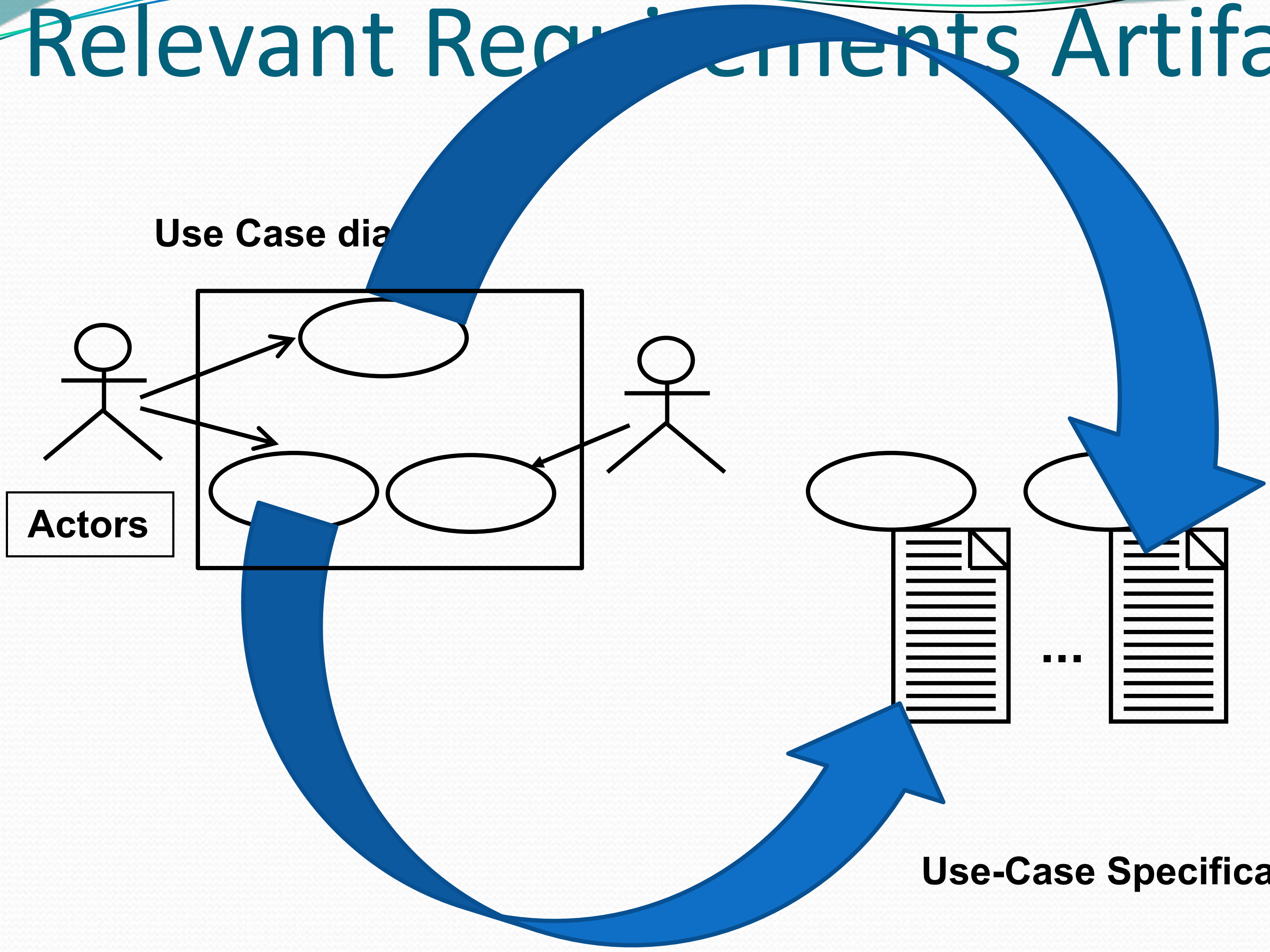
# Use Case Diagrams

Continued

## The Use Case

- Use Cases define the <u>behaviour performed</u> by system
  - Shows how system behaves from <u>**the users' point of view**</u>
- Captures the <u>users'</u> functional requirements of a system
- Describes WHAT the system will do for the user

