Use Case Diagrams

Outline

1. The Use Case Diagram Concept
2. Use Case
3. Actor
4. Other Elements
5. Relationships between Use Cases
6. Relationships between Actors
7. Relationships between Use Cases and Actors
8. Examples
9. User Case Description
10. Creating use case diagrams
11. Challenges

1. The User Case Diagram Concept

- Use case diagram is a model of system functionality. It is used to visualize, specify, construct, and document the (intended) behavior of the system, during requirements capture and analysis.
- Think of main functions a system performs for users – “cases” of using a system
- Provide a way for developers, domain experts and end-users to Communica
- Serve as basis for testin
- Use case diagrams conta
  use cases, actors,
  and their relationships

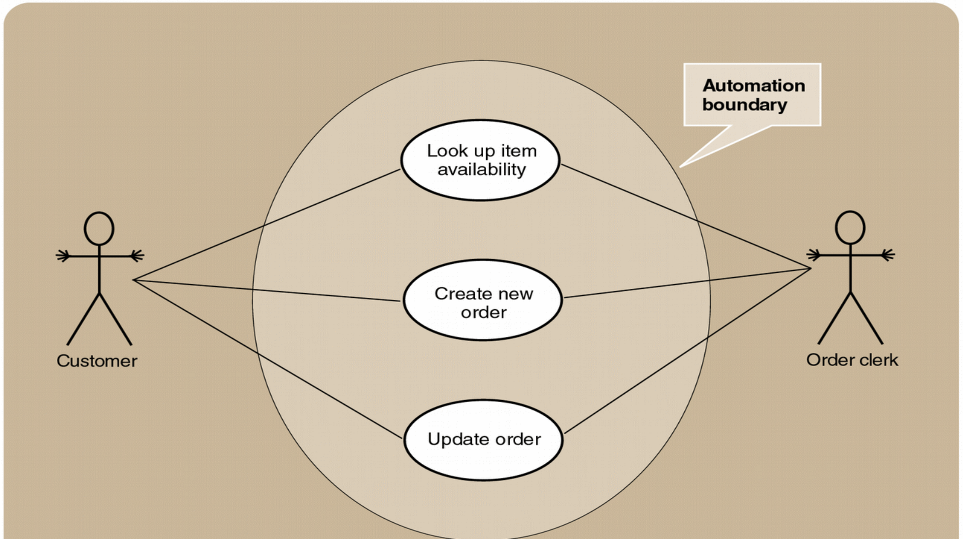
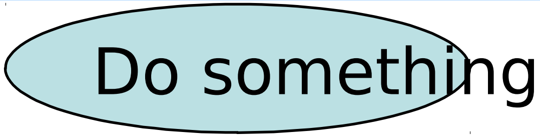
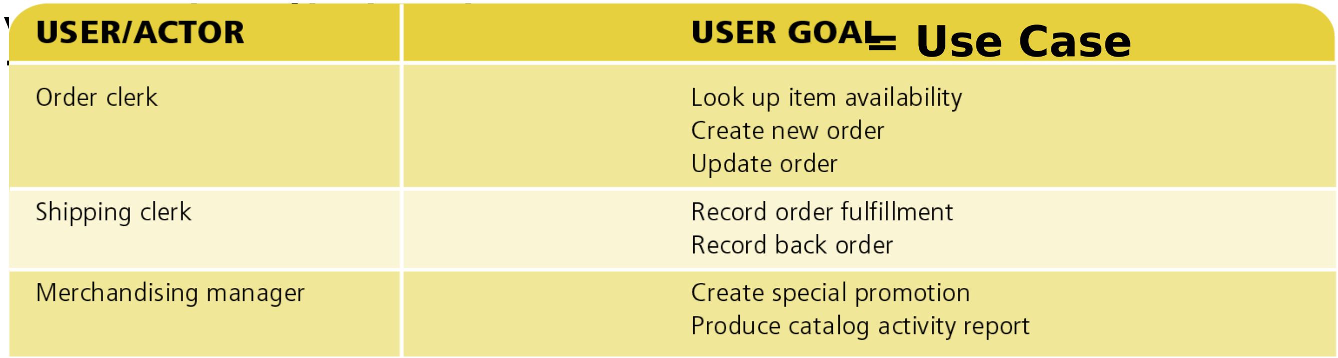


