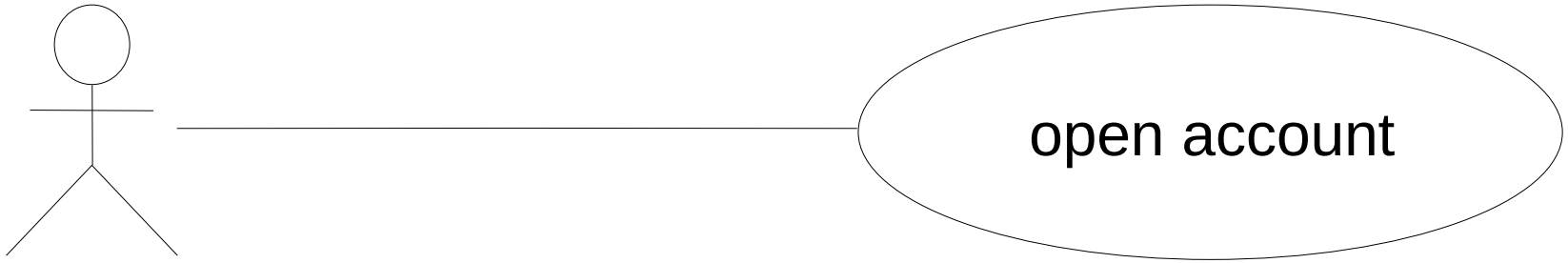
- There are two categories of requirements
- A *functional requirement* specifies an action that the software product must be able to perform
  - Often expressed in terms of inputs and outputs
- A *nonfunctional requirement* specifies properties of the software product itself, such as
  - Platform constraints
  - Response times
  - Reliability

Use Case Basics

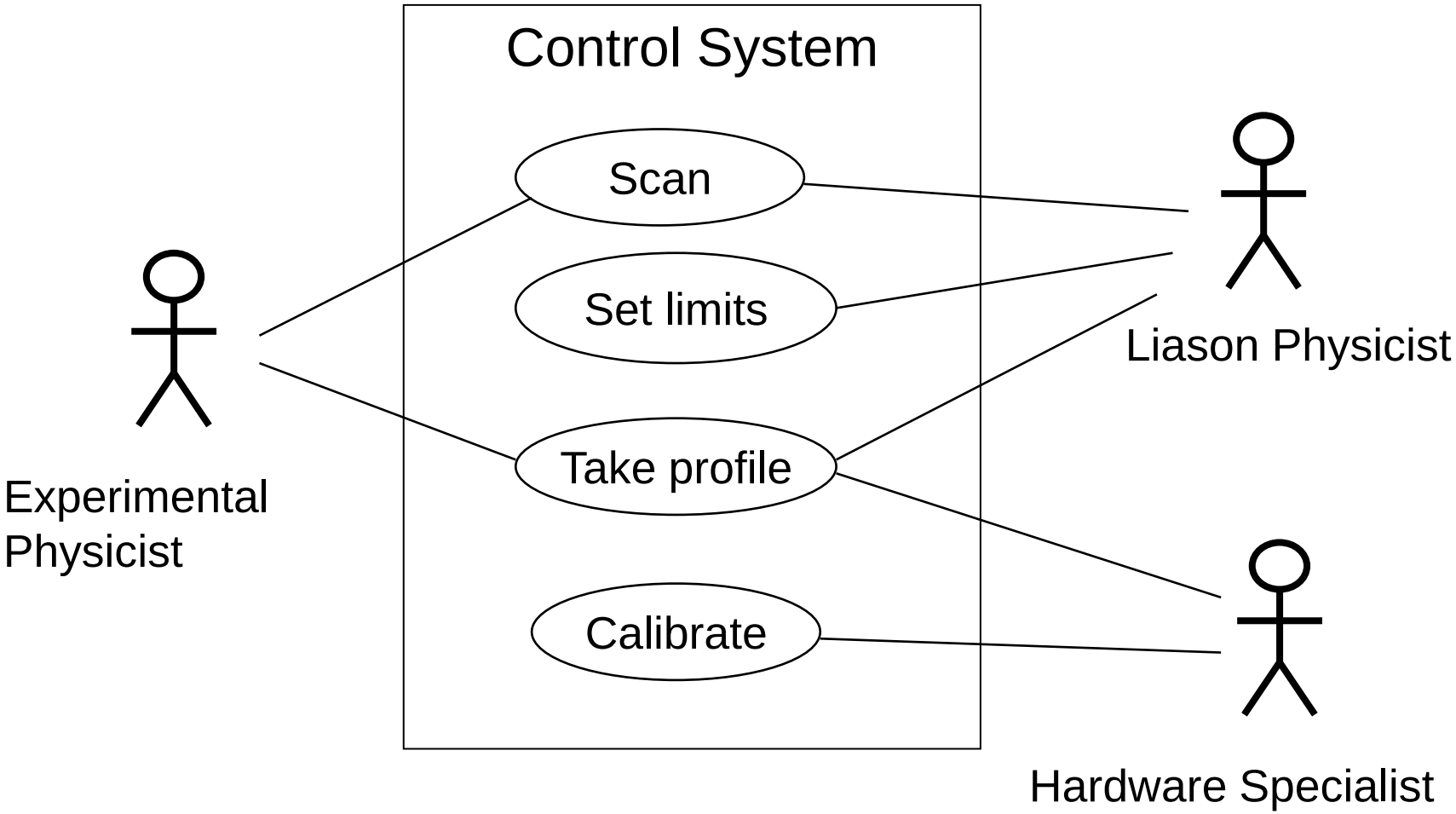
- A use case describes a sequence of actions the system performs that yield an observable result of value to a particular actor.
- An actor is someone or something that interacts with the system.
  - Users, other systems, or devices
- Documenting use cases by using
  - Use case templates: Name, descriptions,etc.
  - Use case Diagrams

