SOFTWARE DESIGN AND ANALYSIS
CS 3004

Use-Case Model: Writing Requirements in Context

Fall 2021
Dr. Muhammad Bilal

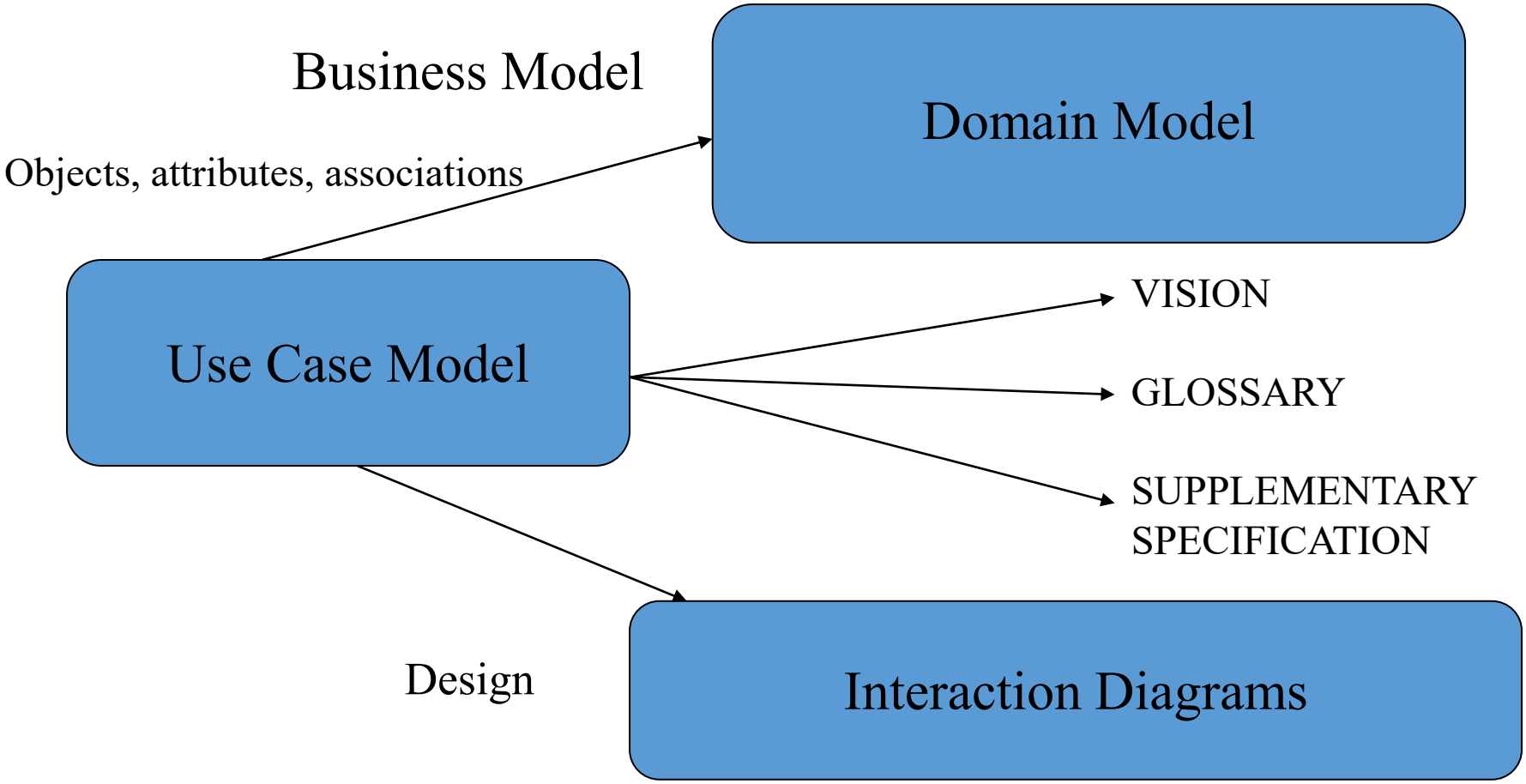
Outline

1. Use Case Relationships
2. Why Use Cases
3. Things that are in Use Cases
4. Goals and Scope of a Use Case
5. Golden Rule of Use-Case Names

