Object-Oriented Software Engineering

Using UML, Patterns, and Java

Outline

- ✓ Scenarios (Lecture Requirements Elicitation)
  - ✓ Finding Scenarios
  - ✓ Identifying actors
- ➢ Use Cases
  - Finding Use Cases
  - Flow of Events
  - Use Case Associations
  - Use Case Refinement
- Summary

Scenario example from Tuesday's lecture: 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

Other Scenarios Possibilities for an Incident Management System

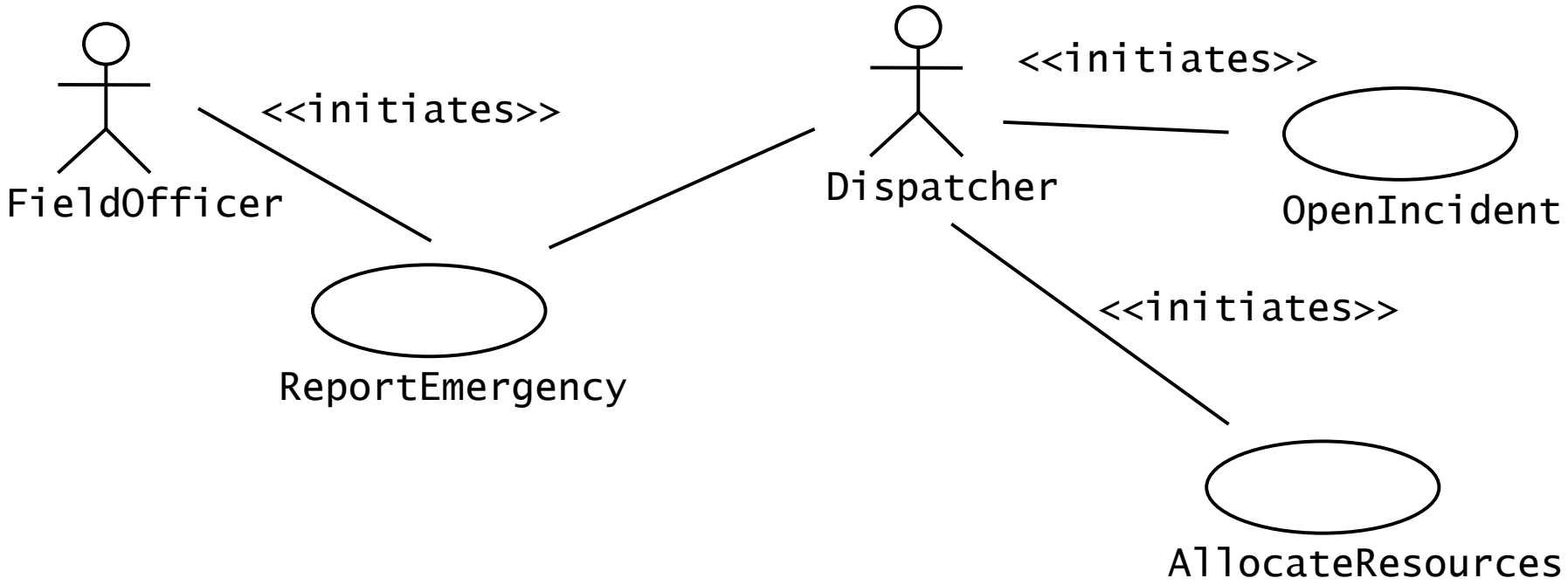
- What needs to be done to report a “Cat in a Tree” incident?
- Who is involved in reporting the incident?
- What does the system do, if no police cars are available? If the police car has an accident on the way to the “Cat in a Tree” incident?
- What do you need to do if the “Cat in the Tree” turns into a “Grandma Has Fallen From the Ladder”?
- Can the system cope with simultaneous incident reports “Cat in the Tree” and “Warehouse on Fire?”

After the scenarios are formulated

- Find all the scenarios that specify how to report a fire and model them in a use case

- Then add more detail to each of these use cases by describing:
  1. Name of the use case
  2. Participating actors
  3. Describe the entry condition
  4. Describe the flow of events
  5. Describe the exit condition
  6. Describe exceptions
  7. Describe quality requirements (nonfunctional requirements).

