UML기반의 패턴지향 설계정보 저장을 위한 혼합형 저장소 설계

최 한 용† · 송 영 재††

요 약

표준화된 설계를 위해 UML이 이용되고 있으나, 설계된 UML 정보를 재사용 가능하도록 저장할 필요성이 있다. 그리고 기존의 설계 정보를 재사용하기 위해 설계 정보를 표준화하고 분류하여 재사용 해야 하는 문제점을 갖고 있다. 본 논문에서는 이러한 문제점을 해결하기 위한 방안으로 설계 정보를 패턴화하여 표준화된 설계 정보와 비 표준화된 설계 정보의 특성을 함께 표현하여 재사용하기 위한 혼합형 저장소 HYREP(HYbrid REPository)를 설계하였다. 그리고 표준화에 의한 설계 유연성을 잃는 단점을 보완하기 위해, 표준화된 설계 정보는 공유 영역에 비 표준화된 설계 정보는 개인 영역에 저장하도록 설계하였다. 그리고 저장소에 저장된 설계정보를 표현하기 위해 OMG의 UML 1.3을 이용하여 객체와 관계를 표현하였다. 따라서 본 논문에서는 패턴-지향형 설계 도구의 구축을 위한 방안으로 설계 정보의 재사용을 위해 설계 정보를 패턴화 하였으며, 패턴화된 설계 정보의 중복 저장을 피하기 위해 설계 정보를 저장할 때 재사용된 설계 정보는 코드화하여 표준화된 설계 정보와 비 표준화된 설계 정보를 표현할 수 있는 혼합형 저장소를 설계하였다.

The Design of Hybrid Repository to store Pattern-Oriented Design Information based on UML

Han-Yong Choi† · Young-Jae Song††

ABSTRACT

UML is used for standardized design, but it has a problem to reuse classified and standardize design information to reuse existing design information. In this paper, the plan to solve these problems design the HYREP(HYbrid REPository) using patternize design information, and to express together specific property of standardized design information and non standardized design information. It save standardized design information in public area, and non design information in private area. And to express saved design information in repository, UML 1.3 of OMG is used and explained the relationship of the object. Therefore, we designed hybrid repository to express standardized design information and non standardized design information using inheritance concept to remove over adding of patternized design information

1. 서 론

시스템 설계 도구의 발전은 정보통신 산업의 발달과 더불어 소프트웨어의 발전에도 많은 변화를 주고 있다. 그러나 이러한 정보통신 소프트웨어의 개발에 앞선 단계로서 시스템을 정의하고 설계하기 위한 설계 도구가 부족하다. 그리고 정보통신 시스템은 사회적, 기술적 변화에 따라 적응력을 갖고 발전하게 되며, 이에 따라 통신 시스템을 확장 또는 새롭게 설계하게 되

† 준 회 원 : 경희대학교 대학원 전자계산공학과
†† 종신회원 : 경희대학교 전자계산공학과 교수
논문접수 : 2000년 7월 8일, 심사완료 : 2000년 8월 24일

고, 정보저장소를 이용하여 기존의 설계 정보를 재사용하기 위한 많은 연구가 이루어지고 있다[1, 2]. 객체지향 소프트웨어(object oriented software)는 캡슐화된 구현 도구인 클래스들이 계층 관계를 이루어 결합한 것으로 객체 사이에 메시지를 교환함으로써 구성 요소의 독립적인 확장 및 재사용 효과를 얻을 수 있다[3]. 이러한 객체지향 방법론은 재사용 객체의 패턴화에 의해 재사용성을 극대화 할 수 있다[4][5]. 그러나 기존의 객체지향 방법론을 지원하는 많은 도구는 아직 표준화되고 효율적인 재사용을 지원하지 못하고 있다. 그리고 시스템 개발을 위한 설계 부분에서 나타날 수 있는 많은 문제점 중의 하나는 반복적인 설계에 의한 분석-설계에 투자되는 시간, 비용의 낭비 문제 그리고 시스템의 유지보수 문제이다[6, 7].

따라서 본 논문에서는 객체지향 CASE 도구의 재사용성을 극대화하기 위한 방안으로 정보 저장소(repository)에 패턴을 적용시켜 패턴지향형 정보저장소를 설계하였다. 이와같은 패턴지향형 도구는 패턴화된 컴포넌트 즉, 패턴 정보를 이용하기 때문에 시스템 구축 개발 기간을 단축시킬 수 있으며, 추상화된 패턴 정보에 의한 재사용성을 높일 수 있다. 또한 설계 정보를 공유하기 때문에 설계자들은 반복적인 설계를 피할 수 있다[8]. 따라서 본 논문의 패턴지향 정보저장소는 객체지향 이론을 바탕으로 패턴을 표준화된 공유 패턴(public pattern)으로 정의하였고, 정의된 패턴내의 중복 저장을 제거하기 위해 패턴 생성기로 분류하여 저장하였다. 그리고 분류된 패턴은 설계하려는 패턴-객체의 모델로 설정할 수 있으며, 상속(inheritance)과 다형성(polymorphism)을 적용한 재사용 가능한 패턴-객체로 설계하였다. 설계된 객체를 표현하기 위한 방법으로는 UML ver. 1.3[9]에 따라 저장된 패턴을 표현할 수 있도록 객체 설계 도구를 설계하였다. 그리고 상속은 정보 저장소 구축에 필요한 패턴-객체의 중복 저장을 막기 위한 방안이며, 다형성은 설계된 패턴-객체를 이용하여 새로운 패턴을 파생시켜 설계자 고유의 개인 패턴(private pattern)을 구축할 수 있도록 하였다. 따라서 공유 패턴과 개인 패턴의 두 가지 설계 정보로 구성하는 혼합형 정보 저장소(hybrid repository)를 이용하여 표준화된 패턴 정보를 이용한 설계가 가능하며, 표준화된 설계 도메인에 유연성을 제공할 수 있도록 하였다. 따라서 시스템의 구축시 설계 정보를 재사용하기 위한 방안으로 설계 정보를 패턴화 하여 재사용할 수 있는 패턴-객체 재사용 시스템을 위한 HYREP (HYbrid Repository)를 설계하였다.

