Software Engineering

Unit 2: Software Requirement Elicitation and Analysis

Software Requirement Elicitation and Analysis

2.1 Software Requirement

2.2 Requirements Elicitation Techniques

- **2.2.1 FAST**
- **2.2.2 Prototyping**

2.3 Initial Requirement Document

2.4 Use Case Approach:

- **2.4.1 Creating Use Case Diagram for Requirement**
- **2.4.2 Use Case Description**
- **2.4.3 Scenario Diagrams**
- **2.4.4 Scenario Matrix**

2.5 Characteristics of Good Requirement

2.6 Software requirement specification document

- **2.6.1 Nature of the SRS document**

CE: 2.1
Software Requirement

CE: 2.1 Software Requirement

- The software requirements are the description of features and functionalities of the target system.
- Requirements convey the expectations of users from the software product.
- The requirements can be understandable or hidden, known or unknown, expected or unexpected from client's point of view.

CE: 2.1 Software Requirement (Conti…)

1. Identification of Stakeholders:

- It is a very important task.
- Every affected person directly or indirectly is a stakeholder of the system.

- Usually business suspects are:
  - ✓ Business Operation Managers
  - ✓ Product Managers
  - ✓ Marketing People
  - ✓ Internal And External Customers
  - ✓ End Users
  - ✓ Consultants
  - ✓ Product Engineers
  - ✓ Software Engineers
  - ✓ Support And Maintenance Engineers And Others.

CE: 2.1 Software Requirement (Conti…)

1. Identification of Stakeholders: (Conti…)

- Every Stakeholder has….
  - ✓ a different views of the system and
  - ✓ achieves different benefits

  - when the system is successfully developed, and is open to different risks if the development effort should fail.

- At the Inception (Beginning), the *Requirement Engineer should create a list of the people* who will going to contribute, when the requirements are elicited (produced).

- The initial list will grow from stakeholders, when they are contacted, because every stakeholder will be asked:

  "Who else do you think I should talk to?"

CE: 2.1 Software Requirement (Conti…)

1. Identification of Stakeholders: (Conti…)

- Some of the categories of stakeholders are:
  1. Internal People of Customer's organization
  2. External People of Customer's organization
  3. Internal People of Developer's organization
  4. External People of Developer's organization

CE: 2.1 Software Requirement (Conti…)

2. Functional and Non-functional Requirement:

❖ Functional Requirements:

- They describe "what" the software will do.
- They are nothing, but the expectations from the system.
- They are also called *product features*.
- They also specify, what the software will not do.

- It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.

- They help us to capture the planned behavior of the system. This behavior may be expressed as *functions*, *services* or *tasks* or *which system is required* to perform.

CE: 2.1 Software Requirement (Conti…)

2. Functional and Non-functional Requirement: (Conti…)

❖ **Non-functional Requirements:**

- They are the attributes of the software quality.
- They signify that, how well the software does, what it has to be.

- They are important for the sustainability of the system.
  Some are…..
  - ✓ Reliability,
  - ✓ Usability,
  - ✓ Maintainability,
  - ✓ Availability,
  - ✓ Portability,
  - ✓ Flexibility and
  - ✓ Testability.

CE: 2.1 Software Requirement (Conti…)

2. Functional and Non-functional Requirement: (Conti…)

❖ **Non-functional Requirements: (Conti…)**

- They are the attributes of the software quality.
- They signify that, how well the software does, what it has to be.
- They make the users happy and satisfied.
- They help us to measure the quality of the software. So, they are also called *quality attributes*.

CE: 2.1 Software Requirement (Conti…)

3. Known and Unknown Requirements:

- **Known requirements** are *the expectations of the stakeholders* and may *make stakeholders happy* if implemented correctly.

- **Unknown requirements** are forgotten by the stakeholders, because they *may not require now* or *due to limitation of domain*, *expertise* and *facilities* of available technology.
  - ✓ So, after identification, they may…
    - Add value to the system and
    - Increase the chances of acceptability and sustainability of the system.

- Therefore, every requirement is known and unknown.

CE: 2.2
Requirements Elicitation Techniques

CE: 2.2 Requirements Elicitation Techniques

- There are many techniques requirements elicitation.
- One or more techniques to *capture the requirements*.

- The selection of a technique may be based on the following:
  1. It is the only technique that we know.
  2. It is our favourite technique.
  3. We believe that a particular technique is the best for this project.

- There are few more effective techniques:
  1. Interviews
  2. Brainstorming Sessions
  3. FAST
  4. Prototyping

CE: 2.2.1 Facilitated Application Specification Technique (FAST)

- FAST is more formal than *brainstorming* sessions.

Who is Facilitator?

- A Facilitator can be a customer, developer, or an outsider who controls the meeting.

- The main objective of this technique is….
  to close the gap between developers and customer.

- It is a technique for the requirements elicitation for software development.

- It is a team-oriented approach for *gathering requirements*.

- All members of the team, are required to follow the *set procedures and guidelines*.

CE: 2.2.1 Facilitated Application Specification Technique (FAST) (Conti…)

❖ Guidelines for conducting FAST session are:

1. It must be conducted at neutral site.
   [Because all the members of joint team i.e. developers and customers travel to that site.]
2. The framed rules for participation, must be circulated to all the members in advance.
3. All members must be feel comfortable to encourage free flow of ideas.
4. The facilitator (organizer) gives the overview of the project.
5. A display mechanism like.... Projector, Wall Stickers, Flip Charts, Whiteboards, etc. should be available in the committee room, where the meeting is conducted.
6. All members should be directed to give their views. Unnecessary delay should be avoided.

