소프트웨어 재사용 지원 정보 저장소 구축

특집
02

김 행 곤
(대구가톨릭대학교)

목 차

1. 서 론

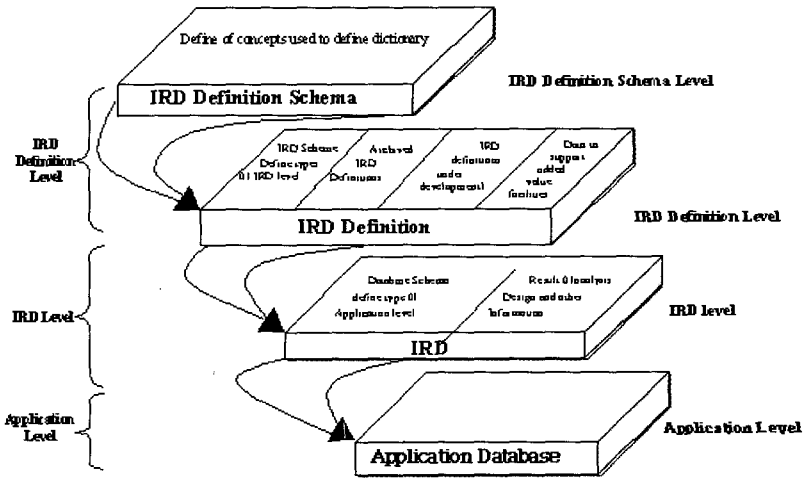
소프트웨어의 품질과 생산성 향상을 위한 방법론들에 관한 연구들이 많이 진행되어 왔다[2]. 기존의 OOP(Object Oriented Program)로 부터 RAD(Rapid Application Development), CBD (Component Based Development), 그리고 시각화 프로그램 개발 단계로 전이하게 되었다. 이는 재사용(reuse), 저장소(repository), 그리고 재공학(reengineering)과 같은 3R로의 접근을 부각시키게 되었고, 그 중에서 재사용 측면에 대한 연구가 활발하다[1].

1990년대에 도입된 객체지향 방법론 역시 생산성과 품질 측면에서 많은 장점을 가지고 있다. 그러나 개발 초기에 많은 비용과 시간을 소비하며, 상용화된 클래스 라이브러리가 부족하다는 단점을 갖고 있다. 비록 소프트웨어 부품이 있다 하더라도 방대한 양의 클래스 라이브러리를 사람의 기억력에 의존하여 검색하기란 불가능하다고 할 수 있다. 그러므로 사용자가 원하는 부품을 정확하고 빨리 찾기 위하여 자동화된 방법이 제공되어야 한다. 따라서 소프트웨어 생산성을 향상시키기 위해서는 객체지향·기법을 이용한 재사용 시스템 활용이 필요하며, 재사용 시스템의 핵심 구성 요소인 정보 저장소와 효율적인 재사용 지원을 위한 저장, 검색 기술이 절실히 요구된다. 본 논문에서는 클래스 정보의 효율인 재사용을 지원하기 위해 저장, 검색 기술을 바탕으로 하는 정보 저장소 형상관리 시스템에 대해 연구한다.

2. 관련 연구

CASE는 대규모 소프트웨어 개발 환경을 제공하는 없어서는 안 될 도구가 되었다. 이러한 소프트웨어 개발 환경에서 도구 통합의 중심적인 기능을 수행하는 것이 정보 저장소이다. 이러한 소프트웨어 저장소를 위해 현재 CASE 도구 통합 및 도구 표준화를 위한 정보 저장소 표준안이 제시되었다.

2.1 IRDS

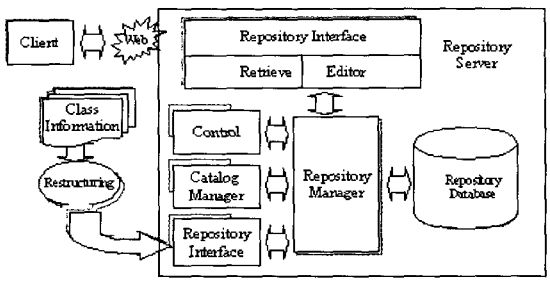


(그림 1) IRDS의 세 계층 구조

IRDS[2]는 기업의 정보 자원을 설계, 모니터, 제어, 보호하기 위한 자료사전이다. IRDS는 CASE 도구를 이용해 사용자 응용프로그램을 개발하는 전제조건이라 할 수 있다. (그림 1)은 IRDS의 세 계층구조 이며, 다음의 세 가지 목적을 위해 개발되었다. 기업의 정보 모델링과 기업의 사무를 전산화하는 사용자 어플리케이션 작성을 위한 소프트웨어 개발 환경 구축, 사용자 어플리케이션의 수행 환경 구축 등이다.

