아날로지를 기반으로 한 객체모델의 재사용

배 제 민†

요 약

소스 코드 재사용은 다른 개발자에 의해 만들어진 코드를 정확하게 이해하거나 검색하기 어렵다는 점에서 몇 가지 제한점을 갖는다. 이러한 문제점을 해결하기 위해서 소스코드 자체보다는 분석 및 설계 정보를 재사용하는 것이 가능해야 한다. 이에 본 논문은 객체 모델 및 패턴을 재사용하기 위해 필요한 analogical 매칭 기법을 제안한다. 그리고 객체 모델과 디자인 패턴을 재사용 컴포넌트로서 저장할 수 있도록 표현하는 방법을 제안한다. 즉, 재사용 라이브러리에 저장된 유사 컴포넌트를 검색하는 analogical 매칭 함수와 매칭을 지원할 수 있는 라이브러리 구조 및 재사용 컴포넌트의 라이브러리내 표현 방법에 대해 기술하였다.

키워드 : 객체모델, 패턴, 재사용, Analogical 매칭, 컴포넌트 라이브러리

Analogy-based Reuse of Object Model

Je-Min Bae†

ABSTRACT

Code reuse in software reuse has several limitations such as difficulties of understanding and retrieval of the reuse code written by other developers. To overcome these problems, it should be possible to reuse the analysis/design information than source code itself. Therefore, this paper present analogical matching techniques for the reuse of object models and patterns. And this paper have suggested the object model and the design patterns as reusable components and the representation techniques to store them. Namely, the contents of the paper are as follows. Analogical matching functions to retrieve analogous components from reusable libraries. And the representation of reusable components to be stored in the library in order to support the analogical matching.

Key Words : Object Model; Pattern; Reuse; Analogical Matching; Component Library

1. INTRODUCTION

The reuse of software can contribute to improve the productivity by reusing previous development experiences to the development processes of new software [15]. Up to this point, efforts for the reuse of software have been focused on reuse at the code level, being either reuse of the algorithm library or the reuse of the class library [10]. In order to overcome the limitations which comes with reuse of code [2, 3], continued effort is being exerted for the reuse of the data material created within the analysis /design phase previous to the coding phase.

Recently, attempts for software development with object -oriented methodologies applied have increased in accordance to activation of object-oriented methodologies [2, 10]. For devclopers with no object modeling or component modeling experience, it is not an easy task to discriminate the appropriate objects and grant the specific attributes or behaviors for the problem at hand [11]. If the models in a similar domain can be referenced, the developer easily can understand the problem. With this understanding, the developer can make new model. Consequently, rather than the reuse of code dependent of the environment and language, the reuse of abstract models are required.

In order to support the reuse of object models within the requirement specification level, this paper proposes a reusable environment for object models through analogical matching by introducing the analogy technique[4, 7, 12, 16] being researched in knowledge engineering. The analogy technique is different from the similarity measurement technique being used for the reuse of existing codes.

Although similarity refers to a characteristic or intrinsic commonness which exists between two objects, analogy refers to the consistent structure or confronting nature between two objects which exists from all point of views

† 종신회원 : 관동대학교 컴퓨터교육과 교수
논문접수 : 2007년 5월 14일, 심사완료 : 2007년 7월 25일

[15, 17, 18].

In order to support the reuse of object models, this paper proposes an environment which reuses object models and patterns based on analogies by introducing the analogical matching technique. Section 2 will describe the techniques necessary for reuse and section 3 will define the library structure and representation technique for model reuse and also define the textual matching technique for patterns as well as the semantic matching techniques based on analogical matching techniques. Section 4 will describe and evaluate the implementation of matching agents which searches for suitable models from the constructed reusable library in a given situation. Section 5 will provide a conclusion.

2. Existing Research for Model Reuse

Currently, research for model reuse can largely be classified into reuse of specifications, reuse of functionality centered products (ex. DFD), and the reuse of object models[15, 16].

The majority of the research uses the SME (Structure Matching Engine) and the ACME (Analogical Constraint Matching Engine) based on a Tree format internal representation technique founded on the structural relatedness of products. For this, the input material for SME and ACME are applied by expressing the knowledge representation structure of models in a mathematical language. Also, analysis and a tool have been developed in order to support the process of reusing the matching results created in SME and ACME as object models.

Rather than knowledge representation which is the format which satisfies the input conditions of SME and ACME, this research supports the reuse of object model by taking the object model itself as the input material. For the reuse of software artifact as proposed by Maiden, a representation technique which expresses patterns, object behaviors, and pattern behaviors based on the required issues of consideration and not only the unit objects is proposed, and the technique which evaluates the similarity between patterns and behaviors is also proposed. Different from Whitehurst, a similarity evaluation technique which takes the various object associations and object behaviors into consideration is being proposed.

