Object-Oriented Software Engineering
Using UML, Patterns, and Java

Chapter 5
Functional Modeling

(Textbook Chapter 4)

Course instructor : TA.Nada Alamoudi

Outline

- ✓ Scenarios (last lecture)
  - ✓ Finding Scenarios
  - ✓ Identifying actors
- ➢ Use Cases
  - Finding Use Cases
  - Flow of Events
  - Use Case Associations
  - Use Case Refinement
- Summary

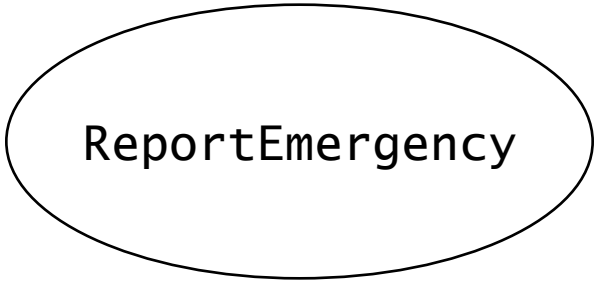
UML Use Case Diagrams

Used during requirements elicitation and analysis to represent external behavior ("visible from the outside of the system")

Use Case Modeling

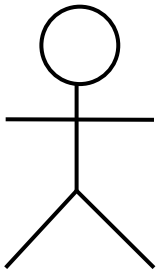
- A use case is a flow of events in the system, including interaction with actors

- Each use case has a name
- Each use case has a termination condition
- Graphical notation: An oval with the name of the use case



Use Case Model: The set of all use cases specifying the complete functionality of the system

Actors



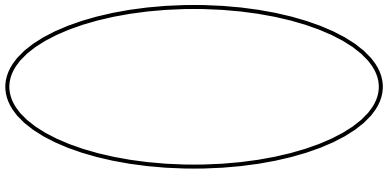
Passenger

- An actor is a model for an external entity which interacts (communicates) with the system:
  - User
  - External system (Another system)
  - Physical environment (e.g. Weather)
- An actor has a unique name and an optional description
- Examples:
  - **Passenger**: A person in the train
  - **GPS satellite**: An external system that provides the system with GPS coordinates.

Optional Description

Name

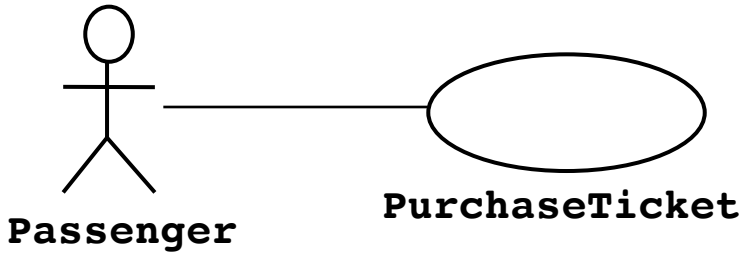
Use Case



PurchaseTicket

- A use case represents a class of functionality provided by the system
- Use cases can be described textually, with a focus on the event flow between actor and system
- The textual use case description consists of 6 parts:
  1. Unique name
  2. Participating actors
  3. Entry conditions
  4. Exit conditions
  5. Flow of events
  6. Special requirements.

Textual Use Case Description Example



1. *Name:* `Purchase ticket`

2. *Participating actor:*
   `Passenger`

3. *Entry condition:*
   - `Passenger` stands in front of ticket distributor
   - `Passenger` has sufficient money to purchase ticket