2.2 PCTE

PCTE[3]는 소프트웨어 공학 환경, 즉, SEE(Software Engineering Environment)의 구축을 위해 개발된 시스템이다. PCTE는 소프트웨어 개발 방법론 혹은 응용 도메인을 위한 소프트웨어 도구와 서비스들의 통합을 지원하는 시스템이다. PCTE는 객체 관리, 스키마 관리, 프로세스의 수행과 상호간 통신, 병렬 수행 및 무결성 제어. 분산 환경, 보안 및 사용자 관리 등의 메커니즘들로 구성되어 있다. 각 메커니즘은 소프트웨어 도구들이 사용할 수 있는 인터페이스를 제공하며, PCTE는 소프트웨어 공학 환경을 구축하기 위한 공통 메커니즘들을 위한 인터페이스를 정의한 것이다.



(그림 2) 저장소 구조

3. 저장소 설계

3.1 저장소 구조

본 논문에서 제안하는 재사용 지원을 위한 저장소 구조는 (그림 2)와 같이 크게 3가지 모듈로 구성되어 있다. 재구성기(Reconstructor), 정보저장소(Repository) 및 관리기(Repository Manager)로 구성된다.

시스템을 구성하는 각 서브 시스템의 기능은 다음과 같다.

3.1.1 저장소 인터페이스

저장소 인터페이스는 저장소 관리 시스템과 클라이언트 시스템과의 인터페이스를 제공한다. 또한 저장소 기능에 대한 상위 레벨 인터페이스를 제공하며, 모든 저장소 접근은 반드시 상위레벨 인터페이스인 서버 인터페이스 계층을 통해서 이루어지며, 클라이언트 시스템과의 통신은 Web을 통해 이루어진다.

3.1.2 카탈로그 관리자

부품 정보에 관한 상위 정보 관리와 부품 정보, 접근 권한, 작성 일자, 작성자 등의 카탈로그를 관리한다. 또한 저장소 내 클래스 정보 목록 등도 관리한다.

3.1.3 버전 및 형상 관리자

버전 관리 시스템은 부품 데이터베이스에 저장된 부품의 여러 형상에 대한 버전을 관리하며, 부품의 등록이나 수정 시에 발생하는 부품의 새로운 버전 처리를 수행한다.

3.1.4 저장소 관리자

부품의 논리적 정보 모델을 하부 저장 레벨로 매핑을 담당하고, 클라이언트와 코드 생성기로부터 부품 저장소에 대한 요구를 하부 저장 모듈에 대한 요구로 변환을 담당한다.

3.1.5 저장소 데이터베이스

재사용 될 부품을, 부품의 분류상태를 잘 나타내고, 검색을 용이하게 할 수 있도록 저장한다. 저장소 데이터베이스는 PSE 데이터베이스를 기반으로 구현된다.

Web을 기반으로 한 클라이언트에서는 부품 정보를 등록하고, 관리, 검색 등의 요구를 서버 측에 전송하면 서버는 클라이언트로부터 요청받은 부품 관리를 위한 연산을 수행하여 클라이언트로 응답한다. 클라이언트는 서버로부터 전달받은 정보를 편집, 출력하는 기능을 담당한다.

본 논문에서는 기존의 클래스 계층 구조를 유사도 계산을 통하여 계층구조를 재구성하는 것으로 제한하였다.

3.2 클래스 계층 재구성

본 절은 클래스 부품의 계층 구조 재구성을 위한 알고리즘을 작성하기 위해 기존의 객체지향 언어의 시멘틱 모델[6]과는 달리 C++ 클래스 구조, 메소드 정의 및 메소드 사이의 인용 관계를 비롯하여 클래스에 가해지는 오퍼레이션을 정의하기 위한 형식적인 클래스 계층 모델을 정의한다.

3.2.1 클래스 계층 모델

클래스 계층 구조 구성을 알고리즘으로 수행하기 위해 본 절에서는 다루고자 하는 클래스 계층을 나타내는 기본 구조를 정의한다.

가. 기본정의 및 구조

<표 1>은 모델을 정의하기 위해 사용하는 기호를 나타낸 부분이다.

