

CSC 322 SOFTWARE ENGINEERING

LAB

USE-CASE MODELING

Fall 2017

USE-CASE DIAGRAMS

- provide a graphical overview of a system's functionality
- consists of a set of actions (referred to as use cases) that the concerned system can perform, one or more actors, and dependencies among them.
- highly useful for exposing requirements and planning almost every project
- helps end-users, developers, and domain experts get a common understanding of a system

ACTORS

- An actor can be defined as an object or set of objects, external to the system, which interacts with the system to get some meaningful work done
- 2 types of actor classification:
 - *Primary actor*: They are principal users of the system, who fulfill their goal by availing some service from the system. For example, a customer uses an ATM to withdraw cash when he needs it. A customer is the primary actor here.
 - *Supporting actor*: They render some kind of service to the system. "Bank representatives", who replenish the stock of cash, is such an example. It may be noted that replenishing stock of cash in an ATM is not the prime functionality of an ATM.

IDENTIFYING AN ACTOR

- Take the following into consideration:
 - Who/what will be interested in the system
 - Who/what will want to change the data in the system
 - Who/what will want to interface with the system
 - Who/what will want information from the system

USE CASE

- A use case is simply a functionality provided by a system
- Use case should ideally begin with a verb. Should not be open ended.
 - Register, wrong.
 - Register New User, right.
- ATM Use-Case Example
 - Withdraw cash
 - Check account balance
 - Change pin

SUBJECT

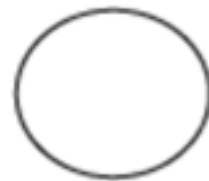
- A subject is simply the system under consideration.
- Use cases apply to a subject.
- For example, an ATM is a subject, having multiple use cases, and multiple actors interact with it. However, one should be careful of external systems interacting with the subject as actors.