UML use case diagrams

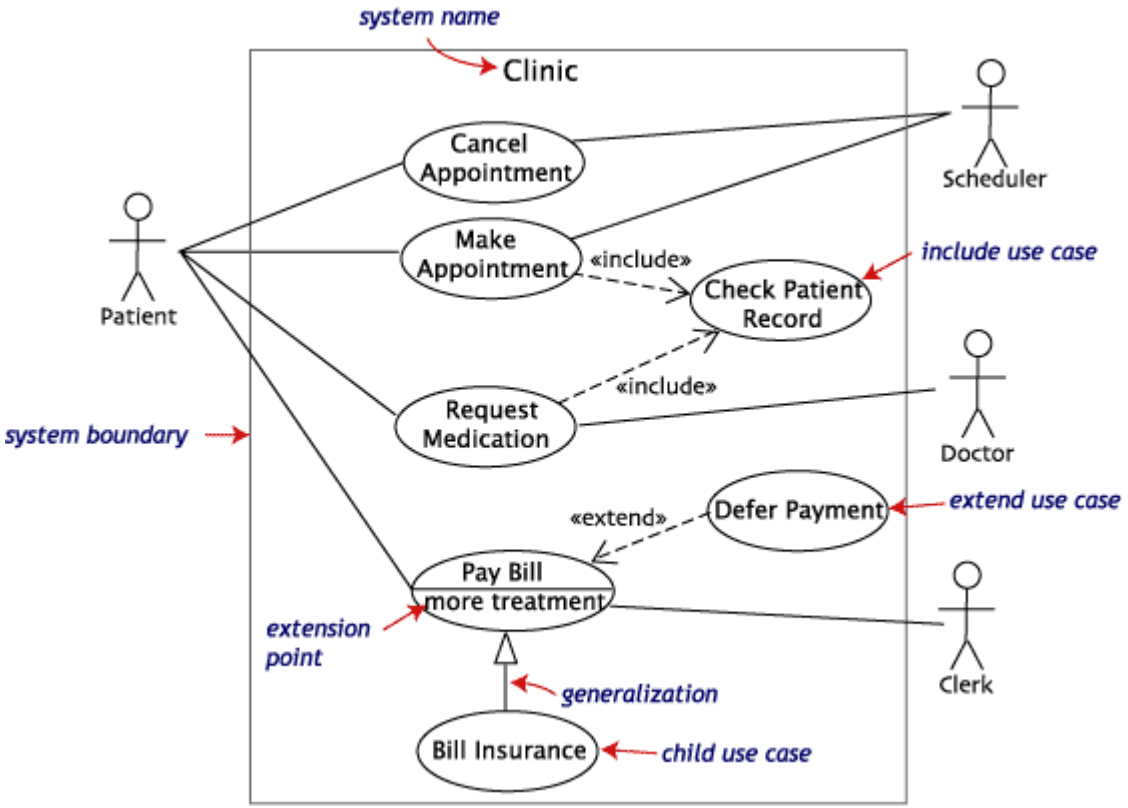
- use cases can be drawn as diagrams, with:
  - actors as stick-men, with their names below
  - use cases as ellipses with their names below or inside
  - association indicated by lines, connecting an actor to a use case in which that actor participates
  - use cases can be connected to other cases that they use / rely on(include/extend)

