Chapter 3
Object-Oriented Analysis

Requirements Analysis
- A requirement is a feature of the system
- Requirements analysis seeks to assess and specify the behavior of the final system
- Requirements analysis can be iteratively carried out or done in a top-down fashion
- It is desirable to establish the breadth of behavior of a system to determine the primary classes that will comprise the system

The Importance of Requirements Analysis
- Frederick Brooks: “The hardest single part of building a software system is deciding precisely what to build”
- Barry Boehm: by investing more up-front effort in verifying and validating the software requirements, software developers will see reduced integration and test costs, as well as higher software reliability and maintainability

Examples of Good Requirements Analysis
- Participatory analysis
  - Involves staff members of all levels from the domain area interacting directly with the development team
- Negotiation-based technique
  - Developed by USC Center for Software Engineering
  - Collaborative analysis approach involving end-users

Requirements Specification
- Requirements analysis results in a complete, consistent, correct, and unambiguous requirements specification
- The initial specification may be created by the end users or by the technical staff
- Independent of the source of the initial specification it must be refined and verified to be correct and complete

Possible Elements of Requirements Specification
- Supported activity list
- Human-computer interface description
- Solved problem list
- Information source list
- Information requesting organization list
- Checks and balances
- Security and fault-tolerant requirements
More: Possible Elements of Requirements Specification
- Inter-operating systems list
- Estimates of present information capacity and projected growth
- Project time frame
- Prioritization of requirements
- Ethical concerns list

Case Study: Library Management System
- Independent of who creates the requirements specification (end users or technical staff), it is the responsibility of the system developers to ensure the user requirements are adequately fleshed out.
- The first step of requirements analysis is to establish and refine the problem statement which takes the form of the requirements specification.

LMS Case Study: Clarifying the Requirements Specification
- What sort of human-computer interface is envisioned?
- What sort of search facility is necessary for library patrons?
- Will patrons be assigned a unique ID number?
- Should the system support inter-library loan requests?

LMS Case Study: More Clarifying the Requirements Specification
- Are there any limits on patrons’ use of research databases?
- How are books retired from the library collection?
- How long are requested books held for patrons?
- Are overdue fees the same for all type of library resources?
- Which online databases will the system interact with?

Creating Quality Requirements Specifications
- The key is to keep in close communication with the end users throughout the development process, but especially during requirements analysis.
- Ideally, a whole array of different end users should be involved with the development team to gain sufficient breadth of input.

LMS Case Study: Supported Activity List
- Support Library staff activities like checking out resources to patrons
- Validating patrons
  - Current membership
  - No resources more than two weeks overdue
  - Not over maximum of checked resources
- Assigning library numbers to patrons
LMS Case Study: More Supported Activity List

- Deleting expired library numbers and associated patron records
- Checking out library resources: books, CDs, etc
- Checking in library resources
- Changing the status of a library resource
- Allowing materials to be placed on reserve
- Allowing the searching of the library’s holdings on line
- Additional activities listed in text

More Elements of the LMS Requirements Specification

- Human-computer interface
- Solved problems list
- Information source list
- Information requesting organizations
- Automated checks and balances
- Security and fault-tolerant requirements
- Present information capacity and projected growth

Verifying Requirements

- A structured walkthrough with the end users is a good technique for ensuring that the user needs are being addressed
- To ensure that the resulting software supports the requirements specification, items on the supported activity list are numbered and propagated through the software models and source code

The Process of Requirements Analysis

- Create verified requirements specification
- Create list of primary classes
- Create informal scenarios
- Create use cases
- Create scenarios
- Create class diagrams
- Create use case diagrams

Determining Primary Classes

- Select nouns from the requirements specification and inspect each noun for the following properties
  - Retained information
  - Needed services
  - Multiple attributes
  - Common attributes
  - Common operations
  - Essential requirements
LMS Case Study: Primary Classes
- Patron
- Student, faculty, library staff
- Resource
- Book, music CD, video, software
- Reference resource, reserved resource, requested resource, online research resource
- Inter-library loan request
- Overdue charge
- Overdue form letters

Identifying Use Cases
- A use case is a description of a scenario (or closely related set of scenarios) in which the system interacts with users of the system
- The behavior of the system is expressed without specifying how the behavior is implemented
- Use cases are initially described narratively, and then modeled graphically by class diagrams and interaction diagrams (to be discuss later)
- Informal scenarios are a good starting point for use cases

