Software Reuse Model using Case-Based Object Approach

Young-Jun Kim
Department of Secretarial Studies, Cheonan College of Foreign Studies
ykim@mail.chonan-c.ac.kr

Abstract

In the step of system analysis, the goal of paper is to design a system of object model reuse and to verify the hypothesis in order to estimate it. This paper enhances the efficiency of software reuse through expressing object model utilizing cases, retrieving object model which is reused and developing object modeling procedure. We develop the methods and procedures of object model on the basis of case-based reasoning, and propose how to express the case of object model with it. Moreover, It expresses both the rules for retrieving the past object models needed for reuse and the rules for modifying object model in accordance with the new problems.

1. Introduction

A part of software engineering, software reuse has attracted the attention of the related academic circles because of the perspective that it can improve the productivity and quality of software development. Software reuse has meant source code reuse until now [4]. However, considering the fact that coding percentage is less than ten percents in the total cost for software development, source code reuse contains many limits in the process of improving the productivity and quality of software development, which is the original purpose of software reuse [10]. Owing to this problem, researchers' interests are concentrated on how to reuse the outputs in the step of system analysis rather than source code reuse itself.

As object-oriented technology is gradually developed, which has been discussed in good earnest since the late 1980s, some researches claiming this technology can maximize the reusing efficiency in the step of analysis are given; and they practically make this matter of reuse possible [2][12]. Recently, the researches on software reuse utilizing case-based reasoning have been shown as well; and it is the time when more active researches that can apply them should be accomplished [5]. This research is, in the step of system analysis, to design a system of object model reuse and to verify the hypothesis in order to estimate it.

2. Background

Software reuse has several merits that can reduce the efforts, cost and time for software development and make software management efficient, many researchers have so far studied this. The Intelligent Design Aid suggested by Lubars and Harandi [7], as the knowledge-base tool supporting the specification in the step of analysis, helps the specification demanded by a user to be consistently maintained. If the specification is input by a user, The Intelligent Design Aid retrieves knowledge-base by using the partial matching on data stream, generates the early specification, and reuses it through utilizing the rules; but the fact remains that a problem related to The Intelligent Design Aid is raised. That is, The Intelligent Design Aid has a fault that modifying rules is not easy when the knowledge about a new domain is added.

Computer-Aided Prototyping System is the system that reuse software component on the basis of a constructed specification [8]. Computer-Aided Prototyping System reuse automatically the information on software component using specification language; however, there exists some problems that this information are limitedly provided only in the specific field, and knowledge acquisition process is difficult.

Reuse Based Object-Oriented Techniques proposed by Farget and Morel, as the system expressing the process of component reuse in detail, shows the process of software component reuse that is based on object-oriented approach; and its ultimate goal is to reuse all outputs from object-oriented technology [3]. However, the problem is reuse procedure by it is very intricate.

3. Modeling of system analysis

Even if the step of analysis demands about six percents of cost and effort in the whole steps of the system, its result is decisively influential in whether the whole system becomes successful or not [10]. Thus, the productivity and quality of software development depends upon effective reusing outputs in the step of analysis. The research related to object-oriented technology is the most remarkable field, notwithstanding the fundamental solution for the matter mentioned above has not been offered yet. Object-oriented technology, unlike the existing other ones, attracts
researchers’ attention because this technology can keep consistency in whole step of software development, and use it again [9].

3.1. Modeling technique

In order to reuse the outputs from object-oriented analysis, we are to identify objects and the attributes of them in the given problem domain, and to express the association and inheritance among the identified objects. In this study, we will utilize class diagram by UML method for expressing object model [1]. UML class diagram is the thing which expresses mutually connected classes and their elements like figure 1.

![Figure 1. Class, association, aggregation, generation](image1)

3.1.1. Class, association, aggregation. Class means the objects that have similar data structure, behavior and association. In figure 1, the association means the relation between classes, expresses a straight line. The aggregation, a special form of the association, displays the relation of 'part-of'; and it means a class consists of several classes. For example, 'student' in figure 2 can be divided into 'undergraduate' and 'postgraduate'; 'postgraduate' can be subdivided into 'research assistant (RA)' and 'teaching assistant (TA)'. In this context, it means 'postgraduate' can succeed the attributes of 'student'; additionally, the association of 'student' can be automatically transmitted to 'RA' and 'TA'.

