Introduction to Object-Oriented Design and Implementation

CS 4354
Fall 2012

Jill Seaman

Two textbooks

- Object-Oriented Software Engineering: Using UML, Patterns, and Java, by Bernd Bruegge and Allen H. Dutoit, Prentice Hall, 3rd edition. ISBN: 0136061257
  - ✦Main textbook for class (will use exercises from this book)
  - ✦Heavy on Software engineering
- Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, by Craig Larman, Prentice Hall, 3rd edition. ISBN: 0131489062
  - ✦More emphasis on object-oriented analysis and design
  - ✦Embedded in the UP (Unified Process) software process model

Object-oriented analysis

- Analysis: an investigation of the problem (rather than developing a solution)
- Requirements analysis: investigation of requirements
- Object-oriented analysis: emphasizes finding and describing the objects (or concepts) in the problem domain.
  - ✦For example, concepts in a Library Information System include Book, Library, and Patron.

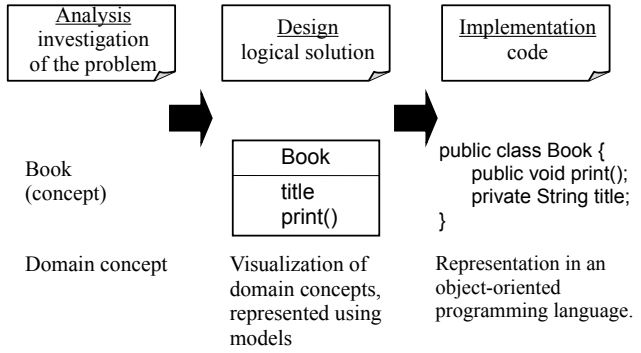
Object-oriented design

- Design: a conceptual solution that fulfills the requirements (rather than the implementation)
  - ✦Ultimately, designs can be implemented.
- Object-oriented design: define software objects and how they collaborate to fulfill the requirements.
  - ✦For example, in the Library Information System, a Book software object may have a title attribute and a getChapter method.
  - ✦Expressed using models (UML)

Object-oriented programming (or implementation)

- Designs are implemented in an object-oriented language such as Java or C++.
  - ✦A Java class for the Book object is written/implemented.
  - ✦Expressed in a program (source code)

Analysis + Design + Implementation



The UML

- The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems
- The standard diagramming notation for object-oriented modeling.
  - ✦Use case diagrams
  - ✦Sequence diagrams
  - ✦Class diagrams
  - ✦State (machine) diagrams
  - ✦Activity diagrams

Review of software engineering

- Software Engineering is a collection of techniques, methodologies and tools that help with the production of
  - ✦a high quality software system
  - ✦with a given budget
  - ✦before a given deadline
- while change occurs.

- Software engineering is an engineering discipline that is concerned with all aspects of software production

Four fundamental activities in a software process

- Software specification, where customers (and engineers) define the software that is to be produced and the constraints on its operation.
- Software development, where the software is designed and programmed (implemented).
- Software validation, where the software is checked to ensure that it is what the customer requires.
- Software evolution, where the software is modified to reflect changing customer and market requirements.

Recall: activities may be interleaved in cycles (iterative development)

Object-oriented software development

- Requirements elicitation
- Analysis
- System design
- Object design
- Implementation
- Testing

Ticket distributor system (case study)

- TicketDistributor is a machine that distributes tickets for trains. Travelers have the option of selecting a ticket for a single trip or for multiple trips, or selecting a time card for a day or a week. The TicketDistributor computes the price of the requested ticket based on the area in which the trip will take place and whether the traveler is a child or an adult. The TicketDistributor must be able to handle several exceptions, such as travelers who do not complete the transaction, travelers who attempt to pay with large bills, and resource outages, such as running out of tickets, change, or power.

Requirements elicitation

- During **requirements elicitation**, the client and developers define the purpose of the system.
- The result of this activity is a description of the system in terms of actors and use cases.
- **Actors** represent the external entities that interact with the system.
  - ✦Examples: end users, other computers, environment
- **Use cases** are general sequences of events that describe all the possible actions between an actor and the system for a given piece of functionality.

