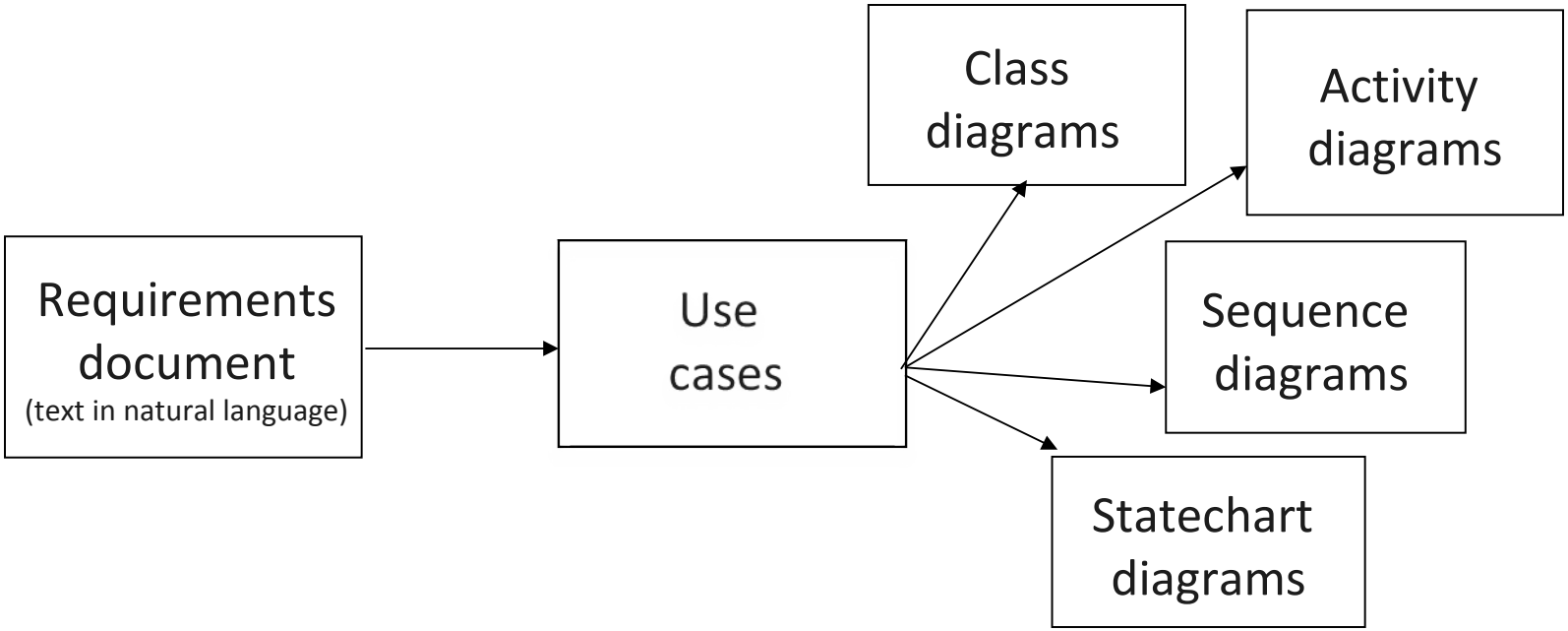
Use Case Diagram

Introduction

Use-cases are descriptions of the functionality of a system from a user perspective.

- Depict the behaviour of the system, as it appears to an outside user.
- Describe the functionality and users (actors) of the system.
- Show the relationships between the actors that use the system, the use cases (functionality) they use, and the relationship between different use cases.
- Document the scope of the system.
- Illustrate the developer's understanding of the user's requirements.