본 논문의 구성은 2장에서 패턴의 분류와 UML 표현방법을 고찰하였으며, 3장에서는 혼합형 저장소를 설계하였으며, 패턴지향 설계 도구에 적용하였다. 그리고 4장에서는 HYREP의 기능과 다른 시스템을 비교 설명하였으며, 끝으로 5장에서는 결론 및 향후 연구 과제에 대하여 서술하였다.

2. 패턴의 분류와 UML 표현

2.1 재사용 가능한 패턴의 분류

패턴은 객체지향 패러다임에서 객체의 정형화를 위한 방법으로 적용되고 있으며, 프로젝트를 설계하는데 사용되는 기본적인 청사진을 제시해준다. 패턴의 근본 개념은 프로그램 설계에 대한 개념과 일치하며, 구체적인 사례를 일반적인 형태로 추상화시키는 과정을 부가하고, 특히 프로그램 디자인의 일반 형태는 변화 가능한 코드와 변화 가능성이 없는 코드를 분리하는 것이다[5]. 결국 변화성을 고려한 모듈화된 디자인을 구축하는 것은 유지 보수에 용이한 디자인을 할 수 있도록 하는 주된 요인이다. 따라서 패턴은 특수한 경우에만 적용되는 것이 아니며 개발에 필요한 세부 사항까지 구체적인 방법을 일일이 해결안을 제시하는 것이 아니고, 상위 설계 단계에서 적용될 수 있는 개념이다. 즉, 프로젝트 내의 공통된 설계 문제에 대한 공통된 해결책이고, 컴포넌트를 조립 라인 방식으로 만들 수 있다. 그리고 프로젝트를 정해진 개발 기한 내에 완성하고, 비용을 줄이고, 필요한 코딩 양을 크게 줄일 수 있게 도와준다. 컴포넌트 작성 방법을 알려주는 패턴이 없으면 개체를 한 개의 클래스로 만들 것인지 아니면 두 개 이상의 클래스로 구성된 클래스 계층으로 만들 것인지 생각해야 한다[10]. 그러나 컴포넌트를 어떻게 만들어야 하는지 정확히 알려주는 잘 정의된 패턴이 있으면 이러한 문제에 대한 해결책을 제시해 준다[11, 12].

본 논문에서는 설계의 경험이 많으며 패턴 분야에 관한 분류를 가장 잘 분류된 Erich Gamma의 분류 방식을 적용하였다[5]. Gamma의 대표적인 설계 패턴 분류인 설계 패턴 카탈로그는 기본 패턴(Fundamental Design Patterns), 생성 패턴(Creational Patterns), 분류 패턴(Partitioning Patterns), 구조화 패턴(Structural

Patterns), 행위 패턴(Behavioral Patterns)으로 분류되어진다. 따라서 설계상의 문제를 해결하고 재사용 할 수 있는 패턴을 Gamma의 방식을 분류하여 이용하고, 분류된 패턴을 저장하기 위해서 공유 저장소(public repository)와 개인 저장소(private repository)로 구성되는 혼합형 저장소(hybrid repository)를 설계하였다. 그리고 분류된 설계 패턴은 Gamma에 의해 도메인에 비 종속적이며, 정형화된 패턴으로 표현된 것이므로 표준화된 패턴을 저장하게 되는 공유 저장소에 저장하게 된다. 그리고 개인 저장소에는 표준화된 패턴을 바탕으로 설계 도메인에 따라 사용자 고유의 패턴 정보가 저장된다.

2.2 UML 표현

방법론은 생각과 행동을 구조화하는 방법을 명백히 제시한 것을 말한다. 예를 들어 사용자가 하나의 모델을 만들 때, 어떻게, 언제, 무엇을, 왜 라는 모든 방법을 제시하는 것이 방법론인 반면에 이러한 모델을 단지 표현하는 것이 모델링 언어(modeling language)이다[13]. UML(Unified Modeling Language)은 객체 지향 분석(analysis)과 설계(design)를 위한 모델링 언어이다. UML은 Booch, Rumbaugh, Jacobson등의 객체 지향 방법을 통합한 표준화된 모델링 개념이다. 또한 객체 기술에 관한 국제 표준화 기구인 OMG(Object Management Group)에서 UML을 모델링 언어의 표준안으로 인정하였다. 따라서 본 논문에서는 패턴-객체를 모델링하기 위한 언어로 OMG의 UML 1.3[9]을 적용하였고, UML의 다이어그램 표기 방법 중 클래스 다이어그램을 이용하여 재사용 가능한 패턴-객체를 표현하였다. 클래스 다이어그램(class diagram)은 여러 가지 객체들의 타입(type), 즉 클래스들을 표현하고 각 클래스들의 정적인 관계를 표현한다. 이러한 정적인 요소를 이용하여 시스템의 생명 주기와 수명을 같이하며 하나의 시스템은 여러 개의 클래스 다이어그램으로 표현이 가능하도록 하였다. 시스템 내의 객체들은 상호간 관계를 갖게되는데, 연관관계(association relationship)는 객체간에 이루어지는 모든 일반적인 관계를 포함한다. 따라서 설계 초기에 클래스 사이의 관계는 "연관관계"로 표현되며, 개발이 진행됨에 따라 일반화(generalization), 집단화(aggregation), 의존성(dependency)으로 구체화되며 클래스간의 관계를 표현하는 표기법과 의미를 알 수 있도록 하였다.

