

Software Engineering


Anil Koyuncu

anil.koyuncu@sabanciuniv.edu

➞ Defining System and Context Boundaries

➡ Purpose of the System Context

Which questions answers thought about the system context?

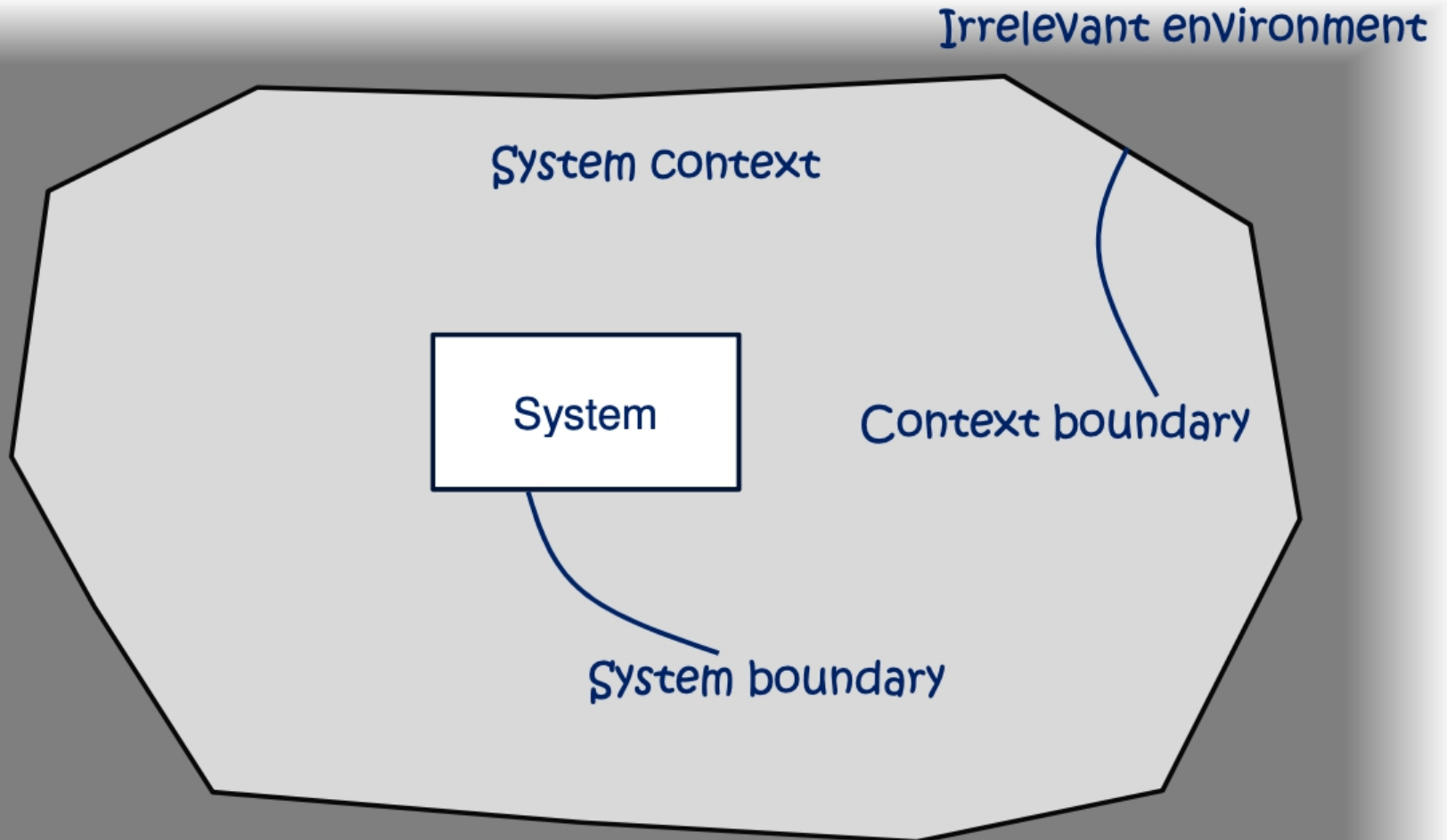
1. Which aspects pertain to the system,
which do not?
- 
- aspects that must
be specified
- aspects that don't have
to be specified

1. Which aspects stand in a direct relation to the
system?
- 
- aspects that must be considered
during specification

➡ System Context

The **system context** is the part of the system environment that is relevant for the definition as well as for the understanding of the requirements of a system to be developed.

➡ Considering Boundaries



➡ Context models

Context models are used to illustrate the operational context of a system - they show **what lies outside** the system boundaries.

Social and organisational concerns may affect the decision on where to position system boundaries.

Architectural models show the system and its **relationship** with other systems.

➞ System boundaries

System boundaries are established to define what is **inside** and what is **outside** the system.

They show other systems that are used or depend on the system being developed.

The position of the system boundary has a profound **effect on the system requirements**.

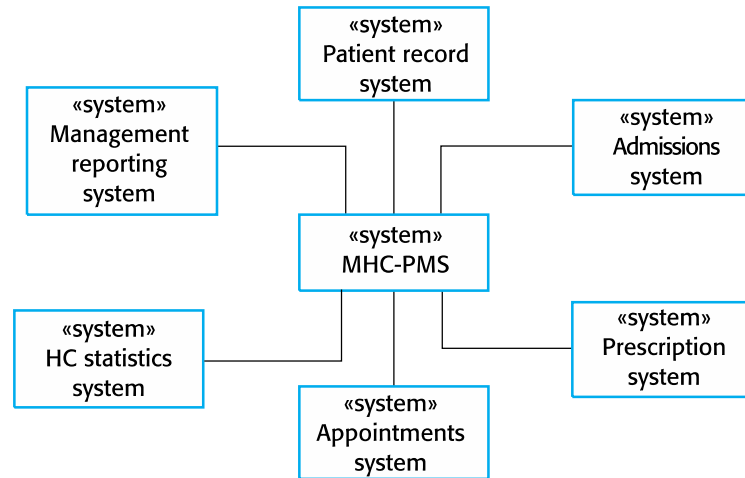
Defining a system boundary is a political judgment

There may be pressures to develop system boundaries that increase / decrease the influence or workload of different parts of an organization.

Context Boundary

The **context boundary** separates the relevant part of the environment of a system to be developed from the irrelevant part, i.e., the part that does not have to be considered during requirements engineering.

➡ The context of the Health Care Patient Management System



➡ What is a use case?

A use case is a written description of a user's interaction with the software system to accomplish a goal.

It is an example behavior of the system

Written from an actor's point of view, not the system's

3-9 clearly written steps lead to a “main success scenario”

➡ What is a use case?

A use case captures some visible function of the system.

A use case is a goal that an actor can accomplish using a system, through a series of user interactions.

- Transfer funds.
- Query balance.

This may be a large or small function.

- Depends on the chosen level of detail.
- Withdraw Funds vs Validate PIN

Use Cases

- Accompanied by a **use case description** detailing the user interactions required to accomplish the use case.
- Acquired through interviews with stakeholders and viewpoint analysis.
- Useful for eliciting and refining requirements.

Use case

Terminology

Actor: someone (or another system) interacting with the system

Primary actor: person who initiates the action

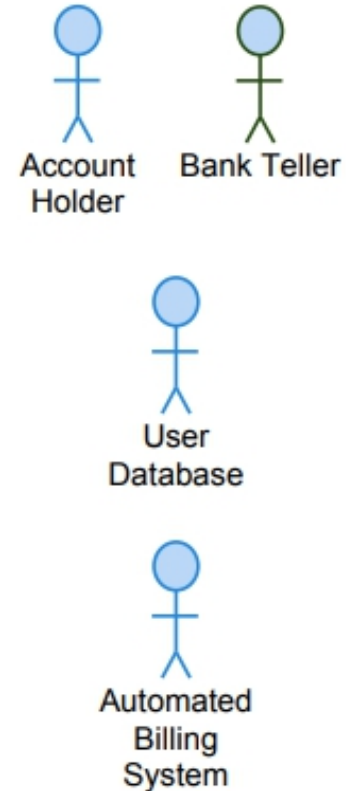
Goal: desired outcome of the primary actor

Use cases capture **functional requirements** of a system!

➡ What is an Actor?

An actor is a role a user plays with respect to the system.

- Actors carry out use cases. An actor can perform many use cases. A use case can involve multiple actors.
- A single user can be multiple actors, depending on how they use a system.
- Actors do not need to be human - can be an external system (hardware or software) that interacts with the system being built.
- Actor is a logical entity, so receiver and sender actors are different (even if the same person)

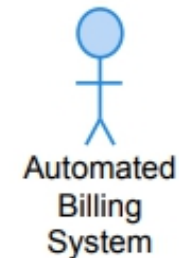
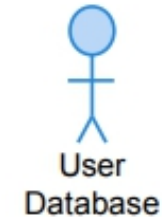
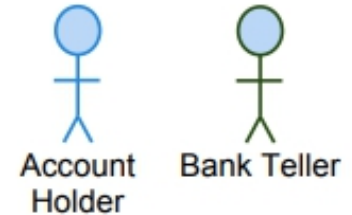


➡ Primary Actor

Primary actor: The main actor who initiates a UC

UC is to satisfy his goals

The actual execution may be done by a system or another person on behalf of the Primary actor



➡ Scenarios

Scenario: a set of actions performed to achieve a goal under some conditions

Actions specified as a sequence of steps

A step is a logically complete action performed either by the actor or the system

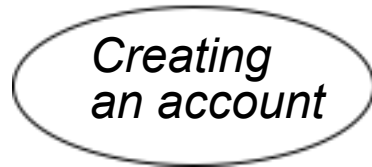
Main success scenario - when things go normally and the goal is achieved

Alternate scenarios: When things go wrong and goals cannot be achieved

➔ Use cases

A **use case** describes the **possible sequences of interactions** among the system and one or more **actors in response to some initial stimulus** by one of the actors

- Each way of using the system is called a use case
- A use case is not a single scenario but rather a description of *a set of scenarios*
- For example: *Creating an account*
- Individual use cases are shown as named ovals in use case diagrams



Use cases

A use case involves a **sequence of interactions** between the initiator and the system, possibly involving other actors.

In a use case, the system is considered as a **black-box**. We are only interested in externally visible behavior

Use cases

To define a use case, group all transactions that are similar in nature

A typical use case might include **a main case**, with alternatives taken in various combinations and including all possible exceptions that can arise in handling them

- Use case for a bank: *Performing a Transaction at the Counter*
 - Subcases could include *Making Deposits, Making Withdrawals, etc.*, together with exceptions such as *Overdrawn* or *Account Closed*
 - *Apply for a Loan* could be a separate use case since it is likely to involve very different interactions

Description of a use case should include events exchanged between objects and the operations performed by the system that are visible to actors



Example

Use Case 1: Buy stocks

Primary Actor: Purchaser

Goals of Stakeholders:

Purchaser: wants to buy stocks

Company: wants full transaction info

Precondition: User already has an account

Example ...