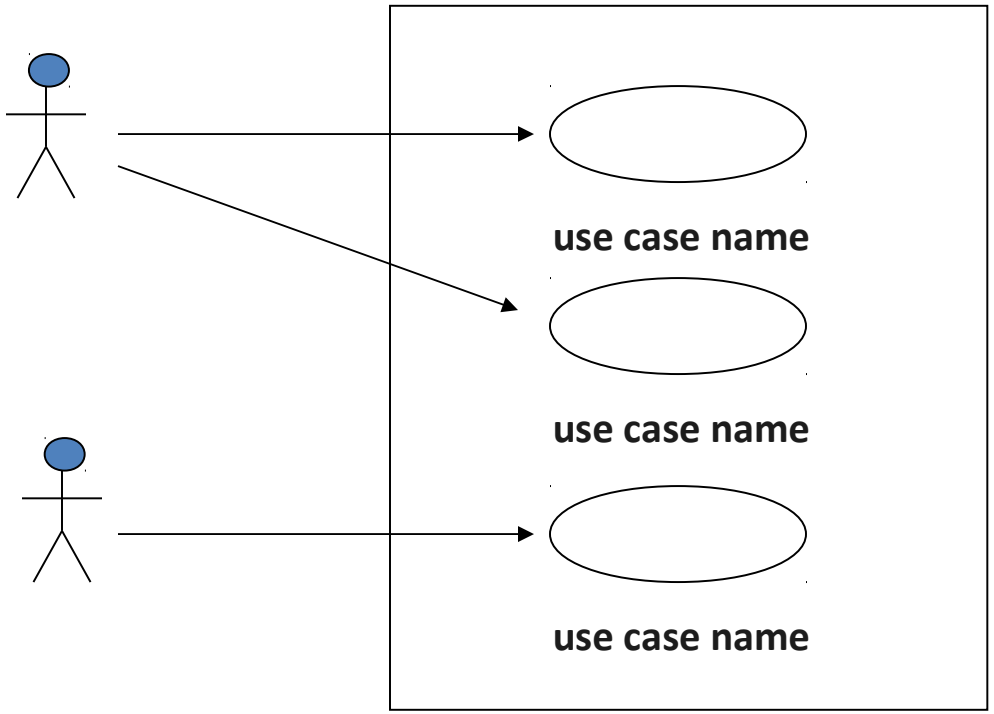
Use Case Diagram, purpose



- Use case models are developed at different levels of abstraction
  - system, system component, or a class.
- Use case modelling is an iterative and incremental process.
  - If user requirements change, the changes should be made in all the affected documents.

Use Case diagrams, basic UML notation

- Use Case: A Use Case is a description of a set of interactions between a user and the system.
- Components of use case diagram:
  - Actor
  - Use case
  - System boundary
  - Relationship

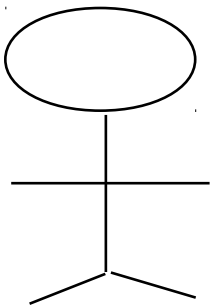


Use cases: Information captured

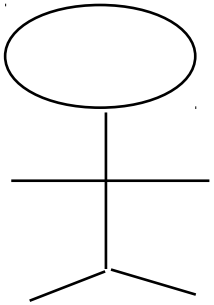
- Actors
- Relationships with other use cases
- Pre-conditions
- Details
- Post-conditions
- Exceptions
- Constraints
- Alternatives

ACTOR

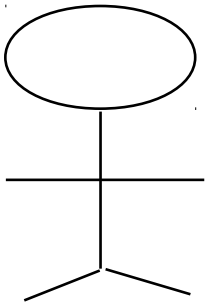
- An actor is some one or something that must interact with the system under development
- Actors can be human or automated systems.
- Actors are not part of the system.
- UML notation for actor is stickman, shown below.



Student



Faculty



Employee

ACTOR (contd..)

- It is role a user plays with respect to system.
- Actors carry out use cases and a single actor may perform more than one use cases.
- Actors are determined by observing the direct uses of the system

Primary and Secondary Actors

- Primary Actor
  - Acts on the system
  - Initiates an interaction with the system
  - Uses the system to fulfill his/her goal
  - Events Something we don't have control over
- Secondary Actor
  - Is acted on/invoked/used by the system
  - Helps the system to fulfills its goal
  - Something the system uses to get its job done

USE CASE

What is USE case?

- A use case is a pattern of behavior, the system exhibits
- The use cases are sequence of actions that the user takes on a system to get particular target
- USE CASE is dialogue between an actor and the system.

