Lecture 24: OO Design Methods: Mathiassen, Part 4

Kenneth M. Anderson

Object-Oriented Analysis and Design

CSCI 6448 - Spring Semester, 2001

Goals of Lecture

- Continue presenting Mathiassen's method for application domain analysis
- Activities
  - Usage (Last Lecture)
  - Functions (This Lecture)
  - Interfaces (This Lecture)

The Function Activity

- Purpose: To determine a system's information processing capabilities
- Definition: A *function* is a facility for making a model useful for actors
  - Mathiassen is referring to the model developed by the problem domain analysis (Mathiassen lectures, part 1 and 2)
- Principles
  - Identify all functions
  - Specify only complex functions
  - Check consistency with use cases and the model

The Function Activity

- Inputs
  - System Definition
  - Use Cases
- Steps (Page 139)
  - Find Functions
  - Specify complex functions
  - Evaluate Critically
- Output
  - Function list and specifications

Rationale behind Activities

- In the function activity, we ask
  - What is the system going to do?
- In the usage activity, we asked
  - How is the system to be used?
- Since its hard to separate "what" and "how," the function and usage activities are closely linked

More on Functions

- Traditionally, a function is a computation that transforms input data into output data
- It is difficult, however, to characterize complete systems from a purely functional point of view
  - They are useful, however, in that they express the intent of a system
  - from that point of view, identifying functions that a system must perform is helping to capture and create a system's requirements

Types of Functions

- Mathiassen identifies four types of functions (page 138-140)
  - Update
    - A function activated by a problem domain event and results in a change in a model's state
  - Signal
    - A function activated by a change in a model's state and results in a reaction (such as displaying information to an actor)
  - Read
    - A function activated by a need for information by an actor's action step and results in displaying information from a model
  - Compute
    - A function activated by a need for information by an actor's action step and consists of a computation that may use information supplied by an actor or model and whose result is then displayed to an actor

Types, continued

- Note: a specific function may not be "pure"
  - that is you might not be able to create functions that are purely "update" functions or purely "read" functions
- However, having four types of functions helps us perform application domain analysis, since it tells us "what to look for"

Goals of the Function Activity

- Produce a list of functions that are complete and consistent with the use cases and system model
  - Functions must support use cases
  - And all parts of the (problem domain) model should be used by some function
    - This involves determining if each class and event from the problem domain is being used by some function
    - If not, then the unused classes and events are modeling information that the system ultimately does not use

Step 1: Find Functions

- Two concerns
  - Where do the system’s function requirements come from?
    - Classes give rise to read and update functions
    - Events give rise to update functions
    - Use cases give rise to all four types
  - How detailed should the function descriptions be?
    - Must provide an overview of the system’s functionality
    - Must be able to serve as a basis of agreement between users and developers

Step 1, continued

- To find functions, ask questions
  - See figures 7.2, 7.3, 7.4, and 7.5 on pages 142-144
- In general,
  - Update functions are connected to events, because events must be recorded by the system
  - Read events are related to classes; the fact that classes capture information implies a need to read that information at a later point
  - Compute functions are needed because often reading information from a model is not enough
  - Signal functions are related to critical states of a system; states which require a reaction by an actor or system

Step 2: Describing Functions

- Most functions should not be described during application domain analysis
  - we should be striving for simple functions
  - only identified!
- If a complex function is identified, however, it is useful to provide additional information on it
  - via a mathematical expression, algorithm (figure 7.7), or further functional partitioning (figure 7.8)

Step 3: Evaluate Systematically

- Three methods
  - Users can review list
  - Review functions using questions from figures 7.2, 7.3, 7.4, and 7.5
  - Compare list with use cases and system model
- Output: A function list
  - See figure 7.6 on page 145

The Interface Activity

- Purpose: Determine a system's interfaces
- Concepts
  - Interface: Facilities that make a system's model and functions available to actors
  - User Interface: An interface to users
  - System Interface: An interface to other systems
- Principles
  - Tailor usability to the application domain
  - Experiment and iterate
  - Identify all interface elements

The Interface Activity, continued

- Results
  - A user interface including dialog styles, presentation forms, a complete list of user-interface elements, selected window diagrams, and a navigation diagram
  - A system interface including class diagrams for external devices and protocols of interaction with other systems

The Interface Activity, continued

- Inputs
  - Function list
  - Class diagrams
  - Use Cases
- Steps (see page 153)
  - Explore Patterns
  - Determine Interface Elements (for both UI and System)
  - Describe Interface Elements (for both UI and System)
  - Evaluate Interface Elements (for both UI and System)
- Output
  - Description of Interfaces

Step 1: Explore UI Patterns

- Menu Selection Pattern
  - Figure 8.3 on page 154
- Form fill-in Pattern
  - Figure 8.4 on page 154
- Command Language Pattern
  - Figure 8.5 on page 155
- Direct Manipulation Pattern
  - Figure 8.6 on page 156
- For more details, take Tammy's UI class!

Step 2: Determine UI Elements

- First, (according to Mathiassen), you must consider the presentation of each class and object of the problem domain
  - e.g. how should a customer, valve, document, order, etc. be represented in the (user interface of the) system?
  - See figure 8.7 on page 156 and the example discussed on pages 156-157

Step 2: Determine UI Elements

- Second, examine the interactions defined by use cases and
  - create sequence diagrams in which the objects are elements of the user interface (not objects of the problem domain)
  - See figure 8.8 on page 157

Step 3: Describe UI Elements

- It is important to specify all elements
  - but not to specify unnecessary detail
- On pages 161 to 163, Mathiassen reviews a number of UI design heuristics
  - Window diagrams for navigation concerns
  - Form guidelines
  - Heuristics for data display and window design
- UI Design is out-of-scope for this class
  - Again, take Tammy's UI Class for more information!

Step 4: Explore System Interface Patterns

- Need to answer the following two questions
  - What data should the system send to other systems?
  - What data should the system receive from other systems?
- Patterns can help; Mathiassen presents two
  - Read External Device (Figure 8.17 on page 165)
  - Interaction Protocol
    - More generally "Design an API" (Application Program Interface)

Step 5: Describe System Interface Facilities

- Mathiassen does not provide much direction in this section
  - He builds off of his two patterns
    - "Read External Device" patterns are described using class diagrams
    - Interaction Protocols can be further described using State diagrams
  - In general, you can use any of the diagrams from UML to describe system interfaces

Step 6: Evaluate the Interface

- Evaluation should focus on
  - The decomposition of the interface into a number of elements
    - Emphasizes navigation and whether our list of elements is complete; use use cases to assess
  - The design of individual interface elements
    - Emphasizes use of each element
    - Requires prototypes

Evaluating the Interface

- User Interface
  - Careful use of prototypes is required
  - Process outlined in Chapter 2
    - planning, development, preparation, test, summary
    - Standard Usability techniques apply (Tammy's UI class!)
- System Interface
  - Review design of API, have potential users review it also
  - Perform experiments, e.g. does the API scale?