<표 1> 계층 모델 표현에 이용되는 기호

<....>	튜플 또는 레코드
{...}	집합 또는 리스트
X[A]	A에 대한 X의 프로젝션, if X=A x B
X	cartesian product
x[A]	x의 성분 A, x∈X
P(s)	s의 멱집합(power set)

(그림 3)은 클래스에 대한 내용을 정의한 것으로 클래스의 계층은 클래스의 집합으로 이루어져 있으며 각 클래스는 인스턴스 변수와 메소드들로 구성된다.

```
Class={Ci}
MethodName={mi}, VarName={vari}, ArgName={Argi}
ArgList ={ ai}
Body = { bi}
Signature = MethodNAme x ArgList
Method = AccessType x Signature x Body
AccessType = {PRIVAT E, PUBLIC, PROTECTED}
Variable = VarName x Type type∈Class
∀(type Type)   or   type∈any
```

(그림 3) 클래스의 기본 모델

나. 클래스의 구성과 상속 관계

(그림 4)는 클래스들이 상속 관계가 있을 때 상위 클래스에서 선언된 메소드와 인스턴스 변수가 하위 클래스로 상속되며 상위 클래스에서 선언된 메소드가 하위 클래스에서 오버라이딩되는 관계를 표시한다.

```
IntroVar:Class->P(Variable)
IntroMeth:Class->p(Method)
Ancestor:Class->P(Class)
Inhvar:Class x Class->P(Method)
∀(c∈Class), ∀(a∈Ancestor(c)), InhVar(c,a)=DefVar(a)
InhMeth:Class x Class -> P(Method)
∀(c∈Class),∀(a∈Ancestor(c)),InhMeth(c,a)=DefMeth(a)
DefVar(c)=IntroVar(c) ∪ ( ∪ InhVar(c,a) ∀(a∈ancestor(c))
DefMeth(c)=Intro(c) ∪ (∪ VisiMeth(c,a)) ∀(a∈ancestor(c))
VisiMeth(c,a)=Inhmeth(c,a)-OverMeth(c,a)
OverMeth(c,a)={m InhMeth(c,a) m[MethodName] IntroMeth[MethodName]}
RefMeth:Class ->P(Method)
RefMeth(c)=DefMeth(c),∀c∈Class
Interface:Class ->P(MethodName), Interface(c)∈DefMeth(c)[MethodName]
```

(그림 4) 상속 관계의 정의 및 메소드 변경 오퍼레이션

다. 인스턴스 변수와의 종속 관계 정의

(그림 5)는 클래스에서 정의된 메소드간의 종속 관계를 표시한 것으로 메소드와 인스턴스 변수간의 종속성 관계는 한 클래스의 메소드를 호출하는 경우이며, 메소드의 내용이 바뀔 때에 고려해야 하는 중요한 관계이다.

```
Methodep:Body ->ClassRef x MethodName
 VarDep:Body -> ClassRef -> Varname
 Methodep(VIRTUAL)=0
ClassRef=class ∪ {This}
  ∀(c∈class), ∀(m∈IntroRefMeth(c))
   ∀(d∈VarDep(m[Body]))
          d[classRef]∈Ancestor(c)
(d[classRef] !=c) -> ∧
          d[VarName∈Inhvar(c,d[classRef])[VarName]
∨(d Methodep(m[body])
  (d[clssRef]!=c]->d[classRef]∈Ancestor(c)∧
          d[MethodName]∈InhMeth(c,d[classRef])[MethodName]
  (d[classRef=c)-> d[Methodname∈IntroMeth(c)[MethodName]
```

(그림 5) 메소드 사이의 종속 관계

라. 기본 오퍼레이션

(그림 6)은 새로운 클래스를 생성하거나 클래스 리스트 내에 추가될 클래스의 위치를 찾고 그 위치에 클래스를 삽입하거나 클래스를 삭제하는 기본 연산을 나타내고 있으며, 그 내용에 대한 상세한 설명 없이 다른 알고리즘에서 이용된다.

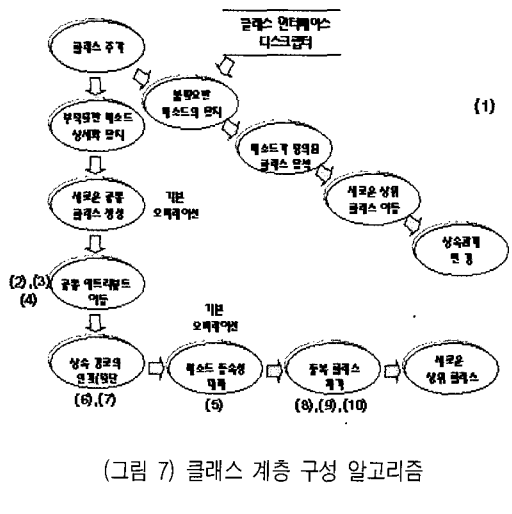
```
Create_class(class)
listposition(class_list,class,pos)
listInsert(class_list,class,pos)
listDelete(class-list,class)
```

(그림 6) 기본 오퍼레이션

3.2.2 알고리즘 적용

클래스 계층 구성을 위한 알고리즘을 적용하기 전에 알고리즘이 만족해야 하는 특성은 다음과 같다.

