# 기존 시스템에서 CBD 지원을 위한 설계 패턴 재공학☆

# Reengineering Legacy systems into Design Patterns of Component Base Design (CBD)

김 국 보*
Guk-Boh Kim

## 요 약

클래스 단위의 응용 시스템 구성은 코드 중심의 컴포넌트 추상화로 각 요소의 독립성 유지 및 재사용이 구현상에만 치우쳐 효과가 미흡하다. 따라서 상위의 개발 단계에 대한 객체 지향적 접근을 위해서는 설계 문제의 추상화와 특정 영역의 일반적인 해결에 대한 정보 표현 및 구성요소 상호 간의 관련성을 패턴을 통하여 나타낼 필요가 있다. 또한 기존 시스템의 성능을 변경, 개선하고 변화하는 환경에 적응하기 위해서는 기존 시스템을 실험, 분석함으로써 정확한 이해를 하고 나아가 재사용 자원으로 활용할 수 있는 소프트웨어 역공학이 필요하다. 따라서 본 논문에서는 기존 객체지향 시스템 코드에서 설계 패턴 추출을 위한 역공학 적용에 관한 타당성과 설계 패턴 자동 추출을 위한 알고리즘을 Java를 적용하여 살펴보고 설계 패턴의 자동 추출을 위한 역공학 및 패턴 재사용을 위한 자동화 도구의 아키텍처를 설계 구현한다.

## Abstract

The effect of Application system with class units is not sufficient because of independency and reuse of Component elements due to component abstraction based on only source code. Therefore We need to apply design pattern approach to represent not only the problem abstraction but also information and relationship between system elements for generic solutions of specific domain. Also, it is essential to software reverse engineering acquiring the correct understandings of the system through examining the existing systems and utilizing the acquired knowledges as reusable resources.

In this paper, the extraction algorithm with JAVA and the validity of applying reverse engineering with extracting design patterns from source codes of the existing object-oriented systems are devised. The architecture of automatic tool is designed and implemented for 1) automatic extraction of design patterns and 2) reuse tool for retrieving, editing and rebuilding of design patterns.

Keyword : Component Base Design (CBD) Component abstraction, Reengineering, Design patterns, Rebuilding of design patterns

## 1. 서 론

소프트웨어 기술의 급속한 발전과 호환성 측면의 문제, 그리고 사용자의 다양한 요구사항 변화는 기존 시스템의 유지보수에 많은 기술적, 비용적 지원을 요구함은 물론 그 생명을 점차 축소시키고 있다. 따라서 시스템 성능을 개선시키고 변화하는 환경에 적응하기 위해 독립적이고 추상화된 구성요소와 이들 간의 관련성 정의를 통해 시스템을 구성하고 또 다른 재사용을 유도하는 객체지향 방법론이 필수적이다. 또한 시스템을 실험하고 분석함으로써 정확한 이해 획득과 미래의 유지보수 비용 감소를 가능하게 하고 나아가 재사용 가능한 자원으로 활용할 수 있도록 하는 소프트웨어 역공학 역시 절실히 요구된다. 소프트웨어 역공학은 기존 원시 코드에서 구성 요소와 그들의 관계 파악을 통해 설계 요소를 추출함으로써 논리적이고 구현에 독립된 추상화된 설계 정보를 제공하고 재사용케 함으로써 그 역할이 더욱 중요시 된다. 객체지향 설계는 클래스를 기본 단위로 이들 간의 정적 관계 표현에는 효율적이

* 종신회원 : 대진대학교 컴퓨터공학과 교수
kgb@daejin.ac.kr(제 1저자)
☆ 본 연구는 2003년도 대진대학교 교내연구비에 의해 연구되었음.

지만 동적인 메시지 흐름에 관한 정보 제시에는 한계를 가지고 있다. 그러므로 설계 문제의 추상화와 특정 영역의 일반적인 해결책에 대한 정보 표현 및 그 관계를 효과적으로 나타내기 위해서는 패턴 형식이 적절하다. 설계 패턴은 객체지향 설계의 기본 문제 해결을 위한 보편적인 추상화 표현으로 특정 문제 해결에서 한정적일 수밖에 없는 개인적 경험을 추상화를 통해 다른 사람들과 공유하게 하며 개발자들 간의 의사 교환 도구로 사용될 공통 어휘를 제공한다[1]. 즉, 공개된 설계 쟁점의 해결책으로 객체지향 시스템에 대한 응용 도메인의 이해를 향상시키고 새로운 아키텍쳐를 위한 자원을 제공한다.