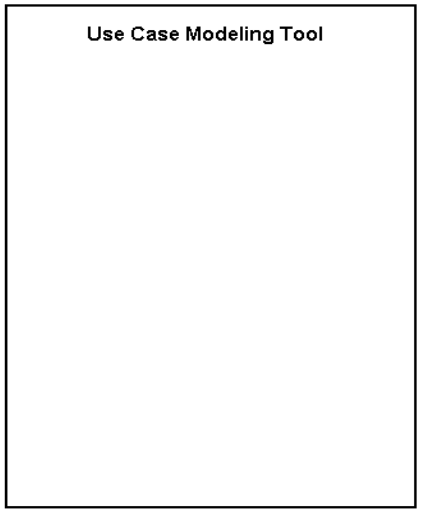
- Examples:

Contd..

- A use case typically represents a major piece of functionality that is complete from beginning to end.
- Most of the use cases are generated in initial phase, but may add some more after proceeding.
- A use case may be small or large. It captures a broad view of a primary functionality of the system in a manner that can be easily grasped by non technical user.

System Boundary

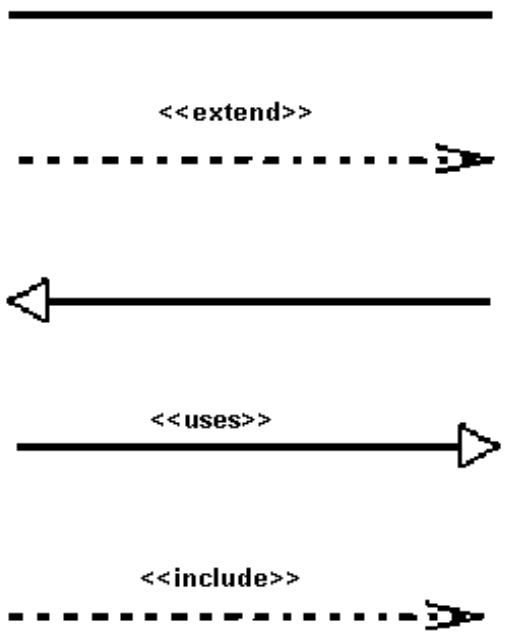
- It is shown as a rectangle.
- It helps to identify what is external versus internal, and what the responsibilities of the system are.
- The external environment is represented only by actors.



Relationship

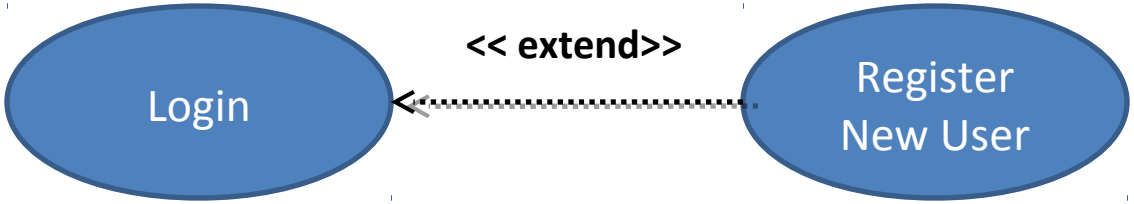
- Relationship is an association between use case and actor.
- There are several Use Case relationships:

  - Association
  - Extend
  - Generalization
  - Uses
  - Include



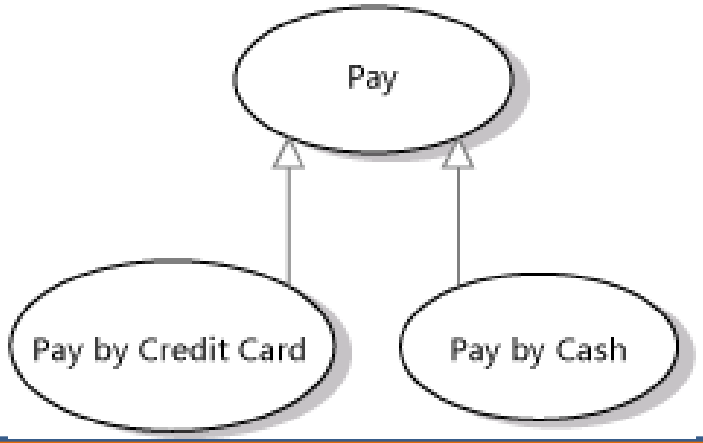
Extend Relationship

- The extended relationship is used to indicate that use case completely consists of the behavior of another use case at one or specific point
- use cases that extend the behavior of other core use cases. Enable to factor variants
- The base use case implicitly incorporates the behavior of another use case at certain points called extension points
- It is shown as a dotted line with an arrow point and labeled <<extend>>



