15-413

Use Case Modeling

Bernd Bruegge
Carnegie Mellon University
School of Computer Science

8 September 1998

Outline of Today's Class

- Use Case Model
- Actors
- Use cases
- Heuristics for finding scenarios and use cases
- Functional vs Object-Oriented Decomposition

Why Use Case Modeling?

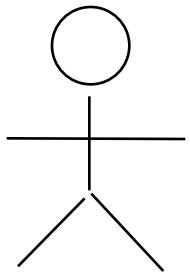
- Utterly comprehensible by the user
  - **Use cases represent the functional requirements of the system (end user functionality, system administration functionality)**
    - **They define every possible event flow through the System**
    - **They provide a description of interaction between objects**
    - **They help to analyze the application domain**
- A great way to start a project
- Use cases can form the basis for the whole development process
  - **Requirements**
  - **System and Object Design**
  - **Implementation**
  - **Test specification**

Use Case Modeling: Key Components

- Actors
- Use Cases
- Relationships between use cases and actors

Actors

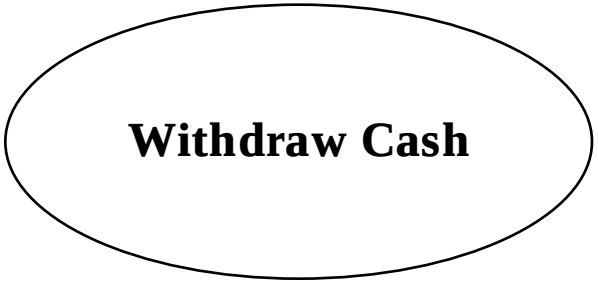
- Actors constitute everything that is external to the system and that communicates and interacts with the system.
  - **human users, external hardware and other systems**
- Actors communicate by sending and receiving stimuli to and from the system. Each actor has a name.
- Graphical Notation: A stick figure with the name of the actor
  - **Bob, Alice and John**



Field Officer

Use Cases

- A use case is a flow of events in the system, including interaction with actors
- It is initiated by an actor
- Each use case has a name
- Each use case has a termination condition
- Graphical Notation: An oval with the name of the use case

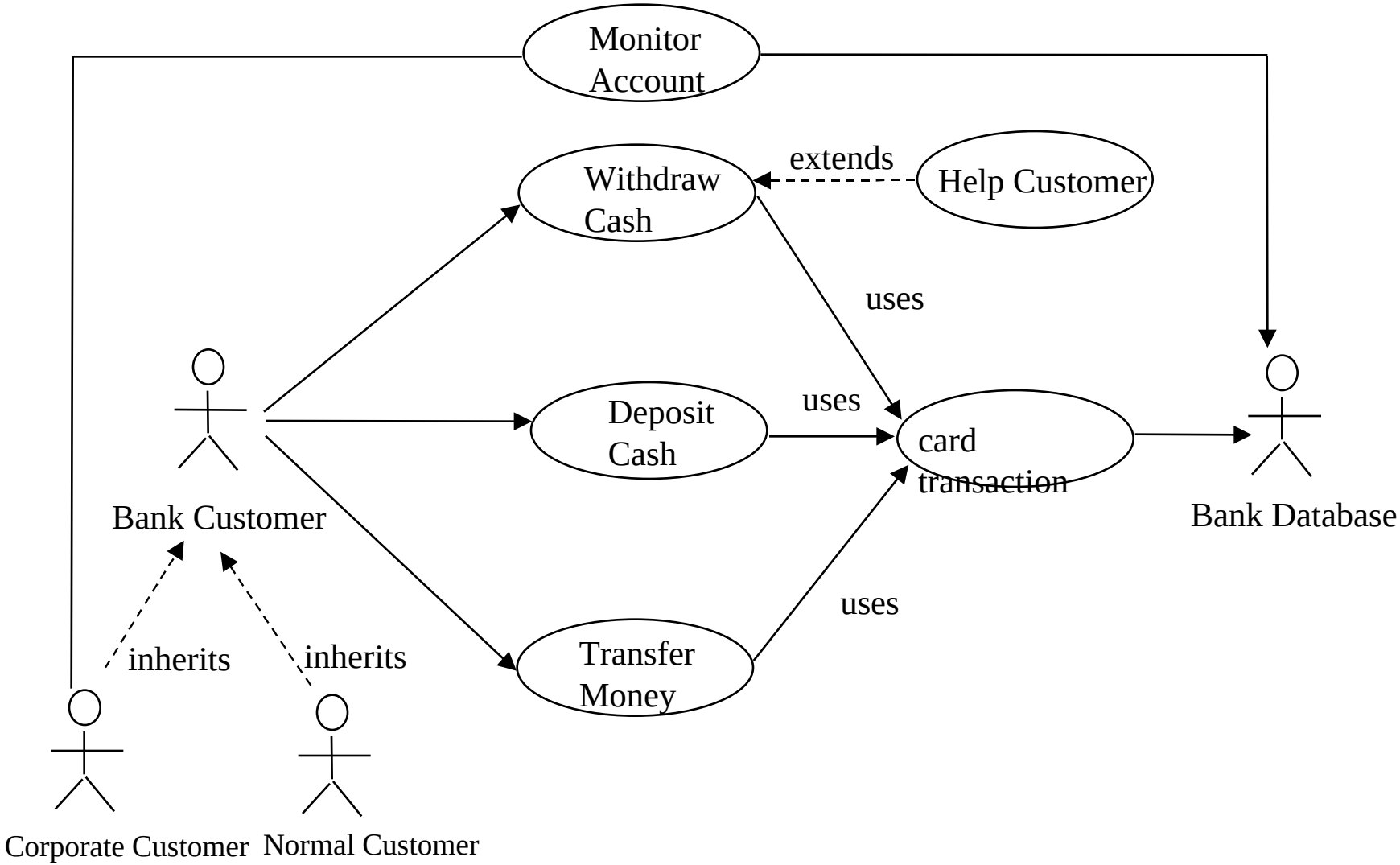


- *Use Case Model:* The set of all use cases specifying the complete functionality of the system