Main Success Scenario

1. User selects to buy stocks
2. System gets name of web site from user for trading
3. Establishes connection
4. User browses and buys stocks
5. System intercepts responses from the site and updates user portfolio
6. System shows user new portfolio standing

Example...

Alternatives

2a: System gives err msg, asks for new suggestion for site, gives option to cancel

3a: Web failure. 1-Sys reports failure to user, backs up to previous step.
2-User exits or tries again

4a: Computer crashes

4b: web site does not ack purchase

5a: web site does not return needed info

➡ 4 steps for creating a use case

1. Identify actors and goals

- Actors: What users and (sub)systems interact with our system?
- Goals: What does each actor need our system to do?

Identify actors/goals example

Consider software for a video store kiosk that takes the place of human clerks.

A customer with an account can use their membership and credit card at the kiosk to check out a video.

The software can look up movies and actors by keywords.

A customer can check out up to 3 movies, for 5 days each.

Late fees can be paid at the time of return or at next checkout.

Identify actors/goals

Exercise: actors/goals for your projects

➡ 4 steps for creating a use case

1. Identify actors and goals
2. Write the main success scenario
 - Main success scenario is the preferred "happy path"
 - Easiest to read and understand
 - Everything else is a complication on this
 - Capture each actor's intent and responsibility, from trigger to goal
 - State what information passes between actors
 - Number each step (line)

➡ 4 steps for creating a use case

1. Identify actors and goals
2. Write the main success scenario
3. List the failure extensions
 - Many steps can fail (e.g., denied credit card, out of stock)
 - Note each failure condition separately, after the main success scenario
 - Describe failure-handling
 - recoverable: back to main scenario (low stock + reduce quantity)
 - non-recoverable: fails (out of stock)
 - each scenario goes from trigger to completion
 - Label with step number (success scenario line) and letter
 - 5a <failure condition>: 5a.1 <fail with error message>

➡ 4 steps for creating a use case

1. Identify actors and goals
2. Write the main success scenario
3. List the failure extensions
4. List the variations
 - Steps can have alternative behaviors
 - Label alternatives with step number (success scenario line) and symbol
 - 5' <Alternative 1 for step 5>
 - 5'' <Alternative 2 for step 5>

➡ Qualities of a good use case

Focuses on interaction

- Starts with a request from an actor to the system

- Ends with the production of all the answers to the request

Focuses on essential behaviors, from actor's point of view

- Does not describe internal system activities

- Does not describe the GUI in detail

Concise, clear, and accessible to non-programmers

- Easy to read

- Summary fits on a page

- Main success scenario and extensions

➡ Do use cases capture these?

Which of these requirements should be represented directly in a use case?

1. Order cost = order item costs * 1.065 tax
2. Promotions may not run longer than 6 months
3. Customers only become Preferred after 1 year
4. A customer has one and only one sales contact
5. Response time is less than 2 seconds
6. Uptime requirement is 99.8%
7. Number of simultaneous users will be 200 max

An example use case

Use case: Check out an item for a borrower

Actors: Checkout clerk

Goals : To help the borrower to borrow the item if they are allowed, and to ensure a proper record is entered of the loan

Preconditions: The borrow must have a valid card and not own any fines. The item must have a valid barcode and not be from the reference section.

Steps:

Actor actions	System responses
1. Scan item's barcode and barcode of the browser's card	2. Display confirmation that the loan is allowed
3. Confirm that the loan is to be Initiated	4. Display confirmation that the loan has been recorded

Postconditions: The system has a record of the fact that the item is borrowed, and the date it is due

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:

User actions

1. Choose 'Open...' command
3. Specify filename
4. Confirm selection

System responses

2. File open dialog appears
5. Dialog disappears

Example (continued)

Use case Open file by typing name

Related use cases:

Specialization of: Open file

Steps

Actor actions

1. Choose 'Open...' command
- 3a. Select text field
- 3b. Type file name
4. Click 'Open'

System responses

2. File open dialog appears
5. Dialog disappears

➡ Example (continued)

Use case: Browse for file (inclusion)

Steps:

Actor actions

1. If the desired file is not displayed, select a directory
3. Repeat step 1 until the desired file is displayed
4. Select a file

System responses

2. Contents of directory is displayed

➡ Example (continued)

Use case: Open file by browsing

Related use cases:

Specialization of: Open file

Includes: Browse for file

Steps:

Actor actions

1. Choose 'Open...' command
3. Browse for file
4. Confirm selection

System responses

2. File open dialog appears
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 (extension point: step4: Choose Open)

Actor actions

1. Choose 'Open...' command
- 3a. Select text field
- 3b. Type file name
4. Choose 'Open'
6. Correct the file name
7. Choose 'Open'

System responses

2. File open dialog appears
5. System indicates that file does not exist
- 8 Dialog disappears

Example 2

Use Case 2: Buy a product

Primary actor: buyer/customer

Goal: purchase some product

Precondition: Customer is already logged in

Example 2...

Main Scenario

1. Customer browses and selects items
2. Customer goes to checkout
3. Customer fills shipping options
4. System presents full pricing info
5. Customer fills credit card info
6. System authorizes purchase
7. System confirms sale
8. System sends confirming email

Example 2...

Alternatives

6a: Credit card authorization fails

- Allows customer to reenter info

3a: Regular customer

- System displays last 4 digits of credit card no
- Asks customer to OK it or change it
- Moves to step 6

➡ Example - An auction site

Use Case1: Put an item up for auction

Primary Actor: Seller

Precondition: Seller has logged in

Main Success Scenario:

- Seller posts an item (its category, description, picture, etc.) for auction

- System shows past prices of similar items to seller

- System specifies the starting bid price and a date when auction will close

- System accepts the item and posts it

Exception Scenarios:

- 2 a) There are no past items of this category

- * System tells the seller this situation

➡ Example - auction site..

Use Case2: Make a bid

Primary Actor: Buyer

Precondition: The buyer has logged in

Main Success Scenario:

Buyer searches or browses and selects some item

System shows the rating of the seller, the starting bid, the current bids, and the highest bid; asks buyer to make a bid

Buyer specifies bid price, max bid price, and increment

System accepts the bid; Blocks funds in bidders account

System updates the bid price of other bidders where needed, and updates the records for the item



Exception Scenarios:

- 3 a) The bid price is lower than the current highest
 - * System informs the bidder and asks to rebid

- 4 a) The bidder does not have enough funds in his account
 - * System cancels the bid, asks the user to get more funds

➡ Example -auction site..

Use Case 3: Complete auction of an item

Primary Actor: Auction System

Precondition: The last date for bidding has been reached

Main Success Scenario:

Select highest bidder; send email to selected bidder and seller informing final bid price; send email to other bidders also

Debit bidder's account and credit seller's account

Transfer from seller's account commission amount to organization's account

Remove item from the site; update records

Exception Scenarios: None

➔ Example - Summary-level Use Case

Use Case: Auction an item

Primary Actor: Auction system

Scope: Auction conducting organization

Precondition: None

Main Success Scenario:

Seller performs put an item for auction

Various bidders make a bid

On final date perform Complete the auction of the item

Get feed back from seller; get feedback from buyer;
update records

Online Shopping

Use case: Place Order **Actors:** Costumer

Precondition: A valid user has logged into the system

Flow of Events:

1. The use case begins when the customer selects Place Order
2. The customer enters his or her name and address
3. If the customer enters only the zip code, the system supplies the city and state
4. The customer enters product codes for products to be ordered
5. For each product code entered
 - a) the system supplies a product description and price
 - b) the system adds the price of the item to the total end loop
6. The customer enters credit card payment information
7. The customer selects Submit
8. The system verifies the information [Exception: Information Incorrect], saves the order as pending, and forwards payment information to the accounting system.
9. When payment is confirmed [Exception: Payment not Confirmed], the order is marked confirmed, an order ID is returned to the customer, and the process terminates.

Online Shopping

Alternative Paths:

- At any time before step 7, the customer can select Cancel. The order is not saved and the use case ends
- In step 2, if the customer enters only the zip code, the system supplies the city and state
- In step 6, if any information is incorrect, the system prompts the customer to correct the information
- In step 7, if payment is not confirmed, the system prompts the customer to correct payment information or cancel. If the customer chooses to correct the information, go back to step 4 in the Basic Path. If the customer chooses to cancel, the use case ends.

Online Shopping

Exceptions:

Payment not Confirmed: the system will prompt the customer to correct payment information or cancel. If the customer chooses to correct the information, go back to step 6 in the Basic Path. If the customer chooses to cancel, the use case terminates.

Information Incorrect: If any information is incorrect, the system prompts the customer to correct it.

Postcondition:

If the order was not canceled, it is saved in the system and marked confirmed

➡ Styles of use cases

1. Use case diagram

often in UML, the Unified Modeling Language

2. Informal use case

3. Formal use case (≠ formal specification)

Let's examine each of these in detail...

