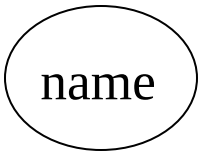
Use Case Diagram

Software Design Methodology

Using Use Case Diagrams

- Use case diagrams are used to visualize, specify, construct, and document the (intended) behavior of the system, during requirements capture and analysis.
- Provide a way for developers, domain experts and end-users to Communicate.
- Serve as basis for testing.
- Use case diagrams contain use cases, actors, and their relationships.

Use Case

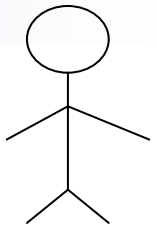


- Use cases specify desired behavior.
- A use case is a description of a set of sequences of actions, including variants, a system performs to yield an observable result of value to an actor.
- Each sequence represent an interaction of actors with the system.

Specifying the Behavior of a Use Case

- Describing the flow of events within the use case.
- Can be done in natural language, formal language or pseudo-code.
- Includes: how and when the use case starts and ends; when the use case interacts with actors and what objects are exchanged; the basic flow and alternative flows of the behavior.

Actors

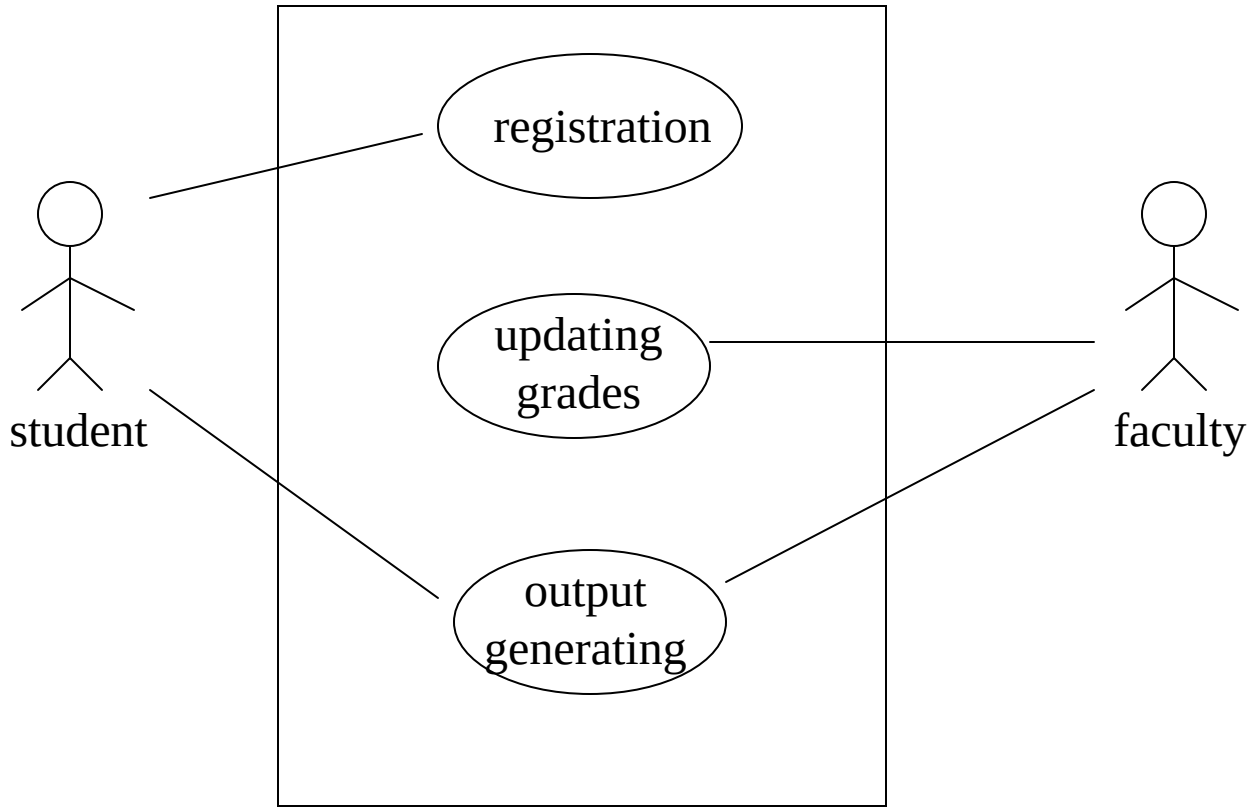


- An actor represents a set of roles that users of use case play when interacting with these use cases.
- Actors can be human or automated systems.
- Actors are entities which require help from the system to perform their task or are needed to execute the system's functions.
- Actors are not part of the system.

