클러스터링 알고리즘을 이용한 컴포넌트 분류 및 검색

Component Classification and Retrieval using Clustering Algorithm

김귀정
건양대학교 IT 학부 교수

Gui-Jung Kim
Professor, Division of Information Technology,
KonYang University.

중심어 : 컴포넌트 재사용, 도메인, 디자인패턴, 클러스터링, Spreading Activation, 컴포넌트검색, 컴포넌트 분류

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 유사함을 비교함으로서 컴포넌트를 분류하는 방법을 제안한다. 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 후보컴포넌트들이 우선순위로 제공됨으로서 컴포넌트 재사용 효율을 높여줄 수 있도록 하였다.

Abstract

This study proposes method to classify components in repository and retrieve them introducing the idea of domain orientation for successful reuse of components. About components of existing systems design pattern was applied to, we suggest component classification method to compare structural similarity between each component in relevant domain and criterion pattern. Component reusability and portability between platforms can be increased through classifying reusable components by function and giving their structures with diagram. Efficiency of component reuse can be raised because the most appropriate component to query and similar candidate components and provided in priority by use of E-SARM algorithm.

I. 서론

소프트웨어 생산성과 유지보수 비용을 줄일 수 있는 기법으로 다양한 컴포넌트 기반의 개발 방법론이 제안되고 있으며, 현재 한국컴포넌트컨소시엄(KCSC)등에서 상호호환을 위한 컴포넌트 개발에 관한 연구가 진행되고있다. 이와 같은 많은 컴포넌트가 개발되었을 경우, 사용자가 원하는 컴포넌트 식별방법은 분석자의 경험에 의존하는 경우가 많다. 다양한 컴포넌트 개발과 성공은 도메인 지향 컴포넌트에 기반을 두어야하며 컴포넌트를 통합, 안정된 도메인 상에서의 컴포넌트 재사용이 이루어져야 한다. 이를 위하여 통합환경에서의 인프라를 제공해야 하며 커스터마이징(Customizing)을 위한 정확한 컴포넌트의 검색 방법이 필요하다[1]. 본 연구에서는 CBSE(Component-Based Software Engineering) 즉, 신속한 시스템 구축, 비용절감을 위하여 다른 application으로부터 컴포넌트를 조립하여 재사용이 가능하도록 하는 컴포넌트 검색 재사용 시스템 개발에 목적을 둔다.

기존의 컴포넌트 검색 연구는 시그니쳐 일치 검색[2], 행위 샘플링에 의한 검색[3], 명세서 일치에 의한 검색[4] 등이 있다. 또한 아키텍쳐 기반 컴포넌트 검색 연구[5]는 재현율을 향상시켰으나 검색시간과 정확성이 떨어지는 단점이 있다.

본 연구에서는 성공적인 컴포넌트의 재사용을 위하여 도메인 지향(domain orientation) 개념을 도입하여 컴포넌트들을 저장소에 분류, 구축하기 위하여 먼저 도메인으로 분류하고 그 후 클러스터링 알고리즘을 이용하여 분류 저장한다. 클러스터링 알고리즘은 디자인 패턴을 기준객체로 설정하여, 해당 도메인 내에 있는 각 컴포넌트와 그 구조를 비교한다. 디자인 패턴을 컴포넌트 설계 시 적용하면 설계 시

간의 단축 및 개발자간 의사소통을 줄일 수 있기 때문에, 기존 시스템에서 패턴을 적용하여 개발된 컴포넌트가 점차로 늘어나고 있는 실정이다[6][7]. 이에 본 연구에서는 설계시 디자인 패턴이 적용된 기존 시스템의 컴포넌트를 대상으로, 기준 패턴과의 구조적 유사함을 비교함으로서 컴포넌트를 분류하는 방법을 제안한다. 또한 분류된 각 컴포넌트와 기준패턴사이에 연관성을 부여하고, Spreading Activation 알고리즘[8]을 개선한 컴포넌트 추출기법을 제안한다. Spreading Activation 알고리즘을 이용하여 검색된 컴포넌트들은 우선 순위가 높은 순서로 검색되기 때문에 상위의 검색 결과물에 가장 근접한 컴포넌트 검색결과가 되는 것이다. 또한 기존 연구에서의 문제점으로 제시되었던 정확성을 높임으로써 그 효율성을 증명하고 Spreading Activation 알고리즘의 단점인 검색시간이 많은 단점을 개선시켜 보다 빠르고 효율적인 컴포넌트 검색이 가능하도록 한다. 따라서 사용자가 원하는 컴포넌트를 쉽게 선택할 수 있도록 하고 또한 검색된 컴포넌트에 대한 사용 명세 정보가 제공되기 때문에 컴포넌트의 재사용이 용이하게 된다. 이렇게 검색된 컴포넌트는 특정 도메인의 로직만을 고려하여 구성된 재가용 단위이기 때문에 플랫폼과 무관하게 재사용될 수 있다는 장점이 있다.