4. *Exit condition:*
   - `Passenger` has ticket

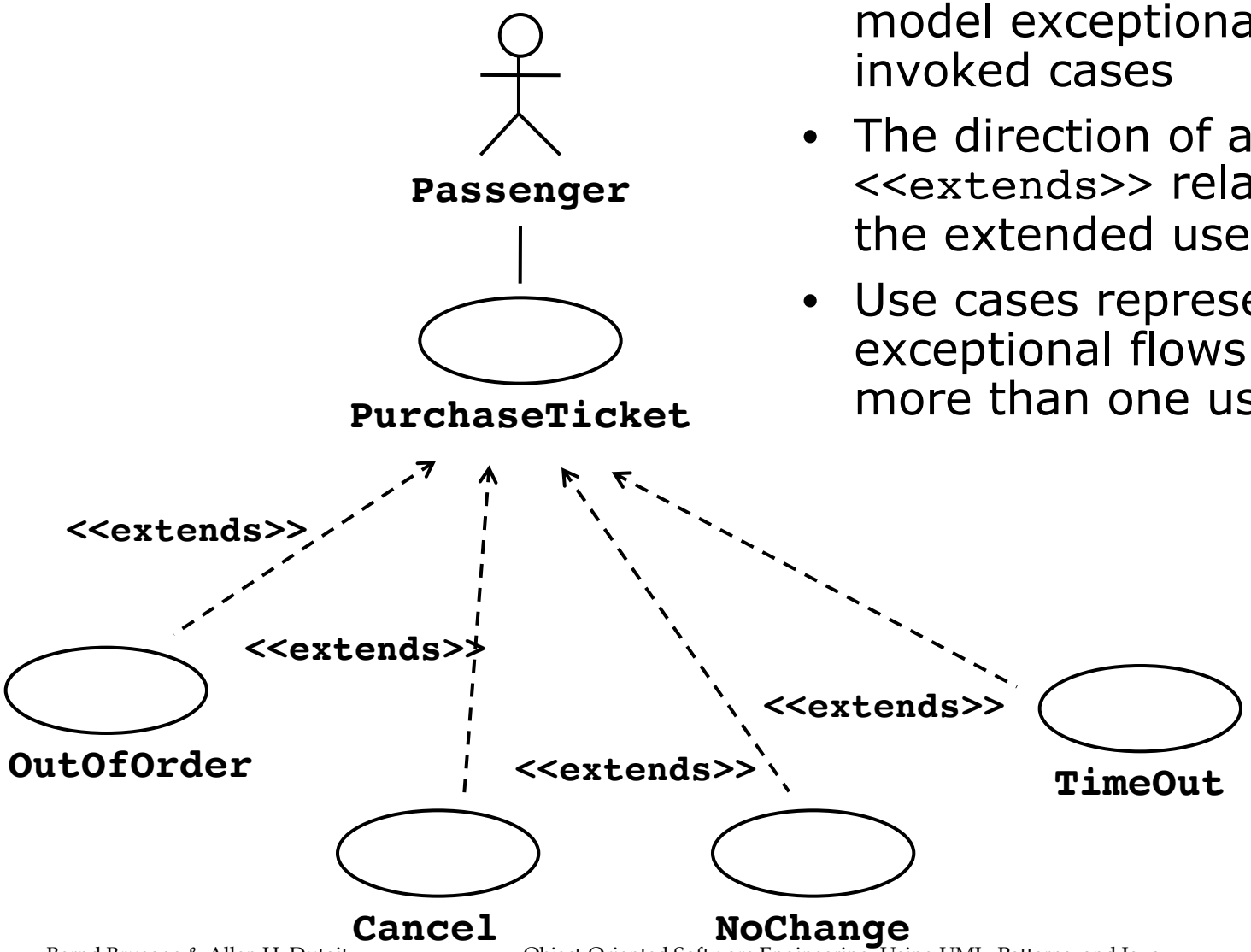
5. *Flow of events:*
   1. `Passenger` selects the number of zones to be traveled
   2. Ticket Distributor displays the amount due
   3. `Passenger` inserts money, at least the amount due
   4. Ticket Distributor returns change
   5. Ticket Distributor issues ticket

6. *Special requirements:*
   None.