Use Case Model for Incident Management



How to find Use Cases

- Select a narrow vertical slice of the system (i.e. one scenario)
  - Discuss it in detail with the user to understand the user's preferred style of interaction
- Select a horizontal slice (i.e. many scenarios) to define the scope of the system.
  - Discuss the scope with the user
- Use illustrative prototypes (mock-ups) as visual support
- Find out what the user does
  - Task observation (Good)
  - Questionnaires (Bad)

Use Case Example: ReportEmergency

1. Use case name: ReportEmergency
2. Participating Actors:

   Field Officer(initiate), Dispatcher
3. Entry Condition:

   The FieldOfficer is logged into the FRIEND System
4. Flow of Events: **on next slide**
5. Exit Condition:

   The FieldOfficer has received an acknowledgement and the selected response OR The FieldOfficer has received an explanation indicating why the transaction could not be processed
6. Exceptions:
   - The FieldOfficer is notified immediately if the connection between terminal and central is lost
7. Quality Requirements:
   - The FieldOfficer's report is acknowledged within 30 seconds.

Use Case Example: ReportEmergency (ctd)

4. Flow of Events:
   1. The **FieldOfficer** activates the "Report Emergency" function of her terminal. The system responds 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 from the scenario to participating actors
  - 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

Flow of Events

- The Bank Customer specifies a Account and provides credentials to the Bank proving that he is authorized to access the Bank Account
- The Bank Customer specifies the amount of money 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 Name **Withdraw Money Using ATM**

Participating Actor: Bank Customer

Entry condition:

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

Exit condition:

- Bank Customer has the requested cash ***or***

  Bank Customer receives an explanation from the ATM about why the cash could not be dispensed.

Flow of Events: A Request-Response Interaction between Actor and System

Actor steps	System steps
1.The Bank Customer inserts 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 .

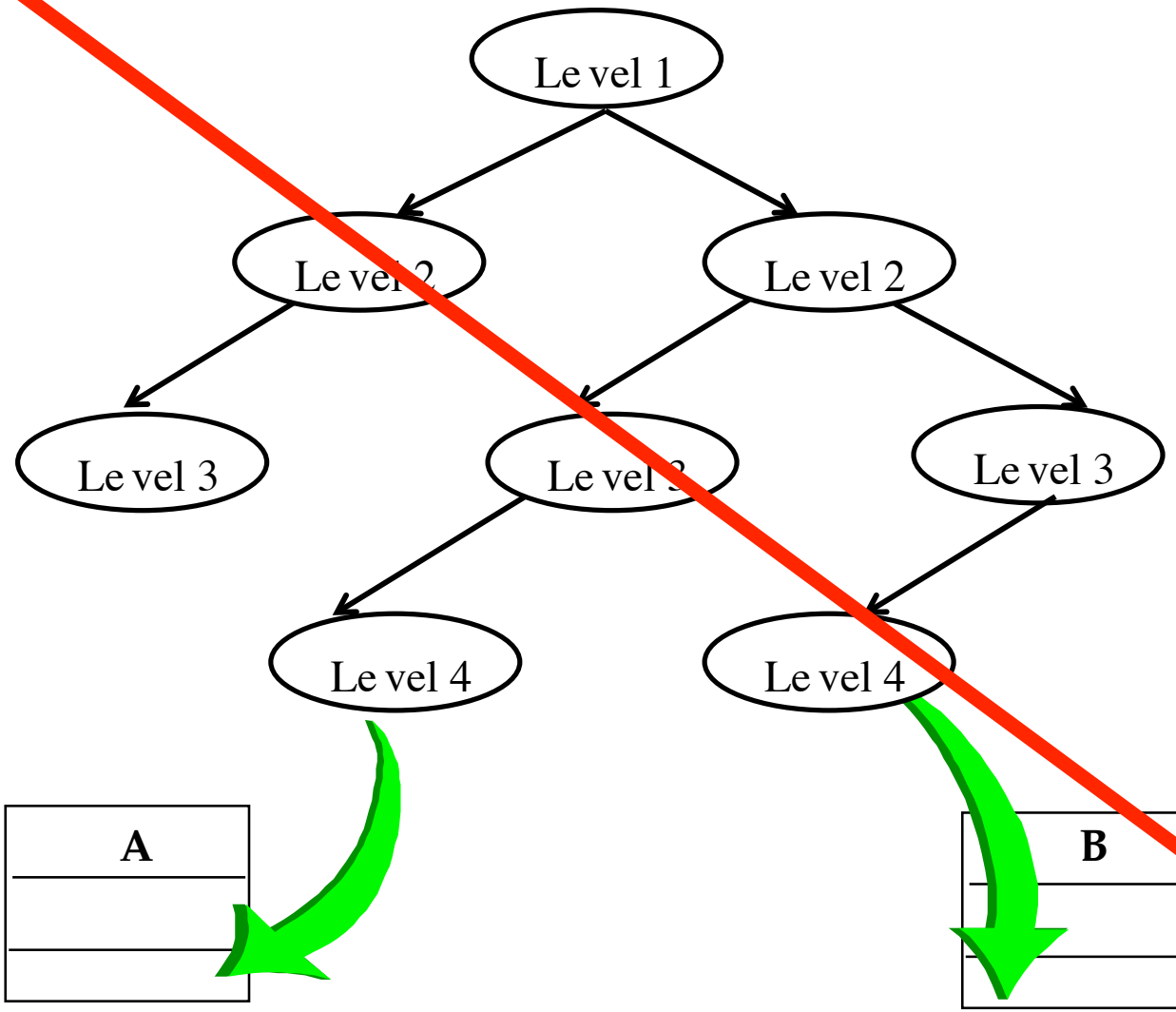
7. The Bank Customer inputs an amount. **[Amount over limit]**