CE: 2.2.1 Facilitated Application Specification Technique (FAST) (Conti…)

❖ Preparation of FAST session:

- Each member is required to make …

1. A list of objects

✓ a part of the environment
✓ produced by the system
✓ used by the system

2. A list of services

✓ that manipulate or interact with the objects.

3. A list of constraints

✓ cost, size, etc.

4. Performance criteria

✓ speed, accuracy, etc.

CE: 2.2.1 Facilitated Application Specification Technique (FAST) (Conti…)

❖ Activities of FAST session are:

1. Each member presents his/her lists of objects, constraints, services and performance for discussion.

2. A small group is constituted to prepare a consolidated list after removing redundant entries.

3. The list is further modified, if required in order to prepare a agreement list.

4. A few small groups are created to draft mini-specifications.

5. Each sub-team presents mini-specifications to all FAST attendees.

CE: 2.2.1 Facilitated Application Specification Technique (FAST) (Conti…)

❖ Activities of FAST session are:

6. Some issues may not be resolved during the meeting.

7. A validation standard is also decided for every requirement.

8. The final draft is prepared considering all inputs of FAST meeting.

- It is a popular traditional technique for the requirements.
- It helps us to prepare the **IRD** in limited time frame under the leadership.

CE: 2.2.2 Prototyping

- It is the rapid development of a system for the purpose of understanding requirements.
- It may be costly for us, but the overall development cost may be reduced.
- Developers generally use this prototyping, to refine the requirements and prepare the final specification document.
- Through prototyping, views of customers are easily incorporated in the IRD.
- The experience and feedback gathered from developing and using the prototype helps in developing the actual system.
- Therefore, the developers should develop the prototype as early as possible to speed up the software development process.

CE: 2.3
Initial Requirements Document

CE: 2.3 Initial Requirements Document (IRD)

- After the requirements are captured, IRD may be prepared.
- IRD is used to document and to list the initial set of requirements gathered through various stakeholders.
- The template for making IRD is…

Title of the project	
Stakeholders involved in capturing requirements	
Techniques used for requirement capturing	
Name of the persons along with designations	
Date	
Version	
Consolidated list of initial requirements:	

CE: 2.4
Use case approach

CE: 2.4 Use case approach

- Use cases address only the functional requirements which explain the expectations of users.

- Functional requirements express…
  “what do we expect from the system”
  without bothering about…
  “how it will be implement”.

- Use cases capture …
  the expectations in terms of achieving goals and interactions of the users with the system.

- There are three interchangeably terms…
  Use case, use case scenario and use case diagram

CE: 2.4 Use case approach (Conti…)

What are use cases?

- Use cases are ***structured outlines or templates for the description of requirements***, written in a natural language like English.

What is use case scenario?

- A use case scenario is ***an instance of a use case***.

What are use case diagrams?

- Use case diagrams are ***graphical representations that may be decomposed*** into further ***level of abstraction***.