➡ 1. Use case summary diagrams

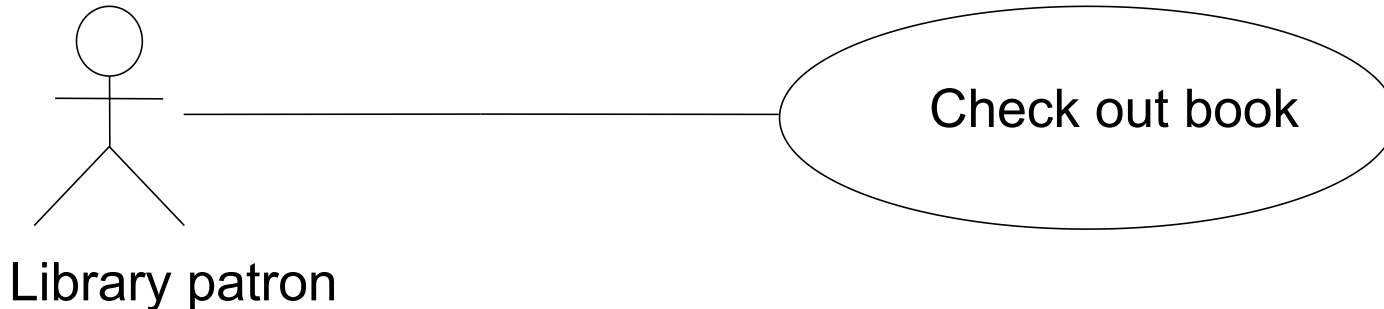
The overall list of your system's use cases can be drawn as high-level diagrams, with:

actors as stick-men, with their names (nouns)

use cases as ellipses, with their names (verbs)

line associations, 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

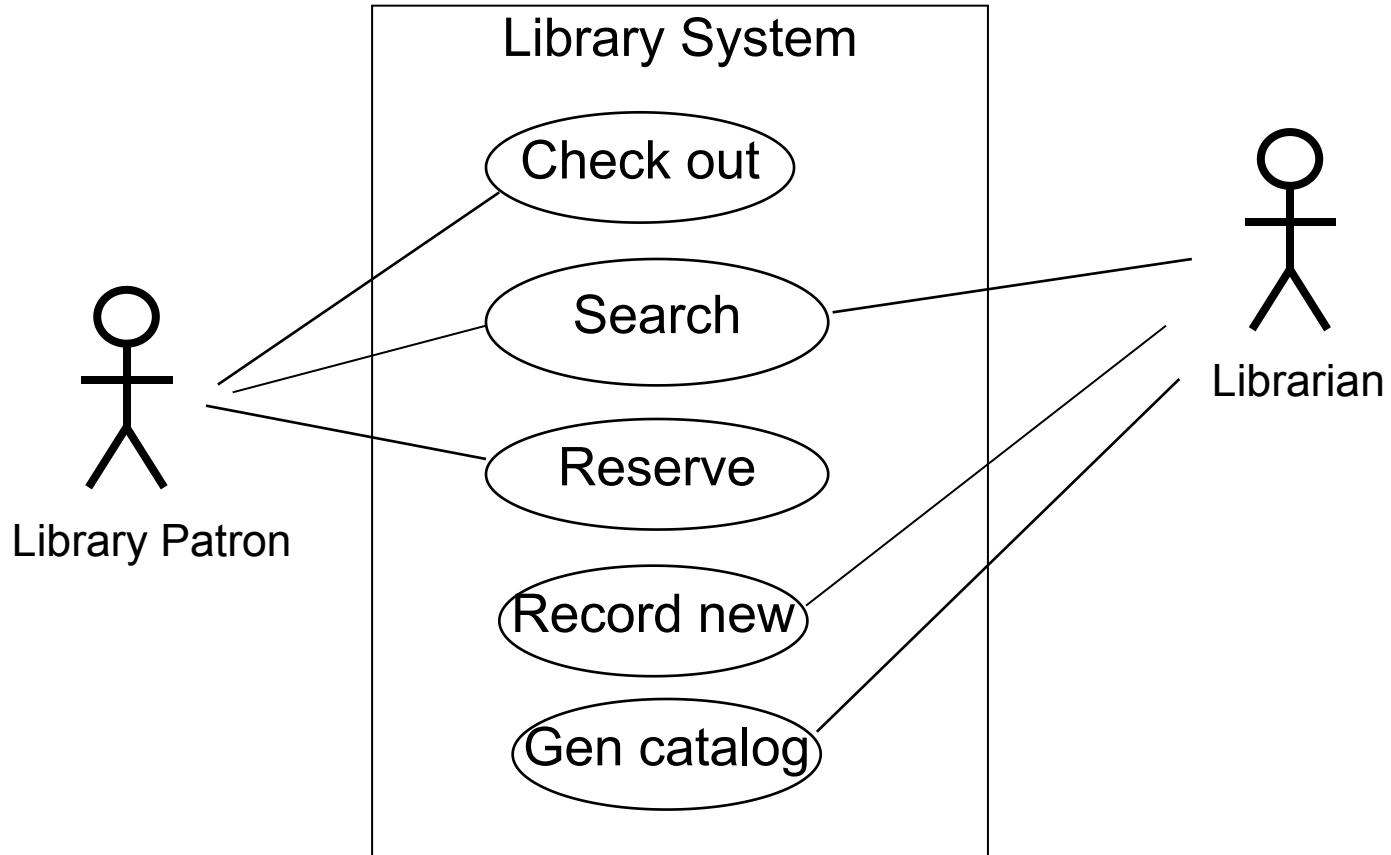


➡ Use case summary diagrams

It can be useful to create a list or table of primary actors and their "goals" (use cases they start). The diagram will then capture this material.

Actor	Goal
Library Patron	Search for a book
	Check out a book
	Return a book
Librarian	Search for a book
	Check availability
	Request a book from another library

➔ Use case summary diagram 1



➡ 2. Informal use case

Informal use case is written as a paragraph describing the scenario/interaction

Example:

Patron Loses a Book

The library patron reports to the librarian that she has lost a book.
The librarian prints out the library record and asks patron to speak with the head librarian, who will arrange for the patron to pay a fee.
The system will be updated to reflect lost book, and patron's record is updated as well.
The head librarian may authorize purchase of a replacement book.

3. Formal use case

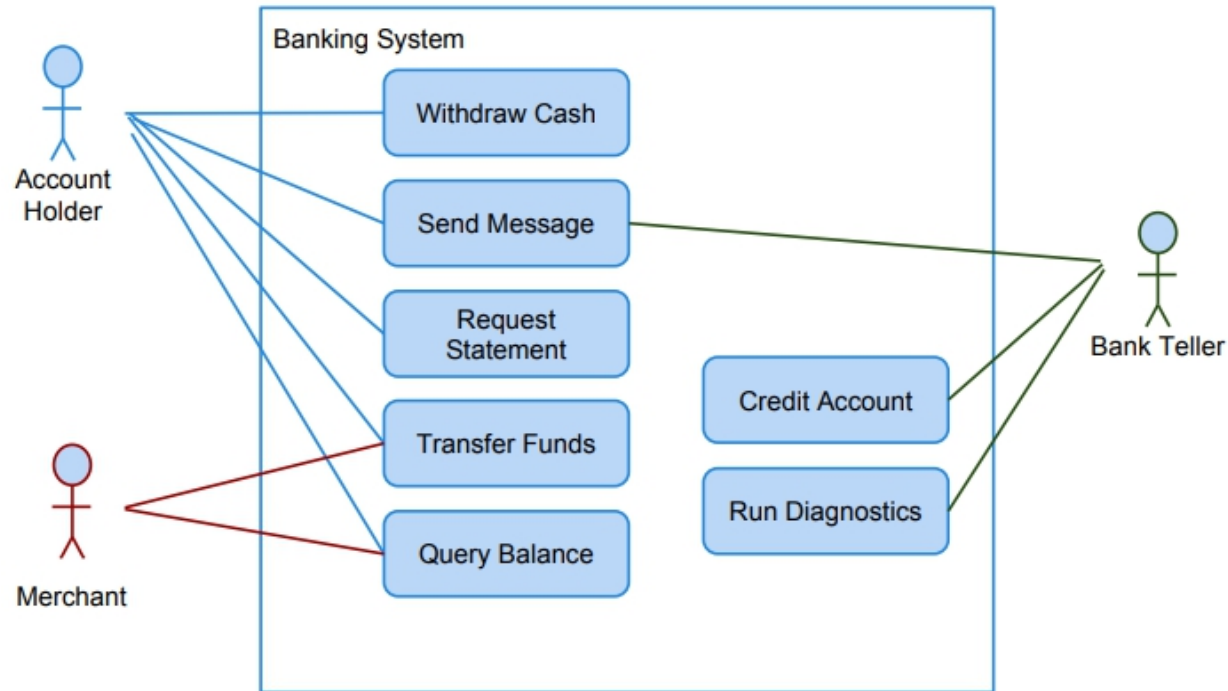
Goal	Patron wishes to reserve a book using the online catalog
Primary actor	Patron
Scope	Library system
Level	User
Precondition	Patron is at the login screen
Success end condition	Book is reserved
Failure end condition	Book is not reserved
Trigger	Patron logs into system

Main Success Scenario	<ol style="list-style-type: none"> 1. Patron enters account and password 2. System verifies and logs patron in 3. System presents catalog with search screen 4. Patron enters book title 5. System finds match and presents location choices to patron 6. Patron selects location and reserves book 7. System confirms reservation and re-presents catalog
Extensions (error scenarios)	<ol style="list-style-type: none"> 2a. Password is incorrect <ol style="list-style-type: none"> 2a.1 System returns patron to login screen 2a.2 Patron backs out or tries again 5a. System cannot find book <ol style="list-style-type: none"> 5a.1 ...
Variations (alternative scenarios)	<ol style="list-style-type: none"> 4. Patron enters author or subject

➡ 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. **Pre-conditions**: State of the system before the use case
- E. **Summary**: Give a short informal description
- F. **Related use cases**
- G. **Steps**: Describe each step using a multi-column format
- H. **Post-conditions**: State of the system in following completion

➡ Online Banking Use Case Diagram



➡ Developing a Use-Case

What are **the main tasks or functions** that are performed by the actor?

What system information will the the actor acquire, produce or change?

Will the actor have to inform the system about changes in the external environment?

What information does the actor desire from the system?

Does the actor wish to be informed about unexpected changes?

Use Case Modeling Benefits

MANY!!!

A view of system behavior from an external person's viewpoint.

An effective tool for validating requirements.

An effective communication tool.

As a basis for a test plan.

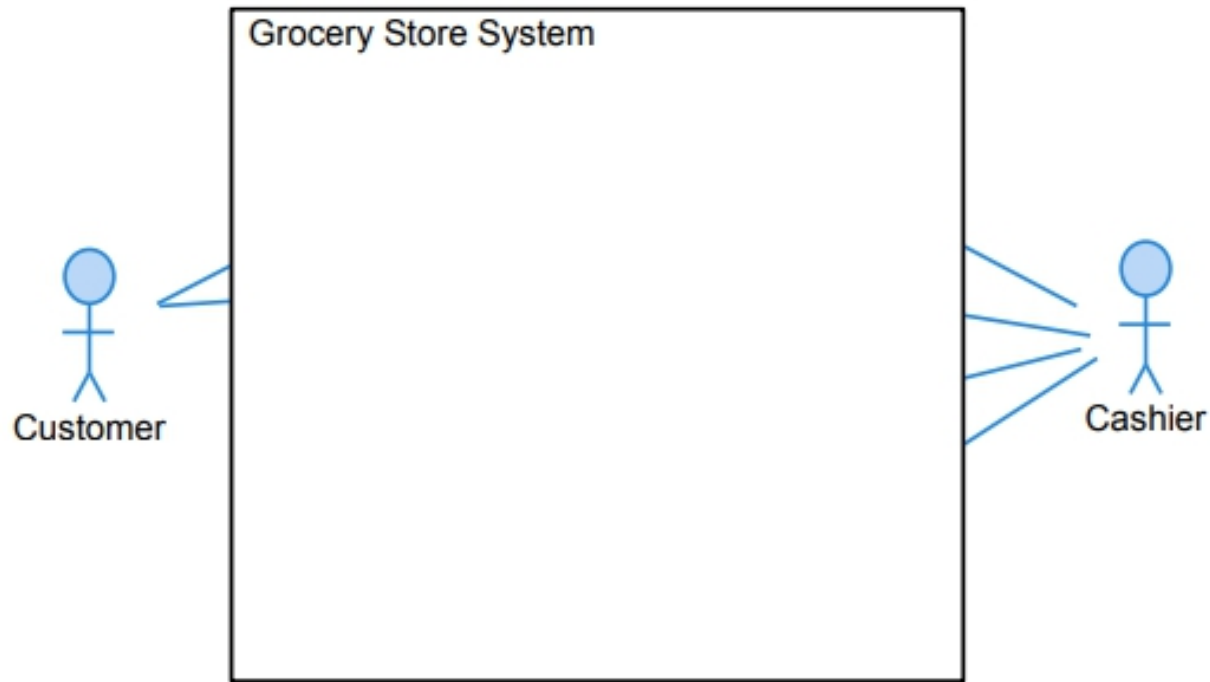
As a basis for a user's manual.