II. 관련 연구

1. 디자인 패턴의 컴포넌트화

디자인 패턴은 시스템 설계시 자주 발생하는 문제들에 대한 "재사용 가능한 해결책"이라 할 수 있다[9]. 디자인 패턴을 컴포넌트 설계 시 적용하면 설계 시간을 단축할 수 있고 개발자간 의사소통을 줄일 수 있다는 장점이 있다. 이에 기존 시스템에서 패턴을 적용하여 개발된 컴포넌트가 점차로 늘어나고 있는 실정이다. 특히 비즈니스 환경의 분산 컴포넌트는 상업적으로 유용한 컴포넌트 구현 기술을 사용하여 생성되는 디자인 패턴의 응용분야로서, 사용자가 원하는 디자인 패턴 중심의 컴포넌트를 등록, 검색할 수 있도록 해주며[6], 데이터베이스의 접근을 캡슐화하고 트랜잭션의 조작의 복잡성을 줄이기 위한 인터페이스로서도 컴포넌트화되어 진다[7]. 이처럼 패턴 기반의 컴포넌트 시스템이 점차로 확대되어가고 있고, 재사용을 위한 컴포넌트 추출에 관련한 여러 연구가 진행되고 있지만, 패턴정보와 연관된 효율적인 검색방법에 대해서는 연구가 미비한 실정이다. 이에 본 연구에서는 디자인 패턴이 적용된 컴포넌트 시스템에서 패턴의 구조적 정보를 이용하여 컴포넌트를 분류, 검색하고자 한다.

패턴에 대한 정의와 분류 그리고 표현 방법에 대해서 Gamma[9], Buschmann[10], Tich[11] 등 다양한 연구가 진행되어 왔다. 이 중 Gamma에 의한 분류 방법이 패턴 활용을 목적으로 가장 많이 활용되고 있는 방법이다. 이에 본 연구에서도 Gamma의 디자인 패턴 구조를 기반으로 하여 컴포넌트를 분류하고자 한다.

2. 도메인 아키텍쳐 지향의 개발

CBSD(Component-Based Software Development)의 활성화를 위해서는 컴포넌트의 효율적인 개발이 중요하다. 이윤이 높은 영역에서는 컴포넌트가 과다하게 생산되고 그렇지 못한 부분에서는 컴포넌트가 부족한 영역별 불균형 현상이 발생하지 않도록 컴포넌트 개발의 상향식, 하향식 방법이 접목되어져야 할 것이다. 이를 위해서 이미 사용하고 있는 기존 시스템의 도메인과 영역을 분석하여 활용해야 한다. 그 다음 분석 결과를 바탕으로 도메인 아키텍쳐를 구성한다[12]. 아키텍쳐에 기술된 컴포넌트를 찾는 과정에서 실제 유사한 컴포넌트가 있더라도 적용하지 못하고 새롭게 개발하는 경우가 있는데, 이를 위해 현재 존재하는 컴포넌트를 식별, 분류하는 작업이 필요하다. 응용 도메인별 적용 가능한 컴포넌트를 제공받는다면, 컴포넌트의 재사용뿐아니라, 아키텍쳐 수준의 대규모 재사용이 가능해 질 수도 있다[13].

3. 기존의 컴포넌트 검색

기존의 컴포넌트 검색 연구는 시그니쳐 일치 검색[2], 행위 샘플링에 의한 검색[3], 명세서 일치에 의한 검색[4] 등이 있다. 시그니쳐 일치 검색은 함수의 파라메터 타입이나 인터페이스와 같은 시그니쳐 정보를 이용하여 컴포넌트를 검색하는 방법이다. 이 방법에는 함수의 타입과 질의의 타입이 완전히 일치하는 컴포넌트를 검색하는 완전 일치 방법과 비슷한 컴포넌트를 검색하는 이완 일치 방법이 있다.

행위 샘플링 검색은 인터페이스 명세서와 호환되는 인터페이스를 가진 저장소의 루틴을 실행하여 그 결과를 비교한다. 이 방법은 검색의 자동화에 중점을 두었다.

명세서 일치 방법은 컴포넌트 명세서를 비교하여 다른 컴포넌트와 대체될 수 있는지를 비교하여 결정한다. 명세서를 평가하는 기준에 따라 각 컴포넌트의 선, 후조건을 비교하는 선조건/후조건일치 방법과 술어 포함 관계를 이용하여 검포넌트를 검색하는 술어 일치방법이 있다.

또한 아키텍쳐 기반 컴포넌트 검색 연구[5]는 컴포넌트의 구조적 측면에서 개발자의 요구에 일치하는 아키텍쳐 검색이 가능하여 재현율을 향상시켰으나 검색시간과 정확성이 떨어지는 단점이 있다.

III. 도메인 기반 컴포넌트 분류와 검색

1. 컴포넌트 분류

컴포넌트 분류 과정은 크게 2단계로 나누어진다. 첫 번째 단계는 기존 시스템의 도메인을 분석하여 응용별 컴포넌트를 분류시키는데 목적이 있다. 두 번째 단계는 도메인 별로 분류된 컴포넌트를 설계시 적용되었던 디자인 패턴별로 클러스터링 하는 과정이다.