Figure 6-3

2. Use Case



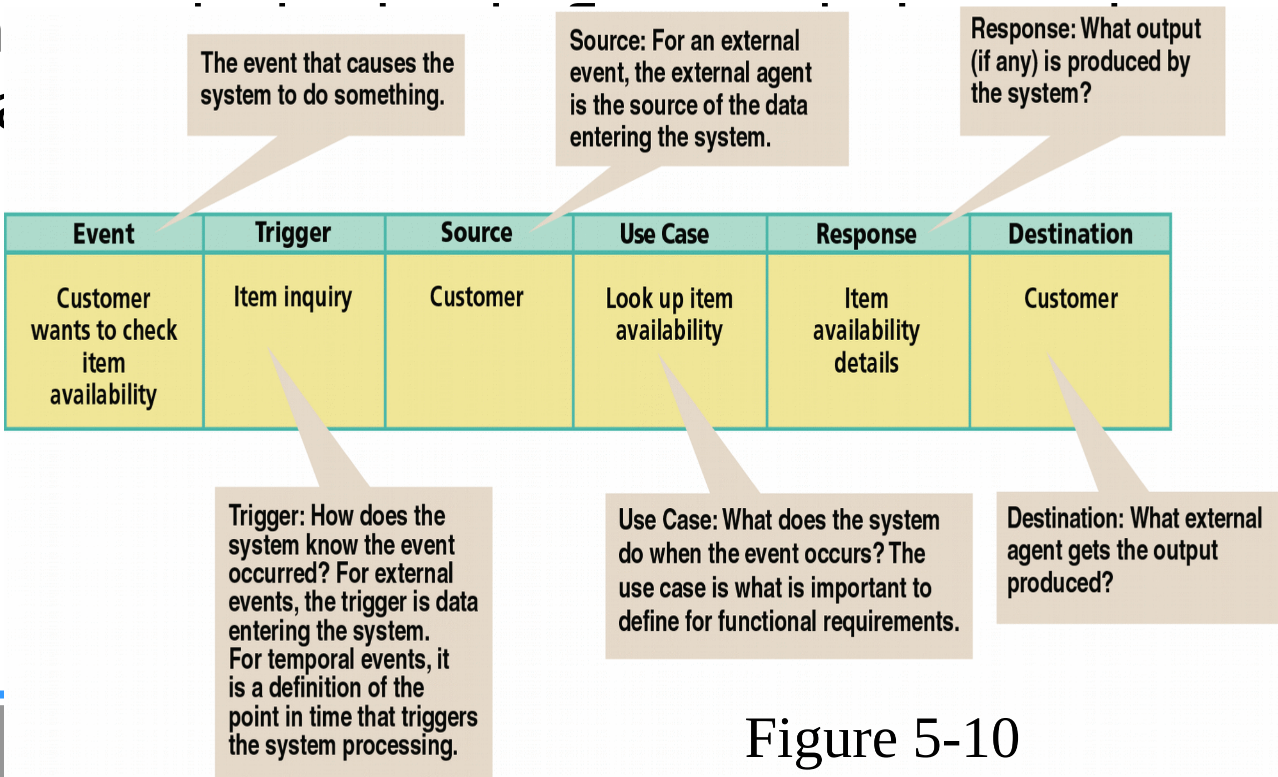
- A system function (process – automated or manual) specifies desired behavior. **Verb**
- A use case is a description of a set of sequences of actions, including variants, a system performs to yield **an observable result** of value to an actor
- Each sequence represent an interaction of actors with the system
- Each Actor must be linked to a use case, while some use cases may



USER/ACTOR	USER GOAL = Use Case
Order clerk	Look up item availability Create new order Update order
Shipping clerk	Record order fulfillment Record back order
Merchandising manager	Create special promotion Produce catalog activity report

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 exch
flows of the beh



Event	Trigger	Source	Use Case	Response	Destination
Customer wants to check item availability	Item inquiry	Customer	Look up item availability	Item availability details	Customer

The event that causes the system to do something.

Source: For an external event, the external agent is the source of the data entering the system.

Response: What output (if any) is produced by the system?

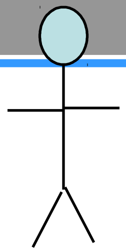
Trigger: How does the system know the event occurred? For external events, the trigger is data entering the system. For temporal events, it is a definition of the point in time that triggers the system processing.

Use Case: What does the system do when the event occurs? The use case is what is important to define for functional requirements.

Destination: What external agent gets the output produced?

Figure 5-10

3. Actor

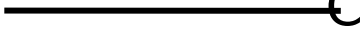


- *Actor* is someone or automated systems interacting with use case. Actor *triggers* use case. **Noun**
- Similar to the concept of user, a user can play different *roles*; (example: a prof. can be instructor and researcher – play 2 roles).
- Actors are entities which require help from the system to perform their task or are needed to execute the system's functions
- Actor has responsibility toward the system (inputs), and Actor have expectations from the system (outputs)
- Actors are not part of the system

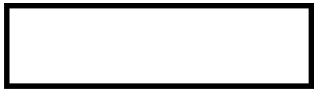
Use Cases and Actors

- 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

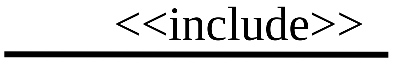
4. Other Elements



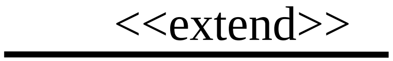
Connection between Actor and Use Case



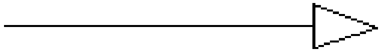
Boundary of system; Automation Boundary



Include relationship between Use Cases (one UC must call another; e.g., Login UC includes User Authentication UC)



Extend relationship between Use Cases (one UC calls Another under certain condition; think of if-then decision points)

