설계 패턴 재사용 라이브러리 구현
(Implementation of Library for Design Patterns Reuse)

김 행 곤 † 김 지 영 ††
(Haeng Kon Kim) (Ji Young Kim)

요 약 다양한 플랫폼 상에서 응용 시스템에 대한 개발, 배포, 운영의 고생산성과 고품질을 얻기 위해서는 소프트웨어 구성 요소들의 체계적인 재사용 접근과 자동화된 도구의 지원이 요구된다. 함수나 클래스 라이브러리에서 설계 패턴과 프레임워크, 그리고 최근의 상업적인 비즈니스 컴포넌트에 이르기까지 여러 재사용 방법이 제시되었으나 기대만큼의 확실한 효과를 얻을 수 없었다. 설계 패턴은 설계 경험에 대한 캡슐화된 빌딩 블럭으로 개발 응용의 표준화된 아키텍쳐 제공을 통해 응용 도메인의 한정성과 클래스 수준의 재사용의 복잡성을 극복할 수 있다. 응용 개발의 표준 아키텍쳐로서의 웹을 통한 재사용 요소들의 공유는 여러 개발자들에 의한 다양한 도메인 요소로서의 전개와 동적이며 실시간적인 라이브러리 관리가 가능하다. 본 논문에서는 웹 환경 하에서 자동화된 설계 패턴 재사용 환경 구축을 목적으로 한다. 그러므로 For Reuse 관점에서 서버 상에 각 도메인별 패턴 라이브러리를 구축하며 With Reuse 관점에서 패턴 검색, 이해, 획득 그리고 재구조화를 통한 응용으로의 전개를 위한 재사용 지원기를 구축한다. 본 시스템은 패턴의 공유에 의한 유사 도메인 응용의 표준화를 유도하고 사용자의 패턴 재정의에 의한 자생적으로 확장 가능한 패턴 라이브러리 제공이 가능하다. 또한 이를 위해 도메인 분석을 통한 패턴의 행위와 의도를 기준으로 설계 패턴들을 분류, 카탈로깅하여 재사용 라이브러리를 제시한다.

Abstract To gain high productivity and quality in development, delivery and management of applications across various platforms, it is necessary to consider systematic approach and automatical reuse support tool for reuse elements of software. There are many reuse methodologies, such as function, class library, design pattern, framework and business components, but we can't acquire the desirable effects. Design patterns are a building blocks encapsulated design experience. They can allow to solve the specific application domain and reuse complexity of implementation. Sharing of reusable elements through Web, as standard architecture of application. In this paper, we target to construct the automatic reuse environment for reuse of design patterns. In aspect of For Reuse, we developed the pattern library system on web, and In aspect of With Reuse, we developed the reuse supporter for retrieving, understanding, acquiring and restructuring of design patterns. Also, we classified and cataloged the design patterns with behavior and intend after domain analysis. Our experience indicates systematic pattern reuse process on reuse library. Client user can share the design patterns by web browser.

1. 서 론

소프트웨어 재사용은 소프트웨어 위기 극복이라는 끊임없는 노력에 대한 최상의 해결책으로서 인식되고 있다. 초기의 "as-is" 개념의 모듈 기반 프로그래밍 정의에서 시작하여 상속 개념을 바탕으로하는 객체, 클래스 중심의 객체지향 기술, 그리고 비즈니스 관점에서 부품의 조립에 초점을 두는 컴포넌트 기술에 이르기까지 다양한 관점의 재사용 방법론이 전개되고 있다. 기존 객체지향 기술은 클래스 라이브러리에서 제공되는 정의된 상속 기반 클래스들이 기능적인 충족성을 제공하는 빌딩 블럭이지만 적절한 재사용 요소의 식별과 응용 개발으로의 적용은 쉬운 작업이 아니며 구현 코드 레벨의 재사용이라는 한계성을 가지고 있다. 현재, 컴포넌트 기반의 응용 개발 기술은 응용의 개발과 유지 보수를 위

† 정 회 원 : 대구가톨릭대학교 컴퓨터정보통신공학부 교수
hangkon@cuth.cataegu.ac.kr
†† 학생회원 : 대구가톨릭대학교 컴퓨터정보통신공학부
kimjy@cuth.cataegu.ac.kr
논문접수 : 2000년 4월 19일
심사완료 : 2000년 10월 18일

한 최상의 해법으로 이슈화되고 있다. 그러나 요구되는 컴포넌트 획득을 위한 복잡한 선행 작업들 즉, 컴포넌트 비즈니스 영역의 도메인 분석과 명세화, 개발과 배포, 그리고 자동 라이브러리 구축들을 동반해야만 한다. 가령, 계층화된 컴포넌트 아키텍처에 따른 컴포넌트 저장소가 제공된다 하더라도 이들을 조립하기 위한 명확한 메카니즘 정의가 어렵다.

새로운 시스템 구축시 객체의 구조와 결합이 일정한 틀로써 유지되는 객체지향 프레임워크와 객체와 클래스들 간의 역할과 연결이 유사한 경우 자주 사용되는 구조를 일정한 형태로 설정한 설계 패턴은 추상화와 입자성의 기준에서 클래스 라이브러리와 컴포넌트 재사용의 절충점이라 할 수 있다. 설계 수준의 재사용성 제공으로 유연한 시스템 변경이 가능하고 응용으로의 적용시 개발자의 융통성이 표현된다. 나아가 이러한 일련의 재사용 프로세스를 프레임워크로 구조화시킴으로서 컴포넌트 조립을 위한 비용을 줄이면서 자동화할 수 있다. 웹은 재사용의 극대화를 위한 좋은 매체로서 역할을 수행한다. 웹 서버 상의 패턴 라이브러리 제공과 클라이언트 상의 개별 응용 개발자들에 의한 공유는 라이브러리 관리자에 의한 독립적인 확장, 전개가 아니라 응용 개발자 자신들의 도메인 요구에 맞는 새로운 패턴의 자연적인 생성을 도모함으로써 패턴의 영역별 체계적인 확장이 가능하다. 즉, 설계 패턴들이 플랫폼이나 네트워크 경계에 대한 고려없이 독립적으로 상호 조작 가능하도록 함으로써 개발된 설계 패턴의 독립적인 전개뿐만 아니라 새롭게 보강된 기능성을 제공하는 다른 패턴들을 대체, 이용할 수 있는 투명한 모듈성을 제공한다. 따라서 이들 기술들은 재사용성 강화와 표준화된 소프트웨어 아키텍쳐 제공으로 이론적으로 약속되어진 객체지향 장점을 현실화할 수 있다[1].

본 논문에서는 For Reuse 관점에서 재사용 가능한 설계 패턴을 모아서 패턴 분류 체계를 정의하고 이를 기준으로 유사 패턴을 식별, 카탈로깅한다. 또, With Reuse 관점은 웹 상에서 구축된 설계 패턴 데이타베이스로부터 재사용 컴포넌트인 설계 패턴을 클라이언트들이 빠르게 검색하고 이해하며 구축 가능한 응용에 사용자 관점에서 쉽게 재구성 할 수 있는 라이브러리를 제시한다. 사용자는 웹 브라우저를 통해 라이브러리로 접근하며 패턴 검색 및 분류 메카니즘에 기반한 패턴의 활용성을 향상시키고 라이브러리를 통한 자동화로 패턴의 발전 자생력 및 응용 생산의 다양성과 용이성을 확보한다. 제 2장에서는 소프트웨어 재사용 기술과 설계 패턴, 프레임워크에 대한 관련 연구와 3장에서는 설계 패턴 재사용 라이브러리에 관한 내용과 4장은 패턴 재사용 라이브러리 프로토타이핑에 대해서 기술하고 마지막으로 5장에서는 결론 및 향후 연구를 제시한다.

2. 관련 연구

2.1 소프트웨어 재사용 기술의 분류

최근 소프트웨어 개발을 위한 가장 핵심 과제는 소프트웨어 재사용 요소들의 라이브러리화와 이들의 식별 및 조립 그리고 이를 위한 과정들의 자동화이다. 그러나 응용 재사용 요소들을 시스템으로 전개하기 위해서는 많은 기술적인 복잡성과 비용이 요구되며 특히 모델 변경이 요구를 통한 소프트웨어의 독립적 구성이라는 사용자 관점의 유연성은 무시되기도 한다. 소프트웨어 재사용 단위는 소프트웨어 개발 단계와 응용 도메인의 한정성, 추상화에 직접적인 영향을 미친다. 즉, 큰 규모의 재사용 단위는 보다 큰 추상성을 제공한다. 표 1은 소프트웨어 재사용의 각 종류들에 대한 특성과 장·단점을 비교한 것이다[2].

2.2 설계 패턴의 개념과 분류 방법

구현 단계 재사용의 플랫폼 환경과 응용 도메인의 한정성에 대한 문제는 설계 패턴을 통해 극복할 수 있다. 설계 문제의 추상화와 특정 영역에 대한 공통적인 해결책을 구성 요소간의 관련성을 정의함으로써 공통 도메인의 응용 구축에서 자동화된 아키텍쳐 생성을 가능하게 하는 설계 패턴은 응용 도메인에 대한 신뢰성 있는 설계 지식을 확보하면서 개별 개발자의 구현 특성을 제공할 수 있다. 이는 추상화된 재사용의 개념에서 자신의 의도를 반영하고 자동화된 개발 프로세스 사용에 의한 아키텍쳐 공유로 표준화된 결과물들을 획득하도록 한다. 이는 인증된 설계 경험을 반영하며 클래스와 객체의 상위 수준 추상화의 이름, 특성을 명시함으로써 가능하다. 설계 패턴과 관련된 연구들은 크게 다음 세 가지로 요약된다[3] :

- ▶ 패턴의 발견, 수집, 서술하고 카탈로깅
- ▶ 패턴의 정규화 및 표준화
- ▶ 패턴 라이브러리 및 패턴 도구의 구축

특히 웹 상의 라이브러리부터 설계 패턴을 확보하고 재구조화를 통한 도메인 응용 구축으로 도메인 한정적인 애플리케이션 생성의 자동화가 지원될 때 보다 진보된 기술이 될 수 있을 것이다.