➡ Defining use cases

1. Identify the boundary of the application, identify the objects outside the boundary that interact with the system
2. Classify the objects by the roles they play, each role defines an actor
3. Each fundamentally different way an actor uses the system is a use case
4. Make up some specific scenarios for each use case (plug in parameters if necessary)
5. Determine the interaction sequences: identify the event that initiates the use case, determine if there are preconditions that must be true before the use case can begin, determine the conclusion
6. Write a prose description of the use case
7. Consider all the exceptions that can occur and how they affect the use case

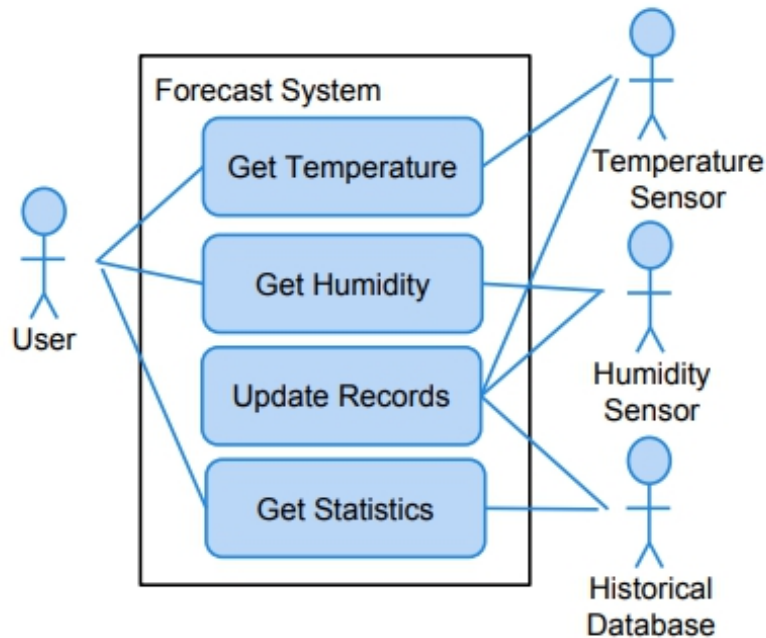
➡ Setting the System Boundary

The system boundary will affect your actors and use-cases.



➡ System Boundary - Weather Forecast

The system boundary will affect your actors and use-cases.

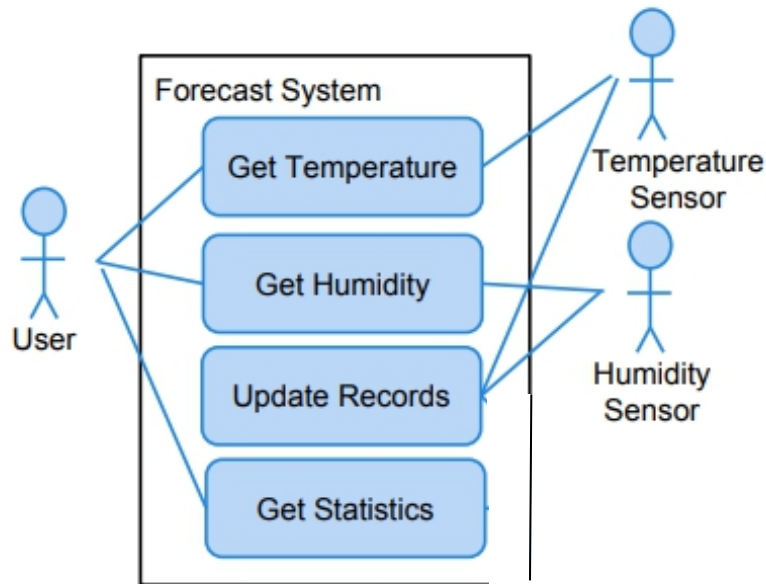


Option 1: Software Boundary

- System is just the software. Users, Sensors, and Database are all actors.
- Four use-cases: Get Temperature, Get Humidity, Get Statistics, Update Update Records Records.

➡ System Boundary - Weather Forecast

The system boundary will affect your actors and use-cases.



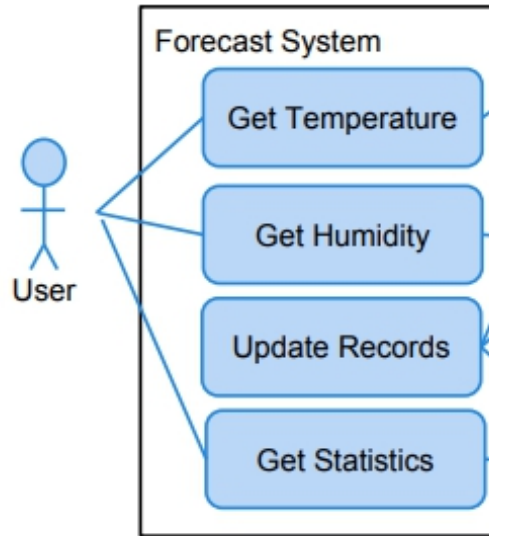
Option 2: Computer Boundary

- System is the computer unit. Database is no longer an external actor.
- Still four use-cases: Get Temperature, Get Humidity, Get Statistics, Update Records.

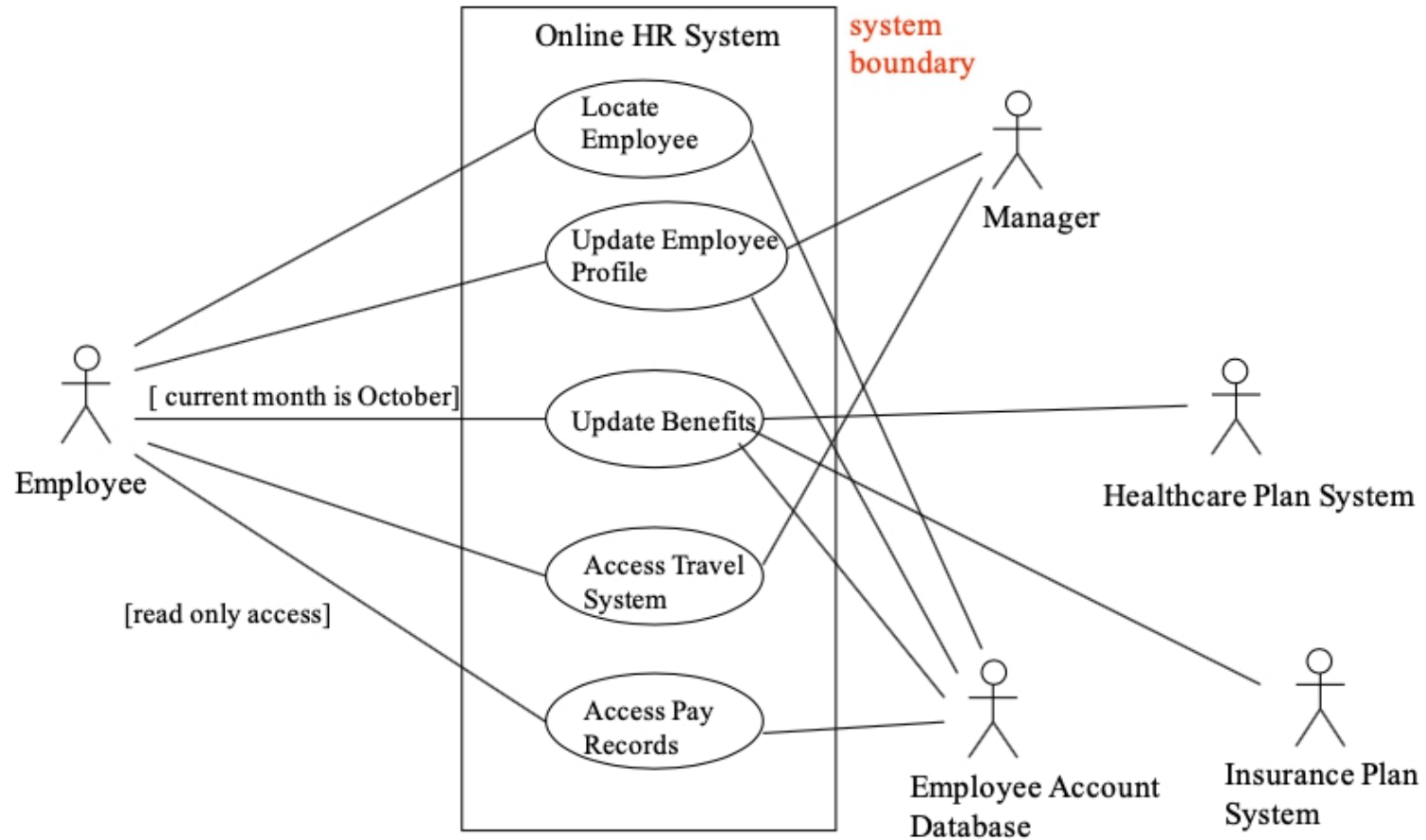
➡ System Boundary - Weather Forecast

The system boundary will affect your actors and use-cases.

Option 3: Computer+Sensors Boundary



➡ Online Human Resources System



Use case: Update Benefits

Actors: Employee, Employee Account Database, Healthcare Plan System, Insurance

Plan System

Precondition: Employee has logged on to the system and selected “update benefits” option

Flow of Events:

Basic Path:

1. System retrieves employee account from Employee Account Database
2. System asks employee to select medical plan type; **uses** Update Medical Plan
3. System asks employee to select dental plan type; **uses** Update Dental Plan
- ...