Use Case Associations

- Dependencies between use cases are represented with use case associations
- Associations are used to reduce complexity
  - Decompose a long use case into shorter ones
  - Separate alternate flows of events
  - Refine abstract use cases
- Types of use case associations
  - Extends
  - Includes
  - Generalization

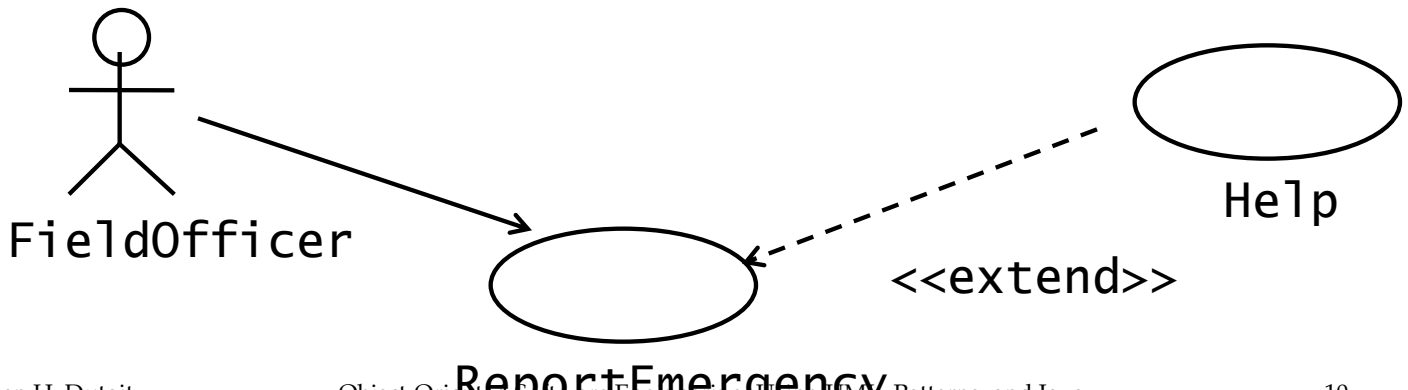
The <<*extends*>> Relationship



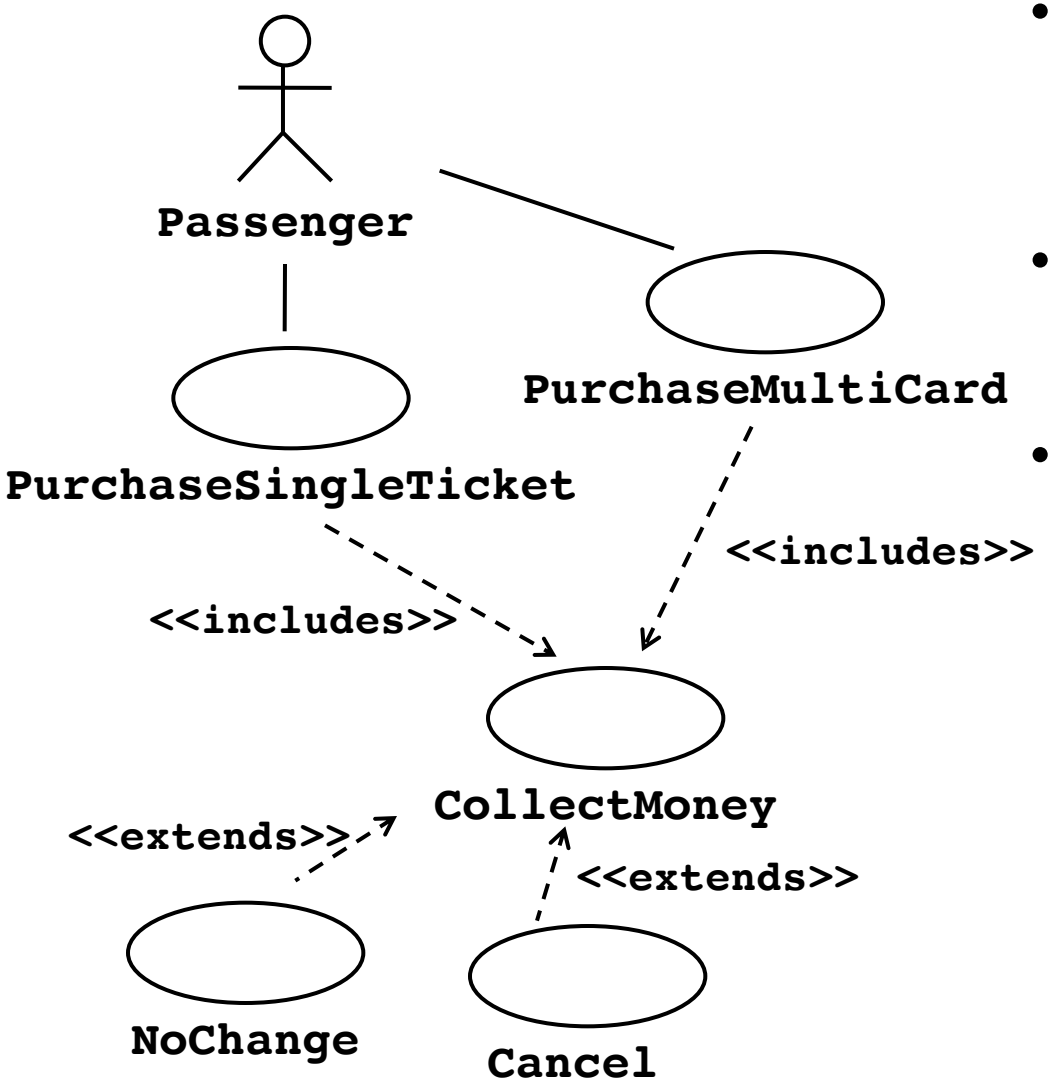
- `<<extends>>` relationships model exceptional or seldom invoked cases
- The direction of an `<<extends>>` relationship is to the extended use case
- Use cases representing exceptional flows can extend more than one use case.