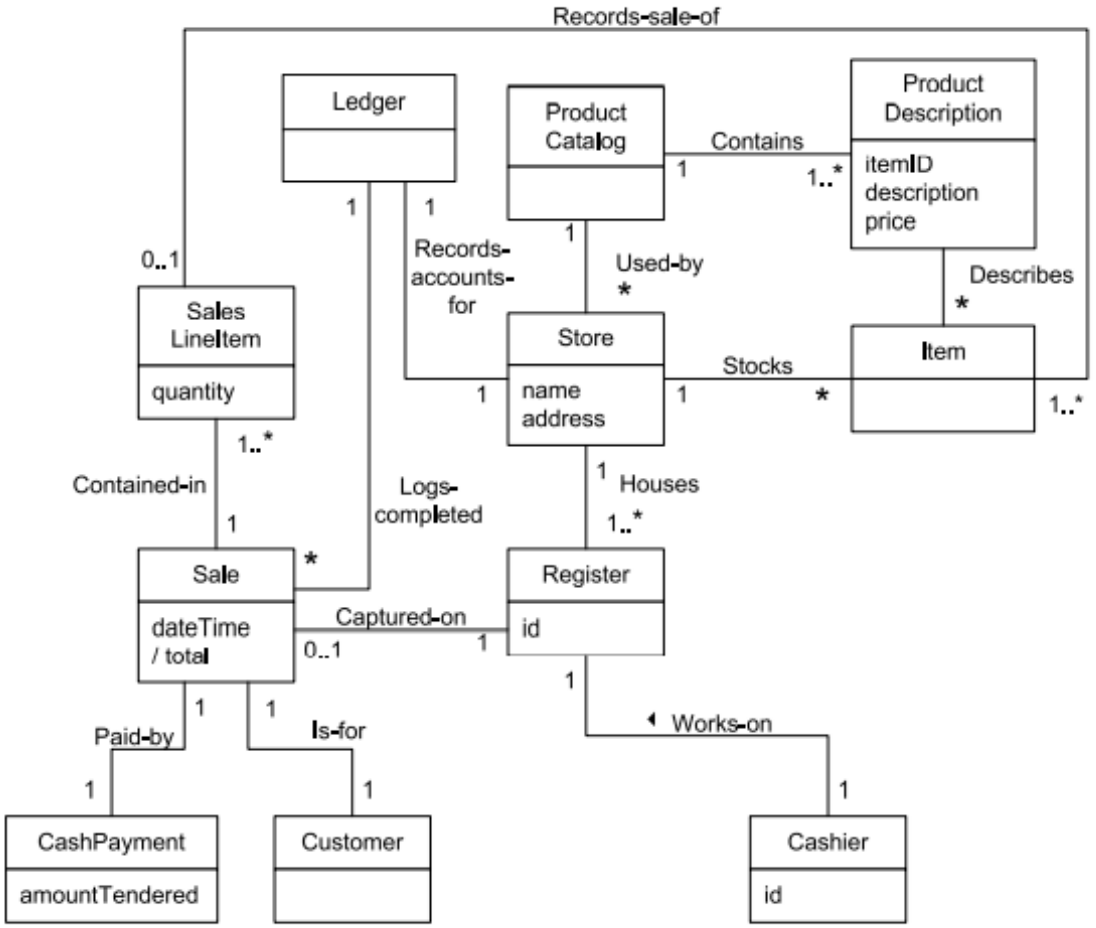
Learning Objectives

- ✔ Define the Domain model.
- ✔ Define the Use case Model.
- ✔ Identify and write use cases.

Use Case Relationships



Example: Attributes in the Domain Models



Use Cases are not Diagrams

- Use Cases may have a diagram associated with them, and a use case diagram is an easy way for an analyst to discuss a process with a subject matter expert (SME).
- But use cases are primarily text. The text is important. The diagram is optional.

Emphasize Goals

- Investigating goals rather than tasks and procedures improves information gathering by focusing on the essence of requirements—the intent behind them.
- Seeing requirements as identifying tasks to be done has a strong bias toward reproducing the existing system, even when it is being replaced because it is seriously defective.

Why Use Cases?

- Simple and familiar story-telling makes it easier, especially for customers, to contribute and review goals.
- Use cases keep it simple
- They emphasize goals and the user perspective.
- New use case writers tend to take them too seriously.

Actors or Use Case First?

- Because you have to understand each part of Use Cases, the parts are presented separately. But those who create use cases switch back and forth. The text describes use cases substantially before paying attention to actors. Typically, both actors and use cases are identified early and then examined to see if more use cases can be found from the actors, or more actors found by examining the use cases.

How Use Cases look like?

