Lecture 5: Overview of Responsibility-Driven Design

Kenneth M. Anderson
University of Colorado, Boulder
January 25, 2005

Design Skills

- Wirfs-Brock and McKean argue that object design does not require rare and special "design" talent
  - They point to Betty Edwards's assertion that children can be taught to draw in the same way they can be taught to read
    - She says "What if we belived that only those fortunately endowed with inborn creative ability could learn to read?
- You can become good at object design with enough practice and experience
  - The key is learning to understand a design problem completely and then learning fundamental strategies for producing an acceptable solution

Overview

- Responsibility-Driven Design (RDD) involves
  - describing the actions and activities for which our software is responsible
  - describing the responsibilities in terms that both users and developers can understand
  - designing software objects that implement those responsibilities
- RDD is not a sequential process
  - We will present the technique in stages but, in practice, you may use the steps in different ways for each iteration of your design process

Project Definition and Planning

- The first step of a software development project involves
  - defining project goals
  - constructing a plan for achieving them
    - A plan describes how the system will be developed, the values important to the project and the people involved, project personnel and processes, and expected deliverables
  - receiving buy-in from various stakeholders before starting
- There are many different ways of planning a project; Fred Brooks in the *The Mythical Man-Month* suggests
  - 1/3 planning, 1/6 coding, 1/4 component test, 1/4 system test

RDD: Analysis

- The analysis stage of RDD consists of three phases
  - System Definition
    - High-Level View of System
  - Detailed Description
    - Detailed View of Development Process, Functional Requirements, and Non-Functional Requirements
  - Object Analysis
    - Construction of Domain Objects

System Definition

- Activities
  - Develop high-level system architecture
    - Make use of UML Deployment Diagrams or just “boxes and arrows”
    - Identify major subsystems
  - Identify System Concepts
    - Document important terms and concepts that are prevalent in early conversations about the system
  - Identify System Responsibilities
    - What are the major responsibilities of the system as a whole; be aware that these responsibilities will be decomposed

Detailed Description (I)

- Activities
  - Specify Development Environment
    - What tools, frameworks, APIs, etc. will be used during development
  - Specify User Tasks
    - Identify the different types of users
    - Create Use Case narratives (high-level task descriptions)
    - Create concrete usage examples via scenarios
  - Analyze Non-Functional Requirements

Detailed Description (II)

- Activities, continued
  - Document System Dynamics
    - Create activity diagrams that capture interactions of use cases
  - Prototype User Interface
    - Screen Mockups / Low-Fidelity Prototypes (sketches)
    - Navigation Design
      - What are the main elements of the user interface, how do they relate, how do you traverse from one section of the application to another

Object Analysis

- Activities
  - Identify Domain Objects with Intuitive Sets of Responsibilities
    - Use CRC cards to identify and work with candidate roles and objects
    - Iterate until an initial object model has been created
  - Document additional concepts and terms
    - Create glossaries or other documentation that define concepts, describe important behaviors, and capture business rules
    - What’s a business rule?
      - A policy that customizes a particular process to a specific organization

RDD: Design

- The design stage of RDD consists of two phases
  - Exploratory Design
    - Highly iterative development of the domain object model
  - Design Refinement
    - Finalize the object model; Prepare for Implementation Phase

Exploratory Design

- Activities
  - Associate domain objects with “solution” objects
  - Assign responsibilities to objects
  - Develop initial collaboration model
- Results
  - CRC model of objects, roles, responsibiliites, and collaborators
  - UML sequence/collaboration diagrams
  - Preliminary Class Definitions
  - Working Prototypes

Design Refinement (I)

- Activities
  - Justify Trade-Offs
    - Document design decisions
  - Design Application Control Styles
    - Identify control styles and decision making responsibilities
  - Decide visibility relationships between objects
    - Create refined UML class diagrams

Design Refinement (II)

- Activities, continued
  - Revise object model for flexibility, consistency, and maintainability
    - Create new object abstractions (develop inheritance hierarchy)
    - Revise object roles; adjust use of role stereotypes
    - Simplify interfaces and patterns of collaboration
    - Assign classes to roles (have classes implement particular interfaces)
    - Make use of design patterns
  - Document the design with UML diagrams
  - Formalize contracts between system components and classes

Coming Up Next

- Examine various techniques useful for analysis
  - CRC Cards
  - Use Cases
  - Scenarios
  - Conversations
- Then cover chapters 3—6
  - Finding Objects
  - Responsibilities
  - Collaborations and Control Styles