



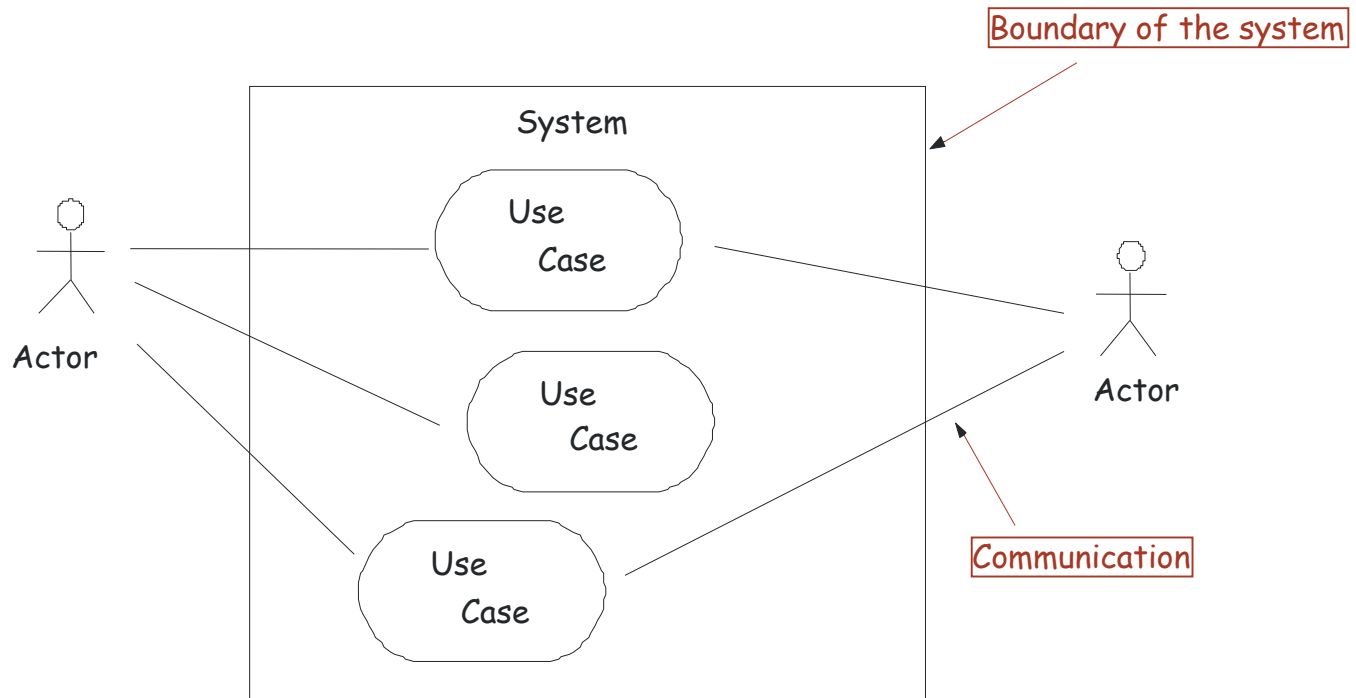
# Chapter: Use cases Models

- Actors & Use Cases
- Relationships
- Use cases Diagrams
  - Context Diagram &
  - Initial Diagram
  - Specification Templates
  - Construction Process

# Use Cases

- A technique to capture how an existing system works or how a future system should work
- Use to capture functional requirements.