패턴에 대한 기존 분류 방법은 크게 세 가지로 정리할 수 있다. Gamma 등에 의해 정의된 분류는 패턴의 적용에 대한 도메인으로서 Scope와 패턴의 행위 즉, 목적에 의한 분류 Purpose로 구분되었다. Gamma의 패턴

표 1 소프트웨어 정보 재사용에 대한 각 타입들의 특징 및 장단점

코드 재사용	특징	• 가장 일반적인 재사용 타입으로 하나의 응용을 기본으로 다중 응용 상에서의 잠재적인 원시 코드 재사용
	장점	• 작성해야할 실제적인 원시 코드의 양을 줄임으로써 궁극적인 개발 및 유지보수 비용을 감소
	단점	• 효과의 범위가 프로그래밍 단계에만 제한되며 응용 프로그램 내의 결합도를 증가
상속성 재사용	특징	• 기존 클래스 내에서 구현되어진 행위를 이용하기 위해 응용에서 상속 사용 • is-a, is-like, is-kind of 관련성을 구현하는 객체지향의 근본 개념
	장점	• 이전에 개발되어진 행위들을 이용하여 응용의 개발과 유지보수 비용 모두를 감소
	단점	• 상속의 오용은 컴포넌트 재사용 기회 상실 유발
템플리트 재사용	특징	• 조직에서 주요한 개발 결과 문서, 모델 그리고 원시 코드를 위한 레이아웃의 일반적인 집합을 재사용
	장점	• 개발 가공물들의 일치성과 품질을 증가
	단점	• 개발자들은 그들 자신의 사용을 위한 템플리트 수정과 공동 작업자의 템플리트 변화를 공유하지 않음
컴포넌트 재사용	특징	• 미리 구축되고 완벽하게 캡슐화된 컴포넌트 사용
	장점	• 코드나 상속 재사용보다 재사용 범위가 더욱 확대 • 널리 알려진 공통의 플랫폼의 사용은 저비용으로 컴포넌트를 생성하고 판매할 수 있는 대규모 벤더를 위한 시장 제공
	단점	• 컴포넌트들은 작고 캡슐화되어진 오직 하나의 개념이기 때문에 컴포넌트들의 관리를 위한 더 큰 라이브러리를 요구
프레임워크 재사용	특징	• 공통적인 기술 혹은 비지니스 모두의 기본적인 기능성을 구현하는 클래스들의 집합을 사용
	장점	• 스캐치 개발의 복잡한 논리를 캡슐화하고 문제 도메인에 대한 해결점을 제시
	단점	• 프레임워크의 복잡성은 개발자가 많은 학습 과정의 요구함으로써 완전 학습이 어렵고 플랫폼 한정적이며 응용의 위험성을 증가시키는 단일 벤더와 결합시킴
소프트웨어 프로젝트 가공물 재사용	특징	• 이전에 생성된 개발 가공물의 이용 사례와 표준 문서, 도메인 한정적인 모델, 프로시쥬어와 지침 그리고 새로운 프로젝트 시작을 위한 응용의 재사용
	장점	• 객체 사이의 일치성을 항상시키고 각각의 새로운 변화에 대한 프로젝트 관리 부담을 감소 • 많은 가공물들을 온라인으로 구매하고 탐색
	단점	• 프로그래머에 의한 중복된 재사용
패턴 재사용	특징	• 공통의 문제를 해결하기 위해 대중적으로 문서화되어진 분석, 설계, 구현 패턴들의 재사용
	장점	• 다중 언어와 플랫폼 사이에서 구현되어질 수 있는 고수준의 재사용을 제공 • 경험 있는 객체지향 개발자들에 의해 인식되어진 문제로의 공통적인 접근을 사용함으로써 응용의 보강과 유지보수성 증가
	단점	• 즉각적인 해결책을 제시하지 않으며 패턴 구현을 위한 코드를 작성 요구
도메인 컴포넌트 재사용	특징	• 대규모의 재사용 가능한 비지니스 컴포넌트의 식별과 개발 • 많은 응용에서 공통적인 업무 행위들에 대한 대규모의 집약적인 번들을 표현하기 때문에 가장 큰 재사용 잠재성을 제공 • 도메인 개발 동안에 생산되어진 모든 것은 재사용 가능한 컴포넌트

은 다양한 도메인에서 일반적으로 사용되어질 수 있는 경험들을 추상적으로 표현한 것이므로 적용 영역에 대한 제한은 최소화할 수 있지만 이 설계 패턴의 분류는 너무 넓어서 주어진 설계 상황에서 소프트웨어 설계자를 위해 유용하지 않다[4].

Buschmann 패턴 카테고리는 모든 패턴은 소프트웨어 아키텍처를 생성, 구성하기 위해 사용되어질 수 있는 빌딩 블록의 시스템 형태를 형성한다는 개념이 바탕이 된다. 이 분류는 패턴들에 대한 일관화된 표현과 패턴들의 조합에 대한 정보를 포함하여 사용 관점에서의 유용성을 강조하였다. 분류 기준으로서 Granularity는 응용의 기존 구조에서부터 상세한 구현 초점까지 커버하기 위한 조건이다[5].

Zimmer의 패턴 카테고리는 Gamma 패턴과 그들의 관련성에 기반하여 더 정확한 의미적 정의와 분류에 의한 새로운 패턴 조합을 위해 세 가지의 의미적 계층으로 패턴을 분류하였다[6].

2.3 프레임워크

프레임워크는 연관되어 있는 특정 종류의 문제 해결책에 대한 재사용 가능한 추상 설계를 포함하고 있는 상호 작용하는 클래스들의 집합으로 정의된다. 프레임워크는 개발 응용에 대한 공통적인 아키텍처를 제공한다. 이는 응용 영역에 공통된 설계 결정을 추출해 냄으로써 애플리케이션 설계자는 개발하려는 애플리케이션에서 새로 요구되는 곳에만 집중하므로 프레임워크의 개발은 코드의 재사용을 넘어 설계의 재사용을 허용한다.

프레임워크는 사용자가 정의된 알고리즘을 조정할 수 있는 메소드를 허용하며 추상 수퍼 클래스에서 정의된 프로토콜과 조화되는 새로운 서브 클래스를 추가함으로써 확장된다. 타겟 도메인과 구축될 응용 도메인에 대한 정확한 이해를 확보한 프레임워크는 한정적인 모든 응용에 대한 공통적인 설계 구조를 제시함으로써 빠른 응용의 구축과 쉬운 유지보수가 가능하다. 설계 결정의 패키지 형태를 확보하기 위해서 프레임워크를 통한 설계 패턴으로의 구성이 필요하다. 프레임워크는 먼저 개발 범위에 따라 plug-in을 통한 블랙 박스 재사용과 핫스팟(hot spot)에 대해 완전하지 않은 클래스를 제공하는 화이트 박스(white box) 그리고 문제 영역에 따른 응용, 도메인, 지원 프레임워크로 구분된다.

그림 1은 프레임워크에 기반한 응용 개발 프로세스를 나타낸다[5].

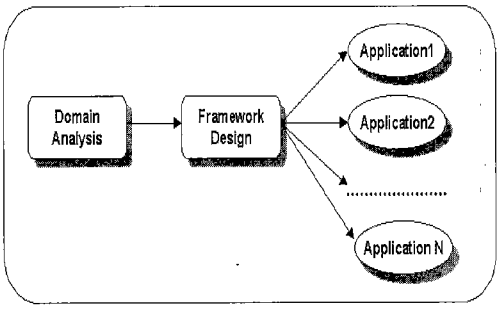


그림 1 프레임워크 기반의 응용 개발

3. 설계 패턴 재사용 라이브러리

3.1 웹 기반의 설계 패턴 재사용 시스템 기능적 요구

클래스 라이브러리는 개발자 고유의 스타일에 종속적인 재사용 요소의 이해와 활용에 많은 비용이 요구되며 응용 구축에 필요한 요소들을 찾고, 조합이 복잡하다. 따라서 상세 정보에 의존하지 않는 보다 추상화된 수준에서 소프트웨어의 재사용이 필요하다. 설계 패턴은 유사 도메인에 대한 표준화된 응용 구조를 생성할 수 있으며, 또한 개발자의 의도에 따라 패턴을 커스터마이즈 할 수 있는 융통성을 제공한다. 이런 설계 패턴에 대한 유용성은 인터넷을 통하여 통신하는 경험있는 소프트웨어 개발자들의 다양한 그룹에서 공유할 때 더욱 가치있게 된다. 경험있는 설계자들은 패턴들을 자신만의 툴킷으로 발전시킬 수 있으며 최근 연구는 이런 경험들을 널리 엑세스 가능하도록 패턴 조립에 초점을 두고 있다. 따라서 많은 개발자들이 설계 패턴을 생성, 문서화하며 WWW과 같은 네크워크 매체를 통해 정규화된 기호로 표현하고 엑세스하여 사용함으로써 소프트웨어 개발 능률을 향상시킨다.