3. EXAMPLE OF ANALOGY BASED REUSE OF OBJECT MODEL

We have two problem statements for object modeling. Two problems are quite different: different objects and different relationship as well as different goals of system. But we can find there are somewhat similar between them even we can say exactly what they are.

3.1 Library Application

Remote lending department of a university library often receives orders for copies of scientific papers. The research staff is the most frequent user of the service, but occasionally the department receives requests for copies from students and technical staff as well. On receiving an order, the department gets a reference to an article and in addition the name of the requester and his/ her account number. The order also tells whether a photocopy, microfiches or microfilm of the article is wanted. The librarian looks up the borrower in a register. This register is computerized, and the librarian has access to it through a networked personal computer. In the register, the librarian finds the address of the borrower The reference gives information on which book or journal he article appeared in. The librarian looks up the information in a punching card archive containing all the books and journals in the library. Thereafter, a copy mail system of the university. Finally, the lending register is updated with the information that the borrower has received a copy of the article.

If the book or journal containing the article is not available, there are two options:

The book/journal exists in the library, but is lent out past the allowed time. In this case, the librarian sends an overdue notice.

The book/journal dose not exists in the library. In this case the librarian forwards the order to other university libraries, both domestic and foreign.

If the borrower is not registered in the borrower database on the PC, the librarian verifies the identity by making a phone call to the borrower's department. Then the borrower is contacted by phone and immediately registered in the database as a new 'customer.' The request is then processed as usual.

3.2 Wholesale Application

A wholesale dealer, selling marine outboard motors daily receives several orders for spare parts. most orders come from retailers, but occasionally there is private customer needing spare parts. Together with the order, the dealer receives both the name and the account number of the customer. The account number is later verified against his or her name. In addition the order tells which part are requested in the order and how the order should be dispatched. the stored assistant looks up the customer in a register. This register is computerized, and the assistant

accesses the register using a networked PC.

In the register, the assistant gets the address of the customer. For private customers, the assistant checks if he can find any nearby retailers. Customers living within a reasonable distance from a retailer are referred to nearest retailer. Afterwards, the assistant looks up the part in a punching card archive containing all the parts in the warehouse. If the right part is found, the punch card for the part is put in a separate archive for parts sold. The ordered part is then delivered as previously decided. Finally, the customer register is updated with information concerning the dispatched order.

If the right part is not found in the warehouse, there are two possibilities:

The assistant checks with the store manager if there is a suitable substitute for the part in the warehouse. The wholesale dealer has a policy to give a discount on expensive substitute parts when requested parts are sold out or cannot be found.

The requested part is ordered from the factory.

If the customer cannot be found in the register, the assistant gives him/her a phone call to confirm the order. Then the customer is given an account number, and the order is dispatched in the usual way. Retailers are asked to pay their debts quarterly, while private customers are invoiced immediately.

We can easily think two problems have very similar structure and properties even if different words and different statements are used to describe the problems. Both problems have their own resources and policy to handle it and people who need what they have.

4. ANALOGICAL MATCHING TECHNIQUE

The structure and representation technique to store objects and patterns as components in the library as well as the technique based on analogical matching which allows the searches of library have been defined.

4.1 Conceptual Model of Component

Regarding object models subject for reuse, the scale and abstraction for the object model have each been classified into two levels. Regarding the scales of the reuse unit, the subjects were classified by the smallest objects and larger size reuse pattern. In addition, the schema level and case level were differentiated in accordance to the abstraction level for the component subject to representation. Schema is defined as the abstract representation of a model, and is also referred to as abstract model, abstract representation, or template. A case can be defined as an actual example regarding a certain concept.

4.2 Library Representation for Models

Evaluating the analogy of the subjects for reuse is the same as saying that the static and dynamic characteristics of the two subjects are being evaluated. By reconfiguring the OMG standardized reference model [16] for object models, the technical format for each subject of reuse can be defined. Regarding the simple structure for a model, the static structure of the model is defined through the signature, from which the dynamic characteristics can be defined through specification. For patterns, the pattern signature and pattern specification is defined for the association of the objects.

4.2.1 Object Signature

In order to conduct analogical matching by saving the object, the minimum unit comprising an object model, into the reuse library, the object is expressed with the following characteristics as shown in (Fig 1).

As the ObjectID is the identifier of objects registered in the reuse library, it is a number created as an identifier within the library while the ObjectName is the name of the object given within the object model. The Set-of-Attributes is the collection of static attributes given to objects, and is the recording of only the names of such attributes. The Set-of-Operations is the collection of interfaces which displays the services of objects. The ObjectKind is the definition of the type of objects. Among the object types already defined this is the representation of what class the objects belong to. They represent either the information regarding the domain within the design library or is represented as the Is-a-lattice within the section recording general information, and can be continuously added.

