**5.5 Object - Oriented Design**

Object-oriented design transforms the analysis model created using object-oriented analysis into a design model that serves as a blueprint for software construction. The unique nature of object-oriented design lies in its ability to build upon four important software design concepts: abstraction, information hiding, functional independence, and modularity. All design methods strive for software that exhibits these fundamental characteristics, but only OOD provides a mechanism that enables the designer to achieve all four without complexity or compromise.

**5.5.1 Design for Object-Oriented System**

The design pyramid for conventional software has four design layers - data, architectural, interface, and component level. For object-oriented systems, it also defines a design pyramid, but the layers are a bit different. Referring to Figure 5.4, the four layers of the OO design pyramid are:

1. **The subsystem layer** contains a representation of each of the subsystems that enable the software to achieve .its customer-defined requirements and to implement the technical infrastructure that supports customer requirements.
2. **The class and object layer** contains the class hierarchies that enable the system to be created using generalizations and increasingly more targeted specializations. This layer also contains representations of each object.
3. **The message layer** contains the design details that enable each object to communicate with its collaborators. This layer establishes the external and internal interfaces for the system.
4. **The responsibilities layer** contains the data structure and algorithmic design for all attributes and operations for each object.

Although similarity between the conventional and OO design models docs exist, it has new names of the layers of the design pyramid to reflect more accurately the nature of an OO design. Figure 5.5 illustrates the relationship between the OO analysis model and design model that will be derived from it.



Figure 5.4 OO Design Pyramid



Figure 5.5 Translating an OOA into an OOD model

The subsystem design is derived by considering overall customer requirements (represented with use-cases) and the events and states that are externally observable (the object-behavior model). Class and object design is mapped from the description of attributes, operations, and collaborations contained in the Class Responsibility- Collaborator (CRC) model. Message design is driven by the object relationship model, and responsibilities design is derived using the attributes, operations, and collaborations described in the CRC model.

**5.5.2 The Object-Oriented Design Process**

It is at this stage that the basic concepts and principles associated with component level design come into play. Local data structures are defined (for attributes) and algorithms (for operations) are designed.

**1. Object Descriptions**

A design description of an object (an instance of a class or subclass) can take one of two forms: (1) a protocol description that establishes the interface of an object by defining each message that the object can receive and the related operation that the object performs when it receives the message or (2) an implementation description that shows implementation details for each operation implied by a message that is passed to an object. Implementation details include information about the object's private part; that is, internal details about the data structures that describe the object's attributes and procedural details that describe operations. The protocol description is nothing more than a set of messages and a corresponding comment for each message.

**2. Designing Algorithms and Data Structures**

A variety of representations contained in the analysis model and the system design provide a specification for all operations and attributes. Algorithms and data structures are designed using an approach that differs little from the data design and component-level design approaches discussed for conventional software engineering. An algorithm is created to implement the specification for each operation. In many cases, the algorithm is a simple computational or procedural sequence that can be implemented as a self-contained software module. However, if the specification of the operation is complex, it may be necessary to modularize the operation. Conventional component-level design techniques can be used to accomplish this. Data structures are designed concurrently with algorithms. Since operations invariably manipulate the attributes of a class, the design of the data structures that best reflect the attributes will have a strong bearing on the algorithmic design of the corresponding operations. Although many different types of operations exist, they can generally be divided into three broad categories: (1) operations that manipulate data in some way (e.g., adding, deleting, reformatting, selecting), (2) operations that perform a computation, and (3) operations that monitor an object for the occurrence of a controlling event.

**3. Program Components and Interfaces**

An important aspect of software design quality is modularity; that is, the specification of program components (modules) that are combined to form a complete program. The Object oriented approach defines the objects a program component that is itself linked to other components (e.g., private data, operations). But defining objects and operations is not enough. During design, we must also identify the interfaces between objects and the overall structure (considered in an architectural sense) of the objects.

**5.5.3 Object-Oriented Programming**

The software engineering viewpoint stresses OOA and OOD and considers OOP (coding) an important, but secondary, activity that is an outgrowth of analysis and design. The reason for this is simple. As the complexity of systems increases, the design architecture of the end product has a significantly stronger influence on its success than the programming language that has been used. And yet, "Language wars" continue to rage. An OO programming language is used to translate the classes, attributes, operations, and messages into a form that can be executed by a machine.