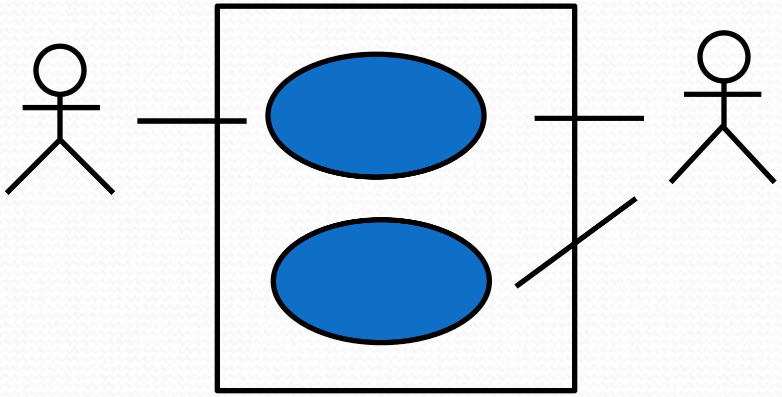
# Relevant Requirements Artifacts



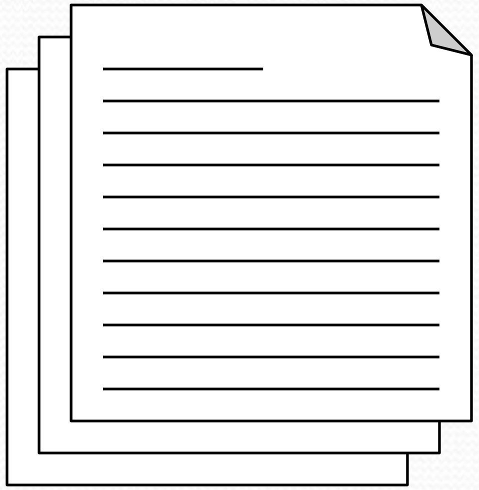
A use-case model is comprised of:

## Capture a use-case model

**Use-case diagrams**
(visual representation)

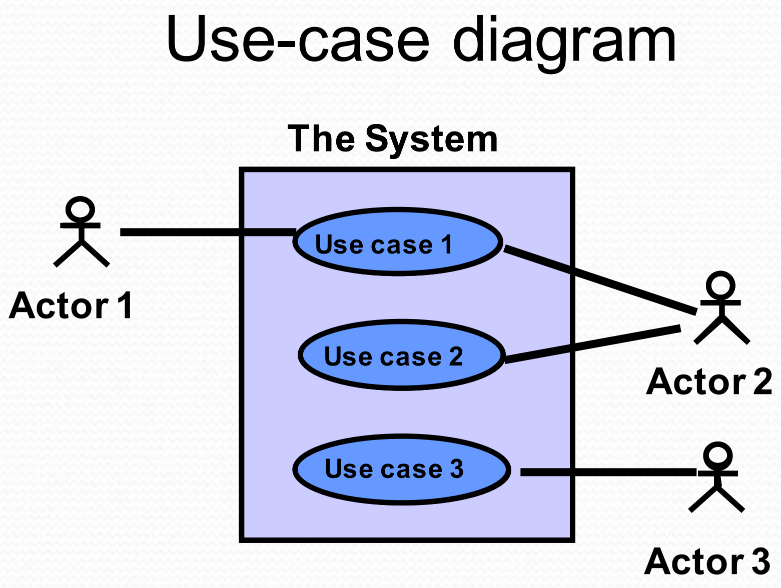


**Use-case specifications**
(text representation)



## Use-case diagram

- Shows a set of use cases and actors and their relationships
- Defines clear boundaries of a system
- Identifies who or what interacts with the system
- Summarizes the behavior of the system



# Process of writing use cases

Find actors

Student

---

Find use cases

Register for Course

**Brief description**: This use case allows a Student to register for course offerings in the current semester. A Schedule of core and option courses is produced.

---

Outline a use case

**Register for Course Outline**

-Flow of events

    -Step by step

---

Detail a use case

**Register for Course Use-Case Specification**

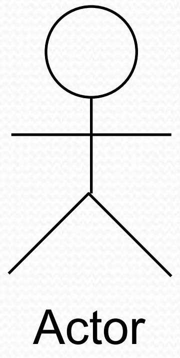
-Detailed Flow of Events

-Special Requirements

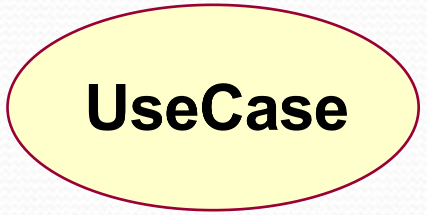
-Pre/Postconditions

## Major Concepts in Use-Case Modeling

- An actor represents anything that interacts with the system.

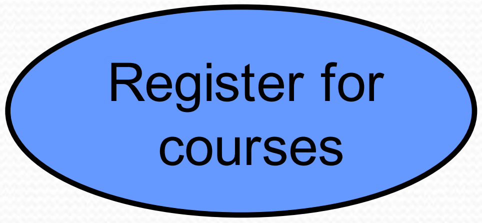


- A use case is a sequence of actions a system performs that yields an observable result of value to a particular actor.