Use Cases and Actors

- From the perspective of a given actor, a use case does something that is of value to the actor, such as calculate a result or change the state of an object.
- The Actors define the environments in which the system lives

Example of Use Case Diagram

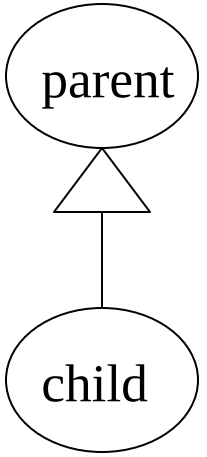


Relationships between Use Cases

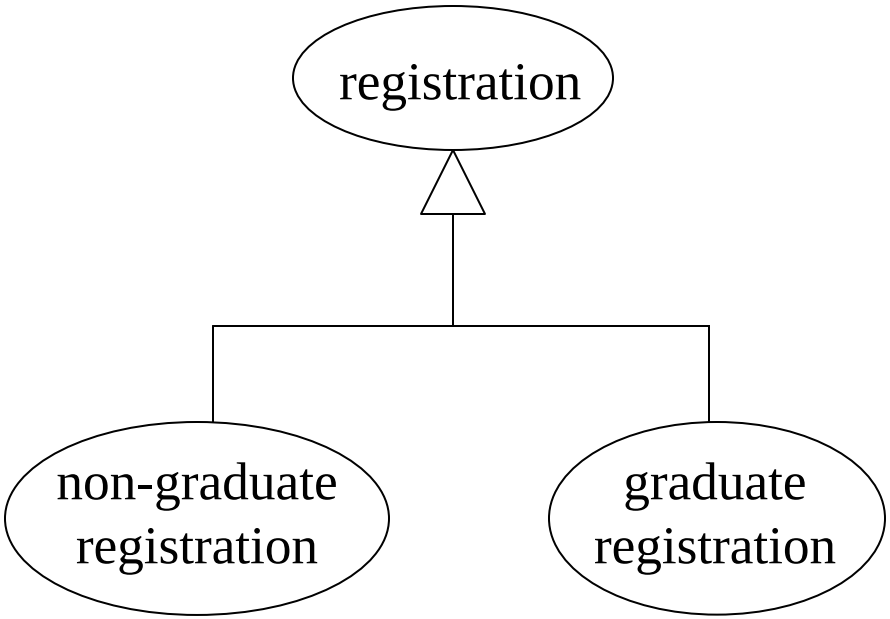
1. Generalization - use cases that are specialized versions of other use cases.
2. Include - use cases that are included as parts of other use cases. Enable to factor common behavior.
3. Extend - use cases that extend the behavior of other core use cases. Enable to factor variants.

1. Generalization

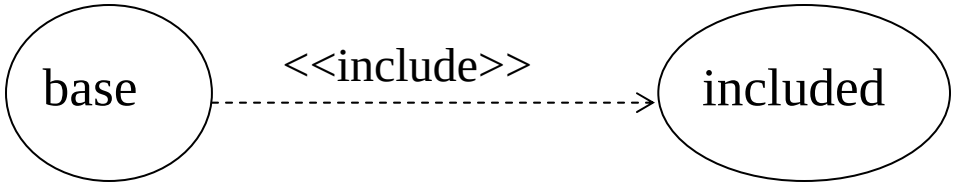
- The child use case inherits the behavior and meaning of the parent use case.
- The child may add to or override the behavior of its parent.