## Notation

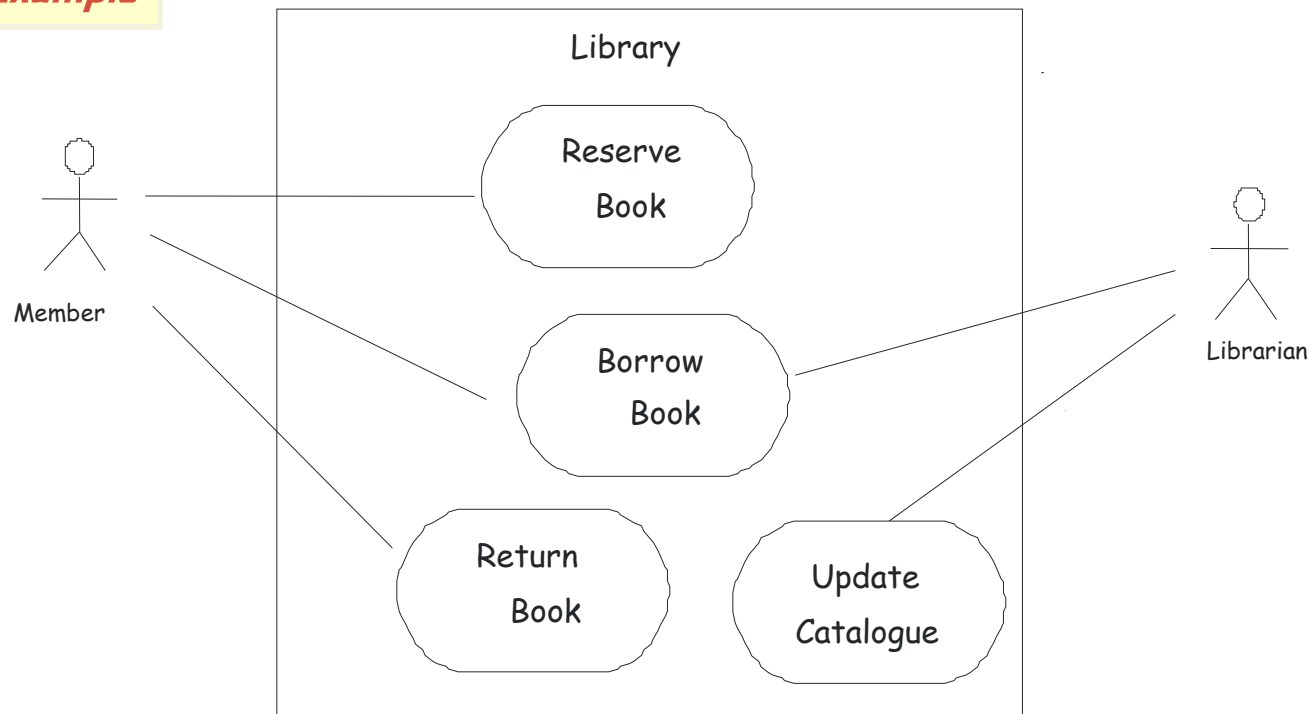


# Actors and Use cases

**Actor:** Entity (Human, Device or another software system) exchanging information with the system

**Use case:** Consists of the sequence of transactions/messages that actors and the system exchange when a given functionality of the system is executed

## Example



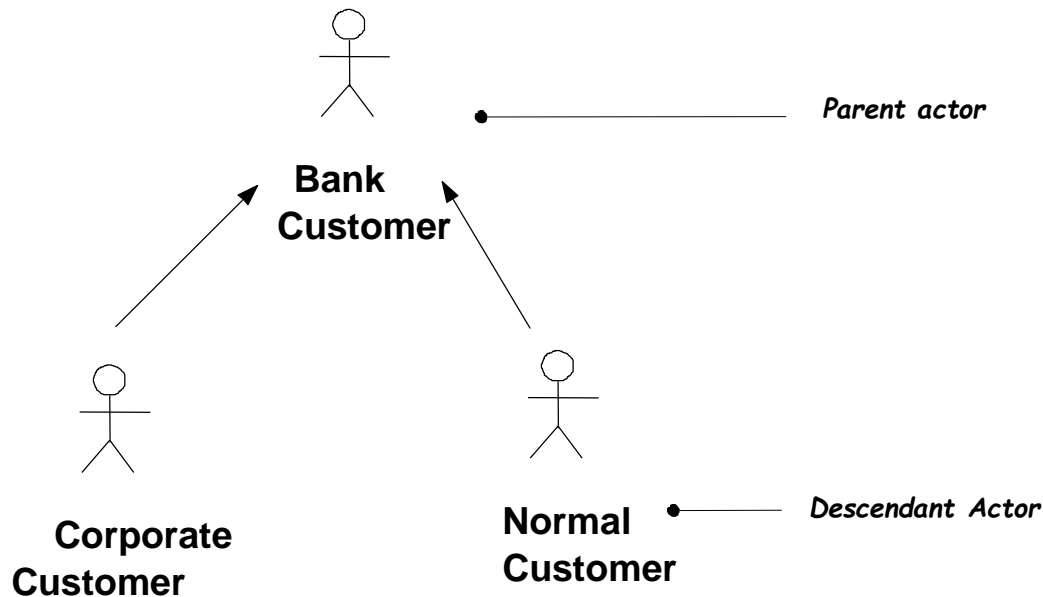
# Description Templates

- Use cases are described using templates and natural language

<i><u>Use Case</u></i>	
<i>Actors</i>	
<i>Summary</i>	
<i>Preconditions</i>	
<i>Postconditions</i>	
<i>Includes</i>	
<i>Extends</i>	
<i>Inherits from</i>	
<i>Flow of events</i>	
<i>Actor</i>	<i>System</i>

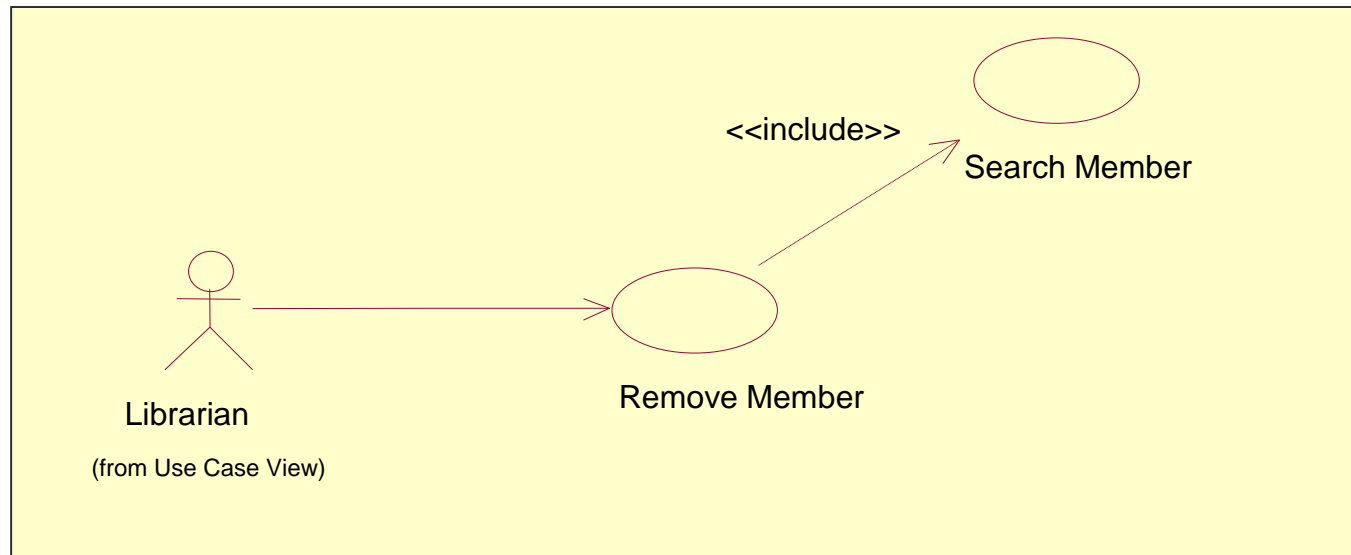
# Relationships between actors- Inheritance

