설계 패턴 기반 컴포넌트 분류와 E-SARM을 이용한 검색

김 귀 정[†] · 한 정 수[††] · 송 영 재[†††]

요 약

본 연구에서는 성공적인 컴포넌트의 재사용을 위하여 도메인 지향(domain orientation) 개념을 도입하여 컴포넌트들을 저장소에 분류, 검색하는 방법을 제안한다. 설계 시 디자인 패턴이 적용된 기존 시스템의 컴포넌트를 대상으로, 해당 도메인 내에 있는 각 컴포넌트와 기준패턴과의 구조적 유사성을 비교함으로서 컴포넌트를 분류하는 방법을 제시하였다. 재사용 가능한 컴포넌트를 기능별로 분할하고 그 구조를 다이어그램으로 제공함으로서 컴포넌트의 재사용 및 플랫폼간의 이식성을 높일 수 있다. 또한 E-SARM 알고리즘을 이용하여 질의와 가장 적합한 컴포넌트와 그와 유사한 후보 컴포넌트들이 우선순위(priority order)로 제공됨으로서 컴포넌트 재사용 효율을 높여줄 수 있도록 하였다.

Design Pattern Based Component Classification and Retrieval using E-SARM

GuiJung Kim[†] · JungSoo Han[††] · YoungJae Song[†††]

ABSTRACT

This paper proposes a method to classify and retrieve components in repository using the idea of domain orientation for the successful reuse of components.A design pattern was applied to existing systems and a component classification method is suggested here to compare the structural similarity between each component in relevant domain and criterion patterns. Classifying reusable components by their functionality and then depicting their structures with a diagram can increase component reusability and portability between platforms. Efficiency of component reuse can be raised because the most appropriate component to query and similar candidate components are provided in priority by use of-SARM algorithm.

키워드 : 컴포넌트 재사용(Component Reuse), 설계 패턴(Design Pattern), Clustering, (Spreading Activation), 컴포넌트 검색(Component Retrieval), 컴포넌트 분류(Component Classification)

1. 서 론

Various methodologies of component based development as techniques to cut down cost of production and maintenance of software are being suggested and studies on compatible component development are being done at many research institutes. Several components' development and success should be based on a domain-oriented component, and integration and reuse of a component should be done on a stabilized domain. For infrastructure in an integrated environment, exact component retrieval is necessary for customizing [1]. This paper aims at developing CBSE (Component-Based Software Engineering) ; a component retrieval and reuse system that makes it possible to assemble components and reuse them from other application for quick system construction and cost reduction.

There already exist component retrieval studies such as retrieval by signature matching [2], retrieval by sampling behavior [3], retrieval by specification matching [4] and so on. An architecture based component retrieval study [5] has enhanced recall rate, but is imperfect because of its precision and retrieval time.

In this paper, we classified components by constructing them in repository using a clustering algorithm after classifying them to domain introducing the idea of domain orientation for successful reuse of components. The clustering algorithm compares the structure with each component in an appropriate domain setting up design pattern as criterion object. Components developed by applying patterns from existing systems are increasing because the design time and communication between developers can be decreased if design pattern is applied at the component design stage [6, 7]. Therefore, this paper proposes a method to classify

† 정 회 원 : 건양대학교 컴퓨터학과 교수
†† 정 회 원 : 천안대학교 정보통신학부 교수
††† 종신회원 : 경희대학교 전자계산공학과 교수
논문접수 : 2004년 5월 6일, 심사완료 : 2004년 7월 20일

components by comparing their structural similarity in existing systems, to which a design pattern was applied, with a criterion pattern. It also suggests a component extraction technique for linkages between each classified component and criterion pattern, and enhances Spreading Activation Algorithm [8]. It raises the precision which is presented with problem from existing research and hence faster and more efficient component retrieval is made possible by shortening the long retrieval time - a disadvantage of the Spreading Activation Algorithm. So component reuse becomes simpler because users can choose components more easily and get specific usage information. Components retrieved in this way are reusable, considering the only logic of certain domain, for they can be reused regardless of platform. The remainder of this paper is as follows. In Section 2, we review related previous work. Section 3 showes component classification and retrieval method. Section 4 decribes system repository construction in detail. Performance evaluation is experimental results are described in Section 5. Finally, we give conclusions and some remarks for future work in Section 6.

2. Related Work

2.1 Design Pattern based Component

Design pattern involves a "reusable solution" to problems made at the time of system design. Applying a design pattern to the component design is beneficial as it reduces design time and communication between developers. So components developed by application of patterns from existing systems are increasing. Distributing components in business is the applicable part of a design pattern created by use of techniques to realize commercially useful components. This makes it possible to register and retrieve components that design pattern users require [6], encapsulates access to database and gives an interface to lower complex transactions [7]. Like this, pattern based component systems are gradually increasing and several researches on reusable component extraction are being processed, but a study on efficient retrieval methods related to pattern information is incomplete. In this paper, we classify and retrieve components by using the structural information of patterns in the component system. Methods of definition, classification and expression have been studied by Gamma [9], Buschmann [10], Tich [11] et al. Classification by Gamma is the method used mostly for the purpose of pattern application, so we classified components on the basis of Gamma's design pattern structure in this paper.