Use Case Model

- A use case model describes the external behavior of the system as seen from an external point of view.
- A use case model consists of a set of use cases and their associations
- Each use case describes a function provided by the system as a set of actions that yield a visible result for an actor

Example: Use Case Model for Bank Transaction System



Use Case Example: Allocate a Resource

- ***From FRIEND (Accident Management System , 15-413 Fall 93)***
- **Actors:**
  - ***Field Supervisor*: This is the official at the emergency site who has been designated as the Field Supervisor by the Dispatcher. This is usually the highest ranking person at the Mobile Command Post on the site.**
  - ***Resource Allocator*: The Resource Allocator is responsible for the commitment and decommitment of the Resources managed by the FRIEND system. The Resource Allocator commits Resources to Emergency Incidents and is responsible for deciding how Resources are utilized when there is more demand than availability of Resources.**
  - ***Dispatcher*: A Dispatcher enters, updates, and removes Emergency Incidents, Actions, and Requests in the system. The Dispatcher also closes Emergency Incidents.**

Use Case Example: Allocate a Resource ctd

- *Entry Condition*
  - **The use case starts after the Resource Allocator has selected an available Resource.**
  - **The Resource is currently not allocated**
- *Flow of Events*
  - **The Resource Allocator selects an Emergency Incident.**
  - **The Resource is committed to the Emergency Incident.**
- *Exit Condition*
  - **The use case terminates when the resource is committed.**
  - **The selected Resource is now unavailable other requests. The Resource remains allocated until deallocated or reallocated.**
- *Special Requirements*
  - **The Field Supervisor is assumed to be responsible for managing the Resources of a site**

Use case vs Scenario

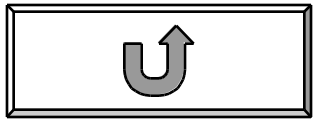
- Distinction between use case and scenario:
  - **A use case is an abstraction because it describes all possible scenarios involving the function described by the use case**
  - **A scenario is an instance describing a set of specific events.**
- Scenario are used as examples illustrating typical cases in which the system is used. **Focus: Understandability**
- Use cases are used to describe all possible cases in which the system is used. **Focus: Completeness.**

How Do I create a Use Case Model?

- Find all the use cases in the scenario that specifies all possible instances of how to report a fire
  - **Example: "Report Emergency " in the first paragraph of the scenario "Warehouse on Fire" is a candidate for a use case**

- Then describe each of these use cases in more detail
  - **Describe the Entry Condition**
  - **Describe the Flow of Events**
  - **Describe the Exit Condition**
  - **Describe Exceptions**
  - **Describe Special Requirements (Constraints, Nonfunctional Requirements)**

Example Scenario: Warehouse on Fire



- Bob, driving down main street in his patrol car notices smoke coming out of a warehouse. His partner, Alice, activates the “Report Emergency” function from her laptop.
- Alice enters the address of the building, a brief description of its location (i.e., north west corner), and an emergency level. In addition to a fire unit, she requests paramedic units on the scene. 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 two paramedic units to the Incident site and sends their estimated arrival time (ETA) to Alice.
- Alice receives the acknowledgment and the ETA.

Heuristics

- First name the use case
  - **Use case name: ReportEmergency**
- Then find the actors
  - **Generalize the concrete names from the scenario ("Bob") to participating actors ("Field officer")**
  - **Participating Actors:**
    - **Field Officer (Bob and Alice in the Scenario)**
    - **Dispatcher (John in the Scenario)**
    -
- Then concentrate on the flow of events
  - **Use informal natural language**

Use Case Example: ReportEmergency Flow of Events

- The **FieldOfficer** activates the “Report Emergency” function of her terminal. FRIEND responds by presenting a form to the officer.
- The FieldOfficer fills the form, by selecting the emergency level, type, location, and brief description of the situation. The FieldOfficer also describes possible responses to the emergency situation. Once the form is completed, the FieldOfficer submits the form, at which point, the **Dispatcher** is notified.
- The Dispatcher reviews the submitted information and creates an Incident in the database by invoking the OpenIncident use case. The Dispatcher selects a response and acknowledges the emergency report.
- The FieldOfficer receives the acknowledgment and the selected response.

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 her terminal and the central is lost.**
  - **The Dispatcher is notified immediately if the connection between any logged in FieldOfficer and the central is lost.**
- 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.**

Another Use Case Example: Allocate a Resource

- ***From FRIEND***
  - **First Responder Interactive Emergency Navigation Database**
  - **15-413 Fall 93**
- ***Actors*:**
  - ***Field Supervisor*: This is the official at the emergency site who is responsible for administering the incident.**
  - ***Resource Allocator*: The Resource Allocator is responsible for the commitment and decommitment of the Resources managed by the FRIEND system.**
  - ***Dispatcher*: A Dispatcher enters, updates, and removes Emergency Incidents, Actions, and Requests in the system. The Dispatcher also closes Emergency Incidents.**

Allocate a Resource, ctd

- *Entry Condition*
  - **The use case starts after the Resource Allocator has selected an available Resource.**
  - **The Resource is currently not allocated**
- *Flow of Events*
  - **The Resource Allocator selects an Emergency Incident.**
  - **The Resource is committed to the Emergency Incident.**
- *Exit Condition*
  - **The use case terminates when the resource is committed.**
  - **The selected Resource is now unavailable to any other Emergency Incidents or Resource Requests.**
- *Special Requirements*
  - **The Field Supervisor is responsible for managing the Resources**

How to Specify a Use Case (Summary)