Characteristics of Use Cases
- Use cases are more abstract than informal scenarios
- A single use case may encompass multiple scenarios
- Use cases avoid redundancy
- Use cases are more formally structured than scenarios
- Use cases seek to capture complete breadth of system behavior

Use Case Layout
- Precondition
  - What conditions must be true at the beginning of the use case?
- Main flow of events
  - Describe the essential behavior associated with the use case
- Post condition
  - What occurs as a result of the use case executing
- Exceptional flow of events (zero to many)
  - Enumerate possible erroneous flow of events

LMS Case Study: Check out Resource Use Case
- Precondition
  - Library staff and patron validated to LMS
  - Library DB initialized
- Main flow of events
  - Enter resource
  - Determine due date
- Exceptional flow of events
  - Patron ID not valid
  - Patron has over due resources or too many checked
  - Resource number not valid

LMS Case Study: Use Cases
- Validate patron
- Check out resource
- Check in resource
- Request resource
- Reserve resource
- Manage Resource
- Manage Patron
- Generate from letter
More LMS Case Study: Check out Resource Use Case

- **Postcondition**
  - Patron DB entry updated to reflect new resource
  - Resource DB entry updated to reflect changed status: checked out
  - Due date assigned to the resource DB entry

Scenario Development

- Scenarios are derived from use cases
- Scenarios are like informal scenarios, but are more formally structured
- Informal scenarios may be modified to produce scenarios
- Use cases are abstract because they do not reference specific values
- Scenarios are concrete because they do reference specific values
- Multiple scenarios may be required for a single use case

Modeling the System with UML

- The process of modeling parallels and supports the process of understanding the system requirements
- Two types of models support the analysis process
  - Class diagrams
  - Use case diagrams

Class Diagrams

- Models the composition of classes and the essential relationships between classes
- Models a static perspective of the system
- May expresses a more or less abstract representation of the system
- The notational building blocks
  - Classes
  - Interfaces
  - Relationships
  - Collaborations

Notational Elements of Class Diagrams

- **Class**
  - Class Name
- **Detailed Class**
  - Interface
- **Relationships**
  - Dependency
  - Association
  - Generalization
  - Collaboration
  - Name

LMS Case Study: Class Diagram

- Patron
- Resource
- Requests
- Checks out
- Browses
- Returns
- Overdue form Letter
- Library staff
- generates
Use Case Diagrams

- Use case diagrams depict use cases interacting with external actors
- External actors are entities that interact with the software system, like system users, databases, or books
- Use cases represent system requirements and show a functional partitioning of the system
- Functional partitioning is useful for dividing a system into smaller pieces

---

Notational Elements of Use Case Diagrams

- Actor
- Use case
  - Use case name
- Relationships:
  - Dependency
  - Association
  - Generalization

---

Steps in UCCD Analysis Process

- Create/refine requirements specification
- Create informal scenarios
- Create list of primary classes
- Create use cases
- Create scenarios from use cases
- Create class diagrams showing basic inter-class relationships
- Model key class collaborations
- Create use case diagrams

---

Evolving the System

- Requirements analysis may be done iteratively throughout system development
- The system to be developed may be partitioned into development subgoals
- Each subgoal has its own requirements analysis phase that it followed by design, implementation, and testing
- Each subset of the system is made work before the next subgoal is analyzed
Analyzing the Class Project

- List the primary classes
- Create a basic class diagram showing aggregation and inheritance
- Create use cases
- Create class diagrams
- Create use case diagrams
- Create one or more scenarios for each use case
- Engage in a structured walkthrough with your instructor

Working in Teams

- Development teams should meet at least once a week
- A common list of primary classes should be created by the team
- The creation of use cases, class diagrams, and scenarios should be divided amongst development team members
- The team as a whole should review the individual products to ensure that the pieces fit together

Additional Pointers for Effective Team Work

- The role of the chair is to facilitate discussion
- Each team member should have equal opportunity to be heard
- The meeting chair to make an extra effort to hear from less aggressive team members
- Team members should not be interrupted unless they are being long-winded
- Everyone should strive to make their points as concisely as possible