2.2 Domain Architecture Oriented Development

For activation of CBSD (Component-Based Software Development), it's important to develop the components efficiently. Both bottom-up and top-down ways should be harmonized to stop an unbalance between overproduction of components in the interest-high field and underproduction on the contrary. We need to use them after analyzing the domain and the scope of the existing system. Next, we construct the domain architecture on the basis of the analysis results [12]. Discrimination and classification of existing components are necessary for developing new components without applying similar components. If there are suitable components by application domain, large-scale reuse at this level of architecture can be possible [13].

2.3 Existing Component Retrieval

Component retrieval studies that already exist include : retrieval by signature matching [2], retrieval by sampling behavior [3], retrieval by specification matching [4] etc. Retrieval by signature matching is the method used to retrieve components by use of their signature information such as parameter type of function or interface. There are two types ; one is the exact matching method which retrieves the component that perfectly matches it's function's type with it's query's type, and the other is the relaxed matching method which retrieves similar components.

Retrieval by sampling behavior is attained by inferring the result after executing the routine in repository that has an interfacecompatible with interface specification. This method focuses on retrieval automation.

Comparing component specification and checking the possibility of replacement with another component achieve retrieval by specification matching. According to criteria to evaluate specification, there are two methods ; the pre/post matching method which compares the previous and post condition of each component and the predicate matching method which retrieves a component using relation of predicate inclusion.

Architecture based component retrieval [5] enhances the recall rate by making it possible to retrieve architecture corresponding with demands from developers in regards to the structural aspect of the component, but it's imperfect because of poor retrieval time and precision.

2.4 SARM

SARM (Spreading Activation Retrieval Method) [8] retrieves similar components including query function between components and queries, and is a method that can

find more accurate and wide ranging components. Calculating their activation values according to link information connected between each query and components retrieves components for reusability. This retrieval process specifies the base activation value 1.0 to query and the activation value of the positive number is calculated by accumulating its connection strength ; an activation value of 0 or a negative number is not passed. Each node accumulates its own activation value, which is inputted in the method. Inputted activation values areregulated by input data (fan-in), the step decay rate,and parameters. The computational procedure is circulated and the activation value becomes so stable that there is almost no change. Components can then be linked directly and similar components are retrieved.

The SARM system stabilizes the activation value, or executes the maximum cycle number that the user sets. However, as SARM measuressimilarity degree by repeating calculations using an activation value, its retrieval time is prolonged. More components increase, as the calculation process of the activation valueincreases geometrically, hence there are many retrieval difficulties.

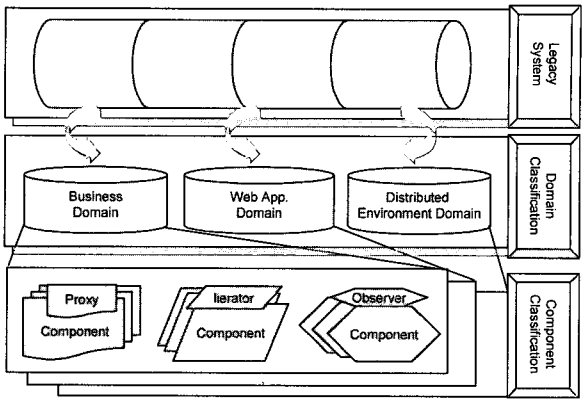
3. Component Classification and Retrieval

3.1 Component Classification

Component classification is processed in two steps. The first step aims at analyzing the domain of the existing system and classifying each component by application. The second step is the process of clustering by each design pattern applied to designing component-classified domain.

3.1.1 Domain Classification

The component software developmentprocess uses an assembly process to put together components and develop them. This study focuses on an assembly process for reuse of components from an existing system. In order to maximize reuse of components, it's important to separate suitable components to the quality of application domain, then extract components appropriate to demands and assemble them. For this work, it's necessary to classify the domain used in the existing system by application. We use the method to analyze commonality and variability, suggested in MARMIII [14]. We can find inter-reference relations between Use case model and class by use of class diagram, sequence diagram and Use case diagram acquired by reverse engineering from the existing system code. Domain information includingthe domain's quality can be extracted and managed. After that, components included in each domain are classified again by comparing structural agreements with design patterns.



(Figure 1) Component Classification

In this paper, we put a limit on components developed by applying design pattern. Developers who reuse components redesign classified domain and its components into a module that can be implemented in the most independent way. Components with design pattern attributes can diminish errors. Large-scale reuse at domain levels as well as components can be possible if there were components having various pattern functions in each domain. (Figure 1) shows the process of component classification.

