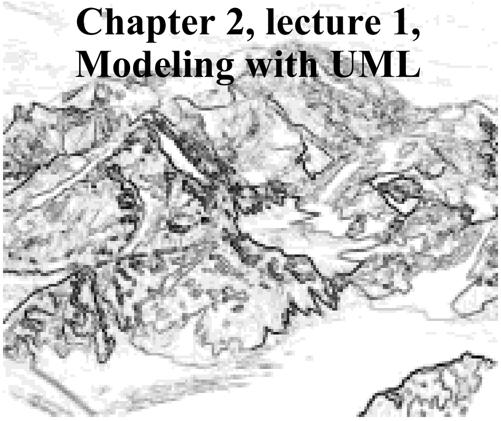
Object-Oriented Software Engineering
Using UML, Patterns, and Java

Chapter 2, lecture 1, Modeling with UML



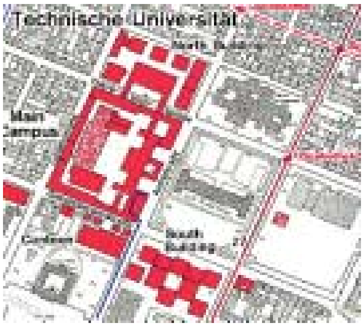
Overview: modeling with UML

- What is modeling?
- What is UML?
- Use case diagrams
- Class diagrams
- Sequence diagrams

What is modeling?

- Modeling consists of building an abstraction of reality.
- Abstractions are simplifications because:
  - **They ignore irrelevant details and**
  - **They only represent the relevant details.**
- What is *relevant* or *irrelevant* depends on the purpose of the model.

Example: street map



Why model software?

Why model software?

- Software is getting increasingly more complex
  - **Windows XP > 40 mio lines of code**
  - **A single programmer cannot manage this amount of code in its entirety.**
- Code is not easily understandable by developers who did not write it
- We need simpler representations for complex systems
  - **Modeling is a mean for dealing with complexity**

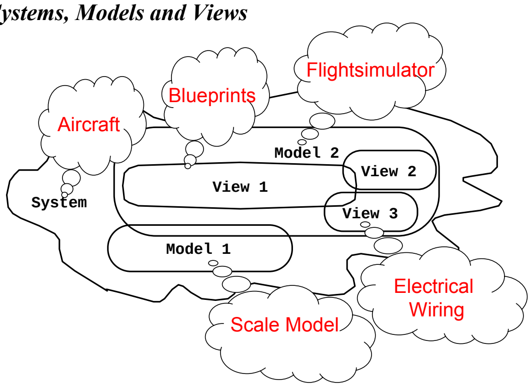
Systems, Models and Views

- A ***model*** is an abstraction describing a subset of a system
- A ***view*** depicts selected aspects of a model
- A ***notation*** is a set of graphical or textual rules for depicting views
- Views and models of a single system may overlap each other

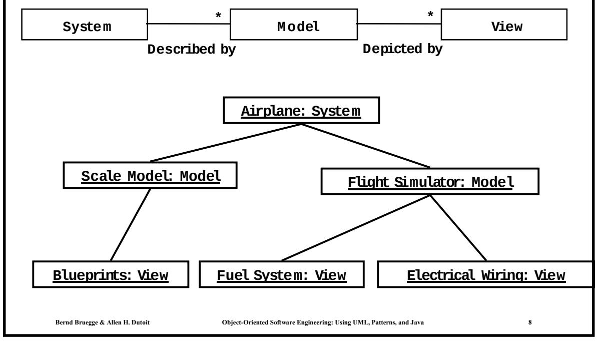
Examples:

- System: Aircraft
- Models: Flight simulator, scale model
- Views: All blueprints, electrical wiring, fuel system

Systems, Models and Views



Models, Views and Systems (UML)



Concepts and Phenomena

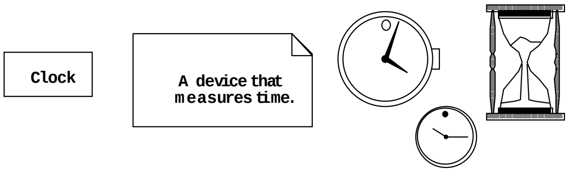
Phenomenon
- **An object in the world of a domain as you perceive it**
- ***Example:* The lecture you are attending**
- ***Example:* My black watch**

Concept
- **Describes the properties of phenomena that are common.**
- ***Example:* Lectures on software engineering**
- ***Example:* Black watches**

Concept is a 3-tuple:
- **Name (To distinguish it from other concepts)**
- **Purpose (Properties that determine if a phenomenon is a member of a concept)**
- **Members (The set of phenomena which are part of the concept)**

Concepts and phenomena

Name **Purpose** **Members**



- Abstraction
  - **Classification of phenomena into concepts**
- Modeling
  - **Development of abstractions to answer specific questions about a set of phenomena while ignoring irrelevant details.**

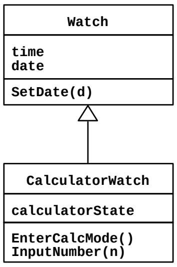
Concepts in software: Type and Instance

- Type:
  - **An abstraction in the context of programming languages**
  - **Name: int, Purpose: integral number, Members: `0, -1, 1, 2, -2, . . .`**
- Instance:
  - **Member of a specific type**
- The type of a variable represents all possible instances the variable can take

The following relationships are similar:
- **"type" <–> "instance"**
- **"concept" <–> "phenomenon"**