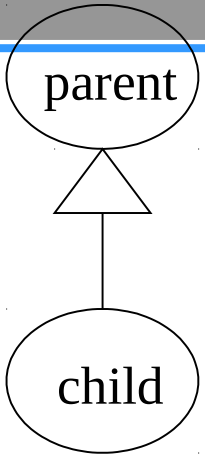


Generalization relationship between Use Cases or between Actors

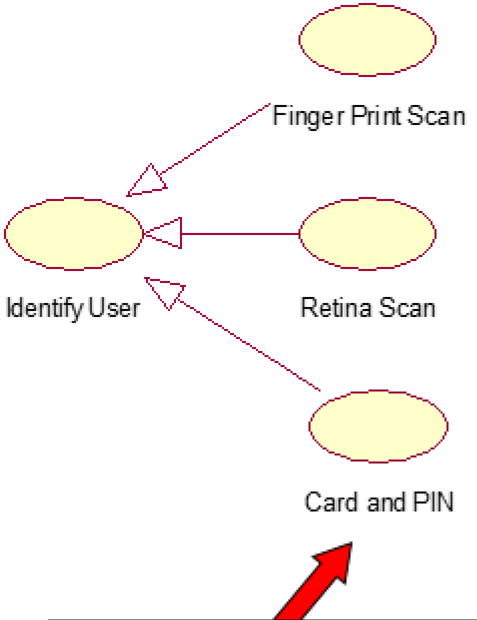
5. Relationships between Use Cases

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**

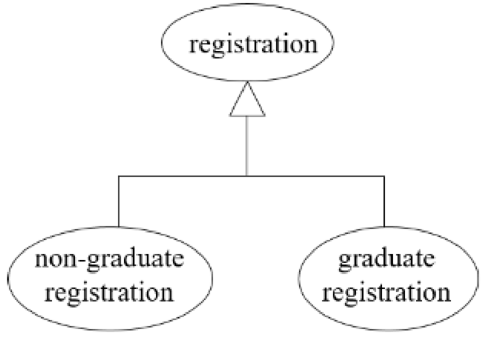
5. 1. Generalization



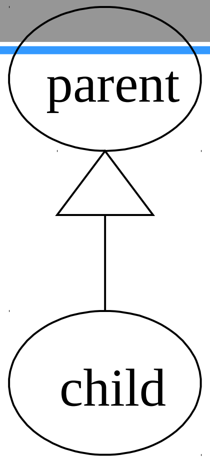
- This occurs when two of more use-cases attempt to achieve the same goal or objective but achieve it via different means
- The child use case inherits the behavior and meaning of the parent use case, and the child may add to or override the behavior of its parent
  - For example, suppose an ID card with a magnetic strip were not the only way that you could identify yourself to a ATM
  - Let's suppose that fingerp scan are also acceptable



'Finger Print Scan', 'Retina Scan' and 'Card and Pin' are all 'kinds of' use case that identify the user

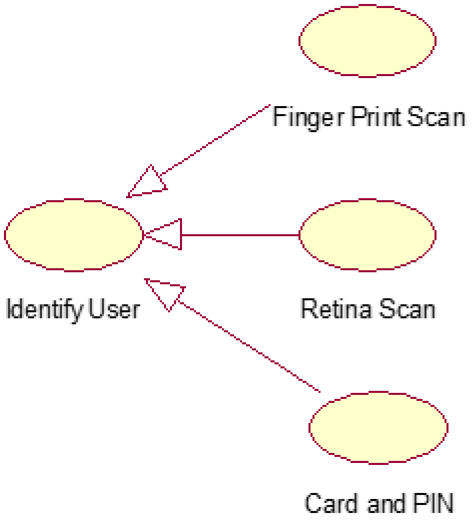


5. 1. Generalization



- When documenting generalisation use cases the *base* or *root* use-case (i.e. Identify User) should be documented in very general terms, i.e. it should just list the objectives of the use-case. For inst suf

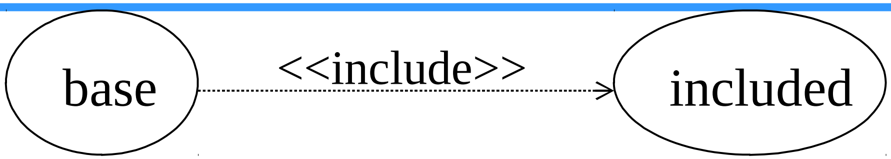
“*Identify the user*”
obtain their account details from the bank central computer



‘Finger Print Scan’, ‘Retina Scan’ and ‘Card and Pin’ are all ‘kinds of’ use case that identify the user

- The three specific or *derived* use-cases can be documented with specific details
- Generalisation then is about isolating common user objectives and expressing that commonality in the base use-case
- The various specific details of achieving that can be

5.2. Include



- The base use case explicitly incorporates the behavior of another use case at a location specified in the base (a base case linked to an **mandatory** use case)
- The included use case never stands alone. It only occurs **as a part** of some larger base that includes it. In other hand, base use case can NOT execute without the included case tight coupling
- Enables to avoid describing the same flow of events several times by putting the common behavior in a use case of its own
  - Example: to *Authorize Car Loan* (base use case), a clerk must run *Check Client's Credit History* (included use case)
  - The standard UC include**s** the mandatory UC (use the verb to