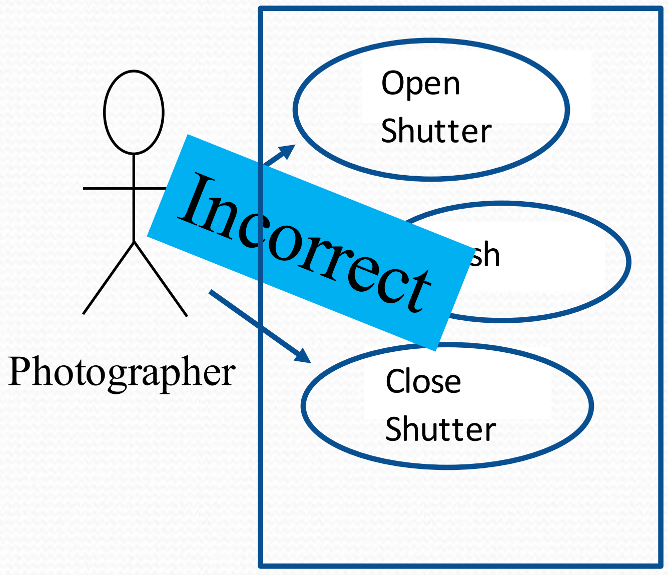
## Name the use case

- A use case name should:
  - Be unique, intuitive, and self-explanatory
  - Define clearly and unambiguously the observable result of value gained from the use case
  - Be from the perspective of the actor that triggers the use case
  - Describe the behavior that the use case supports
  - Start with a verb and use a simple verb-noun combination

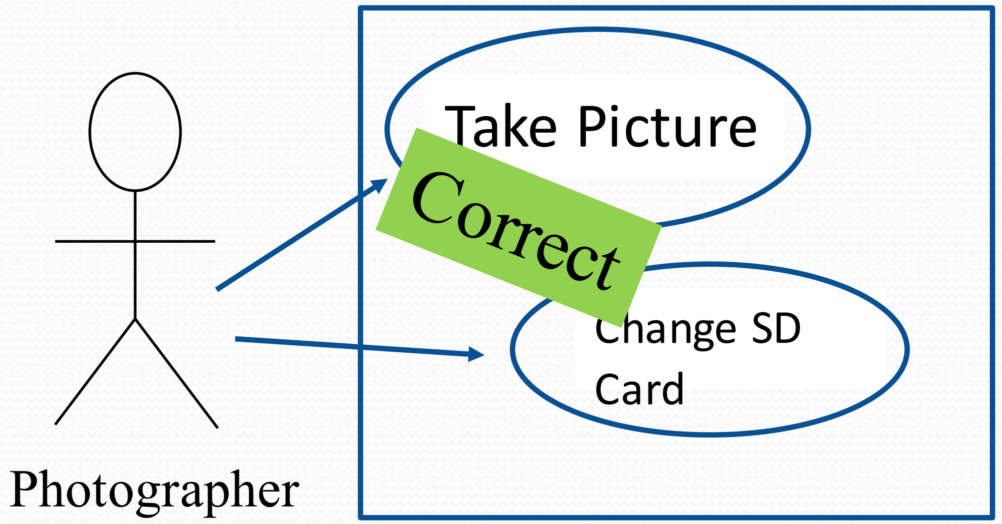


# Example:Use Case for a camera

Certainly these are all behaviours that a camera has, but no photographer would ever pick up their camera, open the shutter, and then put it down, satisfied with their photographic session for the day.



The crucial thing to realize is that these behaviors are not done in isolation, but are rather a **part** of a more high-level use case, "Take Picture".