- Capture the specific ways of using the system as dialogues between an actor and the system.
- Use cases are used to
  - Capture system requirements
  - Communicate with end users and Subject Matter Experts
  - Test the system

USE CASE : Process Sale
(FULLY DRESSED VERSION)

- Primary Actor: Cashier
- Stakeholders and Interests:
- Cashier: Wants accurate and fast entry, no payment errors, …
- Salesperson: Wants sales commissions updated. …
- Preconditions: Cashier is identified and authenticated.
- Success Guarantee (Post conditions):
- Sale is saved. Tax correctly calculated….
- Main success scenario (or basic flow):
- Extensions (or alternative flows):
- Special requirements: Touch screen UI, …

Use case (contd…)

- Technology and Data Variations List:
- Identifier entered by bar code scanner,…
- Open issues: What are the tax law variations? …
- Main success scenario (or basic flow):
- The Customer arrives at a POS checkout with items to purchase.
- The cashier records the identifier for each item. If there is more than one of the same item, the Cashier can enter the quantity as well.
- The system determines the item price and adds the item information to the running sales transaction. The description and the price of the current item are presented.

Use case (contd…)

- On completion of item entry, the Cashier indicates to the POS system that item entry is complete.
- The System calculates and presents the sale total.
- The Cashier tells the customer the total. The Customer gives a cash payment (“cash tendered”) possibly greater than the sale total.
- Extensions (or alternative flows):
- If invalid identifier entered. Indicate error.
- If customer didn’t have enough cash, cancel sales transaction.

Things that are in Use Cases

- Create a written document for each Use Case
  - Clearly define intent of the Use Case
  - Define Main Success Scenario (Happy Path)
  - Define any alternate action paths
  - Use format of Stimulus: Response
  - Each specification must be testable
  - Write from actor's perspective, in actor's vocabulary

Use cases Template source: www.usecases.org

- Name
- Primary Actor
- Scope
- Level
- Stakeholders and Interests
- Minimal Guarantee
- Success Guarantee
- Main Success Scenario
- Extensions

Optional Items

- You can add some of the following items
  - Trigger (after Success Guarantee)
- (at end:)
  - Special requirements (interests of actors)
  - Technology and Data Variations
  - Frequency of Occurrence
  - Open Issues (various business decisions)

Goals and Scope of a Use Case

- At what level and scope should use cases be expressed?
- A: Focus on uses cases at the level of **elementary business process** (EBP).

EBP: a task performed by one person in one place at one time which adds measurable business value and leaves the data in a consistent state.

Approve credit order - OK.

Negotiate a supplier contract - not OK.

It is usually useful to create separate "sub" use cases representing subtasks within a base use case. e.g. Paying by credit

Elements in the Preface

- Only put items that are important to understand before reading the Main Success Scenario.

  These might include:
- Name (*Always needed for identification*)
- Primary Actor
- Stakeholders and Interests List
- Preconditions
- Success guarantee (Post Conditions)

Naming Use Cases

- Must be a complete process from the viewpoint of the end user.
- Usually in verb-object form, like Buy Pizza
- Use enough detail to make it specific
- Use active voice, not passive
- From viewpoint of the actor, not the system

Golden Rule of Use-Case Names

- Each use case should have a name that indicates what value (or goal) is achieved by the actor's interaction with the system
- Here are some good questions to help you adhere to this rule:
  - *Why would the actor initiate this interaction with the system?*
    What goal does the actor have in mind when undertaking these actions?
    What value is achieved and for which actor?

Use Case Name Examples

- Excellent - Purchase Concert Ticket
- Very Good - Purchase Concert Tickets
- Good - Purchase Ticket (insufficient detail)
- Fair - Ticket Purchase (passive)
- Poor - Ticket Order (system view, not user)
- Unacceptable - Pay for Ticket (procedure, not process)

CRUD

- Examples of bad use case names with the acronym CRUD. (All are procedural and reveal nothing about the actor's intentions.)
- C - actor Creates data
- R - actor Retrieves data
- U - actor Updates data
- D - actor Deletes data

Identify Actors

- We cannot understand a system until we know who will use it
  - Direct users
  - Users responsible to operate and maintain it
  - External systems used by the system
  - External systems that interact with the system

Types of Actors

- Primary Actor
  - Has goals to be fulfilled by system
- Supporting Actor
  - Provides service to the system
- Offstage Actor
  - Interested in the behavior, but no contribution
- In diagrams, Primary actors go on the left and others on the right.

Define Actors

- Actors should not be analyzed or described in detail unless the application domain demands it.
- Template for definition:
  - Name
  - Definition
- Example for an ATM application:

Customer: Owner of an account who manages account by depositing and withdrawing funds

Working with Use Cases

- Determine the actors that will interact with the system
- Examine the actors and document their needs
- For each separate need, create a use case
- During Analysis, extend use cases with interaction diagrams

Preconditions

- Anything that must always be true before beginning a scenario is a precondition.
- Preconditions are assumed to be true, not tested within the Use Case itself.
- Ignore obvious preconditions such as the power being turned on. Only document items necessary to understand the Use Case.