5.2. Include

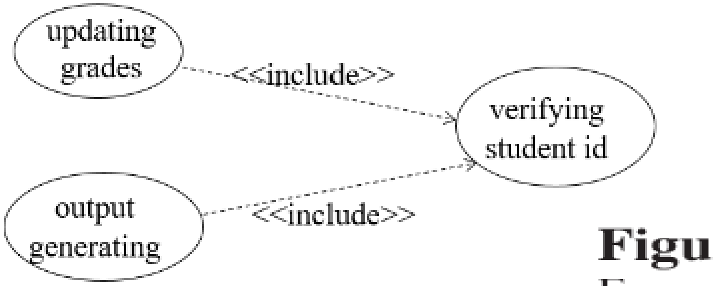
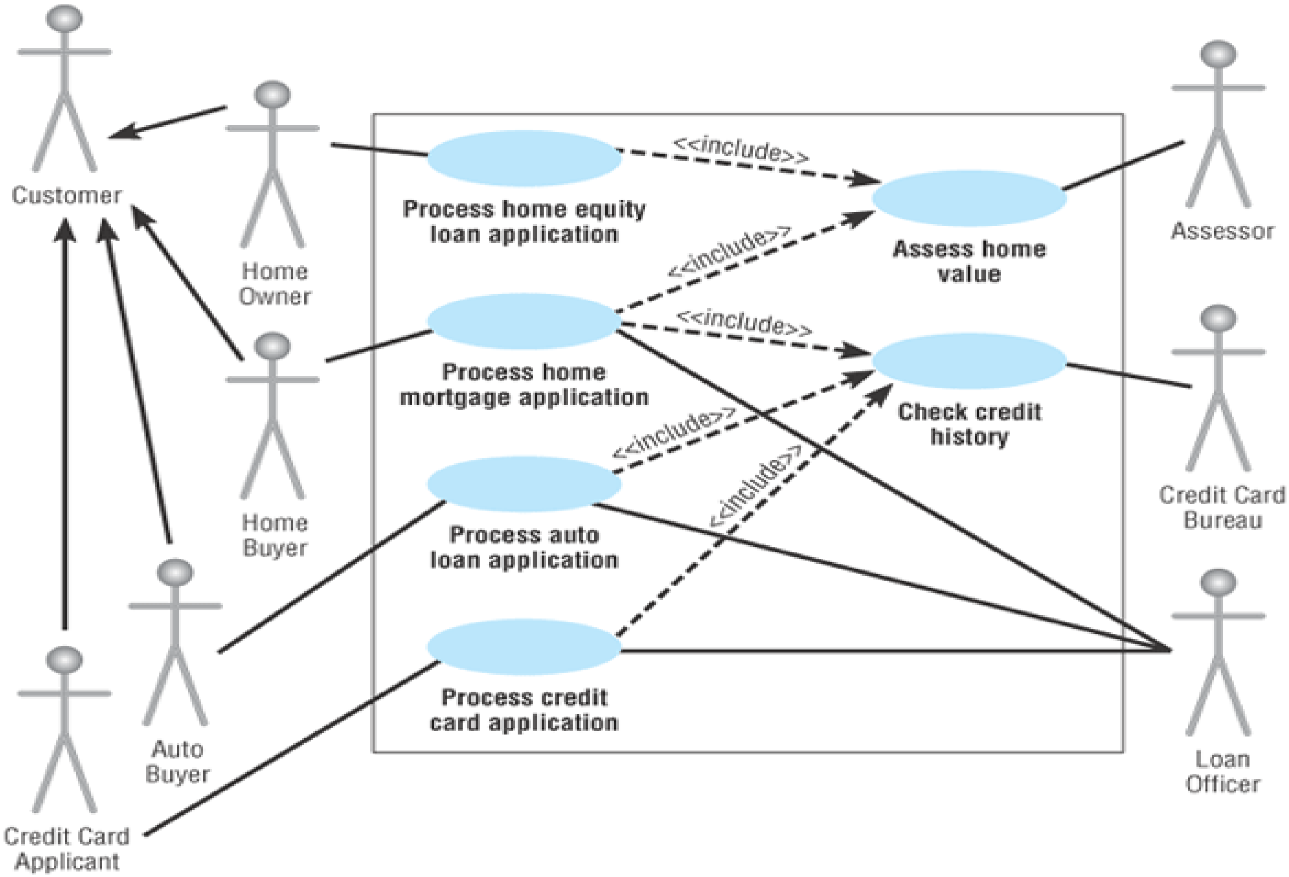
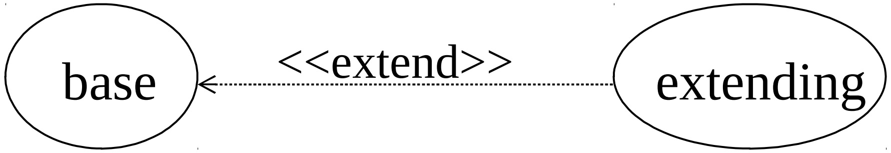


Figure 6.6

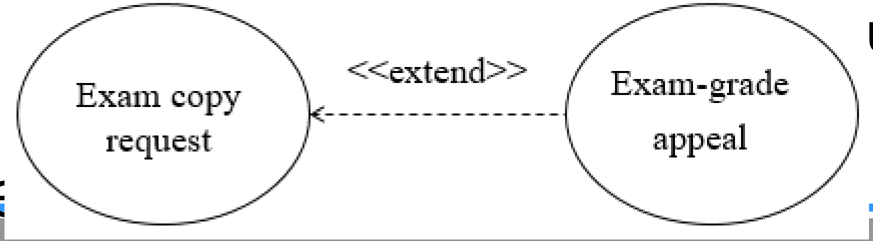
Examples of Generalized Use Cases and Include Relationships