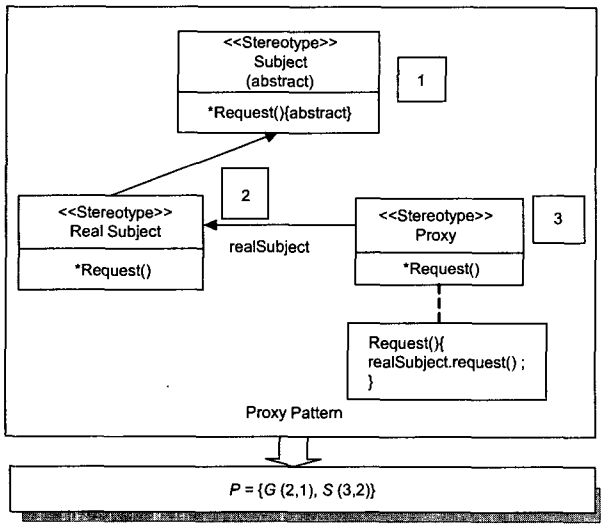
3.1.2 Design pattern Structure Based Clustering

Structure based clustering is the step of clustering by using the structure of components between classes. We set a structural and behavioral pattern of Gamma that is presently most used as the criterion pattern and exempt patterns that don't provide a pattern structure. The clustering algorithm is an algorithm, which joins similar components to a category when their structures agree with the criterion patterns. Structures of patterns and components are expressed with relation between classes in the class diagram of UML. To compare structure, one pattern and component are transformed into a group of order pairs.

$$P = R_k(i,j) \mid (i,j) \in R,\ i \in C,\ j \in C,\ 1 \le k \le n \quad (1)$$

P is the order pair of pattern and component, R is the relation of class, and C is class. Proxy pattern in (Figure 2) can be expressed with order pair, P = {G(2, 1), S(3, 2)}. In each order pair, an alphabet letter means relation of class diagram and numbers mean two classes. <Table 1> shows abbreviated words to express the relationship of the class diagram. The components in (Figure 3) can be transformed

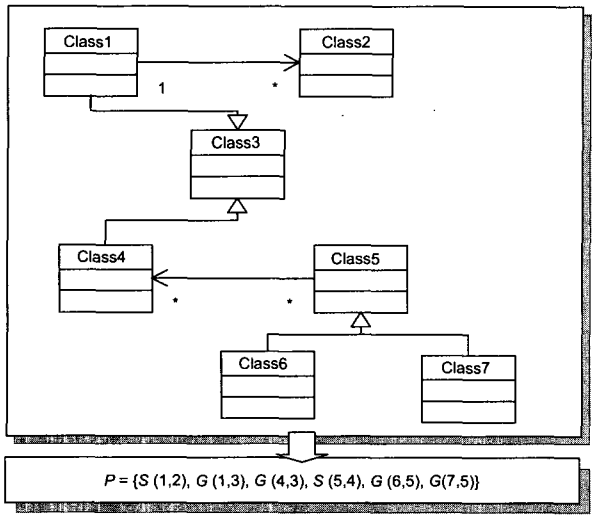
into order pairs - S(1, 2), G(1, 3), G(4, 3), S(5, 4), G(6, 5), G(7, 5). The diagrams show that the structure of proxy pattern is included in the component. In this component, the algorithm to locate proxy pattern is completed by comparing order pairs of two objects.



(Figure 2) Proxy Pattern's order pair

〈Table 1〉 Class Relationship

Relationship	Symbol
Association	S
Generalization	G
Aggregation	E
Composition	D
Dependency	C
Realization	R



(Figure 3) Component's order pair

In order to compare the foundation pattern expressed by the components order pair first, it's necessary to transform the class expressed by numbers. (Figure 4) shows the order pair transformation algorithm.

```
Initial Step P1 = { R1 = (i1, j1) | a→i1, b→j1 }
Step 2 ≤ k ≤ n
   P1 = { Rk(ik, jk) | if ik = i1, ik→a, ji→x
                       if jk = i1, ik→x, ji→a
                       if jk = j1, ik→x, ji→b
                       if ik = j1, ik→b, ji→x }
```

(Figure 4) Component/Pattern order pair transform algorithm

Transformation should be done to prevent bringing different results whenever the order of order pairs change (because it's possible that the class number is optional) and it compares regardless if the order changes. The transformation proceeding changes the first order pair into any letters. Here, the first class expressed in the order pair is changed into 'a' and the second order pair is changed into 'b'. In the remaining order pair, the same number with one of the first class in the first order pair is changed into 'a' and the same number with one of the second class is changed into 'b'. A class coupling with the class changed in the order pair is expressed with 'x' because there's no important meaning in comparison even though it is expressed with any number.