Success Guarantees

- Success Guarantees (or Post conditions) state what must be true if the Use Case is completed successfully. This may include the main success scenario and some alternative paths. For example, if the happy path is a cash sale, a credit sale might also be regarded a success.
- Stakeholders should agree on the guarantee.

Scenarios

- The Main Success Scenario, or “happy path” is the expected primary use of the system, without problems or exceptions.
- Alternative Scenarios or Extensions are used to document other common paths through the system and error handling or exceptions.

Documenting the Happy Path

- The Success Scenario (or basic course) gives the best understanding of the use case
- Each step contains the activities and inputs of the actor and the system response
- If there are three or more items, create a list
- Label steps for configuration management and requirements traceability
- Use present tense and active voice
- Remember that User Interface designers will use this specification

Note: Do not use the term “happy path” in formal documents.

Documenting Extensions

- Use same format as Happy Path
- Document actions that vary from ideal path
- Include error conditions
- Number each alternate, and start with the condition:

3A. Condition: If [*actor*] performs [*action*] the system …

- If subsequent steps are the same as the happy path, identify and label as (same)
- Steps not included in alternate course are assumed not to be performed.

Two Parts for Extensions

- Condition
  - Describe the reason for the alternative flow as a condition that the user can detect
- Handling
  - Describe the flow of processing in the same manner as the happy path, using a numbering system consistent with the original section.

Special Requirements

- If a non-functional requirement , quality attribute, or constraint affects a use case directly, describe it as a special requirement.

Technology and Data Variations List

- Often there are technical differences in how things are done even though what is done is the same. These things can be described in the Technology and Data Variations List.
- For example, if a card reader cannot read the magnetic stripe on a credit card, the cashier might be able to enter it on the keyboard.

Types of Use Cases

- The most common Use Cases are High Level Use Cases and Expanded Essential Use Cases in analysis, and Expanded Real Use Cases in design. The next slide gives definitions.
- In addition, Use Case diagrams may be used in discussions with stakeholders while capturing their requirements.

Elaborating Use Cases

- High Level Use Case (Brief)
  - Name, Actors, Purpose, Overview
- Expanded Use Case (Fully Dressed)
  - Add System Events and System Responses
- Essential Use Case (Black Box)
  - Leave out technological implications
- Real Use Case (White Box)
  - Leave in technology

Defer Decisions

- By using essential use cases as long as possible, and only using real use cases during module design, you allow time to understand the problem before you create a solution. Premature use of real use cases often confirms existing technology when a better technology might be available.

Technology

- The distinction between an essential (black box) use case that leaves out technology and a real (white box) use case that includes technology is fundamental.
- For example, in an Automated Teller Machine, an ***essential*** use case can mention identification or validation, but only a ***real*** use case can mention a key pad or card reader.

Expanded Essential Use Cases

- How to make one:
  - Step 1: Name the Use Case (system function, e.g. “enter timesheet information”).
  - Step 2: Identify the Actor(s) involved.
  - Step 3: Describe the Intent of the Use Case in language the client will understand.
  - Step 4: Identify the Assumptions and Limitations relevant to this Use Case and other Use Cases which the current one might extend or build upon.
  - Step 5: Specify the ideal flow of actions using two columns labeled “Actor Actions” and “System Responses.” Number each step. This constitutes the Happy Path for this Use Case.
  - Step 6: Identify opportunities for user error and create an Alternative Path to handle each.

Post conditions

- Post conditions (or success guarantees) state what always must be true for a use case to succeed. Avoid the obvious, but clearly document any that are not obvious. This is one of the most important parts of a use case.

Conditions and Branching

- Stick to the “Happy Path,” “Sunny Day Scenario,” Typical Flow, or Basic Flow (*all names for the same basic idea*) in the main section and defer all conditional sections and branching to the extensions or alternate flows.

Extension Use Cases

- Users appreciate simplicity, so most use cases leave out alternate courses
- You can do this by extending the use case while leaving the original use case alone

Feature Lists

- Older methods of detailing requirements tended to have many pages of detailed feature lists. Usually the details could not be seen in context.
- Current philosophy is to use a higher level of detail with use cases instead of a list.
- High level System Feature Lists are acceptable when they can give a brief summary of the system.

Use Cases not an OO idea

- Use Cases are not an Object-Oriented methodology. They are common in structured development as well.
- However, the Unified Process encourages use-case driven development.