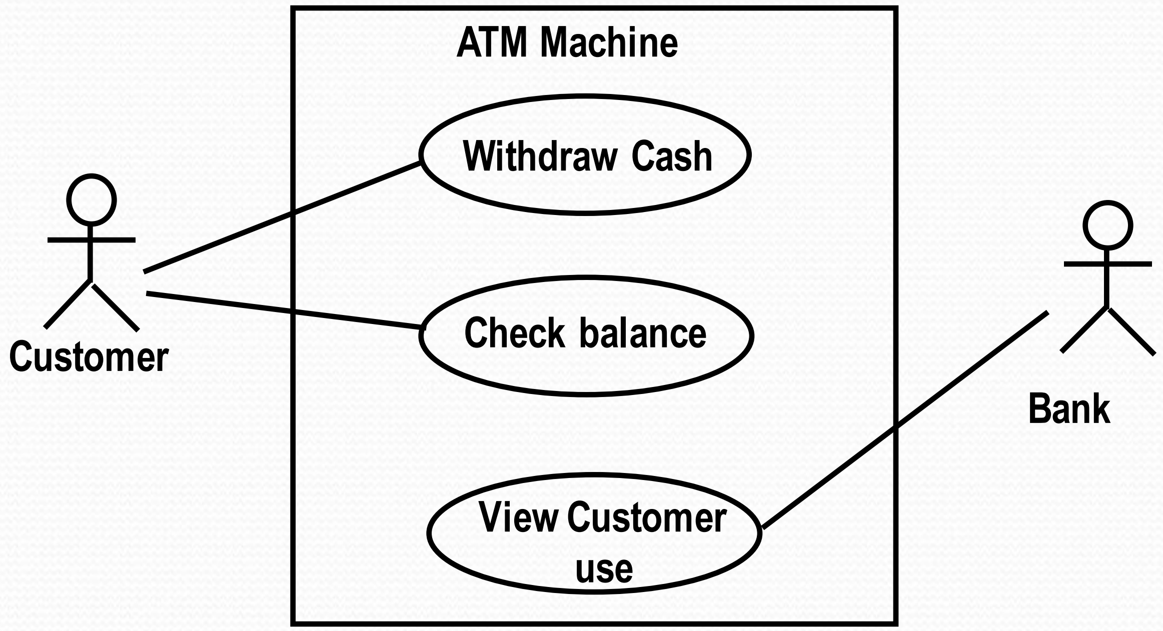


## Use Case ‘Simple’ Description of a requirement.

| | | | |
|---|---|---|---|
| ***Use Case No:*** **11** | ***Use Case Name:*** **Withdraw cash** | | ***Rating Must Have*** |
| ***Purpose:*** | | | |
| ***Main actor:*** | ***Secondary Actors:*** | | |
| ***Description:*** | | | |

# Modelling: Use Case Diagram

## ATM Example



- Shows how system should behave from stakeholders’ perspective

## Use Case ‘Simple’ Description of a requirement.

| | | |
|---|---|---|
| ***Use Case No:*** **11** | ***Use Case Name:*** **Withdraw cash** | ***Rating Must Have*** |

***Purpose: Customer should be able to withdraw*** **a specified amount from their current account**

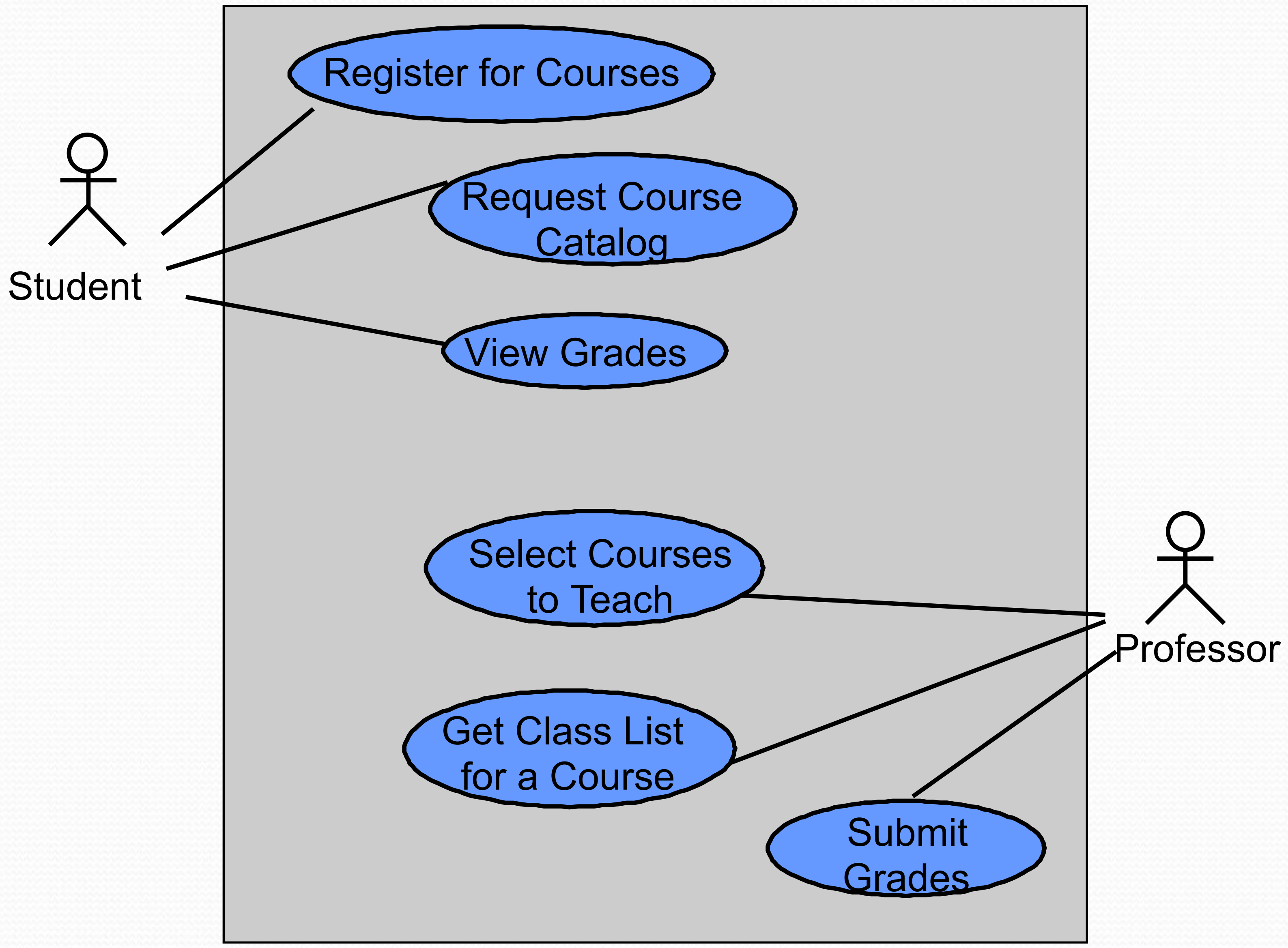
| | |
|---|---|
| ***Main actor:*** **Customer** | ***Secondary Actors:*** |