[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.

From Use Cases to Objects



Top Level Use Case

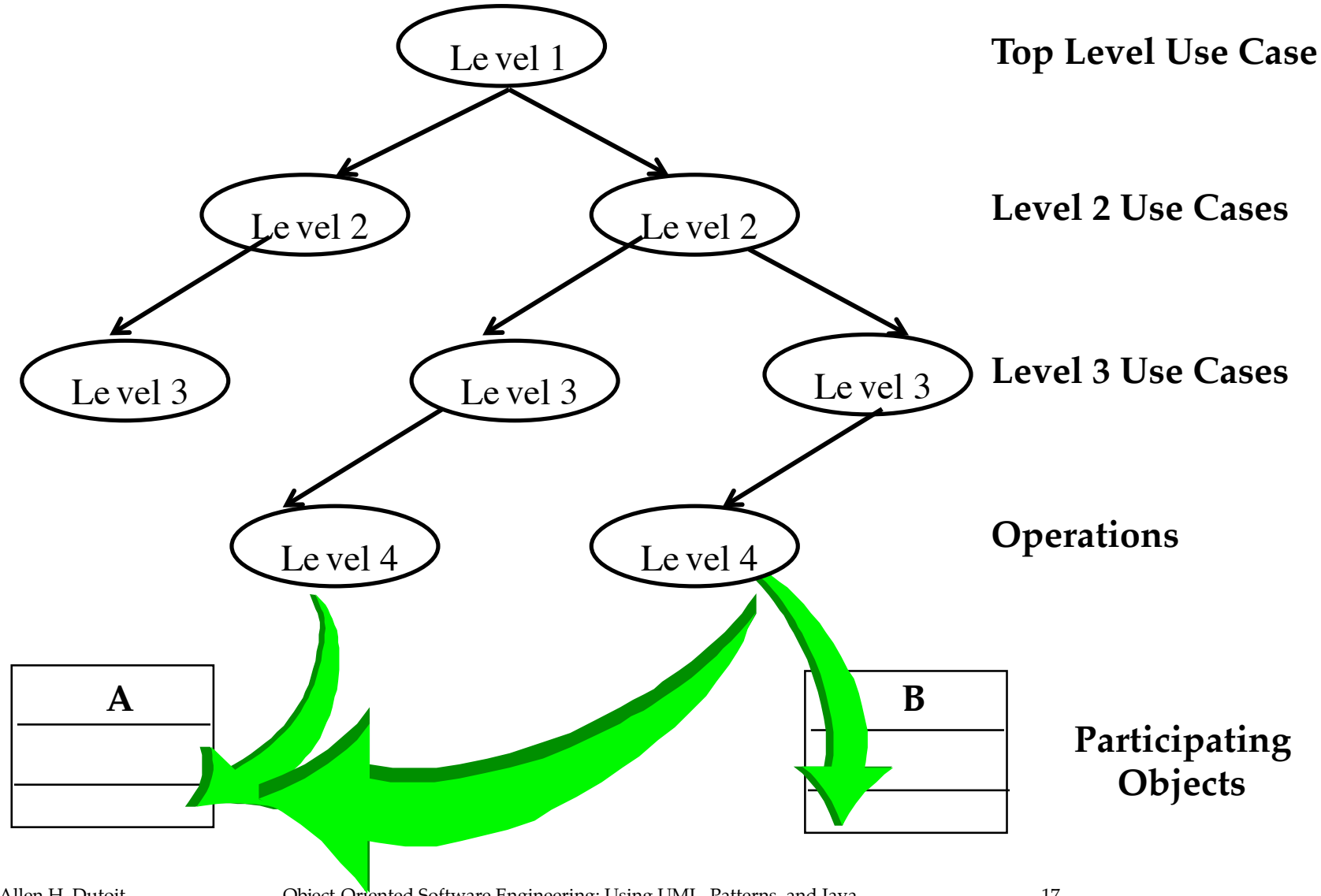
Level 2 Use Cases

Level 3 Use Cases

Operations

A and B are called Participating Objects

Use Cases used by more than one Object