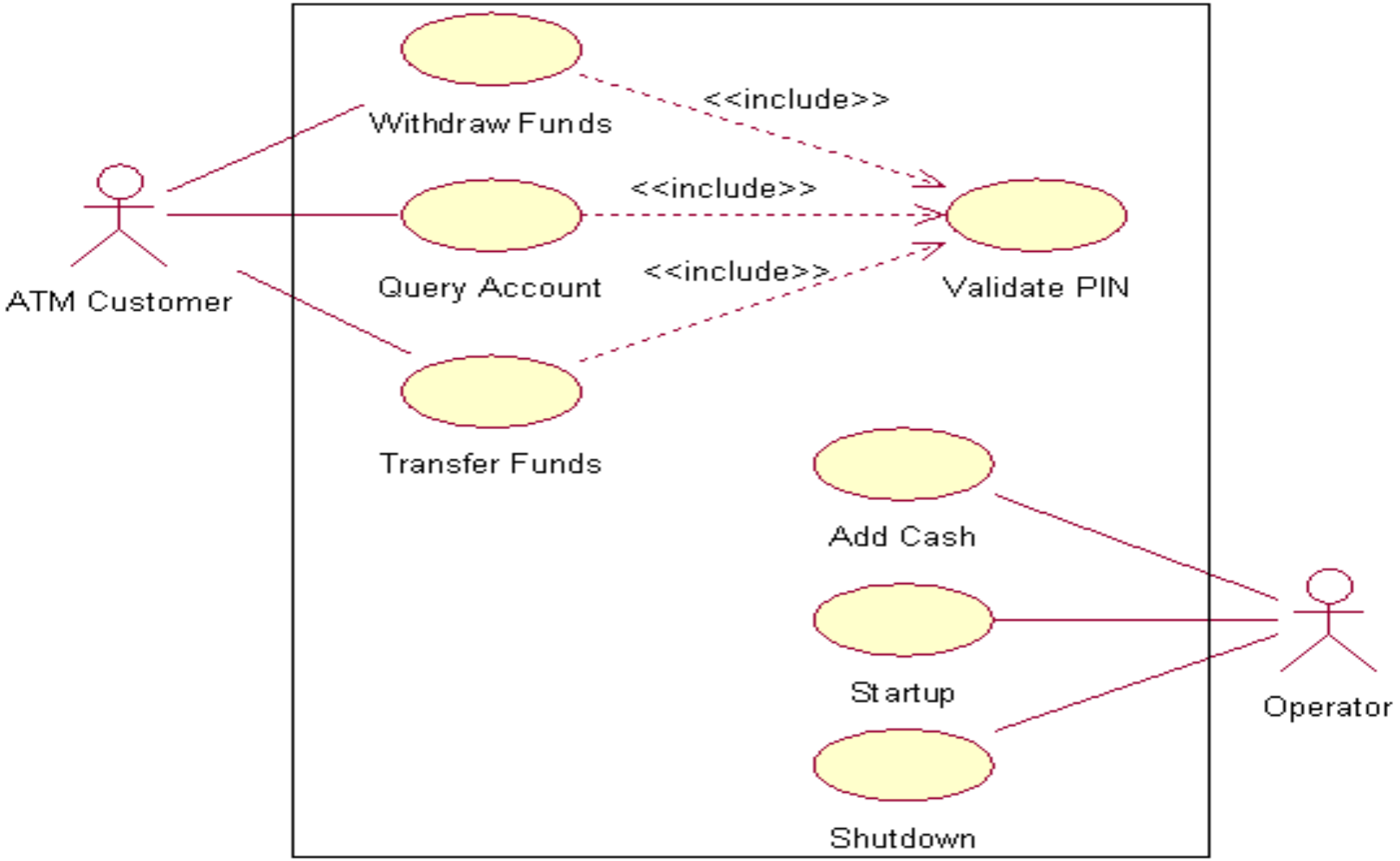


Example : Project level Use Case diagram



Hospital Use Case




Withdraw Funds
<<include>>
<<include>>
<<include>>
ATM Customer
Query Account
Validate PIN
Transfer Funds
Add Cash
Startup
Shutdown
Operator