***Description:***
**The customer Must be prompted to specify the amount of cash. The system will check the customer’s current account to ensure they have that amount to withdraw. The system will them dispense the correct amount to the client and deduct that amount from the customers current account.**

**Non-functional requirements**

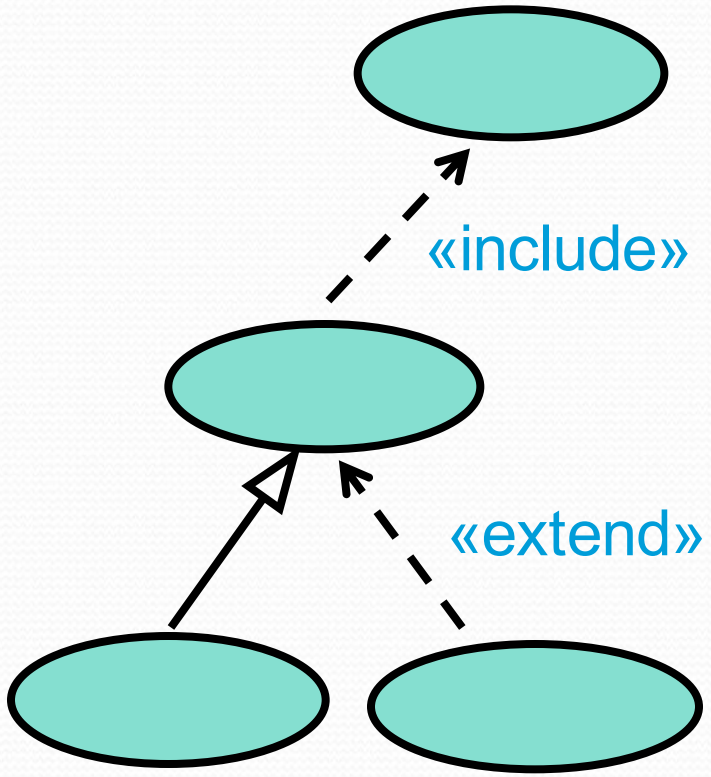
- **Ease of use: A customer must be able to withdraw cash with out any training**
- **Accuracy: The amount of cash dispensed and deducted for the customers account most be accurate 100% of the time.**
- **Security: The Customer must only be able to withdraw cash from their account.**
- **Response: The system must dispense the cash within 1 minute.**