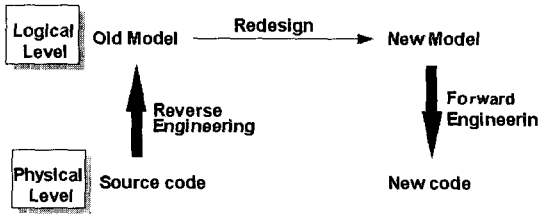
따라서 본 논문은 객체지향 시스템의 기존 원시 코드에서 역공학을 적용하여 정규화되고 패키지화되어 정량적으로 평가되어질 수 있는 활용가능한 설계 패턴 추출을 위한 알고리즘을 정의하고, 설계 패턴 자동 추출을 위한 역공학 도구와 추출 패턴의 재사용 도구를 포함하는 재공학 도구를 설계하여 이를 Java 시스템으로 적용한다.

제 2장에서는 본 논문과 관련된 연구와 용어에 대한 개념을 서술하고 3장에서는 설계 패턴 추출을 위한 알고리즘을 정의한다. 그리고 4장에서는 패턴 추출 및 재사용을 위한 재공학 도구를 설계하고 5장에서는 이를 Java 시스템에 적용시킨 예를 나타낸다. 마지막 6장에서 논문을 결론짓고 향후 연구 방향을 제시한다.

## 2. 관련 연구

### 2.1 소프트웨어 역공학

소프트웨어 재공학의 한 기술 체계인 소프트웨어 역공학은 자동화된 도구를 이용하여 물리적이고 구현 의존적인 시스템 정보를 일반적이고 추상화된 시스템 컴포넌트와 그들 간의 관계를 식별하는 것이다. 이 역공학 과정을 통해서 시스템을 개선하고 구현에 덜 종속적인 고수준의 관점에서 시스템을




(그림 1) 역공학 프로세스의 개념

본질적으로 파악할 수 있어 유지보수 및 재개발(재사용)을 위한 명확한 이해를 제공한다.

그림 1은 기존 원시 코드로부터 새로운 코드의 생성을 위한 소프트웨어 역공학의 개념적 체계를 나타낸 것이다[2]. 역공학은 재공학을 위한 초기 단계로 단지 시스템 분석을 통한 고수준 개념 및 형태의 획득에만 국한되지 않으며 이 과정에서의 생산물들은 현대적인 기술 적용으로 재설계 되어진다.

### 2.2 객체지향 소프트웨어 재사용

객체지향 프로그래밍에서의 원시 코드 재사용은 기본 단위가 클래스가 되며 이것은 문제 영역과 밀접한 관계를 이루고 있는 응용문제의 추상적 표현이므로 유사한 응용 영역에서 미소한 수정을 통해서도 재사용이 가능하다. 재사용 단위를 코드에서 보다 상위 단계의 추상 정보로 확장하여 프레임워크나 설계 패턴을 이용하면 효율적이다. 따라서 유사한 응용의 객체지향 프로그램에서 객체의 구조와 결합이 일정한 틀로서 유지되는 프레임워크를 재사용의 대상으로 하거나 객체들의 역할과 연결이 유사한 경우 자주 사용되는 구조를 일정한 형태로 설정한 설계 패턴을 사용함으로써 높은 효율의 재사용을 도모할 수 있다[3].

### 2.3 설계 패턴

객체지향 설계에서 필요한 클래스와 그것들의 역할과 책임 분산을 하나의 패턴으로 추상화시킴으로써 가치 있는 이전 설계 경험을 재사용할 수 있다. 설계 패턴은 클래스들과 인스턴스들 상의

추상화들을 식별하고 명명하며 정의함으로써 시스템 복잡도를 줄인다. 뿐만 아니라 재사용 컴포넌트 구축을 위해 경험 있는 참여자들에 의해 얻어진 설계 지식을 재사용하기 위한 수단을 정제하고 클래스 계층들의 재조직이나 재요소화의 목표 달성을 용이하게 한다. 설계 패턴에 대한 연구는 소프트웨어 재사용에 대한 다양한 접근 방법 중 하나로 설계 패턴의 생성, 저장, 응용 그리고 효율적인 검색에 초점을 맞추고 있다[4].