설계 패턴 라이브러리는 패턴 구조의 정확한 이해와 새로운 패턴으로의 재구성을 위해서 모델링 도구가 요구되어진다. 또한 참조되어질 패턴을 기본으로 새로운 패턴의 추가와 기존 패턴의 변경에 따른 버전 제어 및 재분류 기능과 설계 패턴 정보와 그것에 대한 메타 정보 관리가 요구된다. 또한 검색의 질의 처리와 패턴의 수정, 삭제를 통해 사용자의 의사 반영에 능동적으로 대처할 수 있도록 지원하는 라이브러리 인터페이스와 패턴 평가 기능이 요구된다. 특히 인터넷 환경에서의 구축을 통해 설계 패턴의 효율적인 공유에 의한 시스템 설계의 능률을 향상시키고 신뢰성을 보장하는 응용 개발 자원을 위해 서버를 통해 클라이언트 요구를 충족시키는 설계 패턴을 검색하고, 검색된 패턴을 자신의 의도에 맞게 사용하기 위한 편집기와 구축기를 제공해야한다.

따라서 본 논문에서는 패턴 재사용 라이브러리로 두 가지 관점에서 접근한다 : 고 생산성의 재사용 패턴을 생성하기 위한 For Reuse 접근과 패턴 재사용의 극대화를 위한 With Reuse. 따라서 서버측의 패턴 라이브러리 구축시 기존 패턴과 새롭게 생성되어진 패턴을 분류 기준 정의를 통한 카탈로깅과 클라이언트에서 재사용 라이브러리 활용의 극대화를 통한 패턴 응용의 생성을 위한 일련의 자동화 도구의 개발을 목적으로 한다. 그러므로 본 시스템은 서버의 패턴 재사용 라이브러리와 클라이언트의 패턴 브라우징/재구조화 등을 포함하는 재사용 지원 도구로 구성된다.

그림 2는 웹 상의 패턴 라이브러리 시스템의 구조를 나타내고 있다[7,8].

3.2 패턴 라이브러리 시스템의 개념적 배경

본 논문에서는 설계 패턴을 컴포넌트로 하여 재사용 가능한 패턴을 모아서 그 적절성을 평가 한 후, 패턴 분류 체계 및 유사성에 따라 패턴을 추가, 갱신한다. 또한 패턴 라이브러리로부터 응용 구축에 필요한 설계 패턴들을 웹 브라우저를 통해 검색하고 이해하며 이를 또

다른 응용 구축을 위한 도메인에 새로운 패턴으로 생성할 수 있는 라이브러리를 제시하였다. 패턴 재구조화는 사용자 정의 도메인 패턴의 생성이 가능하며 나아가 웹을 통해 또 다른 재사용 요소로 클라이언트들에 의해 활용될 수 있다. 따라서 본 라이브러리는 패턴의 활용성 및 융통성 증가 및 라이브러리의 확장 자생력 제공으로 응용 생산의 다양성과 용이성을 확보할 수 있다.

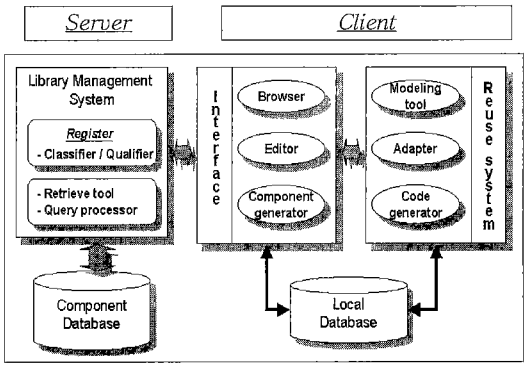


그림 2 웹 상의 패턴 라이브러리 시스템의 구조

3.3 설계 패턴 재사용 라이브러리 구성

추상적인 컴포넌트인 설계 패턴을 이용하여 정의되는 라이브러리를 보다 실행 지향적이며 동적인 환경 변화에 유연하게 대처할 수 있도록 For Component와 With Component 두 가지 측면의 프로세스로 구성된다. 그림 3은 패턴 라이브러리의 아키텍처이다. 패턴 라이브러리를 구축하기 위해 도메인 분석과 모델링을 통해 도메인 종속 설계 패턴을 식별, 구현한다(For). 또한 운용 영역에 따라 기존 설계 패턴을 검색하며 제어하고 필요에 따라 재구성하여 응용 시스템을 생성한다(With).

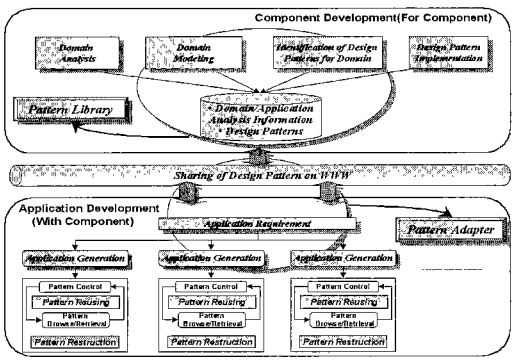


그림 3 패턴 재사용 라이브러리 아키텍처

3.3.1 For Reuse 접근

설계 패턴 라이브러리는 서버 시스템으로 웹 환경에서 설계 패턴을 데이타베이스화한 것이다. 재사용 단위로 설계 패턴을 채택하고 이를 활용하기 위한 표준화된 설계 정보를 구성한다. 사용자는 라이브러리에 저장된 재사용 가능한 설계 패턴을 적절히 검색하고 이해하는 패턴 수요자로서의 역할과 함께 각 응용 도메인에서 새롭게 정의된 패턴을 제공하여 기존 패턴보다 특화된 패턴으로 수정하는 공급자로서의 역할을 수행한다.

본 논문에서 구축한 패턴 라이브러리에 저장된 패턴은 기존에 Gamma가 정의한 대표적 참조 패턴인 Gamma 패턴, 특정 도메인 한정적인 패턴인 하이퍼미디어 패턴, Java AWT와 Application 패턴, 네트워크 패턴으로 구성되어 있다. 사용자가 정의한 이러한 도메인 패턴은 도메인 분석을 통해서 얻어진 정보를 바탕으로 기존 패턴을 확장, 수정하여 획득된 패턴이므로 실제 애플리케이션 적용 시에 높은 재사용 효과를 가질 수 있다.

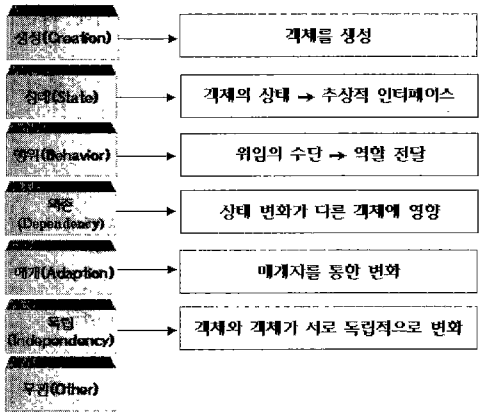
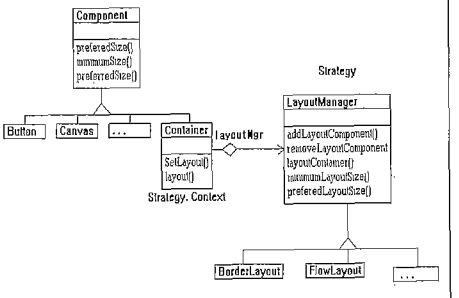
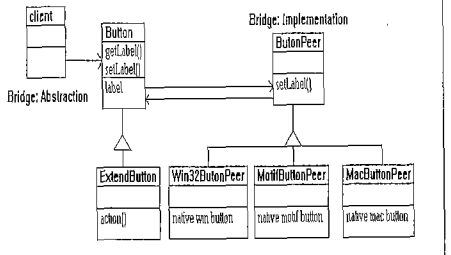
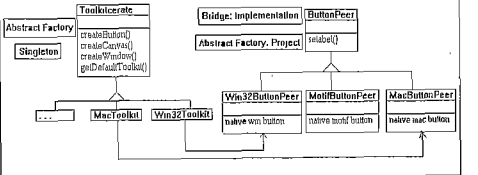
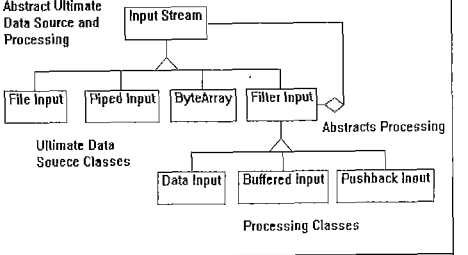
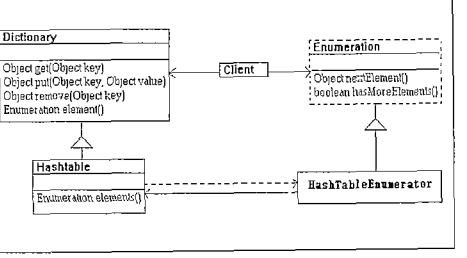


그림 4 패턴 분류 기준

(1) 패턴 분류 방법

분류는 라이브러리의 관리와 검색을 위한 가장 중요한 단계이다. 기존의 패턴 분류 방법 중 Gamma의 분류는 넓은 범위의 광위적 패턴들을 포함하나 정확히 상황에 꼭 적용될 수 있는 패턴 식별이 어렵다. Zimmer의 분류는 Gamma 패턴의 관련성을 기준으로 계층화시켰다. 그러나 이 방법은 구조적인 특성만 고려했을 뿐 패턴의 사용 의도나 의미와 역할에 대한 고려를 무시했다.