# Use-case diagram example



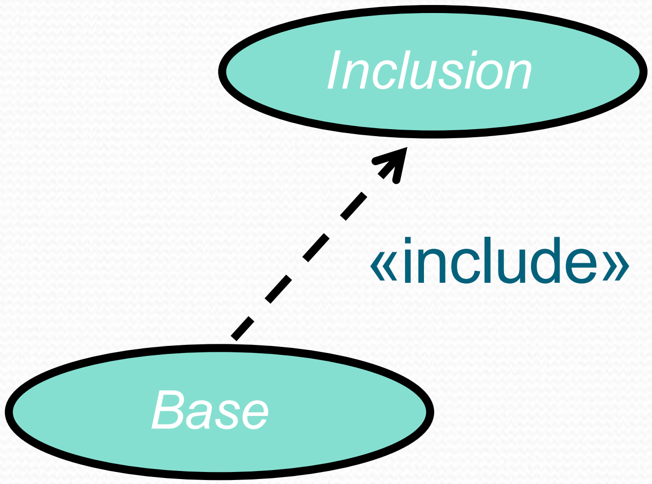
## Relationships Between Use Cases

- Include
- Extend



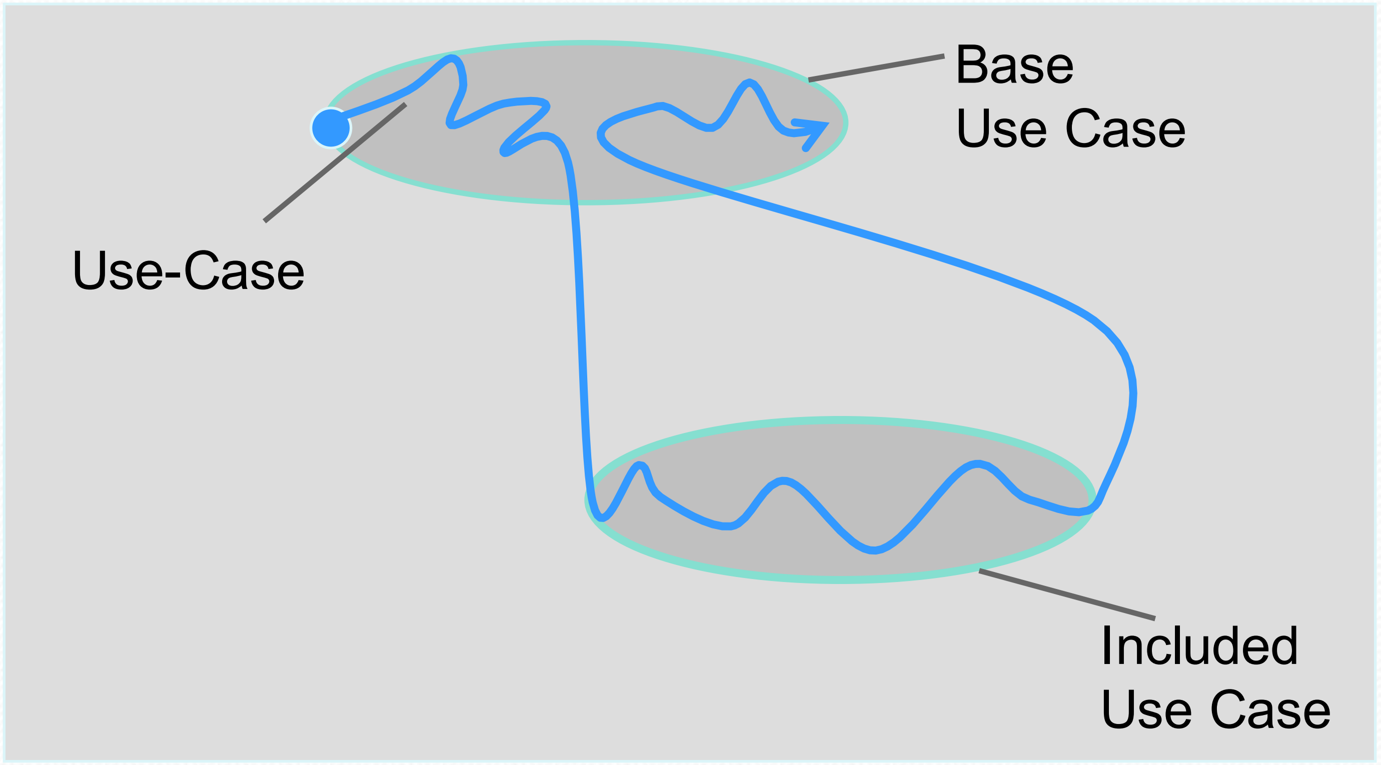
# What Is an Include Relationship?

- A relationship from a base use case to an inclusion use case.
- The behavior defined in the inclusion use case is **explicitly included** into the base use case.