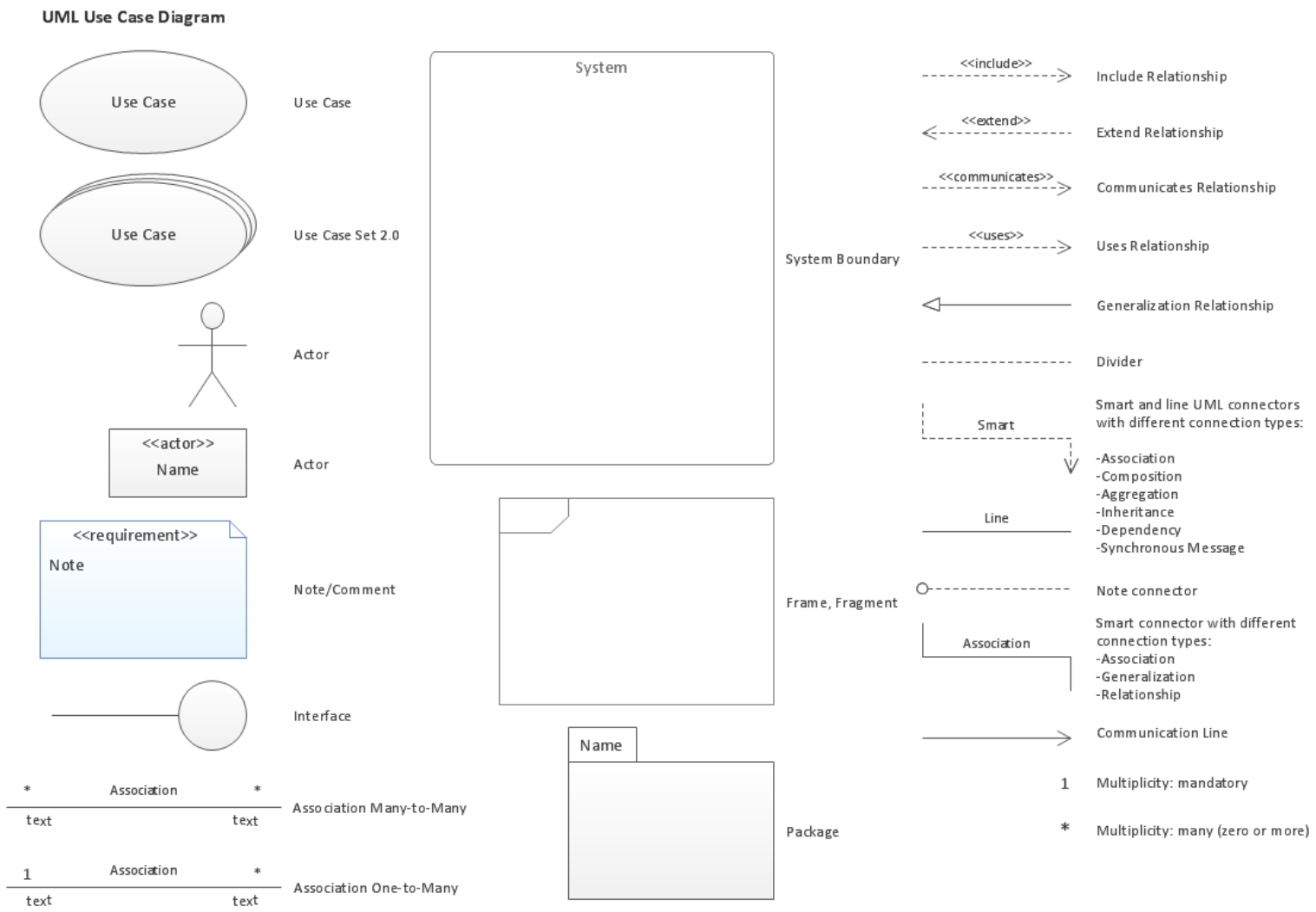
CE: 2.4 Use case approach

1. Use Cases and Actors
2. Identification of Actors
3. Identification of System Use Cases
4. Defining Relationships between System Use Cases
5. **Use Case Diagram**
6. **Use Case Description**
7. **Generation of Scenario Diagrams**
8. **Creation of Use Case Scenario Matrix**

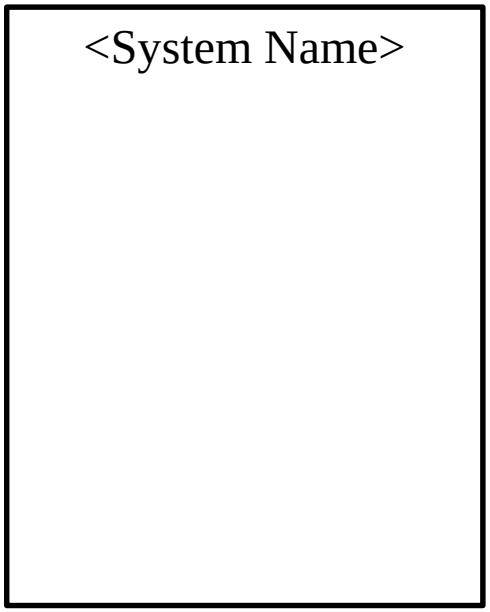
CE: 2.4.1 Use Case Diagram

- Use case diagrams are ***graphical representations that may be decomposed*** into further ***level of abstraction***.
- It is also used to present functionality of the system, but for proper explanation, it should read along with use cases.
- It also show the relationship of use cases and actors.
- It also explains what happens when an actor interacts with the system.
- For small systems, ***one use case diagram may be sufficient to represent the whole system.***
- And for large systems, ***many use case diagrams may be represent for*** various portions of the system.

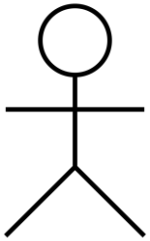
CE: 2.4.1 Use Case Diagram



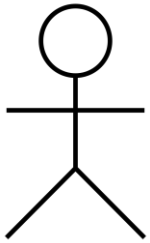
CE: 2.4.1 Use Case Diagram (Conti…)



System Boundary



Actor



<Actor Name>

Actor

Use Case

OR

Relationship/Association between Actor and Use Case

Generalization/ Inheritance

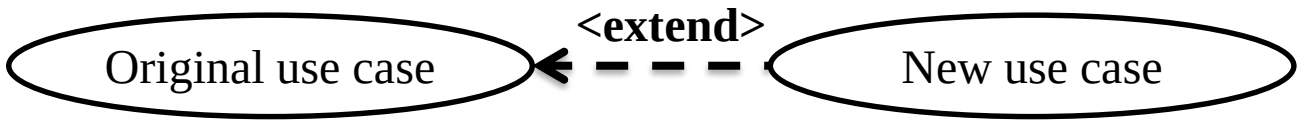
Extend Relationship

Include Relationship

CE: 2.4.1 Use Case Diagram (Conti…)

What is Extend Relationship?

- This relationship is used to ***extend*** the functionally of the ***original use case***.



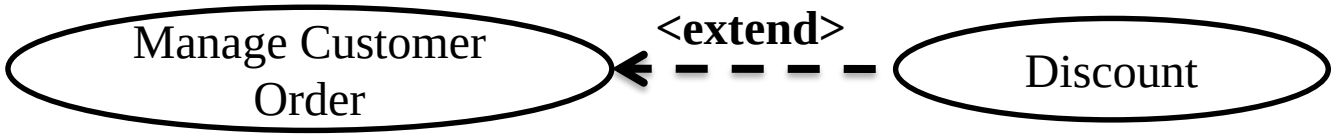
- It allows to show an ***optional*** or ***alternative*** or ***special*** part of the use case.