5.3. Extend

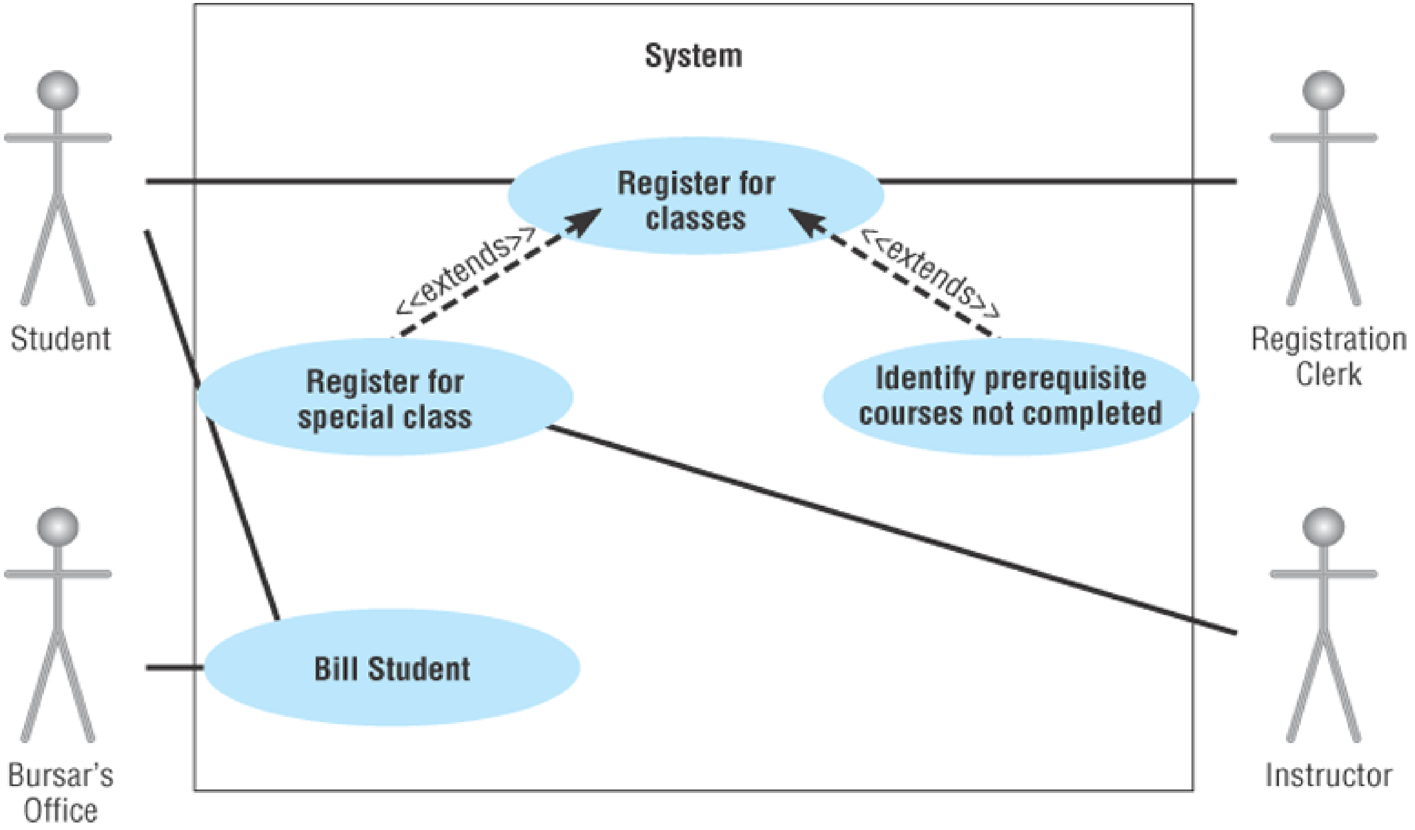


- The base use case implicitly incorporates the behavior of another use case at certain points called extension points.
- The base use case may stand alone, but under certain conditions its behavior may be extended by the behavior of another use case.
- Linking an optional (extending) use case to a base use case
  - Example: *Register Course* (standard use case) may have *Register for Special Class* (extending use case). The optional UC extend**s** ure direction arrow)



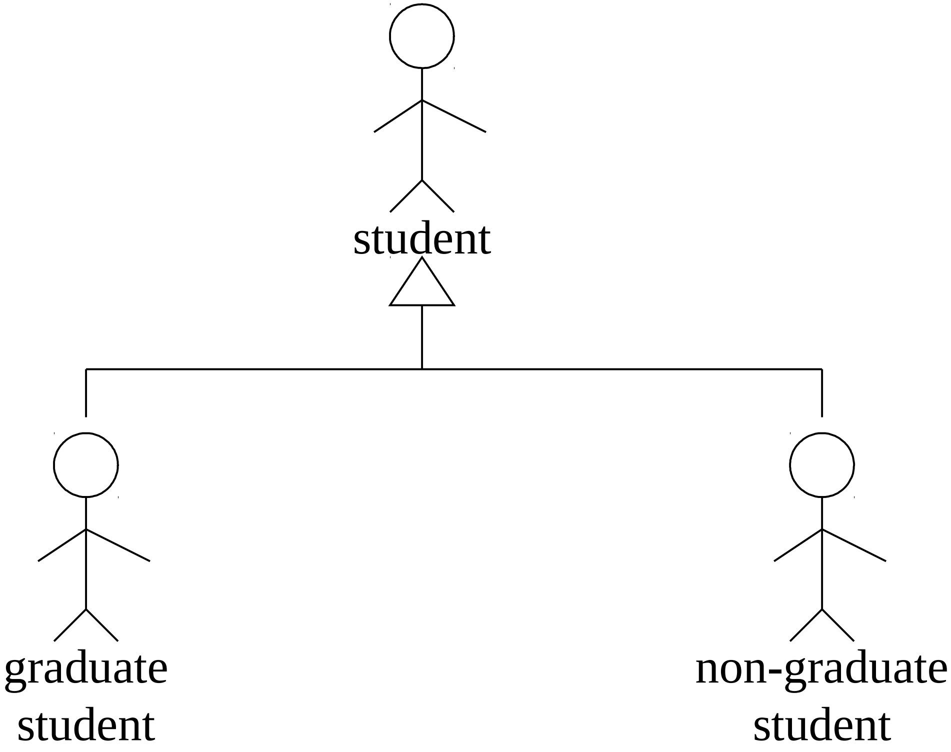
- Standard use the extend case □□ **loose coupling**

Figure 6.4 A Use Case Diagram for a University Registration System Represented with Microsoft's Visio