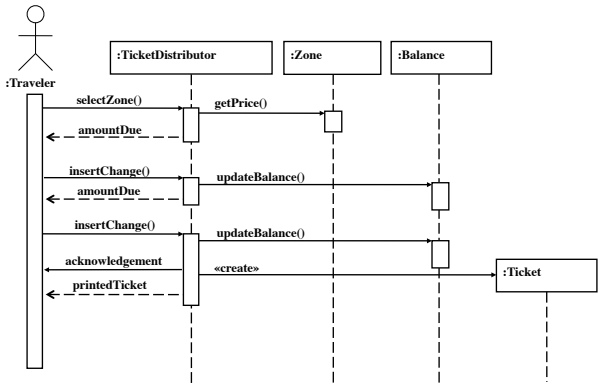
A use case: PurchaseOneWayTicket

Use case name	PurchaseOneWayTicket
Participating actor	Initiated by Traveler
Flow of events	1. The Traveler selects the zone in which the destination station is located. 2. The TicketDistributor displays the price of the ticket. 3. The Traveler inserts an amount of money that is at least as much as the price of the ticket. 4. The TicketDistributor issues the specified ticket to the Traveler and returns any change.
Entry condition	The Traveler stands in front of the TicketDistributor, which may be located at the station of origin or at another station.
Exit condition	The Traveler holds a valid ticket and any excess change.
Quality requirements	If the transaction is not completed after one minute of inactivity, the TicketDistributor returns all inserted change.

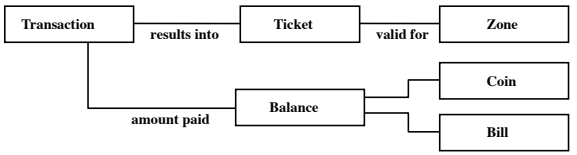
Analysis

- During **analysis**, developers aim to produce a model of the system that is correct, complete, consistent, and unambiguous.
- The result of analysis is a system model annotated with attributes, operations, and associations.
- The system model can be described in terms of its structure and its dynamic interoperation.

A dynamic model for the TicketDistributor



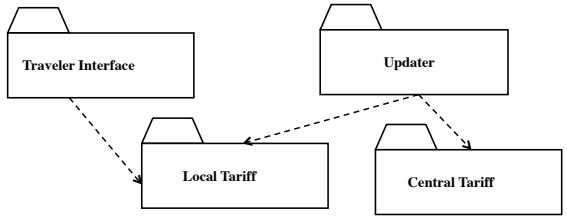
An object model for the TicketDistributor



System Design

- During **system design**, developers define the design goals of the project and decompose the system into smaller subsystems that can be realized by individual teams.
- The result of system design is a clear description of each of these strategies, a subsystem decomposition, and a deployment diagram representing the hardware/software mapping of the system.

A subsystem decomposition for the TicketDistributor



Object Design

- During **object design**, developers define solution domain objects to bridge the gap between the analysis model and the hardware/ software platform defined during system design.
- The result of the object design activity is a detailed object model annotated with constraints and precise descriptions for each element.

Implementation

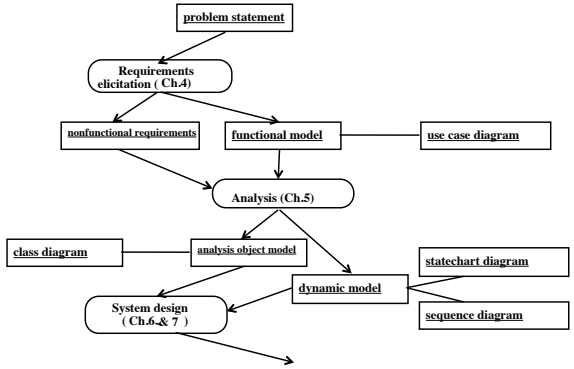
- During **implementation**, developers translate the solution domain model into source code.
- The result is the source code.

Testing

- During **testing**, developers find differences between the system and its models by executing the system (or parts of it) with sample input data sets.
- The planning of test phases occurs in parallel to the other development activities:
  - ✦System tests – requirements elicitation and analysis
  - ✦Integration tests – system design
  - ✦Unit tests – object design
- The execution of test phases generally occurs in the opposite order, during or after implementation.

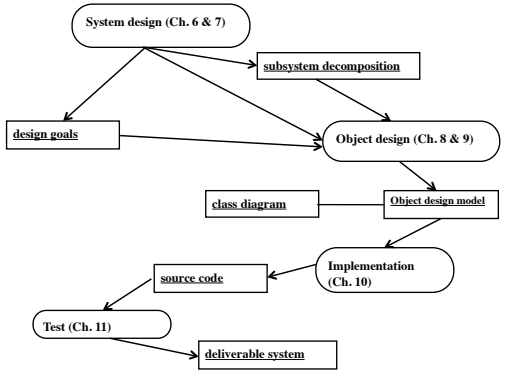
Object-oriented software development

- Activities and their products (part I)



Object-oriented software development

- Activities and their products (part II)



Where does OO Design and Implementation fit in software engineering processes?

- Software specification:
  - ✦Requirements elicitation
- Software development:
  - ✦Analysis
  - ✦System design
  - ✦Object design
  - ✦Implementation
- Software validation
  - ✦Testing