- An *optional system behavior is executed*, only if certain conditions are hold, *otherwise the optional behavior is not executed*.

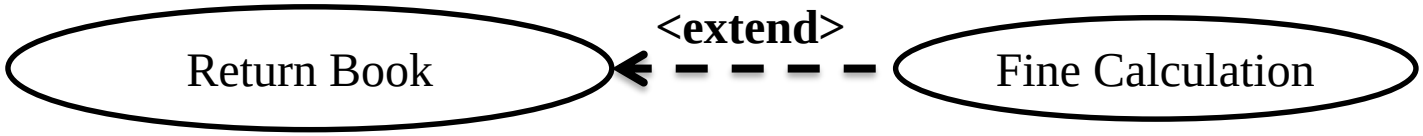
CE: 2.4.1 Use Case Diagram (Conti…)

❖ **Example of Extend Relationship:**

- In any Store Management System…..



- In Library Management System…..

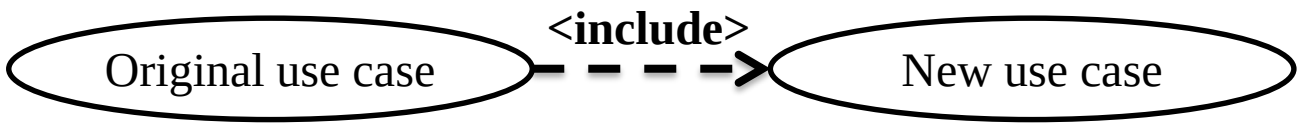


Therefore, the Extend Relationship happens sometimes, not every time.

CE: 2.4.1 Use Case Diagram (Conti…)

What is Include Relationship?

- This relationship implies, ***one use case*** includes the behavior of ***another use cases***.

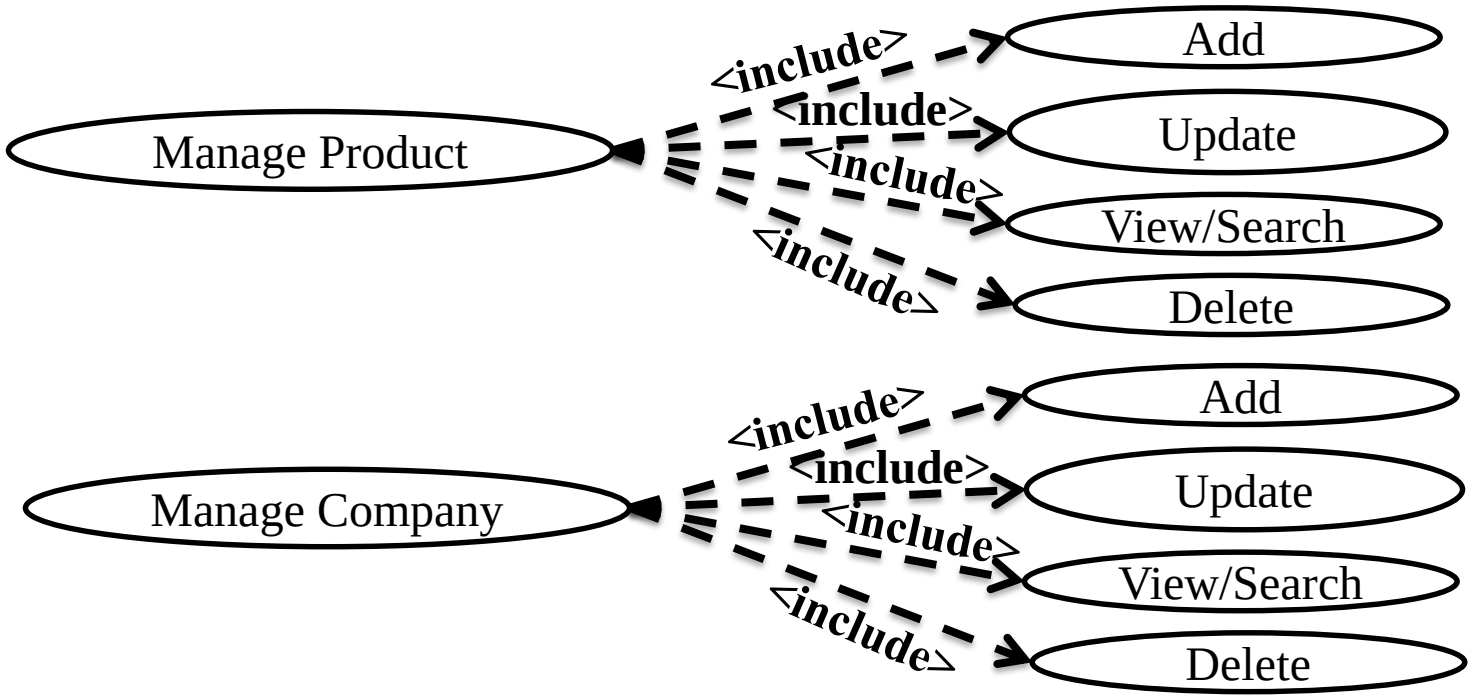


- By using this relationship,
  - ✓ the ***redundant*** and ***repeated functionalities*** among different use cases can be modelled into a single use case.
  - ✓ That means, there must be a common text in two or more use cases.