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.

Event Flow: Use Indentation to show the Interaction between Actor and System

1.The Bank Customer inserts 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

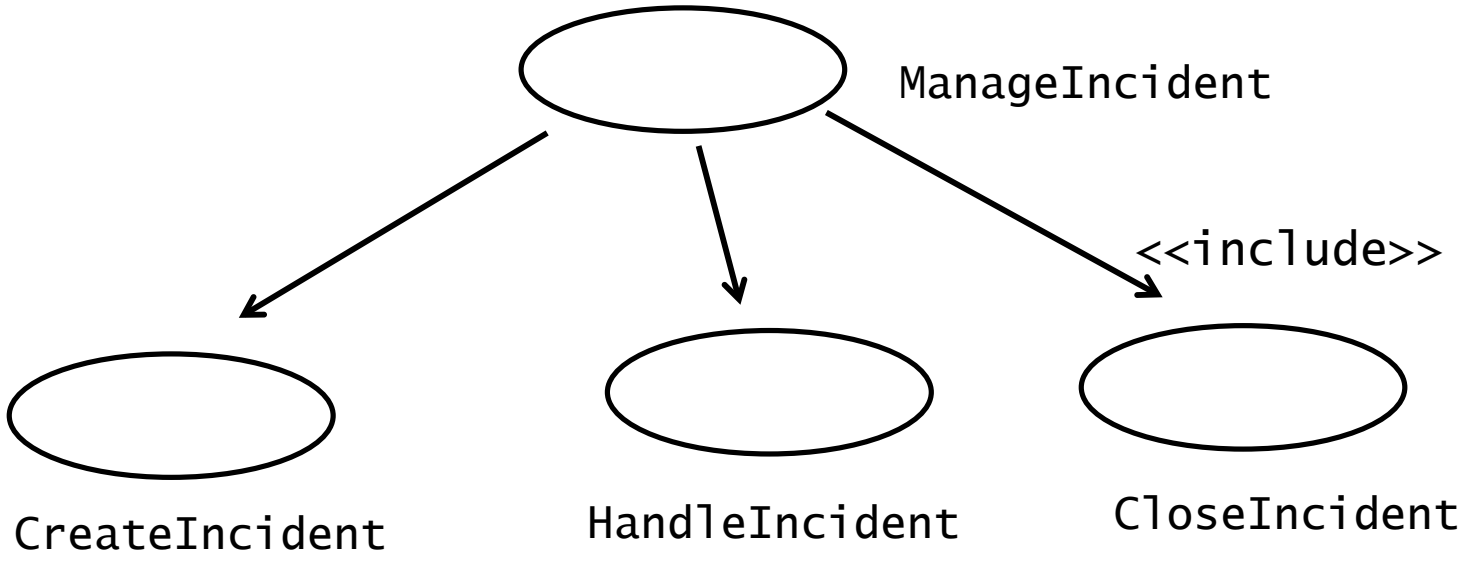
8.The ATM outputs the money and a receipt and stops the interaction.