<<extend>> Association for Use Cases

- Problem: The functionality in the original problem statement needs to be extended.
- Solution: An *extend association* from use case A to use case B
- Example: "ReportEmergency" is complete by itself, but can be extended by use case "Help" for a scenario in which the user requires help



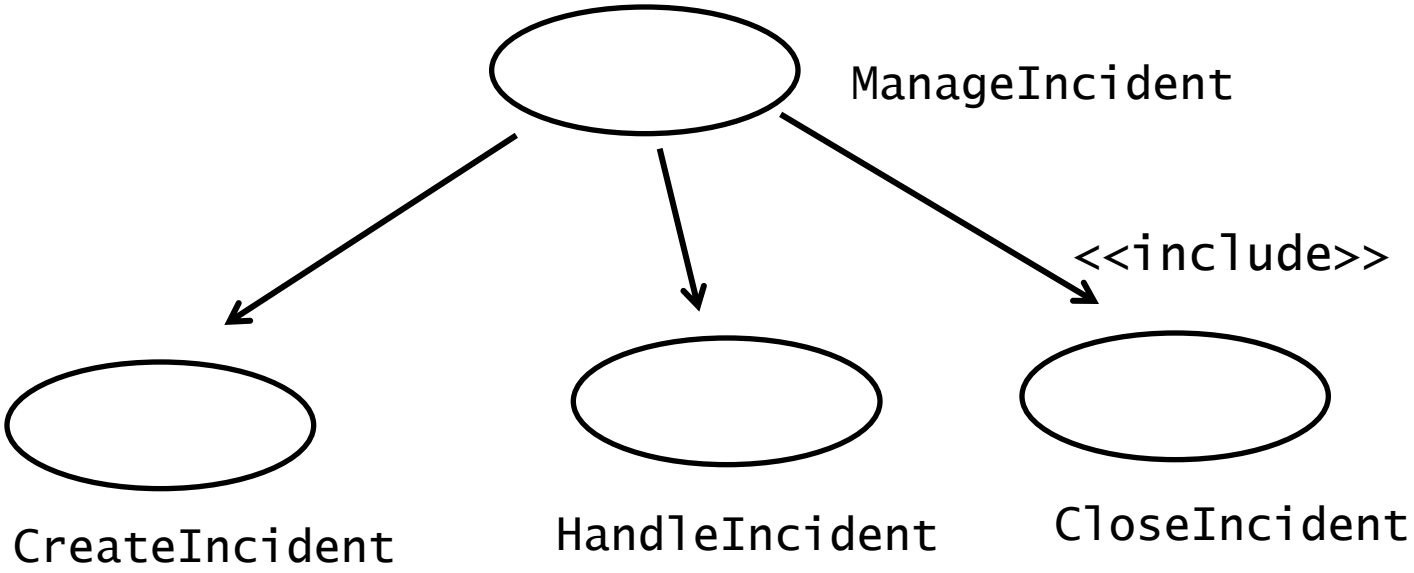
The <<*includes*>> Relationship



- <<includes>> relationship represents common functionality needed in more than one use case
- <<includes>> behavior is factored out for reuse, not because it is an exception
