Chapter 21
Object-Oriented Analysis

Intent of OOA
- To identify all objects relevant to a given problem
- To identify the operations associated with the objects, i.e., their behavior
- To identify the attributes of the objects
- To identify relationships among the objects
- TO DEFINE ALL CLASSES RELEVANT TO A PROBLEM

SA versus OOA
Structured Analysis
- Data are considered separate from the process that transforms the data
- Functional decomposition is prevalent in architecture

Object-Oriented Analysis
- Takes an "object" view in the evolution of requirements
- Data and attributes are an integral part of each object
- Architecture decomposition is object-based

Domain Analysis
Objective: To create a library of reusable classes (components) that will be broadly applicable to an entire category of applications

OO Technologies are leveraged through REUSE

OOA - A Generic View
OOA begins with an understanding of the manner in which the system will be used
- define use cases
- extract candidate classes
- establish basic class relationships
- define a class hierarchy
- identify attributes for each class
- specify methods that service the attributes
- indicate how classes(objects) are related
- build a behavioral model
- iterate on the first five steps

Use Cases -- Objectives
- To define the functional and operational requirements of the system (or product) by defining a scenario of usages
- To provide a clear and unambiguous description of how the end-user and system will interact
- To provide a basis for validation testing
Use Cases -- Embodiment

- a scenario that describes a “thread of usage” for a system
- actors represent roles people or devices play as the system functions
- users can play a number of different roles for a given scenario

Developing a Use Case

- What are the main tasks or functions that are performed by the actor?
- What system information will the the actor acquire, produce or change?
- Will the actor have to inform the system about changes in the external environment?
- What information does the actor desire from the system?
- Does the actor wish to be informed about unexpected changes?

Unified Modeling Language (UML)

User model view. This view represents the system (product) from the user’s (called “actors” in UML) perspective.

Structural model view. Data and functionality is viewed from inside the system. That is, static structure (classes, objects, and relationships) is modeled.

Behavioral model view. This part of the analysis model represents the dynamic or behavioral aspects of the system.

Implementation model view. The structural and behavioral aspects of the system are represented as they are to be built.

Environment model view. The structural and behavioral aspects of the environment in which the system is to be implemented are represented.

Class-Responsibility-Collaborator (CRC) Modeling

Identify classes, indicate their responsibilities, identify collaborations

Using (virtual) index cards

Class-Responsibility-Collaborator (CRC) Modeling

Identify classes, indicate their responsibilities, identify collaborations
Identifying Class Responsibilities

Class responsibilities are defined to be attributes and operations

- **Attributes**
  - extracted from statement of scope
  - described for understanding the nature and operations of a class
- **Operations**
  - perform a grammatical parse of the processing narrative (use cases)... all verbs represent potential operations

Allocating Responsibilities to Classes

- System intelligence should be evenly distributed
  - concentrating intelligence to a few classes makes changes more difficult
  - tends to require more classes, hence more development effort
- Each responsibility should be stated as generally as possible
  - let sub-classes refine responsibilities
  - encourages polymorphism
- Information and the behavior that is related to it (or operations that modify it) should reside within the same class
  - promotes encapsulation and information hiding

Identifying and Allocating Collaborations

Collaborations identified by relationships between classes

- **is-part-of**
  - all classes that are part of an aggregate (super) class
- **has-knowledge-of**
  - when one class must acquire information from another class
- **depends-upon**
  - implies that two classes have a dependency other than one of two above (e.g., is-connected-to)

Reviewing the CRC Model

1. All participants in the review (of the CRC model) are given a subset of the CRC model index cards.
2. All use-case scenarios (and corresponding use-case diagrams) should be organized into categories.
3. The review leader reads the use-case deliberately. As the review leader comes to a named object, the review leader passes a token to the person holding the corresponding class index card.
4. When the token is passed, the holder of the class card is asked to describe the responsibilities noted on the card. The group determines whether one (or more) of the responsibilities satisfies the use-case requirement.
5. If the responsibilities and collaborations noted on the index cards cannot accommodate the use-case, modifications are made to the cards

UML: Class Diagrams

Defining the Structures and Hierarchies

Focusing on the Class model and resulting hierarchies

Generalization / Specialization

Generalization class refined into Specialization classes

Attributes and operations for sensor class are inherited by the specialization class
Composited aggregates
Object represented in initial model might be composed of a number of components, each of which could be defined as objects.

Relationships between Objects
Note similarity between ERD and UML approach to modeling relationships among objects.