- 보존성 : 재구성이 수행되기 이전에 계층 내에 존재하는 모든 클래스의 정의는 보존 되어야 한다.
- 중복성 배제 : 재구성으로 생성된 클래스들에 나타난 인스턴스 변수와 메소드는 그 클래스들의 공통된 조상 클래스로부터 상속되어야 한다.
- 최소성 : 계층을 재배열하는 데 필요한 추가 클래스의 수는 최소화 되어야 한다.



(그림 7) 클래스 계층 구성 알고리즘

```
    procedure Ecec_Common
    parameters
       input-output: add_class∈class
    variable
       new_class∈class
       sup_class∈class
       supflag∈Boolean
Begin
     flag=true
     for each sup_class in Ancestor(add)class) do
              if TargetMeth(add)class,sup_class) !=0
       then
              Creat_class(new_class)
              Link(add_class,new_class,sup_class)
              Transfer_Attrib(add_class,new_class,sup_class)
              Unlink(add_class,sup_link)
              Replace_dep(add_class,sup_class,new_class)
              MapAnc(add_class,sup_class)=sup_class
           else
              MApAnc(add_class,sup_class)=sup_class
           end-if
     for each sup_class in NewAnc(add_class) do
           supress_common(new_class,sup_class,flag)
     done
end
```

(그림 8) 공통 클래스의 추출 알고리즘

이러한 조건을 만족하는 클래스 계층 재구성 알고리즘은 (그림 7)의 흐름도로 표현되며 각 단계별로 나누어 처리된다.

가. 계층 구성 단계

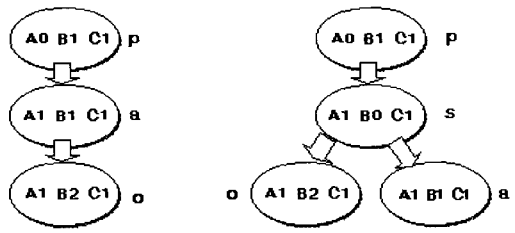
- 적법한 상속 관계가 설정되었는가를 검사한다.
- 구성 단계로서 추가될 부품에 대해 새로운 부품의 생성, 상속 링크의 연결/절단, 메소드와 인스턴스 변수의 이동, 클래스 또는 메소드간의 종속관계 대체를 수행한다.
- 재구성된 클래스들과 추가로 생성된 클래스를 나타내는 결과의 그래프 구조가 반환된다.

(그림 8)은 계층 구성 알고리즘의 전체 흐름을 표현한 것으로 클래스 라이브러리 내에 계층의 부적절한 구성은 기존 클래스를 이용하기 위해 새로운 클래스가 추가될 때 밝혀진다. 공통성의 추출 과정은 우선 부적절한 메소드의 상세화가 탐지될 때마다 클래스 사이의 공통성을 위한 새로운 클래스가 알고리즘 내의 기본 오퍼레이션에 의해 생성되며 공통 애트리뷰트는 새로운 클래스로 이동되고 상속 경로 또한 변경된다.

기존 클래스 사이의 메소드 종속성은 대체되고 공통 클래스를 생성하는 과정에서 작성되는 클래스는 배제된다. 구현을 위한 상속 구조를 적절한 구조로 변경하는 데 있어서 메소드의 행위를 파악하기 위해 클래스 인터페이스 디스크립터를 검사한 후 불필요한 메소드가 정의된 클래스를 찾고 포함관계로 상속 구조를 대체한다.

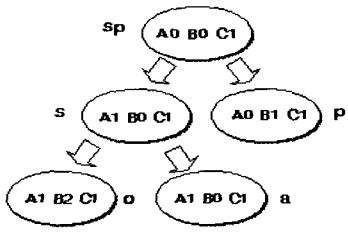
나. 알고리즘의 적용

계층 재구성 알고리즘에 의해 변경되는 클래스 계층을 나타내면 (그림 9)와 같다. 초기클래스 계층은 2개의 클래스로 구성되어 있고 여기에 새로운 클래스를 추가할 때 클래스 계층의 변경을 나타내고 있다. 볼드체는 그 클래스 내에 정의된 메소드를 말하며, 첨자는 같은 메소드를 다르게 구현한 것이며, o의 표시는 그 메소드가 가상 메소드임을 나타낸다.