P = G (1,3), G (4,3), G (6,5), G (7,5), S (1,2), S (5,4)
⇒ P = G(a, b), G(x, b), G(6, 5), G(7, 5), S(a, x), S(5, 4)

Like above, class G (1,3) of the first order pair is transformed into (a, b), 1 into 'a' and 3 into 'b'. Also in other classes, 1 and 3 is changed into 'a' and 'b', and class coupling with order pair (a, b) is changed into 'x' because there's no relation in comparing with any other class.

```
// Transformation algorithms of Basic Pattern
Pf = {Pf1, Pf2, Pf3, ···, Pfk)}
// pattern order pair comparison
If (i = 0 ; i < n ; i++)
// Transformation algorithms of Component
 Pf = {Pa1, Pa2, Pa3, ···, pak)}
// if pattern structure is contained in component structure, save
and exit
 Pf ⊂ Pa
// if structure is Not the Same, change order pair
 Pa1 → Pan, Pak → Pa(k-1)
EndIf
```

(Figure 5) Structure comparison algorithm

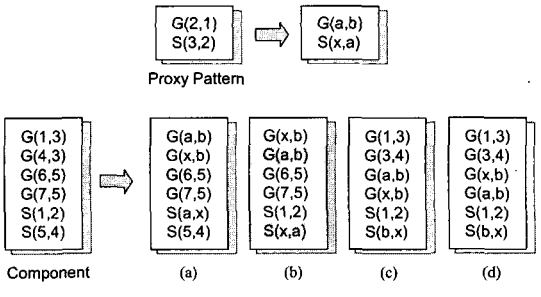
In the algorithm of (Figure 5), Pf is the criterion pattern and Pa istransformation. In order to compare components

using the criterion pattern, a user transforms the criterion pattern and the components with the transformation algorithm and then compares to see if these two objects are the same. To find a pattern related with a criterion pattern among some order pairs showing relation of component, these order pairs are transformed in order.

After this transformation proceeding, if order pairs satisfy conditional formula (2) below, component, Pa is clustered into the criterion pattern, Pf. If an order pair doesn't satisfy a conditional formula (2) even after transformation of all other order pairs, the component is compared to other.

$$P_f \subset P_a \tag{2}$$

(Figure 6) shows a concrete diagramming of comparison proceedings of the proxy pattern and components.The proxy pattern is composed of one association relation and one inheritance relation. Proxy pattern, G(1, 3) is transformed to G(a, b) to search for association relations and inheritance relations of the same pattern among components. In other order pairs, 1 is transformed into a, 3 into b. After transformation, the order pair of the proxy pattern becomes (a, b), S(x, b). A component is compared through transformation proceedings and has 4 association relations. Among these relations, to find the one with the proxy pattern's structure, one of them will be compared to the order pair transformed from the proxy pattern through transformation proceeding.If there are order pairs having structures of (a, b), S(x, b), a user pattern is clustered to the group of proxy pattern. Otherwise, the proxy pattern and the component can't be called the same structure pattern and comparing will be continued over again.

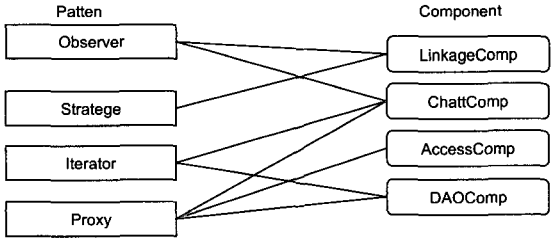


(Figure 6) Component comparison process

Components clustered by a criterion pattern give linkage with a relevant criterion pattern. They also give linkage to all related criterion patterns even when a component has more than one pattern structure. It even efficiently retrieves candidate components using linkage at the time of the component retrieval.

3.2 Component Retrieval using E-SARM

SARM (Spreading Activation Retrieval Method) is the method used to find more precise and inclusive component by searching for similar components with a query function between components and query. Reusable components are retrieved by calculating the activation value according to each link connected between queries and components. The purpose of repeating the process of calculating the activation value is to retrieve similar components with the exact activation value. This method is time-consuming because each query language and activation value of components are affected by other languages and activation values of other components. So, in this study, we solve this disadvantage of SARM and introduce it into component retrieval. We reduced retrieval time by decreasing the calculation times,which is possible by removing components that have the least number of linkage information. Link information means the linkage given between components clustered by the criterion pattern at the component classification and criterion patterns. Namely, E-SARM (Enhanced Spreading Activation Retrieval Method) [15] narrows down the extension scope of retrieval and retrieves much more related candidate components by removing the link information of the criterion pattern or of the component and then exempting it from operation, after the circulation process is repeated at fixed times.



(Figure 7) Class and Query Relationship

(Figure 7) gives an exampleof the relationship between criterion patterns and components. It shows that a component linked to more than two criterion patterns has structures and qualities of more than two criterion patterns inside the component structure. If you input a criterion pattern 「Observer」 as a query, three components are retrieved. The criterion pattern, 「Observer」 is linked directly to components, 「LinkageComp」 and 「ChattComp」, but not linked directly to the components, 「AccessComp」 and 「DAOComp」. However, it is linked to two components, 「AccessComp」 and 「DAOComp」 through 「Observer」(query : criterion pattern) → 「ChattComp」(component) → 「Iterator」(criterion

