Modeling with UML
Chapter 2, part 1

CS 4354
Summer II 2015

Jill Seaman

What is UML?

- Unified Modeling Language
- UML is a graphical notation to articulate (and communicate) complex ideas in Object-Oriented software development
- UML resulted from a unification of many existing notations.
- The goal of UML is to provide a standard notation that can be used by all object-oriented development methods (software processes)
- UML includes:
  - ✦Use Cases and Use Case Diagrams
  - ✦Class Diagrams
  - ✦State Diagrams
  - ✦Sequence Diagrams
  - ✦Activity Diagrams

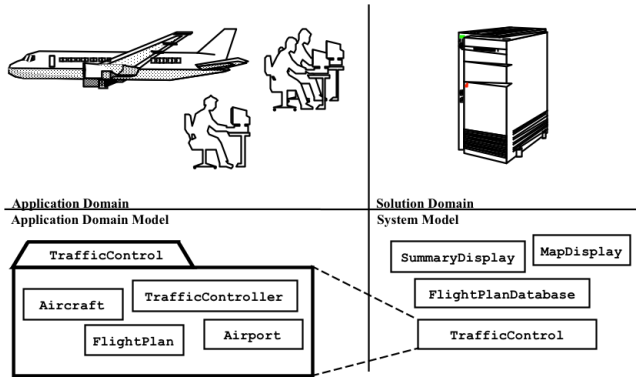
What is a Model?

- A Model is
  - ✦ a means for dealing with complexity.
  - ✦ an abstract description of a system that focuses on interesting aspects and ignores irrelevant details
  - ✦ an abstraction describing a subset of a system.
  - ✦ NOT a diagram or notation

Object-oriented modeling

- The application domain is all aspects of the customer's "problem".
  - ✦ physical environment
  - ✦ users and other people
  - ✦ their work processes, etc.
  - ✦**Object-oriented analysis** models the application domain
- The solution domain is the modeling space of all possible solutions.
  - ✦ represents system design and object design
  - ✦ richer and more volatile than application domain, more detail.
  - ✦ **Object-oriented design** models the solution domain
- Both use the same representation (classes, associations, etc)

Application domain and Solution Domain

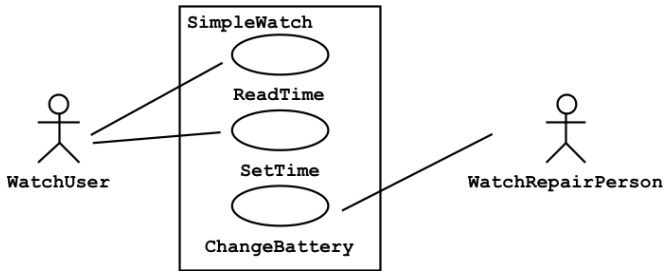


Note the System Model contains the Application domain model.

Use Case Diagrams

- Use Cases represent the functionality of the system from an external point of view.
- An Actor is an external entity that interacts with the system.
  - ✦ different kinds of users (roles), other systems, etc.
- A Use case is a textual description of the behavior of the system from an actor's point of view.
  - ✦ overview of one user/system interaction
  - ✦ focused on one goal of an actor
  - ✦ described as a set of events.
  - ✦ yields a visible/observable result for the actor

Use case diagram for a simple watch



- Actors are stick figures
- Use cases are ovals, with name usually inside the oval
- The boundary of the system is the box
- Each oval must have a textual description of the use case!!

Use Case: textual descriptions

- The textbook uses a template with six fields
  - ✦ **Name**: unique in the system model, usually a verb-noun phrase
  - ✦ **Participating actors**: actors interacting with use case
  - ✦ **Entry conditions**: must be true to initiate use case
  - ✦ **Flow of events**: describes the sequence of interactions, numbered for reference. States what user does and what the system does.
  - ✦ **Exit conditions**: will be true after use case is complete
  - ✦ **Quality requirements**: not related to functionality (constraints on performance, etc.)
- These fields are not "standard", may see many variations.

Use case textual description for Report Emergency

Use Case Name	Report Emergency
Participating Actors	Initiated by FieldOfficer, Communicates with Dispatcher
Flow of Events	1. The FieldOfficer Activates the "Report Emergency" function of her terminal. 2. The System responds by presenting a form to the FieldOfficer. 3. The FieldOfficer fills out 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. When complete, the FieldOfficer submits the form. 4. The System receives the form and notifies the Dispatcher. 5. 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 report. 6. The System displays the acknowledgment and the selected response to the FieldOfficer.
Entry Condition	The FieldOfficer is logged into the system.
Exit Condition	The FieldOfficer has received an acknowledgement and the selected response from the Dispatcher
Quality 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.