(a) (b)

(그림 9) (a) 클래스 O의 추가 (b) 1차로 재구성된 형태



(그림 10) 최종 계층의 재구성 상태

(그림 10)은 (그림 9)의 (a)와 같은 방법으로 새로운 s, p를 생성하고 상속 경로의 연결/절단 및 메소드의 종속성 대체가 발생한다.

3.3 분류 및 인덱싱 방법

부품의 분류 및 인덱싱 방법은 시스템에서 제공되는 검색 방법을 효율적으로 지원해 주며, 부품의 대상 영역의 특징을 잘 표현해 줄 수 있어야 한다. 본 논문에서는 객체 부품의 질의에 의한 검색과 브라우징을 통한 검색의 두 가지 검색 방법을 제공한다.

3.3.1 클러스터링

데이터베이스 관리 시스템에서는 원하는 정보를 검색할 경우, 정보는 질의에 완전히 부합될 경우에만 검색된다. 그러나 이러한 제한적인 방법을 부품 저장소에 적용하기는 바람직하지 못하다[5]. 이러한 문제를 해결하기 위해서는 저장소의 부품을 분류하는 단계에서 비슷한 속성을 지니고 있는 부품들을 하나의 클러스터로 모아야 한다. 이러한 분류의 가장 중요한 목적은 사용자 요구에 가장 근접하고 수정이 가장 용이한 부품을 제공하여 기능이 비슷한 부품들로 브라우징을 가능하게 하는 것이다.

본 논문에서 제안한 클러스터링을 통해 부품의 전체적인 구조를 재구성하여 브라우징에 의한 탐색을 지원한다. 이는 부품을 등록할 때마다 유사도를 계산하여 부품의 전체 계층구조 중에서 새로운 부품에 가장 알맞은 자리를 찾아 새로운 부품을 추가하여 주는 방법이다.

가. 유사도 계산

부품을 클러스터링할 경우, 새로운 부품과 저장소 내의 부품간의 유사도 계산을 통하여 새로운 계층구조를 구축한다.

1) 데이터 유사도

유사도 함수의 적합성은 다음 조건을 만족하면 유사도 함수가 소프트웨어 부품을 분류하는데 의미가 있다고 하였다.

- 매칭조건 : 두 부품에서 공통된 멤버 데이터(또는 멤버 함수) 또는 공통되지 않은 멤버 데이터를 변수로 유사성 함수가 구성되어야 한다.
- 단조성 : 두 부품에서 공통된 멤버 데이터가 하나 추가되면 유사도는 증가하고, 공통되지 않은 멤버 데이터가 추가되면 유사도는 감소한다.
- 두 클래스 사이에 공통된 멤버 데이터가 존재하지 않으면 유사도는 0이다.

위와 같은 조건에 부합되도록 다음과 같은 부품의 데이터 유사도 함수를 정의할 수 있다.

$$S_d(A,B)=\frac{d^2(A\cap B)}{d(A)\times d(B)}$$

- d(A) : 클래스 A의 멤버 데이터 갯수,
- d(B) : 클래스 B의 멤버 데이터 개수,
- d^2(A∩B) : A와 B의 공통된 멤버 데이터 갯수.

여기서 두 부품의 공통된 멤버 데이터란, 두 부품에 모두 존재하는 동일한 멤버 데이터를 말하며, 두 멤버 데이터가 동일하다는 것은 두 멤버 데이터의 이름과 데이터의 형이 같다는 것으로 정의한다.

2) 함수의 유사도

클래스 함수에 대한 유사도 데이터의 유사도와 마찬가지로, 두 클래스간의 공통된 멤버 함수가 많을수록 유사도가 증가하고, 공통되지 않은 멤버 함수가 많을수록 유사도가 감소한다.

$$S_f(A,B)=\frac{f^2(A\cap B)}{f(A)\times f(B)}$$

- f(A) : 클래스 A의 멤버 함수의 개수,
- f(B) : 클래스 B의 멤버 함수의 개수,
- f^2(A∩B) : A 와 B의 공통된 멤버 함수의 개수

여기서 부품 A 와 부품 B의 공통된 멤버 함수란, 클래스 A와 B 동시에 존재하는 동일한 멤버 함수를 말하며, 두 멤버함수가 동일하다는 것은

두 멤버함수의 이름이 같고, 반환 값의 데이터 형이 같고, 인수의 이름과 데이터형, 인수의 갯수가 같다는 것으로 정의한다.