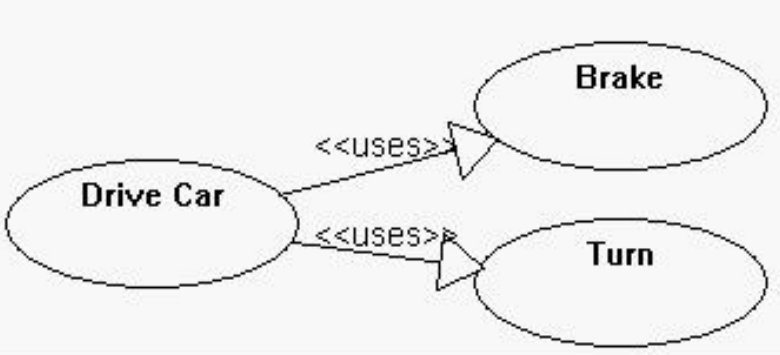
Generalization

- Generalization is a relationship between a general use case and a more specific use case that inherits and extends features to it
- use cases that are specialized versions of other use cases
- It is shown as a solid line with a hollow arrow point



Uses Relationship

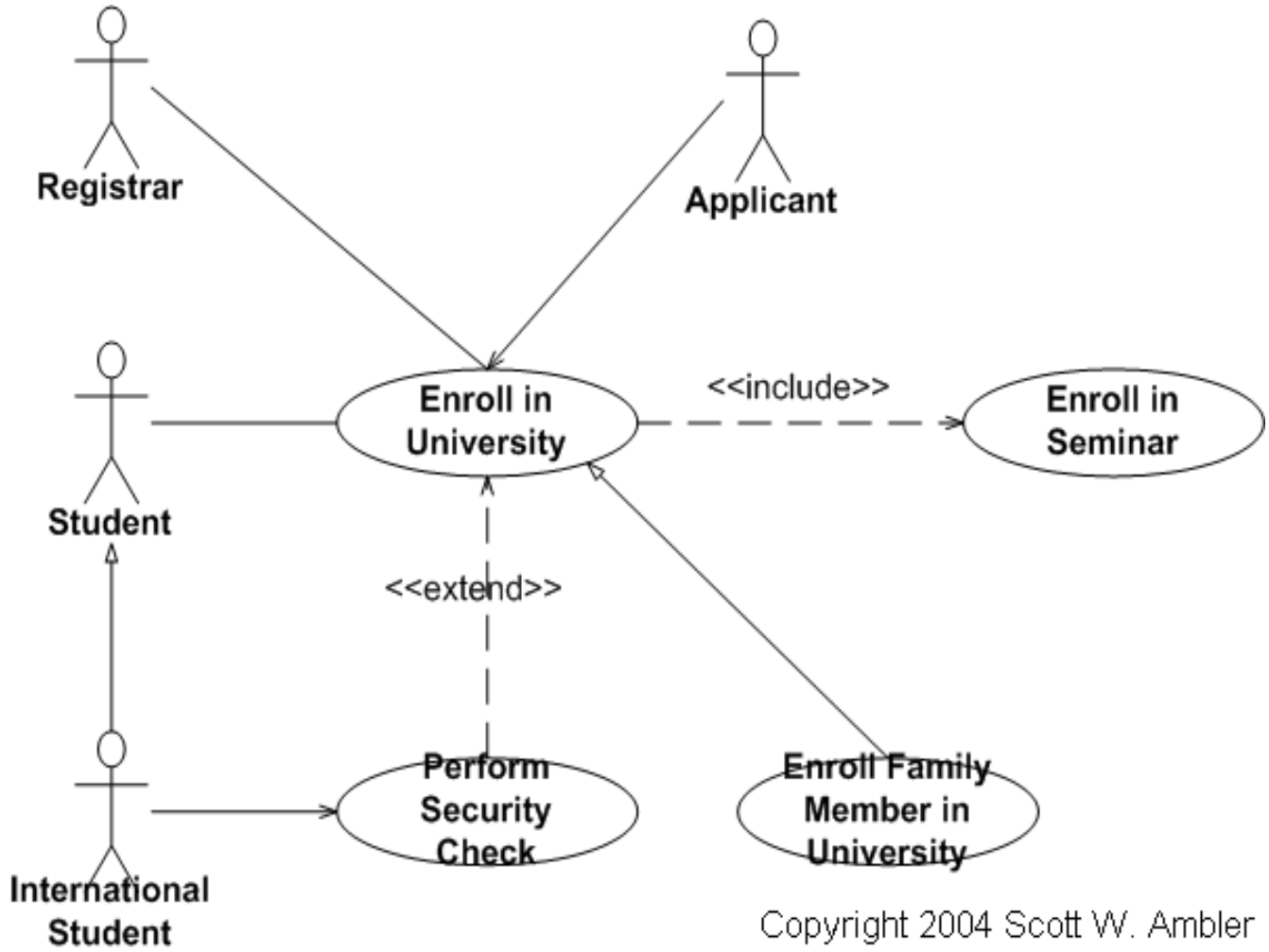
- When a use case uses another process, the relationship can be shown with the uses relationship
- This is shown as a solid line with a hollow arrow point and the <<uses>> keyword