Use Case 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
  - Includes
  - Extends
  - Generalization

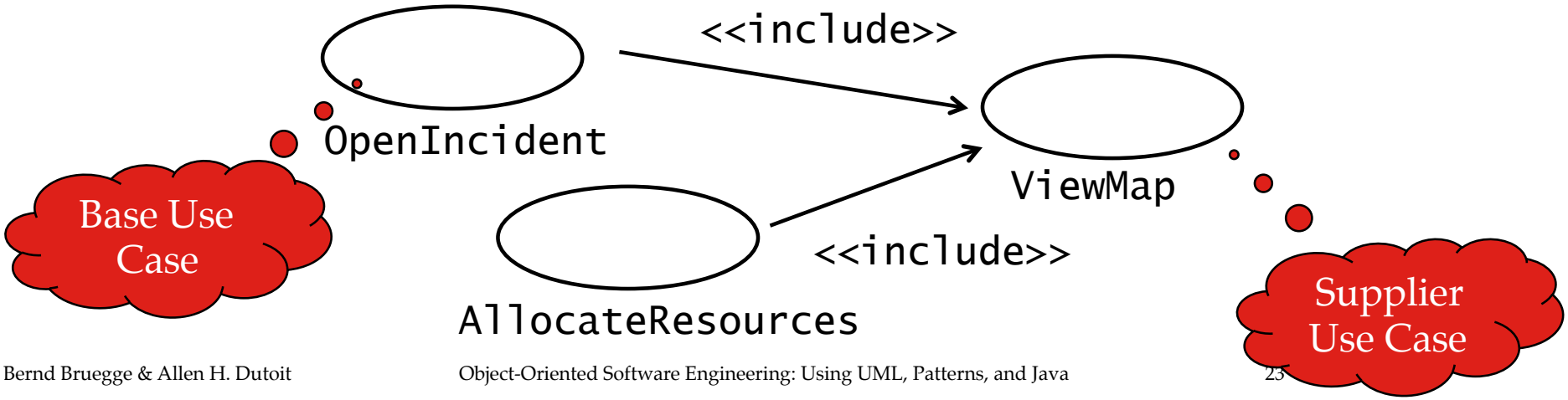
<<include>>: Functional Decomposition

- 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



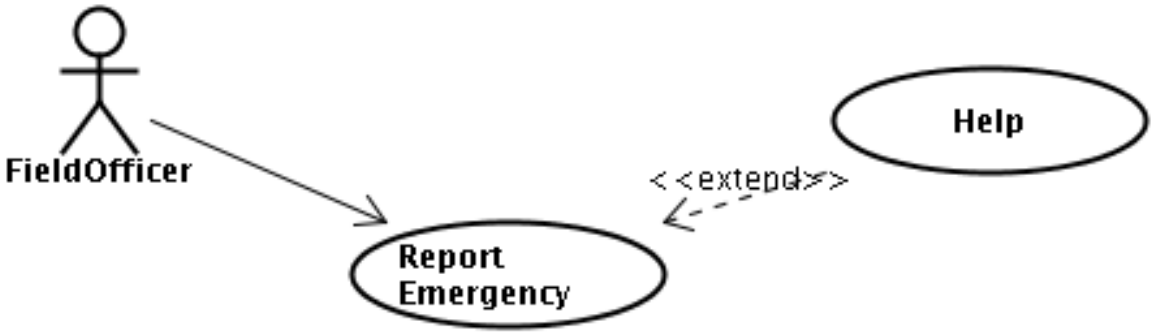
<<include>>: Reuse of Existing Functionality

- 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)



<<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



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"