Overview of Object-Oriented System Development Method and UML Modeling Language

Ying Peng

Department of economics and management, Dehong normal college, Princeton Yunnan, Dehong 678400, China

Keywords: Object-oriented method; Modeling; UML

Abstract: This paper outlines the object-oriented methods and related technologies, and describes the object-oriented system development methods, including object-oriented analysis (OOA), object-oriented design (OOD), UML unified modeling language and Rational Rose modeling tools. Unified Modeling Language (UML) and Rational Rose, a modeling tool, have established a requirement analysis model and design model suitable for management system software. At the same time, they have provided a relatively complete analysis method and design ideas for the development of teaching management system in Colleges and universities.

1. Introduction

Object-Oriented Method, which originated from Object-Oriented Programming Language (Simula, Object-c), has gradually been applied to the process of system analysis, design and implementation with the development of object-oriented programming technology. The idea of object-oriented method to solve the problem is to start with the objective objects (such as people and things) in the real world, and try to use the natural thinking mode of human beings to construct the software system. In the object-oriented method, all objective things are regarded as objects, so that the process of system development and the process of people's understanding of the objective world keep the maximum consistency. Therefore, the software system obtained by object-oriented development method has high quality, adaptability, reliability, reusability and maintainability. In the process of internal and external environment changes, the system is easy to maintain a longer stability and life cycle [3].

From the point of view of programming method, object-oriented is a new paradigm. Its basic idea is to use the basic concepts of object, class, inheritance, encapsulation, aggregation, association, message and polymorphism to program [4]. Since the 1980s, object-oriented methods have penetrated into almost all branches of computer software. It is not only some specific software development technologies and strategies, but also a set of software methodology on how to view the relationship between software system and the real world, what viewpoints to study and solve problems, and how to construct systems.

1.1. Basic Concepts of Object-Oriented.

(1) Object

Object is the basic construction unit of object-oriented system and a set of related variables and methods. It is often used to build some object models in the real world, such as desks, chairs, computers and so on, which can be regarded as objects. Object description usually consists of three parts: identification, attribute and operation. Object description usually consists of three parts:

1) Identification: The name of the object used to distinguish other objects in the problem domain;

2) Attribute: A data item describing the static characteristics of an object;

3) Operation: An action sequence that describes the dynamic characteristics (behavior) of an object.

When building a software system from the objective things in the real world, it is mainly to think and understand the problems directly centered on the things in the problem domain (the real world). According to the essential characteristics of these things, they are abstractly represented as objects

  DOI: 10.25236/iciss.2019.005

in the system, as the basic unit of the system. This enables the system to map the problem domain directly and keep the original appearance (object) of the things in the problem domain and their interrelations.

(2) Class

Class is a set of objects with the same attributes and operations. It provides a unified abstract description for all objects belonging to the class. The relationship between classes and objects is similar to that between a die and a casting made with this die. A class gives a unified definition for all its objects, and its birth object is an entity that conforms to this definition, so an object is also called an instance of a class. The definition of class mainly includes the following elements:

1) Identification: the name of a class to distinguish other classes;

2) Data structure: description of attribute name and type;

3) Operations: The specific implementation description of the method in the class is what operations the object of the class is called to perform.

(3) Message

Message refers to the information describing the occurrence of events. It is the way in which objects relate and interact with each other. A message consists of five parts: sending object, receiving object, delivery mode, content and parameters, and message return. The purpose of incoming message content is twofold: one is to let the recipient get information about the execution of the task, and the other is the behavior instruction.

1.2. Pair-oriented Features.

Object-oriented technology uses the thinking method commonly used by human beings in the process of recognizing the objective world to describe the related things intuitively and naturally. Abstraction, encapsulation, inheritance and polymorphism are the basic characteristics of object-oriented.

(1) Abstraction

Abstraction is to ignore the non-essential characteristics of things that are not related to the current goals, emphasize the characteristics related to the current things, and classify things correctly to get the abstract model of things, usually through classes to achieve the abstraction of object state and behavior.

(2) Encapsulation

Encapsulation has two meanings: first, it combines all the States and behaviors of the object to form an indivisible whole. Second, hide the internal details of the object as much as possible, and the connection with the outside world can only be achieved through the external interface. However, in the process of use, excessive encapsulation should also be avoided.