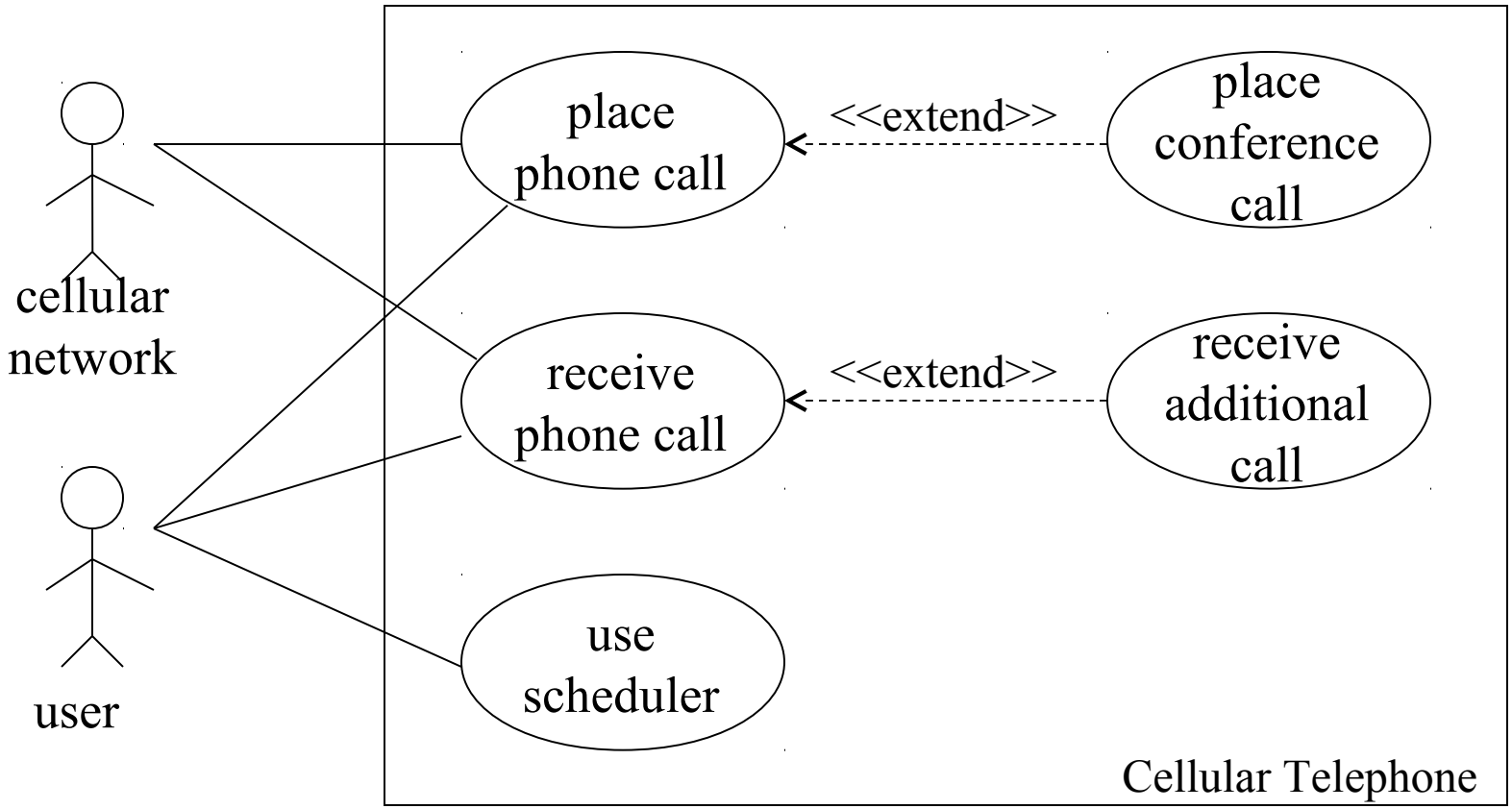
Include Relationship

- Include relationships insert additional behavior into a base use case
- use cases that are included as parts of other use cases. Enable to factor common behavior.
- The base use case explicitly incorporates the behavior of another use case at a location specified in the base.
- They are shown as a dotted line with an open arrow and the key word <<include>>

Extend vs Include



Example


cellular network
user
place phone call
receive phone call
use scheduler
<<extend>>
<<extend>>
place conference call
receive additional call
Cellular Telephone

Use Case Description

:Each use case may include all or part of the following

- Title or Reference Name - meaningful name of the UC
- Author/Date - the author and creation date
- Modification/Date - last modification and its date
- Purpose - specifies the goal to be achieved
- Overview - short description of the processes
- Cross References - requirements references
- Actors - agents participating
- Pre Conditions - must be true to allow execution
- Post Conditions - will be set when completes normally
- Normal flow of events - regular flow of activities
- Alternative flow of events - other flow of activities
- Exceptional flow of events - unusual situations
- Implementation issues - foreseen implementation problems

Example- Money Withdraw

- Use Case: Withdraw Money
- Author: PKD
- Date: 11-09-2013
- Purpose: To withdraw some cash from user's bank account
- Overview: *The use case starts when the customer inserts his card into the system. The system requests the user PIN. The system validates the PIN. If the validation succeeded, the customer can choose the withdraw operation else alternative 1 – validation failure is executed. The customer enters the amount of cash to withdraw. The system checks the amount of cash in the user account, its credit limit. If the withdraw amount in the range between the current amount + credit limit the system dispense the cash and prints a withdraw receipt, else alternative 2 – amount exceeded is executed.*
- Cross References: R1.1, R1.2, R7

- Actors: Customer
- Pre Condition:
  - The ATM must be in a state ready to accept transactions
  - The ATM must have at least some cash on hand that it can dispense