- The inheritance relationship indicates that the descendant actor may play all the roles of its predecessor actor.



# Relationships between use cases- Inclusion

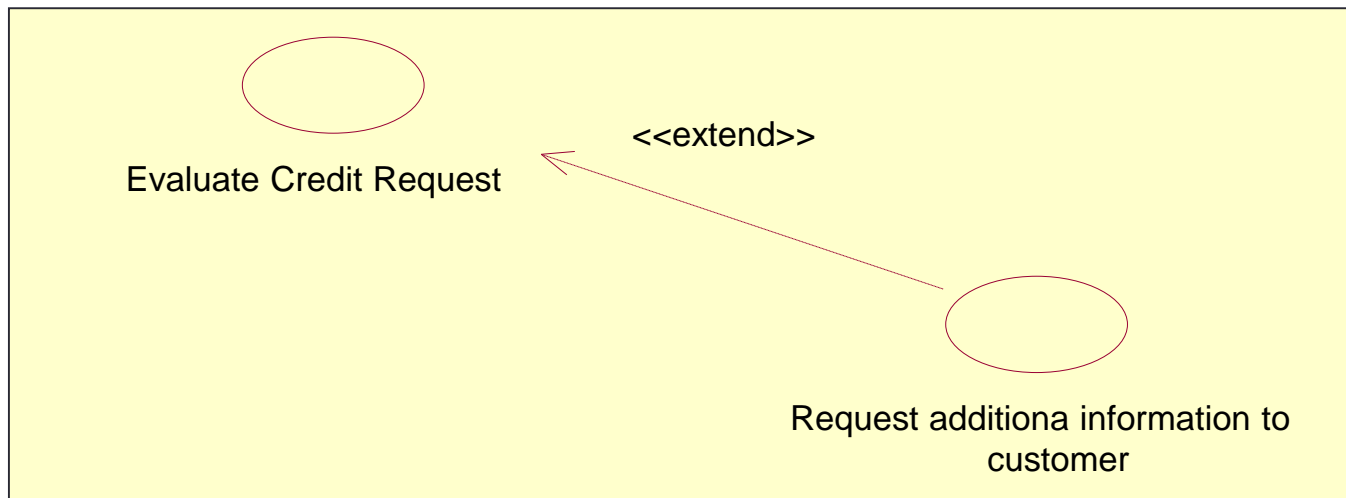
- A use case A includes a use case B, if an instance of A executes all the events that are described in B.



*The instantiation of Remove Member always uses the flow of events defined in Search Member*

# Relationships between use cases- Extension

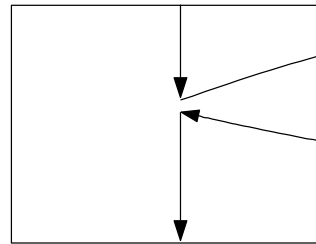
- A use case B extends a use case A, if there is a condition in the description of A that if evaluated to true all the events described in B are executed.