### 2.4 역공학에 의한 설계 패턴 추출

역공학은 물리적이고 한정적인 표현 정보들을 보다 일반적이고 추상화되어진 정보로 변화시키며 이후 재사용을 위한 공유 저장소에 적당한 형태로 저장한다. 따라서 전체 시스템의 일반적인 설계 구조를 추출하여 시스템 분석과 설계의 생산성을 향상시키기 위해서는 표준화된 설계 용어 및 컴포넌트 관계 구조를 통해 재사용을 용이하게 하며 품질을 향상시키는 설계 패턴 추출을 위해 역공학이 필요하다. 역공학적 관점에서 설계 패턴은 기존 시스템의 설계 과정에 대해 공유될 수 있는 공통 용어를 형성하고 특정 문제의 해결책 및 응용에 대한 상반 관계를 제시함으로써 기존 설계 이해의 일반성을 획득하고 이를 사용하도록 한다. 또한 구현에 한정적인 객체지향 프레임워크에 대한 문서화 도구로 활용될 수 있어 유지보수성이 높은 유사한 구조의 새로운 시스템 구축시에 크게 도움을 줄 수 있다. 즉, 설계 패턴을 위한 역공학으로 시스템 이해성 증진은 물론 소프트웨어 재사용 효과의 극대화를 객체지향 시스템 (재)개발에 최상으로 적용시킬 수 있다.

### 2.5 관련 시스템[5]

역공학의 적용으로 원시 코드로부터 보다 추상적 정보를 획득하기 위한 기존 연구들은 대개 비객체지향적인 시스템을 고수준 개념의 시스템으로 접근 가능하도록 변형하는 것이다. 따라서 전형적인 절차적 서브시스템을 객체 지향적인 아키텍쳐로 전환에 초점을 둔다.

동일 소프트웨어 시스템 내에서 절차적인 코드와 객체 지향적인 코드의 통합을 위한 원칙은 Dietrich에 의해 제안되었으며 Jacobson은 시스템에서 재공학되어진 부분과 그렇지 않은 부분 사이의 인터페이스를 평가하는 방법을 제시했다. 그리고 Haughton과 Lano는 비객체지향적인 원시 코드로부터 객체를 추출하고 $Z^{++}$를 사용하여 정규적으로 표현되어질 수 있는 객체를 형성하는 프로그램의 추상적인 데이터 타입을 식별했다. 객체지향 아키텍쳐로 소프트웨어를 재구조화하는 일반적인 프레임워크는 Jacobson에 의해 제시되어졌는데 이것은 시스템이 어떻게 점진적으로 현대화되어지는지에 대한 일반적인 규칙을 설명한다. 또한 개념적 데이터 모델과 나아가 도메인 모델을 원시 코드로부터 유도할 수 있으며 이것을 소프트웨어 재공학을 위해 사용할 수 있게 도메인 모델 기반의 재공학이 제안되었다. 그리고 원시 코드로부터 객체 및 객체 특성을 식별하기 위한 접근은 전역 및 연속적인 데이터, 형식 파라미터의 타입과 반환 값에 기초한다. 이 접근은 타입 한정적이지만 객체 식별을 위한 데이터 구조 사이의 기능적 관련성을 고려하였다. COREM(Capsule Oriented Reverse Engineering Method) 프로세스는 응용 도메인 지식을 사용함으로써 절차적인 시스템을 객체지향적인 아키텍쳐로 변형하는 재구조화 프로세스이다. 이것은 역공학 과정에서 입력인 원시코드의 추상화를 향상시키고 순공학 프로세스로 개발되어진 목표 아키텍쳐와 획득된 결과를 연관시킴으로써 더욱 호환성 있는 새로운 아키텍쳐에 의해 시스템을 재생산한다.