Alternative Paths:

➡ Use Case Modeling Process - Steps.

Step 1: Identifying Any Additional Actors and Use Cases

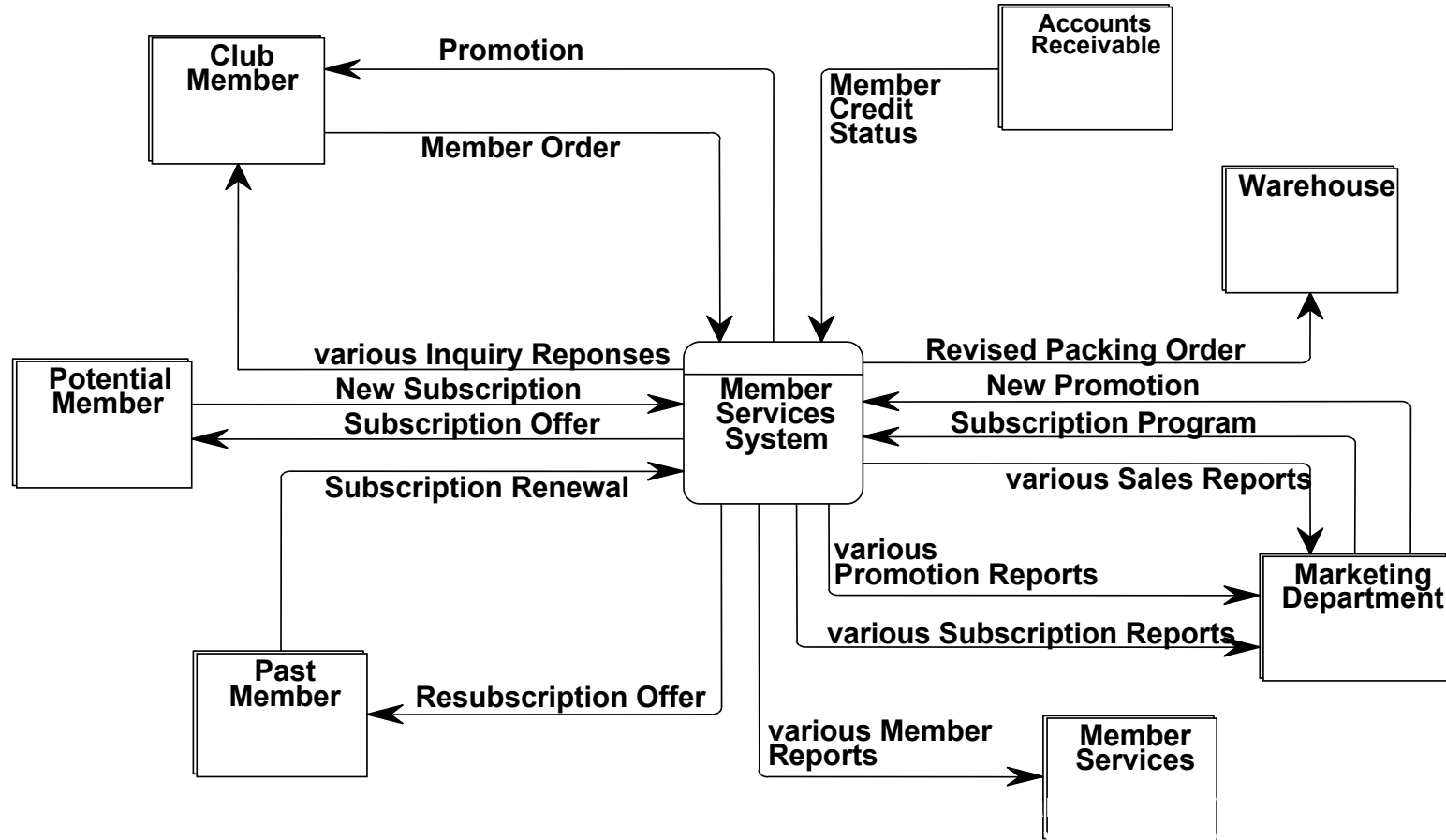
One approach: interviewing - to identify actors and use cases.

An excellent approach: Analyze the **context model diagram** of the system

Context model identifies boundary of system and shows external entities that interact with system to provide inputs and receive outputs.

- If external party initiates, it is considered an actor.

➡ Member Services System Context Model Diagram



➔ Step 1: Identifying Actors and Use Cases

Actor	Use Case Name	Use Case Description
Potential Member	SUBMIT NEW SUBSCRIPTION	Potential member joins the club by subscribing. ("Take any 12 CDs for one penny and agree to buy 4 more at regular club prices within two years.")
Club Member	PLACE NEW MEMBER ORDER	Club member places order.
Club Member	MAKE ACCOUNT INQUIRY	Club member wants to examine his or her account history. (90-day time limit)
Club Member	MAKE PURCHASE INQUIRY	Club member inquires about his/her purchase history. (three-year time limit)
Club Member	MAINTAIN MEMBER ORDER	Club member wants to revise an order or cancel an order.
Club Member	SUBMIT CHANGE OF ADDRESS	Club member changes address. (including e-mail and privacy code)
Past Member	SUBMIT RESUBSCRIPTION	Past member rejoins the club by resubscribing.
Marketing	SUBMIT NEW MEMBER SUBSCRIPTION PROGRAM	Marketing establishes a new membership resubscription plan to entice new members.
Marketing	SUBMIT PAST MEMBER RESUBSCRIPTION PROGRAM	Marketing establishes a new membership resubscription plan to lure back former members.
Marketing	SUBMIT NEW PROMOTION	Marketing initiates a promotion. (Note: A promotion features specific titles, usually new, that company is trying to sell at a special price. These promotions are integrated into a catalog sent (or communicated) to all members.)
Time	GENERATE QUARTERLY PROMOTION ANALYSIS	Print quarterly promotion analysis report.
Time	GENERATE QUARTERLY SALES ANALYSIS	Print annual sales analysis report.
Time	GENERATE QUARTERLY MEMBERSHIP ANALYSIS	Print annual membership analysis report.
Time	GENERATE ANNUAL SALES ANALYSIS	Print annual sales analysis report.
Time	GENERATE ANNUAL MEMBERSHIP ANALYSIS	Print annual membership analysis report.

Result of identifying use cases and actors from previous slide....

➡ Use Case Modeling Process - Steps - continued.

Step 2: Constructing a Use Case Model

Here, we will create a Use Case Model Diagram

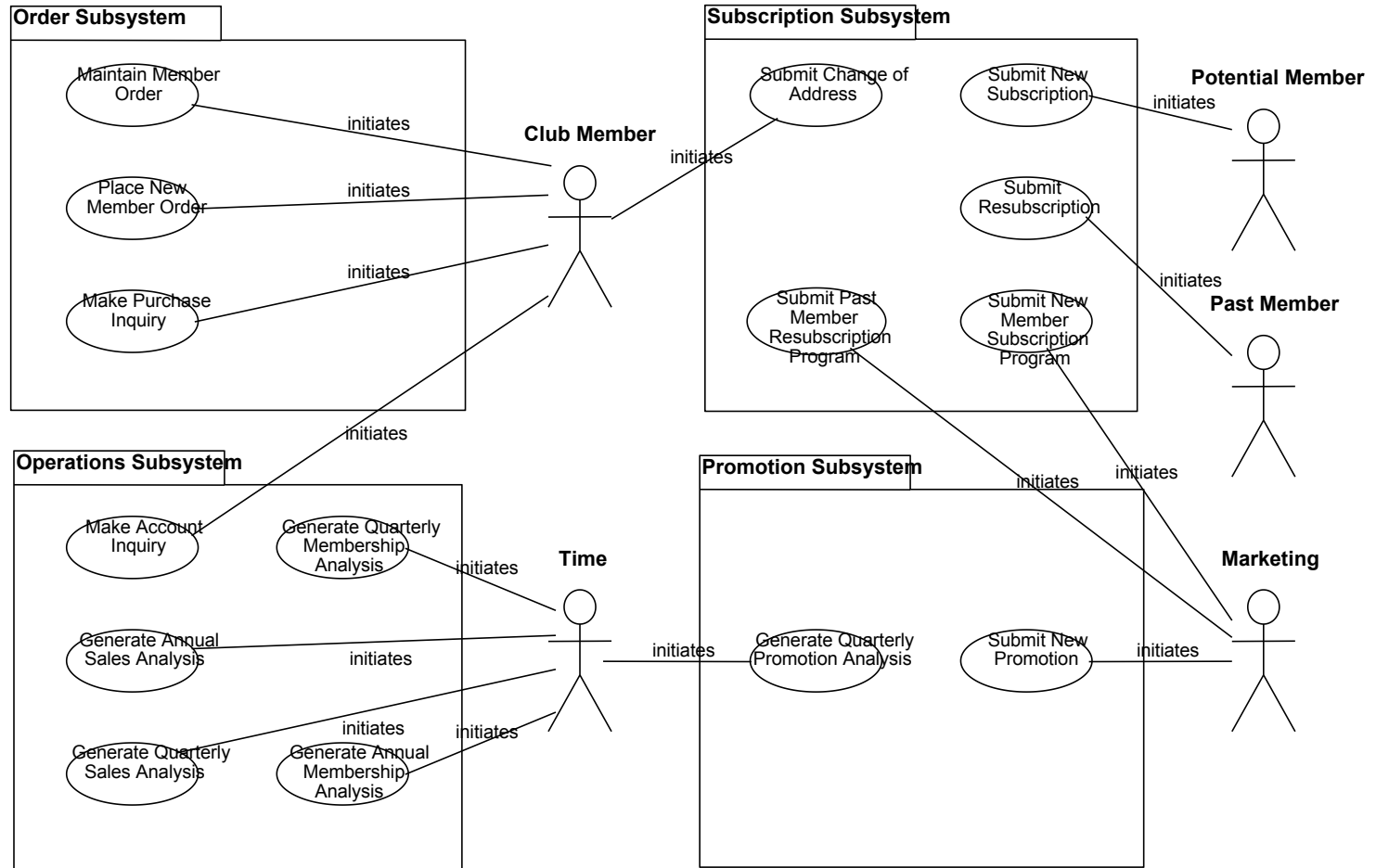
- shows system scope and boundaries.

Notice the Use Cases are partitioned into 'subsystems.'

- This subsystem is a UML package description



Step 2: Constructing a Use Case Model Diagram



Use Case Modeling

A use case model diagram can be used to graphically depict the system scope and boundaries in terms of use cases and actors.

The use case model diagram for the Member Services System is shown in the previous slide.

It represents the relationships between the actors and use cases defined for each business subsystem.

The subsystems (UML package symbol) represent logical functional areas of business processes.