CE: 2.4.1 Use Case Diagram (Conti…)

❖ **Example of Include Relationship:**

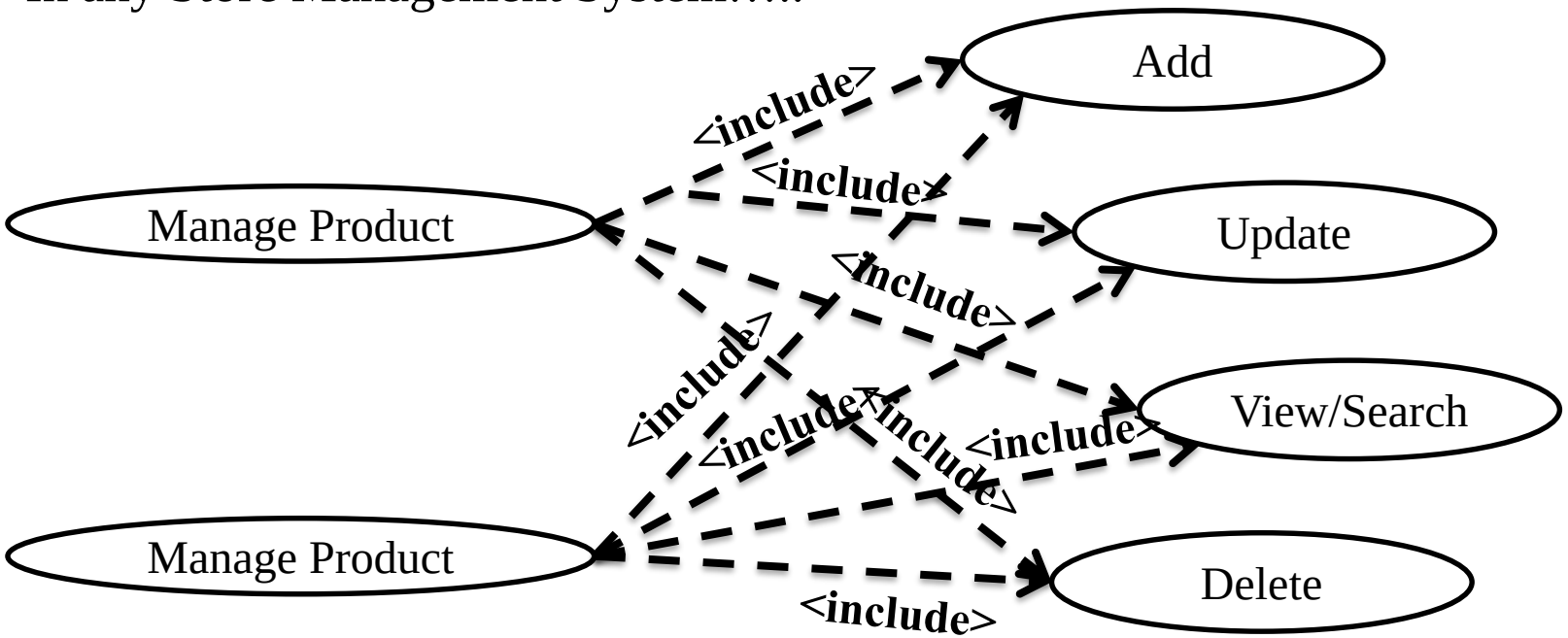
- In any Store Management System…..



CE: 2.4.1 Use Case Diagram (Conti…)

❖ **Example of Include Relationship:**

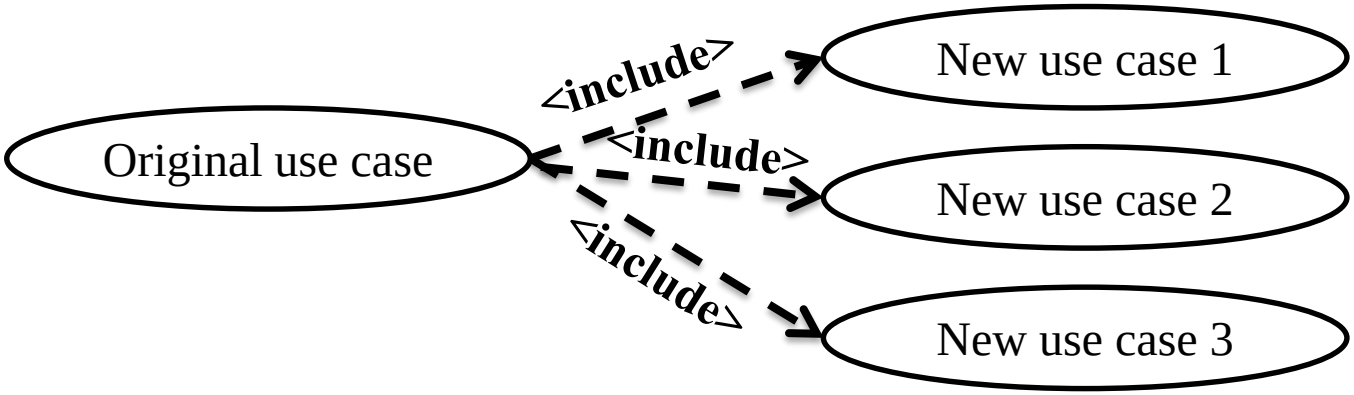
- In any Store Management System…..