3. 패턴지향 설계정보를 위한 혼합형 저장소 설계

3.1 패턴 객체의 투플 구조 설계

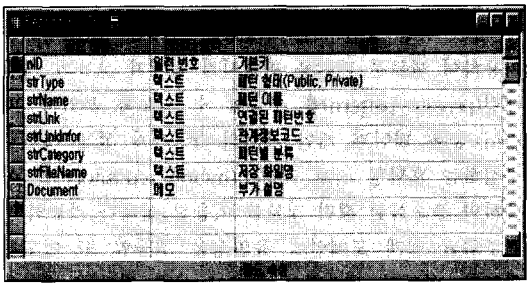
본 논문에서 HYREP은 혼합형 저장소의 구조를 이용하고 있으므로 저장소에 적용될 패턴은 표준화된 공유 패턴(public pattern)과 비 표준화된 개인 패턴(private pattern)을 모두 수용할 수 있는 구조를 갖도록 설계하였다. 따라서 패턴 객체를 표현하기 위한 투플은 저장된 패턴을 서로 구분할 수 있도록 하였고, 패턴 타입을 표현할 수 있도록 하였다. 그리고 복합 패턴은 재사용한 패턴의 연결 정보와 패턴들의 관계 정보로 표현할 수 있도록 설계하였다.

이와 같이 패턴 객체의 구성 요건을 만족하도록 8개의 속성으로 구성하였으며, 설계한 데이터베이스 내에서 각 속성의 의미는 <표 1>과 같다.

〈표 1〉 설계된 개체의 속성

속 성	의 미
nID	패턴을 서로 구분하는 투플의 키값
strType	패턴의 유형 코드
strName	패턴의 이름
strLink	연결된 패턴 번호
strLinkInfor	연결된 패턴간의 관계정보 코드
strCategory	패턴별 분류
strFileName	저장 화일명
Document	부가 설명

<표 1>에 설계된 데이터베이스의 속성들은 (그림 1)과 같이 관계 스키마(relation schema)로 정의하였으며, 정의된 속성에서 "strLinkInfor"는 표준화된 객체를 이용하여 복합된 객체를 생성한 경우 연결된 객체들간



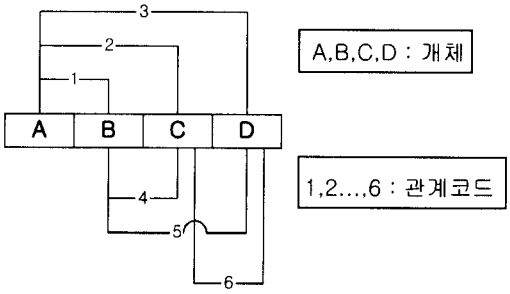
(그림 1) 객체저장 데이터베이스

의 연결 관계를 코드화하여 표현하도록 하였다. 그리고 객체들 사이의 연결정보는 최소한의 정보를 이용하여 저장하기 위해서 객체들 사이의 관계를 부호화하여 저장하였으며, 이때 이용하는 연결 정보의 부호화에 사용된 코드 구조는 <표 2>와 같이 정의하였다.

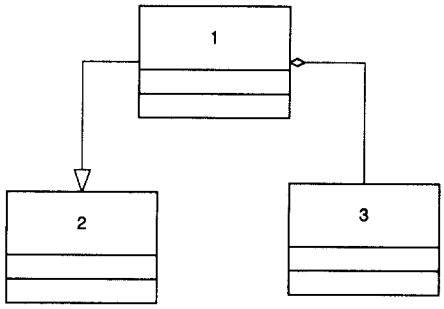
〈표 2〉 연결정보 코드 구조

코 드	의 미
1	Dependency
2	Generalization
3	Association
4	Aggregation
5	Refinement

이때 단일 개체의 "nID"간의 연결 정보는 순차적으로 하나의 코드를 갖게된다. 각각의 코드는 객체와 객체간의 관계를 설정하기 위해 다섯 가지 관계로 코드를 정의하였다.



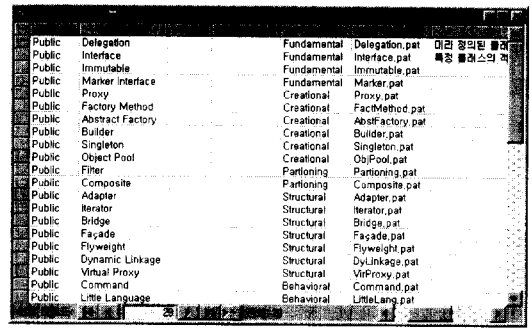
(그림 2) 개체와 연결정보 코드 표기



(그림 3) +2-4+0 코드의 관계 구조

(그림 2)에서와 같이 A, B, C, D 로 구성된 복합 객체인 경우, 이들의 순서는 순차적으로 A와 연결 가능한 조합인 B, C, D이고, 다음으로 B와 연결 가능한 조합은 C, D, 그리고 C와 연결 가능한 조합은 D이다. 그리고 그들의 방향에 따라 순방향은 +로 표기하고, 역방향은 -로 표기하도록 하였다. 예를 들어 "strLink" 값이 "123"이라면 1, 2, 3번의 패턴을 이용하였다는 정보를 나타내며, 이때 "strLinkInfor"값이 "+2-4+0"의 코드 값을 갖게되면 (그림 3)과 같이 1-2 객체 관계는 일반화(generalization), 1-3 객체 관계는 집단화(aggregation), 2-3 객체 관계는 아무런 관계가 없음을 코드화하여 표현한 것이다.