4.2.2 Object Specification

As a representation of the state or behavior concepts possessed by an object, the Object Specification area is defined by adding the syntactical characteristics or the meanings of each interfaces of the object. This is a section which falls under dynamic modeling of objects.

```
Object = ( ObjectID, ObjectName, set-of-Attribute, Set-of-Operation, ObjectType )
ObjectType = resource | process | history | logic | role | interaction
Attribute = ( ObjectID, AttributeName, AttributeType, AttributeStructure )
AttributeType = identifier | descriptive | referential
AttributeStructure = simple | composite
Operation = ( ObjectID, OperationName, ResultType, Set-of-Argument )
Argument = ( OperationName, ArgumentName, ArgumentType, ArgumentStructure )
ArgumentStructure = simple | composite | object
```

(Fig. 1) Representation of Object Signature

(1) Representation of State Domain

Among object behaviors, the state which represents one level can be represented through a combination of attributes defined syntactically. They are defined as (Fig. 2).

```
State = (set-of-StateDomains, set-of-Invariants,
         set-of-StateConstraints)
StateDomain = AbstractState | Predicate
AbstractState = abstract concept of state
Predicate = concrete predicate of state
Invariant = always predicate
StateConstraint = (AttributeName, Relation, SourceAttribute)
```

(Fig. 2) Representation of State Domain

The state domain is an important element which defines the meaning which the objects possess within an application. Even objects which possess the same name and operation may possess differing state domains in accordance to the requirements of the application.

The state expressed in the dynamic domain is generally expressed as an abstract concept defined through an association of various attribute values, referred to as the AbstractState. In addition, they many be defined as a definitive predicate in order to deliver a more precise meaning. Invariants, necessary to define the intrinsic characteristic of models and carrying the characteristic that they must be satisfied with any behavior the object may operate, are expressed with the condition that they must be satisfied together with the reserved word, always.

Constraints, similar to Invariants, define the relative associativeness between one attribute and another attribute. They are expressed by defining relationally what characteristic a certain attribute possesses in comparison to another attribute.

(2) Representation of Behavior

Object behaviors, along with object states, provide important dynamic meanings for objects, and can be viewed as the transition of a certain object from one state to another state through an event. This is defined as (Fig. 3).

```
Behavior = (EventName, Pre-State, Post State, set-of-UpdateAttributes,
set-of-BehaviorConstraints)
Pre-State = StateDomain
Post-State = StateDomain
UpdateAttribute = AttributeName
BehaviorConstraint = (EventName, Relation, EventName)
```

(Fig. 3) Representation of Behavior

The message which corresponds to the state and defined conditions converts it's own state by operating the behavior set to be conducted by the object. Behaviors are defined through Post State which becomes defined after operating the preceding condition Pre-State and behaviors which first need to be satisfied in order to operate the behavior and EventName of an event. The names of attributes which change in value during a behavior from the attributes defined to itself are referred to as UpdateAttributes. The Constraint is the representation of the correlational relationship between behaviors and records the occurring sequence between events.

4.3. Analogical Matching Function

In this passage, the logic necessary to examine the analogy between subjects will be recorded. Evaluating the analogy between two subjects is saying the static characteristics and the similarity of the dynamic characteristics of the two subjects will be evaluated. For this, the matching for objects and the matching for patterns have each been defined.

Regarding the static characteristics of objects, the interface matching function which evaluates the consistency level between interfaces have been defined, and the specification matching function has been defined in order to evaluate the dynamic characteristics of objects. In the case of patterns which defines the mutual application between objects, the pattern structural matching function, which evaluates the associative consistency level for objects comprising a pattern and the pattern specification matching function, which evaluates the behavioral consistency level in accordance to the correlational relationship between elements comprising a pattern, have been defined.

First, the general analogical matching is defined as (Definition 1).

(Definition 1) Analogical Matching function between Query (Q) and Component (C)

$amatch(Q_{signature}, C_{signature}) \vee amatch(Q_{specification}, C_{specification}) \vee amatch(Q_{pattern}, C_{pattern}) \vee amatch(Q_{framework}, C_{framework})$

An analogy between two subjects is determined to exist if the signature or specification on object level between query models and component models is similar or if the interface or the mutual application of patterns on pattern level is similar. The fact that an analogy exists between a query and component means that a successive transformation rules which can be applied to either the query or component exists. The basic matching function necessary for this has been defined as follows.