- The direction of a <<includes>> relationship is to the using use case (unlike the direction of the <<extends>> relationship).

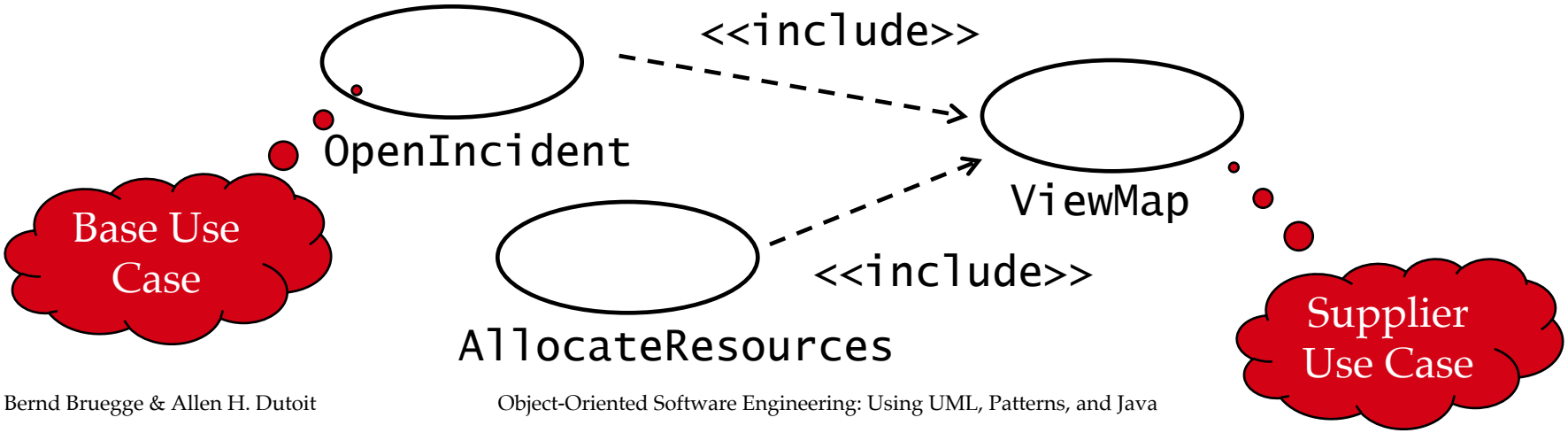
<<include>>: Functional Decomposition

- Problem:
  - A function in the original problem statement is too complex
- Solution:
  - Describe the function as the aggregation of a set of simpler functions. The associated use case is decomposed into shorter use cases



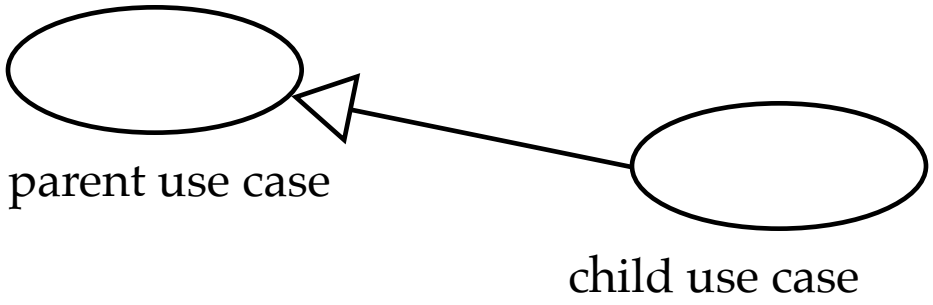
<<include>>: Reuse of Existing Functionality