More about Generalization



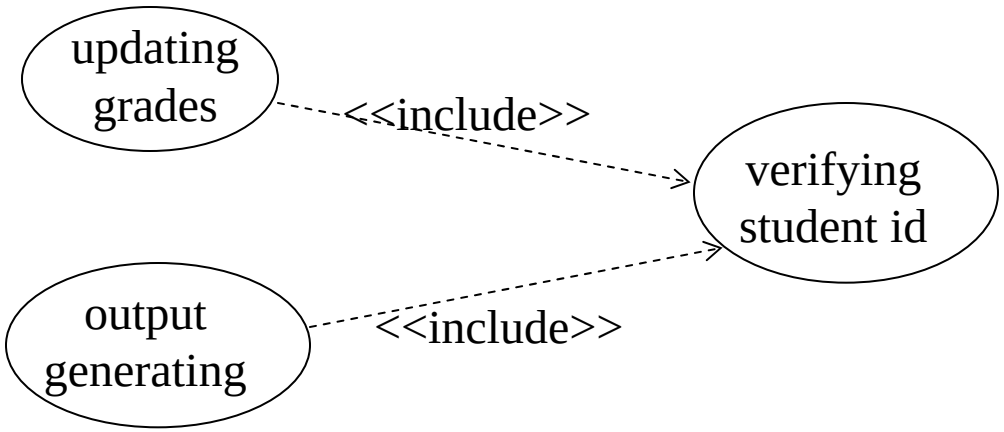
2. Include



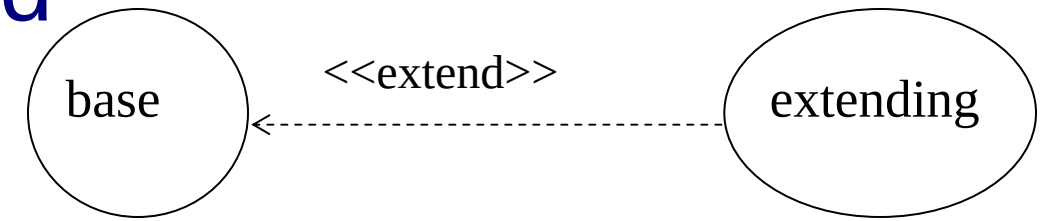
- The base use case explicitly incorporates the behavior of another use case at a location specified in the base.
- The included use case never stands alone. It only occurs as a part of some larger base that includes it.

More about Include

- Enables to avoid describing the same flow of events several times by putting the common behavior in a use case of its own.



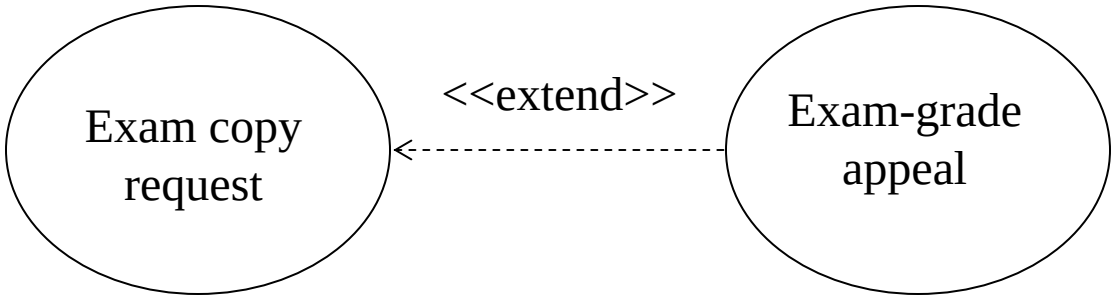
3. Extend



- The base use case implicitly incorporates the behavior of another use case at certain points called extension points.
- The base use case may stand alone, but under certain conditions its behavior may be extended by the behavior of another use case.

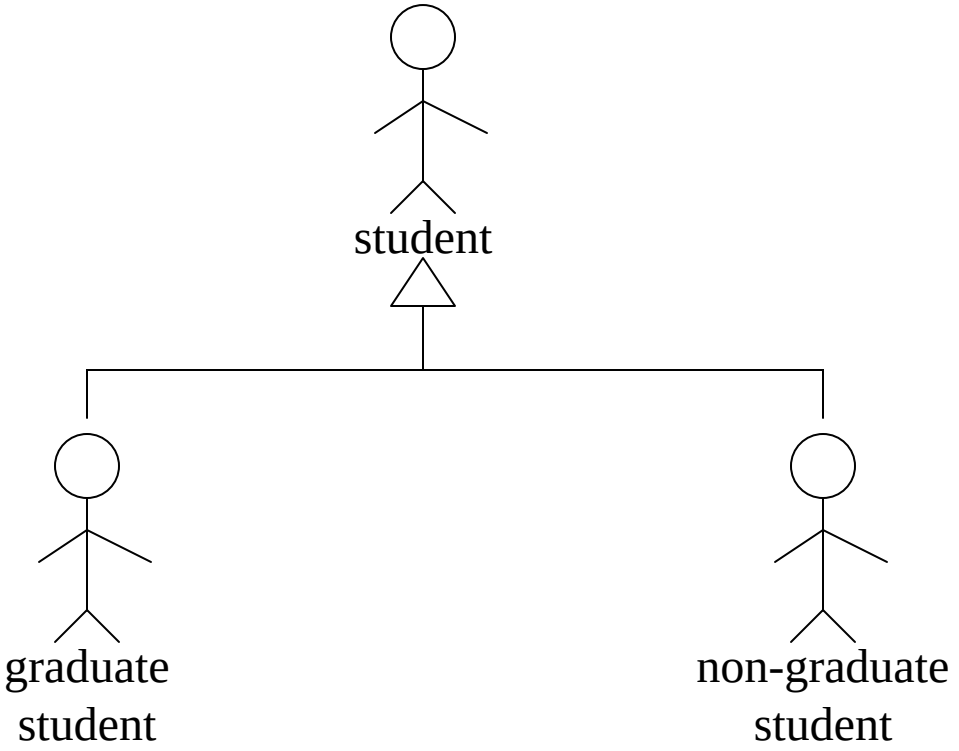
More about Extend

- Enables to model optional behavior or branching under conditions.