# Relationships between use cases

- Inclusion: The description of use case A includes a reference to B

Flow of events use case A

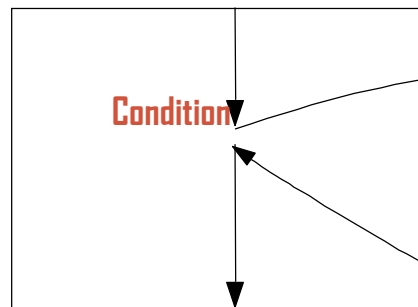


Flow of events use case B

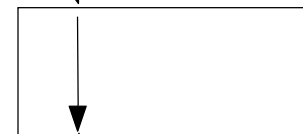


- Extension: Equivalent to an inclusion + a condition

Flow of events use case A



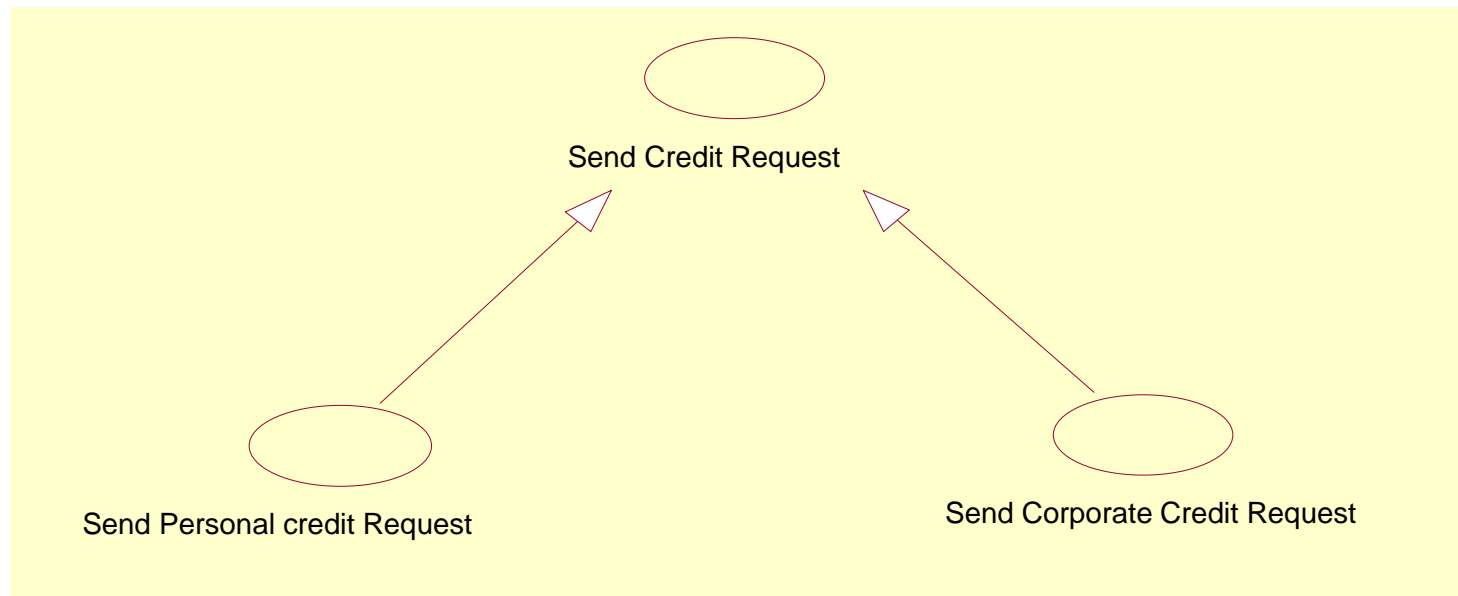
Flow of events use case B





# Relationships between use cases- Inheritance

- A use case B is an specialization of another use case A, if the flow of events in B is a refinement of the flow of events in A
  - Similar to inheritance in OO (it allows the separation between a generic interaction pattern (parent use case) and a more specific case (descendant case)).

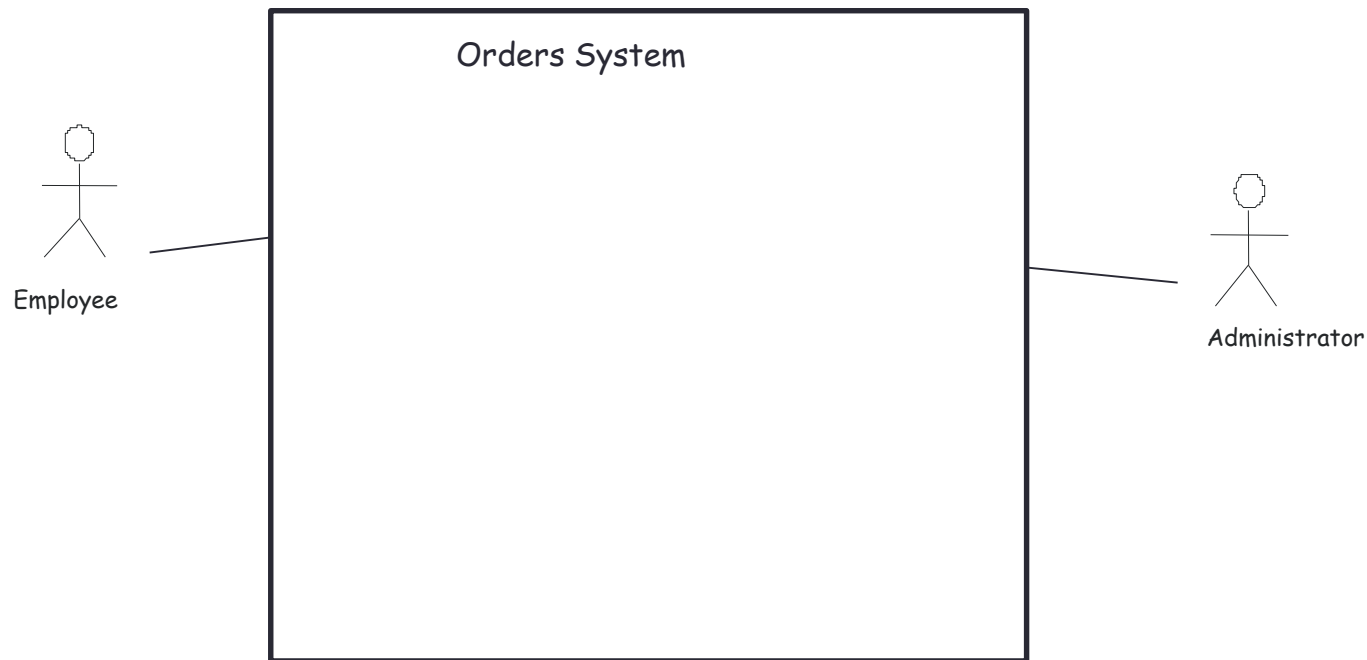


# Use cases Diagrams

- Structured in three layers
  - Context diagram and Initial context
  - Description templates
  - Structured Diagram