## 3. 설계 패턴 추출

### 3.1 기존 시스템으로부터의 설계 패턴 추출

본 논문에서 사용하는 패턴 추출 알고리즘은

(표 1) 본 논문에서 제시한 설계 패턴 추출 단계

| |
|---|
| 1) 역공학 도구를 이용하여 원시 코드에 포함된 시스템의 분석, 설계 정보를 파악한다. |
| 2) 설계 당시의 의도를 파악하여 클래스 간의 역할 및 협동, 역할 분산 관계를 파악한다. |
| 3) 패턴 추출 알고리즘을 적용하여 가능한 패턴을 추출한다. |
| 4) 추출된 패턴을 구축된 패턴과 비교하여 적절성을 평가한다. |
| 5) 임시로 평가된 패턴을 반복적인 정제를 통해 도메인 한정적인 패턴으로 생성한다. |
| 6) 재사용 가능한 설계 패턴을 분류한다. |

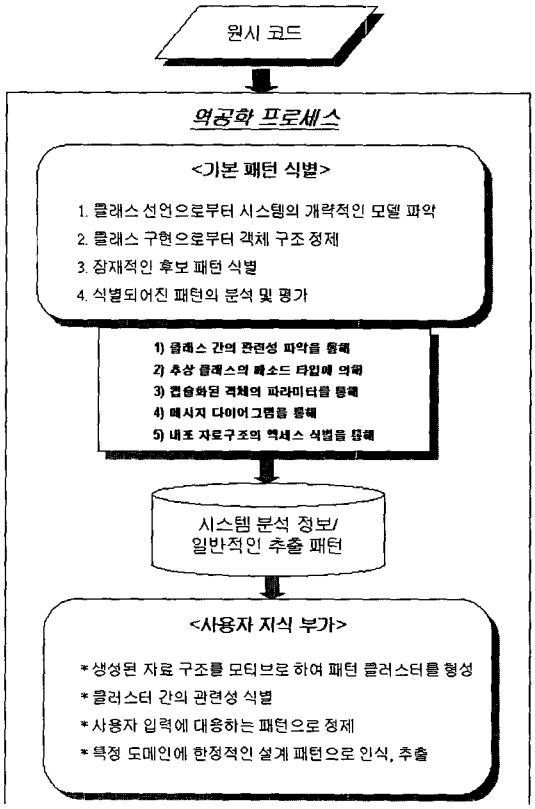
설계자 임의로 유사 패턴으로의 유도에 따른 많은 피드백 비용을 감소시킴은 물론 충분한 분석, 이해 정보를 바탕으로 뚜렷이 확인되어질 수 있는 패턴의 추출 방법이다. 그 과정은 표 1과 그림 2와 같이 개략적으로 요약될 수 있다. 표 1의 1단계에서 4단계는 그림 2의 역공학 프로세스 중 기본 패턴 식별 과정에 속하며 5단계, 6단계는 사용자 지식 추가 부분에 해당된다. 이 방법은 설계 패턴 추출 과정을 통해 자신만의 설계 과정에서 반복적으로 이용될 수 있는 응용 한정적인 패턴들의 지식베이스를 구축하도록 한다. 기존 시스템에서 추출되어진 설계 패턴은 참조 패턴 집합의 의미로서 이미 데이터베이스로 구축되어진 Gamma의 패턴과의 비교를 통해 타당성을 평가받고 사용자 제어 하에서 계속적인 정제 과정의 반복으로 사용자가 구축하고자 하는 응용 도메인에 가장 밀접한 의미적 정보를 포함하는 설계 패턴이 가능하므로 설계 문제의 해결책으로 이용되어질 수 있다[6,7,8,9].

(1) 기본 패턴 식별

이 프로세스는 알고리즘의 단계적인 적용으로 원시 코드에 포함되어진 구조적인 정보와 설계 수준의 정보를 파악하여 본 논문에서 정의한 정보 구조로 저장하고 기본 패턴을 식별한다.

**1) 클래스 선언에서 개략적인 시스템 모델 파악**

; 패턴 요소로서 클래스의 선언 및 클래스간의




(그림 2) 역공학 프로세스의 전체적인 구조

상호작용의 발견으로 개략적인 구조를 인식한다. 특히 포인터 구현에 의한 연관성과 메소드의 반환값의 연결을 중심으로 클래스 사이의 관련성에 초점을 둔다.