Use Case Basics

- Documenting use cases by using
- UML Diagrams
- use case templates:
  - Name
  - Description
  - Actor(s)
  - Flow of events
  - Pre-conditions
  - Post-conditions
  - Etc ...

Use Case Model

- Individual use case describes how a particular actor interacts with the system to achieve a result of value to the specific actor.
- The set of all use cases together describes the complete behavior of the system.
- The complete set of use cases, actors, and their interactions constitutes the use-case model for the system.

Building a uses case model

- write individual use cases and then add them all .. Not a good idea.
- Instead, build a context model of the system and successively refine it.
- That's better for understanding, communicating, and refining the behavior of the system in an iterative development.

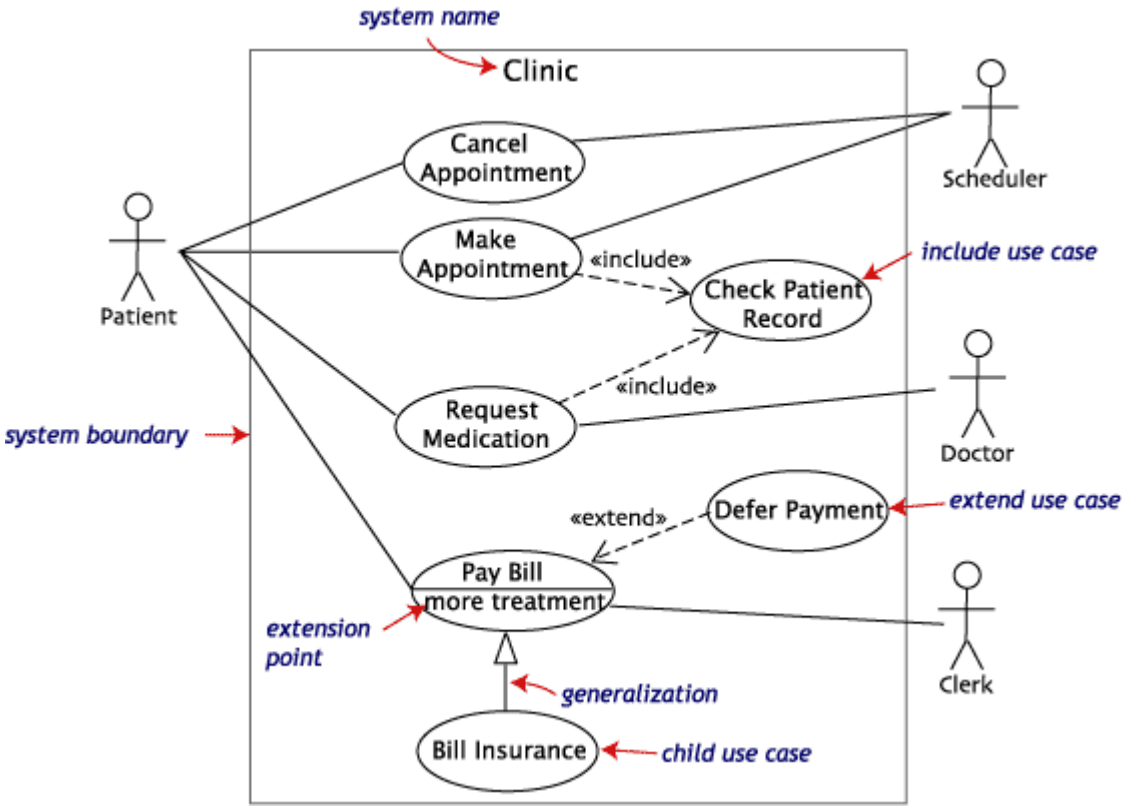
Building the Use-Case Model

- Step 1: Identify and Describe the Actors
- Step 2: Identify the Use Cases and Write a Brief Description
- Step 3: Identify the Actor and Use-Case Relationships
- Step 4: Outline the Individual Use Cases
- Step 5: Refine the Use Cases

Step 1: Identify and Describe the Actors

- Who uses the system?
- Who gets info from the system?
- Who provides info to the system?
- Where in the company is the system used?
- Who supports the and maintain the system?
- What other systems use this system?