Gamma의 패턴을 부품화 하기 위해서 <표 1>의 속성을 갖도록 (그림 1)과 같이 설계하고, 패턴정보를 저장소(repository)에 저장하였을 때 (그림 4)와 같은 투플(tuple)구조를 갖게된다. 각 투플은 하나의 패턴 객체를 나타낼 수 있는 구조를 갖고 있도록 설계하였으며, 이때 단일 패턴이 아닌 복합 패턴인 경우 복합된 패턴은 이미 기존에 설계된 패턴 정보를 재사용하여 구성한 패턴 정보이다.

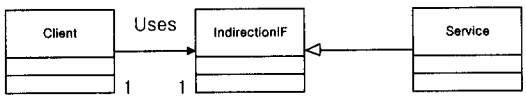


Public	Delegation	Fundamental	Delegation.pat	미리 정의된 클래
Public	Interface	Fundamental	Interface.pat	특정 클래스의 객
Public	Immutable	Fundamental	Immutable.pat	
Public	Marker Interface	Fundamental	Marker.pat	
Public	Proxy	Creational	Proxy.pat	
Public	Factory Method	Creational	FactMethod.pat	
Public	Abstract Factory	Creational	AbstFactory.pat	
Public	Builder	Creational	Builder.pat	
Public	Singleton	Creational	Singleton.pat	
Public	Object Pool	Creational	ObjPool.pat	
Public	Filter	Partioning	Partioning.pat	
Public	Composite	Partioning	Composite.pat	
Public	Adapter	Structural	Adapter.pat	
Public	Iterator	Structural	Iterator.pat	
Public	Bridge	Structural	Bridge.pat	
Public	Façade	Structural	Façade.pat	
Public	Flyweight	Structural	Flyweight.pat	
Public	Dynamic Linkage	Structural	DyLinkage.pat	
Public	Virtual Proxy	Structural	VirProxy.pat	
Public	Command	Behavioral	Command.pat	
Public	Little Language	Behavioral	LittleLang.pat	

(그림 4) 설계한 데이터베이스의 투플 구조

따라서 본 논문에서 복합 패턴의 표기는 적용된 패턴을 나타내기 위하여 코드화된 정보를 이용하였고, 패턴 구성에 사용된 연결 정보를 분석하여 저장소에 저장되어있는 패턴을 재사용하도록 하였다. 따라서 HYREP은 중복 저장을 피하여 복합 패턴을 구성할 수 있도록 설계되었고, 이를 조합하고 분해하는 기능은 코드화 되도록 설계하였다. 각 투플은 (그림 1)에 정의된 제한적 속성을 갖게 되며 도메인 종속성을 갖게된다. 두 번째 투플인 인터페이스 패턴(interface pattern)은 자바 내부에서 키워드를 이용해 자체적으로 제공하는 기법이며, 특정 클래스의 기능을 액세스하기 위해

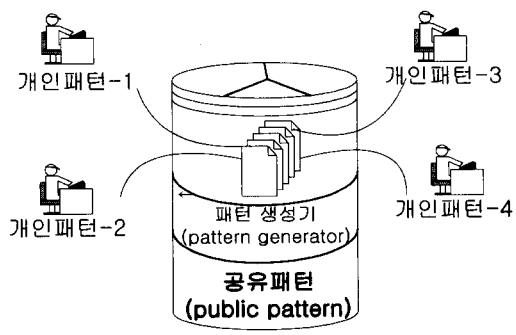
인터페이스 클래스에 대한 인스턴스를 중간 매개체처럼 이용하는 방식이다. 인터페이스 패턴에 대한 UML 다이어그램은 (그림 5)와 같이 클래스간의 의존성 관계를 최소화하기 위한 인터페이스 패턴의 기본 유형을 클래스 다이어그램으로 표현하였다.



(그림 5) 인터페이스 패턴 유형

3.2 HYREP의 구조

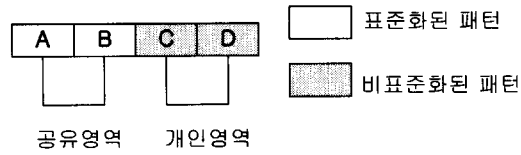
기존의 CASE 도구에서 사용하는 저장소에서는 데이터 모델로 E-R모델 또는 OO모델 설계 정보를 저장하고, 이때 저장된 설계정보의 컴포넌트 형식은 표준화되었으나 설계정보 자체는 표준화하지 않았다[14, 15]. 따라서 본 논문에서는 정보저장소에 저장하게 될 설계정보를 표준화하여 저장함으로써, 설계정보 재사용을 용이하도록 하기위해 HYREP의 공유영역에 표준화된 패턴 설계 정보를 저장하였으며, 저장소에 설계의 중복성을 줄이기 위하여 중복되는 설계상의 정보는 상속 구조를 이용하여 저장하였다. 그리고 표준화된 설계의 제한성을 극복하기 위하여 저장소의 영역을 공유 저장소와 개인 저장소의 혼합형 구조를 갖는 HYREP (HYbrid REPository)을 설계하였다. 따라서 시스템은 설계서를 재사용하기 위해 표준화된 소프트웨어의 패턴을 추출하여 패턴 저장소에 저장하며, 이때 저장되는 패턴들간의 중복 데이터는 객체의 상속 매커니즘을 이용하여 저장하였다.



(그림 6) HYREP 구조

패턴이 저장될 때는 (그림 6)과 같이 표준화된 패턴 정보가 저장되는 공유 패턴 영역과 특정 도메인을 모델링한 개인 패턴 영역으로 나누어 저장되도록 하였다. 공유 패턴과 개인 패턴으로 분리하여 혼합형 구조를 이용할 수 있도록 설계하였기 때문에, 표준화된 설계패턴 영역인 공유영역은 설계패턴으로 표준화될 수 있으며 표준객체를 이용하여 설계된 설계 패턴은 공유 패턴 영역에 저장하도록 하였다.