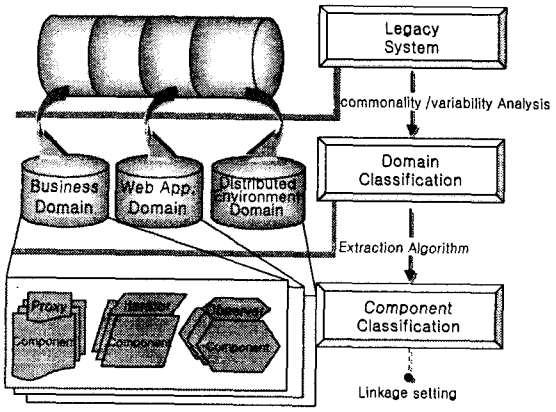

그림 1. 컴포넌트 분류 과정

1.1. 도메인 분류

컴포넌트 소프트웨어 개발 프로세스는 컴포넌트를 조립하여 어플리케이션을 개발하는 조립 프로세스와 컴포넌트를 개발하여 사용하는 생성 프로세스로 나뉜다. 본 논문에서는 기존 시스템의 컴포넌트 재사용을 바탕으로 한 조립 프로세스에 초점을 맞추도록 한다. 컴포넌트의 재사용을 극대화시키기 위해서는 응용 도메인의 특성에 맞는 컴포넌트를 분리해서 관리하고 요구사항에 맞는 적절한 컴포넌트를 추출하여 조립하는데 있다. 이를 위해 기존 시스템에서 사용하고 있는 도메인을 응용별로 분류해야 하는데, 마르미III [14]에서 제안한 공통성과 가변성 분석 방법을 이용한다. 기존 시스템 코드로부터 역공학을 이용하여 클래스 다이어그램, 순차도, 그리고 Use case 다이어그램을 얻을 수 있는데, 이를 이용하여 Use case 모델과 클래스간의 상호 참조 관계를 알 수 있다. 이로부터 도메인의 특성을 나타내는 도메인 영역 정보를 추출할 수 있다. 그 결과를 각 도메인특성 정보로 관리한다. 그 후 각 도메인에 포함된 컴포넌트에 대해 디자인 패턴과의 구조적 일치를 비교하여 재분류한다. 본 연구에서는 이용되는 컴포넌트는 설계시 디자인 패턴이 적용되어 개발된 컴포넌트에 제한을 둔다. 그렇게 함으로서 컴포넌트를 재사용하는 개발자는 분류된 도메인과 그에 속한 컴포넌트를 최대한 독립적으로 실행 가능한 모듈로 재설계 할 수 있을 것이다. 디자인 패턴의 속성을 가지고 있는 컴포넌트는 그렇지 않은 것보다 설계 시 오류를 최소한으로 줄일 수 있으며, 각 도메인 별로 다양한 패턴 기능을 포함한 컴포넌트들이 존재한다면 컴포넌트 단위의 재사용뿐만 아니라 도메인 수준의 대규모 재사용도 가능할 것이다. 그림 1은 컴포넌트 분류 과정을 보여준다.

1.2. 구조 기반 클러스터링

컴포넌트들이 가지고 있는 클래스들 간의 구조를 이용하여 클러스터링 하는 단계이다. 비교기준이 되는 패턴은 현재 가장 많이 사용되고 있는 Gamma의 구조, 행위 패턴을 기본으로 하였으며, 패턴 구조를 제공하지 않는 패턴은 제외시켰다. 클러스터링 알고리즘은 기준으로 정한 패턴 중에서 컴포넌트의 구조와 일치하는 구조가 있는 경우에 같은 그룹으로 클러스터링 되도록 하는 알고리즘이다. 패턴과 컴포넌트의 구조는 UML의 클래스 다이어그램에서 클래스와 클래스간의 관계로 나타낸다. 구조를 비교하기 위해 클래스와 클래스간의 관계를 하나의 순서쌍으로 하여 하나의 패턴과 컴포넌트를 순서쌍의 그룹으로 변환한다.

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

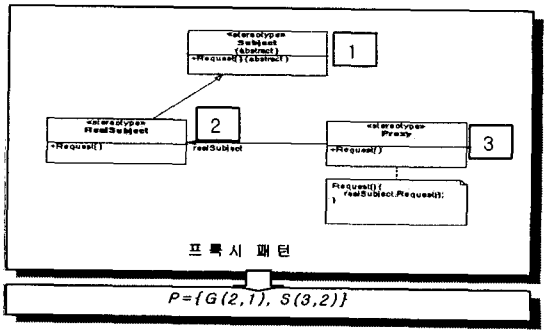

그림 2. 프록시 패턴의 순서쌍 표현

여기서 P는 패턴과 컴포넌트의 순서쌍을 나타낸다. R은 클래스의 관계를 나타내고 C는 클래스를 나타낸다. 그림 2는 프록시 패턴을 보여주며 이를 순서쌍으로 변환하면 P={G(2,1),S(3,2)} 으로 표현된다. 각 하나의 순서쌍에서 앞부분의 영어는 클래스 다이어그램의 관계를, 숫자는 클래스를 나타낸다. 표 1은 클래스다이어그램에서의 관계를 나타내는 기호와 순서쌍에서의 약자 표시를 표로 나타내었다. 순서쌍의 뒤 부분은 관계를 가지는 두 클래스를 나타내었다. 또한 컴포넌트가 그림 3과 같다면 이를 순서쌍으로 변환하면 P=S(1,2), G(1,3), G(4,3), S(5,4), G(6,5), G(7,5)으로 변환할 수 있다. 도해적으로 보았을 때, 컴포넌트내에 프록시 패턴의 구조가 포함되어 있음을 알 수 있다. 이 컴포넌트 안에서 프록시 패턴을 찾는 알고리즘은 두 객체를 나타내는 순서쌍의 비교로써 이루어진다.