- Problem: There are overlaps among use cases. How can we *reuse* flows of events instead of duplicating them?
- Solution: The *includes association* from use case A to use case B indicates that an instance of use case A performs all the behavior described in use case B ("A delegates to B")
- Example: Use case "ViewMap" describes behavior that can be used by use case "OpenIncident" ("ViewMap" is factored out)



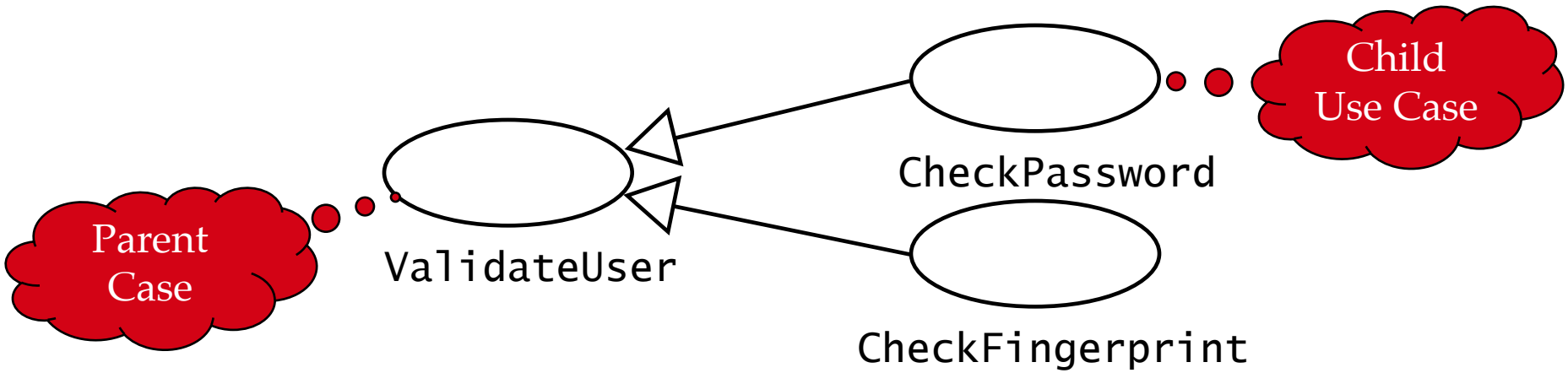
The Generalization Relationship

- Generalization between use cases is similar to generalization between classes – child use case inherits properties and behavior of the parent use case and may override the behavior of the parent.
- Generalization is shown as a solid directed line with a large hollow triangle arrowhead, directed from the more specific use case to the general use case.



Generalization in Use Cases

- Problem: We want to factor out common (but not identical) behavior.
- Solution: The child use cases inherit the behavior and meaning of the parent use case and add or override some behavior.
- Example: "ValidateUser" is responsible for verifying the identity of the user. The customer might require two realizations: "CheckPassword" and "CheckFingerprint"