(Definition 2) Basic Matching function

(1) Type Equality $T_1 \equiv_e T_2$

① T_1 and T_2 has same name

② T_1 and T_2 has same type

③ if $T_1(e_1,e_2,...e_n)$, $T_2(m_1,m_2,...,m_m)$ are complex type
$\forall e_i, \exists m_j \cdot e_i \equiv_e m_j, (1\le i\le n, 1\le j\le m)$

④ if $T_1(e_1,e_2,...e_n)$ is complex type
$\exists e_i \cdot e_i \equiv_e T_2 \quad (1\le i\le n$

(2) Renaming $E \equiv_r E'$

① $R(E) \equiv_r E'$ or

② Is $a(E, E') \in$ DesignLibrary

(3) Type Equivalence $T_1 \equiv_q T_2$

① $T_1 \equiv_r T_2$

② if $T_1(e_1,e_2,...e_n)$, $T_2(m_1,m_2,...,m_m)$ are complex type
$\forall e_i, \exists m_j \cdot e_i \equiv_r m_j, \quad (1\le i\le n, 1\le j\le m)$

③ if only $T_1(e_1,e_2,...e_n)$is complex type
$\exists e_i \cdot e_i \equiv_r T_2, \quad (1\le i\le n\ )$

(4) Matching by reOrdering $O1 \equiv_o O2$
$\forall e_i\in O_1, \exists m_j\in O_2 \cdot e_i \equiv_e m_j, \quad (1\le i\le n, 1\le j\le m)$
O : Parameter Set | Attribute Set | Operation Set
e_i, m_j : ParameterName | AttributeName | OperationName

The (1) in (Definition 2) is the definition of the standard which evaluates whether the type of the two subjects are consistent, while (2) verifies whether a homogeneosity can be recognized even if the names and data types of the two subjects for matching are not the same. In addition, in (3) of (Definition 2), the level which can possess a semantic homogeneosity has been defined, even if the two subjects for matching are not completely consistent, while in (4), because the parameter, attribute name, and operation name are represented as a collective concept, it has been defined that matching can occur irrelevant to the sequence which has been recorded.

Besides the function necessary in order to transform the syntactical characteristics represented in the signature, semantic matching based on the specification is necessary in order to determine the objectives and intentions of the model.

The basic concepts used for semantic matching is homomorphism and isomorphism[8]. (Definition 3) is the definition of general homomorphism and joint morphism.

(Definition 3)

$\Sigma=(S,\le,F)$: signature , $A,B\in Alg(\Sigma)$.

Σ homomorphism $\phi:A\to B$ is function satisfying the following condition.

① $\phi_s(f^A_{s1...sns}(a_1, ..., a_n)) = f^B_{s1...sns}(\phi_{s1}(a_1), ..., \phi_{sn}(a_n))$
for all $f:(s_1, ..., s_n) \to s\in F$, $a_i\in A_{si}$, i=1, ..., n,

② $s\le t$ implies $\phi_s(a)=\phi_t(a)$ for all $a\in A_s$.

In (Definition 3) of above, ① is the definition of homomorphism, which means that mapping which responds to B can be discovered in accordance to the successive mapping function in regards to the mapping defined in A, whereas ② is the definition of joint morphism, which means that not only can A respond to B, but B can respond to A, as well.

4.3.1 Object Signature Matching

If the type of object is completely consistent or if judged to share homogeneosity based on domain information defined in the design library, the objects is considered similar. In addition, the two objects are judged to be similar if there is either consistency or homogeneosity between the define attributes or if the characteristic between operations are considered homogeneous.

(Definition 4) Complete Matching Function of object queries vs between object components

Exact_amatch($OQ_{interface}$, $OC_{interface}$)

① (ObjectKind(OQ) $\equiv_e$ ObjectKind(OC) $\vee$ ObjectKind(OQ) $\equiv_q$ ObjectKind(OC)) or

② ($\forall$attribute$_q\in$set of Attributes(OQ), $\exists$attribute$_c\in$set of Attributes(OC) attribute$_q \equiv_e$ attribute$_c \vee$ attribute$_q \equiv_q$ attribute$_c$) or

③ ($\forall$operation$_q\in$set of Operations(OQ), $\exists$operation$_c\in$set of Operations(OC) operation$_q \equiv_e$ operation$_c \vee$ operation$_q \equiv_q$ operation$_c$)

(Definition 4) is to distinguish cases in which all of the characteristics represented within the query are represented in the component. In this case, things not represented within the query can exist within components, and the query can be redefined based on the additionally representations of the components.