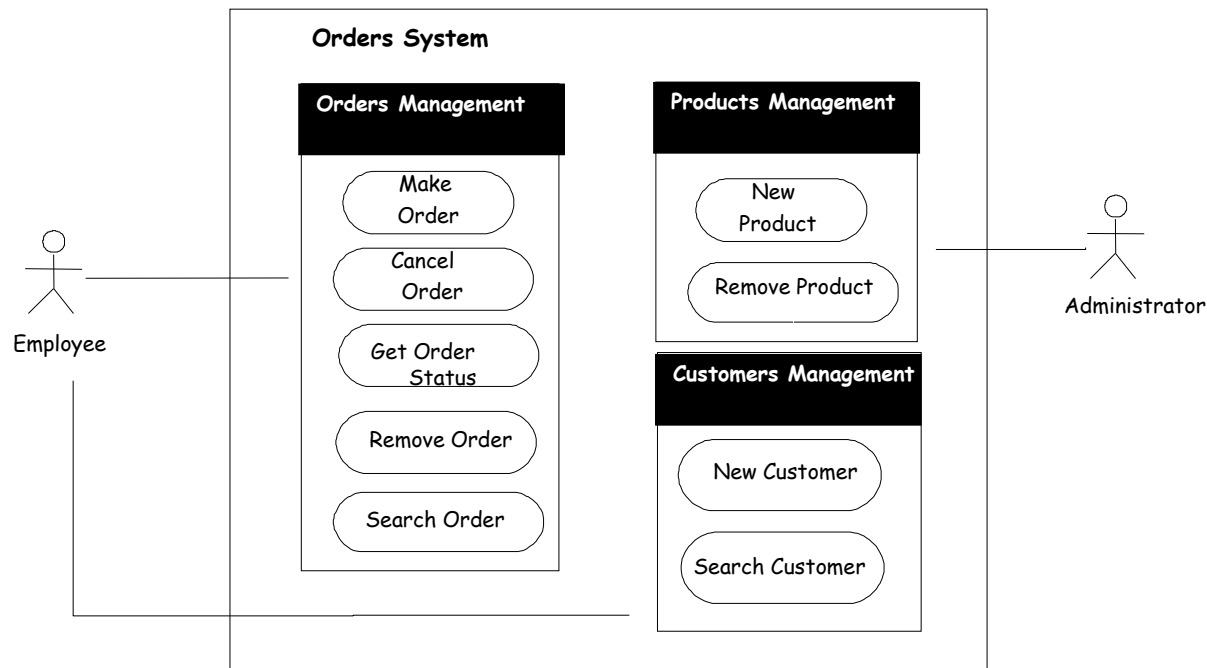
# Context Diagram

- It shows the boundaries of the system and the interacting actors

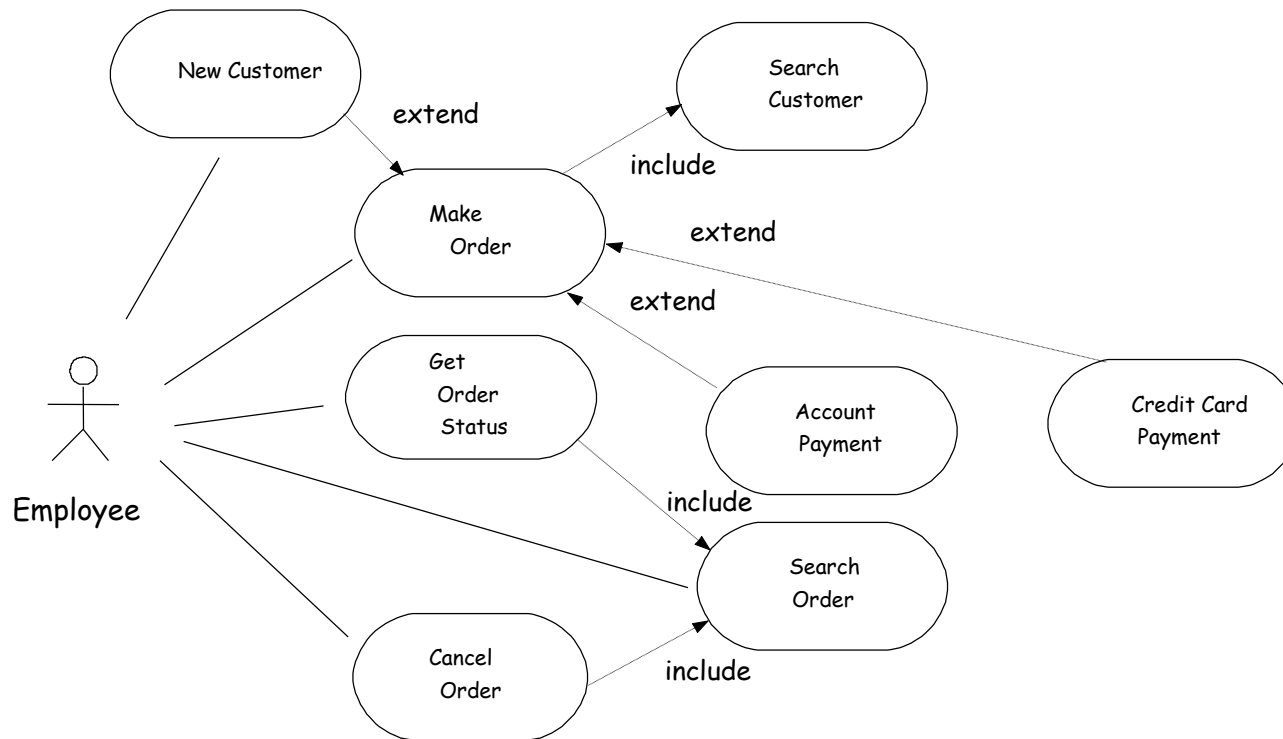