표 2 Java 도메인 패턴

도메인	사용 예		패턴 구조도
Java AWT	STRATEGY	• 인터페이스 뒤에서 알고리즘을 캡슐화하는 Strategy와 Strategy Object를 유지하고 이에 대한 알고리즘 구현을 위임하는 Strategy Context 역할 수행 • LayoutManager는 Layout 정책을 위한 공통 인터페이스를 정의하며 Container는 단지 Layout Manager에 의해서 정의된 인터페이스에만 독립적이고 이 인터페이스는 구현하는 모든 Layout들과 같이 동작 • Strategy는 StrategyContext를 조작하며 Strategy Context는 클라이언트 요청을 Strategy로 전송하며 클라이언트들은 StrategyContext와만 상호작용	
	Bridge	• 자바 API는 플랫폼에는 독립적인 반면 내부적으로는 플랫폼 한정적인 위젯 사용 • Bridge 패턴은 구현으로부터 추상화를 분리함으로써 분리된 2개의 추상화는 구현에 독립적이며 동적으로 변화 • 컴포넌트 객체들은 "Peer" 객체를 가지며 이것에 대한 인터페이스는 Peer 인터페이스에 의해 정의되어지고 한정적인 위젯에 연결 관계를 차례로 제공 • Peer는 구현에, 컴포넌트는 추상화에 대응	
	Abstract Factory	• Peer 인터페이스들이 다른 플랫폼에 대해서는 다른 컴포넌트 구현들을 허락하기 위해서 실행시 선택된 컴포넌트들을 위한 적절한 Peer 구현을 위한 지식 필요 • 툴킷 클래스가 Abstract Factory로 이것은 Peer 객체들의 생성을 위한 메소드를 정의	
Java I/O 패키지	Decorator	• 다양하고 정규적으로 순서화된 입력, 출력 그리고 프로세싱 알고리즘을 추상화 • 데이타 입력과 출력에 관계없이 독립적으로 그리고 data가 또는 다르게 처리되었는지에 관계없이 독립적으로 InputStream과 OutputStream 상에서 조작 • 전체 클래스가 아닌 개별적인 객체에 부가적인 역할을 추가하는 것으로 기능성 확장시 서브 클래스가 유동적으로 대처	
Utilities Package	Iterator	• Dictionary는 Dictionary 객체 관리를 위한 오퍼레이션을 정의하고 Enumeration 객체를 생성하기 위한 인터페이스를 정의 • Enumeration은 Iterator 패턴의 기본적인 반복 능력을 정의하여 클라이언트의 방문이 완전해질 때까지 Dicionary 요소들을 방문 • Iterator 패턴은 Dictionary 클래스를 액세스하고 방문하며 그것을 Enumeration 객체로 삽입	

본 논문에서 제시하는 분류 기준은 i) 패턴의 메타 정보와 ii) 적용 가능한 도메인과 iii) 패턴의 행위적 특성 즉, 패턴 인터페이스간의 관련성을 기반으로 한다. 따라서 기존 패턴들을 재정의하여 패턴들의 역할에 따라 조합하기 위한 그림 4와 같은 7가지 기준을 제시한다. 7가지로 분류한 이유는 각 패턴이 사용되는 의도에 따라서 의미적으로 패턴의 행위가 다음과 같이 분류될 수 있었고 또한 패턴 검색에서 사용함으로써 좀더 명확한 재사용 가능한 패턴을 획득할 수 있다. 분류 기준으로 생성(creation)은 객체를 생성하기 위한 것이고, 상태(state)는 추상적 인터페이스를 통한 객체의 상태, 행위(behavior)는 위임의 수단으로써 역할을 전달하며 의존(dependency)은 객체의 상태 변화가 다른 객체에 영향을 미치는 것을 표현한다. 또 매개(adaptation)는 객체 사이의 매개자를 통한 변화를 나타내고, 독립(independency)은 객체와 객체가 서로 독립적으로 변화하는 것이며, 나머지 패턴들은 무관(other)으로 간주하여 분류되어진다.

그림 5는 패턴의 행위적 분류 기준에 따라 Gamma 패턴을 분류한 것이다.

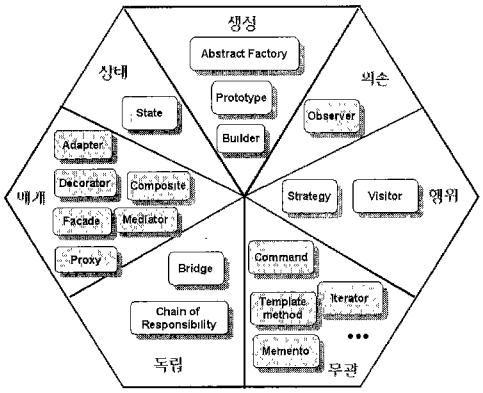


그림 5 Gamma 패턴 분류 예

(2) 패턴 유사성 식별

패턴의 유사성은 패턴 관련성에 초점을 두고 식별하게 된다. 새로운 패턴 등록 시에는 패턴이 나타내고 있는 속성을 위한 분류 항목의 단어에 맞는 것을 찾고, 분류된 항목에 속하는 패턴의 클래스 관련성과 등록되어질 패턴에서의 클래스 관련성(연관, 상속, 포함성)을 조사한다. 그리고 가장 일반적인 Gamma 패턴을 기준으로 도메인 한정적인 패턴으로의 계층화를 만들어 나가고, 도메인 한정적인 패턴은 Gamma 패턴과 비교하여 공통적인 부분(관련성)을 제외한 다른 확장된 부분이 있으면 좀더 특화된 패턴으로 간주되어 하위 계층으로 구분된다.

(3) 식별 패턴

라이브러리 구축을 위해 본 논문에서 식별 패턴은 Gamma 패턴에서 제시된 기존 패턴과 이것의 범용성을 수용하여 사용자가 도메인 분석을 통해 얻어진 정보로 확장한 특화된 도메인 패턴으로 구성된다. 이것은 범용성과 함께 도메인에 밀접한 한정성을 획득할 수 있어 높은 재사용성을 가질 수 있다. 즉, 사용자 도메인 패턴은 특정 응용 도메인을 위한 것으로 사용자 관점을 많이 반영하지만, 유사한 의도의 개발 응용에 정확하게 사용 할 수 있다. 패턴 라이브러리에 저장되어진 패턴으로 Java 패턴 표 2와 네트워크 패턴 표 3, 하이퍼미디어 패턴 표 4가 있다[9,10,11].

예로 Web 기반의 Java API를 통한 네트워크 관리를 위해서 Client는 Web Server로부터 Applet을 다운 로드하고, CORBA IIOP 프로토콜을 사용하여 서버로 관리 정보를 요청하고 서버는 ORB를 이용해 해당 관리 시스템 정보를 Client에게 Applet 형태로 디스플레이한다. 이는 기술적 수준의 도메인 분석을 통해 모델링 후 개략적인 패턴이 유도될 수 있다.

그림 6과 그림 7 그리고 그림 8은 클라이언트의 정보 요청과 서버의 응답에 관한 간단한 모델링 예와 이를 바탕으로 개략적으로 유도될 수 있는 패턴이다. 여기서는 Gamma의 Iterator 패턴을 이용하여 적용했다.

Java 도메인에서 Java 인터페이스는 구현을 명확하게 구별하고 클라이언트로부터 구현 클래스를 은닉하며 서브 클래스를 재정의 하거나 확장 할 수 없도록 하는 기존 설계 패턴의 이슈를 충실히 따르고 있다. 따라서 Java 프로그램을 위한 설계 패턴의 적용은 높은 효율성을 제공한다.