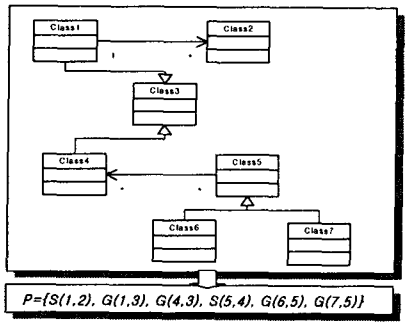

그림 3. 컴포넌트 순서쌍 표현

표 1. 클래스의 관계 표시

관 계	약 자
연관관계(Association)	S
일반화관계(Generalization)	G
집합연관관계(Aggregation)	E
복합연관관계(Composition)	D
의존관계(Dependency)	C
실체화관계(Realization)	R

Initial Step $P_1 = \{ R_1=(i_1, j_1) \mid a \rightarrow i_1, b \rightarrow j_1 \}$

Step $2 \le k \le n$

$P_1 = \{ R_k(i_k , j_k) \mid i_k=i_1$ 이면, $i_k \rightarrow a, j_j \rightarrow x$

$j_k=i_1$ 이면, $i_k \rightarrow x, j_j \rightarrow a$

$j_k=j_1$ 이면, $i_k \rightarrow x, j_j \rightarrow b$

$i_k=j_1$ 이면, $i_k \rightarrow b, j_j \rightarrow x \}$

그림 4. 컴포넌트, 패턴 순서쌍 변환 알고리즘

순서쌍으로 표현된 기본패턴과 컴포넌트를 비교하기 위하여먼저 숫자로 표현된 클래스를 변환해 주는 과정을 거친다. 그림 4는 순서쌍을 변환하는 알고리즘을 나타낸다. 변환 이유는 클래스의 숫자는 임의적인 숫자를 사용하기 때문에 순서쌍이 놓이게 되는 순서가 바뀔 때마다 서로 비교가 되지 않고 다른 결과를 나타내는 것을 방지하고 순서쌍이 놓이는 순서와 상관없이 비교하기 위한 과정이다. 변환과정은 처음에 놓이는 순서쌍을 임의의 문자로 바꾼다. 여기서는 클래스를 나타내는 첫 번째 순서쌍의 숫자로 나타낸 첫 번째 클래스는 a로 바꾸고 두 번째 클래스는 b로 바꾸어준다. 나머지 순서쌍에서 첫 번째 순서쌍의 첫 번째 클래스를 나타내는 숫자와 같은 숫자는 a로 바꾸어 주고 첫 번째 순서쌍의 두 번째 클래스를 나타내는 숫자와 같은 숫자는 b로 바꾸어 준다. 순서쌍에서 변경된 클래스와 쌍을 이루는 클래스는 어떠한 숫자로 표현된 클래스라도 비교에 있어 큰 의미가 없으므로 x로 표현하여 준다.

P=G(1,3), G(4,3), G(6,5), G(7,5), S(1,2), S(5,4)

$\Rightarrow$P=G(a,b), G(x,b), G(6,5), G(7,5), S(a,x), S(5,4)

위에서 처럼 첫 번째 순서쌍의 클래스를 (a,b)로 변환하고 1은 a로 변환되고 3은 b로 변환되었기 때문에 나머지 클래스 중에서 1은 a로 3은 b로 변환하고 a, b로 변환된 순서쌍과 쌍을 이루는 클래스는 비교에 있어 어떤 숫자를 나타내는 클래스가 와도 상관없으므로 x로 변환하였다.

```
//기본패턴 변환
Pf={Pf1,Pf2,Pf3,....Pfk)
// 패턴 순서쌍 비교를 위한 루프
If(i=0 ; i<n ; i++)
//비교를 위해 컴포넌트를 변환
Pf={Pa1,Pa2,Pa3,....pak)
//패턴과 컴포넌트를 비교해서 포함관계이면 저장하고 루프를 나온다
Pf ⊂ Pa : Break Addition();
// 구조가 불일치면 순서쌍의 순서를 바꾸어서 비교
Pa1 → Pan, Pak→Pa(k-1)
Endif
```

