Lecture 12: Requirements Specification

Kenneth M. Anderson
Object-Oriented Analysis and Design
CSCI 6448 - Spring Semester, 2003

Requirements Specification

- Produces three types of models
  - State Models (Lecture 11)
    - Use Cases (some actors become classes)
    - Class Diagrams
  - Behavior Models (This Lecture)
    - Activity Diagrams
    - Interaction Diagrams
  - State Change Models (This Lecture)
    - State Chart Diagrams

Behavior Specifications (I)

- Behavior of a system, as it appears to an outside user, is specified in use cases
  - During analysis, use cases specify "what" a system needs to do (not "how")
- Use cases require computations to be performed
- Computations are divided into activities
  - and can be modeled using activity diagrams;
- Activities are carried out by interacting objects;
  - interactions are modeled using sequence diagrams

Credit where Credit is Due

- Some material presented in this lecture is taken from section 4 of Maciaszek’s “Requirements Analysis and System Design”. © Addison Wesley, 2000
Behavior Specifications (II)

- Provide an *operational view* of the system

**Main Tasks**

- Define use cases and determine which classes are used to execute a use case
- Identify operations on classes

- State specifications in analysis typically reveal entity classes; behavior specifications will often reveal *controller* classes and *boundary* classes (user interface classes)

---

Maciaszek’s Take: Use Cases

- A use case represents
  - a *complete* piece of functionality
    - Including main and alternate flows of logic
  - a piece of *externally visible* functionality
  - an *orthogonal* piece of functionality
    - use cases can share objects but execute independently from each other
  - a piece of functionality *initiated by an actor*
  - a piece of functionality that delivers *value to an actor*

---

Finding Use Cases

- Use cases are discovered via analysis of
  - requirements in the reqs. doc
  - actors and their purpose

- Jacobson suggests asking the following questions concerning actors to help identify use cases
  - What are the main tasks performed by the actor?
  - Will an actor access or modify information in the system?
  - Will an actor inform the system about changes in other systems?
  - Should an actor be informed about unexpected changes in the system?

---

Use Case Relationships

- **Association**
  - a communication path

- **Generalization**
  - a specialized use case can change any aspect of the base use case
  - include
    - directly includes steps of another use case
  - extend
    - customize an extension point

- See (poor) example on page 137 for the University Enrollment case study
- in general, the material from Lecture 9 and 10 supercedes any use case material provided by Maciaszek
Modeling Activities

- Activities capture the flow of logic within a system
- Both sequential and parallel control can be modeled
- Since activities do not reference classes, they can be created without the need for a class diagram
- Most often used to graphically represent the steps of a use case
  - Can show main flow and extensions at once
  - See example on page 142 (based on use case on page 139)
- Activities are best discovered by analyzing the action steps of use cases, with verbs indicating candidate activities

Modeling Interactions

- One level of abstraction below activities
- Interaction models require at least one iteration of state specification to be performed
  - Since we need to have classes to which each object belongs
  - Interaction diagrams do not model object state changes; however they may show the actions that lead to a state change
  - Interactions can help determine operations; any message to an object in an interaction must be handled by an operation (actually a method)
    - Recall that a method implements an operation; indeed there may be many methods available for a single operation

Discovering Message Sequences

- The sequence of messages in an interaction is determined by its associated activity (from the activity diagram)
  - The event that starts the activity is the first message in the interaction
  - The event that ends the activity is the last message in the interaction
  - We need to figure out what occurs in between; typically straightforward

Specifying message sequences

- Useful to distinguish between
  - Signals
    - Asynchronous inter-object communication
    - Often shown with “half-arrow notation”
  - Calls
    - Synchronous inter-object communication
    - Control returns to caller (usually)
- University Enrollment example on page 145 (shows the use of calls, not signals)
Defining Operations

A public interface of a class consists of operations that offer services to entities external to the class. Operations are best discovered from sequence diagrams, since every message must be serviced by an operation. Other operations can be found using the CRUD (create, read, update, delete) paradigm: classes need special functionality to provide these services regardless of their domain.

State Change Specifications

State change specifications define how an object changes state over time in response to particular events. States are discovered by analyzing the values of attributes and determining which have special interest to use cases. Interesting states are discovered by analyzing the values of attributes over time in response to particular events. Defining how an object changes state over time is referred to as the state change specification.

State Change Specifications

State change specifications define how an object changes state over time in response to particular events. States are discovered by analyzing the values of attributes and determining which have special interest to use cases. Interesting states are discovered by analyzing the values of attributes over time in response to particular events. Defining how an object changes state over time is referred to as the state change specification.

State Change Specifications

State change specifications define how an object changes state over time in response to particular events. States are discovered by analyzing the values of attributes and determining which have special interest to use cases. Interesting states are discovered by analyzing the values of attributes over time in response to particular events. Defining how an object changes state over time is referred to as the state change specification.

State Change Specifications

State change specifications define how an object changes state over time in response to particular events. States are discovered by analyzing the values of attributes and determining which have special interest to use cases. Interesting states are discovered by analyzing the values of attributes over time in response to particular events. Defining how an object changes state over time is referred to as the state change specification.

State Change Specifications

State change specifications define how an object changes state over time in response to particular events. States are discovered by analyzing the values of attributes and determining which have special interest to use cases. Interesting states are discovered by analyzing the values of attributes over time in response to particular events. Defining how an object changes state over time is referred to as the state change specification.