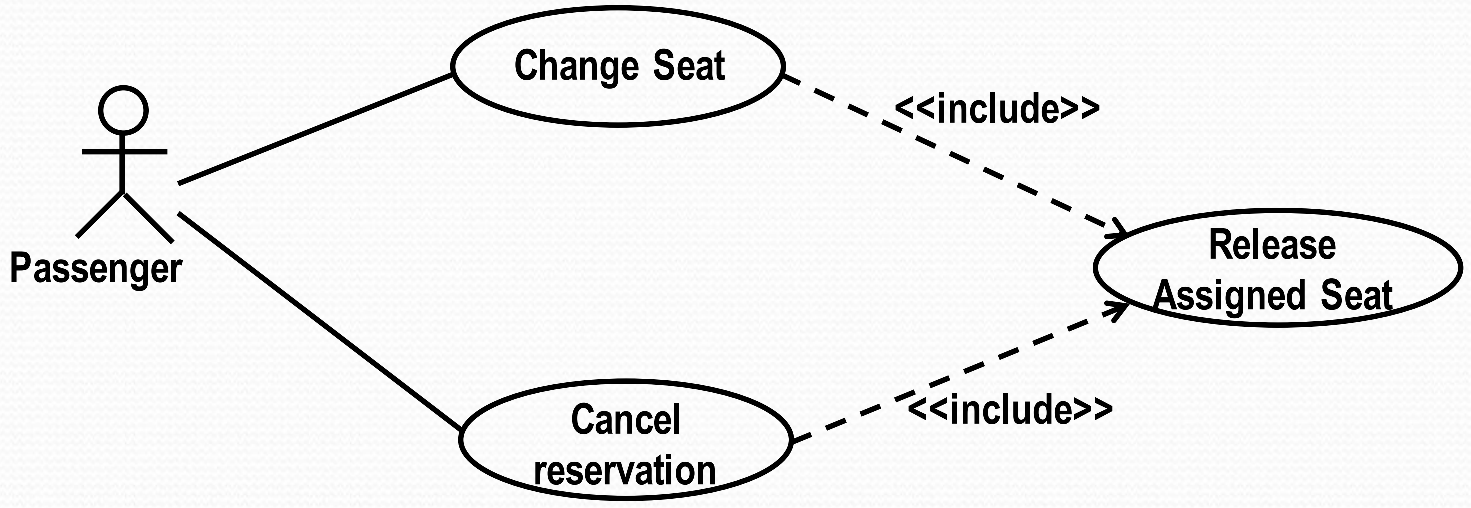
## Execution of an Include Relationship

- Fully executed when the inclusion point is reached.



## Include Relationship: Example

- Allows us to factor out common behaviour from two or more use cases



# Why Use an Include Relationship?

- Factor out behaviour common to two or more use cases.
  - Avoid describing same behavior multiple times.
  - Assure common behavior remains consistent.
- Factor out and encapsulate behaviour from a base use case that is complex or long

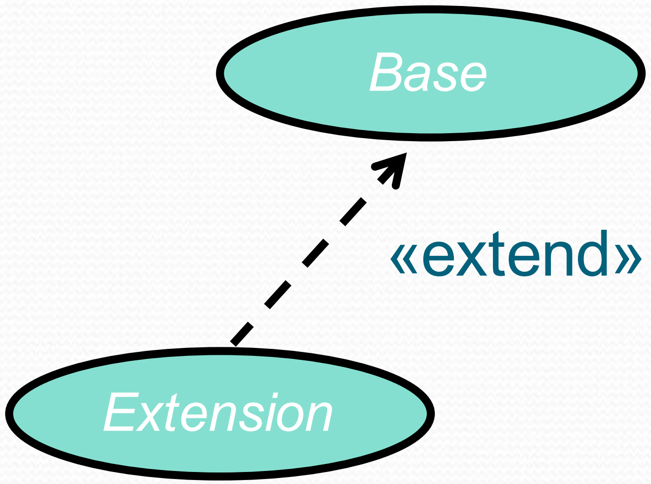
(when the base use case only depends on the result and not the method used to produce the result of the factored behaviour).

  - Simplify complex flow of events.
  - Factor out behavior not part of primary purpose.