  - The ATM must have enough paper to print a receipt for at least one transaction
- Post Condition:
  - The current amount of cash in the user account is the amount before the withdraw minus the withdraw amount
  - A receipt was printed on the withdraw amount
  - The withdraw transaction was audit in the System log file

Typical course of events

Actor Actions	System Actions
1. Begins when a Customer arrives at ATM	
2. Customer inserts a Credit card into ATM	3. System verifies the customer ID and status
5. Customer chooses "Withdraw" operation	4. System asks for an operation type
7. Customer enters the cash amount	6. System asks for the withdraw amount
	8. System checks if withdraw amount is legal
	9. System dispenses the cash
	10. System deduces the withdraw amount from account
	11. System prints a receipt
13. Customer takes the cash and the receipt	12. System ejects the cash card

- Alternative flow of events:
  - Step 3: Customer authorization failed. Display an error message, cancel the transaction and eject the card.
  - Step 8: Customer has insufficient funds in its account. Display an error message, and go to step 6.
  - Step 8: Customer exceeds its legal amount. Display an error message, and go to step 6.
- Exceptional flow of events:
  - Power failure in the process of the transaction before step 9, cancel the transaction and eject the card

- One method to identify use cases is actor-based:
  - Identify the actors related to a system or organization.
  - For each actor, identify the processes they initiate or participate in.
- A second method to identify use cases is event-based:
  - Identify the external events that a system must respond to.
  - Relate the events to actors and use cases.
- The following questions may be used to help identify the use cases for a system:
  - What are tasks of each actor ?
  - Will any actor create, store, change, remove, or read information in the system ?
  - What use cases will create, store, change, remove, or read this information ?
  - Will any actor need to inform the system about sudden, external changes ?
  - Does any actor need to be informed about certain occurrences in the system ?
  - Can all functional requirements be performed by the use cases ?