pattern) → 「DAOComp」(component), and 「Observer」(query : criterion pattern) → 「ChattComp」 (component) → 「Iterator」(criterion pattern) → 「AccessComp」(component). However, 「AccessComp」 is exempted in this linkage because it is less referred to in retrieval processes. Each criterion pattern and component calculates an activation value referring to it's link node. The reason why 「LinkageComp」 is a more frequent result than 「ChattComp」 is because 「LinkageComp」 is referred to more than 「ChattComp」. The simulation results of average reference times for existing SARM and E-SARM show that the number of activation value calculations dropped by 37.8%. This algorithm stores component and base patterns in an arrangement of columns and rows, and calculates and delivers activation values, which connect with components.

```
while
 Get_Level = Get_Pos[0] / End1 ; /* position of row matrix */
 Get_Col = Get_Pos[0] % End1 ; /* position of column matrix */
 if(exist component)
   for(all component number)
       Array[ ] =each component value(0 or 1)
 else
   for(all query number)
       Array[ ]= each query value(0 or 1)
 for(query or component number)
   if(exist relationship and index is Not last_index)
       push Current_index in Stack
 query_visit_count ++ component_visit_count ++
 if(first query)
   initialize query_act_value=1.0
 else if(a query)
      Di(t+1)=
        δ_Di(t) + φ_Di(t)(M − D_i(t)) if φ_Di(t) > 0
        δ_Di(t) + φ_Di(t)(D_i(t) − m) if φ_Di(t) ≤ 0
        δ_Di = (1 − θ_D) D_i(t)
    else if(a component)
        Ti(t+1)=
          δ_Ti(t) + φ_Ti(t)(M − T_i(t)) if φ_Ti(t) > 0
          δ_Ti(t) + φ_Ti(t)(T_i(t) − m) if φ_Ti(t) ≤ 0
          δ_Ti = (1 − θ_T) D_i(t)
          else Err("Not Calculate Activation Value")
    pop(Get_Pos) ;
    if( Not MAX_CYCLE )
    { cycle++ ;
     if(cycle > MAX_CYCLE) break ;  /* MAX_CYCLE = 3~5 */
     if(cycle is between 2 & 3)
       for(all query and components)
         if(query_visit_count or component_visit_count
           = current_cycle_count)
          { AvgVisit += VisitNum[ ] ; cnt++ ; }
         if(cnt==0) return 0 ;
         else AvgVisit /= cnt ;
      }
     if(visit_count exist)
       for( all query and component)
         if(VisitNum <= AvgVisit){
           for(j = 0 ; j < End1 ; j++) level0_1[j][i%End1] = 0 ;
         Cut_component_value = -999.0 ; }
endwhile
```

(Figure 8) E-SARM algorithm

This algorithm increases the number of references and initializes the activation valueby 1. Whenever a reference happens, the activation value for each query and component is calculated. After comparison with the average of the number of references, this algorithm removes the lowest node(s) based to a standard. Removing node about the number of reference by such a method and the number of activation value calculations decrease, hence the speed improves at the same time. The retrieval result is kept existent with SARM's maximum result. The exclusion standard is improved so that errors of existent SARM and 1.23 retrieved components may be able to exist but similar components are retrieved faster.

<Table 2> E-SARM Parameter

Symbol	Definition
$\delta_{D_i}(t)$	Pre-activation value decreased by criterion pattern reduction ratio
$\phi_{D_i}(t)$	Input value coming into present component i
M	Maximum activation value = 1.0
$D_i(t)$	Cumulative component activation value of previous step
θ_D	Component reduction ratio = 0.1
$\delta_{T_i}(t)$	Pre-activation value decreased by component reduction ratio
$\phi_{T_i}(t)$	Input value coming into present criterion pattern I
m	Minimum activation value = -0.2
$T_i(t)$	Cumulative criterion pattern activation value of previous step
θ_T	Criterion pattern reduction ratio = -0.2

<Table 2> gives us a definition of the variables for the activation value calculation used in the E-SARM algorithm (Figure 8), where the criterion pattern and components activation value is calculated using metrics. The more times circulation is repeated, the more stable the activation value becomes. The part where reference times don't conform a standard is removed automatically and the calculation process ends.

4. System Design

4.1 system repository

The system repository in this paper is composed of an information database where general information on the criterion pattern and components are saved, and the structure database where structure information of the UML class diagram is also saved. strCategory gives information on the domain classification. docDocument is the field which explains the whole-component it's principle, intention, applicable circumstances and what is necessary to recognize the component realization ; pitfalls, hints and techniques. relCom is a criterion pattern related to components classified by clustering. UserID stand for the identity of the user who

creates and registers a component. <Table 3> shows the layout of the component information database. <Table 4> depicts the component structure database. Information of component structure includes image expressing the structure of the components class diagram and information of the component's order pair to conduct clustering for component classification. As shown in <Table 5>, pattern information uses a form that is used on Gamma's design pattern expression, and pattern classification by Agerbo's estimation and user ID that are registered are stored in addition to the pattern information database that stores the information.