GRAPHICAL REPRESENTATION

Actor



Usecase

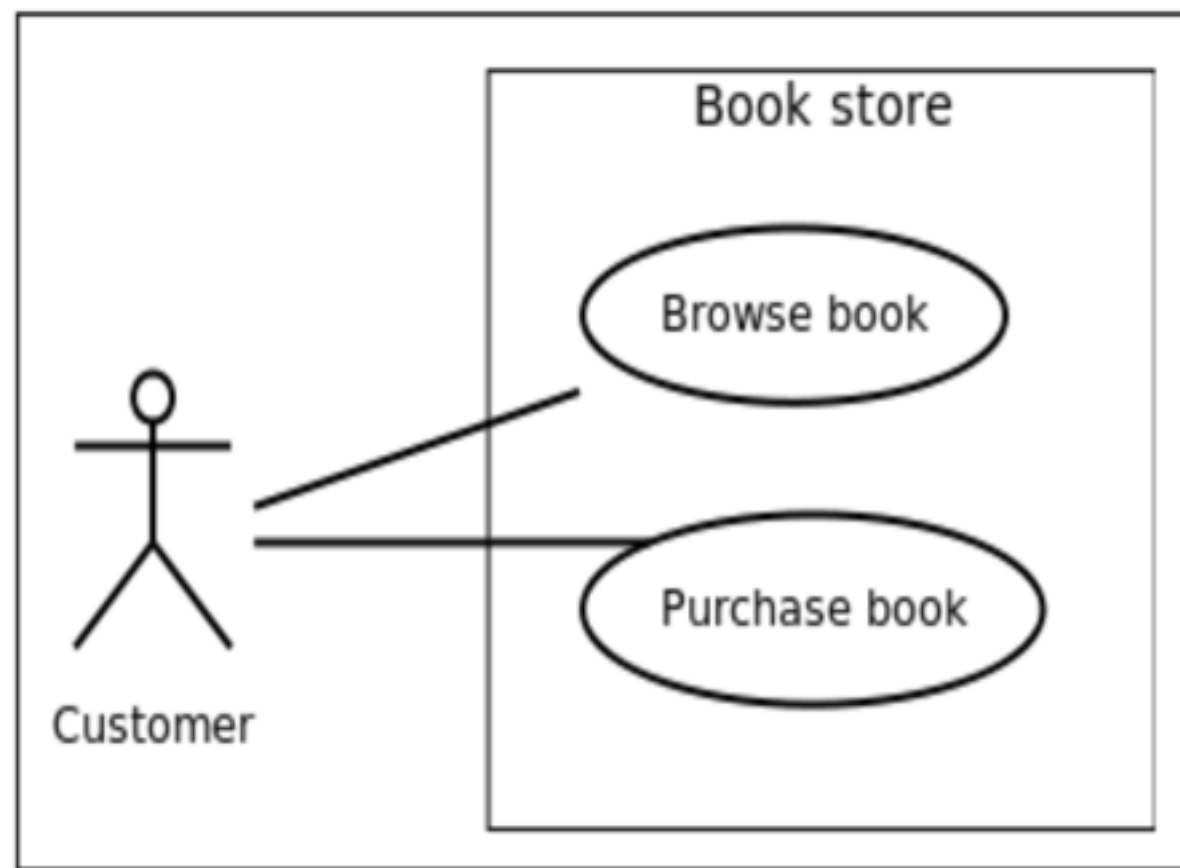


System



- An actor is represented by a stick figure and name of the actor is written below it
- A use case is depicted by an ellipse and name of the use case is written inside it
- A subject or system is shown by drawing a rectangle.
 - Anything within the box represents functionality that is in scope and anything outside is not

A USE CASE DIAGRAM FOR A BOOK STORE

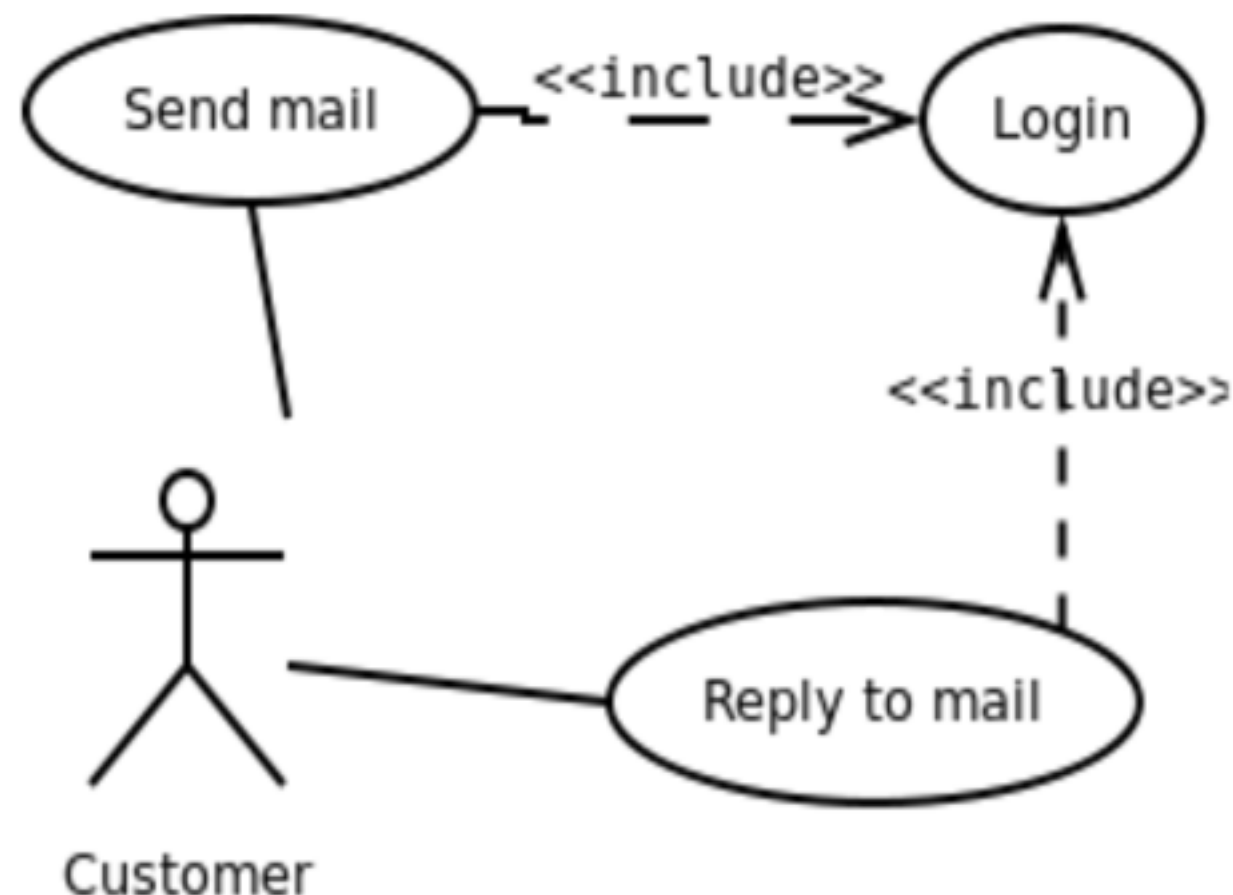


USE CASE RELATIONSHIPS

- Three types of relationships exist among use cases:
 - Include relationship
 - Extend relationship
 - Use case generalization