(Definition 5) Partial Matching Function of object queries versus between object components

Partial_amatch($OQ_{interface}$, $OC_{interface}$)

① ($\forall$attribute$_c\in$set-of-Attributes(OC), $\exists$attribute$_q\in$set-of-Attributes(OQ) attribute$_q \equiv_e$ attribute$_c \vee$ attribute$_q \equiv_q$ attribute$_c$) or

② ($\forall$operation$_c\in$set-of-Operations(OC), $\exists$operation$_q\in$set-of-Operations(OQ) operation$_q \equiv_e$ operation$_c \vee$ operation$_q \equiv_q$ operation$_c$)

In the case of (Definition 5), although everything represented within the component can respond to query elements, this is a case in which some of the elements represented in the query may not be represented within the component. However, it is still considered similar because the query can be either redefined through the component or reconfigure the query by inheriting the components. This

is defined the as partial matching function.

(Definition 6) Partial Matching Function of object queries versus between object components

Partial_amatch($OQ_{interface}$, $PC_{interface}$)
($\exists attribute_q \in$ set of Attributes(OQ),
$\exists object_c \in$ set of Objects(PC)
$attribute_q \equiv_c Object_c \vee attribute_q \equiv_q Object_c$)

In (Definition 6), this is the partial matching function which determines the structural similarity between queries written through objects and between patterns saved within the reuse library. This is a function which identifies instances of when certain attributes represented in certain objects are represented as independent objects in a different model.

4.3.2 Object Specification Matching

In order to determine the behavioral similarity between objects, Object Specification Matching is something which determines the level of consistency and behavioral consistency regarding the state domain for objects. In addition, in cases when restrictive conditions regarding the attributes of an object exist, the consistency of such conditions must be considered, as well.

(Definition 7) amatch($OQ_{behavior}$, $OC_{behavior}$)
if Pre State($operation_q$) $\equiv_c$ Pre State($operation_c$) or
Pre State($operation_q$) $\equiv_q$ Pre State($operation_c$)
thenPost State($operation_q$) $\equiv_c$ Post State($operation_c$) or
Post State($operation_q$) $\equiv_q$ Post State($operation_c$)

(Definition 7) refers to the similarity of the method of response for similar operations in similar states. Although is represented as an abstract state concept (Full, Waiting for something, etc) in most object models, it is also represented as a specific conditional expression (x.time < 100 $\wedge$ y.interval > 10). Instances of representation as a specific conditional expression such as this apply techniques used in existing techniques for reusing specifications. In existing reuse of specifications, the existing specification is defined to be possible for reuse if the implication principle between the preceding conditions and following conditions can be established. The matching for the entire dynamic characteristics of an object has been defined as (Definition 8).

(Definition 8) Behavior Graph BG(V, E)
V = {v_i | v_i is AbstractState}
E = {(u, v, l) | (u, v $\in$ V) $\wedge$ (l is EventName)}

Behaviors which demonstrate the dynamic characteristics of objects can be expressed as a directional graph. In (Definition 8), the state represented in behaviors have been expressed as nodes on the graph, and the transition from one state to another have be expressed as a direction edge. The event name is expressed as the edge label.

(Definition 9) complete matching function of object specification Exact_amatch(BGQ, BGC)
$\forall (u,v,l) \in BGQ \cdot \exists (f(u), f(v), f(l)) \in BGC$,
(단, $f : E \rightarrow E^*$)
// BGQ(V, E) : Object Behavior Graph for Query
$BGC(V^*, E^*)$: Object Behavior Graph of Component

By representing object behaviors through graphs, the analogical matching of object behaviors can apply concepts regarding homomorphism and joint morphism on the graph.

The full matching of object specification signifies that each state transition principles as defined in queries can correspond to the state transition principles of components and as defined in (Definition 9) refers to instances when the behavioral type of an object is completely consistent, However, since these types of matching through isomorphism is too limiting, (Definition 10) proposes partial matching through homomorphism.

(Definition 10) Partial Matching Function for Object Specification Partial_amatch(BGQ, BGC)
① $\exists (u, v, l) \in BGQ \cdot \exists (f(u), f(v), f(l)) \in BGC$, $(f: E \rightarrow E^*)$
② Exact_amatch(BGQ', BGC) exists for BGQ'
③ Exact_amatch(BGQ, BGC') exists for BGQ'
// BGQ'(V',E') : extended graph of BGQ
$V' = V \cup \{v_i\}$,
$E' = E \cup \{(v_j,v_i,l'), (v_i,v_k,l'')\}$
($v_j,v_k \in V$, $v_i \notin V$, $(v_j,v_k,l) \in E$, (v_j,v_i,l'), $(v_i,v_k,l'') \notin E'$,
l',l'' : newly defined event)
BGC'(V',E') : extended graph of BGC
$V' = V \cup \{v_i\}$
$E' = E \cup \{(v_j,v_i,l'), (v_i,v_k,l'')\}$
($v_j,v_k \in V$, $v_i \notin V$, $(v_j,v_k,l) \in E$, (v_j,v_i,l'), $(v_i,v_k,l'') \notin E'$
l',l'' : newly defined event)