Hospital Use Case



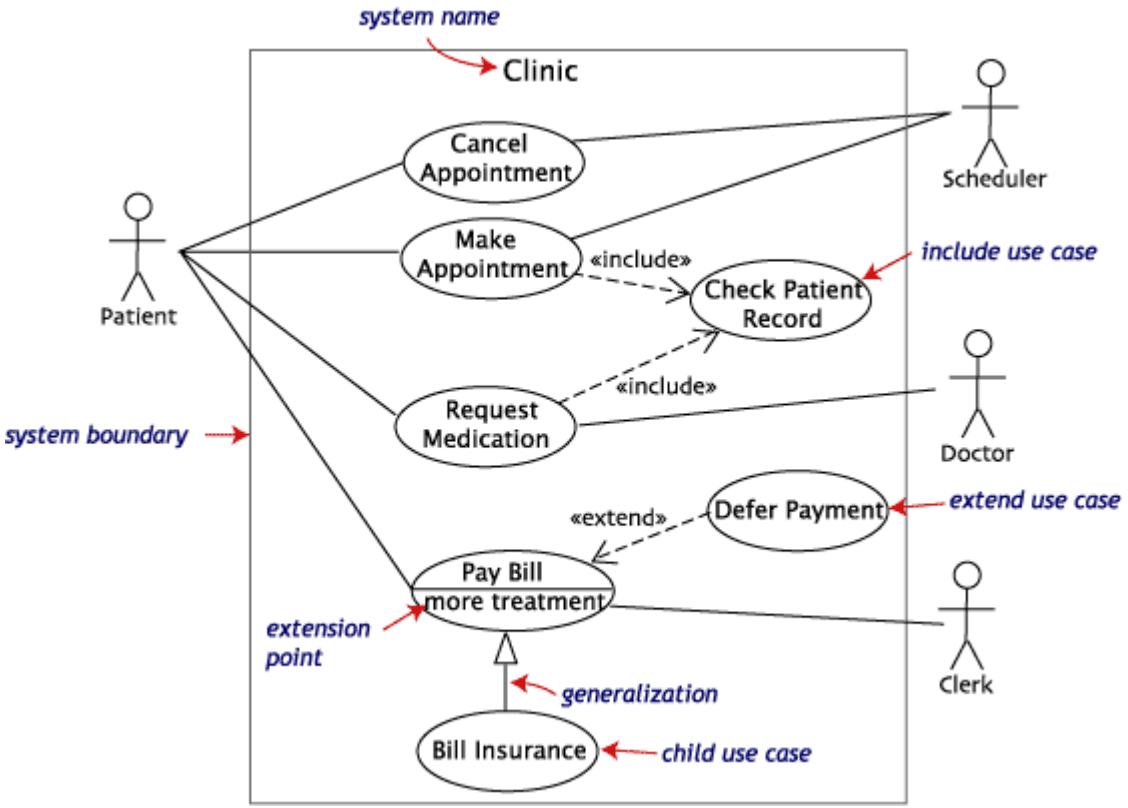
Step 2: Identify the Use Cases and Write a Brief Description

- Typically, the use-case name is a few words or a short phrase that starts with an action verb and communicates what the actor achieves with the use case.
- To identify the use cases, ask the following for each actor:

Step 2: Identify the Use Cases and Write a Brief Description

- What will the actor use the system for?
- Will the actor create, store, change, remove, or read data in the system?
- 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?