The partitioning of system behavior into subsystems is very important in understanding the system architecture and is very key to defining your development strategy – which use cases will be developed first and by whom.

➡ Use Case Modeling Process - Steps - continued.

Step 3: Document the Use Case Course of Events

Create the Use Case narrative.

Ensure you do both typical course and alternative courses.

Supply only general information about the business events.

Looking to validate requirements

Design Use cases (later) will address implementation.

➡ Step 3: Documenting the Use Case Typical Course

Author: S. Shepard

Date:

Use Case Name:	Submit New Member Order	
Actor(s):	Member	
Description:	This use case describes the process of a member submitting an order for Sound Stage products. On completion, the member will be sent a notification that	
	the order ¹ as accepted.	
References: Typical Course of Event ² :	<p>Actor Action</p> <p>Step 1: This use case is initiated when a member submits an order to be processed</p> <p>Step 7: This use case concludes when the member receives the order confirmation notice.</p>	<p>System response</p> <p>Step 2: The member's personal information such as address is validated against what is currently recorded in member services.</p> <p>Step 3: The member's credit status is checked with Accounts Receivable to make sure no payments are outstanding.</p> <p>Step 4: For each product being ordered, validate the product number and then check the availability in inventory and record the ordered product information.</p> <p>Step 5: Create a picking ticket for the member order containing all ordered products that are available and route it to the warehouse for processing.</p> <p>Step 6: Generate an order confirmation notice indicating the status of the order and send it to the member.</p>



Step 3: Documenting the Use Case Typical Course (concluded)

Alternate Courses: an 3 them to	Step 2: If the club member has indicated an address or telephone number change on the promotion order, update the club member's record with the new information. Step 3: If Accounts Receivable returns a credit status that the customer is in arrears, send order rejection notice to the member. Step 4: If the product number is not valid, send a notification to the member requesting
the 4	submit a valid product number. If the product being ordered is not available, record
Pre-condition: 5	ordered product information and mark as "back-ordered." Orders can only be submitted by members.
Post-condition: 6	Member order has been recorded and the picking ticket has been routed to the warehouse.
Assumptions:	None at this time.

Contents: More General Features....

1. A reference to the requirement(s) in which use case may be traced.
2. A typical event course describing the use case's major steps, from beginning to end of this interaction with the actor.
3. Alternate courses describing exceptions to the typical course of events.
4. Precondition describing the state the system is in before the use case is executed.
5. Postcondition describing the state the system is in after the use case is executed.
6. An assumptions section, which includes any non-behavioral issues, such as performance or security, that is associated with the use case, but is difficult to model within the use case's course of events.

➞ Extension and Abstract Use Cases

In creating analysis use cases, you may discover a good deal of complexity in a use case with several steps.

As stated, you may extract the more complex steps into their own use case.

This type of use case is called an **extension use case**.

- (extends the functionality of the original use case).

An **extension use case** extends the functionality (typical course) of an original use case. An extension use case can only be invoked by the use case it is extending.

➔ Extended Use Cases

Extension and abstract use cases are usually discovered during analysis and their main purpose is to simplify the original use cases - basically decompose the original use case into a less complex form.

You are creating additional use cases, however.

An 'extension' use case literally 'extends' a base use case.

➞ Extension and Abstract Use Cases

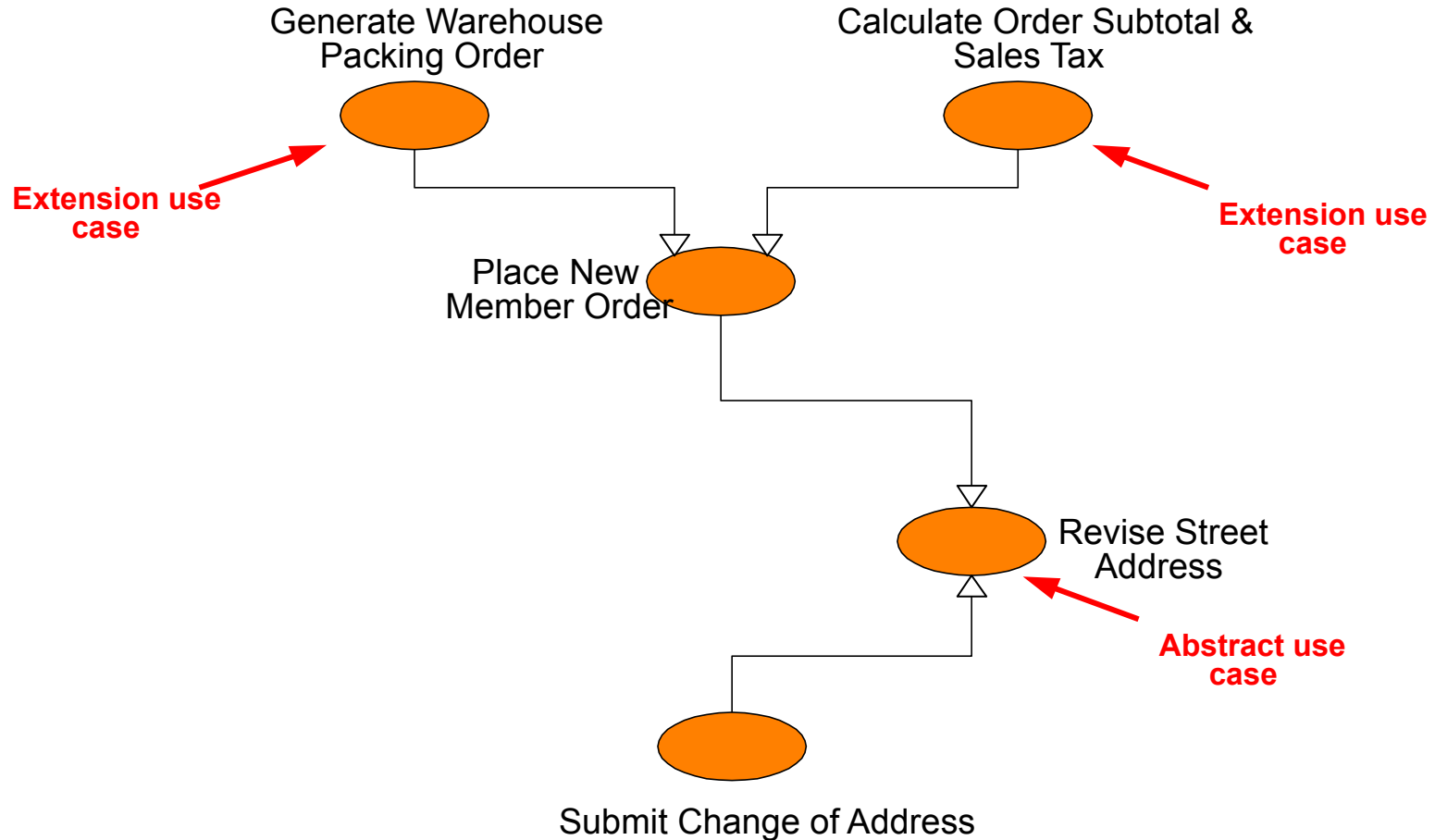
Sometimes we note that use cases contain the same functionality. Extract this common functionality into its own use case, called an 'abstract use case.'

In UML, we call this an 'included' use case.

Nice feature about an abstract use case is that it supports reusability!!

Consider the next overhead...

➡ Depicting Extension and Abstract Use Cases using UML Notation



➡ Combining use cases

A use case **extends** another use case when it embeds new behavior into a complete base case

- *Check Baggage* extends the base case *Check in for Flight*
- You do not have to check baggage to check in for flight.

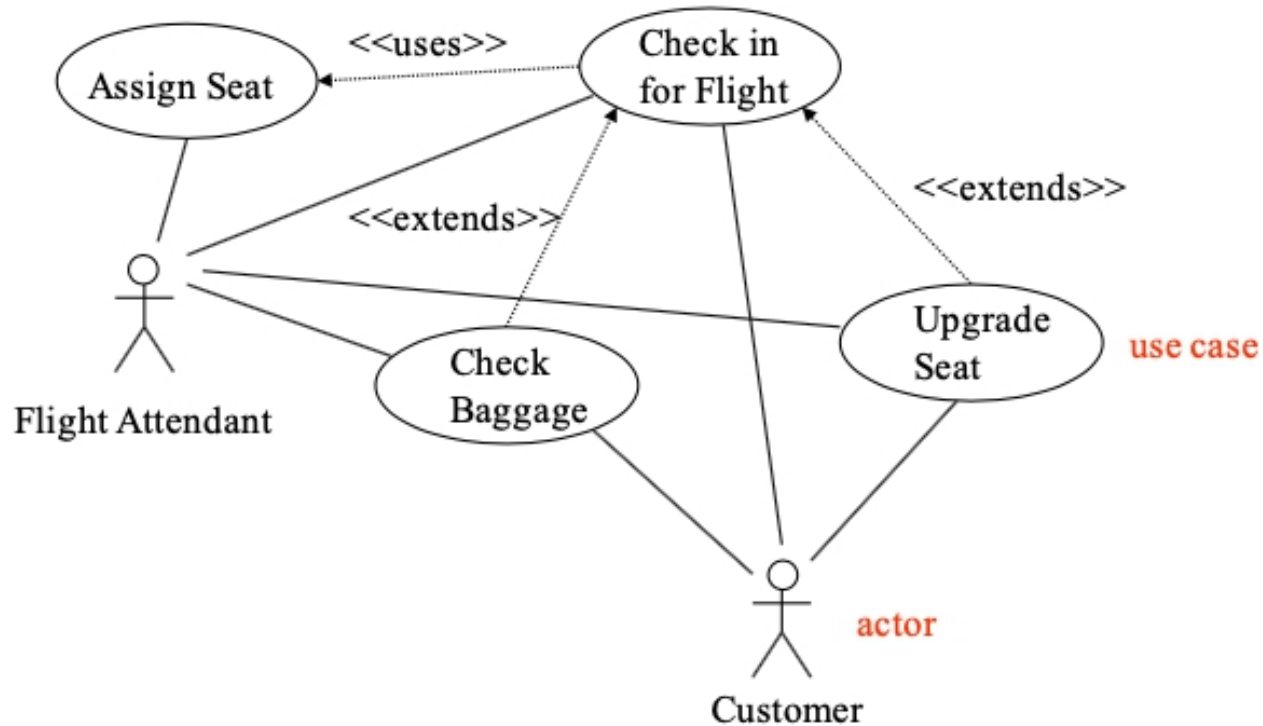
A use case **uses** another use case when it embeds a subsequence as a necessary part of a larger case (In some texts this relationship is called **includes** instead of **uses**)