3) 클래스 유사도

클래스의 유사도는 데이터 유사도와 함수 유사도를 이용하여 구할 수 있으며, 각각의 데이터 함수가 가지는 중요도는 동일하다고 가정한다. 클래스의 유사도는 다음과 같이 정의할 수 있다.

$$S(A,B) = P \times S_d(A,B) + (1-P) \times S_f(A,B)$$

- P : 두 클래스의 멤버 데이터의 총 수를 두 클래스의 멤버 데이터와 함수의 총 수로 나눈 값

3.3.2 부품의 계층구조와 등록

가. 부품의 계층구조

클라이언트로부터 새로운 부품이 등록되면 저장소에 저장되어 있는 기존 부품의 계층구조에 유사도 조사에 의한 결과에 따라 알맞은 위치에 삽입된다. 부품의 계층 구조는 브라우징을 통한 경로 탐험 검색을 위한 구조이며, 이는 좀 더 일반적인 기능을 가지는 부품의 상위 계층에 위치하며, 좀 더 특수한 기능을 갖는 부품은 하위 계층에 위치되는 구조이다. 이를 위하여 부품의 일반성을<표 2>와 같이 정의한다.

〈표 2〉 부품의 일반성

- 부품의 일반성

클래스 A 와 클래스 B에 대하여

$m(A-B) \le m(B-A)$일 때,

- 클래스 A는 클래스 B 보다 일반적이다
- m(A-B) : 클래스 A에서 클래스 B와 공통되는 멤버를 제외한 멤버의 수
- m(B-A) : 클래스 B에서 클래스 A와 공통되는 멤버를 제외한 멤버의 수

본 논문에서 부품의 계층구조는 위에서 정의한 일반성을 기반으로, 더 일반적인 부품이 상위 계층으로, 더 특수한 부품이 하위 계층으로 위치하는 트리 구조를 가진다. 위에서 정의한 일반성은 부분 순서 관계를 이루며, 트리 구조에서 최상위 부품에서 트리 구조의 단말 노드에 이르는 모든 경로는 일반성의 정의에 의한 순서 관계를 가진다.

나. 부품 등록 방법

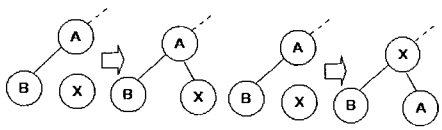
새로운 부품을 계층구조에 삽입하기 위해서는, 우선, 기존의 부품 중 새로운 부품과 유사도가 가장 높은 부품을 찾아 부품을 삽입한다. 그 과정은 다음과 같다.

- 새로운 부품 X가 등록되면 기존의 부품과 새로운 부품의 클래스 유사도를 측정하여 새로운 부품과 가장 유사도가 높은 부품 A를 찾는다.
- 새로운 부품 X를 부품의 계층구조에 삽입한다.

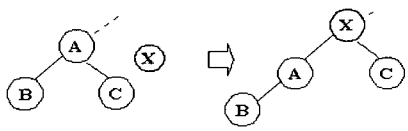
부품 A의 형태에 따라 다음의 세 가지 삽입 형태가 있다.

- A의 자식 노드가 없을 경우
- A의 자식 노드가 하나 있을 때
- A가 두 개의 자식 노드를 가질 때

저장소에 클래스 정보를 등록하는 과정은 다음 순서에 따른다.



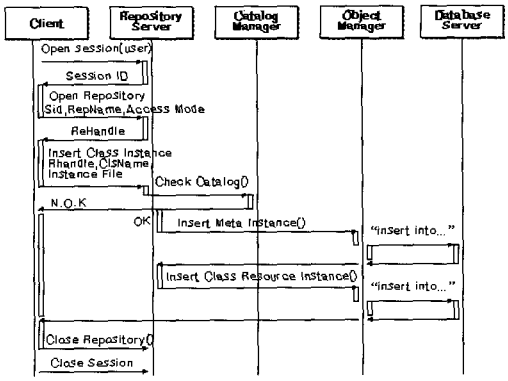
(그림 11) 자식 노드가 없는 경우



(그림 12) X가 B 보다 일반적인 경우

3.3.3 검색 방법

객체 클래스 저장소에 저장되어 있는 부품에



(그림 13) 클래스 정보 등록 절차

대한 검색은 클라이언트 시스템의 요청에 대하여 저장소가 서비스를 제공하는 형태로 이루어진다. 클라이언트에서 요청할 수 있는 검색 형태는 질의에 의한 검색과 브라우징에 의한 경로 탐험 검색의 두 가지로 이루어져 있다.