Use-case driven development

- Requirements are primarily recorded in the Use Case model.
- Iterations are planned around implementing particular Use Cases.
- Use Case Realizations drive design.
- Use Case often influence the way user manuals are organized.

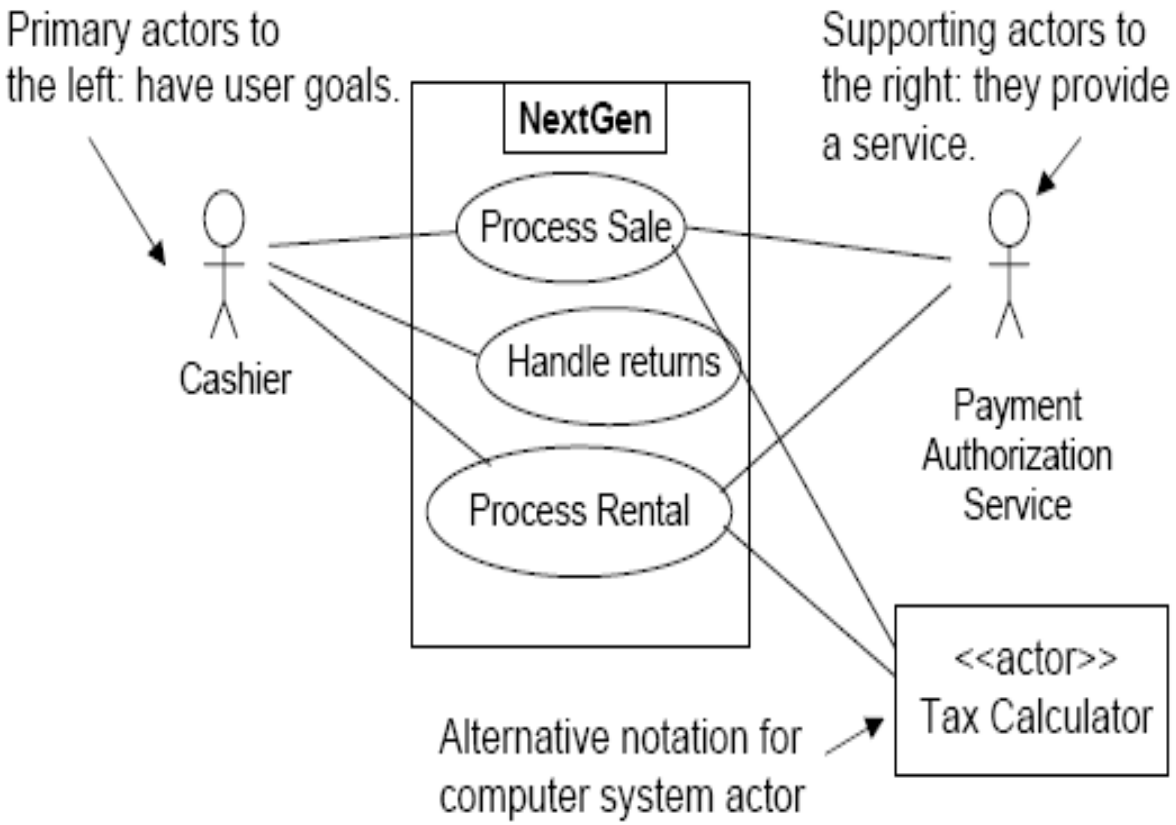
Use Cases are always wrong!

- Written documentation gives the illusion of authority and correctness, but it is an illusion.
- Use cases give a preliminary understanding that users and developers can discuss and agree on.
- But there should be constant feedback from customers in the development process to correct missing information and misinformation before it jeopardizes the functionality of the program.

Diagramming Use Cases

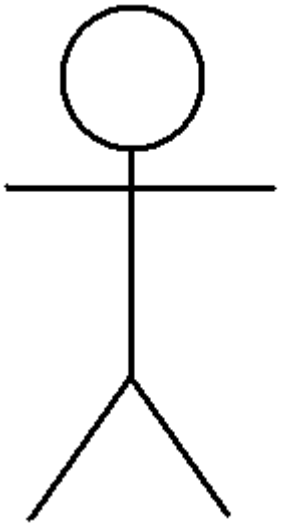
- The text is the Use Case!
- Diagrams may supplement the text or help during discovery.