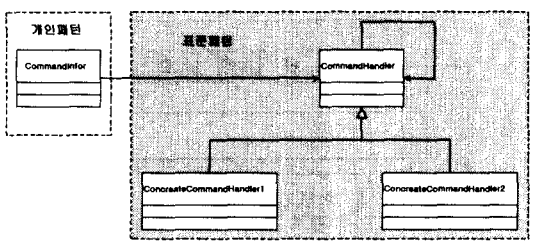
그러므로 공유 영역에 저장된 설계 정보를 이용하여 사용자는 언제든지 최적의 방법으로 시스템을 설계할 수 있다. 이때 공유 패턴 정보 외에 추가적으로 사용자의 설계 도메인상의 특성에 의해 나타나는 설계 정보는 개인 패턴 영역에 저장되도록 하였다. 이러한 설계 정보 저장 구조의 이중구조는 설계 정보의 표준화를 유지할 수 있으며, 설계 도메인의 특성에 의한 유연성을 갖는다. (그림 7)의 복합 객체는 혼합형 객체로써 표준화된 패턴과 비 표준화된 사용자 도메인 상의 설계 정보가 포함된 객체이다. 이때 표준화된 패턴 정보는 공유 영역에 저장되어 있고, 비 표준화된 패턴은 개인 영역에 저장하도록 하였다. 그리고 이때 A, B, C, D의 연결 정보는 (그림 2)에 표현된 방법을 이용하여 코드화하여 저장하도록 설계하였다.



(그림 7) 혼합형 객체표현

(그림 8)은 표준 패턴으로 공유 영역에 저장되어 있는 설계 정보를 이용하여 사용자의 개인 패턴 정보와 혼합하여 설계된 상태를 보여주고 있다. 이때 공유 정보와 개인 정보는 분리되어 표현되도록 설계하였다. 그리고 본 논문에서 재사용 컴포넌트로 저장된 패턴은 소프트웨어 설계시 객체의 요청이 있을 경우 저장된 패턴을 검색하여 사용자가 원하는 패턴-객체를 재사용할 수 있는데, 이때 저장된 패턴의 검색은 유사도 측정에 의한 개선된 Spreading Activation 방법[16]을 이용하게 되므로 사용자가 검색할 내용의 유사 값만을 알고있어도 검색이 가능하게 되는데, 활성 값 계산을 위한 순환을 반복하는 것은 정확한 활성 값으로 인한

유사 부품을 검색하기 위한 목적이다. 따라서 Spreading Activation 방법은 저장소에 재사용 정보를 구축한 경우 이를 재사용하기 위한 검색에서 사용자가 원하는 재사용 컴포넌트를 정확히 모르는 경우 사용자가 검색하려는 컴포넌트와 유사한 여러 개의 패턴 정보를 검색할 수 있으며, 재사용 가능한 표준 컴포넌트를 제공하는 공유(public) 저장소가 있다. 그리고 본 논문에서는 표준화되어 저장된 패턴을 바탕으로 설계자의 고유한 설계 목적을 이룰 수 있는 고유 패턴을 구성할 수 있도록 설계자만의 개인(private) 저장소를 구축할 수 있도록 HYREP을 설계하였다.



(그림 8) 혼합객체의 UML 표현

3.3 패턴지향 설계 도구에 HYREP 적용

본 논문에서 제안하는 HYREP(HYbrid REPository)은 설계 중 특별한 문제에 대한 알맞은 패턴(pattern)의 선택이 아니라 설계와 프로그램의 생성, 재조직화(reorganization)의 과정에서 패턴을 적용하게되는 혼합형 구조의 저장소이다.

패턴지향 설계 도구에 HYREP을 적용하기 위해 제안한 (그림 9)의 저장소와 UML표현을 위한 뷰 모델에서 다음의 4가지 기능에 의해 동작하도록 설계하였다.

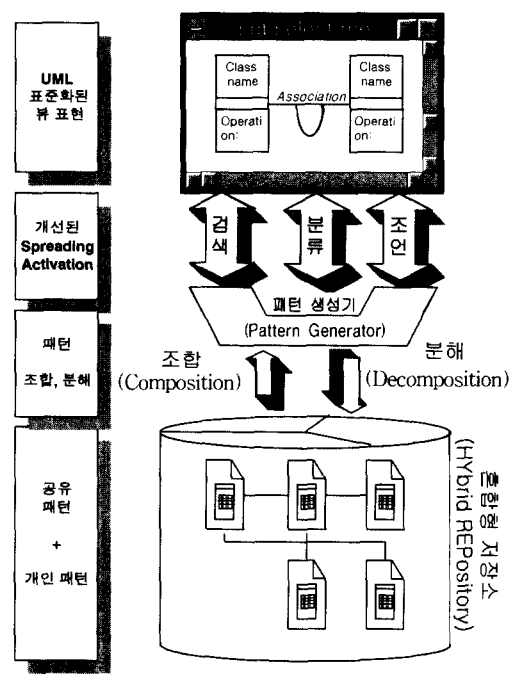
첫째, 상속성에 따른 저장 구축 방법을 따르고 있으므로 컴포넌트를 조합/분해하여 저장하고 이용할 수 있는 기능을 갖는다.

둘째, 패턴의 확장된 모임으로부터 얻어진 패턴의 새로운 인스턴스(instance)를 위한 프로그램 요소(class, hierarchies 등)를 생성한다.

셋째, 통합된 패턴(integrating pattern)은 프로그램 요소를 패턴의 역할에 연결함으로써 저장소를 구축한다.

넷째, 복합 구조의 객체 저장소에 표준화된 영역과 비 표준화된 영역을 분리하여 저장한다.