- **uses** relationship permits the same behavior to be embedded in many otherwise unrelated use cases
- For example *Check in for Flight* use case **uses** *Assign Seat* use case

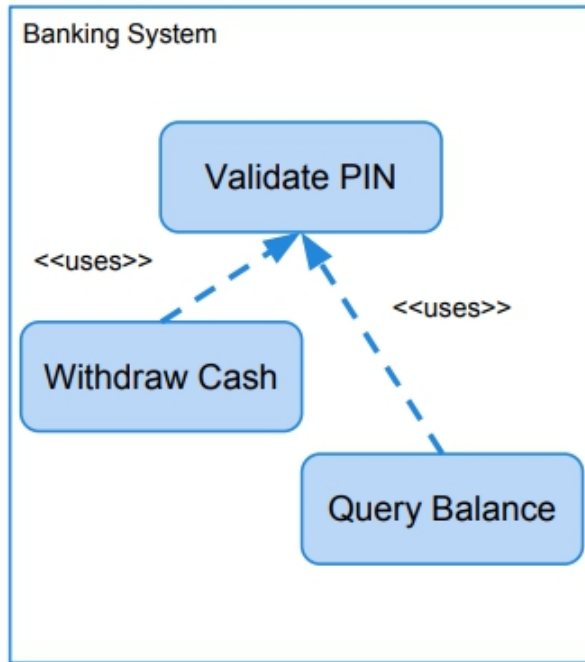
The difference is

- In the **extends** the extended use case is a valid use case by itself
- In the **uses** the use case which is using the other use case is not complete without it

➡ UML use case diagrams



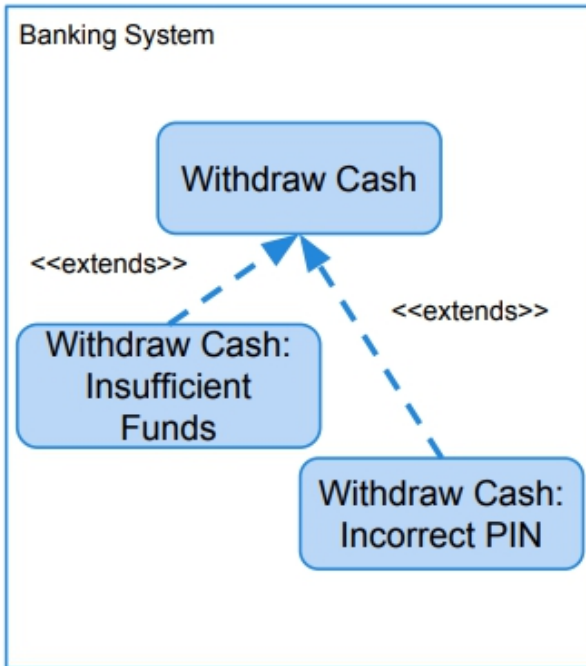
➡ Use Case Relationships: Uses



Uses

- You have a piece of behavior that is similar across many use-cases.
- Break this out as a separate use-case and let the others “use” it.
- Avoids repetition in written use-cases.
 - Step 1: Complete “Validate PIN” use-case.
 - Step 2: Select account.
 - ...

➡ Use Case Relationships: Uses



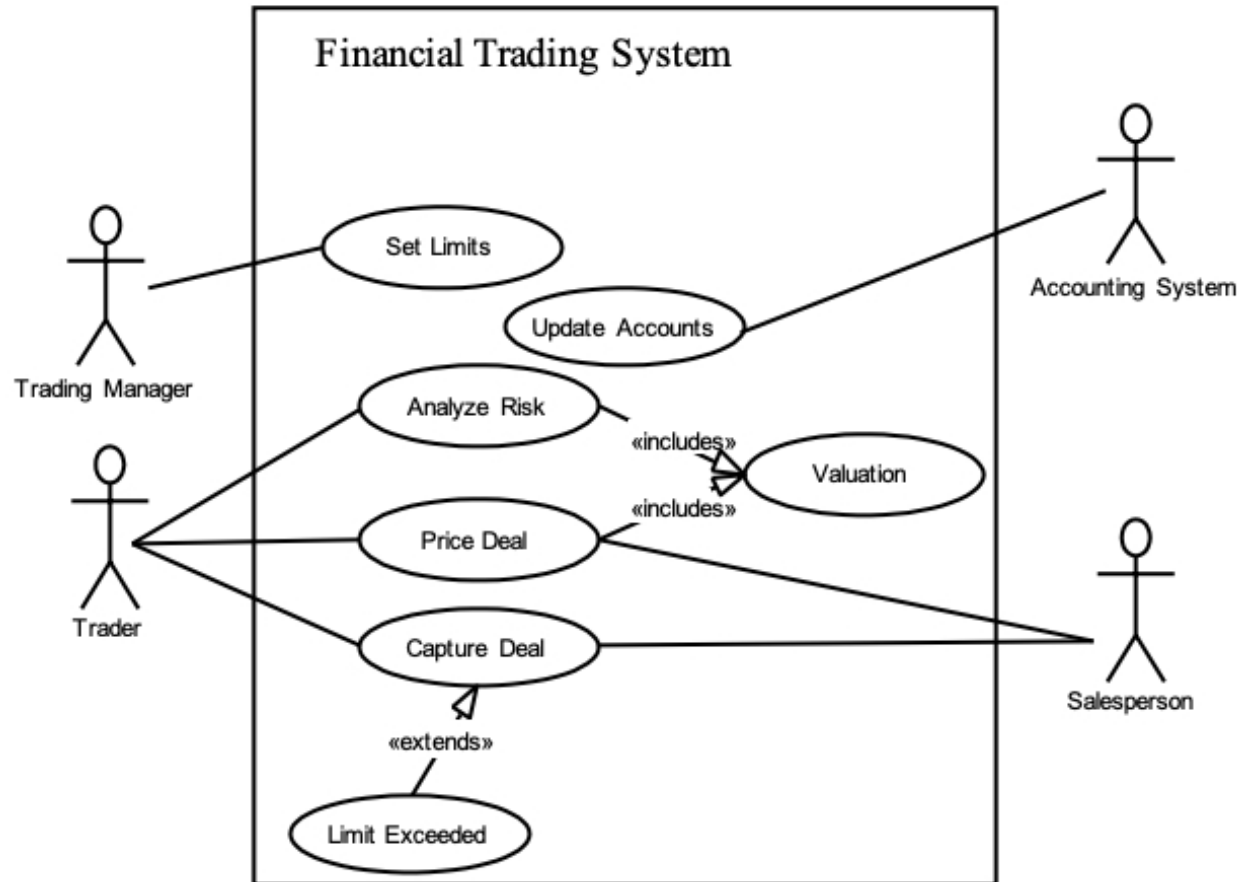
Extends

● A use-case is similar to another, but does more or takes an alternate path. ● Put the normal behavior in one use-case and the exceptional behavior somewhere else:

- Capture the normal behavior.
- Try to figure out what went wrong in each step.
- Capture the exceptional cases in separate use-cases

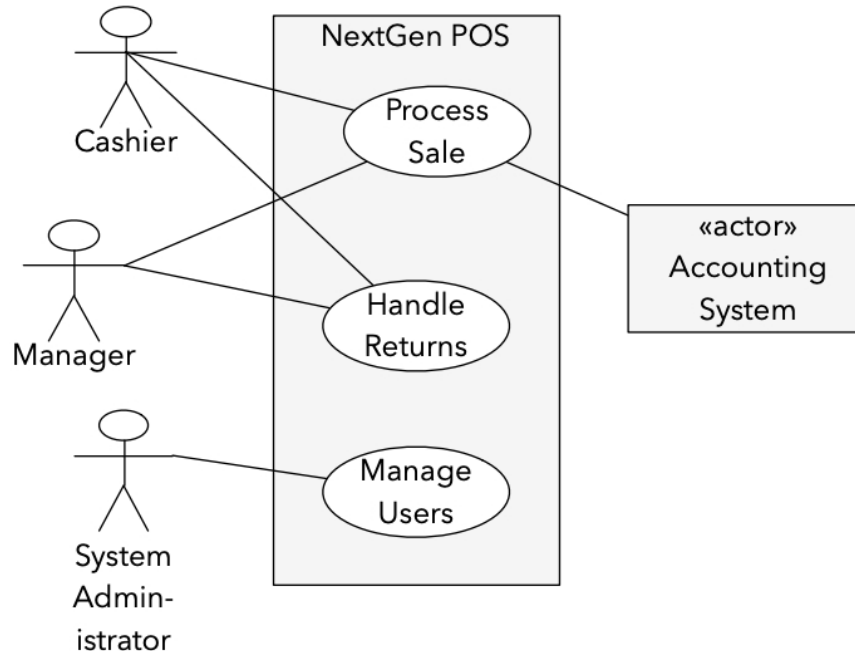
● Allows for easier to understand descriptions.

➡ Use Case Examples



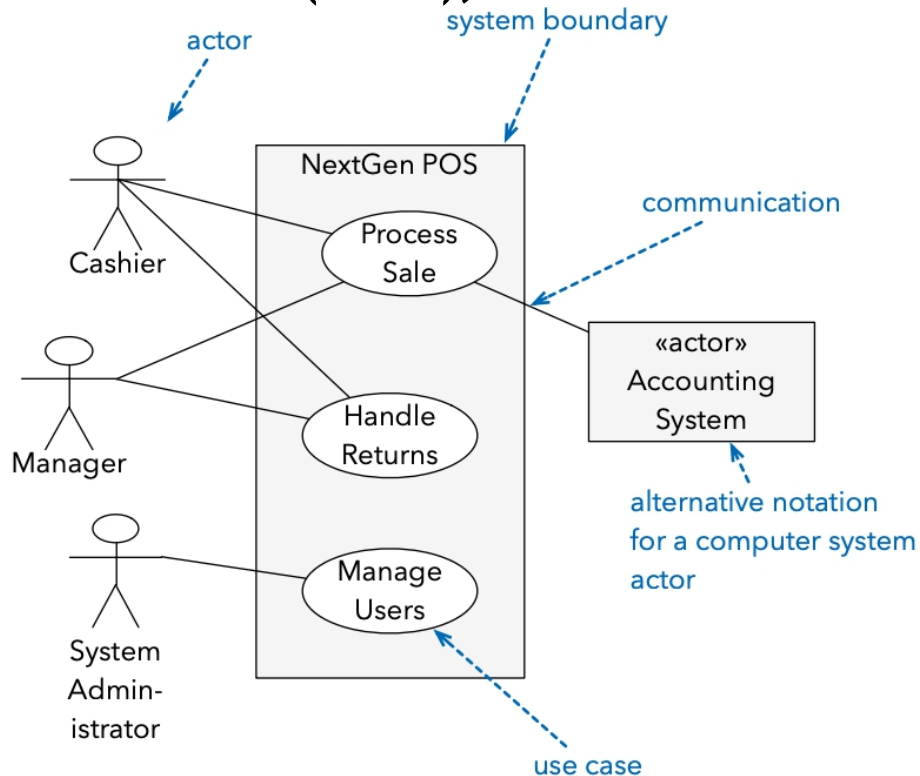
➞ UML Use Case Diagrams

UML use case diagrams provide a notation to illustrate the names of use cases and actors (roles), and the relationship between them