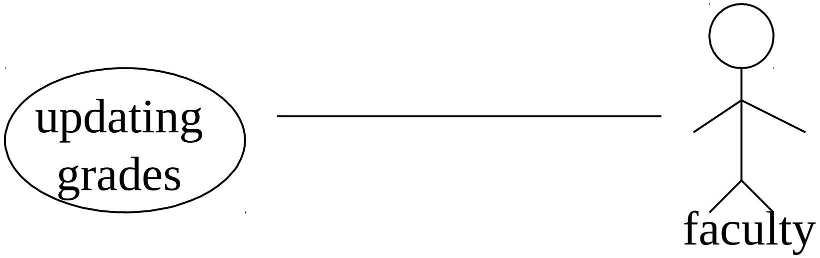
6. Relationships between Actors

- Generalization

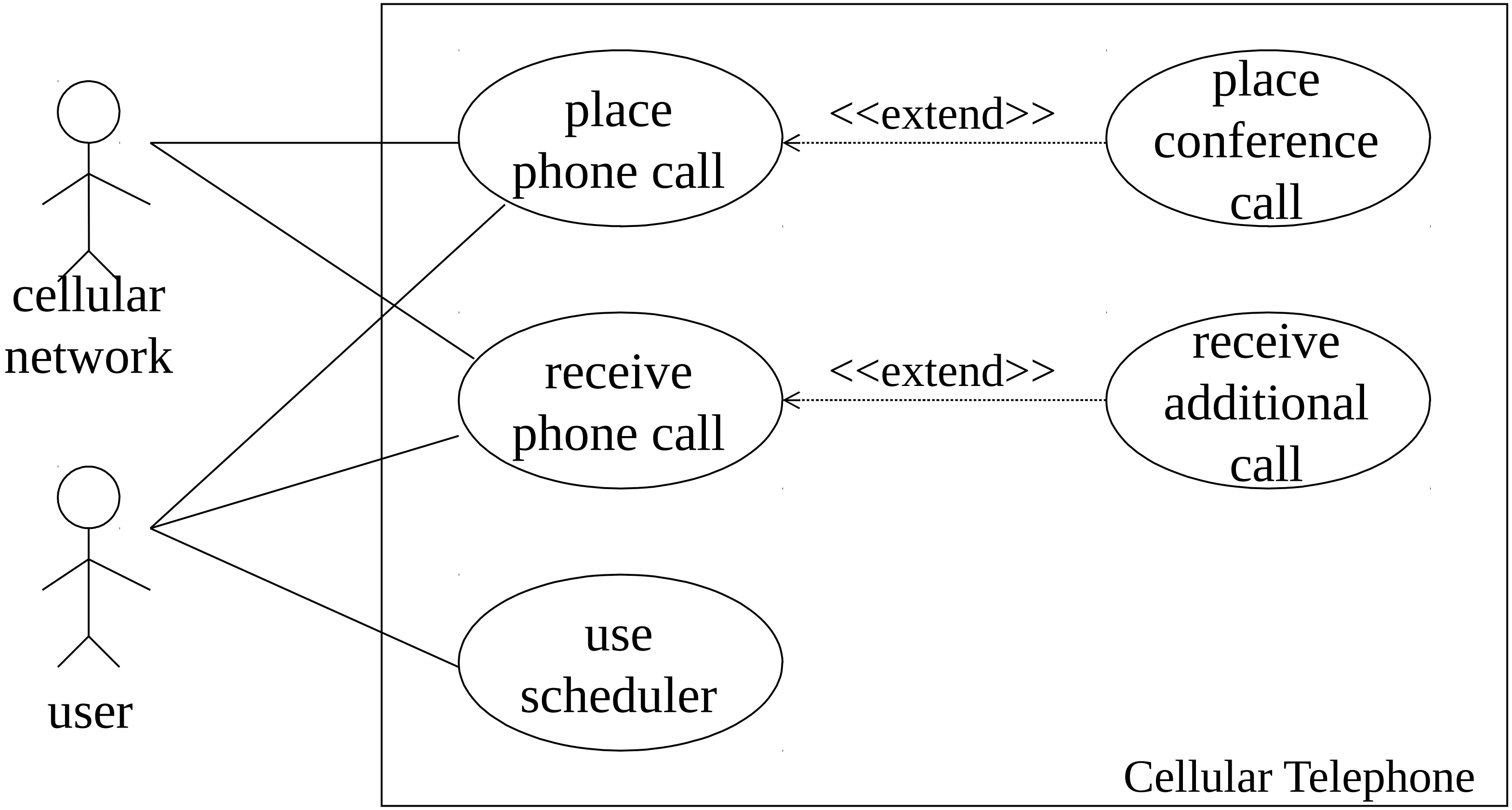


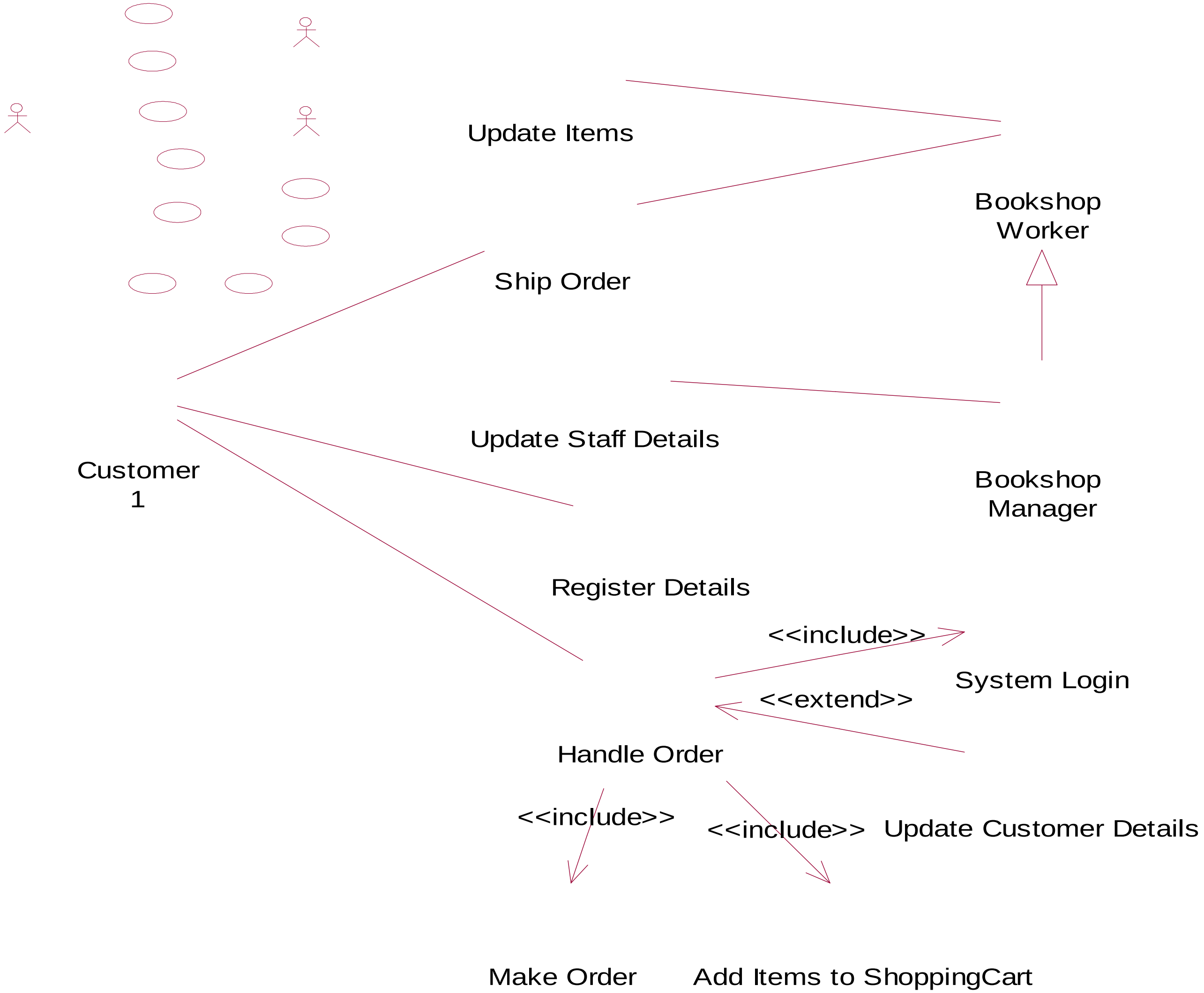
7. Relationships between Use Cases and Actors