〈Table 3〉 component information databas

Field Name	Data Format	Description
strName	char	Name of component
strCategory	char	Category by domain classification
docDocument	text	Description about component
relCom	Link	Criterion pattern related to component
UserID	char	Component registrant's ID

〈Table 4〉 component structure database

Field Name	Data Format	Description
strName	char	Name of component
imgStructure	image	Image of component structure
StrPair	char	Information of order pair
strPattern	char	Relevant pattern

〈Table 5〉 Pattern information database

Field Name	Data Format	Description
strCategory	char	Classification by use purpose
strAnalysis	char	Classification by verification (FDP, LDDP, RDP)
StrName	char	Design Pattern Name
docIntent	text	Principle and intention
docMotiv	text	Motivation that helps abstract understanding of pattern
docApplic	text	pattern application
imgStruct	image	Image of pattern structure
docPartic	text	Class and object of design pattern
docConseq	text	Result of pattern use
docImple	text	Pitfall, hint, technology at pattern implementation
docCode	text	Pattern code by C++, Java
docRelated	text	Difference of Related pattern
Mem_ID	char	User ID

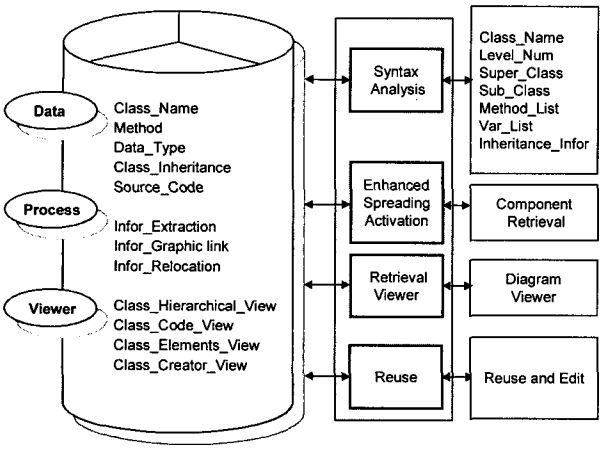
A pattern that is classified according to use purpose by the strCategory value is used as a facet in facet retrieval with the strAnalysis field value that is classified by FDP, LDDP, and RDP. Also, an user who registers a pattern can correct and delete this pattern to register by oneself because they can confirm the user ID through the Mem-ID field. The DocPartic field amounts input value that can describe the class name such as the class or the object that takes part in the structure of the design pattern, and change it's specific pattern structure to a pattern structure that consists of abstract class names when creating a code.

4.2 System Structure

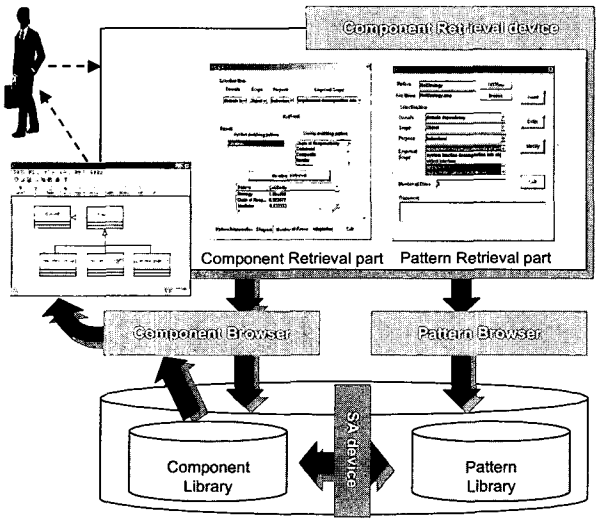
As shown in (Figure 9), this system stores information that includes the class name, level, super-class, and the member function of the component information in repository through a syntactic analyzer. The repository consists of three areas : Data, Process, and Viewer area and if the user inputs a query through the retrieval viewer, after the activation value is calculated by E-SARM, the components appear by their priority of their activation values. This time, if the user copied and pasted the selected component in the diagram to reuse this component, the component information, the source code information, and the graphic information move automatically. Also this system corrects components in order to ensure compatibility with the support editor. And this made editor supports the functions of the member function, the variable declaration etc. If any class is selected in the class diagram, the editor is supported with its source code, and if this is stored after modifying the code, the code information is updated in the component repository and can then be reused. The whole system structure in this study is composed of four tools - a component retrieval section, a component browser, a pattern browser and a S.A device. We constructed a pattern library and a component library separately to save an abstract pattern structure and concrete component structure.

We used the structural pattern and the behavioral pattern of Gamma as a criterion pattern and omitted patterns that didn't provide any pattern structures. Developers can retrieve a component that they wish to reuse by use of a component retrieval device. If one selects qualities such as domain, function, type and OS of the component, one wants to retrieve components, using an interface that provides the component information, and then a component browser retrieves this component satisfying these qualities in a component library. A S.A device calculates the activation value of the retrieved component with a criterion pattern and then extracts similar candidate components. Besides this, it also conducts a query by a pattern as well as by a component and supports to retrieve components doing a function or

a role of a criterion pattern even if a component cannot be properly set up. At this time, the pattern browser retrieves the criterion pattern and the S.A device retrieves the components. The system suggested in this study can raise the efficiency of component reuse by providing the most appropriate component to query and similar candidate components in order of their activation values.