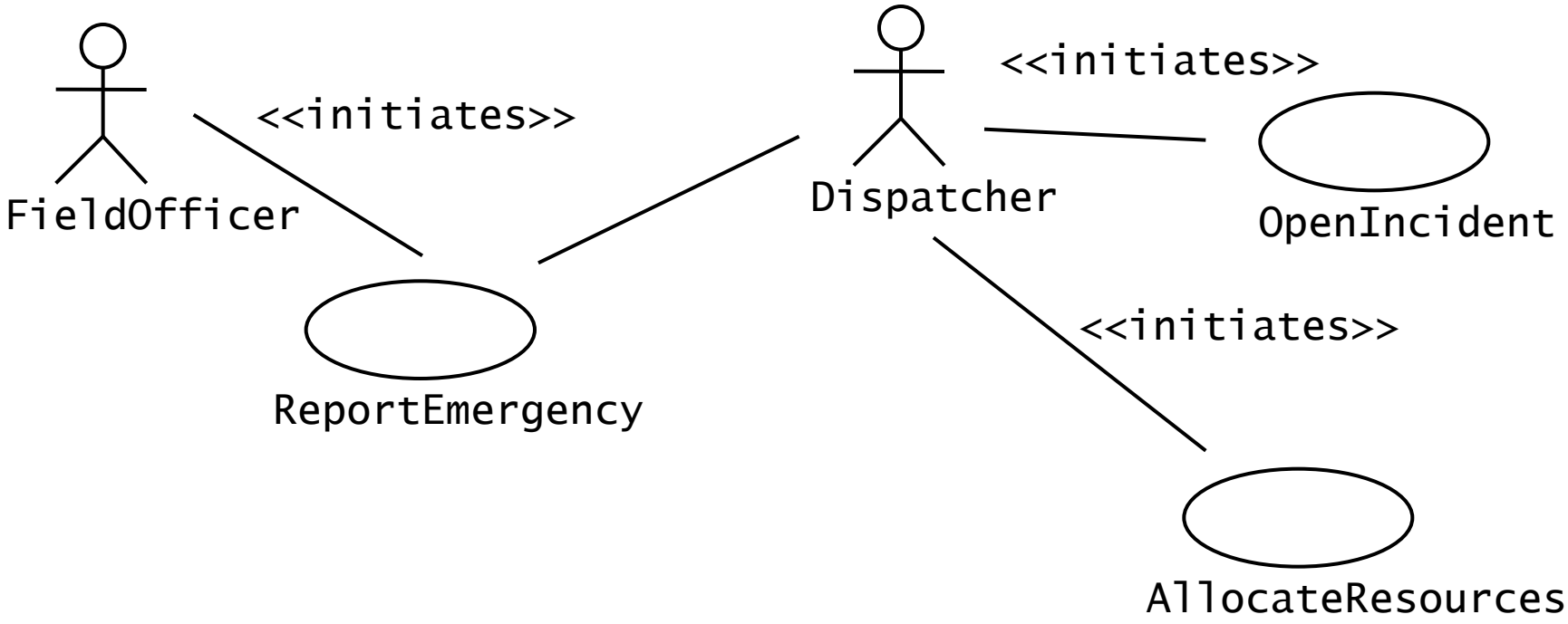
Scenario example: Warehouse on Fire

- Bob, driving down main street in his patrol car notices smoke coming out of a warehouse. His partner, Alice, reports the emergency from her car.
- Alice enters the address of the building into her wearable computer, a brief description of its location (i.e., north west corner), and an emergency level.
- She confirms her input and waits for an acknowledgment.
- John, the dispatcher, is alerted to the emergency by a beep of his workstation. He reviews the information submitted by Alice and acknowledges the report. He allocates a fire unit and sends the estimated arrival time (ETA) to Alice.
- Alice received the acknowledgment and the ETA.

Observations about Warehouse on Fire Scenario

- Concrete scenario
  - Describes a single instance of reporting a fire incident.
  - Does not describe all possible situations in which a fire can be reported.
- Participating actors
  - Bob, Alice and John

Use Case Model for Incident Management



Use Case Example: ReportEmergency

- Use case name: ReportEmergency
- Participating Actors:
  - Field Officer (Bob and Alice in the Scenario)
  - Dispatcher (John in the Scenario)
- Exceptions:
  - The FieldOfficer is notified immediately if the connection between terminal and central is lost.
  - The Dispatcher is notified immediately if the connection between a FieldOfficer and central is lost.
- Flow of Events: **on next slide.**
- Special Requirements:
  - The FieldOfficer's report is acknowledged within 30 seconds. The selected response arrives no later than 30 seconds after it is sent by the Dispatcher.