Overview



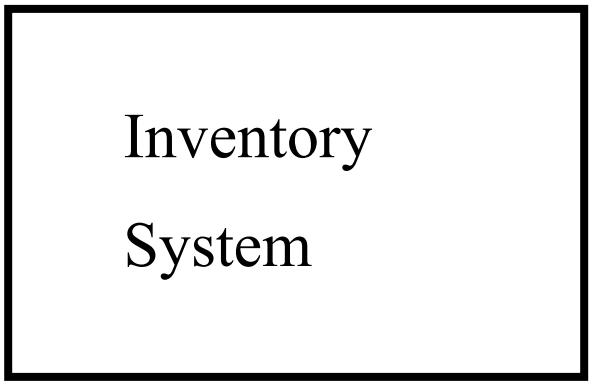
A use case diagram identifies transactions between actors and a system as individual use cases

Actor



- An actor is an idealized user of a system
- Actors can be users, processes, and other systems
- Many users can be one actor, in a common role
- One user can be different actors, based on different roles
- An actor is labeled with the name of the role

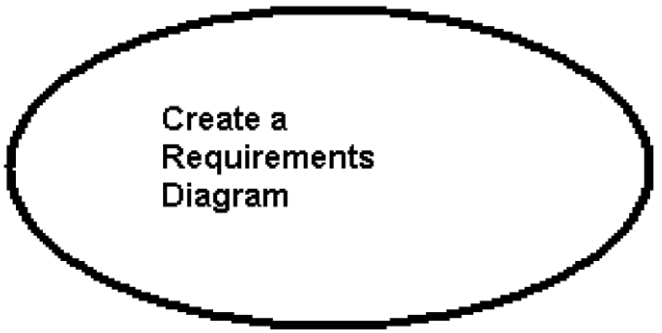
Non-human Actor



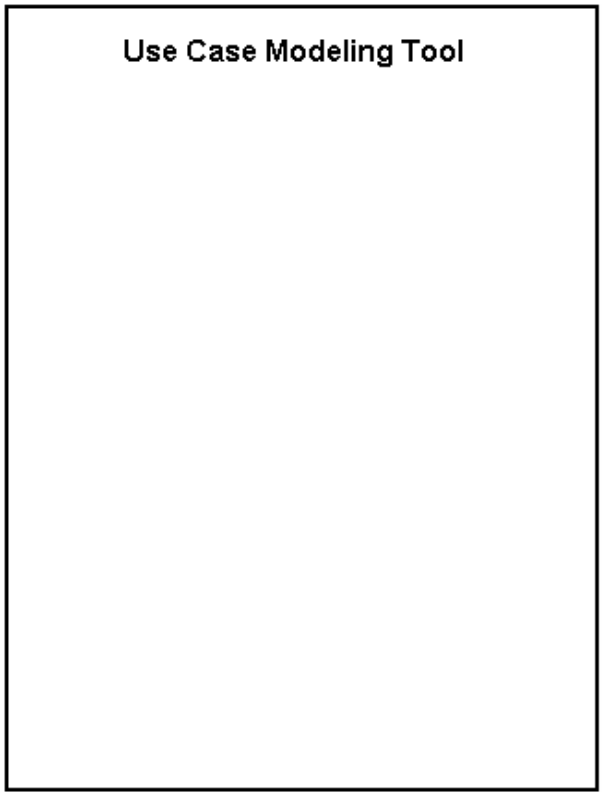
- Actors can be users, processes, and other systems.
- Show non human actors in a different manner, usually as a rectangle
- Non human actors are usually not primary users, and thus are usually shown on the right, not the left.

Use Case

- A use case is a coherent unit of externally visible functionality provided by a system and expressed by a sequence of messages
- Additional behavior can be shown with parent-child, extend and incl use cases
- It is labeled with a name that the can understand

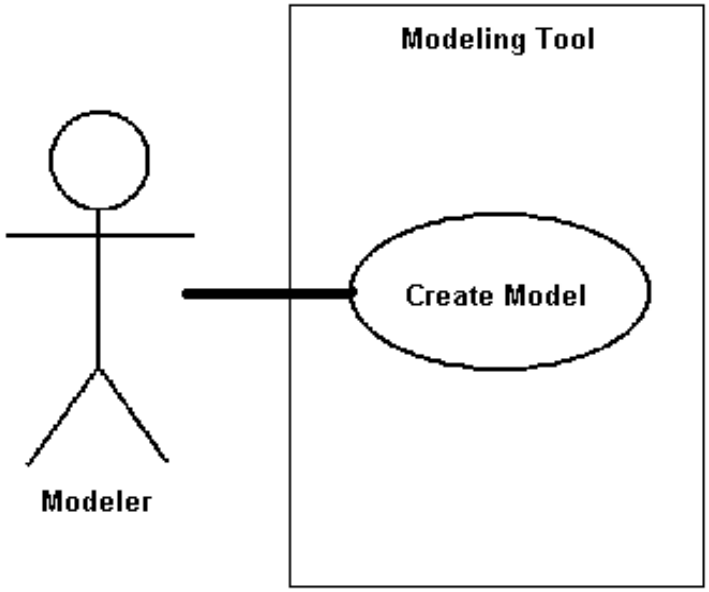


System



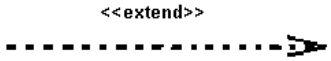
- A system is shown as a rectangle, labeled with the system name
- Actors are outside the system
- Use cases are inside the system
- The rectangle shows the scope or boundary of the system