(3) Inheritance

Inheritance is a hierarchical model connecting classes and classes, in which objects of special classes have attributes and behaviors of their general classes. Special classes automatically and implicitly possess the attributes and behaviors of their general classes. Sometimes general classes are called base classes and special classes are called derivative classes. Inheritance can only simplify the definition of derived classes, reuse and expand existing class library resources, and more importantly, make software easy to maintain and modify.

(4) Polymorphism

As far as object-oriented programming is concerned, polymorphism means that the same function name corresponds to many different functions with similar functions in two or more different classes.

2. Object-Oriented System Development

The object-oriented system development cycle is similar to the traditional software life cycle, but the problems solved in each stage and the descriptions adopted are quite different.

2.1. Object-Oriented Analysis (OOA).

Object-oriented analysis is the application of object-oriented method to system analysis [6]. Its basic task is to use the object-oriented method to analyze and understand the problem domain and system responsibility, to have a correct understanding of the relationship between them, to find out the classes and objects needed to describe the problem domain and system responsibility, and to define the attributes and operations of these classes and objects, as well as the various relationships formed between them. The ultimate goal is to produce an OOA model that meets the needs of users and directly reflects the problem domain and system responsibility. OOA model refers to the establishment and system model by using object-oriented analysis method. It includes three parts: basic model (class diagram), requirement model (use case diagram) and auxiliary model (package diagram, sequence diagram, activity diagram, etc.). Among them, the basic model expresses the most important system construction information intuitively; the requirement model is used to define user requirements; and the auxiliary model provides several auxiliary graphics to organize or enhance understanding of the basic model.

(1) Class diagram. Class diagrams show that the static structure of the system consists of classes, attributes, operations, whole and part structures, general and special structures, associations and messages.

(2) Use case diagram. Use cases are sometimes referred to as use cases. This concept is derived from the Jacobson method [], which describes the use of each system function by participants outside the boundaries of the system.

(3) Package drawings. Package is a mechanism for organizing other model elements to form system units of trap granularity. This mechanism of organizing model elements is called topic in Coad/Yourdon method []. The idea is that when there are many components in a system model, it is difficult to read and understand the whole model. By using the principle of granularity, we can organize some classes which are closely related in concept, structure or behavior together to form fewer topics, which can make the model clearer. The graphical representation of packages and their relationships is called package graphs. The package diagram shows which packages are contained in the system and the relationships among them.

(4) Sequence diagram. A sequence diagram is usually only suitable for representing the interaction between a limited number of objects in the system, but not for representing the behavior relationship of the whole system.

(5) Activity maps. The function of activity diagram is to model the behavior of the system. It represents a behavior in the system as an activity that can be executed by a computer, person or other executor. It describes the behavior of the system by giving the various actions in the activity and the transfer relationship between the actions.

The OOA process includes the following main activities:

1) Establish the requirement model-use case diagram. Related activities include: determining system boundaries; discovering participants; and defining use cases.

2) Establishing the basic model-class diagram. Relevant activities include: discovering objects and defining their classes; defining the internal characteristics of objects - attributes and operations; defining the external relations of objects - General and special structures, associations and messages, overall and partial structures.

3) Establishing an auxiliary model. Relevant activities include: dividing packages, building package diagrams; establishing sequence diagrams; building activity diagrams; building other model diagrams.

2.2. Object-Oriented Design (OOD).

Object-oriented design is the process of transforming the requirements obtained in the analysis phase into abstract system implementation solutions that meet the cost and quality requirements [7]. From object-oriented analysis to object-oriented design, it is a process of gradually expanding the model. The system design determines the strategy of realizing the system and the high-level structure of the target system. The high-level structure of the system includes the decomposition of

subsystems, the allocation of subsystems to hardware and software data storage management, resource coordination, software control implementation and human-computer interaction interface. The system design usually starts from the high level, then refines. The design of the system depends on the whole structure and style, which provides the basis for the design of more detailed strategies in the later design stage.

(1) System design

The general steps of the whole system design are shown below.

1) System decomposition. The main component of the system is called subsystem, which is not an object or a general processor, but a set of classes, associations, operations, events and constraints.

2) Processor and task allocation. Each concurrent subsystem must be assigned to a single hardware unit, either a general processor or a specific functional unit.

3) Data storage management. Usually, each data storage can combine data structure, file and database. Different data storage needs to compromise between cost, access time, capacity and reliability.