Hospital Use Case

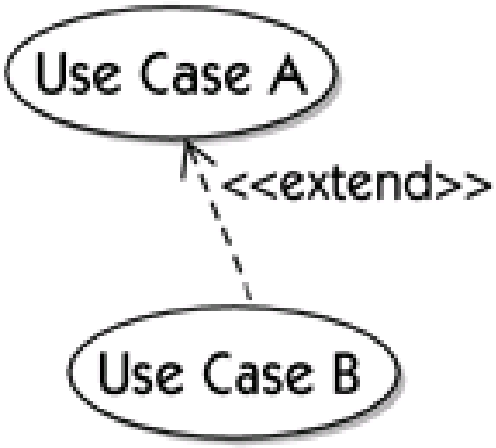


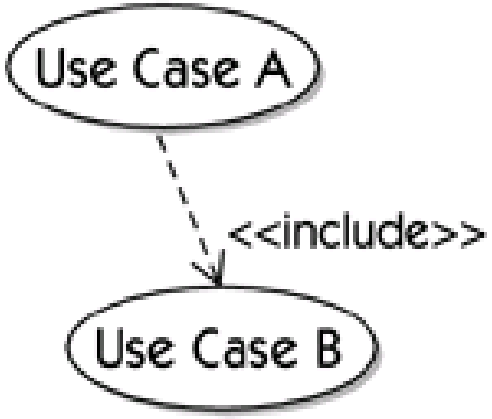
Step 3: Identify the Actor and Use-Case Relationships

- Only one actor can initiate a use case
- However, many actors can be involved in a use case.
- Each use case is analyzed to see what actors interact with it and
- Each actor's behavior is reviewed to make sure that all of the results he needs to see are achieved the system.
- If this becomes complex, uses diagrams.

Dependency Relationships between Use Cases

- Extend relationship defines that instances of a use case that may be augmented by some additional behaviour in an extended use case.
- Include relationship is a directed relationship between use cases, implying that the behaviour in the additional use case is inserted into the behaviour of the base use case.





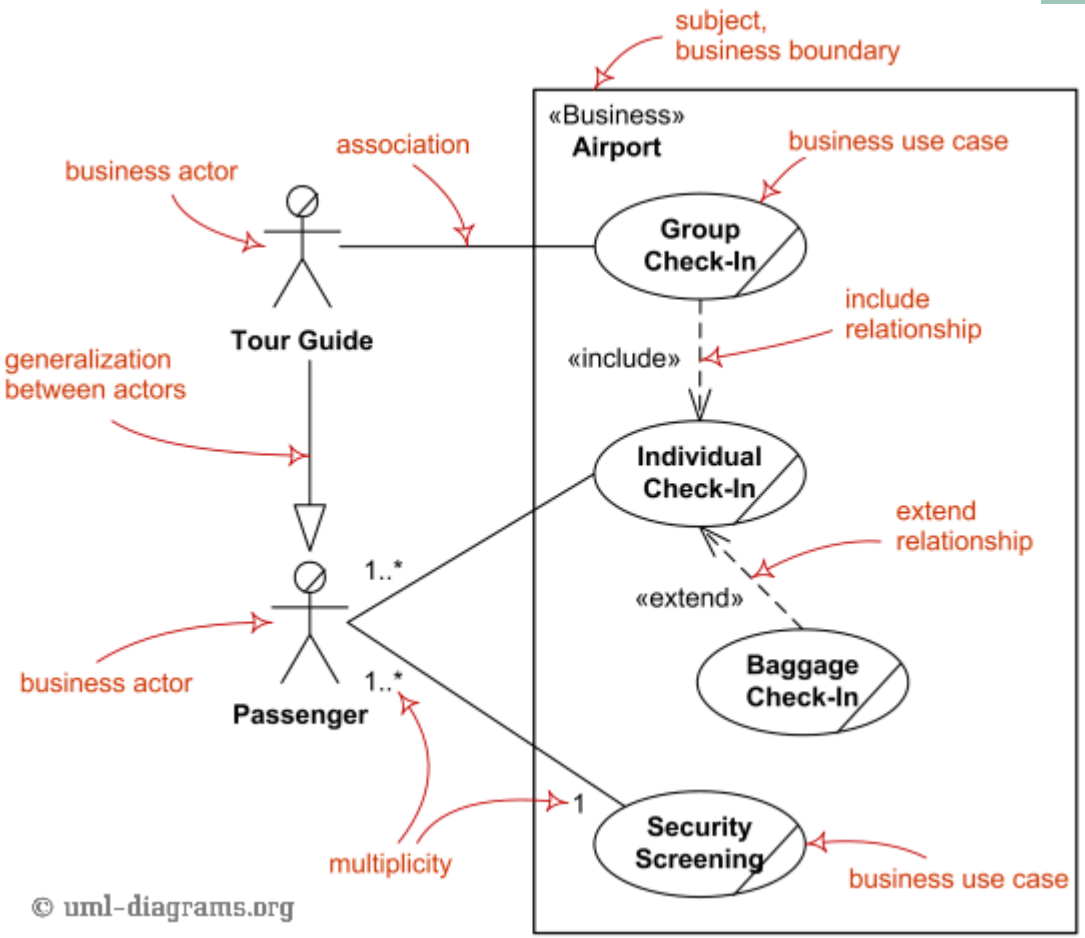
Extending Use Cases

- Why use the extend concept at all?