그림 5. 구조 비교 알고리즘

그림 5의 알고리즘에서 Pf는 변환후의 기준패턴을 나타내며 Pa는 변환후의 컴포넌트를 나타낸다. 컴포넌트를 기

준패턴과 비교하기 위해서 우선 기준패턴을 변환 알고리즘을 사용하여 변환한 다음 컴포넌트 또한 변환 과정을 거쳐 두 객체가 같은 구조를 가지고 있는지 비교한다. 이 때 컴포넌트의 관계를 나타내는 여러 개의 순서쌍 중에 기준패턴과 관련이 있는 구조를 찾아내기 위해 컴포넌트의 관계를 나타내는 순서쌍을 차례대로 변환한다. 컴포넌트의 관계를 나타내는 순서쌍에 대해 하나씩 변환 과정을 거쳐서 아래 조건 식(2)에 만족하면 컴포넌트 Pa는 기준패턴 Pf에 클러스터링된다. 모든 순서쌍을 변환하였는데도 조건 식(2)에 만족하지 않으면 다른 기준패턴과 비교하는 과정을 거친다.

$$P_f \subset P_a \qquad (2)$$

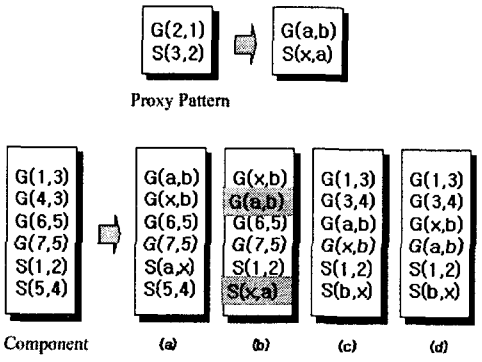

그림 6. 컴포넌트 비교 과정

그림 6은 컴포넌트과 프록시 패턴과의 비교 과정을 구체적으로 도식화 한 것이다. 프록시 패턴은 하나의 연관관계와 상속관계로 이루어져있다. 컴포넌트 중에 같은 구조를 가지는 연관관계와 상속관계를 찾아내기 위해서 프록시 패턴을 나타내는 G(1,3)을 G(a,b)로 변환하여 준다. 그리고 다른 순서쌍에도 1은 a로 3은 b로 변환을 모든 해준다. 변환 후의 프록시 패턴의 순서쌍은 G(a,b), S(x,b)로 변환된다. 컴포넌트도 변환 과정을 거쳐 비교를 하는데 컴포넌트는 4개의 상속관계를 가진다. 이중에 실제적으로 프록시 패턴의 구조를 나타내는 상속관계를 찾기 위하여 4개의 상속관계 중 하나를 변화과정을 거쳐 프록시 패턴의 변환된 순서쌍과 비교한다. 이 변환과정을 거쳐 G(a,b),S(x,b)구조를 가지는 순서쌍을 포함하고 있으면 컴포넌트는 프록시 패턴의 구조를 가지는 그룹으로 클러스터링된다. 4개의 상속관계를 나타내는 순서쌍을 비교하여 G(a,b),S(x,b)를 가지는 구조가 나타나지 않으면 프록시 패턴과 컴포넌트는 같은 구조를 갖는 패턴이라 할 수 없으므로 다른 패턴과의 비교과정을 반복한다. 기준패턴을 중심으로 클러스터링 된 컴포넌트들은 해당 기준패턴과 연관성을 부여한다. 또한 컴포넌트가 하나 이상의 패턴 구조를 가지고 있을 경우에도 해당되는 기준패턴 모두에 연관성을 짓는다. 이는 컴포넌트 검색 시 연관성을 이용하여 후보 컴포넌트들까지도 효율적으로 검색하기 위함이다.

2. E-SARM을 이용한 컴포넌트 검색

SARM(Spreading Activation Retrieval Method)은 부품과 질의어 사이에 질의어 기능을 포함하는 유사한 부품들을 검색하여 보다 더 정확하고 넓은 범위의 부품들을 찾을 수 있는 방법이다. 재사용을 위한 부품들은 각 질의어와 부품들 사이에 연결된 링크에 따라 활성값을 계산하여 검색된다. SARM에서 활성값 계산을 위한 순환을 반복하는 것은 정확한 활성값으로 인한 유사 부품을 검색하기 위한 목적이다. 이 방법으로는 각 질의어와 부품들의 활성값이 다른 질의어와 부품들의 활성값 계산에 영향을 주기 때문에 시간이 많이 걸리는 단점이 있었다. 이에 본 연구에서는 SARM의 단점을 해결하고 이를 컴포넌트 검색에 도입하였다. 해결 방법은 인덱싱이 적은 컴포넌트들의 연결정보를 제거함으로써 활성값 계산회수를 줄여 검색시간을 단축시키는 방법으로 개선하였다. 연결정보는 컴포넌트 분류시 기준패턴을 중심으로 클러스터링된 컴포넌트와 기준패턴 사이에 부여된 연관성을 의미한다. 즉, 개선된 SARM(Enhanced Spreading Activation Retrieval Method : E-SARM)[15]은 순환과정이 일정 수준 반복된 후 기준에 미치지 못하는 기준패턴이나 컴포넌트들의 연결정보를 제거하여 연산에서 제외시킴으로써 검색의 확장범위를 줄여 보다 관계가 깊은 후보컴포넌트들만을 검색하도록 하였다.