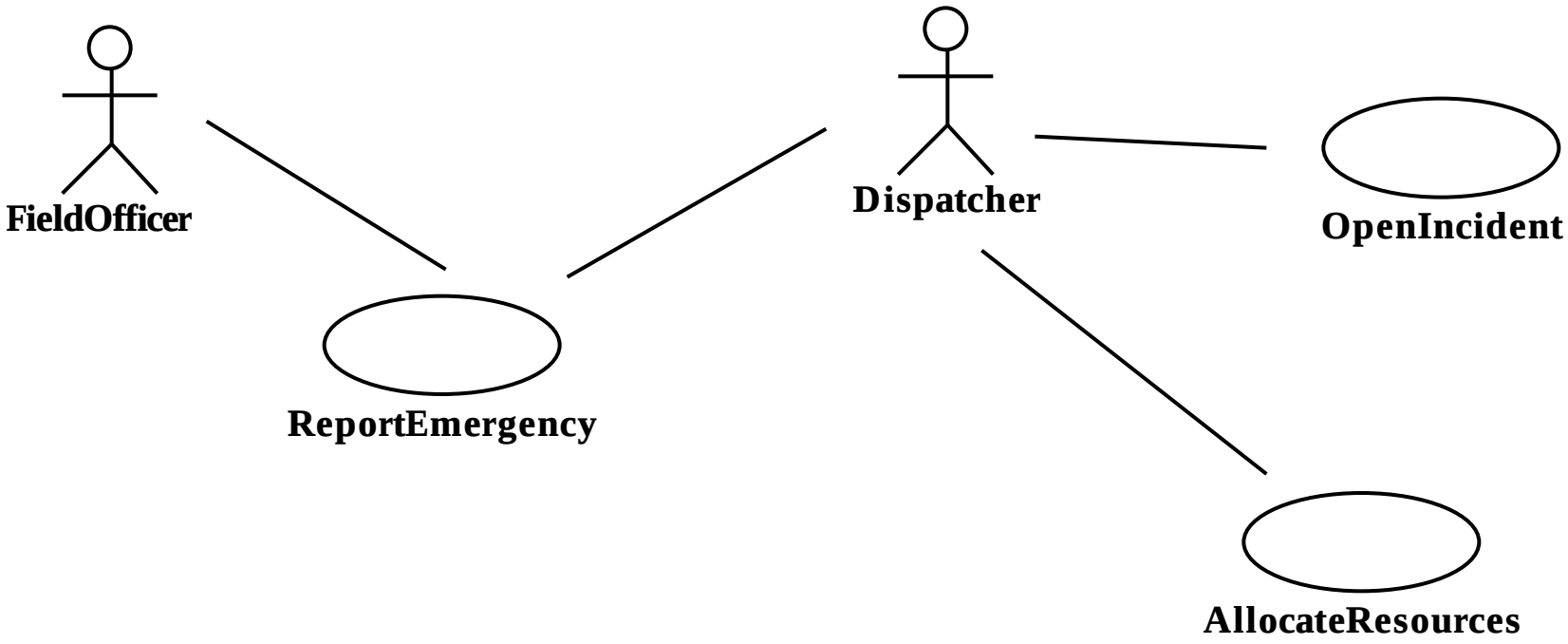
- *Name of Use Case*
- *Actors*
  - **Description of Actors involved in use case**
- *Entry condition*
  - **"This use case starts when…"**
- *Flow of Events*
  - **Free form natural language text describing the flow of events associated with the use case**
- *Exit condition*
  - **"This use cases terminates when…"**
- *Exceptions*
  - **Describe what happens if things go wrong**
- *Special Requirements*
  - **Nonfunctional requirements and constraints that apply to use case**

Use Cases

- A use case is a flow of events in the system, including interaction with actors
- It is initiated by an actor
- 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

Example: Use Case Model for Incident Management

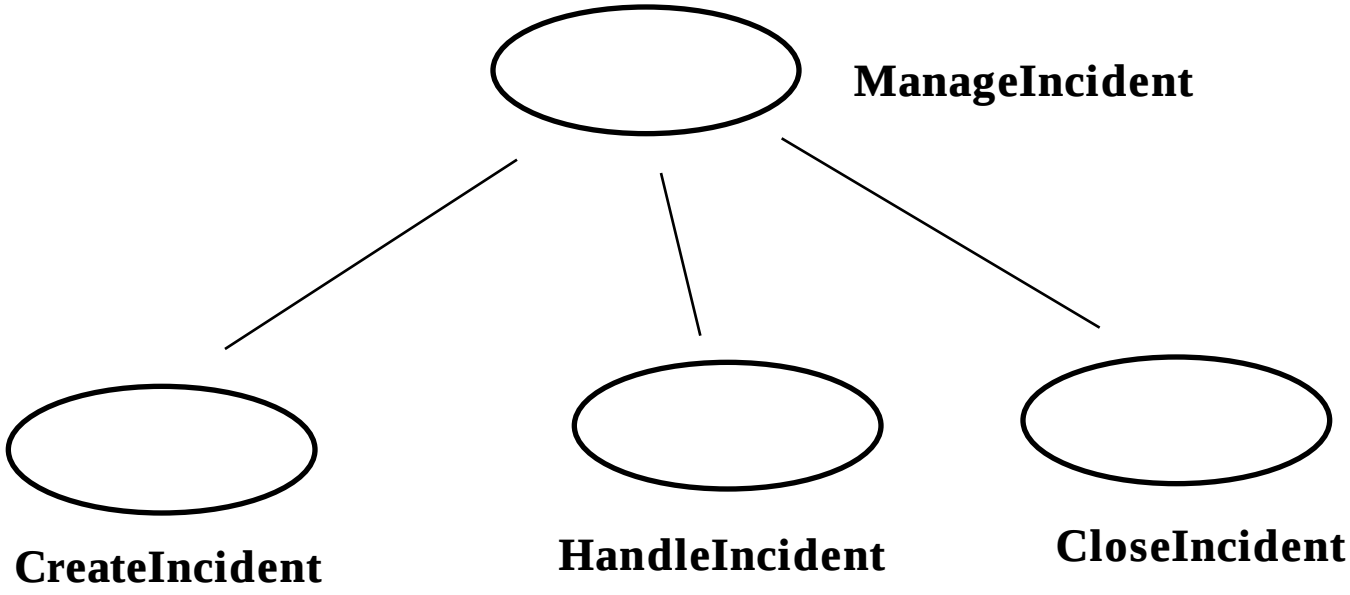


Use Case Associations

- Use Case Association = Relationship between use cases
- Important types:
  - **Consists of**
    - **A use cases consists of other use cases (“functional decomposition”)**
  - **Extends**
    - **A use case extends another use case**
  - **Uses**
    - **A use case uses another use case**