1. It can simplify maintenance and allow us to focus only on the extended functionality
2. Extension points for envisioned extensions can be provided in the base use case, which is an indication to future intent
3. The extended use case may represent optional behavior as opposed to a new, basic or alternative flow

Including Use Cases in Other Use Cases

- Certain patterns of user and system behavior reoccur (re-use) in a variety of places
- e.g., entering passwords, performing a system status check, selecting items from a table, etc.
- To avoid redundancy, the include relationship can be used.
- When used properly, the include relationship can simplify the development and maintenance activities.


subject,
business boundary
«Business»
Airport
business use case
association
business actor
Group
Check-In
include
relationship
«include»
Tour Guide
generalization
between actors
Individual
Check-In
extend
relationship
«extend»
1..*
business actor
Baggage
Check-In
1..*
Passenger
1
Security
Screening
multiplicity
business use case
© uml-diagrams.org

Step 4: Outline the Individual Use Cases

- To understand the system more think about the flow of events (basic and alternatives) for each use case.
- Basic flow: the most common path from start to finish through the system
- Alternative flows: other possible paths based on regular or exceptional circumstances.

Step 4: Outline the Individual Use Cases

- To do that ask the following questions:
- Basic flow:
  - What events start the use case?
  - How does the use case end?
  - How does the use case repeat some behavior?
- Alternative Flows:
  - Are there optional situations in the use case?
  - What odd cases might happen?
  - What variants might happen?
  - What may go wrong? ..etc

Formal use case example

Use Case 1 **Buy Stocks over the Web**

Primary Actor: Purchaser

Scope: Personal Advisors / Finance package (PAF)

Level: User goal

Stakeholders and Interests:

Purchaser—wants to buy stocks and get them added to the PAF portfolio automatically.

Stock agency—wants full purchase information.

Precondition: User already has PAF open.

Minimal Guarantee: Sufficient logging information will exist so that PAF can detect that something went wrong and ask the user to provide details.

Success Guarantee: Remote web site has acknowledged the purchase; the logs and the user's portfolio are updated.

Main Success Scenario:

1. Purchaser selects to buy stocks over the web.
2. PAF gets name of web site to use (E*Trade, Schwab, etc.) from user.
3. PAF opens web connection to the site, retaining control.
4. Purchaser browses and buys stock from the web site.
5. PAF intercepts responses from the web site and updates the purchaser's portfolio.
6. PAF shows the user the new portfolio standing.

Extensions:

2a. Purchaser wants a web site PAF does not support:

2a1. System gets new suggestion from purchaser, with option to cancel use case.

3a. Web failure of any sort during setup:

3a1. System reports failure to purchaser with advice, backs up to previous step.

3a2. Purchaser either backs out of this use case or tries again.

4a. Computer crashes or is switched off during purchase transaction:

4a1. (What do we do here?)

4b. Web site does not acknowledge purchase, but puts it on delay:

4b1. PAF logs the delay, sets a timer to ask the purchaser about the outcome.

5a. Web site does not return the needed information from the purchase:

5a1. PAF logs the lack of information, has the purchaser update questioned purchase.

Use case tables

- formal use cases can also be written as a table:

USE CASE NAME	Submit Promotion Order
ACTOR	Club Member
DESCRIPTION	Describes the process when a club member submits a club promotion order to either indicate the products they are interested in ordering or declining to order during this promotion
Normal Course	1. This use is initiated when the club member submits the promotion order to be proceeded
	2. The club member's personal information such as address is validated against what is currently recorded in member services
	3. The promotion order is verified to see if product is being ordered
	4. The club member's credit status is checked with Accounts Receivable to make sure no payments are outstanding
	5. For each product being ordered, validate the product number
	6. For each product being ordered, check the availability in inventory and record the ordered information which includes "quantity being ordered" and give each ordered product a status of "open"
	7. Create a Picking Ticket for the promotion order containing all ordered products which have a status "open"
	8. Route the picking ticket to the warehouse
PRECONDITION	Use case *send club promotion* has been processed
POST CONDITION	Promotion order has been recorded and the picking ticket has been routed to the warehouse
ASSUMPTIONS	