Relationships between Actors

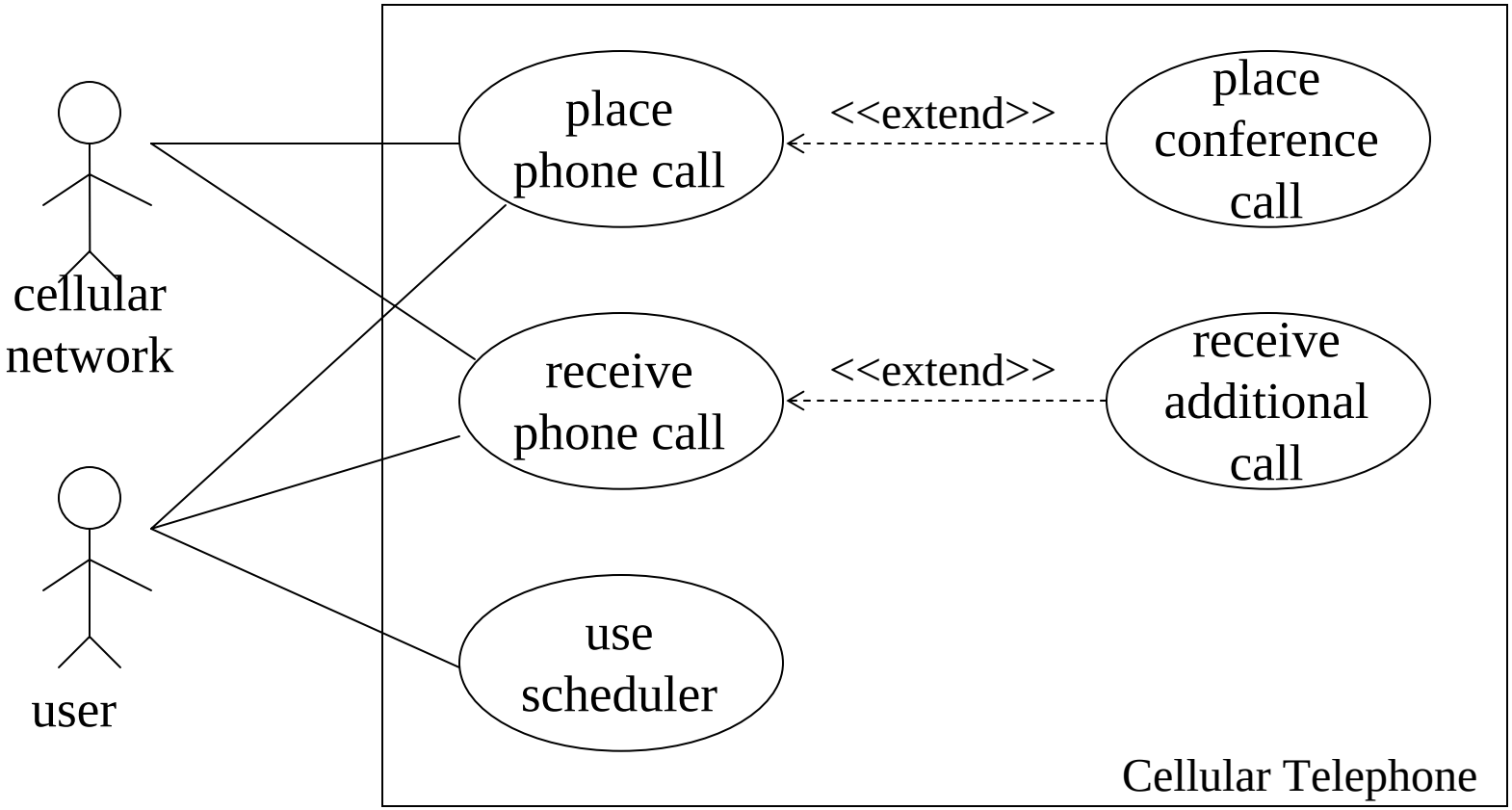
- Generalization.



