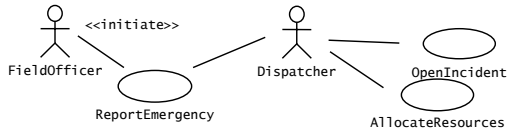


Functional Modeling

- ◆ How can we identify the purpose (functionality) of a system?
 - ◆ Scenarios
 - ◆ Use cases: Abstractions of scenarios



Why Scenarios and Use Cases?

- ◆ Comprehensible by the user
 - ◆ Use cases model a system from the users' point of view (functional requirements)
 - ◆ Define every possible event flow through the system
 - ◆ Description of interaction between objects
- ◆ Great tools to manage a project. Use cases can form basis for whole development process
 - ◆ User manual
 - ◆ System design and object design
 - ◆ Implementation
 - ◆ Test specification
 - ◆ Client acceptance test
- ◆ An excellent basis for incremental & iterative development
- ◆ Use cases have also been proposed for business process reengineering (Ivar Jacobson)

How do we find scenarios?

- ◆ Don't expect the client to be verbal if the system does not exist (greenfield engineering)
- ◆ Don't wait for information even if the system exists
- ◆ Engage in a dialectic approach (evolutionary, incremental)
 - ◆ You help the client to formulate the requirements
 - ◆ The client helps you to understand the requirements
 - ◆ The requirements evolve while the scenarios are being developed

Example: Accident Management System

- ◆ What needs to be done to report a "Cat in a Tree" incident?
- ◆ What do you need to do if a person reports "Warehouse on Fire?"
- ◆ Who is involved in reporting an 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 a simultaneous incident report "Warehouse on Fire?"

Scenario Name: warehouseOnFire

Participating Actors: bob, alice:FieldOfficer, john:Dispatcher

Flow of Events:

1. 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.
2. 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 several paramedic units on the scene given that area appear to be relatively busy. She confirms her input and waits for an acknowledgment.
3. 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.
4. 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

Next goal, after the scenarios are formulated:

- ♦ Find a use case in the scenario that specifies all possible instances of how to report a fire
 - ♦ Example: “Report Emergency” in the first paragraph of the scenario is a candidate for a use case
- ♦ Describe this use case 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 of steps in formulating a use case

- ♦ First name the use case
 - ♦ Use case name: ReportEmergency
- ♦ Then 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)
 - ♦ ?
- ♦ Then concentrate on the flow of events
 - ♦ Use informal natural language

Example of steps in formulating a use case

- ♦ Formulate the Flow of Events:
 - ♦ The FieldOfficer activates the “Report Emergency” function on 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.

Example of steps in formulating a use case

- ♦ Specify the 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.
- ♦ Identify and write down any 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.

See Fig. 4-7

How to Specify a Use Case (Summary)

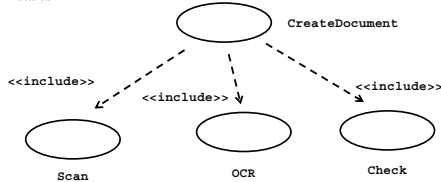
- ♦ Name of Use Case
- ♦ Actors
 - ♦ Description of actors involved in use case
- ♦ Entry condition
 - ♦ Use a syntactic phrase such as “This use case starts when...”
- ♦ Flow of Events
 - ♦ Free form, informal natural language
- ♦ Exit condition
 - ♦ Star with “This use cases terminates when...”
- ♦ Exceptions
 - ♦ Describe what happens if things go wrong
- ♦ Special Requirements
 - ♦ List nonfunctional requirements and constraints

Use Case Associations

- ♦ Use case association is a relationship between use cases
- ♦ Important types:
 - ♦ **Extends**
 - ♦ A use case extends another use case
 - ♦ **Include**
 - ♦ A use case uses another use case (“functional decomposition”)
 - ♦ **Generalization**
 - ♦ An abstract use case has different specializations