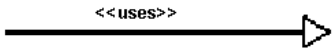
Association Relationship

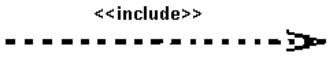


- An association is the communication path between an actor and the use case that it participates in
- It is shown as a solid line
- It does not have an arrow, and is normally read from left to right
- Here, the association is between a Modeler and the Create Model use case

Relationships in Use Cases

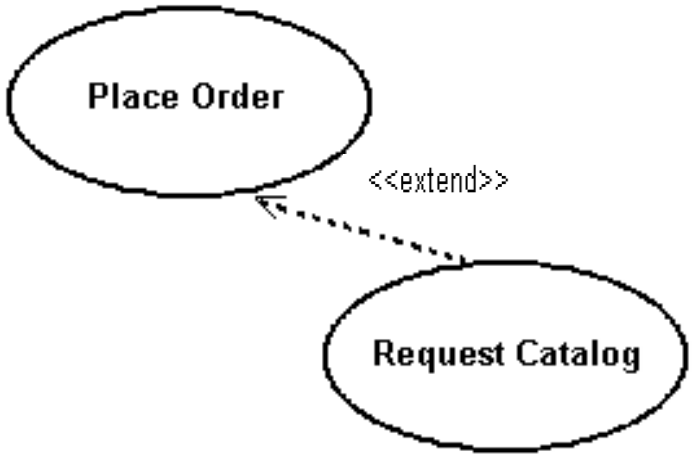
- There are several Use Case relationships:
- Association
- Extend 

- Generalization 
- Uses 

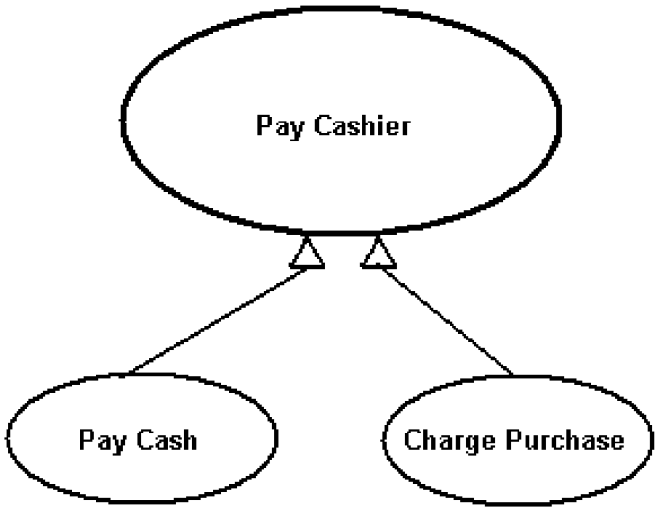
- Include 

Extend Relationship

- Extend puts additional behavior in a use case that does not know about it.
- It is shown as a dotted line with arrow point and labeled <<extend>>
- In this case, a customer can request a catalog when placing a order

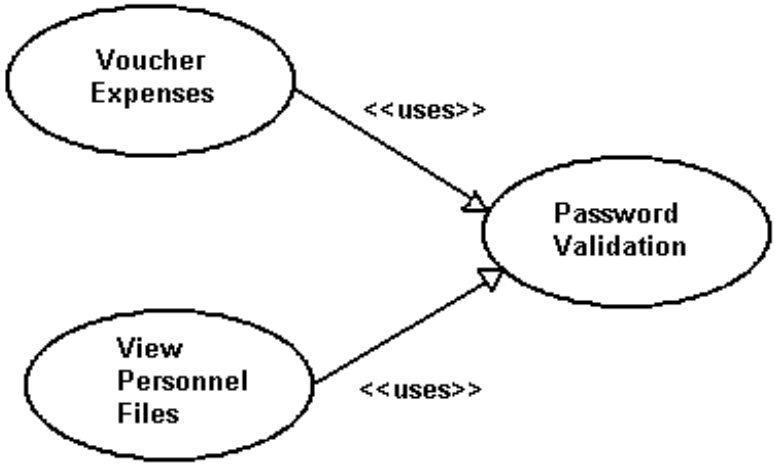


Use Case Generalization



- Generalization is a relationship between a general use case and a more specific use case that inherits and extends features to it
- It is shown as a solid line with a closed arrow point
- Here, the payment process is modified for cash and charge cards