4) Global resource processing. Global resources must be identified, and strategies for accessing global resources must be developed.

5) Choosing software control mechanism. The system design must choose one method from many methods to realize the control of software.

6) Design of human-computer interaction interface. Most of the work in the design is related to stable state behavior, but the user's interaction interface with the system must be considered.

(2) Guidelines for Object-Oriented Design

When using object-oriented technology in system design, certain criteria should be followed, including modularity, abstraction, information hiding, low coupling and high cohesion.

1) Modularization. Opposite-oriented development method naturally supports the design principle of decomposing the system into modules, that is, objects are modules. It is a module that combines the data structure with the method of operating these data structures. Class design well supports the modularity principle, which makes the system more maintainable.

2) Abstraction. Object-oriented method does not support abstraction of process and data. The quality of abstract methods and the level of abstraction have a great influence on the design of the system.

3) Information hiding. In the object-oriented method, information hiding is realized by the closeness of the object. The number of interfaces exposed by objects and the quality of interfaces have a great impact on system design.

4) Low coupling. In object-oriented method, object is the most basic module, so coupling mainly refers to the degree of tightness between different interrelated. Low coupling is an important criterion for design, because it helps to minimize the impact of changes in one part of the system on other parts.

5) High cohesion. In object-oriented methods, high cohesion is also a condition that must be met. High cohesion means that as many logically relevant computing resources should be collected in an object class as possible. If a module is responsible for only one thing, it means that the module has a high degree of cohesion; if a module is responsible for many irrelevant things, it means that the cohesion of the module is very low. Cohesive modules are usually easy to understand, reuse, extend and maintain.

3. UML Modeling Technology

UML (Unified Modeling Language) is a general visual modeling language for describing, visualizing, processing, constructing and manufacturing software system products. UML unifies the basic concepts of Booch, OMT [] and OOSE [] etc. [5].

3.1. Basic Composition of UML Language.

We collectively refer to the basic concepts that can be used in diagrams as model elements.

Model elements are defined using relevant semantics and formal definitions of elements, with definite statements to express precise meanings. Model elements are represented by their corresponding element symbols in the graph. The model elements can be visually expressed by using the symbols of relevant elements. An element symbol can exist in many different types of graphs.

3.2. Things.

Things are the basic object-oriented elements and modules in UML model, which belong to the static part of the model. In UML, four basic object-oriented things are defined.

Structural things: Structural things are noun parts in UML models. These nouns often constitute the static part of the model and are responsible for describing static concepts and objective elements. In the UML specification, the following structural things are defined: classes, interfaces, use cases, etc.

Behavior: Behavior refers to the related dynamic behavior of UML model, which is the dynamic part of UML model. It can be used to describe the behavior across time and space. Active things are usually represented by verbs in models. For example, "registration", "destruction" and so on. Behavior can be divided into two categories: interaction and activity.

Grouping things: Grouping things is a mechanism for UML to group things among various components of the model. At present, there is only one grouping, package. UML organizes the whole model by grouping things like packages, including all the graphic modeling elements that make up a complete model.

Annotating things: Annotating things is the explanatory part of the UML model, which is used to further illustrate any other part of the UML model. We can use annotations to describe, illustrate and annotate any element in the entire UML model. There is one of the most important annotations, which we call annotations.

3.3. UML Element Relations.

UML model is composed of all kinds of things and all kinds of relations among them. Relationships refer to rules governing and coordinating the existence and mutual use of various model elements. There are five relationships in UML: dependency, association, generalization, aggregation and implementation.

Dependency-Relevance: Dependency refers to a semantic relationship between two things. When one thing (independent thing) changes, it will affect the semantics of another thing (dependent thing). Relevance is a structural relationship between things. We use it to describe a group of chains, which are the connections between objects.

Generalization-Realization: Generalization is a special/general relationship between things. Objects of special elements (sub-elements) can replace objects of general elements (parent elements). It describes the semantic relationship between a set of operation specifications and a set of specific implementations of operations.

Aggregating: Aggregating is a kind of association and a strong association. Aggregation is the relationship between the whole and the individual, such as the relationship between the company and the legal adviser. Composition is a stronger correlation than aggregation. It requires the object representing the whole in the general aggregation relationship to be responsible for the life cycle of the object representing the part, such as the relationship between car wash and engine.