The ① in (Definition 10) signifies that among the behavioral types which exist fundamentally in queries, even if one of these types can be discovered in the component. it is proposed to be a reuse model candidate. With this foundation, in ② and ③ of (Definition 10), by adding new states(v_i) within the behaviors of queries or

the behaviors of components, if full consistency between the behaviors of queries and the behaviors of components occurs, then a partial matching between the two can be thought to be established. By adding new states and defining new state transition principles, this is to determine whether isomorphism can be defined between the two behavioral graphs. Through this, reuse candidates can be identified by carrying similar behavioral structures, even if they possess differing state domains or if the transition principles are not completely consistent.

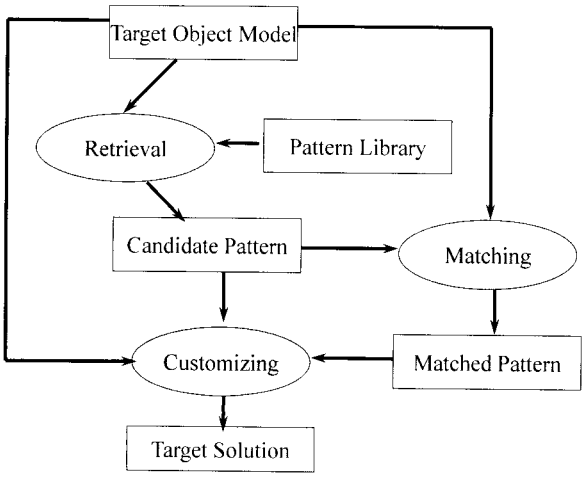
(Definition 11) Behavior matching function with behavior comformance ($Q_{behavior}$, $C_{behavior}$)

① Pre-State(Q) ⇒ Pre-State(C)

② (Pre-State(Q) ⇒ Post-State(Q)) ⇔ Post-State(C)

4.4 Analogical Agent

Based on previous defined analogy matching function, we are constructing the analogical agent. (Fig. 4) show the process model of agent. The developers construct their initial query with object model. Then, the model interpreter analyzes the model and transforms the internal representation for matching. During this step, the domain knowledge of design library can help the transformation. If the domain knowledge were not enough, the agent can interact with the modeler for further domain information. If the internal representation made, the analogical matching engine retrieves the candidate models from the library. So, new developer can customize her model with previous models and identify what models are better. Also, they can know how to define the relationship among the classes. We are developing the first prototype with architecture define in (Fig. 5). In the library, object, pattern and framework are stored and associated sematinc information in terns of signature and specification. Agent retrieve the candiate model stored in library comparing with user model using mathing engine

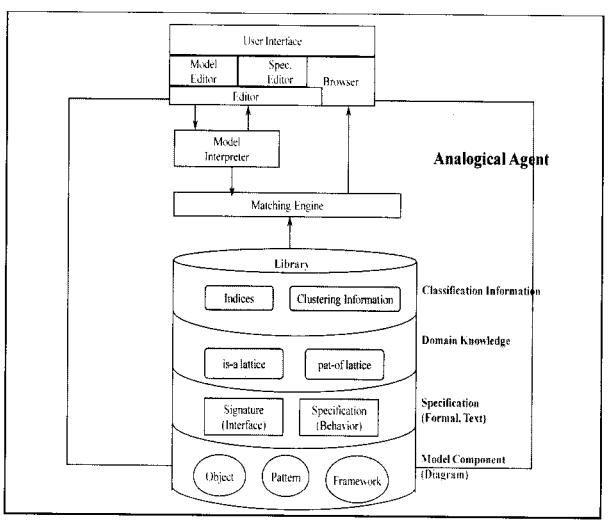


(Fig. 4) Flow Diagram of Reuse System

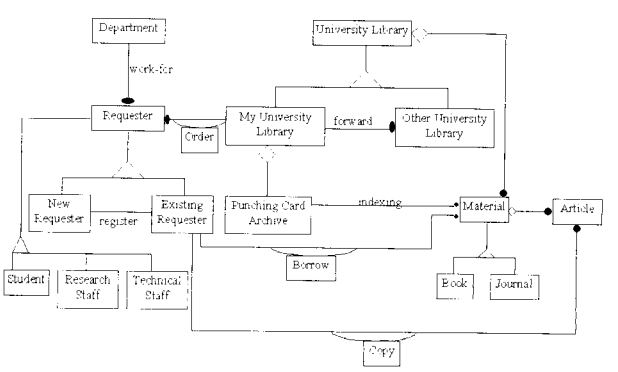
4.5 Case Study

We showed simple case study for revising the exsiting object model based retrieved by analogical reasoning agent. It is based on problem staement written in section 3. In the library, there are order/delievery object model whose structure is similar with library model even though two are the different application and their object and relations are different. But there are relativeness among the strucutre and behavior. For example, item to be delivered by customer corresponds to book to be borrowed by member. In other words, two different model are analogous. During the modeling the library application, order/delievery object model can be good design pattern for modeling.