**2) 클래스 구현으로부터 객체 구조 정제**

; 관련성에 대한 모든 상세한 정보 획득을 위해 클래스 에트리뷰트와 클래스 메소드에 전달되어진 파라미터, 클래스 타입 등을 조사하여 객체간의 통신을 획득한다. 여기서는 관련성의 종류(즉, has-a, is-a, part-of)를 비롯해 다중성(일대 일, 일대 다, 다대 다)을 확인한다.

**3) 잠재적 후보 패턴 식별**

; 전 단계의 정보를 바탕으로 참조 패턴과의 유사성을 찾음으로써 추출 가능한 패턴을 식별

(표 2) 패턴 식별 메소드

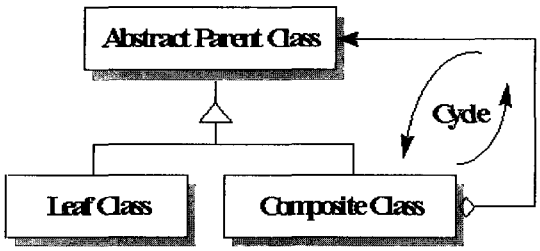
| 적용 메소드 | 추출 가능한 패턴 |
|---|---|
| ① 클래스 간의 관련성<br>i. 상위와 하위 클래스 간의 상속/포함 관계<br>ii. 추상 클래스와 하위 클래스의 분리 관계 | Composite, Decorator Bridge |
| ② 추상 클래스의 메소드 타입에 의한 추출 | Template Method |
| ③ 캡슐화 된 객체의 파라미터를 통한 추출 | Strategy, State, Command |
| ④ 객체 메시지 다이어그램 적용 통한 추출 | Chain of Responsibility |
| ⑤ 연속형 내포 자료의 엑세스 식별로 추출 | Iterator |

한다. 표 2와 같은 매핑적 관계의 추출 메소드를 적용시키면 가능한 잠재적인 패턴이 식별된다.

① 클래스간의 관련성에 의한 추출

i. 상위 및 하위 클래스간의 상속/포함 관계 파악

서브 클래스들은 추상 부모 클래스의 메소드 구현을 위해 독립적으로 재정의 하거나 상위 클래스의 계속적 참조를 위해 상속/포함 관계에 의한 사이클을 형성한다. 후자와 같은 사이클의 발견을 통해 Composite와 Decorator 패턴을 찾을 수 있다. 이들 패턴들은 concrete 개념의 leaf 클래스와 컨테이너 개념의 composite 클래스를 포함하여 프로토콜을 구현하는 추상/수퍼 클래스를 생성한다. 그러므로 상속/조합 그래프에서 표현되어질 사이클 즉, 상속받은 클래스에서 포함 관련성을 찾음으로써 감지할 수 있다(그림 3).



(그림 3) Composite 패턴에서의 사이클

ii. 추상클래스와 하위클래스의 분리 관계 감지

객체지향 시스템에서 공통적인 인터페이스와 각각의 상세한 구현 클래스의 관계를 상속 관계가 아닌 다른 관계로 설정함으로써 변화의 융통성과 재사용성을 도모할 수 있다. 이러한 관계의 파악이 가능하다면 Bridge 패턴의 식별이 가능하다. Bridge 패턴은 구현으로부터 추상화를 분리함으로써 구현과 추상화는 독립적이며 동적으로 변화할 수 있도록 한다. 즉, 추상화의 컴포넌트는 실제 구현을 담당하기 위해 모든 환경에 대해서 정의되어진 peer 객체와 계속적인 연결 관계를 유지하며 실행되어간다.

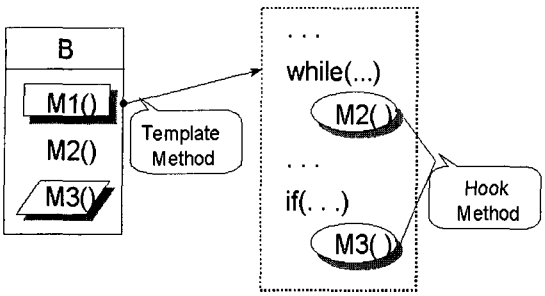
② 추상 클래스의 메소드 타입에 의한 추출

Template 메소드는 객체들 사이에서 관련성이나 추상 행위 혹은 제어의 일반적 흐름을 정의함으로써 알고리즘을 대략적으로 서술하는 것이며, 그 내부에 선언되어진 Hook 메소드들은 서브 클래스에서 오버라이딩되어짐으로써 알고리즘 상에 많은 다양성을 제공한다. 즉, 추상 클래스에서 메소드 타입 각각을 분류함으로써 Template Method 패턴을 추출할 수 있다.