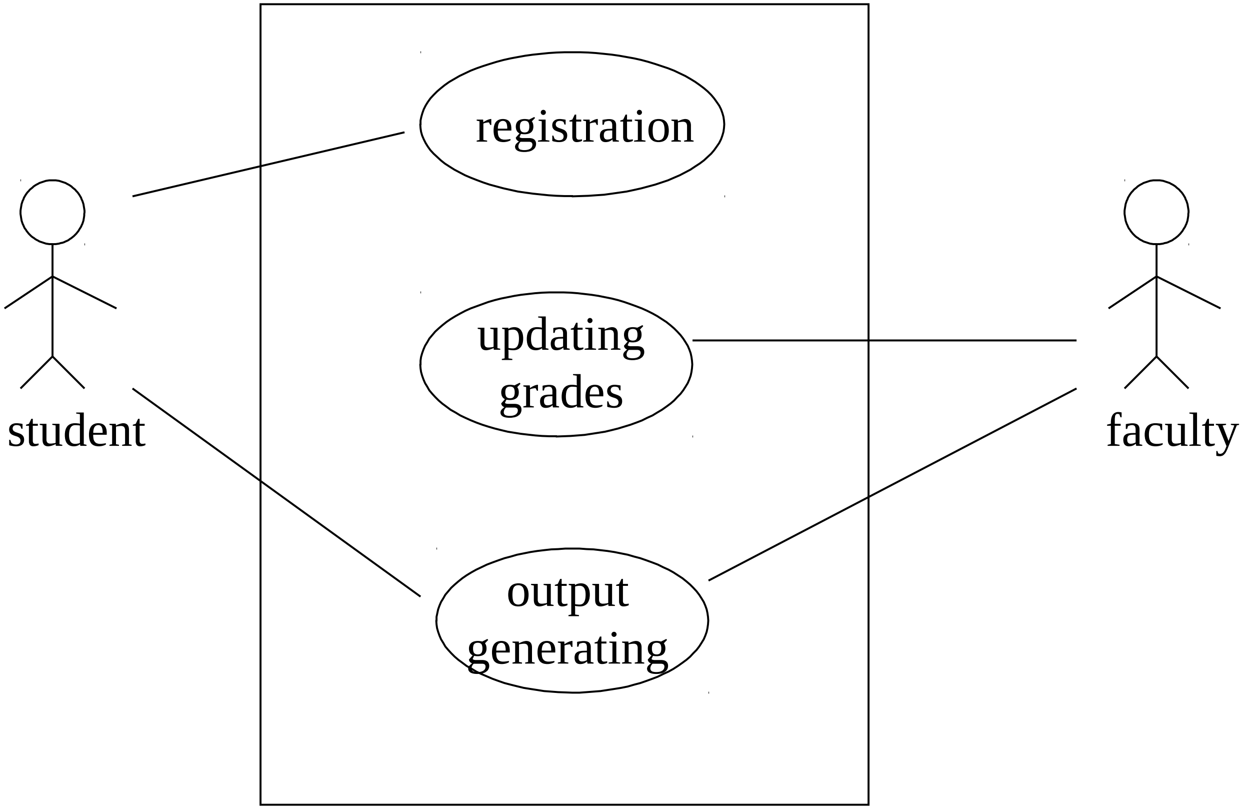
- Actors may be connected to use cases by associations, indicating that the actor and the use case communicate with one another using messages