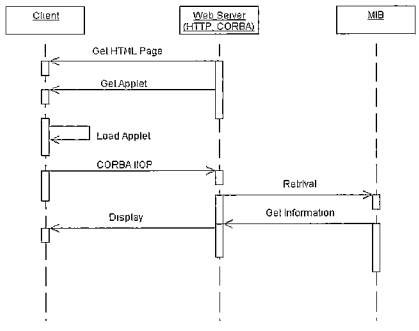


그림 6 Event-Trace 다이어그램

표 3 네트워크 도메인 패턴

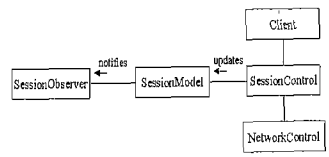
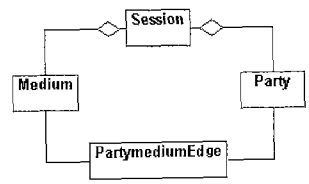
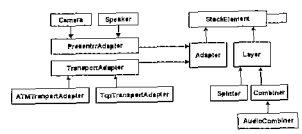
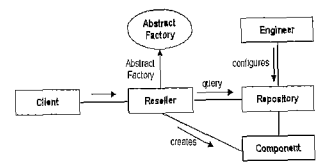
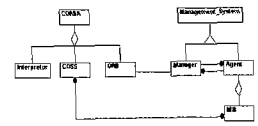
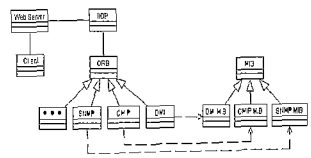
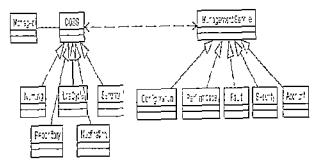
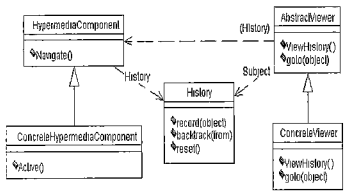
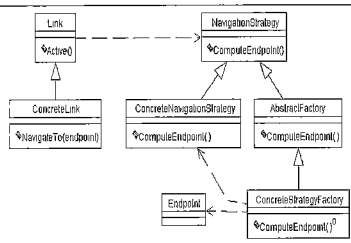
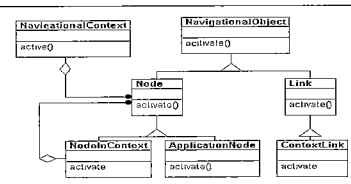
도메인	사용 예		패턴 구조도
Session Management	Session Control & Observer	• 섹션 상태를 제어, 유지, 노티파이 • N-ISDN과 B-ISDN과 같은 시그날링 프로토콜은 사용자 프로토콜(Cplane)로부터 제어 프로토콜(Uplanes)을 분해 • C plane는 U plane 프로토콜의 특성과 교섭 • 세가지 클래스, SessionModel은 상태를 유지하고 SessionObserver에게 상태 변화를 알림. SessionControl은 Client입장에 Network-Control을 사용하여 네트워크 한정적인 제어프로시쥬어를 실행하고 SessionControl은 즉각적으로 SessionModel을 반영	
	Parties & Media as first class citizen	• Session Control & Observation 패턴 내에서 SessionModel의 역할의 주요 구조를 형성하기 위해 적용 • 다중 parties와 연결을 수반하는 섹션을 구축하는 클라이언트는 복잡한 상태 정보를 유지해야 하는데 섹션이 진행 되어감에 따라 새로운 연결의 추가와 parties의 제거등의 다양한 방법으로 수정 • 각 세션은 수반되어지는 Parties와 Medium를 유지하는 PartyMediaum-Edge를 media와 연관되어지는 각 Party를 위한 상태를 유지하기 위해 사용하고 Parties와 Media는 섹션 내에서 first class citize	
Multimedia Realization	Layers	• Buschman에 의해 정의 • 네트워크 프로토콜의 시스템의 분리를 요구 • 프로토콜 컴포넌트가 어떻게 모델링 되는지를 해결하기 위해 시스템을 top에서의 최상 수준의 프로토콜과 bottom에서 최저 수준의 프로토콜을 가지는 적절할 개수의 프로토콜 계층으로 구조화	
Application Engineering	Pluggable Factory	• (C++)프레임워크의 재사용 가능한 확장성을 제공 • 프레임워크는 실체화되어질 수 없는 기본 클래스들에 대해 알고 있음에도 불구하고 실체화 시키기 위해 Factory Method는 기본 클래스 객체를 반환하는 시그네쳐를 갖는 추상 인터페이스 상에 기초한 해결책을 제시 • 사용자들은 반환되어지는 concrete 클래스들의 인스턴스들의 상속을 통해 이들 인터페이스들의 구현을 커스터마이즈	
Network Management	CORBA to Management 패턴	• CORBA와 network 관리 서비스 간의 매핑 관계 • CORBA 객체 서비스와 CORBA Facility는 네트워크 관리 및 통신을 위한 기본 서비스를 제공 • 중복되고 시스템 한정적인 서비스의 호환성과 추상화를 제공 • CORBA 서비스는 게이트웨이로써 다양한 관리 시스템과 동적으로 매핑 가능	
	Manager-Server 패턴	• 웹 기반의 Java API를 통한 네트워크 관리 • Client는 IIOP Protocol을 통해서 바로 관리 정보를 요청 • Server는 ORB를 이용해 해당 관리 시스템의 서비스 결과를 Java 가상 기계를 통해 획득, Client는 이를 Applet의 형태로 다운로드 • 웹 브라우저가 있는 어느 곳에서도 관리가 수행 가능 • JMAPI의 지원으로 확장성 있는 시스템 운영 가능	
	COSS-Management Service 패턴	• COSS와 관리 서비스의 투명한 수행 • CORBA를 통한 COSS와 CORBA 객체 지원 서비스들은 표준 네트워크 관리 서비스 수행의 추상성이 극대화된 네트워크 통신 프레임워크를 제공 • 이질적인 네트워크 환경에서도 단일화 가능한 미들웨어 메카니즘을 제공	

표 4 하이퍼미디어 도메인 패턴

도메인	사용 예		패턴 구조도
Hypermedia	Navigation Observer	• navigation history viewer의 구성 • 하이퍼미디어 공간을 항해하는 동안 방문된 노드와 링크의 기록을 가짐 • viewer의 각각 새로운 타입을 위한 하이퍼미디어 컴포넌트 수정 • 동일한 설계와 인터페이스는 다른 애플리케이션에 재사용 가능하고, 다양한 viewer 스타일과 history구현 스타일을 허용	
	Navigation Strategy	• 노드 끝점의 계산으로부터 하이퍼미디어 링크의 활성화를 분리하는 알고리즘 • 알고리즘의 캡슐화를 설계하고 이를 사용하는 클라이언트로부터 독립적이므로 링크는 알고리즘과 함께구성 • 서브클래싱을 제공하는 하이퍼미디어 환경에서 동적 생성과 노드의 링크를 허용 • Navigation프로세스 분리는 링크의 다른 종류를 정의하고 단일, 다중 끝점의 다른 타입을 허용	
	Navigationa -l Context	• 더 나은 정보의 표현과 이해를 위해 context-related 가이드라인과 관련성을 포함하는 폐쇄된 항해 subspace를 제공하며, 동일한 노드는 또한 다른 context안에서 인식될 때, 다른 시각허용 • decorator는 노드가 방문되어진 context에 따라서 정보와 anchor를 추가	

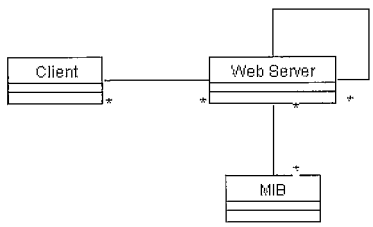


그림 7 Class 다이어그램

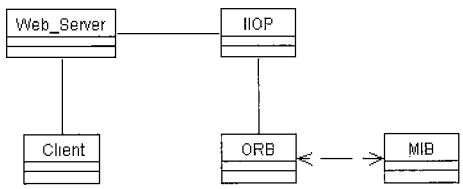


그림 8 유도된 패턴

(4) 재정의 패턴 군

그림 9는 본 논문에서 기존 패턴의 분류를 기반으로 새롭게 재정의된 패턴 행위적 분류 기반의 특정 도메인 패턴을 분류한 것이다.

3.3.2 With Reuse 접근

웹 상에 분산된 클라이언트들은 제공되는 UI를 통해 질의를 작성하고 패턴을 검색하며 응용에 맞도록 커스터마이즈 하고 응용에 적용시킨다.

그림 10은 패턴 재사용 라이브러리의 전체적인 시스템 구조이다.

(1) 패턴 검색

본 패턴 재사용 라이브러리에서 제공하는 검색 방법은 다음과 같이 3가지를 제공한다.

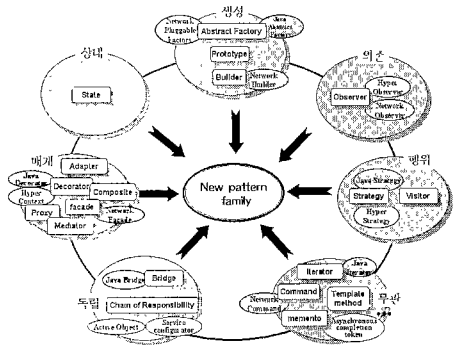


그림 9 재정의되어진 패턴 카탈로그

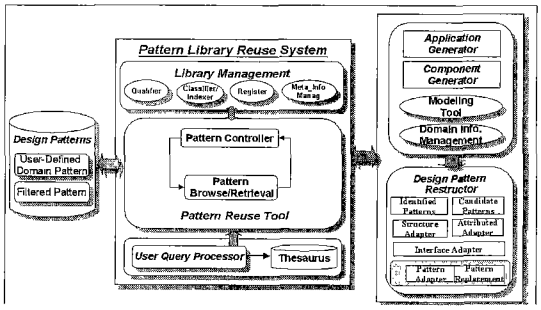


그림 10 패턴 재사용 라이브러리의 구조

① **브라우징 검색** : 패턴 라이브러리의 분류 구조를 직접 찾아 들어가면서 원하는 패턴을 찾는다. 본 논문에서는 도메인 별로 분류, 정의된 패턴을 하이퍼미디어 형식에 따라 세부적인 도메인 패턴으로 점차적으로 접근한다.

② **패싯 검색** : 도메인 패턴과 Gamma 참조 패턴의 관점을 참조해서 패싯 항목으로 6개(scope, purpose, domain, 관련패턴, 패턴특성, 사용목적)을 설정하여 이들의 조합에 의해 패턴을 검색한다. 항목으로는 Gamma의 분류 체계인 'scope', 'purpose'와 Java, 하이퍼미디어, 네트워크 등의 '도메인'에 초점을 둔다. '관련 패턴'은 찾고자 하는 패턴과 유사한 패턴을 의미하며, '패턴 특성'은 패턴의 성질을 서술한다. '사용 목적'은 패턴의 행위에 초점을 두고 객체 생성, 상태, 행위, 의존, 매개, 독립을 나타내는 패턴 분류 표현이다.

③ **이름 검색** : 패턴의 이름을 직접 키 필드에 입력함으로써 패턴의 정보를 획득한다.

(2) 패턴 이해 정보

검색 패턴에 대한 상세 정보를 제시함으로써 최상의 설계 패턴 선택을 유도한다. 이는 Gamma가 정의하고 있는 형식인 패턴의 의도, 응용성, 구조, 구성 요소, 결과 등에 관한 정보를 획득할 수 있다. 이를 바탕으로 적절한 패턴을 선택한 후 패턴 정보를 다운 받아 사용자의 의도에 맞게 재사용 가능하다.