그림 4는 추상 클래스를 구성하는 메소드 타입이다. Template Method 패턴은 추상 클래스 Template 메소드 내에 선언된 Hook 메소드들의 서브 클래스 재정의로 변화되며 전체 알고리즘에는 변화가 없는데 그 이 두 메소드의 구별을 통해 인식할 수 있다.

③ 캡슐화된 객체의 파라미터를 통한 추출

동등 레벨에서 각 객체의 구체화된 역할 책임을 캡슐화하고 추상 부모 클래스의 인터페이스를 사용함으로써 클라이언트 객체가 함수의 파라미터화를 통해 객체를 구체화시키는 패턴들 즉, Strategy, State, Command들은 대표 객체에




(그림 4) Template 메소드와 Hook 메소드

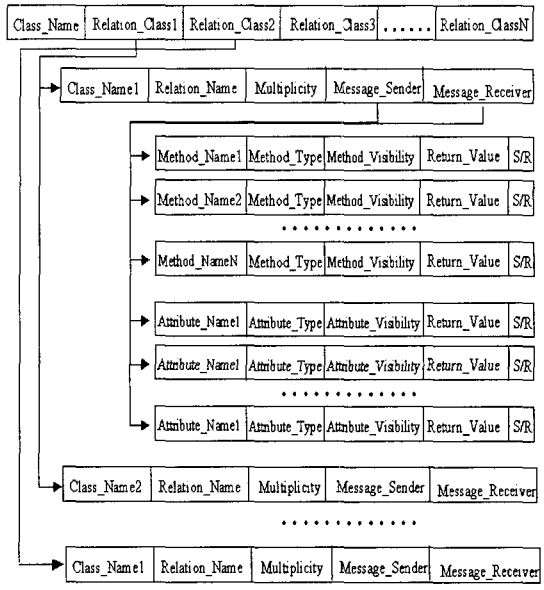
의한 캡슐화된 객체의 참조를 포함한다. 이러한 경우 추상 클래스의 엑세스를 통해 서브 클래스의 캡슐화된 알고리즘 구현을 위임하는 대표 클라이언트 객체 식별을 통해 몇몇 패턴을 식별할 수 있다.

④ 다이어그램을 통한 패턴 추출

패턴 Responsibility의 체인은 Decorator와 Composite의 객체 구조와 결합하여 구성되므로 원시 코드의 구조 분석보다는 객체 메시지 다이어그램을 사용하여 감지하는 것이 용이하다. 이것은 요구를 제어할 기회를 하나 이상의 객체에 부여함으로써 요구의 송신자가 수신자와의 결합을 피하는 패턴으로 객체가 요구를 제어할 때까지 수신 객체를 연결하고 체인을 따라 전달한다. 따라서 객체 메시지 다이어그램에서 동일한 메시지 선택기를 가지는 메시지 전송의 순차적인 시리즈는 역할 연결의 사용으로 인식될 수 있다.

⑤ 연속형 내포 자료구조의 엑세스 객체 식별

논리적 혹은 물리적인 연속적 연결 관계를 유지하는 배열, 리스트 등의 표준 구조체 형식일 경우 실제적 객체 구조 각 요소들을 내부적 형태에 관계없이 접근 참조할 수 있는 방법 제공의 클래스가 필요하다. 즉, 연속적인 요소를 포함하고 있는 구조체를 정의하고 이들 각 요소들을 엑세스하며 참조의 자취를 유지할 수 있는 클래스의 식별은 Iterator 패턴을 감지할 수

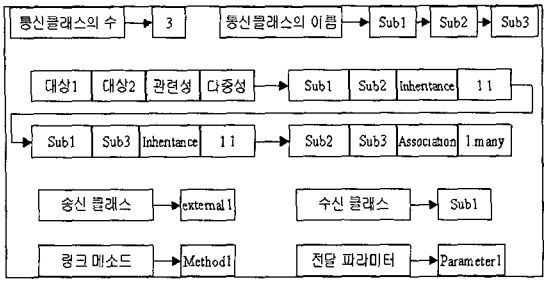



(그림 5) 패턴 라이브러리의 정보 저장 구조

있다. Iterator 패턴을 통해 여러 구조체에 포함된 요소들을 투명한 방식으로 이용할 수 있는 접근 메소드를 정의, 구현하는데 시스템 구성 객체 관계의 효율성 향상에 유용하다.