Abstract Data Types & Classes

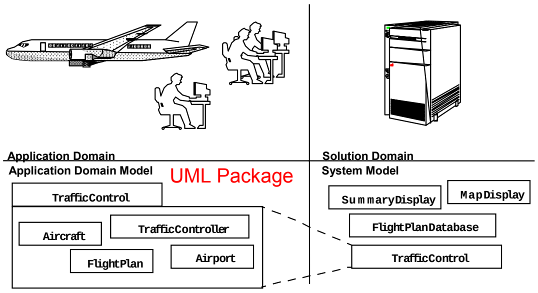
- Abstract data type
  - **Special type whose implementation is hidden from the rest of the system.**
- Class:
  - **An abstraction in the context of object-oriented languages**
- Like an abstract data type, a class encapsulates both state (variables) and behavior (methods)
  - **Class Vector**
- Unlike abstract data types, classes can be defined in terms of other classes using inheritance



Application and Solution Domain

- Application Domain (Requirements Analysis):
  - **The environment in which the system is operating**
- Solution Domain (System Design, Object Design):
  - **The available technologies to build the system**

Object-oriented modeling



What is UML?

- UML (Unified Modeling Language)
  - **An emerging standard for modeling object-oriented software.**
  - **Resulted from the convergence of notations from three leading object-oriented methods:**
    - **OMT (James Rumbaugh)**
    - **OOSE (Ivar Jacobson)**
    - **Booch (Grady Booch)**
- Reference: “The Unified Modeling Language User Guide”, Addison Wesley, 1999.
- Supported by several CASE tools
  - **Rational ROSE**
  - **TogetherJ**

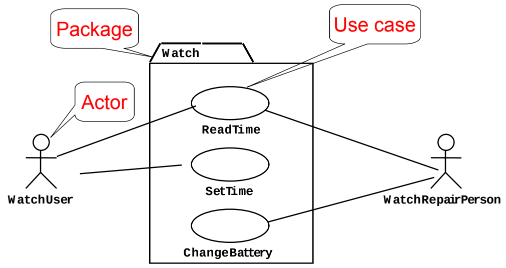
UML: First Pass

- You can model 80% of most problems by using about 20 % UML
- We teach you those 20%

UML First Pass

- Use case Diagrams
  - **Describe the functional behavior of the system as seen by the user.**
- Class diagrams
  - **Describe the static structure of the system: Objects, Attributes, Associations**
- Sequence diagrams
  - **Describe the dynamic behavior between actors and the system and between objects of the system**
- Statechart diagrams
  - **Describe the dynamic behavior of an individual object (essentially a finite state automaton)**
- Activity Diagrams
  - **Model the dynamic behavior of a system, in particular the workflow (essentially a flowchart)**

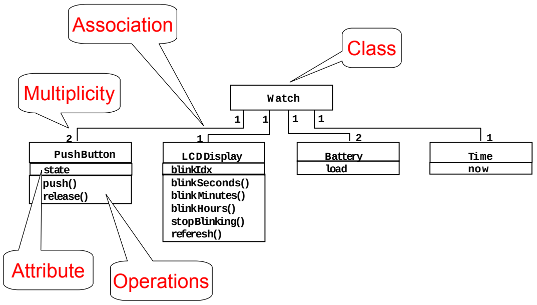
UML first pass: Use case diagrams



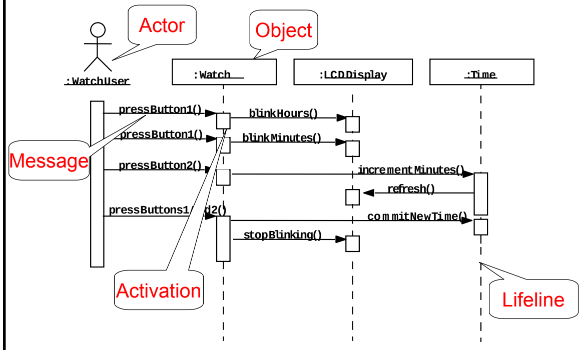
Use case diagrams represent the functionality of the system from user’s point of view

UML first pass: Class diagrams

Class diagrams represent the structure of the system

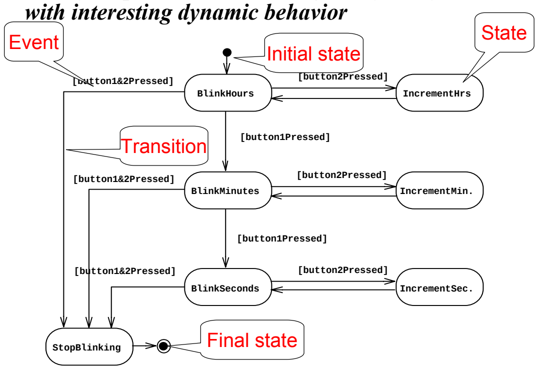


UML first pass: Sequence diagram



Sequence diagrams represent the behavior as interactions

UML first pass: Statechart diagrams for objects with interesting dynamic behavior



Represent behavior as states and transitions

UML Summary

- UML provides a wide variety of notations for representing many aspects of software development
  - **Powerful, but complex language**
  - **Can be misused to generate unreadable models**
  - **Can be misunderstood when using too many exotic features**

- For now we concentrate on a few notations:
  - **Functional model: Use case diagram**
  - **Object model: class diagram**
  - **Dynamic model: sequence diagrams, statechart and activity diagrams**