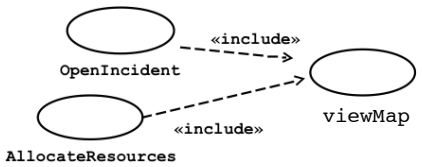
Use Case Diagrams:

- 3 kinds of relationships (to represent "reuse"/avoid copying)
  - **Inclusion**
  - **Inheritance/Generalization**
  - **Extension**

Include relationship

- Used when multiple use cases share a common step or event
  - Like a function call.
- For example, assume the Dispatcher can at any time press a key to access a street map:
  - Make a new use case called "ViewMap" (with its own description)
  - In the OpenIncident and AllocateResources use cases, the flow of events will specify when the ViewMap use case can take place.

Example include relationship use case diagram

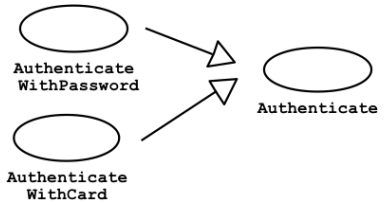


- Dashed arrow points to the included use case
- Arrow is labeled with <<include>>

Inheritance/Generalization relationship

- Used when one use case is a specialization of another
  - ✦It adds more detail
- For example, assume FieldOfficers are required to authenticate before they can use Friend:
  - ✦ Early stages of modeling has one simple use case called Authenticate
  - ✦ Later stages add two specific ways of authenticating, called AuthenticateWithPassword and AuthenticateWithCard
- Textual description of the specialized use case:
  - ✦ Inherits all of the parts of the base use case, except as specified in the derived use case.
  - ✦ For example, only the Flow of Events are different from the base use case.

Example inheritance relationship use case diagram

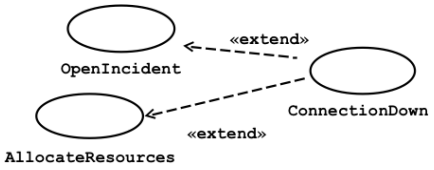


- Arrow with open arrowhead points from derived use case to the base use case

Extend relationship

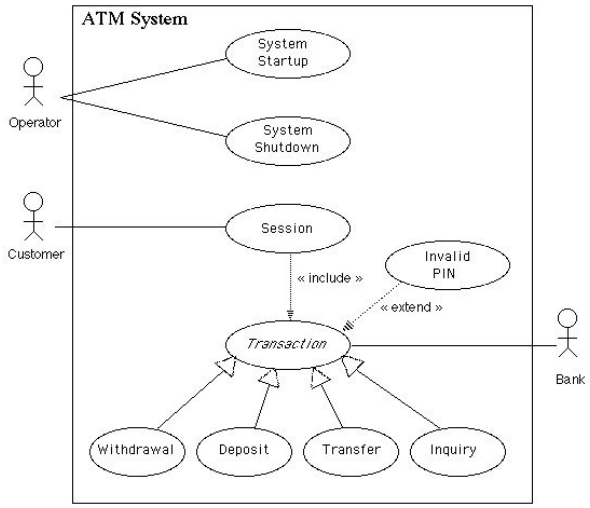
- Used and described differently in different sources
- Used to add events or details to an existing use case, especially exceptional behavior
- M. Fowler says the “base” use case must declare certain “extension points”, like certain variables or parameters in the base use case.
- The extending use case provides values or detail for these extension points.
- An extending use case can extend more than one base case, as long as the base cases have the same extension points.
  - ✦The base use case requires no “knowledge” of its extending use cases.

Example extend relationship use case diagram



- Dashed arrow points from the extended use case to the base use case
- Arrow is labeled with <<extend>>
- Extension points can be listed in the base use case oval.

A larger sample use case diagram



How could the names of the use cases be improved?

When and how to use Use Case (Diagram)s

- Use during requirements analysis to capture the requirements.
  - ✦ Do it in combination with modeling the application domain concepts (this helps explain and uncover new use cases).
- Remember: Use cases represent an external view of the system (don't expect them to appear as classes in the system).
- With Use cases, the textual descriptions are more important than the Use case diagrams.
  - ✦ The use case textual descriptions contain the details necessary for design and implementation.
  - ✦ The diagram is just a good summary of the use cases.