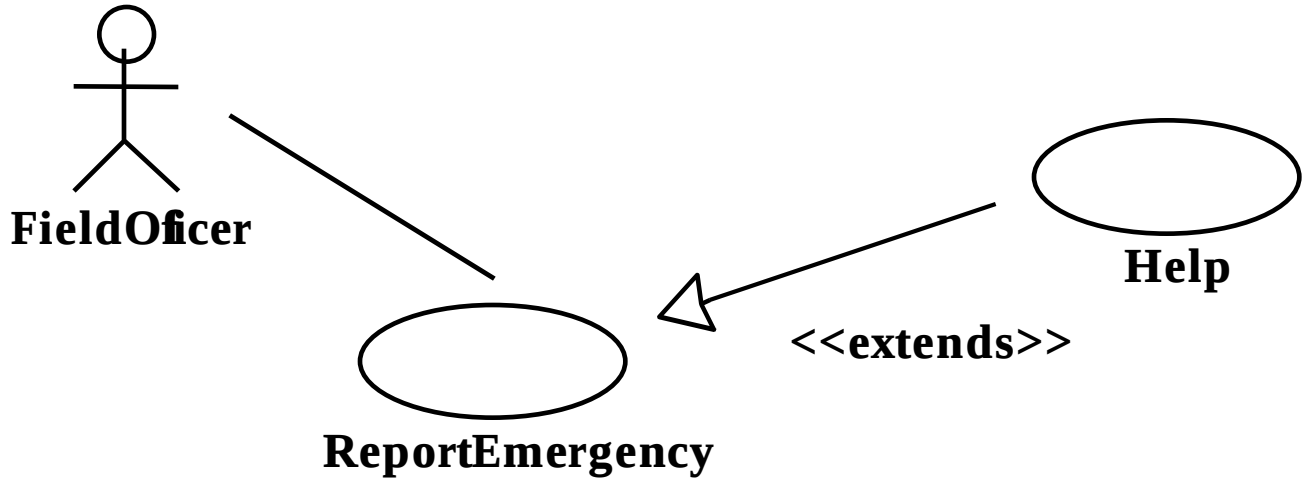
"Consists of" Assocation

- Problem:
  - **A function in the original problem statement is too complex to be solvable**
- Solution:
  - **Describe the function as the aggregation of a set of simpler functions. The associated use case is refined into smaller use cases**



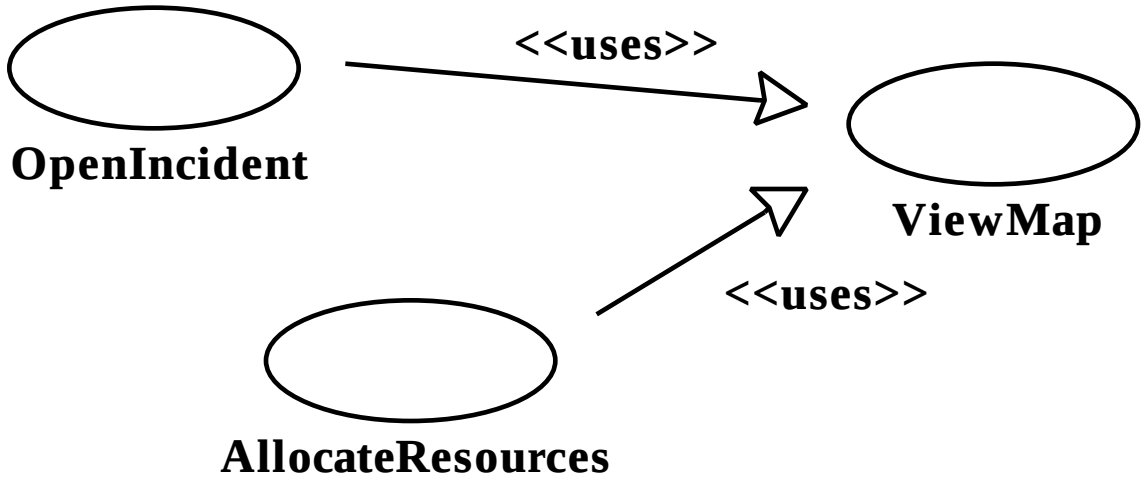
"Extends" Association for Use Cases

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



"Uses" Association for Use Cases

- Problem:
  - **There are already existing functions. How can we *reuse* them?**
- Solution:
  - **The *uses association* from a use case A to a use case B indicates that an instance of the use case A can perform all behavior described for the use case B**
- Example:
  - **The use case "ViewMap" describes behavior that can be used by the use case "OpenIncident" ("ViewMap" is factored out)**



Heuristics: How do I 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 mock-ups as visual support
- Find out what the user does
  - **Task observation (Good)**
  - **Questionnaires (Bad)**

Is there Life after Scenarios and Use Cases?