(3) 패턴 재구조화

응용으로의 설계 패턴의 재사용이 직접적으로 이루어지기 위한 전단계로서 패턴의 응용에 꼭 필요한 요소로 변환시킨다. 이것은 기존의 일반적 특성의 설계 패턴을 도메인에 밀접한 특성을 갖는 설계 패턴으로 재구조화함으로써 구축 응용을 위한 보다 정확한 패턴을 확보할 수 있도록 할 뿐 아니라 패턴 라이브러리의 도메인에 기반한 자연스런 확장을 지원함으로써 유사 응용 개발에 있어서 패턴 히스토리에 의한 큰 단위의 재사용을 가능하게 한다.

본 논문에서는 기존 패턴을 사용자가 패턴 라이브러리를 통해 응용에 필요한 특수한 기능의 패턴을 확보함으로써 보다 재사용의 효율성을 증가시키고 나아가 자연스러운 패턴 라이브러리의 확장을 지원하는 방법으로 패턴 재구조화를 제시한다. 이것은 도메인에 적절한 기능성을 표현하고 동일 도메인에 대한 히스토리 정보 유지를 통해 구축 응용 및 차후의 응용과 패턴 생성의 바탕이 된다.

패턴 재구조화를 위한 프로세스는 컴포넌트 기반의 응용 구축에서의 접근에 기초한다. 따라서 적절한 도메인 분석과 이에 대한 모델링, 그리고 패턴으로 대체되어질 수 있는 요소들의 식별과 적용을 위한 패턴과 응용의 인터페이스 및 구조의 조정을 통해 도메인 기능성이 포함되어 확장되어진 모델 생성에 의해 이루어진다. 이는 다음과 같은 프로세스를 가진다.

① **도메인 분석** : 구축하고자 하는 응용의 특성과 사용자의 요구 조건을 파악

② **응용 모델링** : 구축할 응용의 구성 요소와 이들 간의 관련성을 나타내는 객체 다이어그램을 모델링 도구를 이용하여 작성

③ **패턴 적용 요소 식별** : 응용 분석 정보를 기반으로 모델링에서 일반적으로 적용되어진 패턴 식별

④ **도메인 패턴으로 재구조화** : 응용의 도메인 특성을 더욱 강화시키기 위해 구조 수정, 새로운 속성을 추가, 정의 등을 통해 도메인 분석 정보에 적절하게 식별된 패턴의 재구성

⑤ **확장된 모델 작성** : 식별되어진 패턴들을 기존에 작성되어진 응용 모델에 대체함으로써 확장된 응용 모델을 확보

(4) 응용 예

설계 패턴의 적용을 통한 응용 생성은 패턴 재구조화기를 통해 도메인에 보다 밀접한 패턴으로 생성되어진 후 실세계의 개념과 연관지어 관련성을 정의하고 이를 객체지향 모델링으로 작성함으로써 이루어진다. 따라서 Generator는 특정 도메인에 관한 또 다른 재사용 요소를 생성하여 계속적인 라이브러리를 전개하는 Component Generator와 이를 컴포넌트 기반의 응용을 위한 아키텍쳐 생성을 위해 사용함으로써 실질적인 패턴 재사용을 수행하는 Application Generator로 이루어진다. 특별히 여기서는 패턴의 도메인 특성을 부여하고 관리함으로써 패턴 재사용의 효율성에 직접적으로 기여할 수 있는 Domain Info. Management 도구와 도메인 응용 구축을 위한 시각적 모델링 도구가 지원된다. 특히

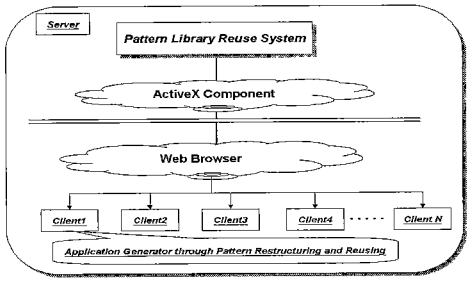


그림 11 패턴 재사용 라이브러리의 이용 구조

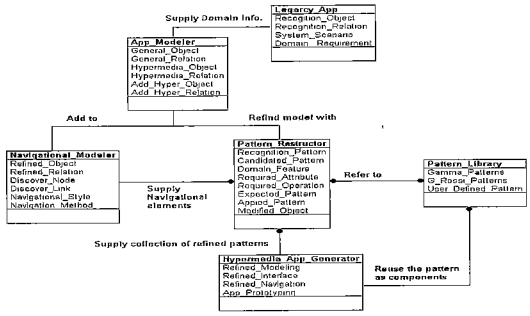


그림 12 패턴 재사용 라이브러리 객체 모델

외부 요소들과의 인터페이스의 관련성 조절을 통해 완벽한 객체지향 설계를 가지도록 재구성하며 구현을 위한 프로토타이핑을 작성한다.

본 논문에서는 설계 패턴을 웹을 사용함으로써 환경적 요소에 관계없이 엑세스 가능하며 패턴 정보를 기반으로 파라미터 형식의 패턴 인터페이스 조작으로 패턴을 재구조화하여 응용과 조합한다. 그리고 실행시의 활용은 대화적 방식으로 인터페이스의 템플리트 제공과 매개 변수 전달을 통해 응용으로 적용한다. 그림 11은 클라이언트와 서버간의 이용 구조를 나타낸 것이다.

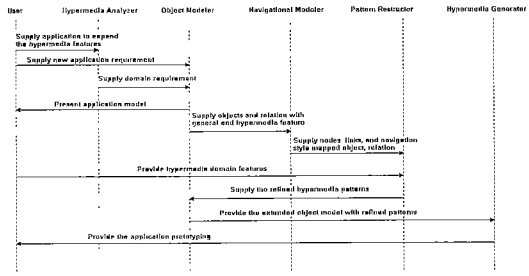


그림 13 패턴 재사용 라이브러리의 동적 모델

4. 패턴 재사용 라이브러리 프로토타이핑

4.1 구현 기술

설계 패턴 재사용 라이브러리는 Windows NT에서 지원하는 IIS 4.0 Web Server에서 VB를 근간으로한 ASP 스크립트와 데이타베이스로 MS-SQL Server 6.5를 사용하고 웹 브라우저로는 Internet Explorer 혹은 Netscape를 사용하여 구현하였다. ASP 페이지는 Script와 HTML Code의 결합으로 구성되며 Script와 HTML Code는 ASP가 지원하는 내장 객체에 대한 호출을 포함할 수 있다. IIS 4.0 Web Server에는 ASP Engine이 있는데 이 Engine은 웹 브라우저에서 ASP Page를 호출하면 Script 언어를 처리하고 그 결과를 HTML Stream으로 삽입한 다음에 요청한 Web Browser로 그 결과를 반환한다. 또한, ActiveX Control 생성을 지원하는 모든 언어로 Server Control을 생성할 수 있다. 또한 Visual J++로 개발한 Java Component로 ASP 스크립트를 교체할 수 있다.

패턴 데이타베이스로 SQL(Structured Query Language) Server를 사용했다. 이는 최근에는 웹 서버 구축을 위한 기반 데이타베이스 시스템으로 많이 사용되고 있다.

4.2 시스템 설계

그림 12와 그림 13은 설계 패턴 재사용 라이브러리의 전체 객체 모델로 라이브러리를 구성하는 대표적인 클래스들의 객체 다이어그램과 동적인 시나리오를 간략히 나타내고 있다. 크게 패턴 데이타베이스와 패턴 라이브러리 재사용 시스템과 Generator/Restructor로 구성된다.

4.3 응용 개발 프로세스

패턴 라이브러리를 통해 응용을 개발하는 절차는 다음과 같이 정의된다. 응용의 도메인 분석에서부터 시작하여 필요 패턴의 재구조화를 통한 설계, 구현으로 새롭게 도메인 특성을 갖는 컴퍼넌트들을 생성하고 이것들의 조합으로 사용자 요구에 부응하는 응용을 생성한다.

① **도메인 분석** : 아키텍처 구성의 특성을 파악하고 동일 도메인에 사용될 후보 컴퍼넌트들을 식별

② **응용 모델링** : 구축할 응용의 구성 요소와 이들 간의 관련성을 객체 다이어그램으로 작성

③ **적용 패턴 식별** : 패턴에 대한 본질적인 지식을 바탕으로 패턴 구조의 매칭 스키마를 이용하여 시 스템 구축시 적용 가능한 후보 패턴을 식별

④ **라이브러리로부터 패턴 검색** : 패턴들을 서버에 구축되어진 패턴 라이브러리로부터 검색

⑤ **설계 패턴 적용/정제** : 검색된 패턴의 구조를 응용의 모델에 적용. 유사 구조는 패턴이 에트리뷰트와 오퍼레이션 수정을 통해 대체되며 사용자 선택에 의한 패턴

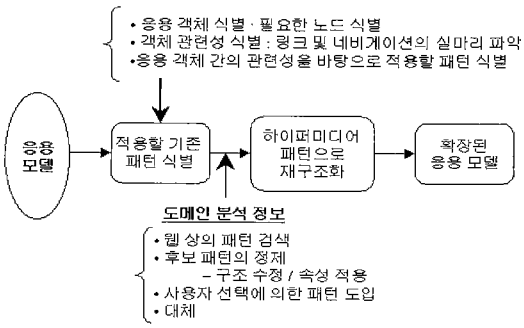


그림 14 하이퍼미디어 응용으로의 재구조화 프로세스

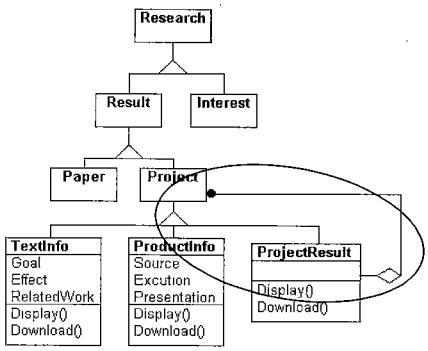


그림 15 객체 모델의 예

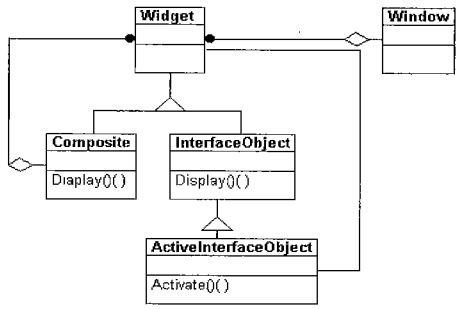


그림 16 식별된 하이퍼미디어 패턴

사용은 대화적인 인터페이스를 점차로 채워나가면서 가능한 패턴으로 대체

⑥ 완성된 코드 작성 : 자동 코드 생성기를 이용하여 확장된 응용 모델에 대한 템플리트 코드를 작성, 편집하여 실제 응용을 위한 완성된 코드를 작성 예로 하이퍼미디어 응용을 위한 재구조화는 다음과 같이 표현될 수 있다.