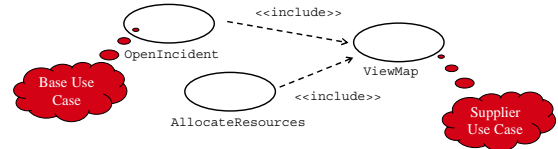
<<Include>>: Functional Decomposition

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

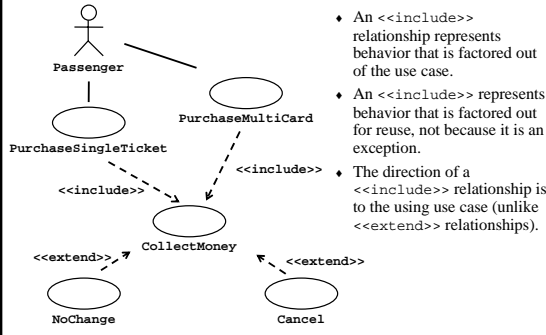


<<Include>>: Reuse of Existing Functionality

- ◆ Problem:
 - There are already existing functions. How can we reuse them?
- ◆ Solution:
 - The include association from a use case A to a use case B indicates that an instance of the use case A performs all the behavior described in the use case B ("A delegates to B")
- ◆ Example:
 - The use case "ViewMap" describes behavior that can be used by the use case "OpenIncident" ("ViewMap" is factored out)
- ◆ Note: The base case cannot exist alone. It is always called with the supplier use case



The <<include>> Relationship



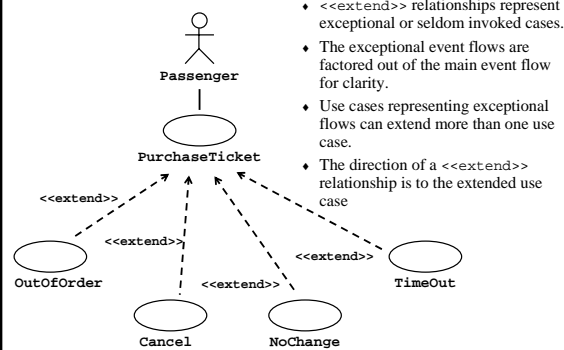
- ◆ An <<include>> relationship represents behavior that is factored out of the use case.
- ◆ An <<include>> represents behavior that is factored out for reuse, not because it is an exception.
- ◆ The direction of a <<include>> relationship is to the using use case (unlike <<extend>> relationships).

<<Extend>> Association for Use Cases

- ◆ Problem:
 - The functionality in the original problem statement needs to be extended.
- ◆ Solution:
 - An extend 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



The <<extend>> Relationship



- ◆ <<extend>> relationships represent exceptional or seldom invoked cases.
- ◆ The exceptional event flows are factored out of the main event flow for clarity.
- ◆ Use cases representing exceptional flows can extend more than one use case.
- ◆ The direction of a <<extend>> relationship is to the extended use case

<p>ReportEmergency (include relationship)</p> <ol style="list-style-type: none"> 1. ... 2. ... 3. The FieldOfficer completes the form by selecting the emergency level, type, location, and local 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. 4. If the connection is still active, the Dispatcher reviews the submitted information and creates an incident in the database by invoking the CreateIncident use case. The Dispatcher submits a response and acknowledges the emergency report. 5. If the connection is broken, the ConnectIncident use case is used. 	<p>ReportEmergency (extend relationship)</p> <ol style="list-style-type: none"> 1. ... 2. ... 3. The FieldOfficer completes the form by selecting the emergency level, type, location, and local 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. 4. The Dispatcher reviews the submitted information and creates an incident in the database by invoking the CreateIncident use case. The Dispatcher submits a response and acknowledges the emergency report. 5. ...
<p>ConnectIncident (include relationship)</p> <ol style="list-style-type: none"> 1. The FieldOfficer and the Dispatcher are advised of the possible reasons why such an status is a "connect". 2. The situation is logged by the system and processed when the connection is reestablished. 3. The FieldOfficer and the Dispatcher enter the ConnectIncident use case through their status and the Dispatcher submits ReportEmergency from the Dispatcher status. 	<p>ConnectIncident (extend relationship)</p> <p>The ConnectIncident use case extends only use cases in which the communication between the FieldOfficer and the Dispatcher can be lost.</p> <ol style="list-style-type: none"> 1. The FieldOfficer and the Dispatcher are notified that the connection is broken. They are advised of the possible reasons why such an event would occur (e.g., "In the FieldOfficer status is a connect"). 2. The situation is logged by the system and processed when the connection is reestablished. 3. The FieldOfficer and the Dispatcher enter the ConnectIncident use case through their status and the Dispatcher submits ReportEmergency from the Dispatcher status.

Figure 4-14 Addition of ConnectIncident exceptional condition to ReportEmergency. An extend relationship is used for exceptional/optional flow of events because it yields a more modular design.

