Lecture 23: OO Design Methods: Mathiassen, Part 3

Kenneth M. Anderson

Object-Oriented Analysis and Design

CSCI 6448 - Spring Semester, 2001

Goals of Lecture

- Introduce Mathiassen's method for application domain analysis
- Activities
  - Usage (Develop Use Cases)
  - Functions (Develop functional capabilities)
  - Interfaces (Develop user/system interface)

Application Domain Analysis

- How will the target system be used?
  - Goal is to identify the requirements for a system's functions and interfaces
- Application Domain analysis interacts with with problem domain analysis
  - What is the target domain?
  - Helps to define vocabulary that can be used throughout system development

Order of Analysis

- Mathiassen reveals that you can start with either application domain or problem domain analysis
  - Strategic Trade-off
    - Application Domain => focus on user's work
    - Problem Domain => focus on "business logic"
  - Starting with App. Domain is easier but starting with problem domain yields a better understanding of domain objects

Two Principles for Application Domain Analysis

- Determine the application domain with use cases
  - Use cases focus on the interaction between users and the target system
- Collaborate with users
  - Participatory design is required to get application domain analysis right

Usage Activity

- Derive actors and use cases for system
  - Actor: An abstraction of users or other systems that interact with the target system
  - Use case: A pattern for interaction between the system and actors in the application domain
- Steps (See page 120)
  - Find actors and use cases
  - Evaluate systematically
  - Explore patterns

Multiple Facets to Use Case Development

- It demands cooperation between users and developers:
  - users
    - formulate needs and contribute insights
  - developers
    - formulate use cases and contribute technical knowledge
- Determining Use Cases is an analytical as well as a creative activity
  - Use cases originate from needs and conditions in the application domain, but a use case itself is an expression of a solution (requiring creativity)

Multiple Facets, continued

- Creating use cases is a descriptive and experimental activity
  - User collaboration is key
    - Mathiassen recommends presenting use cases to users via prototypes; this will help to refine your understanding of particular use cases
- Use cases define both the target system and its application domain
  - Changes to a company's information systems affect the company's organization and way of working

Actor Tables

- Actor tables show the interaction between actors and use cases
  - See page 121
- Mathiassen claims that an actor table takes up less space (but shows the same information as) a use case diagram

Usage Activity: Step 1

- Find Actors and Use Cases
  - Who will use the system?
  - How will it be used?
- Identify Actors
  - To identify actors, you must determine the division of labor and the task-related roles in the target system's context
  - The criterion for determining actors is the dissimilarity of roles, as expressed by the use cases in which actors are involved

Finding Actors and Use Cases

- Describing Actors
  - Mathiassen describes actors using an actor specification (see page 126)
  - These consist of
    - a name
    - a goal (describes Actor's role)
    - characteristics (important aspects)
    - examples (clarify characteristics)

Find Actors and Use Cases

- Identify Use Cases
  - Use cases are defined based on a specific actor's viewpoint (what we called the primary actor earlier in the semester)
- Finding Use Cases
  - Produce a list of potential use cases by examining application domain tasks

Find Actors and Use Cases

- Describe Use Cases
  - Use cases can be described using state charts or textual descriptions
    - See page 127 and 128
  - State charts are good for defining an overview of the dynamic process and the logic of a use case, but it omits many details
  - Text descriptions conveys overview of usage details, e.g. the interaction, but makes it difficult to specify logic (think main success scenario and extensions)

Use Case and Actor Structures

- The fundamental structure between actors and use cases is participation
- Use case can be logically grouped
  - See page 129

Explore Patterns

- The Procedural Pattern
  - A basic sequence that ensures that business rules are followed
  - See page 130
- The Material Pattern
  - Characterized by actor being in one general state, where each action or sequence of actions eventually end back at the general state
  - See page 131 for text editor example

Evaluate Systematically: Three Methods

- Carefully review actor and use case descriptions to find mistakes and inconsistencies
  - Each use case should be simple and constitute a coherent whole
  - Descriptions should promote understanding and overview
  - Use cases need to be described in enough detail to enable identification of functions and interface elements
- Test use cases (with user) to see if they work in practice; use prototypes
- Evaluate the social changes in the application domain caused by the system; see page 132