# Initial Diagram

- It contains a grouping of the main use cases



# Structured Diagram

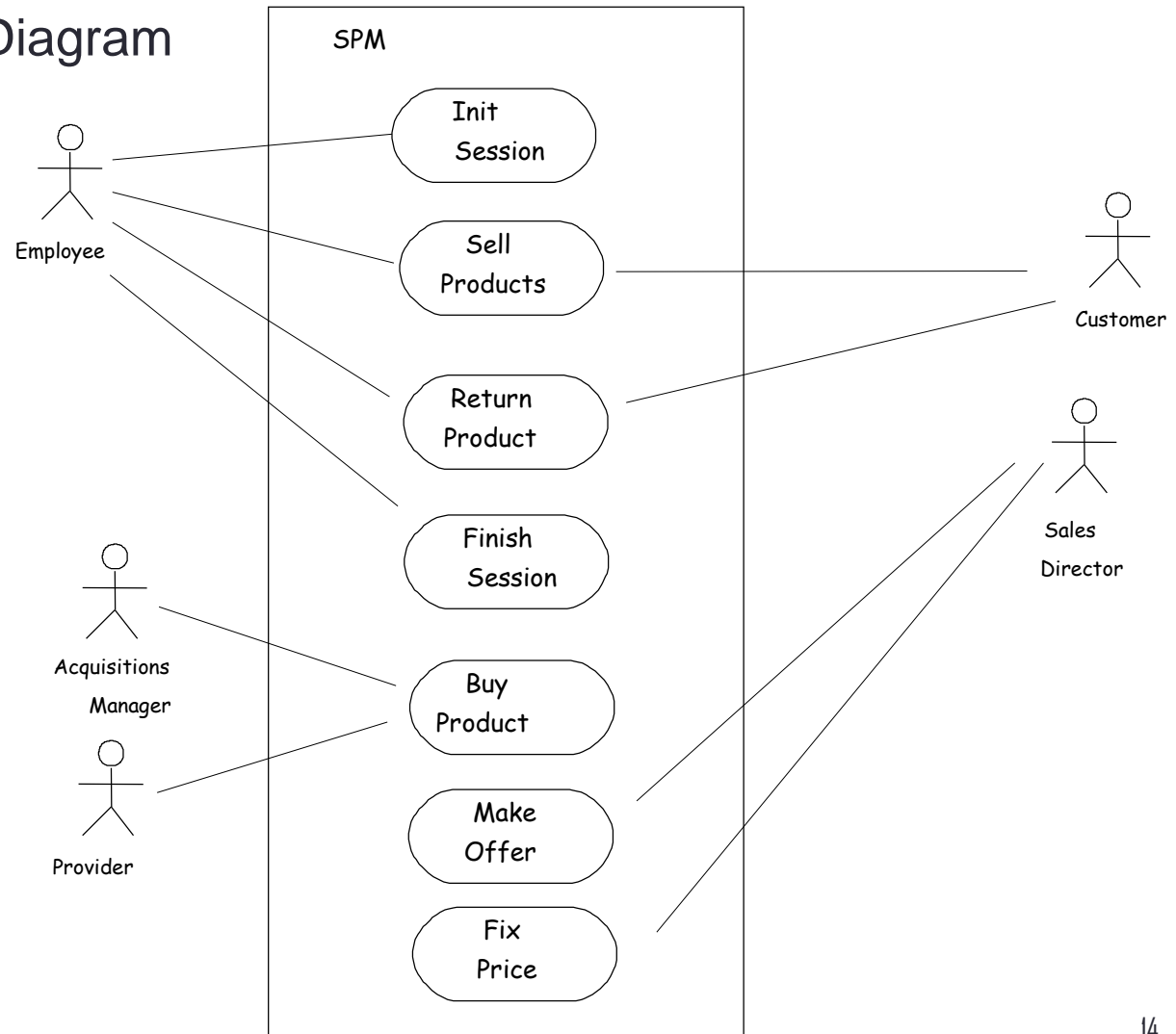


(... the model is incomplete)