가. 질의에 의한 검색

질의에 의한 검색은 사용자가 입력한 값에 가장 가까운 부품을 찾아 주는 것이다. 사용자 질의의 형식은 각 패싯 값과 저장소에 저장된 부품의 패싯 값을 비교하여 가장 비슷한 부품을 찾아내어 사용자에게 제시한다.

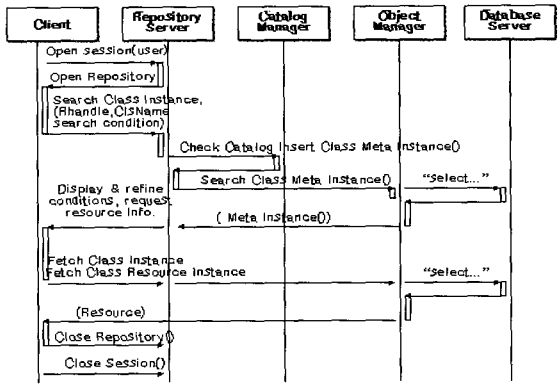
나. 브라우징에 의한 경로 탐험 검색

브라우징에 의한 경로 탐험 검색은 저장소에 저장된 부품의 계층구조를 사용자에게 표시하여 주고, 사용자가 그 계층 구조를 찾아다니면서 자신이 원하는 부품을 찾아내는 방식이다. 클래스 저장소는 클라이언트의 요청에 대해 부품의 계층구조를 클라이언트에 전달하여 주고, 클라이언트에서는 전달받은 구조를 사용자에게 보여준다.

다. 검색과정

클래스 정보를 검색하는 과정은 클래스 정보 재사용 과정에서 매우 중요한 단계인 클래스 정보 선택 단계를 위해 필수적인 기능이다. 강력하고 편리한 탐색 기능은 클래스 정보 저장소의 효용성을 결정짓는 주요 요인이기도 하다.

본 논문에서는 검색 기능에 있어서 패싯으로 사용한 키워드 검색과 해당 클래스 정보의 추출을 위한 기능에 중점을 두어 개발하였다. 클래스 정보 검색은 크게 2단계로 나누어 진행된다. 첫 번째 단계는 탐색 단계로서, 여러 클래스 정보 중에서 특정 조건에 맞는 클래스 정보를 검사하는 단계이다. 이 단계에서는 클래스 정보 메타 정보만을 일차적으로 검색한 후 출력한다. 다음 단계는 각 클래스 정보 메타 정보에 대응하는 클래스 정보를 추출하는 단계로서, 필요한 클래스 리소스 정보를 선별적으로 추출하여 원하는 클래스 정보를 선택하는 데 도움을 준다.



(그림 14) 클래스 검색 절차

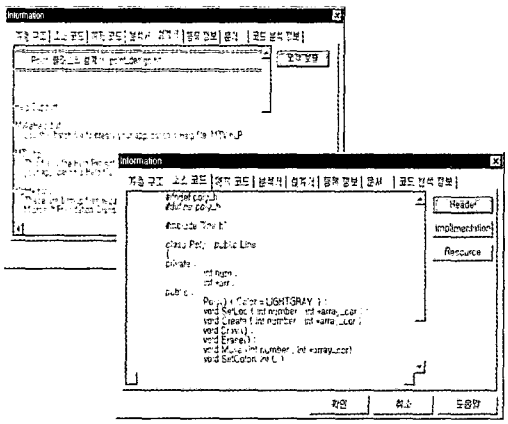
4. 구현 및 적용

저장소 서버 관리 시스템은 PC를 기반으로 한 Windows NT 환경에서 구현했다. 데이터베이스는 Web 환경에서 Objectstore의 PSE를 사용하여 구축했다.

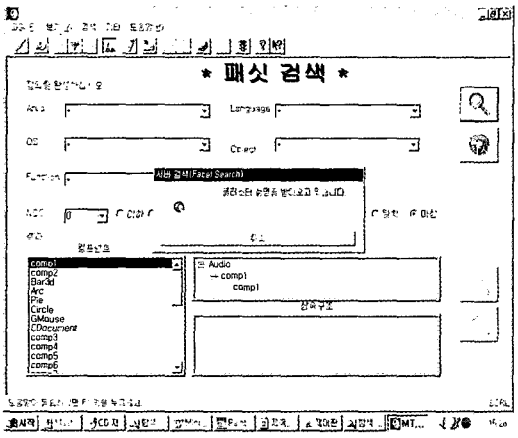
저장소는 사용자로부터 Web을 통해 저장소 서버에 연결을 요청하면 (그림 15)와 같은 초기 화면을 보여준다. 이 화면은 데이터베이스를 갖는 서버와의 연결 과정을 나타내는 대화상자로, 연결 도중, 사용자가 연결을 취소할 수 있다.