Therefore, the Include Relationship happens many times.

CE: 2.4.1 Use Case Diagram (Conti…)

What is Include Relationship? (Conti…)

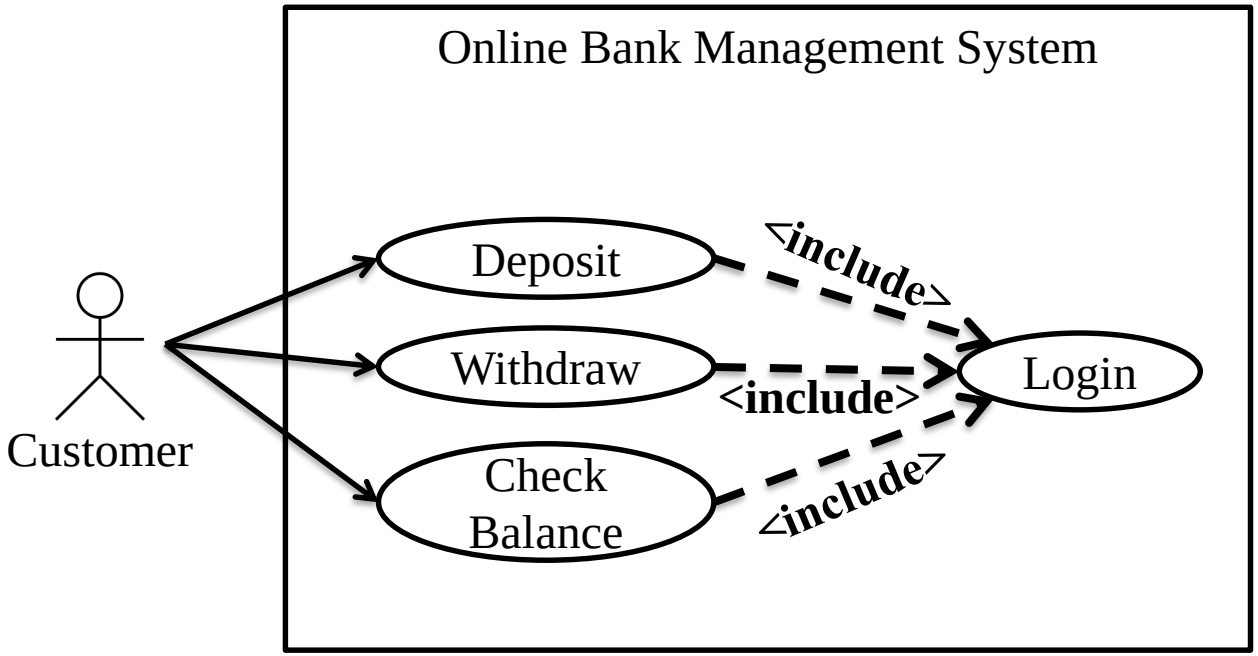


- Therefore, in ***include*** relationship, original use case may include several use cases.

CE: 2.4.1 Use Case Diagram (Conti…)

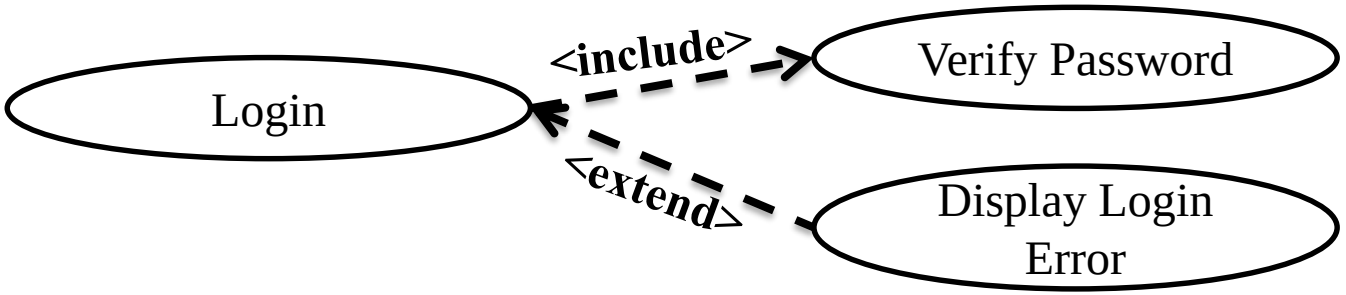
❖ **Example of Include Relationship:**

- For any bank customer…..