Uses Relationship

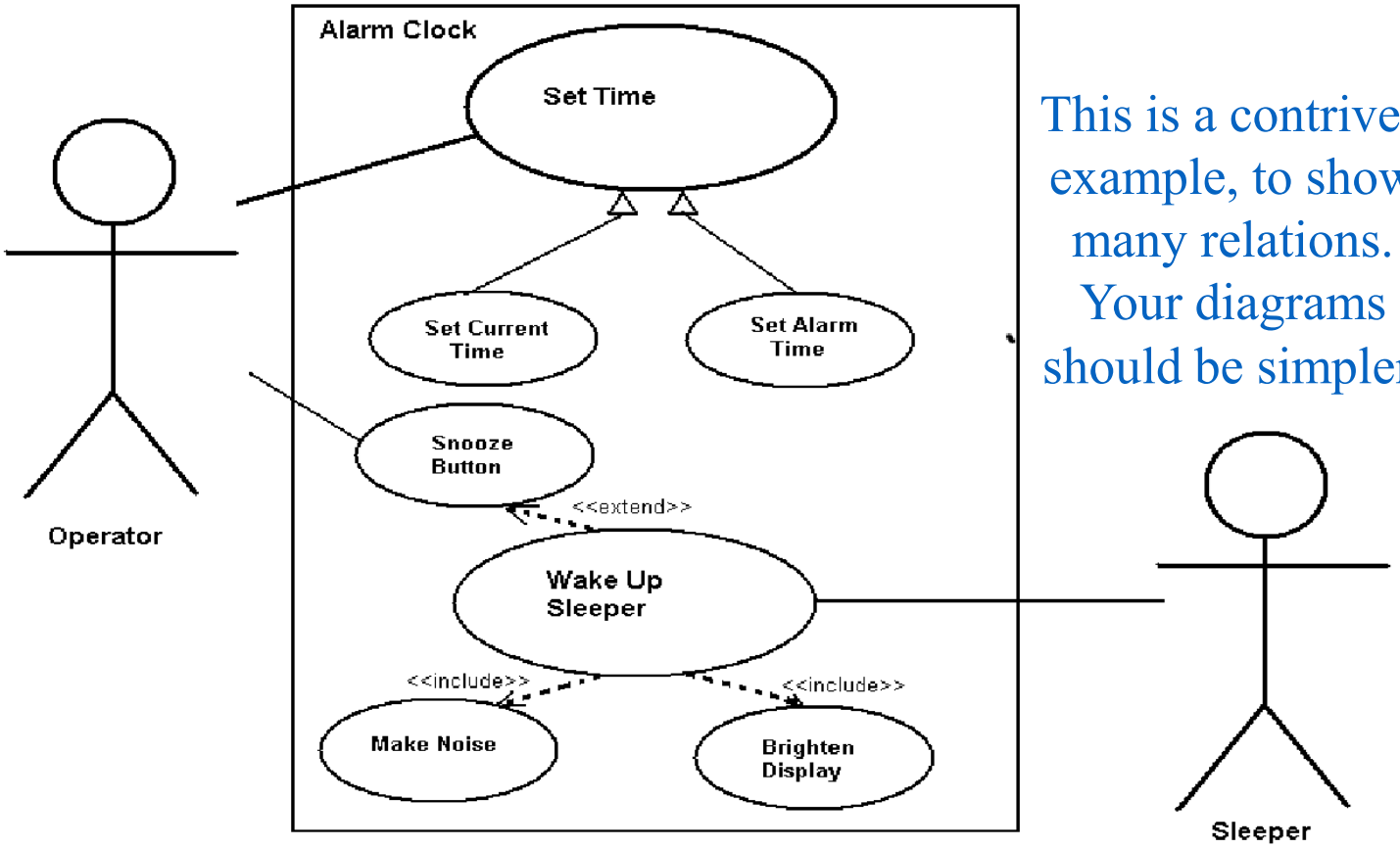


- When a use case uses another process, the relationship can be shown with the uses relationship
- This is shown as a solid line with a closed arrow point and the <<uses>> keyword
- Here different system processes can use the logon use case

Include Relationship

- Include relationships insert additional behavior into a base use case
- They are shown as a dotted line with an open arrow and the key word <<include>>
- Shown is a process that I observed in an earlier career

Use Case Example: Alarm Clock



This is a contrived example, to show many relations. Your diagrams should be simpler.

Use Case Example - POS system