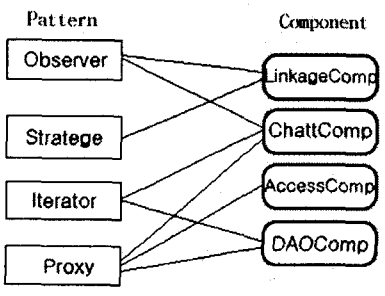

그림 7. 컴포넌트와 기준패턴과의 관계

그림 7은 기준패턴와 컴포넌트의 연결관계를 예를 들어 보여주고 있다. 두 개 이상의 기준패턴과 연결된 컴포넌트는 컴포넌트 구조 내에 두 개 이상의 기준패턴 구조와 특성을 가지고 있음을 나타낸다. 질의로 기준패턴 「Observer」를 입력하면 3개의 컴포넌트가 검색된다. 여기서 기준패턴 「Observer」는 컴포넌트 「LinkageComp」와 「ChattComp」에 직접 연결되어 있지만 컴포넌트 「AccessComp」와 「DAOComp」에는 연결되어 있지 않다. 그러나 「Observer」(질의:기준패턴)→「ChattComp」(컴포넌트)→「Iterator」(기준패턴)→「DAOComp」(컴포넌트)를 통하여 연결되고, 「Observer」(질의:기준패턴)→「ChattComp」(컴포넌트)→「Iterator」(기준패턴)→「AccessComp」(컴포넌트)를 통하여 2개의 컴포넌트(「AccessComp」, 「DAOComp」)가 연결됨을 알 수 있다. 하지만 「AccessComp」는 검색과정에서 적게 참조되므로 연결이 제거된다. 이처럼 각 기준패턴와 컴포넌트는 서로 연결되어 있는 노드를 참조해 가면서 활성값을 계산하게 된다. 또한 계산결과 「LinkageComp」가 「ChattComp」보다 높게 나타나는데 그 이유는 「LinkageComp」가 「ChattComp」보다 참조 회수가 더 많기 때문이다. 기존의 SARM과 E-SARM의 평균 참조 회수 시뮬레이션 결과 활성값 계산회수가 37.8% 감소됨을 알 수 있었다. 표 2는 그림 8의 E-SARM알고리즘에서 사용하는 활성값 계산 변수들에 대한 정의를 보여주고 있고, 알고리즘에서 기준패턴과 컴포넌트들은 메트릭(metrics)을 이용하여 활성값을 계산하였다. 순환이 반복될수록 활성값은 안정되며 참조회수가 기준에 미달되는 부분은 자동으로 제거되어 계산과정이 종료된다.

표 2. E-SARM 파라미터

기호	정 의
$\delta_{D_i}(t)$	기준패턴 감소비율로 감소된 이전의 활성값
$\psi_{D_i}(t)$	현재(t) 컴포넌트 i에게 들어오는 입력값
M	최대 활성값 = 1.0
$D_i(t)$	이전 단계의 누적 컴포넌트 활성값
θ_D	컴포넌트 감소 비율 = 0.1
$\delta_{T_i}(t)$	컴포넌트 감소비율로 감소된 이전의 활성값
$\psi_{T_i}(t)$	현재(t) 기준패턴 i에게 들어오는 입력값
m	최소 활성값 = -0.2
$T_i(t)$	이전 단계의 누적 기준패턴 활성값
θ_T	기준패턴 감소 비율 = -0.2

```
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) + ψ_Di(t)(M − D_i(t))  if ψ_Di(t) > 0
 δ_Di(t) + ψ_Di(t)(D_i(t) − m)  if ψ_Di(t) ≤ 0
 δ_Di = (1 − θ_D)D_i(t)
    else if(a component)
 Tj(t+1) =
 δ_Ti(t) + ψ_Ti(t)(M − T_i(t))  if ψ_Ti(t) > 0
 δ_Ti(t) + ψ_Ti(t)(T_j(t) − m)  if ψ_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) /*MAX_CYCLE=3~5*/
          break;
  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;
       // 제거된 노드의 초기화
     }
endwhile
```

IV. 시스템 설계

1. 데이터베이스

본 연구의 시스템 저장소는 기준패턴과 컴포넌트에 대한 일반적인 정보가 저장되는 정보 데이터베이스와 UML의 클래스 다이어그램인 구조에 대한 정보가 저장되는 구조 데이터베이스로 구성되어 있다.

strCargegory는 컴포넌트가 속한 도메인 분류에 대한 정보이며, docDocument는 컴포넌트의 원리와 의도, 적용될 수 있는 상황, 컴포넌트를 구현 시 인식해야 하는 함정, 힌트, 기술 등 컴포넌트에 대한 전반적인 설명을 위한 필드이다. relCom는 클러스터링에 의해 분류된 컴포넌트에 연관된 기준패턴을 나타내며, UserID는 컴포넌트를 생성하여 등록한 사용자에 대한 아이디를 나타낸다. 표 3은 컴포넌트정보 데이터베이스에 대한 테이블 구조를 나타낸다. 패턴정보 데이터베이스에 대한 테이블 구조도 이와 유사하다. 표 4는 컴포넌트의 구조를 표현하기 위한 정보를 가지고 있는 데이터베이스로서 컴포넌트구조 정보로는 컴포넌트에 대한 클래스 다이어그램의 구조를 표현한 이미지와 컴포넌트를 분류하기 위해 클러스터링을 하기 위한 컴포넌트에 대한 순서쌍 정보를 포함하고 있다. 패턴구조 데이터베이스에 대한 테이블 구조도 이와 유사하다.