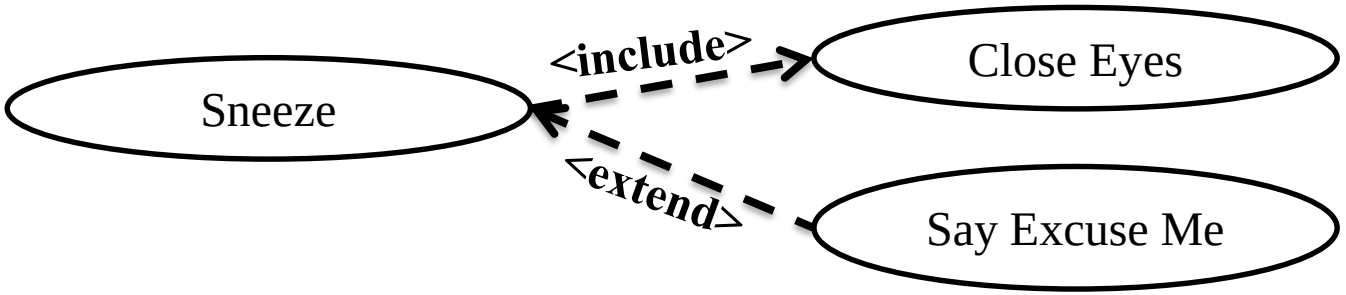


CE: 2.4.1 Use Case Diagram (Conti…)

❖ **Example of Extend and Include Relationship:**

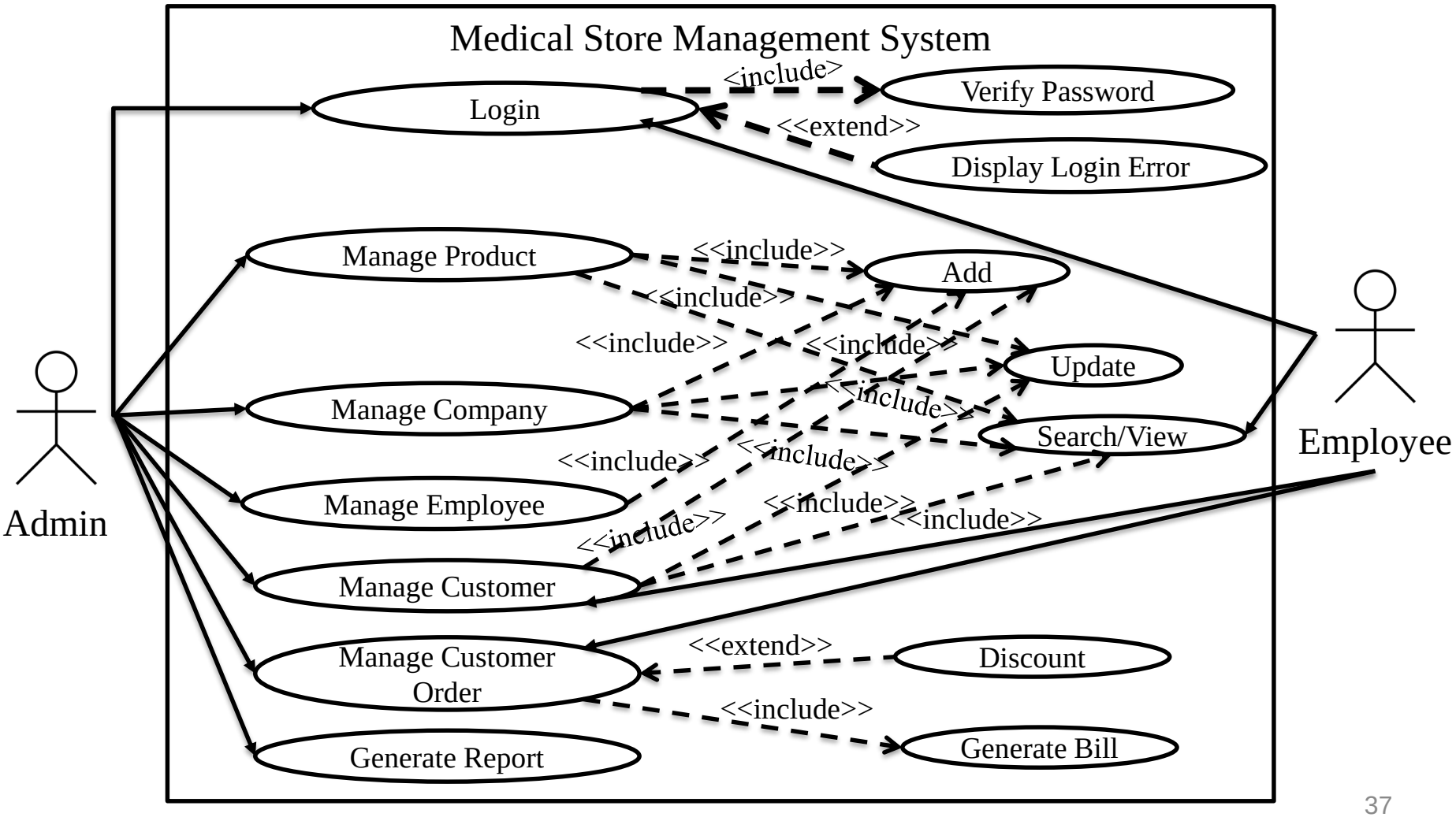


Like in real life….



CE: 2.4.1 Use Case Diagram (Conti…)

Use Case Diagram for Medical Store Management System



CE: 2.4.1 Use Case Diagram (Conti…)

❖ Tips for drawing a use-case diagram:

1. It should be as simple as possible.
2. It should be complete.
3. It should represent all interactions with the use case.
4. If there are too many use cases or actors, then only the essential use cases should be represented.
5. It should describe at least a single module of a system.
6. If the use case diagram is large, then it should be generalized.