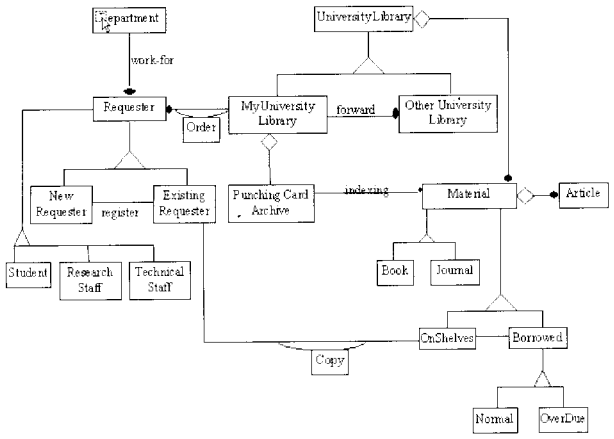
In our case, (Fig. 6) is the first object model for library system. Based on this object model, analogical agent retrieve the library for extracting analogous object model and get the order/delievery object model. With this, object



(Fig. 5) Architecture of Analogical Agent



(Fig. 6) Original object model for library system



(Fig. 7) Revised object model

modeler revied original model. For example more detail object model from requester can be identified and new association object can be identified. Revised result was shown in (Fig.7).

5. Evaluation

In order to verify the efficiency of the reuse techniques proposed in this paper, evaluation was conducted through widely used evaluation standards of recall and precision.

5.1 Evaluation through Recall and Precision

Recall is the evaluation standard which tests the capability to remove unrelated components from the search and is defined as a ratio for components which are suitable with the actual query among the searched components. Precision is the evaluation regarding the search capability for related components. It is defined as a ratio of the searched components from all of the components which satisfy the query[14].

<Table.1> presents the results for full matching and partial matching regarding both the object unit search and pattern unit search.

〈Table. 1〉 Evaluation Results in accordance to Recall, Precision

(a) Evaluation Results for Recall

	Object Search
Full Search	56.3
Partial Search	75.8

(b) Evaluation Results for Precision

	Object Search
Full Search	78.6
Partial Search	66.3

The evaluation standard dependent upon recall and precision satisfied average standards[14] upon application to the system constructed in this paper. In terms of precision and search scale, it was found that cases for object searches were better than cases for pattern searches. Although the matching results is not largely influenced by the designer as there are only one match subject for an object, with patterns, on the other hand, even if the same associativeness is defined it is completely up to the designer to determine what types of mutual applications will occur between the objects through such associativeness and which behaviors will be shared.

This is an issue which cannot be completely resolved even by applying homomorphism within signature matching of patterns or by conducting matches based on pattern specification. The precision of pattern matching can be seen to decrease even within the evaluation results. This is because the matching function was applied centered on the pattern structure in the matching for pattern specification, and due to the fact that characteristics of the classes which comprise a pattern are attempting to solve different problems. However, with partial matching, a satisfying recall rate was acquire even in the case of patterns, and through this base, a model required within a problematic domain can be completed through user dependent modifications.

4.2 Comparison with different classification techniques

Models and pattern searches based on matching techniques carry the following advantages when compared to the facet technique or decimal classification used in existing search systems. In the case of the Facet technique or the decimal classification technique, the library developer writes the hierarchy between models by first possessing a classification standard of models and deciding the terms included between Facet and Facet. However, when examining examples proposed for Facets regarding actual software, such as Function, Object, Media, Language, Environment, and others as proposed by Diaz, they represent either the possibilities for operation of a function or subjects for application. In other words, it is extremely difficult to have a set of already defined terms or facets. Also, it is also difficult to define the hierarchy between models. In other words, defining the conceptual or implementational hierarchial structure between models is difficult to do. From this viewpoint, it is difficult to apply the facet technique or the hierarchical classification technique, which defines a fixed frame, during the search or reuse process for a model in which th the intentions of the developer are expressed. In comparison to this, after saving a well-established

model or previously verified model into the library, the matching technique allows the user who is writing a model within a new domain to conduct searches by matching with similar models already saved in the library, which can be used to serve as a foundation.