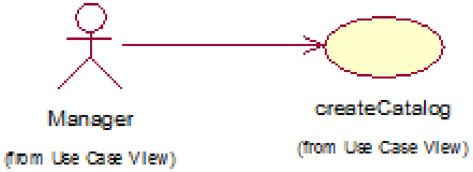
8. Examples

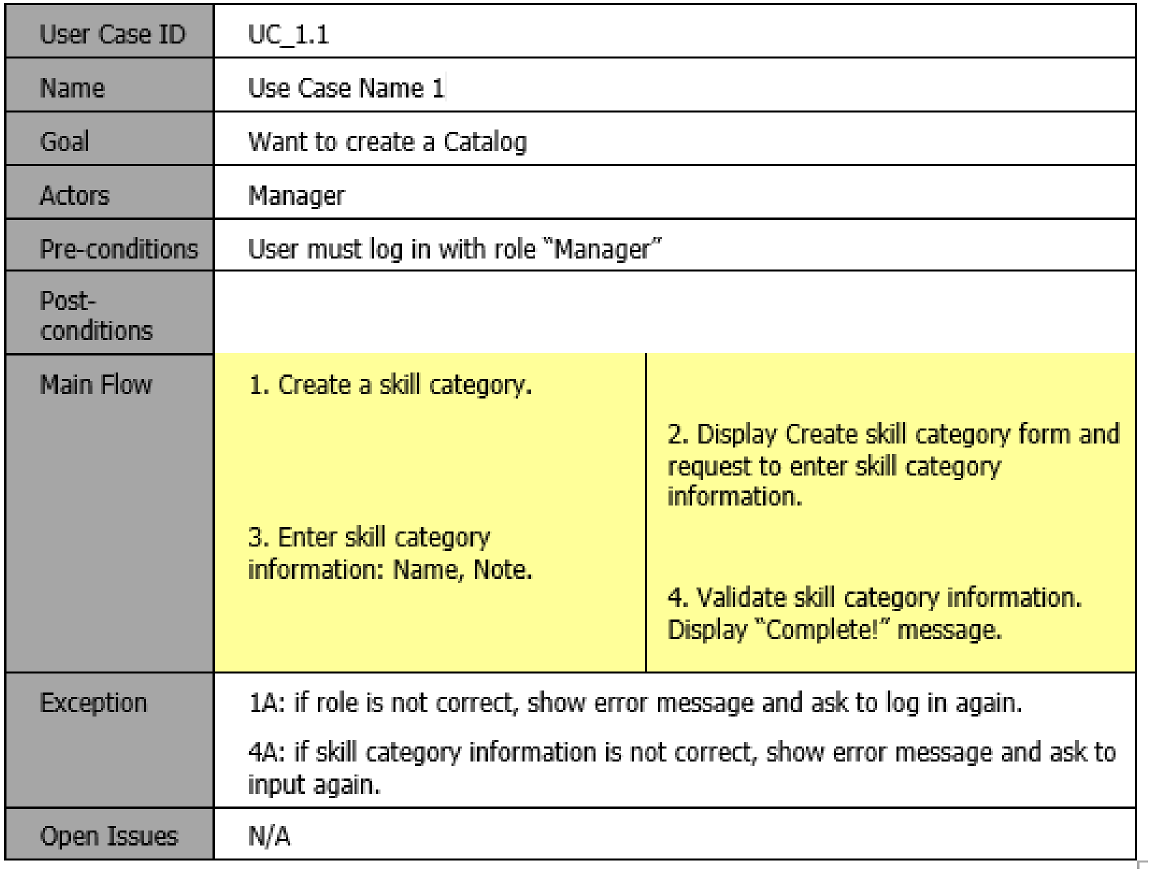

place phone call
<<extend>>
place conference call
cellular network
receive phone call
<<extend>>
receive additional call
user
use scheduler
Cellular Telephone


Update Items
Bookshop Worker
Ship Order
Update Staff Details
Customer 1
Bookshop Manager
Register Details
<<include>>
System Login
<<extend>>
Handle Order
<<include>>
<<include>>
Update Customer Details
Make Order
Add Items to ShoppingCart


registration
updating grades
student
faculty
output generating

9. Use Case Description





User Case ID	UC_1.1	
Name	Use Case Name 1	
Goal	Want to create a Catalog	
Actors	Manager	
Pre-conditions	User must log in with role "Manager"	
Post-conditions		
Main Flow	1. Create a skill category.	
		2. Display Create skill category form and request to enter skill category information.
	3. Enter skill category information: Name, Note.	
		4. Validate skill category information. Display "Complete!" message.
Exception	1A: if role is not correct, show error message and ask to log in again. 4A: if skill category information is not correct, show error message and ask to input again.	
Open Issues	N/A	

9. 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.)*

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

10. How to Create Use Case Diagram

1. List main system functions (use cases) in a column (think of business events demanding system’s response; think of users’ goals to be accomplished via the system)
2. Draw ovals around the function labels
3. Draw system boundary
4. Draw actors and connect them with use cases (if more intuitive, this can be done as step 2)
5. Specify include and extend relationships between use cases

10. How to Create Use Case Diagram (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 ?

10. How to Create Use Case Diagram (cont.)

- 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.

11. Challenges

- Develop a Use Case Diagram to describe the ATM System

No more documents, please!

Q & A