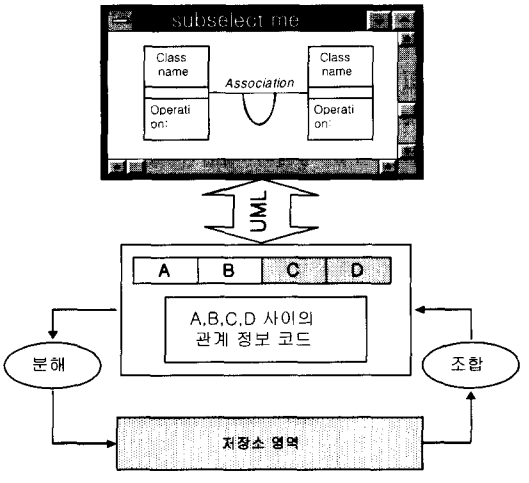
기존 설계서에서의 재조직화 동작(reorganization operation)에 의한 설계서의 패턴 추출은 역공학(reverse engineering)에 사용될 수 있다. 즉, 프로그램에서 패턴 발생의 문서화와 프로그램의 수정은 패턴의 표준화된 구조에 더욱 영향을 준다. 패턴 생성기로부터 얻어진 정형화된 패턴은 객체 저장소(object repository)에 분해(decomposition)되어 저장되며 이후 설계 패턴의 재사용(reuse)시에는 결합(composition) 과정에서 필요한 패턴을 재조직화(reorganization)한다. 분해(decomposition)에 의해 저장될 때 저장소의 정보와 중첩을 피하기 위하여 상속 관계로 정보를 저장한다. 사용 목적에 따라 객체의 결합에 의해 패턴을 조직화하는 경우 저장되어 있는 패턴을 합성한 복합 패턴을 지원하게 된다. 그리고 객체 저장소는 시스템의 정형화된 표준 패턴을 구성한 공유 영역과 개인적 특성화에 의한 개인 영역의 복합 구조를 갖으며, 개인 영역의 패턴은 공유 영역의 정규화된 패턴을 다형성에 의해 확장할 수 있는 특징을 제공하게 된다. 본 논문에서 HYREP과 UML 뷰 모델사이의 대표적인 기능을 다음과 같이 정의하였다.



(그림 9) 저장소와 뷰 모델의 구조

3.3.1 표준화된 뷰(view) 표현

설계 정보를 공유하거나 재사용하기 위해서는 표준화된 표현 방법을 이용하여야한다. 따라서 본 논문에서는 설계에 사용하는 클래스 다이어그램의 표기법과 설계되어 저장되어있는 패턴 정보를 표현하기 위한 표준화된 뷰 표현은 UML의 클래스 다이어그램을 이용하였다. (그림 10)과 같이 UML에 의해 설계된 뷰 정보는 객체와 관계로 표현되며, 각각은 공유 영역과 개인 영역에 저장할 부분으로 분리되어 저장하였다



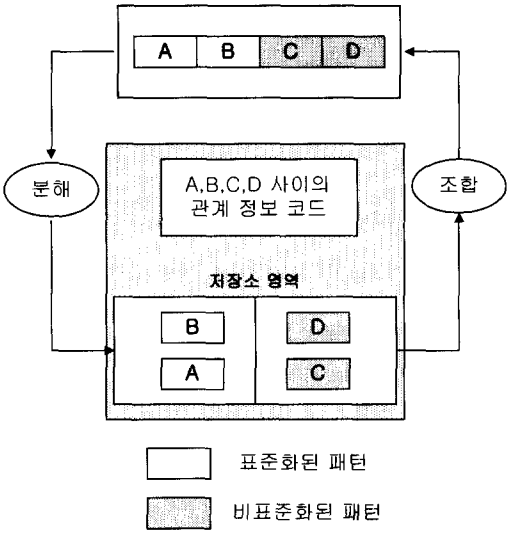
(그림 10) 저장소 정보 표현을 위한 뷰 모델

(그림 10) 에서 A와 B는 UML 정보 중 표준화된 설계정보를 저장한 것이며, C와 D는 UML정보 중 비 표준화된 객체를 저장한 것이다. A, B, C, D 패턴은 각각 분해와 조합을 통하여 UML로 표현이 가능하도록 하였다. 그리고 뷰(View)는 저장된 패턴 정보의 UML 표현을 위하여 객체 정보와 관계 정보를 이용한다.

3.3.2 패턴의 조합(composition)과 분해(decomposition)

본 논문에서는 표준화된 설계 정보와 비 표준화된 설계 정보를 분리하여 저장하는 혼합형 저장소를 구축하였다. 설계된 정보를 저장소에 저장하기 위해서는 우선적으로 공유 영역의 설계 정보와 개인 영역의 설계 정보를 분해(composition) 해야한다. 그리고 분리된 각각의 정보는 공유 영역과 개인 영역에 저장하기 위해 분해된 컴포넌트는 Public과 Private의 코드 값을 갖게 된다. (그림 11)과 같이 분해된 컴포넌트 사이의 관계는 각각의 관계를 부호화된 코드를 이용하여 저장하도록 설계하였다. 그리고 조합(composition)의 기능은 분해(decomposition)에 의해 조각난 컴포넌트를 다시 조립하여 설계 정보를 표현하도록 한다.

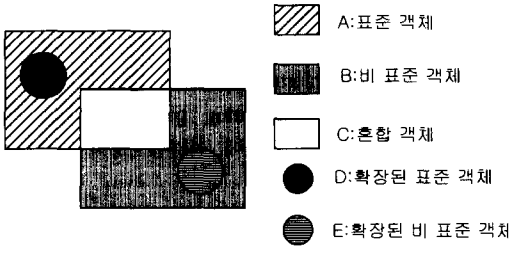
이때 조합은 투플 내의 속성 중에서 설계에 적용된 패턴의 번호들을 얻게되며, 각 패턴의 조합 규칙을 위한 관계를 코드 정보로 얻게된다. 그러면 조합 기능은 두 가지 정보를 이용하여 필요한 객체를 수집하고 수집된 객체를 관계 정보 코드를 해석해가며 연결하도록 하였다.