(그림 16)은 패싯 검색할 때, 서버에서 패싯 검색을 위한 자료를 전송받는 과정을 나타낸다.

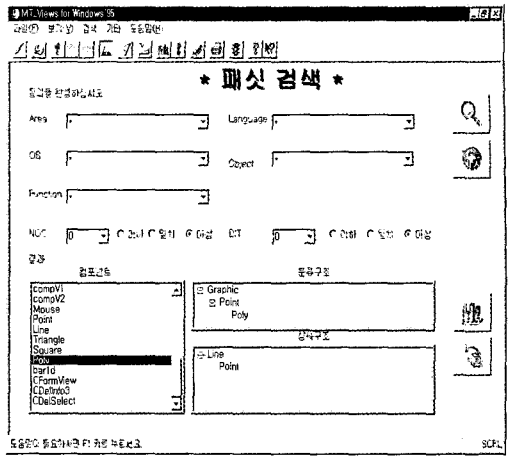
(그림 17)은 패싯으로 분류된 부품들을 검색하여 제시된 후보 부품들 중 'Poly'에 대해 분류 구조와 상속 구조를 기본 정보로 제공함을 나타낸 것이다. (그림 18)은 검색중인 부품의 이해 정보를 제공하는 화면이다.



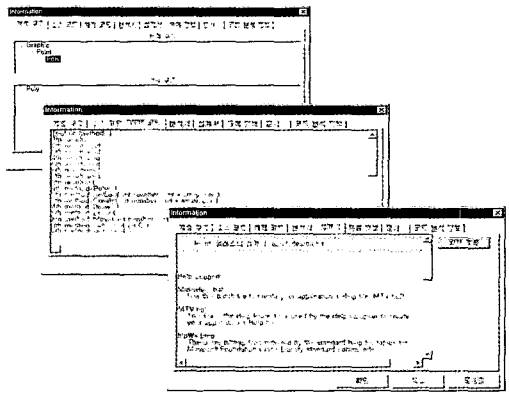
(그림 15) 저장소 서버 초기화면



(그림 16) 패싯 검색과정



(그림 17) 부품 검색을 위한 리스트



(그림 18) 검색중 부품의 이해 정보

5. 결 론

본 논문에서는 클라이언트/서버 모형을 근간으로 외부의 재사용 시스템인 MT-View와, 기존의 정보를 재구성하여 재사용 부품을 생성하는 재구성기, 그리고 재사용 부품을 저장하는 저장소 모듈들로 구성되었다. 저장소에 저장된 재사용 부품을 사용하는 클라이언트는 Web을 통해 저장소 서버에 접근하여 필요한 부품을 쉽게 검색하고 자신의 컴퓨터로 전송받아 새롭게 개발하고자 하는 응용 프로그램에 적용할 수 있다.

향후 연구 과제로, 저장소 서버의 기본 기능을

확장하고 필요로 하는 부품을 검색하는 다양한 검색 기법에 대한 연구와, 최근 관심을 끌고 있는 컴퍼넌트 기반 소프트웨어 개발에 대한 연구가 필요하다.

참고문헌

[1] 윤영태, "분산 객체의 미래", 월간 경영과 컴퓨터, http://www.inhub.co.kr

[2] Mikio Aoyma, "New Age of Software Development : New Component-Based Software Engineering Changes the Way of Software Development", 1998 International Workshop on Component-Based Software, ICSE, pp 124-128, 2003.

[3] Mike P. Papazoglou, "Agent-Oriented Technology in support of e-business", Communication of the ACM, pp 250-254, April, 2004.

[4] 장영범, "전자상거래와 정보시스템", http://www.inhub.co.kr/, 11.1998.

[5] 차정은, 컴포넌트 기반 개발 프로세스 지원을 위한 컴포넌트 저장소의 설계 및 구현, 대구가톨릭대학교대학원 전산통계학 전공 박사학위 청구논문, February, 2001.

저자약력

김 행 곤

1985년 중앙대학교전자계산학과 (공학사)
1987년 중앙대학교 대학원(공학석사)
1991년 중앙대학교 대학원(공학박사)
1978년~1979년 미 NASA 연구원
1987년~1989년 미 Bell Lab. 연구원
1979년~1983년 공군방공포병학교 교관
2000년~2002년 미 CMU 교환교수
1990년- 현대구가톨릭대학교 교수
관심분야 : CBSE , Web 서버 구현, 소프트웨어 아키텍처, 소프트웨어 재공학 및 역공학
이 메 일 : hangkon@cu.ac.kr