### 4) 식별되어진 패턴의 분석 및 평가

추출된 패턴의 평가는 시스템 분석/이해 정보를 바탕으로 추출된 설계 패턴과 참조 패턴 집합에 미리 저장되어져 있는 Gamma의 패턴과의 일치성 조사에 의해 수행된다. 여기에 적용되어지는 비교의 관점은 구조적 측면, 기능적 측면 그리고 구현적 측면의 3가지로 구분되는데 이것은 다시 매핑 정도 및 관점에 따라 평가되어지며 이후 패턴 라이브러리 구축시 분류되어진다[9]. 기본 패턴 식별 과정에서 일단 인식되어진 패턴은 참조 패턴과의 비교로 타당성을 확인받은 후 사용자 자신의 의도하에서 반복적으로 도메인 의존적인 패턴으로의 정제를 통해 응용에 적합한 패턴으로 생성되기 위해 라이브러리에 저장된다. 그림 5는 추출 패턴 라이브러리의 정보 저장 구조를 나타낸 것이다.




(그림 6) 패턴 정제를 위한 사용자의 지식 구조

(2) 도메인 패턴 추출

저장된 정보와 식별된 패턴은 사용자 고유의 경험 지식을 부가시킴으로써 도메인 한정적인 패턴으로 식별되어진다. 즉, 추출되어진 일반적인 Gamma의 패턴은 사용자가 가진 지식을 기반으로 한 반복적인 정제로 인해 응용에 적합한 패턴으로 형성될 수 있다.

**1) 패턴 구조 형성**

; 라이브러리에 저장된 정보 구조를 중심으로 관련된 구조의 결합으로 가능한 패턴 구조를 형성한다. 이는 개별 관련성에 대한 집합적 클러스터를 형성한다.

**2) 형성된 패턴 구조의 관련성 식별**

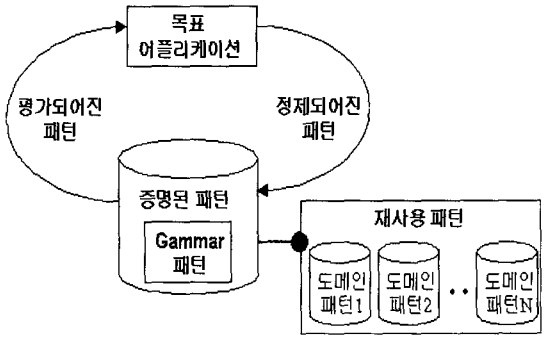
; 독립적으로 구별된 패턴 구조 즉, 클러스터 간의 중복성을 제거하고 클러스터 간을 결합시킴으로써 일반적인 패턴 모델을 구성한다.

**3) 사용자 입력에 따른 패턴으로 정제**

; 사용자는 대화적 방식으로 추출하고자 하는 패턴의 도메인 한정적인 지식을 패턴 정보 구조로부터 요구함으로써 그에 따른 매핑적 결과에 의해 적절한 패턴으로 정제한다. 그림 6은 사용자의 지식 제공을 위한 패턴의 정보 구조이다.

**4) 도메인 한정적인 설계 패턴 추출**

; 정제되어진 패턴은 사용자의 유도 하에 도메인 한정적으로 패키지화된 활용 가능한 설계 패턴으로 참조 패턴에 의해 품질 평가가 어느 정도 이루어지며 사용자가 꼭 필요한 특성 패턴으로 제공된다.