표 3. 컴포넌트정보 데이터베이스의 테이블 구조

필드 이름	데이터 형식	설명
strName	char	컴포넌트에 대한 이름
strCartegory	char	도메인 분류에 의한 카테고리
docDocument	text	컴포넌트에 대한 설명
relCom	Link	컴포넌트에 연관된 기준패턴
UserID	char	컴포넌트 등록자 이름

표 4. 컴포넌트구조 데이터베이스의 테이블 구조

필드 이름	데이터 형식	설명
strName	char	컴포넌트에 대한 이름
imgStructure	image	컴포넌트구조에 대한 이미지
StrPair	char	순서쌍 정보
strPattern	char	관련 패턴

2. 시스템 구조

본 연구의 전체 시스템 구조는 그림 9와 같이 컴포넌트 검색기, 컴포넌트 브라우저, 패턴 브라우저, S·A 실행기의 4개 도구로 이루어져 있다. 추상적인 패턴 구조와 구체적인 컴포넌트 구조를 저장하기 위하여 패턴 라이브러리와 컴포넌트 라이브러리를 분리하여 구축하였다.

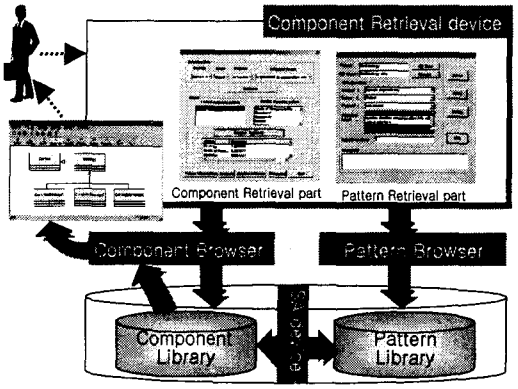

그림 9. 시스템 구조

기준패턴이 되는 대상 디자인 패턴은 Gamma의 패턴 중 구조패턴과 행위패턴을 사용하며, 패턴 구조를 제공하지 않은 패턴은 제외시켰다. 개발자는 컴포넌트 검색기를 사용하여 재사용 하고자하는 컴포넌트를 검색할 수 있다. 컴포넌트에 대한 정보를 함께 제공해주는 인터페이스를 이용하여 검색하고자하는 컴포넌트의 도메인 분야, 기능, 컴포넌트 타입, 사용 OS 등과 같은 특징으로 선택하면, 컴포넌트 브라우저가 컴포넌트 라이브러리에서 이를 만족하는 컴포넌트를 검색한다. 검색된 컴포넌트는 S·A 실행기에 의해 기준패턴과의 활성값이 계산되어져 유사한 후보컴포넌트들을 추출한다. 또한 질의를 컴포넌트뿐 아니라 패턴에 의해서도 수행될 수 있도록 하여, 원하는 컴포넌트를 정확히 설정할 수 없을 때, 질의한 기준패턴의 역할이나 기능을 수행하는 컴포넌트를 검색할 수 있도록 지원한다. 이때에는 패턴 브라우저가 기준패턴을 검색하고 S·A 실행기를 구동하여 컴포넌트들을 검색한다. 본 연구에서 제안한 시스템은 질의와 가장 적합한 컴포넌트와 그와 유사한 후보컴포넌트들이 활성값 순으로 제공됨으로서 컴포넌트 재사용 효율을 높여 줄 수 있다.

V. 성능 평가

기존의 컴포넌트 관련 시스템과 본 연구에서 설계한 시스템을 비교하면 표 5와 같다. 대표적인 컴포넌트 개발 도구는 Select 사의 SCF(Software Component Factory)[16]와 TogetherSoft사의 Together[17], 그리고 Computer Associates사의 COOL:Joe2.0[18]등이 있다. 이들 세 가지 시스템과 본 연구에서 제안한 방법을 분류와 검색 측면에 초점을 두고 비교하였다. 특히 컴포넌트의 생성 프로세스의 단계 중 설계단계에서 컴포넌트의 재사용을 얼마나 효율적으로 실행할 수 있는가를 비교하였다.

표 5. 타 시스템과의 비교

	SCF	Together	COOL:Joe 2.0	제안한 시스템
컴포넌트 식별 지원	×	×	△	○
패턴 적용	×	×	×	○
후보컴포넌트 검색	×	×	×	○
도메인 단위 재사용	×	×	×	○
컴포넌트 모델링지원	×	×	플랫폼 종속적	플랫폼 독립적
도메인 모델링지원	-BP모델링 -UC모델링	-UML 다이어그램 모델링	-UC모델링 -Type다이어그램	-UC모델링 -객체모델링 -순차도

"×"로 표시된 부분은 시스템에서 지원하지 않는 기능이고, "△"는 일부 또는 미약하게 지원되는 부분이며, "○"는 완전한 기능을 제공해 주는 것을 나타낸다. SCF와 Together는 컴포넌트 내부 설계와 구현 기능은 지원하지만, 컴포넌트 식별과 컴포넌트를 다이어그램으로 표현해주는 컴포넌트 모델링 기능은 지원하지 못한다. COOL:Joe2.0은 시스템 아키텍쳐 상 EJB 컴포넌트만을 식별할 수 있으며, 컴포넌트 모델링 기능도 지원해 준다. 하지만, COOL:Joe는 플랫폼 의존적인 컴포넌트를 생성하기 때문에 추후에 다른 플랫폼인 CCM이나 COM 환경에서는 재사용하는데 문제가 발생한다.