6. Conclusion

The reuse of models carries many advantages. As models show the abstract concept in regards to a problem, it facilitates in the understanding of the problems within a corresponding area by providing knowledge and a problem solution viewpoint for a corresponding domain for developers. inexperienced in regards to a problem faced during a new development. In addition, by referencing existing models during the modelling phase, the experience from previous modelling can be reused and consequently will assist in correct construction of the model.

When writing object models regarding new problems, this research proposes a method allowing for the reuse of models and patterns within the library by determining the analogy regarding queries and components in order to provide an environment which allows reuse of previous experience during the abstraction process of models and during the process establishing associativeness between objects. In order to reuse models, patterns, which include an object, the smallest unit of object models, and the associativeness between objects, were used to distinguish subjects for reuse and to establish the representation technique necessary for saving each one. The analogical matching technique was applied since the technique for the reuse of models differs from the technique for reuse of code, and consequently, is not possible to determine the similarity level through full consistency of the components. For this, the necessary information structures and the functions necessary to evaluate matching were defined by subjects for reuse. In addition, the analogical agent was designed in order to provide an environment which provides models suitable for users during the model write up process. Developers can write partial models centered around objects distinguished for the first time upon confronting a problem and with this foundation, similar objects from the library can be searched, and from this, the objects can be distinguished and information necessary in order to define the characteristics of objects can be received. By repeating these types of processes, the objects can be distinguished, the associativeness between the distinguished objects can be defined, and the pattern unit can be searched from the library

References

[1] R. Breu, 'Algebraic Specification Techniques in Object Oriented Programming Environment,' Spring-Verlag, 1991.

[2] Peter Deutsch, 'Design reuse and framework in the Smalltalk-80 system,' In Ted J. Biggerstaff and Alan J. Perlis, Editors, Software Reusability (II), 57-72, ACM Press, 1989.

[3] Ruben Prieto Diaz, 'A Software Classification Scheme,' Ph.D. thesis, University of Irvine, 1985.

[4] Brian Falkenhainer, et. al., 'The Structure Mapping Engine: Algorithm and Examples,' Artificial Intelligence, 41, 1-63, 1989/90.

[5] Martin Fowler, 'Analysis Patterns, Reusable Object Models,' Addison Wesley, 1997.

[6] Erich Gamma, et. al., 'Design Patterns: Elements of Reusable Object-Oriented Software,' Addison-Wesley, 1995.

[7] Gedre Gentner, 'The mechanisms of analogical learning,' In Bruce G. Buchanan and David C. Wilkins, editors, Readings in Knowledge Acquisition and Learning, Morgan Kaufmann Publishers, 673-694, 1993.

[8] Seymour Lipschutz, 'Discrete Mathematics,' McGraw-Hill

[9] Neil Arthur McDougall Maiden, 'Analogical Specification Reuse During Requirements Analysis,' Ph.D. thesis, City University, London, July 1992.

[10] Bertrand Meyer, 'Reusability: The case for object-oriented design,' In Ted J. Biggerstaff and Alan J. Perlis, editors, Software Reusability (II), 1-34, ACM Press, 1989.

[11] Tim O'Shea, 'The learnability of object oriented programming systems,' OOPSLA'86 proceedings, 502-504, Sep. 1986.

[12] Bruce W. Porter, et. al., 'Concept Learning and Heuristic Classification in Weak Theory Domains,' Artificial Intelligence, 45, 229-263, 1990.

[13] James Rumbauch, et. al., 'Object Oriented Modeling and Design,' Prentice Hall, 1991.

[14] G. Salton and M. J. Mcgill, 'Introduction to Modern Information Retrieval,' McGraw-Hill, 1983.

[15] R. Alan Whitehurst, 'Systemic Software Reuse Through Analogical Reasoning,' Ph.D. thesis, University of Illinois at Urbana-Champaign, 1995.

[16] Frank Feiks, David Hemer, 'Specification Matching of Object-Oriented Components,' sefm, p.182, First International Conference on Software Engineering and Formal Methods (SEFM'03), 2003.

[17] David Hemer, 'Specification matching of state-based modular components,' apsec, p.446, 10th Asia-Pacific Software Engineering Conference (APSEC'03), 2003.

[18] David Hemer, Peter Lindsay, 'Specification-Based Retrieval Strategies for Module Reuse,' aswec, p.0235, 13th Australian Software Engineering Conference (ASWEC'01), 2001.

배 제 민

e-mail : gemini@kd.ac.kr

1991년 중앙대학교 전자계산학과(공학사)

1993년 중앙대학교 대학원 컴퓨터공학과 (공학석사)

1998년 중앙대학교 대학원 컴퓨터공학과 (공학박사)

2003년~2007년 8월 31일 관동대학교 교육과학연구소 소장

1999년~현 재 관동대학교 컴퓨터교육과 교수

관심분야 : 소프트웨어프로세스개선, 컴퓨터교육