(Figure 9) Component Reuse System



(Figure 10) System structure

5. Performance Evaluation

In <Table 6>, we compare the system we designed to existing component related systems. SCF (Software Component Factory)[16] of Select corp., Together [17] of TogetherSoft corp. and COOL : Joe2.0 [18] of Computer Associates corp. are representative tools for component development. We compared our proposed method in this paper with these three systems to see how efficiently our system reuses components during the component creation process.

<Table 6> Reusability Comparison

	SCF	Together	COOL : Joe2.0	Suggested system
component discrimination	×	×	△	○
pattern application	×	×	×	○
Domain reuse	×	×	×	○
Component Modeling	×	×	Platform subordinate	Platform independent
Domain Modeling	- BP modeling - UC modeling	- UML diagram modeling	- UC modeling Type diagram	- UC modeling - Object modeling - Sequence diagram

"×" means that the system doesn't support a certain function at all, "△" means an imperfect support, and "○"means a perfect function. SCF and Together support the function of the component interior design and realization, but don't support the component modeling function, which discriminates some components and expresses them diagrammatically. COOL : Joe2.0 can discriminate the EJB component on the system architecture and support component modeling function. However, this system causes a reusing problem in circumstance of other platforms like CCM or COM because it creates a platform-dependent component. In order to increase reusability between platforms, the suggested system hides the specific code for realization by retrieving the component in consideration of the only pattern structure applied to the component, and gives us the component structure with a diagram. Component reusability and portability between platforms can be increased by classifying reusable components by function and showing their structures with a diagram. In addition, it supports the retrieval of candidate components and classifies components by domain ; so large-scale reuse at domain levels can become possible.

Also, through classification by criterion pattern, components retrieval consists easily using E-SARM if the extended components have connection information for a basic pattern because components are classified in a form that they extend in this basic pattern. Via characteristics of the retrieval model, the classification method, the similar pattern retrieval, modeling tool etc., <Table 7> shows comparisons between existing systems and the system that has been designed in this paper. We compared our system with ModelMaker [19], Omnibuilder [20], Plastic [21] and MetaEdit+[22] which are often used as a CASE tool. Plastic or MetaEdit+ is available by UML modeling but manage-

ment and retrieval of components do not consist efficiently because an independent retrieval model or classification model does not exist. Also, pattern based systems, such as Omnibuilder and ModelMaker, are using Gamma's pattern classification as a standard but there are difficulties to retrieve and manage patterns that are added continuously because they use a string matching method for retrieval, which is only available ifthe pattern name is known.

Thus, in this paper, our proposed method is that UML modeling which is already available, and automatic classification by structure based clustering which is also available, are to run in parallel with a similarity component retrieval method using E-SARM

〈Table 7〉 Comparing of existing

	Modelmaker [19]	Omnibuilder [20]	Plastic 3.0 [21]	Metaedit+ [22]	Suggested system
Component	Pattern	Pattern	Class	Class	Pattern
Retrieval Model	String matching	String matching	String matching	String matching	E-SARM
Modeling Tool	Template	Template	Component	Component	Pattern
Addition	O	O	O	O	O
Similar Retrieval	O	O	O	O	O
Classification Method	Gamma	Object	Enumeration	Enumeration	Clustering

〈Table 8〉 Precision and Recall

Recall	SARM (Precision)	E-SARM (Precision)
0.1	0.72	0.98
0.2	0.70	0.94
0.3	0.69	0.90
0.4	0.68	0.86
0.5	0.66	0.83
0.6	0.63	0.76
0.7	0.61	0.72
0.8	0.57	0.66
0.9	0.53	0.60
1.0	0.48	0.53
Avg. of Precision	0.627	0.778
Enhanced Avg.	-	24.08%

In this paper, we measured the recall and the precision to evaluate the efficiency of the retrieval system which uses the E-SARM. The method is the recall and precision in case that E-SARM and existing SARM is applied. 100 queries is selected with option, and average of the precision is compared after measuring the precision change of two cases, changing the recall with 0.1 units. <Table 8> is to show the precision and recall of two cases. The result of this simulation shows that the E-SARM is higher 24.08% (((0.778-0.627)/0.627)*100) degree than existing SARM.

6. Conclusion

We need an interface that makes it easy to retrieve a component among other diverse components, assemble this component and use it, hence satisfying users' demands. When we develop software as a unit of a component, component classification is required for efficient management of many components. In this paper, we try to manage components efficiently and provide a more exact component retrieval method using similar candidate components satisfying demands of users. We classified components of existing systems into an application domain, did clustering by use of the criterion pattern's structure in each domain, and constructed a component repository. The structure of criterion patterns and components change between classes into order pairs and checks to ensure there is a criterion pattern's structure in each component.

A clustered component has a linkage with a criterion pattern. Using the E-SARM algorithm based on relevant information has enhanced the precision of component retrieval. Retrieved components appear as several candidates in priority and their structures are shown diagrammatically. Therefore, developers can do a platform-independent design because they can reuse components without regard to a specific code or realization in designing a new system.