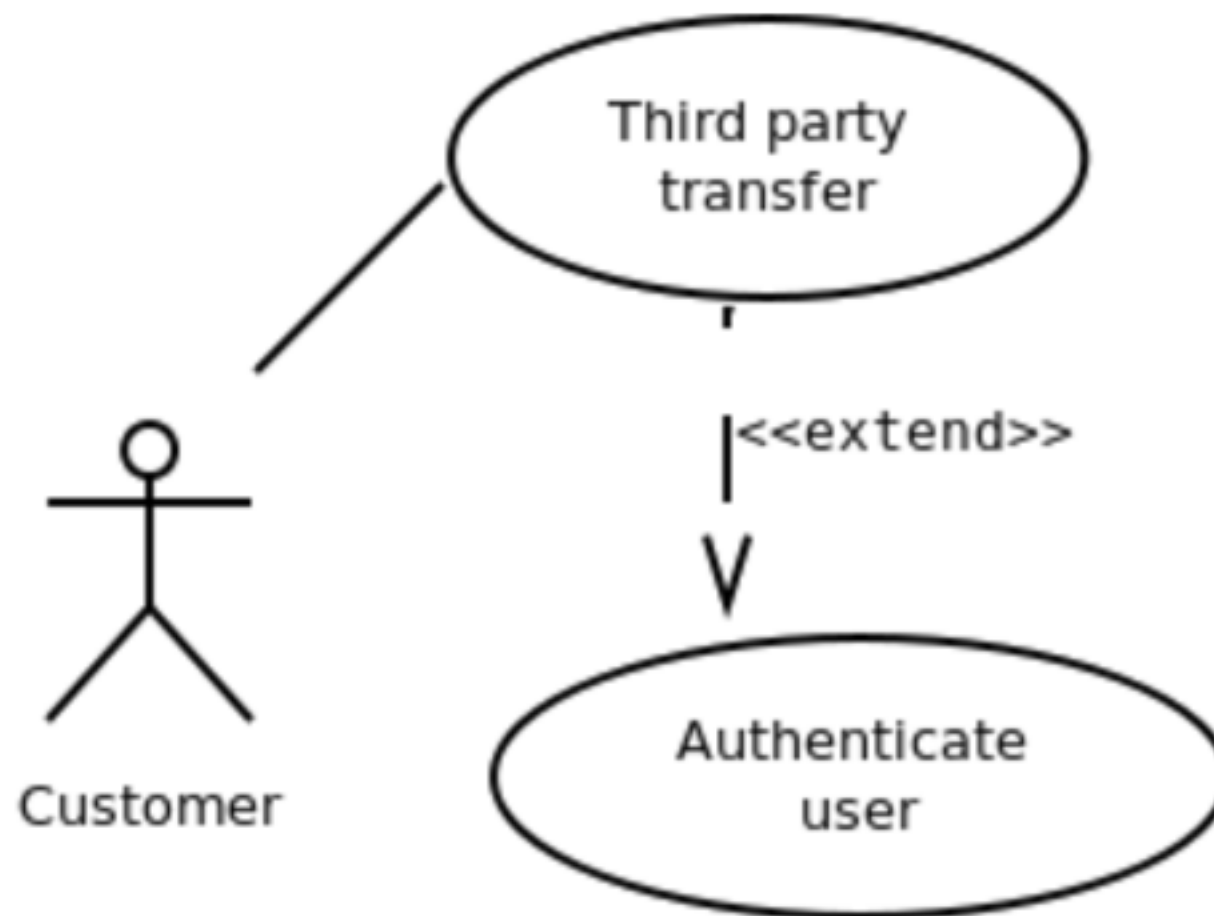
INCLUDE RELATIONSHIP

- Include relationships are used to depict common behavior that are shared by multiple use cases.
- Include relationship is depicted by a dashed arrow with an «include» stereotype from the including use case to the included use case.