# Example SPM

*Sales Point Machine*

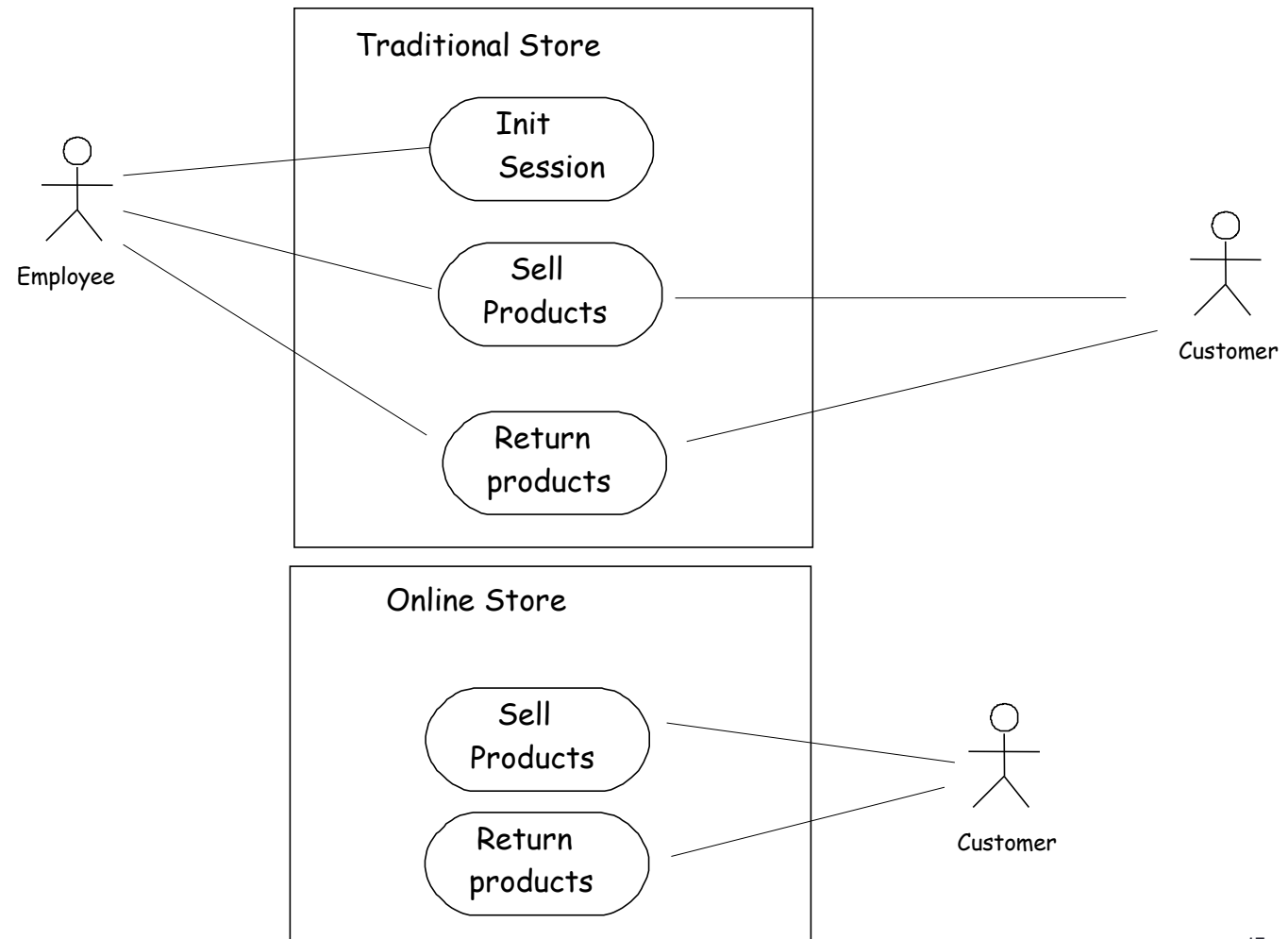
## Context and Initial Diagram



# ... Example SPM

*Sales Point Machine*

Variations

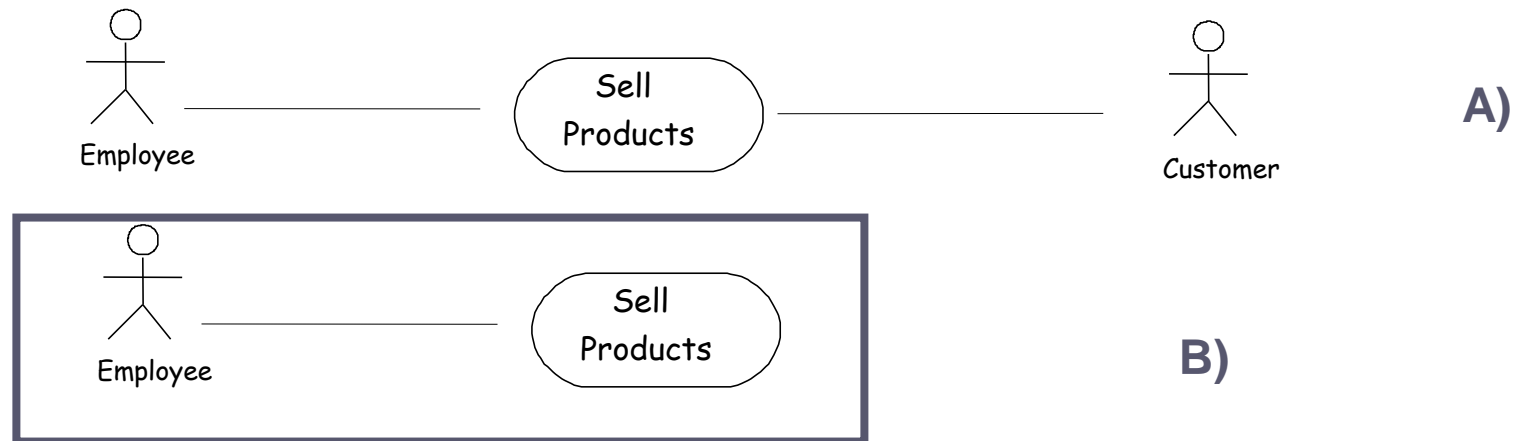


# ... Example SPM

*Sales Point Machine*

Variations on the example

If only actors interacting with the computer-based system are shown.





# ... Example SPM

*Sales Point Machine*

Description  
template



B)

<b>Use Case</b>	Sell Products
<b>Actors</b>	Employee (initiator)
<b>Goal</b>	Capture a sale and its cash payment
<b>Summary</b>	A customer arrives to the sales point with products. The employee registers products and manages payments in cash. The customer leaves with the products.
<b>Preconditions</b>	The employee is logged in the system.
<b>Postconditions</b>	The sale is stored in the system.
<b>Includes</b>	-
<b>Extends</b>	-
<b>Inherits from</b>	-

# ... Example SPM

*Sales Point Machine*

... Description  
template

User Intentions	System Obligations
1. The employee indicates a new sale starts.	2. The system records the start of a new sale
3. The employee inserts the code of the product and the quantity	4. The system calculates the cost of the product and adds the information to the bill.
5. The employee indicates the end of the sale	6. The system calculates and shows the total cost.
7. The employee indicates the money received	8. The system calculates and shows the change. It prints a receipt and records the sale.
<b>Synchronous Extensions</b>	
#1. If at step 3 a code of a non-existent product is inserted the system generates an error message.	
#2. At 7 the employee may cancel the sale.	