Step 5: Refine the Use Cases

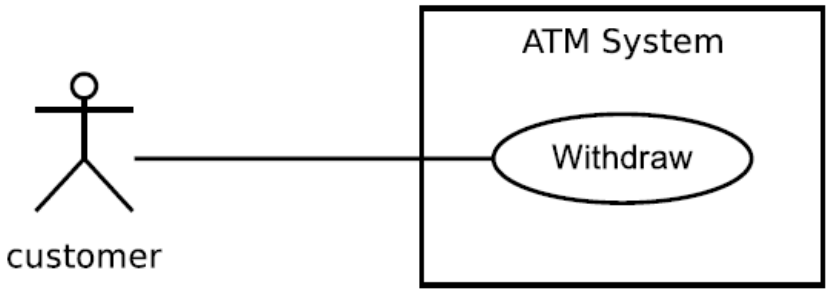
- When to refine and think about the next level of details?
- Depends on
  - All alternative flows including exception conditions
  - Pre- and post-conditions

Use Case template

Use case	**Accept Payment**
Brief description	This use case allows an Employee to accept the payment from `Customer` for a video rental.
Actors	Employee, Customer.
Preconditions	Customer expresses readiness to rent the video and he/she possesses valid membership card and the video is available for rental.
Main flow	The use case begins when the Customer decides to pay for the video rental and offers cash or debit/credit card payment. The Employee requests the system to display the rental charge...
Alternative flows	The Customer does not have sufficient cash and does not offer the card payment. The `Employee` asks the system to verify ...
Postconditions	If the use case was successful, the payment is recorded in the system's database. Otherwise, ...

Example: Automated Teller Machine

In this case study, look only at withdrawal:



Main flow:

insert card, enter PIN, request cash amount, get money, return card

Alternative flows: wrong PIN, cash amount exceeds balance, etc.

In this work: put a more precise informal description in the *post condition* of the use case. . .

How Use Cases Evolve

- The test for enough use cases should be the following:
- A complete collection of use cases should describe
  - all possible ways in which the system can be used,
  - at a level of specificity suitable to drive design, implementation, and testing.

The Scope of a Use Case

- Consider the use of a recycling machine.
- The customer
  - inserts cans and bottles into the recycling machine,
  - presses a button,
  - and receives a printed receipt that can be exchanged for money.
- Are there 3 uses cases?
  - one use case to insert a deposit item,
  - another use case to press the button,
  - and another to acquire the receipt?
- Or is it just one use case?

The Scope of a Use Case

- Three actions occur, but one without the others is of little value to the customer.
- The complete process is required to make sense to the customer.
- Thus, the complete dialogue
  - from inserting the first deposit item to pressing the button to getting the receipt

is a complete instance of use, of one use case.

The Benefits of Use Cases

- They are relatively easy to write and easier to read.
- They force developers to think through the design of a system from the perspective of a user.
- They engage the users in the requirements process:
  - helping them understand the system that is being proposed.
  - giving them a way to communicate and document their needs.
- They give context for the requirements of the system:
  - One can understand why a requirement is what it is
  - as well as how the system meets its objectives.

The Benefits of Use Cases

- They provide an ordering mechanism for requirements:
  - one can tell what has to happen before the next thing happens, and so on.
- In most circumstances, developers write the use cases. That means not only that there actually are understood requirements but also that the developers know they are responsible for determining them.
- It is a critical tool in the analysis process, helping us understand what the system needs to do and how it might go about doing it.

The Benefits of Use Cases

- It is a critical tool in the design and implementation process, reducing the risk of transitioning from an expression of requirements to a differing implementation
- They carry over directly into the testing process, helping to assure that the system actually does what it was intended to do
- They serve as inputs to the user documentation, conveniently organized in a step-by-step format.

Key Points

- Use cases carry the majority of the requirements for the system.
- The development team, with user involvement, writes the use cases.
- Use cases are built on a common, standard format.
- Use cases later drive test case development.