![Figure 2. Aggregation](image2)

3.1.2. Generalization, inheritance. Generalization means the method for sharing similar attributes, maintaining each attribute between classes. Inheritance is expressed in triangles and straight line. In figure 3, 'student' can be divided into 'undergraduate' and 'postgraduate' in accordance with whether they enter the higher level or not, there exists the generalization among them. A student can take several courses, and only one professor has to be in charge of a course. It should be confirmed which students applied for which courses; in this case, every student needs not take some specific subjects. The achievement in every course is evaluated by examinations; and these could be repeatedly carried out several times.

![Figure 3. Sample of object model](image3)

3.2. CBR technique

Case-based reasoning (CBR) is the method that seeks the past cases which are similar to the present problems through comparing the present problems with the past cases, applies the problem solving of the past to the present problem; and ultimately solves them [11]. In other words, it means a reuse for solving new problems through modifying the solutions used in the past. For this reason, many researchers become very interested in CBR as a good alternative that can surmount the problems in knowledge acquisition; because this method suggests automatic leaning through reuse in the process of solving problems.
In this study, cases are expressed by object and are stored in case-base. Object model management system, which can manage object is expressed, and knowledge-base is designed, which has the object model implementation rules and reuse rules. While CBR is a systematic effort for reusing the similar past experiences in order to solve new problems, software reuse is a systematic one for reusing the experiences and knowledge software development in the past for the purpose of new software development. Thus, they are essentially the same, and the problem solving process of them is similar to each other, except that software reuse is confined to the software field.

4. Rules of object model reuse

The rules for object model reuse are as following: the similarity rule, the rule for retrieving a similar object model, the rule for a retrieved object model reuse, the rule for modifying object model and the maintenance rule for adding object model to case-base and so on.

Class, attribute and operation can be shown as several terms with similar meanings in object model. Thus, the names of class, attribute and operation should be expressed in similar terms. For instance, class 'customer' can be expressed as 'member' or 'client'; then these similar terms are stored, in the set of similar terms, with similarity value showing similarity degree of the term 'customer'. The similarity of object model can exist between association as well as class. For example, assuming that there is an object model about 'Class management of school A' among the existing object models, the class 'member' of object model 'Membership management in center B' can add the class 'student' of the object model 'Class management of school A' to the set of similar terms as well as the existing one.

4.1. Retrieval and reuse rules

In CBR, how to express cases, how to construct case-base, how to retrieve the constructed cases and how to modify them are important. The retrieval step of this reasoning needs retrieval rules and the related knowledge. The general retrieval step can be divided into 2 sub-steps as following: the step for calculating similarity value which displays the similarity between problems and cases, the step for selecting case by using similarity value of each case. The total similarity value for each case in figure 4 is produced by class and association similarity value.

For the residual part of this study, we will define the terms that are used soon as followings: Problem object model (POM) means the object model that is the most similar to POM among COMs. Initial Object Model (IOM) means the new object model generated by applying reuse rules. Final Object Model (FOM) means the object model finally completed through modifying process.

4.2. Modification and maintenance rules

Modification rules mean the rules for modifying the part of model that users are not satisfied with among object model. The algorithm of reuse and modification rule, in figure 5, is displayed; and these rules are basically applied for the specific class or association in object model. In the case that the IOM generated by system can't fulfill POM, some modifying alternatives are produced from object models excluding ROM, the modified alternative whose similarity value is the highest is selected from them, and IOM is modified by it. Object model is gradually competed under this process.

<table>
<thead>
<tr>
<th>Total Similarity Value $TSV = \Sigma C_{si} + \Sigma A_{sj}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{si}$ : similarity value of class $i$</td>
</tr>
<tr>
<td>$A_{sj}$ : similarity value of association $j$</td>
</tr>
<tr>
<td>$M$ : class number of case</td>
</tr>
<tr>
<td>$N$ : association number of case</td>
</tr>
</tbody>
</table>

Figure 4. Total similarity value (TSV)

Reuse rules are applied when the IOM is generated object model reuse by retrieval rules. If it discovers the class of POM that is similar to the specific class of ROM, using class similarity value calculated by this rules, it can combine two classes into one; and the class name here uses the name of POM.