<b>Asynchronous Extensions</b>	
None	

# Description Styles



**Get Cash,  
(Concrete Use case)**

User Action	System answer
Insert card	
	Read magnetic band
	Request PIN
InsertPIN	
	Verify PIN
	Show Main Menu
Press Key	
	Show Accounts Menu
Press Key	
	Request Ammount
Insert Ammount	
	Show Ammount
Press Key	
	Return card
Take Card	
	Give Money
Take Money	

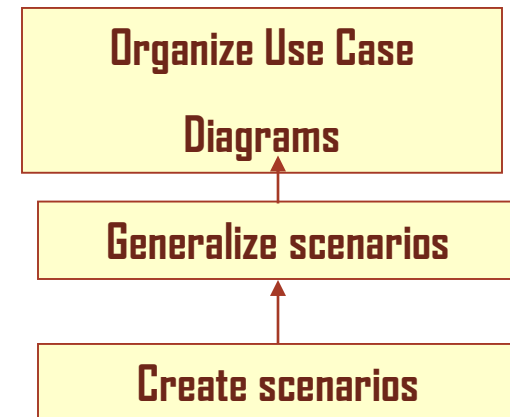
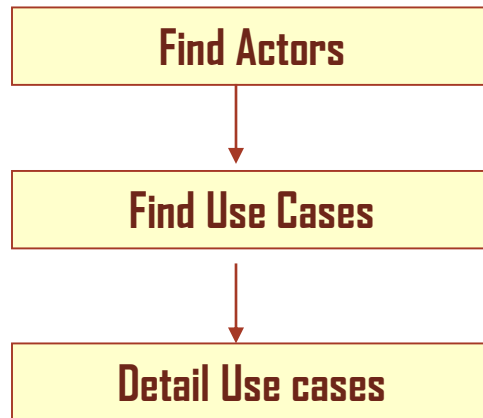
**Get Cash,  
(Essential use case)**

User Intention	System Responsability
Identify	
	Verify identity
	Show Operations
Selection	
	Give Money
Take Money	

# Building the diagram

- Top-down technique
- Bottom-up technique

Top Down



Bottom Up

# Building the diagram

*Rules to find Actors*

- Users play roles when interacting with the system
  - A user may correspond to many roles
- 
- ✌ Any group or individual in one of the following categories:
    - Who will use the system?
    - Who will install the system?
    - Who will maintain the system?
    - Who will switch off the system?
    - What other systems will communicate with this one?
    - Who gets information?
    - Who provides information?

# Building the diagram

## *Rules to identify Use Cases*

- Paying attention to actors
  - What are the tasks required by actors from the system?.
  - Will an actor be able to create, store, change or remove information from the system?.
  - Will an actor inform to the system about changes occurring outside?.
  - Will any actor be informed about state changes of the system?.

The answers to these questions represent flows of events that are associated to potential candidate use cases.