CE: 2.4.2 Use Case Description

Template 01

- **Jacobson's Use Cases Template:**

1.	Brief description:
2.	Actors:
3.	Flow of Events: 3.1. Basic flow: 3.2. Alternative flow:
4.	Special requirements:
5.	Precondition:
6.	Postcondition:
7.	Extension points:

This template captures the requirements in a disciplined and effective way and has become a popular format.

CE: 2.4.2 Use Case Description

Template 02

- **Alternative Use Cases Template:**

1.	Introduction:
2.	Actors:
3.	Precondition:
4.	Postcondition:
5.	Flow of Events: 5.1. Basic flow: 5.2. Alternative flow:
6.	Special requirements:
7.	Associated use cases:

This template is also use by many software organizations.

CE: 2.4.2 Use Case Description

❖ **Alternative Use Cases Template: (Conti…)**

What will come in Preconditions?

- The conditions that must hold ***for the use case to begin***.

What will come in Postconditions?

- The conditions that must hold ***once the use case has completed***.

What is Basic flow?

- The most frequent scenario or scenarios of the use case.

What is Alternative flow or exception flows?

- The scenarios that are less frequent. The exception flows may reference extension points.
- Generally, it represents, that are not directly in support of the goals of the basic flow.

CE: 2.4.2 Use Case Description

❖ **Alternative Use Cases Template: (Conti…)**

What are special requirements?

- The requirements that are not related to the functionality of the system. These include constraints on the performance of the system, its implementation, the hardware platforms it runs on, and so on.

CE: 2.4.2 Use Case Description

Use case descriptions of manage product use case

1.	**Introduction:** This use case documents must be followed to maintain the details of product.
2.	**Actors:** Admin, Employee
3.	**Precondition:** The Admin/Employee *must be logged on to the system, before the use case begins.*
4.	**Postcondition:** *If the use case is successful*, then the admin will be able to maintain the product details and the employee with be able to view and search the product details, *otherwise the system state remains unchanged.*
5.	**Flow of Events:** **5.1. Basic flow:** 1. The admin shall be able to enter (or add) new product details, category and sub-category wise. 2. The admin shall be able to update the product details, category and sub-category wise. 3. The admin and employee shall be able to view and search the product details, category and sub-category wise.

CE: 2.4.2 Use Case Description

Use case descriptions of manage product use case

5.	4. When the new product details is added, the information is saved into the database. 5. When the existing product details is changed or updated, the product information is updated in the database. 6. When any existing product is searched with its category and sub-category, a particular product details shall be viewed or displayed on the screen. **5.2. Alternative flow:** 1. If the admin login is not successful, then the admin shall not be able to maintain product details.
6.	**Special requirements:** The product details must be viewed with in 5 seconds.
7.	**Associated use cases:** Login, Add, Update, View/ Search

CE: 2.5
Characteristics of Good Requirement

CE: 2.5 Characteristics of Good Requirement

- The requirements can be good, if we consider the following characteristics:
  1. Correct
  2. Unambiguous
  3. Complete
  4. Consistent
  5. Verifiable
  6. Modifiable
  7. Clear (concise, terse, simple, precise)
  8. Feasible (realistic, possible)
  9. Necessary
  10. Understandable

CE: 2.6
Software requirement specification document

CE: 2.6 Software requirement specification document

- After requirements elicitations,
  We prepare IRD , which gives overview of the system.

- After IRD is prepared,
  the detailed documentation is to be prepared on the basis of IRD, which is know as software requirements specifications (SRS) document.

- SRS is used as a legal document which acts as a contract between customers and developers.

- On the basis of this document, the developers know what to build, and the customers know what to expect.

- **We use IEEE standard 830-1998 format for preparing the SRS document.**

CE: 2.6.1 Nature of the SRS document:

- While preparing SRS, the following issues shall be addressed by the SRS writer:

1. Functionality:

- What are functions that supposed to perform by the software?

2. External interface:

- With what external entities does the system interact?
  [External entities like number of users, response time, recovery time, processing time, etc.]

3. Performance:

- How the software address on performance issue?
  [address like people, hardware, database, etc.]

CE: 2.6.1 Nature of the SRS document: (Conti…)

4. Quality attributes:

- What are the non-functional requirements in the system?

  [availability, correctness, maintainability, portability, reliability, security, testability, etc.

  These non-functional requirements should also be properly placed in the SRS documents.]

5. Design constraints imposed on implementation:

- All the constraints should be highlighted which have an impact on implementation.

  [like operating system environment, programming language, database, networking, etc.]

- It may also includes report format, naming of data, accounting procedures, audit tracing, etc.

CE: 2.6.1 Nature of the SRS document: (Conti…)

- The SRS writer(s) should not include design and implementation details.
- And also it should be written in a very simple, clear, unambiguous language, which may be understandable to all developers and customers.

CE: 2.6.2 IEEE standard 830-1998 format

- Few important SRS document outlines are:
  - ✓ Purpose
  - ✓ Scope
  - ✓ Definition
  - ✓ Hardware and software interfaces
  - ✓ User characteristics
  - ✓ Functions (in the form use case description)
  - ✓ Design constraints
  - ✓ Software system attributes (in the form non-functional requirement)


Thank You