➡ UML Use Case Diagrams

UML use case diagrams provide a notation to illustrate the names of use cases and actors (roles), and the relationship between them



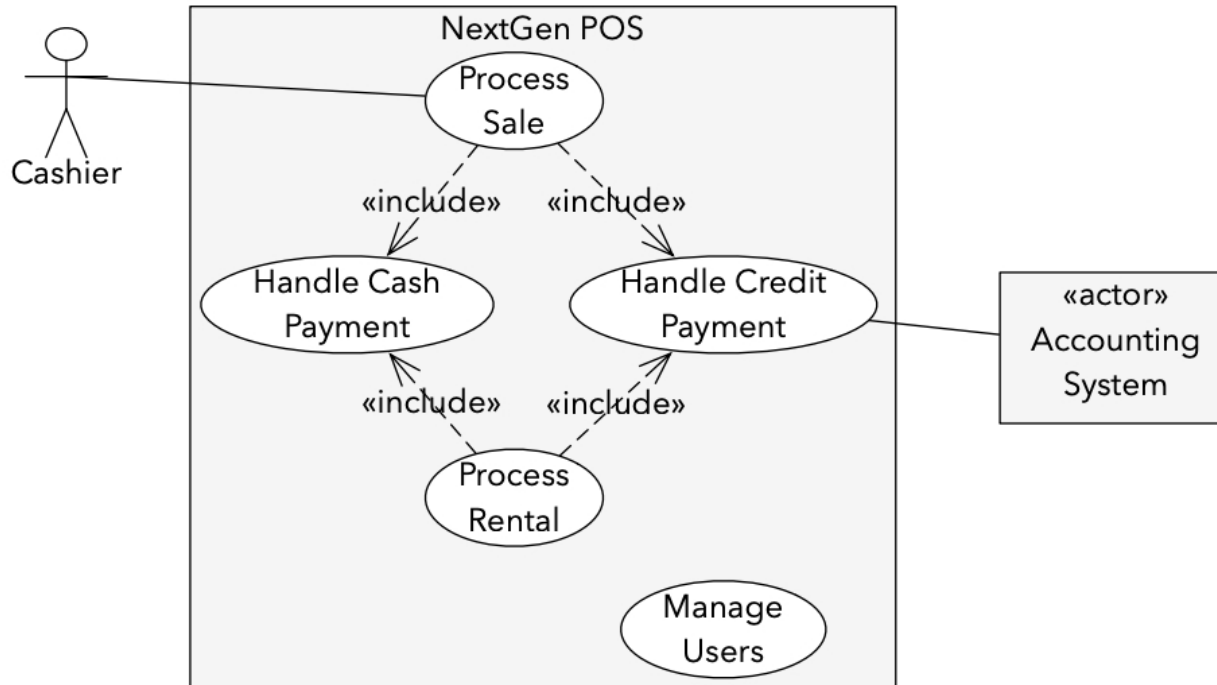
➞ UML use case diagrams

UML use case diagram provide a black-box view on a system

They are particularly useful during the early phases of a software project.

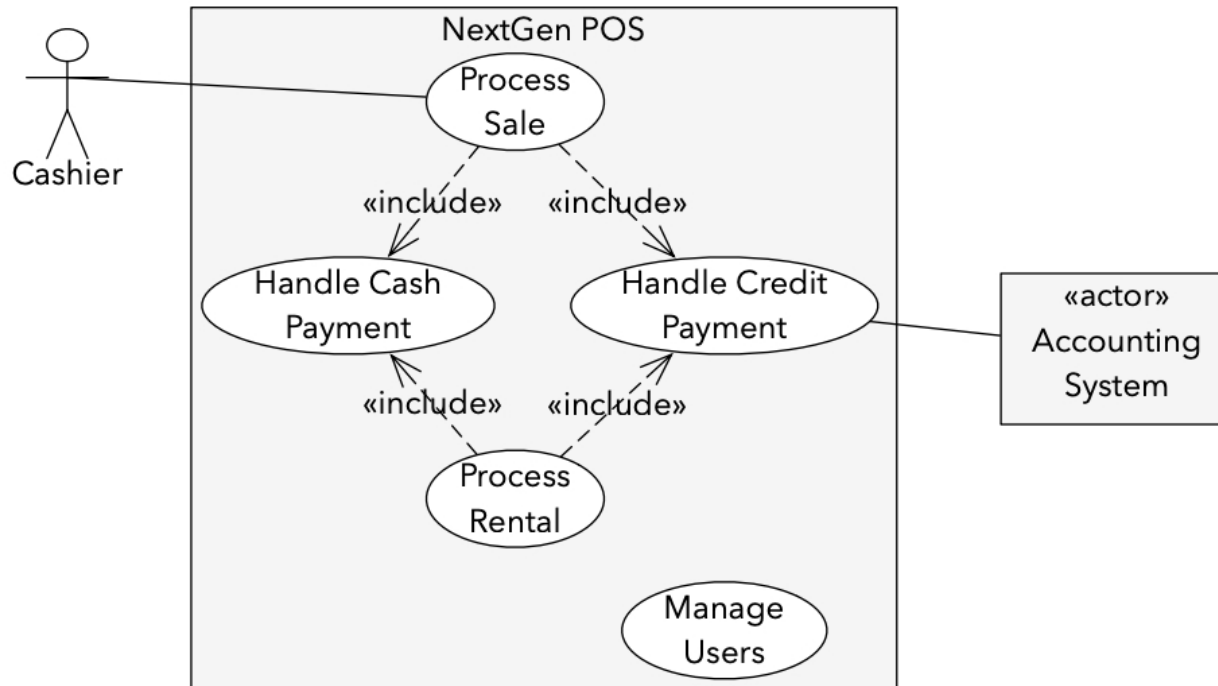
➡ The Include Relationship in UML Use Case Diagrams

For partial behavior that is common across several use cases (e.g. “Pay by Credit” occurs in “Process Sale”, “Process Rental”,...) it is desirable to separate it into its own sub-function use case and indicate



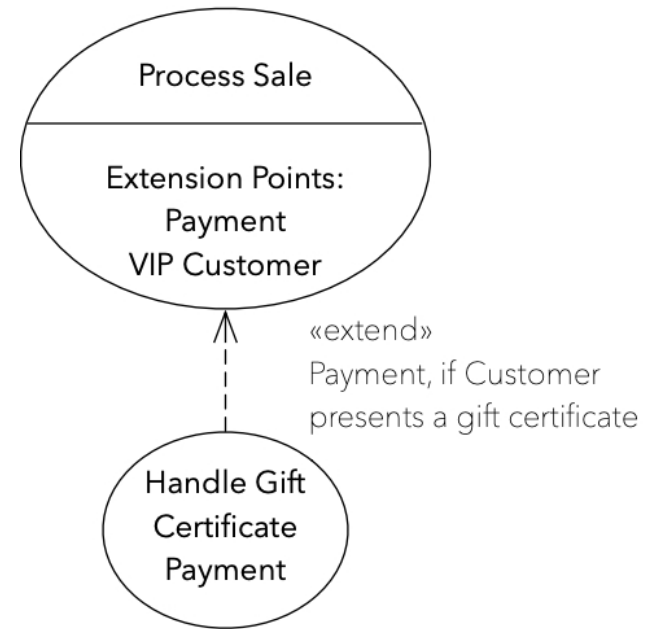
➡ The Include Relationship in UML Use Case Diagrams

The primary purpose is to avoid repetition or to decompose extremely long use cases to make them comprehensible



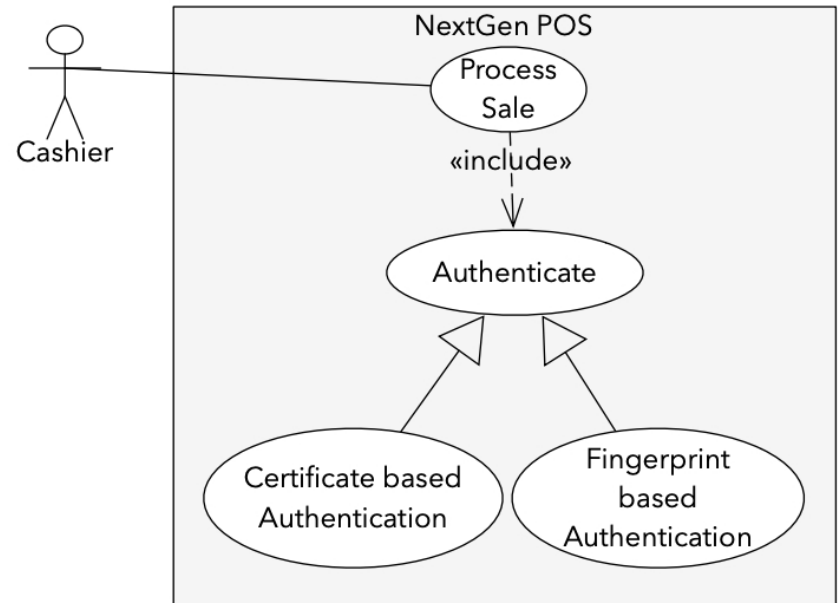
➔ The Extend Relationship in UML Use Case Diagrams

The extend relationship can be used to describe where and under what condition an extending or additional use case extends the behavior of some base use case.



➡ The Inheritance Relationship in Use Case Diagrams

The inheriting use case replaces one or more of the courses of action of the inherited use case. The inheriting use case overrides the behavior of the inherited use case.



➡ A Use case diagram looks so simple, why bother with it?

Firstly it shows the BIG-PICTURE without getting bogged down in the details of design and implementation (i.e. It's pretty much independent of hardware, OS, GUI, programming language, databases, networks etc) so it focuses on what needs to be done, not on how it will be done.

Secondly, the customer can easily relate to a use-case diagram and identify the major objectives of their business within it.

This means that any major or miss-understood functionality required from the system is less likely to be overlooked during analysis (very important) or incorrectly interpreted.

From a developers point of view they can immediately assess the functionality required and assess the risk involved in implementing each use case. Resources, hiring and training can then be allocated appropriately as part of the project planning.