# What Is an Extend Relationship?

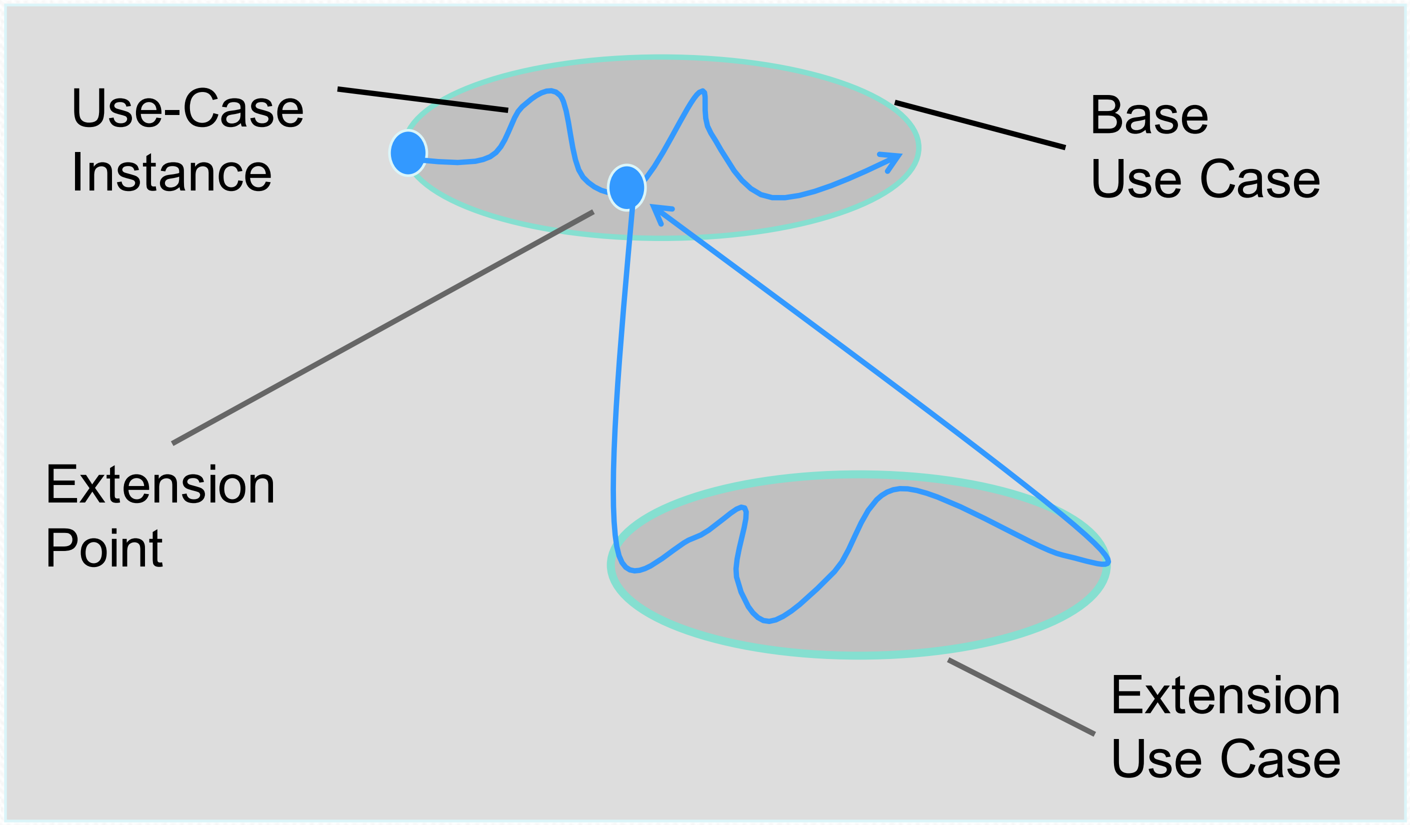
Optional behaviour

- Connects from an extension use case to a base use case.
  - extension use case's behavior into base use case.
  - only if the extending condition is true.
  - Insert into base use case at named extension point.

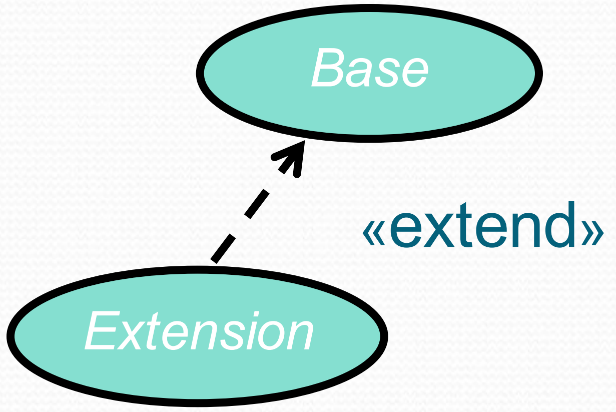


# Execution of an Extend

- Executed when the extension point is reached and the extending condition is true.



# Why Use an Extend Relationship?



- Factor out optional or exceptional behavior.
  - Executed only under certain conditions.
  - Factoring out simplifies flow of events in base use case.

Example: Triggering an alarm.

## «include» vs «extend»

- Both try to factor out common behaviour from several use cases to a single one.
- Apply the following rules
  - Use «extend» when **you are describing a variation on normal behaviour**
  - Use «include» **when you are repeating yourself in two or more separate use cases**

## Final remarks

- We will return to UML and class diagrams a bit later but for now

  You have all the information needed for the whole of coursework 1 - requirements

Change of deadline for draft hand in

Now

9:30am Wednesday 21st Nov