3.4. Representation of Model Graph in UML.

UML as a visual modeling language, its main manifestation is the graphical representation of the model. The UML specification strictly defines the symbols of various model elements, and also includes the abstract grammar and semantics of these models and symbols. When using these diagrams, it makes the developed application easier to understand. The most commonly used UML diagrams include use case diagrams, class diagrams, sequence diagrams, activity diagrams, etc.

(1) UML use case diagram

UML use case diagrams describe a functional unit provided by the system. The main purpose is

to help the development team understand the functional requirements of the system in a visual way, including the process-based "role" relationship (inheritance relationship), and the relationship between the use cases in the system (including include, extend relationship). The way to draw a use case in a use case diagram is to draw an ellipse in it, and then place the name of the use case in the center of the ellipse or in the middle under the ellipse. The way to draw a role on a use case diagram is to draw a human symbol.

(2) UML Class Diagram

Class descriptions in UML class diagrams use three-part rectangular descriptions as shown in Figure 2-8. The top rectangular part shows the name of the class, the middle rectangular part shows the properties of the class, and the bottom rectangular part shows the operation or method of the class. In class diagrams, the relationship between classes and classes usually has three relationships: dependency, generalization and association. If the interface is also considered as a class, then there is implementation relationship, that is, the implementation of class-to-class interface. In UML, a line segment with an arrow pointing to the parent class at the vertex is usually used to draw a generalization relationship, and this arrow is a complete triangle. If the two classes know each other, they should use a solid line to represent the association and a line segment with an arrow pointing to the interface at the vertex.

(3) Sequence diagram

There are two dimensions in UML order: vertical dimension and horizontal dimension. In the process of drawing sequence diagrams, each box represents an instance or object of each class across the top of the diagram. In the box, the class instance name and the class name are separated by a colon. If a class instance sends a message to another class instance, draw a line with an open arrow pointing to the receiving class instance and place the name of the message or method on the line. By reading the sequence diagram, you can understand the process of administrator login system.

(4) UML activity diagram

In UML, the initial activity begins with a solid circle, the end of activity drawing is represented by a circle with solid congratulations inside, and the activity is represented by a rounded rectangle. Usually the name of the activity is included in the rounded rectangle.

3.5. Rational Rose Tool.

Rational Rose is a complete and flexible solution developed by Rational Inc. [3,6], which can meet the requirements of all modeling environments, including Web development, database modeling, and various development tools and languages. The following support is provided for UML: (1) providing basic drawing function for UML; (2) providing a repository for model elements; (3) providing navigation function for various views and graphs; (4) providing code generation function; (5) providing reverse engineering function; (6) providing model interchange function. The model established by Rational Rose includes Use Case View, Logical View and so on. These views are automatically included when we create a Rational Rose project.

This paper mainly introduces the idea of object-oriented system development, UML modeling language and Rational Rose, which provides a solid theoretical basis for the analysis and design of UML-based management system.

References

[1] Zhao Chilong, Jiang Yiping, Zhang Jian. Software Engineering Practice Course [M]. Electronic Industry Press, 2007.

[2] Liu Zhicheng, Chenghuan. Software Engineering and Rose Modeling Case Course [M]. Dalian University of Technology Press, 2009.

[3] Wang Yan. Student Management Information System Analysis and Design Research Based on Object-Oriented Method [D]. Southwest Jiaotong University, 2006.

[4] Song Haoyuan. An overview of object-oriented programming method [J]. Journal of Chongqing University of Science and Technology: Natural Science Edition, 2008, 10(2): 99-102.

[5] Yang Jiying. Object-oriented system analysis and design [J]. Science and technology information, 2008 (20): 48-49.

[6] Qi Zhichang, Tan Qingping, Ninghong. Software Engineering [M]. Higher Education Press, 2012.

[7] Li Xingpeng. Modeling research of university scientific research management system based on UML [J]. Journal of Suzhou Agricultural Vocational and Technical College, 2012, 30 (1): 22-25.

[8] Shao Weizhong, Yang Fuqing. Object-oriented system analysis [M]. Tsinghua University Press, 2012.

[9] Shao Weizhong, Yang Fuqing. Object-oriented system design [M]. Tsinghua University Press, 2012.

[10] Hu Ho-fen, Gao Fei. Course of Object-Oriented Analysis and Design in UML. Tsinghua University Press, 2012.