- Functional Decomposition says no

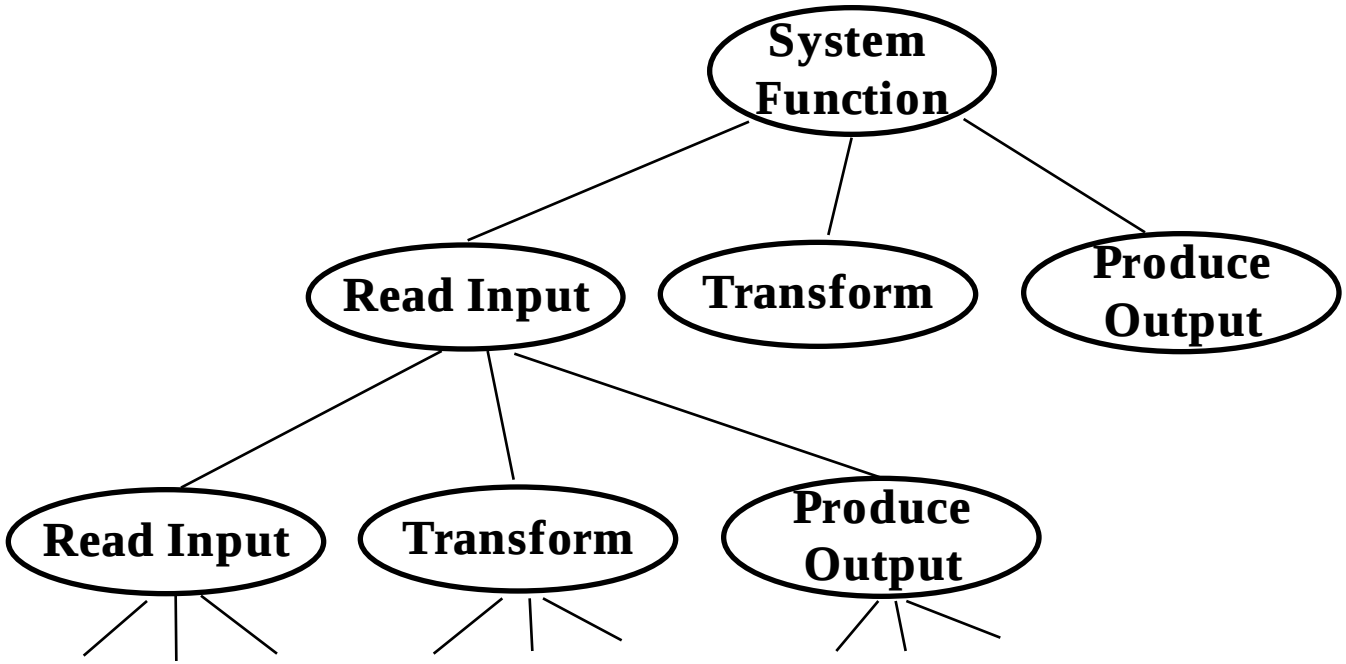
- Every use case is refined into a set of lower level use cases. Either
  - **The use cases consists of lower level use cases**

or

  - **The use case is extended by another use case**

- This refinement is repeatedly done until the lowest level use cases are machine instructions that can be executed by the target machine