(그림 11) 패턴의 분해와 조합

3.3.3 다양한 패턴 표현

설계 정보는 표준화된 공유 패턴과 비 표준화된 개인 패턴으로 분류하였다. 그리고 두 영역을 세분화하



(그림 12) 객체의 종류 구분

면 설계정보는 (그림 12)에서 표현되는 것과 같이 다섯 가지 객체 영역으로 구분할 수 있다.

순수하게 표준화된 객체(A)와 비 표준화된 객체(B)로 이루어진 영역과 순수한 표준 객체의 집합으로 구성된 혼합 객체(D), 순수한 비표준 객체의 집합으로 구성된 혼합 객체(E), 마지막으로 표준 객체와 비표준 객체가 혼합되어있는 혼합 객체(C)로 구분하였다.

$$C = \{x \mid x \in A \bigwedge x \in B\}$$
$$D = \{x \mid x \in A \bigwedge x \notin B \mid X \subset A\}$$
$$E = \{x \mid x \notin A \bigwedge x \in B \mid X \subset B\}$$

따라서 시스템 설계시 표준 객체의 영역은 재사용 영역을 나타내는 것이며, 비표준 객체 부분은 시스템의 특성화에 의해 새로운 설계정보가 구축되었음을 나타낸다. 그러므로 각 객체의 구분에 의해 시스템에서 제공하는 객체를 확장하여 재사용할 수 있는 객체를 생성한 객체확장 비율과 전체 시스템 설계 중 표준화된 객체를 이용한 표준화된 설계표준 비율, 그리고 기존의 객체를 이용한 객체 재사용 비율을 계산할 수 있다.

- 객체확장 비율 $= \dfrac{D+E}{A+D+B+E}$
- 설계표준 비율 $= \dfrac{A+D+(C-C')}{\text{전체 객체}}$
  (c':혼합객체의 비표준)
- 패턴 재사용 비율 $= \dfrac{A+C+E}{\text{전체 객체}}$

4. 비교 분석

기존의 CASE 도구는 자료의 통합적인 측면에서 자료가 중복되는 것을 피하고 도구들간의 자료 공유가 용이하도록 정보 저장소(repository)를 구축하여 프로세서 관련 산출물을 저장하고 관리한다. 본 논문에서 제안하는 혼합형 저장소와 기존의 CASE 도구에서 사용하고 있는 저장소를 <표 3>에서 비교하였다. 기존의 CASE 도구에서는 E-R 모델이나 객체 모델을 데이터 모델로 설정하였으나 본 논문에서는 표준화된 설계 정보를 표현하는 Gamma의 설계 패턴을 데이터 모델로 설정하였다. 그리고 저장소의 기능으로 중복 데이터의 제거를 위해 상속의 개념을 이용하여 컴포넌트의 저장시 재사용된 설계 정보는 상속되어 저장되므로 재사용된 설계 정보는 중복되어 저장되지 않도록 하였다.

<표 3>에서 기존의 정보저장소는 E-R모델이나 객체 모델을 사용하였지만 본 연구에서 제안한 HYREP은 Gamma의 패턴모델을 사용하였다. 또한 기존의 저장소는 단일구조인 반면 본 연구는 개인패턴과 공유패턴으로 구성된 혼합구조를 이루고 있다. 또한 데이터의 표준화는 같지만 본 연구는 설계정보를 표준화함으로써 설계시 나타날 수 있는 설계 경험 부족으로 인한 어려움을 해결할 수 있다. 설계 표준화는 기존의 저장소에 저장되는 컴포넌트 형식이 E-R 모델 또는 객체 모델 데이터가 표준화된 것이며, 설계 정보 자체가 표준화 된 것은 아니다.

〈표 3〉 기존 시스템과 비교

	Adele	ALF	Matisee	Rose	HYREP
데이터 모델	E-R 모델	E-R 모델	객체 모델	객체 모델	설계 모델
중복성 제 거	○	○	○	○	○
저장소 구 조	단일구조	단일구조	단일구조	단일구조	혼합구조
UML 지 원	×	×	×	○	○
데이터 표준화	○	○	○	○	○
설 계 표준화	×	×	×	×	○
설 계 가독성	○	○	○	○	○

본 논문에서는 이러한 단점을 보완하여 UML을 이용하여 데이터의 표현법을 표준화하였으며, UML을 이용한 설계정보가 표준화된 방식으로 설계되도록 하기 위해 저장 데이터 모델을 패턴으로 설정하였다. 따라서 설계상의 문제점을 해결하고 표준화된 객체를 재사용하여 설계할 경우 설계 당시부터 표준화된 설계 정보로 구성되어 저장되도록 하였다. 또한 표준화된 패턴을 모델로 하여 저장된 컴포넌트는 항상 표준화되어 저장되도록 되어있기 때문에 설계정보의 가독성이 높아지게 되므로 설계자의 변화에 적응력이 높다. 그리고 표준화가 갖고있는 단점은 설계의 유연성을 잃게 하므로, 표준화된 설계 정보를 재사용하는 설계자는 설계자의 의도나 설계 도메인의 특성에 의해 설계할 수 있도록 저장소를 표준화 영역과 비 표준화 영역을 혼합하여 사용할 수 있도록 두 가지 특성을 만족시키는 혼합형 저장소 방식을 설계하였다.

따라서 기존의 저장소를 구성하고 있는 컴포넌트의 단위를 패턴화하여 설계자들간의 표준화된 정보를 공유할 수 있도록 하였으며, 중복 데이터의 제거를 위해 상속 매커니즘을 이용하여 저장소의 가장 기본적인 목적인 중복 데이터의 제거와 정보의 공유를 만족시킬 수 있도록 하였고, 혼합형 저장소를 구축하여 설계의 표준화와 설계 도메인에 따라 적응력을 갖도록 저장소를 설계하였다.