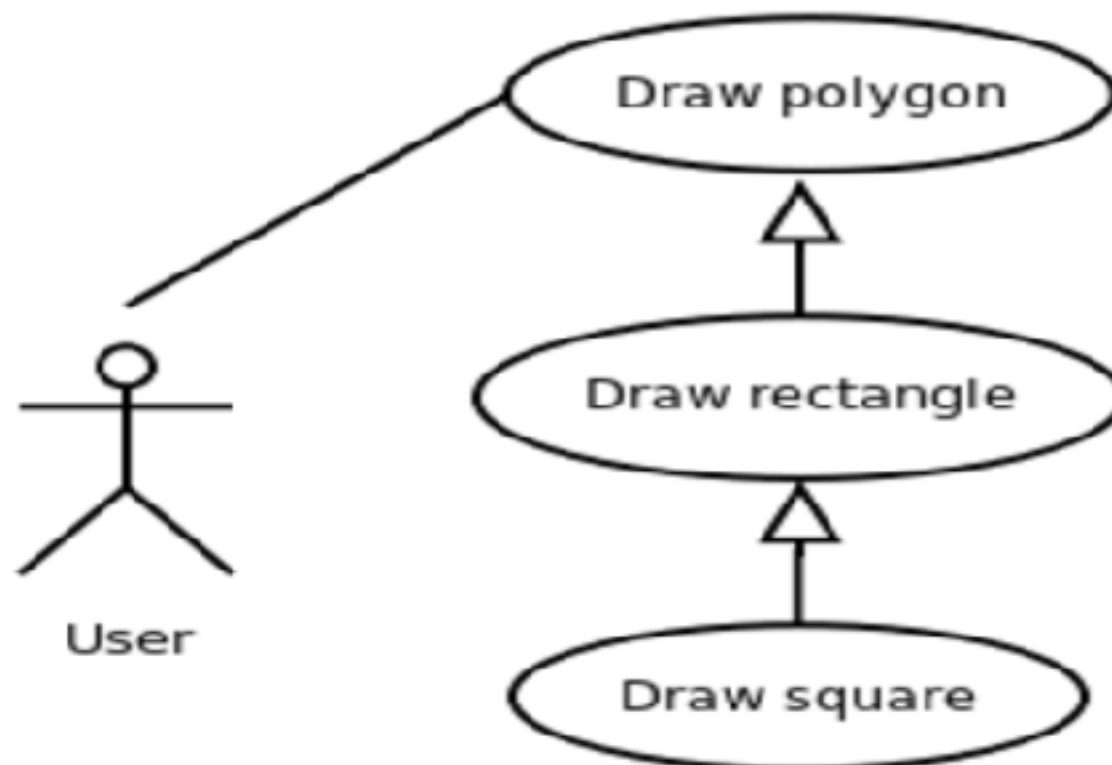
EXTEND RELATIONSHIP

- Use case extensions are used to depict any variation to an existing use case.
- Extend relationship is depicted by a dashed arrow with an «extend» stereotype from the extending use case to the extended use case.



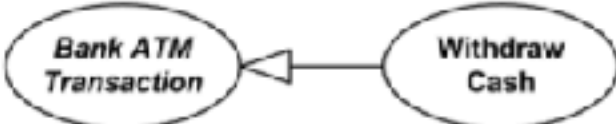
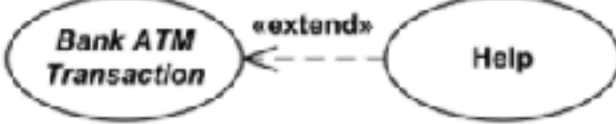
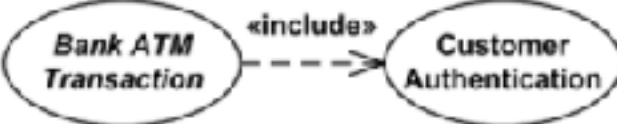
GENERALIZATION RELATIONSHIP

- Generalization relationships are used to represent the inheritance between use cases.
- A derived use case specializes some functionality it has already inherited from the base use case.
- Generalization relationship is depicted by a solid arrow from the specialized (derived) use case to the more generalized (base) use case.



Use Case Relationships Compared

This site received many requests related to which use case relationship should be used in which situation. I combined several key points from UML 2.4 Specification into the table below. Also, take a look at related discussion in the next paragraph.

Generalization	Extend	Include
		
Base use case could be abstract use case (incomplete) or concrete (complete).	Base use case is complete (concrete) by itself, defined independently.	Base use case is incomplete (abstract use case).
Specialized use case is required, not optional, if base use case is abstract.	Extending use case is optional, supplementary.	Included use case required, not optional.
No explicit location to use specialization.	Has at least one explicit extension location.	No explicit inclusion location but is included at some location.
No explicit condition to use specialization.	Could have optional extension condition.	No explicit inclusion condition.