From now on, the study of specific modeling that is applied largely to diverse domains and components and the study of an efficient method to assemble components are required. A study on a classification method in case of more than two pattern structures being applied on a component is currently being conducted.

References

[1] George T. Heineman, William T. Council, "Component Based Software Engineering," addision-wesley, pp.143-160, 2001.

[2] A. M. Zaremski, J. M. Wing, "Signature Matching : A Tool for Using Software Libraries," ACM Transaction Software Engineering and Methodology, Vol.4, No.2, 1995.

[3] A. Podgurski, L. Pierce, "Retrieving Reusable Software by Sampling Behavior," ACM Transaction Software Engin-

eering and Methodology, Vol.2, No.3, 1993.
[4] A. M. Zaremski, J. M. Wing, "Specification Matching of Software Components," In Proceedings of the third ACM SIGSOFT symposium on the foundations of software engineering, 1995.
[5] Seung-Geun Lee, Chi-Don Ahn, "Reusable Component Retrieval Based on Software Architecture," The Transactions of Korea Information Science Society, Vol.27, No.11, pp.1099-1105, Nov., 2000.
[6] Haeng-Kon Kim, Ha-Jung Choi, Eun-Ju Han, "The e-Business Component Construction based on Distributed Component Specification," The Transactions of Korea InformationProcessing Society, Vol.8, No.6, pp.705-714, Dec., 2001.
[7] Seong-Man Choi, Jeong-Yeal Lee, "Design and Implementation of IDAO for Efficient Access of Database in EJB Based Application," The Transactions of Korea InformationProcessing Society, Vol.8, No.6, pp.637-644, Dec., 2001.
[8] Scott Henninger, "Information Access Tools for Software Reuse," System Software, pp.231-247, 1995.
[9] E. Gamma, R. Helm, R. Johnson and J. Vlissides, "Design Pattern : Elements of Reusable Object-Oriented Software," Addison-Wesley, 1995.
[10] F. Buschman, R. Meunier, H. Rohnert, P. Sommerland and Stal Michael, "Pattern-Oriented Software Architecture-A of Pattern," John Wiley & Sons, 1996.
[11] W. Tichy, Essential Software Design Pattern, University of Karsruhe, 1997.
[12] P. Bengtsson and J. Bosch, "Scenario-based Software Architecture Reengineering," in Proceeding 20th ICSE, IEEE, Jun., 1998.
[13] Shim U. K, Back I. Sup. Lee J. T. Ryu K. Y., "The Value-Added Brokerage for Steering the CBSD Environments," The Transactions of Korea Information Processing Society, Vol.8, No.6, pp.681-690, Dec., 2001.
[14] Joe H. J., Ha J. S., Kim J. S., Park C. S., "Component based System development Methodology Marmi-III," Project Management Technology, Vol.4, pp.1-13, 2001.
[15] Jung-Soo Han, Young-Jae Song, "Orient-Oriented Components Reuse System using Enhanced SARM," The Transactions of Korea Information Processing Society, Vol.7, No.4, pp.1092-1102, Apr., 2000.
[16] http://www.anonix.com/.
[17] http://www.together.com/.
[18] http://www.cai.com/.
[19] www.modelmaker.demon.nl/.
[20] www.omnibuilder.com/.
[21] www.pasticsoftware.com/.
[22] www.metacase.com.

김 귀 정

e-mail : scallet@case.kyunghee.ac.kr
1994년 한남대학교 전자계산공학과(공학사)
1996년 한남대학교 전자계산공학과 (공학석사)
2003년 경희대학교 전자계산공학과 (공학박사)
2001년~현재 건양대학교 컴퓨터학과 조교수
관심분야 : 소프트웨어공학, S/W 재사용, CBD

한 정 수

e-mail : jshan@cheonan.ac.kr
1990년 경희대학교 전자계산공학과(공학사)
1992년 경희대학교 전자계산공학과 (공학석사)
2000년 경희대학교 전자계산공학과 (공학박사)
2001년~현재 천안대학교 정보통신학부 조교수
관심분야 : CBSE, 컴포넌트 형상관리, EJB, S/W 재공학

송 영 재

e-mail : yjsong@khu.ac.kr
1969년 인하대학교 전기공학과(공학사)
1976년 일본 Keio University 전산학과 (공학석사)
1979년 명지대학교 대학원졸업(공학박사)
1971년~1973년 일본 Toyo Seiko 연구원
1982년~1983년 미국 Univ. of Maryland 전산학과 연구교수
1985년~1989년 IEEE Computer Society 한국지회부 회장
1984년~1989년 경희대학교 전자계산소장
1993년~1995년 경희대학교 교무처장
1996년~1998년 경희대학교 공과대학장
1998년~2000년 경희대학교 기획조정실장
1976년~현재 경희대학교 전자계산공학과 교수
관심분야 : 소프트웨어공학, OOP/S, CASE 도구, S/W 개발도구론, S/W 재사용, CBD