5. 결 론

본 논문은 CASE 도구의 정보 저장소에 저장되는 컴포넌트를 표준화된 설계 정보를 제공하는 표준 패턴과 비 표준화된 설계 패턴을 적용하여 저장할 수 있는 **혼합형** 저장소를 설계하였다. 그리고 저장소 구축 방법을 복합 구조로 분리하여 시스템의 유연성을 높였으며, 상속 개념을 이용한 저장소의 저장 방법으로 컴포넌트의 중복 저장을 제거하고, 다형성에 의한 설계 컴포넌트간의 복합 컴포넌트를 제공할 수 있도록 하였다. 따라서 CASE 도구에 패턴을 적용하여 설계 정보를 표준화하고 이를 재사용 함으로써 재사용율과 확장성을 높일 수 있도록 HYREP(HYbrid REPository)을 설계하였다. 또한 패턴을 이용하여 과거의 설계 개념을 재사용 하므로 유지 보수를 위해 패턴을 수정할 수 있도록 하였으며, 시스템의 개발에 투자되는 비용의 중복 투자를 제거할 수 있도록 하였고, 개발 기간을 단축시킬 수 있는 기반을 제시하였다.

따라서 본 논문에서 정보통신연구진흥원 정보통신 우수시범학교 지원 사업에 의해 표준화된 패턴을 적용한 재사용 시스템의 기반을 제시함으로써 급변하는 정보 통신 시스템의 설계시 기존의 설계 정보를 재사용할 수 있도록 하였다. 향후의 연구 계획으로는 인터넷 기반의 설계 정보 공유를 위해 설계 정보를 XML로 저장할 수 있는 변환기를 설계하여 TCP/IP를 기반으로 설계 정보를 공유하도록 해야 한다.

참 고 문 헌

[1] J. M. Sagawa, "Repository Manager Technology," IBM System Journal , Vol.29, No.2, 1990

[2] Theo Harder, Wolfgang Mahnke, Norbert Ritter, Hans-Peter Steiert, "Generating Versioning Facilities for a Design-Data Repository Supporting Cooperative Applications," IJCIS 2000, pp.117-146.

[3] Orfali, Harkey, Edward, "The Essential Distributed Objects Survival Guide," WILEY, 1996.

[4] Dirk Riehle, "Composite Design patterns," OOPSLA '97, pp.218-228.

[5] E. Gamma, R. Helm, R. Johnson, and J.Vlissides, "Design Pattern : Elements of Reusable Object-Oriented Software," Addison-Wesley, 1995.

[6] W. Kozaczynski, E. Liongosari, J. Ning, and A. Olafs son, "Architecture Specification Support for Component Integration," IWCASE '95, pp.30-39, 1995.

[7] Matthias Jarke, "Strategies for Integrating CASE Environments," IEEE Software, 1992, pp.54-61.

[8] Rudolf K. Keller, Richard Schauer, "Design Components : Towards Software Composition at the Design Level," ICSE '98, pp.282-291.

[9] HTTP://WWW.OMG.ORG.

[10] Frank Buschman, Regine Meunier, Hans Rohnet, Peter Sommerland and Michael Stal, "Pattern Oriented Software Architecture, A Pattern System," Draft, 1995.

[11] 김치수 "설계패턴을 이용한 객체지향 방법론에 관 한 연구", 한국정보처리학회 논문지, 제6권 제6호, pp.1556-1562, 1999.

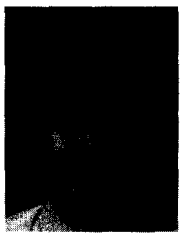
[12] Mark Grand, "Patterns in Java," WILEY, 1998

[13] Grady Booch, James Rumbaugh, Iver Jacobson, "The Ufified Modeling Language User Guide," Addison-Wesley, 1998.

[14] M.wein, S.MacKay, W. Gentleman, "Evolution is Essential for Software Tool Development," IW-CASE '95, pp.196-205.

[15] T. D. Han, Sandeep Purao, Veda C. Storey, "A Methodology for Building a Repository of Object-Oriented Design Fragments," ER 1999, pp.203-217.

[16] 한정수, 송영재, "개선된 SARM을 이용한 객체지 향 부품 재사용 시스템", 정보처리학회 논문지, 제7권 제4호, pp.1092-1102, 2000.



최 한 용

e-mail : hychoi@case.khu.ac.kr

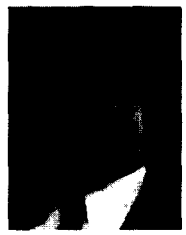
1994년 경희대학교 전자계산공학과 (공학사)

1998년 경희대학교 전자계산공학과 (공학석사)

2000년 경희대학교 전자계산공학과 박사과정 수료

1998년~현재 경희대학교 전자계산공학과 박사과정

관심분야 : 소프트웨어공학, CASE 도구, S/W재사용



송 영 재

e-mail : yjsong@khu.ac.kr

1969년 인하대학교 전기공학과 (공학사)

1976년 일본 Keio Univ. 전산학과 (공학석사)

1979년 명지대학교 대학원 졸업 (공학박사)

1971년~1973년 일본 Toyo Seiko 연구원

1982년~1983년 미국 Univ. of Maryland 전산학과 연구교수

1989년~1990년 일본 Keio Univ. 전산학과 객원교수

1984년~1989년 경희대학교 전자계산소장

1993년~1995년 경희대학교 교무처장

1996년~1998년 경희대학교 공과대학장

1976년~현재 경희대학교 전자계산공학과 교수

1998년~현재 경희대학교 기획조정실장

관심분야 : 소프트웨어공학, OOP/S, CASE 도구, S/W 재사용