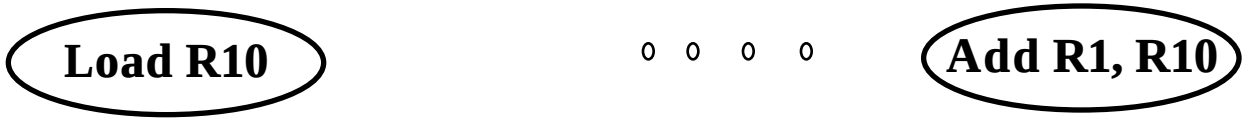
Functional Decomposition



Top Level Use Case

Level 2 Use Cases

Level 3 Use Cases

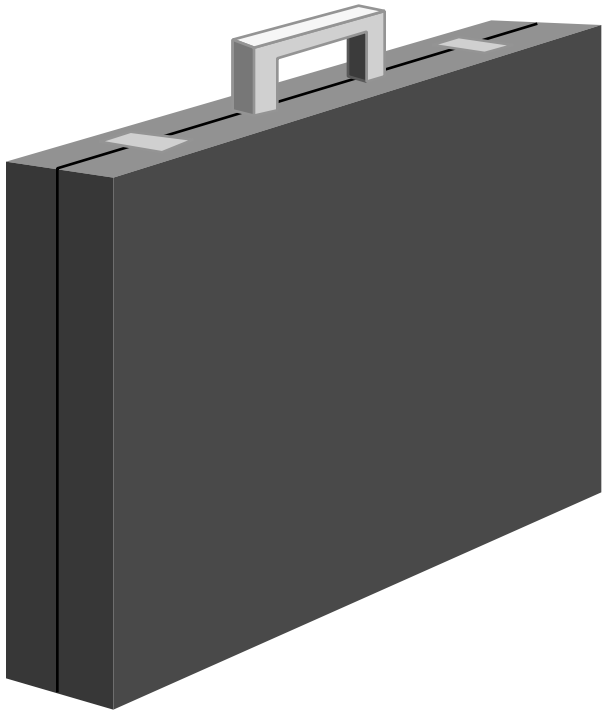


Machine Instructions

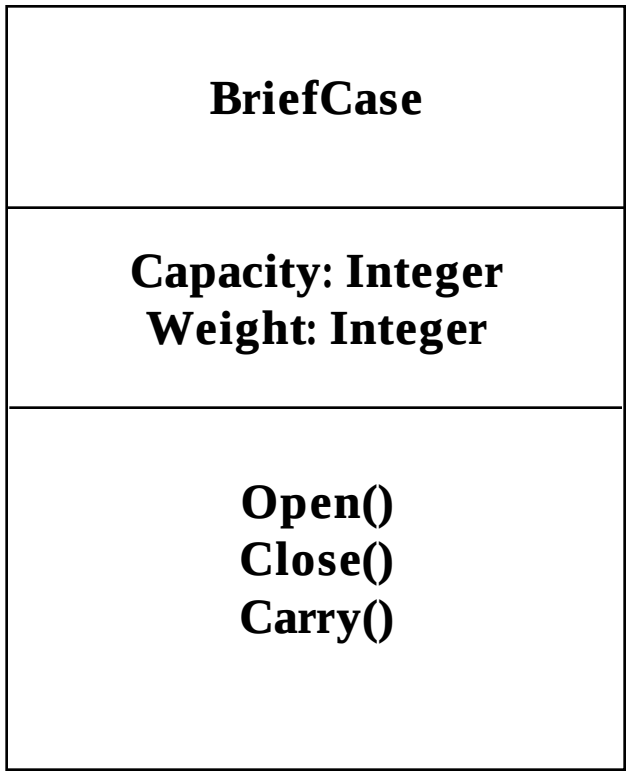
Problems with Functional Decomposition

- High cost of recompilation
  - **Adding a new device usually requires compilation of every file that uses the device**
- Leads to code that is hard to maintain
- The better way is to start with functional decomposition and then to find objects
  - **Object identification**
  - **Sequence Diagrams**

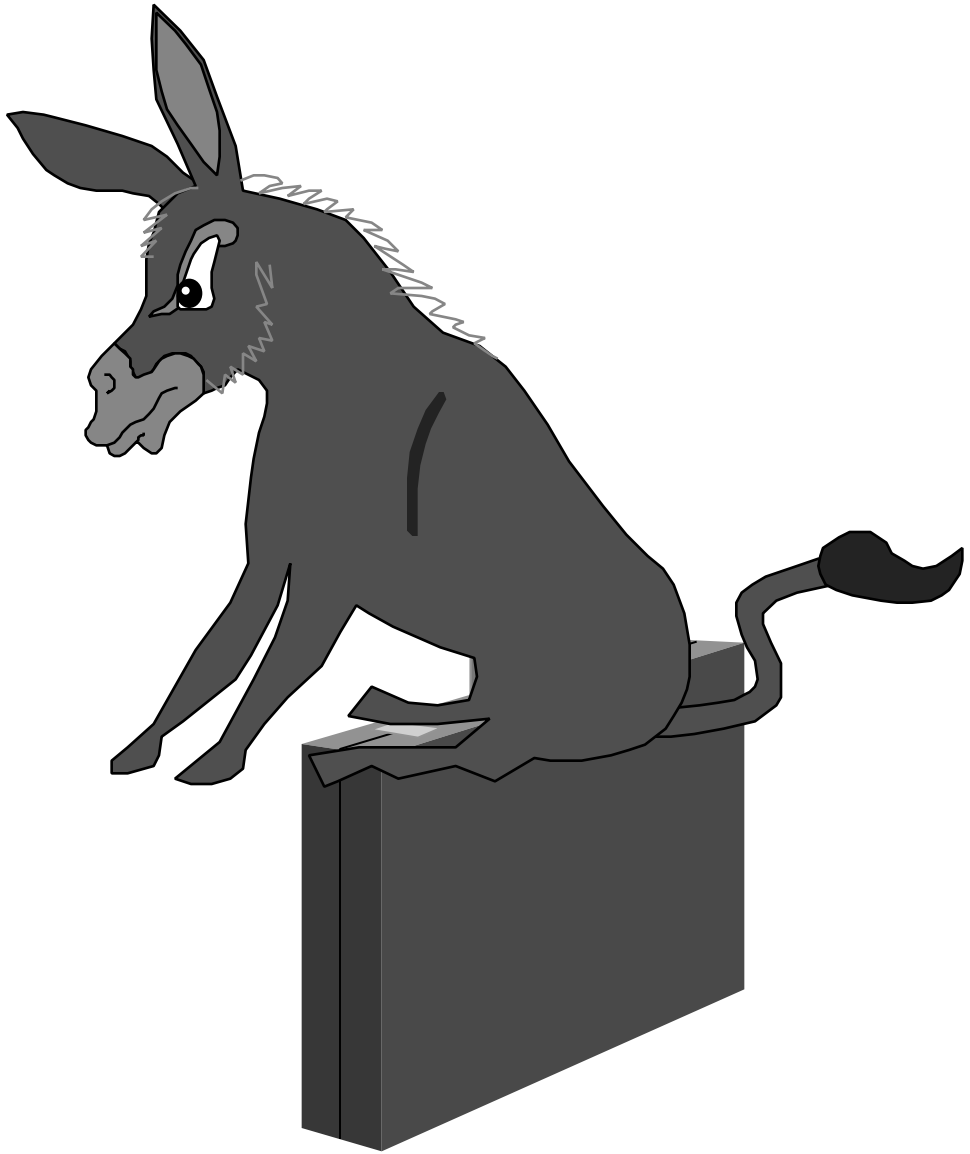
What is this Thing?