Use Case Example: ReportEmergency Flow of Events

1. The **FieldOfficer** activates the "Report Emergency" function of her terminal. The System responses by presenting a form to the officer.
2. The FieldOfficer fills the form, by selecting the emergency level, type, location, and brief description of the situation. The FieldOfficer also describes a response to the emergency situation. Once the form is completed, the FieldOfficer submits the form, and the **Dispatcher** is notified.
3. The Dispatcher creates an Incident in the database by invoking the OpenIncident use case. He selects a response and acknowledges the report.
4. The FieldOfficer receives the acknowledgment and the selected response.

Order of steps when formulating use cases

- First step: Name the use case
  - Use case name: ReportEmergency
- Second step: Find the actors
  - Generalize the concrete names (“Bob”) to participating actors (“Field officer”)
  - Participating Actors:
    - Field Officer (Bob and Alice in the Scenario)
    - Dispatcher (John in the Scenario)
- Third step: Concentrate on the flow of events
  - Use informal natural language

Another Use Case Example

Actor **Bank Customer**

- Person who owns one or more Accounts in the Bank.

Withdraw Money

- The Bank Customer specifies an Account and provides credentials to the Bank proving that s/he is authorized to access the Bank Account.
- The Bank Customer specifies the amount of money s/he wishes to withdraw.
- The Bank checks if the amount is consistent with the rules of the Bank and the state of the Bank Customer's account. If that is the case, the Bank Customer receives the money in cash.

Use Case Attributes

Use Case **Withdraw Money Using ATM**

Initiatiating actor:

- Bank Customer

Preconditions:

- Bank Customer has opened a Bank Account with the Bank ***and***
- Bank Customer has received an ATM Card and PIN

Postconditions:

- Bank Customer has the requested cash ***or***
- Bank Customer receives an explanation from the ATM about why the cash could not be dispensed

Use Case Flow of Events

Actor steps	System steps
1.The Bank Customer inputs the card into the ATM.	
	2.The ATM requests the input of a four-digit PIN.
3. The Bank Customer types in PIN.	
	4. If several accounts are recorded on the card, the ATM offers a choice of the account numbers for selection by the Bank Customer
5. The Bank Customer selects an account.	
	6.If only one account is recorded on the card or after the selection, the ATM requests the amount to be withdrawn.
7. The Bank Customer inputs an amount.	
	8.The ATM outputs the money and a receipt and stops the interaction.

Use Case Exceptions

Actor steps

1. The Bank Customer inputs her card into the ATM.**[Invalid card]**

3. The Bank Customer types in PIN. **[Invalid PIN]**

5. The Bank Customer selects an account .

7. The Bank Customer inputs an amount. **[Amount over limit]**

[Invalid card]
The ATM outputs the card and stops the interaction.

[Invalid PIN]
The ATM announces the failure and offers a 2nd try as well as canceling the whole use case. After 3 failures, it announces the possible retention of the card. After the 4th failure it keeps the card and stops the interaction.

[Amount over limit]
The ATM announces the failure and the available limit and offers a second try as well as canceling the whole use case.

Guidelines for Formulation of Use Cases (1)

- Name
  - Use a verb phrase to name the use case.
  - The name should indicate what the user is trying to accomplish.
  - Examples:
    - "Request Meeting", "Schedule Meeting", "Propose Alternate Date"
- Length
  - A use case description should not exceed 1-2 pages. If longer, use include relationships.
  - A use case should describe a complete set of interactions.

Guidelines for Formulation of Use Cases (2)

Flow of events:

- Use the active voice. Steps should start either with "The Actor" or "The System …".
- The causal relationship between the steps should be clear.
- All flow of events should be described (not only the main flow of event).
- The boundaries of the system should be clear. Components external to the system should be described as such.
- Define important terms in the glossary.

How to write a use case (Summary)

- Name of Use Case
- Actors
  - Description of Actors involved in use case
- Entry condition
  - "This use case starts when…"
- Flow of Events
  - Free form, informal natural language
- Exit condition
  - "This use cases terminates when…"
- Exceptions
  - Describe what happens if things go wrong
- Special Requirements
  - Nonfunctional Requirements, Constraints