본 연구가 제안한 시스템은 플랫폼 간의 재사용을 높이기 위하여 컴포넌트에 적용된 패턴 구조만을 고려하여 컴포넌트를 검색함으로써 구현 상세 코드는 숨기고 컴포넌트 구조를 다이어그램으로 제공해 준다. 재사용 가능한 컴포넌트를 기능별로 분할하고 그 구조를 다이어그램으로 제공함으로서 컴포넌트의 재사용 및 플랫폼간의 이식성을 높일 수 있다. 또한 후보컴포넌트의 검색도 지원해 주며, 도메인별로 컴포넌트를 분류하였기 때문에 도메인 단위의 대규모 재사용도 가능하다.

VI. 결 론

컴포넌트 단위로 소프트웨어를 개발할 때, 다양한 컴포넌트들에 대한 사용자의 요구사항에 맞는 컴포넌트의 검색과 그 컴포넌트의 조립 활용이 용이한 인터페이스가 필요하다. 또한 많은 컴포넌트들의 효율적 관리를 위한 컴포넌트 분류방법이 필수적이다. 이에 본 연구에서는 컴포넌트의 효율적 관리와 사용자 요구에 맞는 유사한 후보컴포넌트까지 우선 순위로 검색하여 보다 정확한 컴포넌트를 제공할 수 있도록 하였다. 기존 시스템의 컴포넌트들을 응용 도메인으로 나누어 각 도메인 별로 기준패턴의 구조를 이용한 클러스터링을 시행하여 컴포넌트 저장소를 구축한다. 기준패턴과 컴포넌트의 구조는 클래스와 클래스간의 관계를 순서쌍으로 변화시켜 컴포넌트 내에 기준패턴의 구조가 있는지를 조사한다. 클러스터링된 컴포넌트는 기준패턴과 연관성을 가지게 되고, 이 연관정보를 이용한 E-SARM 알고리즘을 시행함으로써 컴포넌트 검색에 정확성을 높였다. 검색된 컴포넌트는 우선 순위에 따라 여러 개의 후보컴포넌트로 나타나며, 그 구조를 다이어그램으로 제공함으로써 개발자가 새로운 시스템을 설계할 때 상세코드나 구현에 구애받지 않고 설계 단계에서의 컴포넌트를 재사용 할 수 있으므로 플랫폼 독립적인 설계가 가능하다.

향후 연구는 다양한 도메인과 컴포넌트에 폭 넓게 적용하여 플랫폼 특성에 맞는 상세 모델링과 효율적인 컴포넌트 조립방법을 연구하는데 있다. 또한 하나의 컴포넌트 내에 두 개 이상의 패턴 구조가 적용될 때의 분류방법에 대한 연구가 진행중이다.

참 고 문 헌

[1] George T. Heineman, William T. Councill, "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 Engineering and Methodology, Vol. 2, No. 3, 1993.

[4] A. M. Zaremski, J. M. Wing, "Specification Matching of Software Components," 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 Information Processing 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 Information Processing 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, J.Vlissides, "Design Pattern : Elements of Reusable Object-Oriented Software," Addison-Wesley, 1995.

[10] F. Buschman, R.Meunier, H.Rohnert, P.Sommerland, Stal Michael, "Pattern-Oriented Software Architecture-A System of Pattern," John Wiley&Sons, 1996.

[11] W.Tichy, "Essential Software Design Pattern," University of Karsruhe, 1997.

[12] P.Bengtsson, J.Bosch, "Scenario-based Software Architecture Reengineering," in Proceeding 20th ICSE, IEEE, Jun. 1998.

[13] Shim, In-Sup Pail "The Value-Added Brokerae Concept for Steering the CBSD Environments," The Transactions of Korea Information Processing Society, Vol.8, No.6, pp. 681-690, Dec. 2001.

[14] 조회진, 하정수, 김진삼, 박창순, "컴포넌트 기반 시스템 개발 방법론 마르미-III," 프로젝트 관리 기술 논문집, 제 4권(통권 제 4호), 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, 4. 2000.

[16] http://www.anonix.com/

[17] http://www.together.com/

[18] http://www.cai.com/

김 귀 정(Gui-Jung Kim) 정회원

1994년 2월 : 한남대학교 전자계산공학과 졸업(공학사)
1996년 2월 : 한남대학교 전자계산공학과 졸업(공학석사)
1999년 8월 : 경희대학교 전자계산공학과 수료
2001년 9월 ~ 현재 : 건양대학교 IT학부 교수
<관심분야> : S/W 재사용, Component, CASE