Procedure Generate_rules(Case_Base)
Reuse_Rule_Set = ;
Modification_Rule_Set = ;
For $I = 1$ To Case_Number -1
  Problem = Case_Base($i$)
  For $J = I + 1$ To Case_Number
    Target = Case_Base($j$)
    Diff = Calculate_DiffOfAttributes(Problem, Target)
    If Diff <= Threshold Then
      Single_Rule = Make_Reuse_Rule(Problem, Target)
      Modification_Rule = Single_Rule U Retrieval_Rule
      Next J
    End If
  Next J
Next I
Refine_Reuse_Rules(Reuse_Rule_Set)

Figure 5. Algorithm of modification rules

Maintenance rules mean the rules for adding the completed object model to case-base by applying retrieval, reuse and modification rules. The CBR for object model reuse adds a new object model to the present case-base
through maintenance rules; and it makes the automatic learning for knowledge acquisition possible. Therefore, maintenance rules can be applied to the new problem domains without any additional knowledge acquisition, and this rules enables knowledge extension of CBR for object model reuse to be easier.

5. Design of COMOR system

Now we will design COMOR System (Case-based Object MOdel Reuse System), applying several rules that have been surveyed so far. COMOR System supports the process of object model reuse, utilizing retrieval, reuse, modification and maintenance rules of CBR.

5.1. Composition of system

This system offers graphic user interface and minimize the time and efforts adding cases to case-base for the effective reuse of object model. This system is mainly composed as followings: Graphic User Interface (GUI) is expressed a UML class. Object Model Management System (OMMS) has the function of reasoning and management and generates the object model using case-base and knowledge-base. Case-base stores the object models of the past. Knowledge-base stores class diagram knowledge and the various knowledge needed in reasoning process of OMMS.

5.2. Procedure of system

The Procedure of system, like in Figure 6, generates IOM by applying retrieval, reuse and modification rules, if POM is input by the user. In the next step, modification rules for IOM are applied again, or FOM complete through a direct modification by the user. The user inputs object model about the related work specification through GUI. This object model can exist as various forms in accordance with the degree of users' experience and efforts for modeling; namely, from the object models that has some simple classes to the perfect ones, various object models can be input.

If POM is input, OMMS retrieves the object model that is the most similar to POM from case-base, using retrieval rules in knowledge-base. This system can generate IOM with applying reuse rules in knowledge-base. If the user isn't satisfied with IOM, he can require modification after he chose a part of IOM; OMMS here will retrieve whether there is the similar object among object models or not, and offers the user the object as an alternative, which has similarity value beyond the specific standard. After the repeats of this process, if the user's requirement disappears, OMMS makes FOM be completed through reuse rules in knowledge-base.

5.3. Design of system

An example for system design is 'Educational management of college C', and its work specification is as followings. The goal of college C is to be well-rounded national and world citizens, as true freemen to perform their jobs ably and professionally and to practice the love and sacrifice needed to fashion a stable world. It educates and conducts research in professional knowledge and theory, specializing in teaching foreign language and practical skills for the workplace. The college consists of academic affairs, student and general and planning and management affairs, library, computer center and so on. A student can take several courses, and only one professor has to be in charge of a course. It should be confirmed which students applied for which courses; in this case, every student needs not take some specific subjects. The achievement in every course is evaluated by examinations; and these could be repeatedly carried out several times.
This COMOR System supporting the object modeling of the college C through this work specification stores the object models in various fields such as transaction, order and membership managements. IOM of this system is generated on the basis of POM. it can be extended by modification rules and FOM is completed through this processing. First of all, like in figure 7, the user inputs class, attribute and operation through GUI, which are identified from the specification of the college C. The elicited classes can be variable in accordance with the users' understanding degree for the work specification.

5.4. Expected significance

The merits of this system will bring about the effects of reducing the efforts and cost, which are expected on account of shortening the time required for software reuse. This system, offering GUI which expresses the outputs in the step of system analysis, will enable object modeling to be easier than source code reuse. At last, as this system will allow the users to avoid the problems of knowledge acquisition, it helps them to easily develop the new object models. Namely, software reuse by this system reduces software developing time and efforts, consequently improves productivity; and it is expected that the natural knowledge of this system can enhance software quality and carries out the efficiency of software management.

6. Discussion

This paper for a software reuse, it designs the reuse model with CBR in order to reuse object model, the object-oriented outputs in the step of system analysis. We develop the methods and procedures of object model on the basis of CBR, and propose how to express the case of object model with it. Moreover, this research expresses both the rules for retrieving the past object models needed for reuse and the rules for modifying object model in accordance with the new problems. Furthermore, this research confirms its adequacy through applying the system for object model reuse to the real cases.

We claim this research is very meaningful in terms of these facts. This research solves the problems of knowledge acquisition by applying CBR, which is regarded as the most difficult problem in object model reuse. This research offers the model that enlarges the extent of reusing CBR not to the dimension of source code but that of outputs in the step of system analysis.

References