하이퍼미디어 패턴으로의 재구조화를 위한 개념적인 구조는 그림 14와 같이 나타낼 수 있다. 이것은 작성된 응용 모델에서 적용 패턴의 식별과 이를 도메인의 요구에 따른 하이퍼미디어 패턴으로의 적용 두 부분으로 구분되며 그림 15와 같이 작성되어진 객체 모델에서 식별되어진 설계 패턴을 적절한 하이퍼미디어 응용 요구를 수용하는 하이퍼미디어 설계 패턴으로의 재구조화하는 과정은 다음과 같다.

① 도메인 분석 : 응용의 설계 요구와 기능적 특성을 파악함으로써 노드와 링크 식별, 네비게이션 방법의 성격을 결정한다. 본 논문의 예에서는 프로젝트 결과물 제시를 사용자의 선택에 따라 상세화된 정보(전체 정보의 서브 정보)를 제시하는 구조를 요구한다.

② 적용된 기존 패턴 식별 : 작성된 모델로부터 일반적인 패턴을 식별하는데 예에서는 프로젝트 결과 형식의 다양성 제공을 위한 Composite 패턴을 감지할 수 있다(그림 15에서의 타원 부분).

③ 재구조화될 후보 패턴 식별 : 응용 도메인 분석의 결과에 따라 네비게이션 요소를 식별한다. 기본적으로 하나의 객체는 하나 이상의 노드로 매핑되며 링크는 정의된 관련성을 바탕으로 사용자가 제시한 요구 사항으로 연결된다. 만약 도메인 분석에서 복잡한 의존성이 확인된다면 다양한 하이퍼미디어 특성이 추가되어야 하며 네비게이션 경로도 복잡해진다. 따라서 각 노드의 특성에 따라 구별된 네비게이션 방법을 결정한다. 적용할 수 있는 패턴들은 일반적인 응용 구조를 위한 범용성의 Gamma 패턴 및 하이퍼미디어 응용의 특성 구현을 위한 하이퍼미디어 설계 패턴들이다. 여기서 모델을 구성하는 응용 객체들은 하이퍼미디어 시스템에서의 구성 노드가 되며 객체들 간의 관련성은 링크로 매핑되며 네비게이션을 위한 실마리를 제공한다. 예제에서는 사용자가 원하는 항목의 결정에 따라 네비게이션 없이 적절한 인터페이스를 통해 자세한 서브 정보를 제공 및 체계화를 위한 Information On Demand를 식별할 수 있다(그림 16).

④하이퍼미디어 패턴으로 정제 : 응용의 도메인 특성을 더욱 강화시키기 위해 구조 수정, 새로운 속성을 추가, 정의 등을 통해 도메인의 분석 정보를 패턴에 적용시킨다. 이를 위해서는 웹상에서 제공되어지는 설계 패턴 라이브러리로부터 검색한 설계 패턴을 참조로 하여 인식되어진 후보 패턴들을 구조 수정이나 새로운 속성을 정의, 추가함으로써 정제하거나 도메인의 특성에 따라 특정 패턴을 도입한다.

⑤ 설계 패턴 도입/대체 : 정제되어지거나 패턴 검색을 통해 확보되어진 패턴들을 응용에 새롭게 도입하거나

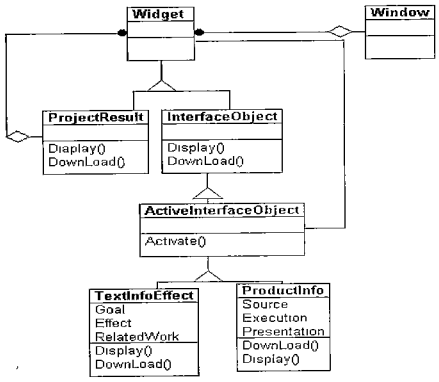


그림 17 패턴 도입에 의한 모델 확장

기존 패턴들과 대체시킨다. 예에서 Window는 컴퍼넌트 즉, widget의 집합이며 widget은 또한 여러 컴퍼넌트의 composite이다. 인터페이스 객체는 제시되어질 특별한 에트리뷰트 정보를 의미하며 ActivationInterface Object는 외부의 이벤트를 제어객체(사용자가 선택하는 버튼)로부터 확보함으로써 이벤트에 따라 표시할 윈도우 정보를 한정시킨다(그림 17).

⑥ 확장된 모델 작성 : 확정된 패턴들을 기존에 작성된 응용 모델에 대체함으로써 확장된 하이퍼미디어 응용 모델을 획득한다.

4.4 실행 예

그림 18은 사용자 인증을 위한 인터페이스로 만약 클라이언트 사용자가 시스템에 등록되어진 ID와 패스워드를 갖는다면 라이브러리를 엑세스할 수 있다. 만약 ID를 갖지 않는다면 새롭게 등록한다.

패싯 검색을 위한 인터페이스는 그림 19에 나타난다. 이 메소드는 선택되어진 패싯 항목의 조합으로 패턴을 검색하는데 본 시스템에서는 최상의 재사용 패턴의 결정에 도움을 주기 위해 검색되어진 패턴에 대한 이해 정보를 제공하며 또한 앞에서 분류한 7개의 분류 기준을 검색을 위한 패싯 항목으로 채택함으로써 의도된 패턴을 획득할 수 있다. 그리고 좀더 풍부한 패턴을 검색하기 위해 전자의 3항목은 AND 조건으로, 후자의 3항목은 OR 조건으로 검색할 수 있어 패턴의 부재를 최소화할 수 있다.

그림 20은 검색되어진 잠재적인 재사용 패턴의 상세 정보를 제공하는 인터페이스이다. 클라이언트의 사용자들은 패턴 라이브러리로부터 검색한 패턴의 구조를 다운로드 받아 local의 모델링 도구를 사용하여 새로운 응용을 위해 재구조화 함으로써 재사용 가능하다. 패턴을

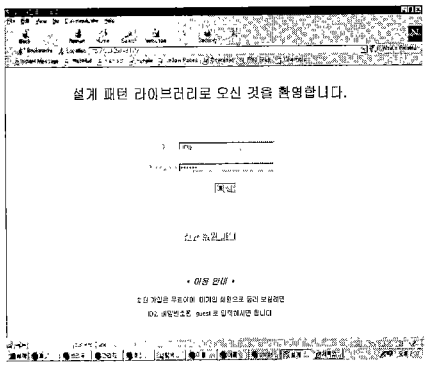


그림 18 패턴 라이브러리의 로그인

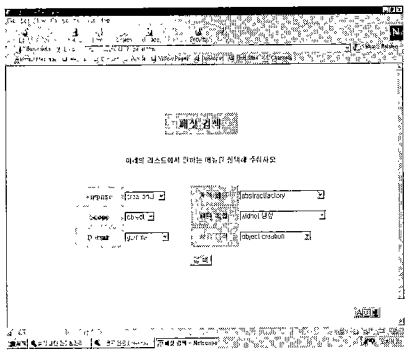


그림 19 패싯 검색을 위한 인터페이스

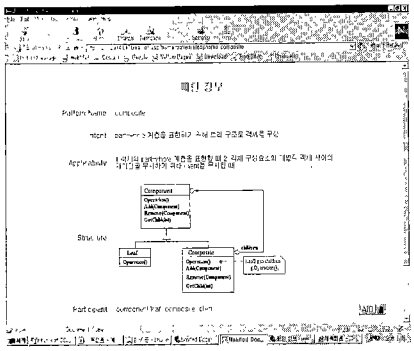


그림 20 패턴의 이해 정보

서버로부터 다운 받기 위한 인터페이스는 그림 21에 제시된다. 그림 22는 검색되어 다운로드되어진 하이퍼미디어 패턴 중 Information On Demand 패턴을 Rational Rose를 사용하여 사용자가 의도하는 하이퍼미디어 패턴으로 재구조화 즉, 패턴의 정제와 도입을 통한 확장된 모델을 제시한다.