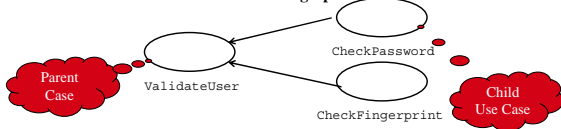
Missing

Missing

Appears Here

Generalization association in use cases

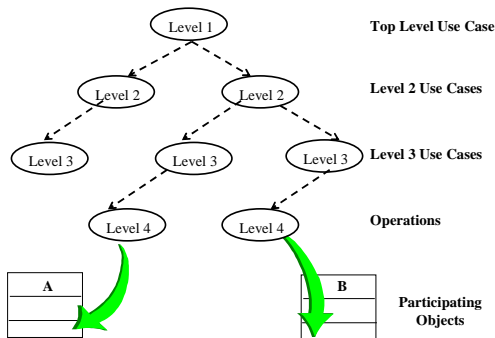
- ◆ Problem:
 - ◆ You have common behavior among use cases and want to factor this out.
- ◆ Solution:
 - ◆ The generalization association among use cases factors out common behavior. The child use cases inherit the behavior and meaning of the parent use case and add or override some behavior. The children use cases share some common flow of events
- ◆ Example:
 - ◆ Consider the use case "ValidateUser", responsible for verifying the identity of the user. The customer might require two realizations: "CheckPassword" and "CheckFingerprint"



How do I find use cases?

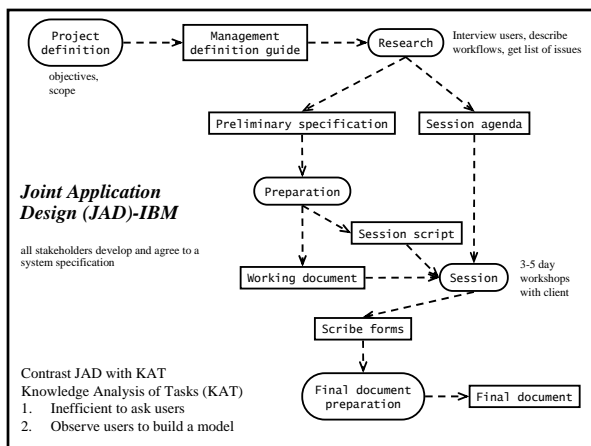
- ◆ Select a narrow vertical slice of the system (i.e. one scenario)
 - ◆ Discuss it in detail with the user to understand the user's preferred style of interaction
- ◆ Select a horizontal slice (i.e. many scenarios) to define the scope of the system.
 - ◆ Discuss the scope with the user
- ◆ Use mock-ups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

From Use Cases to Objects



Finding Participating Objects in Use Cases

- ◆ For any use case do the following:
 - ◆ Find terms that developers or users need to clarify in order to understand the flow of events
 - ◆ Always start with the user's terms, then negotiate:
 - FieldOfficerStationBoundary or FieldOfficerStation?
 - IncidentBoundary or IncidentForm?
 - EOPControl or EOP?
 - ◆ Identify real world entities that the system needs to track.
 - ◆ Examples: FieldOfficer, Dispatcher, Resource
 - ◆ Identify real world procedures that the system needs track.
 - ◆ Example: EmergencyOperationsPlan
 - ◆ Identify data sources or sinks. Example: Printer
 - ◆ Identify interface artifacts. Example: PoliceStation
 - ◆ Do textual analysis to find additional objects
 - ◆ Model the flow of events with a sequence diagram



Requirements Analysis Document	
1. Introduction	
1.1 Purpose of the system	
1.2 Scope of the system	
1.3 Objectives and success criteria of the project	
1.4 Definitions, acronyms, and abbreviations	
1.5 References	
1.6 Overview	
2. Current system	
3. Proposed system	
3.1 Overview	
3.2 Functional requirements	
3.3 Nonfunctional requirements	
3.3.1 Usability	
3.3.2 Reliability	
3.3.3 Performance	
3.3.4 Supportability	
3.3.5 Implementation	
3.3.6 Interface	
3.3.7 Packaging	
3.3.8 Legal	
3.4 System models	
3.4.1 Scenarios	
3.4.2 Use case model	
3.4.3 Object model	
3.4.4 Dynamic model	
3.4.5 User interface—navigational paths and screens mock-ups	
4. Glossary	

Figure 4-16 Outline of the Requirements Analysis Document (RAD). Sections in *italics* are completed during analysis (see next chapter).

◆RAD serves as a contract between client and developers

◆Describes the functional and nonfunctional requirements

◆RAD will be required as part of your final term project

Summary

In these slides, we reviewed the requirements elicitation activities aimed at defining the boundary of the system:

- ◆ Scenario identification
- ◆ Use case identification and refinement
- ◆ Identification of participating objects

Requirements elicitation builds a functional model of the system which will then be used during analysis to build an object model and a dynamic model.