Modeling a Briefcase



A new Use Case for a Briefcase



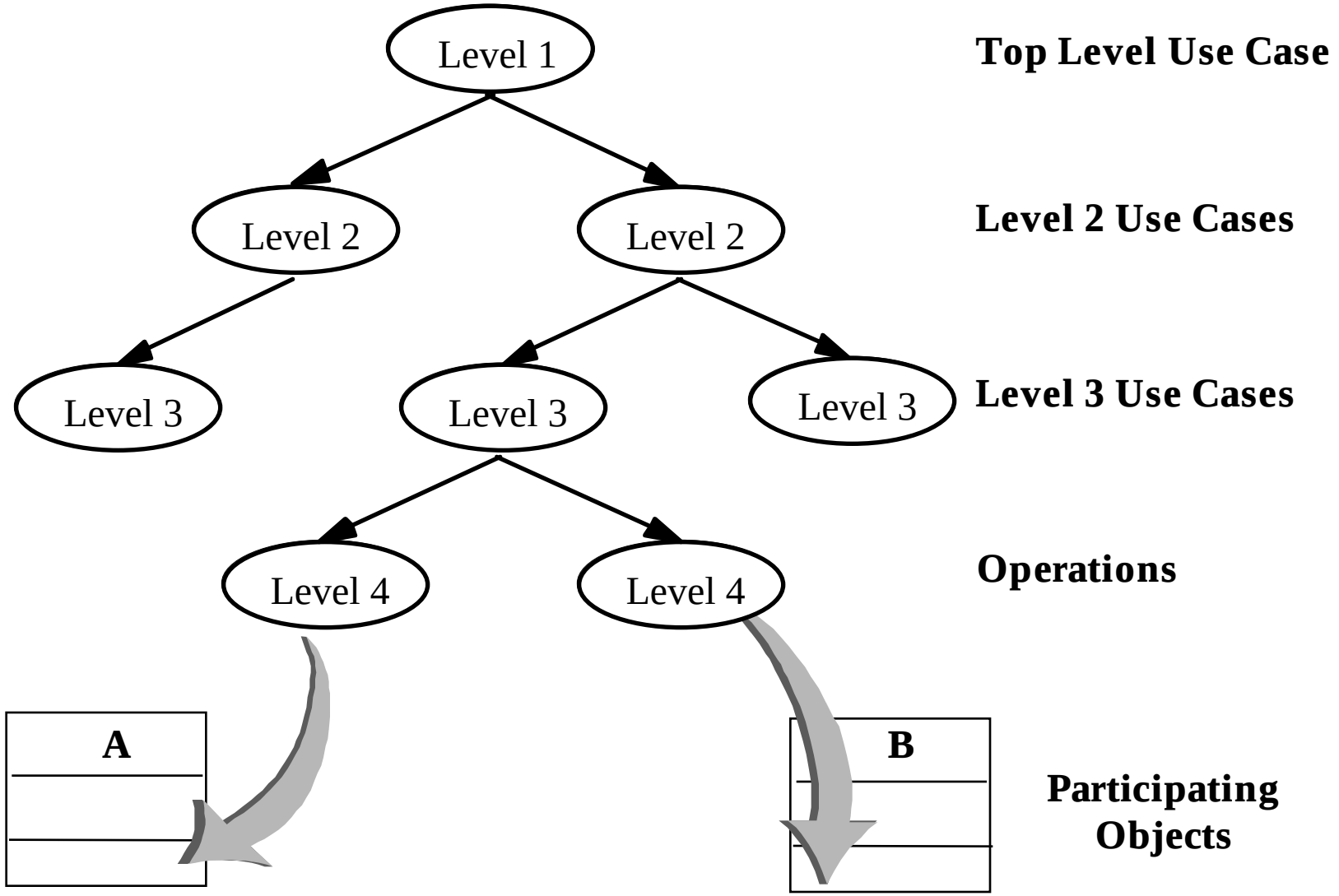
BriefCase
Capacity: Integer Weight: Integer
Open() Close() Carry() SitOnIt()

Questions

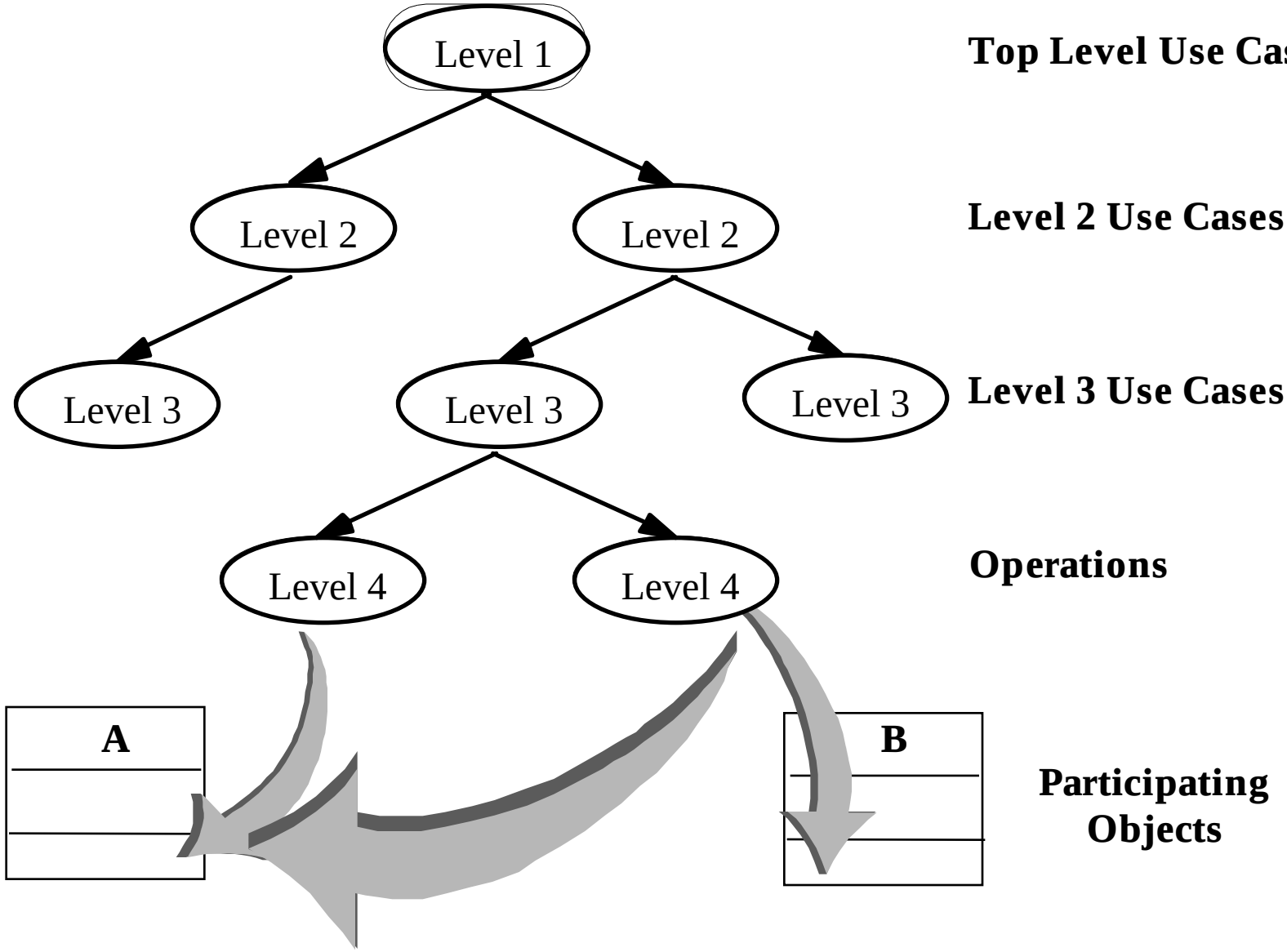
- Why did we model the thing as “Briefcase”?
- Why did we not model it as a chair?
- What do we do if the SitDown() operation is the most frequently used operation?
- The briefcase is only used for sitting on it during video conferences. It is never opened nor closed. Is it a “Chair”or a “Briefcase”?
- How can we live with our mistake?