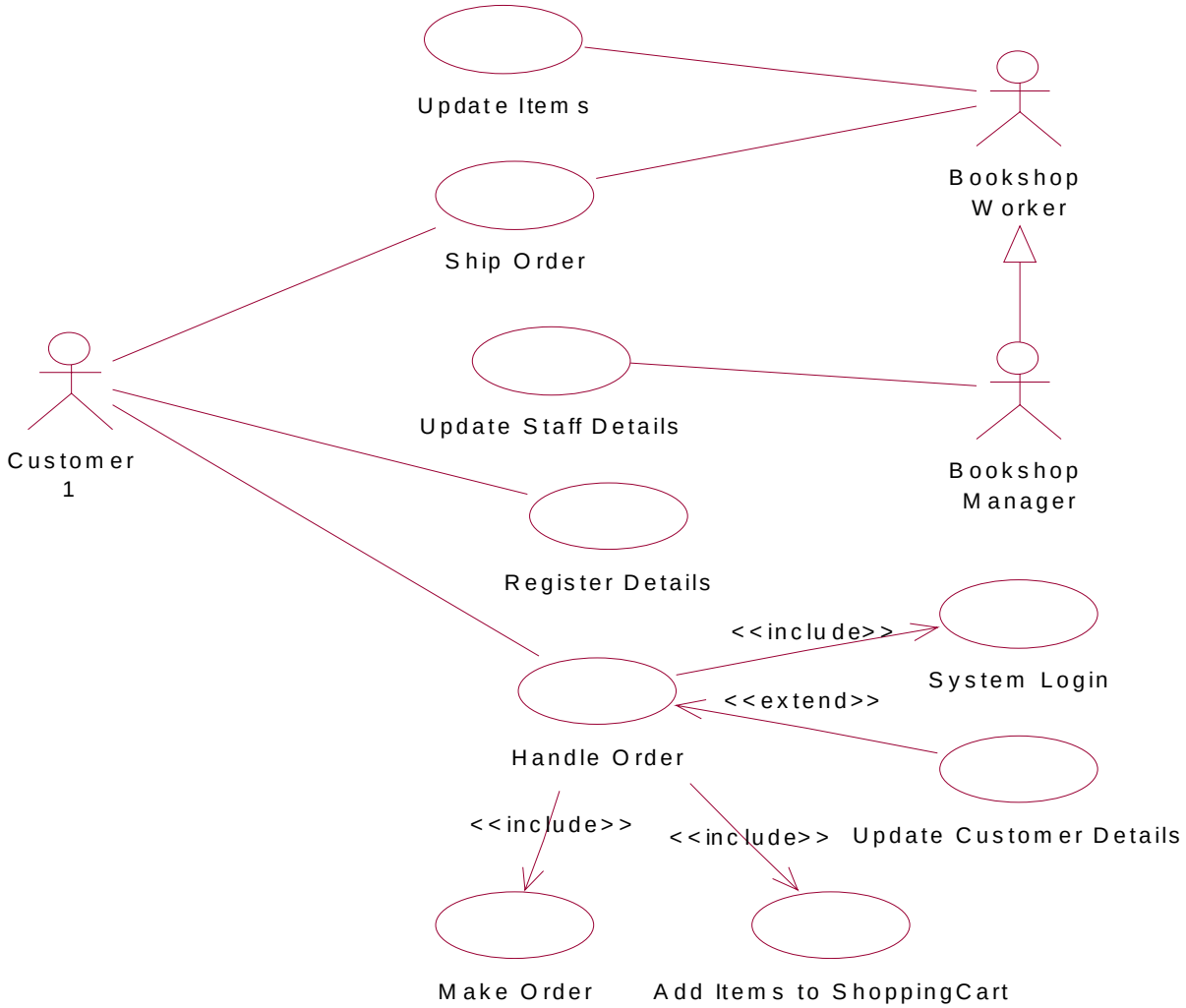
Relationships between Use Cases and Actors

- Actors may be connected to use cases by associations, indicating that the actor and the use case communicate with one another using messages.

Example


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

A More Complicate Example



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: ZB
- Date: 1-OCT-2004
- Purpose: To withdraw some cash from user's bank account
- Overview: The use case starts when the customer inserts his credit 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

Example- Money Withdraw (cont.)

- 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

Example- Money Withdraw (cont.)

- **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

Example- Money Withdraw (cont.)

- 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

Example- Money Withdraw (cont.)

- 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 ?

Moving on

- The “things” that “live” inside the system are responsible for carrying out the behavior the actors on the outside expect the system to provide.
- To implement a use case, we create a society of classes that work together to carry out the behavior of the use case.