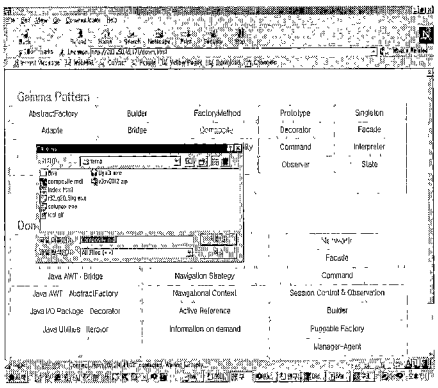


그림 21 패턴 정보 받기 인터페이스

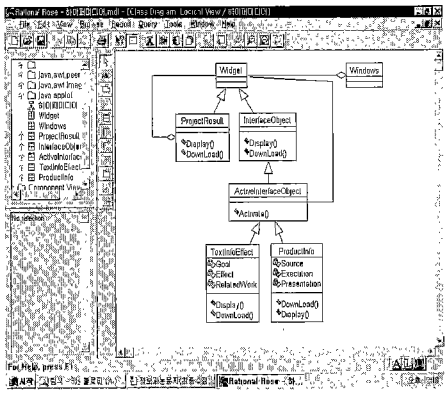


그림 22 패턴 재구조화 및 모델링 적용 예

4.5 시스템 평가 및 특징

4.5.1 평가 기준

설계 패턴 재사용 라이브러리는 유용한 패턴의 빠르고 정확한 식별과 재구조화 과정의 용이성 그리고 인터페이스 수준에서의 결합에 의한 응용 구축으로의 전개 등의 관점에서 효율성을 평가할 수 있다. 패턴 식별은 기존의 클래스 재사용 시스템에서와 같이 재현율과 정확도를 사용한다. 그러므로 정확도는 검색된 패턴 중 적절한 패턴의 비율이며 재현율은 라이브러리에 존재하는 패턴 중 사용자 질의에 부합하는 패턴의 검색 비율이다. 표 5는 부품의 정확도와 재현율의 의미이다.

표 5 검색 효율 평가를 위한 부품의 분류

(a) 재현율 = $\frac{\text{검색된 적절한 부품의 수}}{\text{전체 부품 중 적절한 부품의 수}}$
(b) 정확도 = $\frac{\text{검색된 적절한 부품의 수}}{\text{검색된 전체 부품 수}}$

패턴 재구조화에서의 용이성은 검색된 패턴의 이해 정보에 기반하여 얼마나 적절한 패턴을 선택했느냐에 따라 결정된다. 이것은 패턴 구조를 중심으로 변화된 패턴 모델 작성이 결과로 제시되기 때문에 적절한 구조와 명확히 구분되는 유사한 관련성 확보가 중요하다.

패턴의 적용은 라이브러리로부터 검색된 패턴을 사용자의 로컬 시스템으로 다운 받은 후 응용에 포함시킴으로써 가능하다. 이를 위해 제공받은 패턴 정보의 종류에 따라 다른 응용 구축기가 클라이언트에서 제공되어야 하는데 만약 원시 코드라면 응용 구축의 설계 수준에서 패턴의 추가로 응용을 확장시킬 수 있는 모델링 도구가 요구된다.

4.5.2 설계 패턴 재사용 라이브러리의 특징

시스템을 구성하는 여러 요소 중 주요 관점에 따라 특징을 요약하면 아래와 같다.

① **패턴 분류** : 도메인 분석을 통해 식별된 도메인 패턴을 가지고 새로운 패턴 분류 형식인 패턴 행위적 특성을 기준에 의해 7개로 재분류한다. 패턴 분류 기준은 생성(creation), 상태(state), 행위(behavior), 의존(dependency), 매개(adaptation), 독립(Independency), 무관(other)으로 구성된다.

② **패턴 검색** : 사용자의 선행 지식을 기반으로 원하는 검색 방식을 선택하여 점진적이면서, 하이퍼미디어적인 형식에 따라 혹은 사용자가 잘 알고 있는 패턴 이름을 입력하고 또한 사용자의 질의 작성에 편의를 위해 미리 작성해 놓은 질의 항목을 제시함으로써 선택적으로 패턴을 검색할 수 있는 다양한 질의 유형을 제공한다.

③ **패턴 데이타베이스** : 참조 패턴과 사용자 도메인패턴을 구분하여 계층화시켰다. 본 시스템에서는 Java, 하이퍼미디어, 네트워크 패턴을 제공한다.

④ **라이브러리 시스템** : 클라이언트 사용자의 요구에 부응하는 패턴과 관련 정보 제시 및 새로운 패턴의 등록과 패턴들의 관리를 목적으로 패턴에 대한 정보와 메타 정보 그리고 사용자 활용에 초점을 맞춘 도메인 정보 관리를 위한 서브 시스템들로 구성된다. 특히 웹 상의 다중 사용자는 웹 브라우저를 통해 쉽게 접근할 수 있다.

⑤ **패턴 재구조화** : 클라이언트에서 응용 구축에 패턴을 적용시키기 위해서는 구축할 응용의 도메인 특성을 충분히 포함하여 사용자가 최소한의 관여만으로도 적용이 되도록 해야 한다. 설계 패턴 재사용 라이브러리는 응용 도메인의 요구를 기존 유사한 도메인에 사용되어진 패턴들을 적절하게 재구조화하는 프로세스를 제공한다.

5. 결론 및 향후 연구

다양한 플랫폼과 도메인 상의 응용들의 효과적인 개발, 배포, 운영을 위해서는 환경적인 제약에 투명한 매개체를 통해 독립적인 컴포넌트의 조립을 위한 체계적인 접근과 자동화된 도구의 지원이 요구된다. 설계 패턴은 설계 경험에 대한 캡슐화된 빌딩 블록으로 개발 응용의 표준화된 아키텍처 제공을 통해 응용 도메인의 한정성과 클래스 수준의 재사용의 복잡성을 극복한다. 표준 아키텍처로서의 웹을 통한 패턴들의 공유는 여러 개발자들에 의해 다양한 도메인 요소로의 전개를 통한 패턴 라이브러리의 확장을 유도하고 동적이며 실시간적인 라이브러리 관리가 가능하다.

본 논문은 웹 환경 상에서 설계 패턴 재사용을 위한 자동화된 환경 구축을 위한 라이브러리를 개발한다. 따라서 For Reuse 관점에서 서버에는 각 도메인별 패턴 라이브러리를 구축하며, With Reuse 관점에서 패턴 검색, 이해, 획득 그리고 재구조화를 통한 응용으로의 전개를 위한 재사용 지원기를 구축한다. 본 시스템은 패턴 공유에 의한 유사 도메인 응용에서 기존 패턴의 신뢰성을 유지하며 사용자의 패턴 재정의를 통해 응용에 특화된 패턴을 포함한다. 또한 응용의 도메인 분석을 통해 패턴의 행위와 의도를 바탕으로 7개의 분류 기준을 설정하고 설계 패턴들을 기존 패턴을 바탕으로 분류 카탈로깅하였다.

향후 패턴 단위의 비실행성을 극복한 컴포넌트 라이브러리를 개발하고 반수동적인 패턴 라이브러리의 시각적인 통합 환경을 개발하며 나아가 컴포넌트 재사용을 위한 라이브러리를 구축할 것이다.

참 고 문 헌

[1] Mikio Aoyama, "New Age of Software Development : New Component-Based Software Engineering Changes the Way of Software Development," 1998 International Workshop on Component-Based Software Engineering, ICSE, p.124~128, 1998. http://www.sei.cmu.edu/cbs/icse98/papers/p14.html

[2] Scott Ambler, " A Realistic Look at Object-oriented Reuse," Software Development Magazine, July, 1998. http://msdn.microsoft.com/ developer/news/sdmag/ambler.htm

[3] Computer Systems Group, "A Designer' Assistant Tool Home Page," University of Waterloo, April, 1996. http://csg.uwaterloo. ca:80/ dptool/

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

[5] Michel Mattsson, "Object-Oriented methodological issues," Master Thesis, Development of Computer Science, University College of Karlskrona, 1996. 8.

[6] Walter Zimmer, Relationships Between Design Patterns, Pattern Languages of Programs 2, Vlissides et. al., eds., Addison Wesley, 1996.

[7] 김 행곤 외, "웹 환경에서 설계 패턴 재사용을 위한 라이브러리 구축", 한국정보과학회 춘계학술발표회, 제 24권, 1호, pp. 554-556, 1998. 4.

[8] 김 행곤외, "웹 상에서 설계 패턴 라이브러리에 기반한 재사용 시스템 구현", 한국정보과학회 추계학술발표회, pp.551-553, 1999, 10.

[9] G. Gossi, et. al, "Design Reuse in Hypermedia Applications Development," Hypermedia '97 Proceeding of 8th ACM Conference on Hypertext, pp. 57 ~ 66, 1997.

[10] 김 행곤 외, "자바 프로그램 개발을 위한 설계 패턴에 대한 연구", 한국정보처리학회 학술발표대회 논문지, 제 4권 1호, pp. 520 ~ 523, 1996. 4

[11] E. Gamma, "Applying Desgin Patterns in Java," Java Report, 1999, 11.

김 행 곤

1985년 중앙대학교 전자계산학과 졸업(공학사). 1987년 중앙대학교 대학원 전자계산학과 졸업(공학석사). 1991년 중앙대학교 대학원 전자계산학과 졸업(공학박사). 1978년 ~ 1979년 미 항공우주국 객원연구원. 1987년 ~ 1989년 AT&T 객원연구원. 1990년 ~ 현재 대구가톨릭대학교 컴퓨터정보통신공학부 부교수. 관심분야는 객체지향 소프트웨어공학, 재사용, 설계패턴, 컴포넌트기술

김 지 영

1997년 대구효성가톨릭대학교 전자계산학과 졸업(이학사). 2000년 대구효성가톨릭대학교 전산통계학과 전자계산학전공(이학석사). 2000년 ~ 현재 대구가톨릭대학교 전산통계학과 전자계산학전공 박사과정. 관심분야는 객체지향 소프트웨어공학, 재사용, 설계패턴, 컴포넌트기술