What is this?

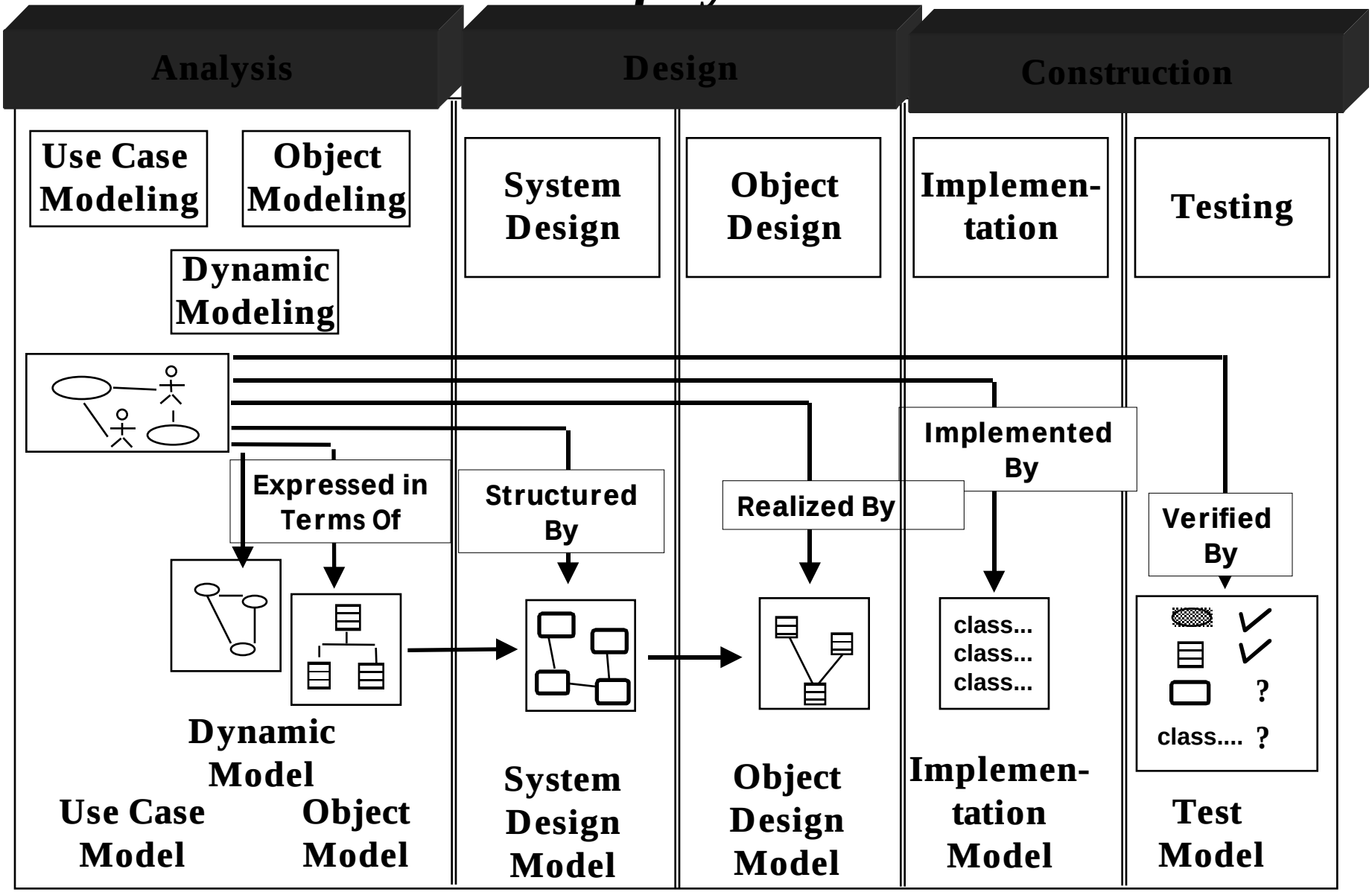
From Use Cases to Objects



Use Cases are used by more than one object



Use Cases and15-413 Lifecycle Activities



Summary

- Scenarios:
  - **Great way to establish communication with client**
  - **As-Is Scenarios, Visionary scenarios, Evalution scenarios Training scenarios**
- Use cases: Abstraction of scenarios
- Pure functional decomposition is bad:
  - **Leads to unmaintainable code**
- Pure object identification is bad:
  - **May lead to wrong objects, wrong attributes, wrong methods**
- The key to successful analysis:
  - **Start with use cases and then find the participating objects**
  - **If somebody asks "What is this?", do not answer right away. Return the question or observe: "What is it used for?"**