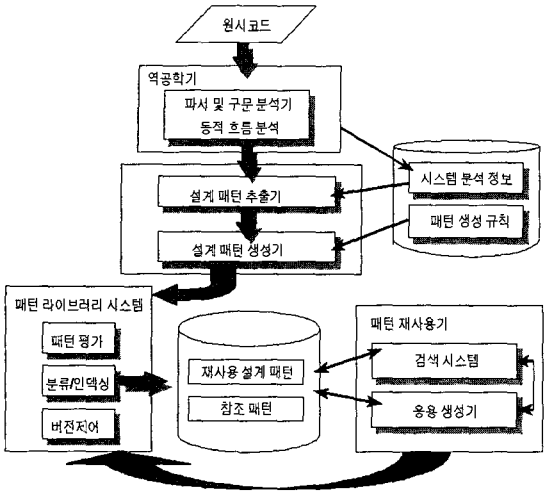

(그림 7) 사용자 패턴 라이브러리 구축

즉, 역공학적 방식에 따라 먼저 추출되어진 설계 정보가 클래스 간의 관련성을 중심으로 한 구조적 특성을 기반으로 패턴의 일반적인 형태를 추출하고 시스템의 분석 정보를 정보 구조상에 저장하였다. 그리고 대화적 방식에 의해 사용자 정보 추가 과정을 거치면서 보다 구체적이며 활용 가능한 설계 패턴으로 식별된다.

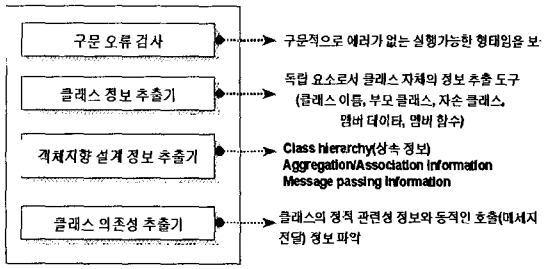
그러므로 추출 과정을 통해 획득되어진 정보들은 최종 사용자의 제공 정보와의 매칭을 통해 도메인에 보다 근접한, 그래서 설계 경험의 구체적 재사용을 지원할 수 있는 패턴으로 생성될 수 있다.

## 3.3 패턴 라이브러리

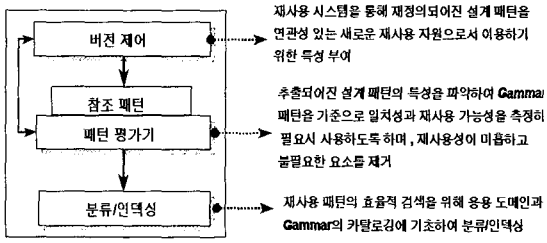
그림 7은 본 논문에서 제안하는 2단계의 패턴 추출 프로세스를 통해 패턴 라이브러리를 구축하는 과정이다. 참조 패턴과의 비교에 의해 패턴 평가가 이루어지며 사용자의 목적 의도에 의해 도메인 한정적인 패턴으로 생성한다. 즉, 검증되어진 패턴 라이브러리는 주어진 도메인에서 응용의 구축을 위해 활용할 수 있도록 이전 경험을 통해 확인된 사용자 자신의 재사용 가능한 설계 패턴 라이브러리를 생성해 준다.




(그림 8) 설계 패턴 추출을 위한 역/재공학 도구




(그림 9) 역공학기



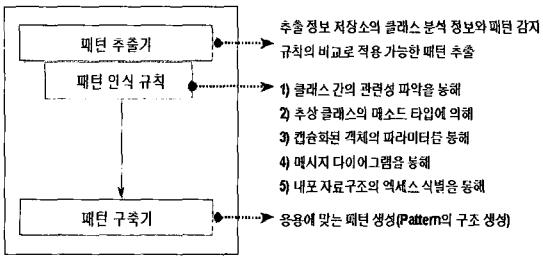

(그림 10) 설계 패턴 추출기

## 4. 설계 패턴 추출 도구의 설계 및 프로토타이핑

기존 시스템에서 역공학의 적용으로 설계 패턴을 추출하기 위한 본 연구의 시스템 구조는 그림 8과 같다. 대상 시스템의 원시 코드는 역공학기를 통해 개념적이며 논리적인 데이터 구조가 추출되며 이것은 추상화된 일반적인 클래스로 변환된다. 이 과정에서 구문 및 동적 흐름 분석 정보와 각 인스턴스 간의 호출 관계에 의한 기능성 종속성 파악이 필요하다. 이렇게 형성된 클래스들은 저장소에 저장된다. 다음 단계는 추출된 클래스 정보를 파악하여 가능한 패턴들을 추출, 생성하며 적절한 패턴이 생산되어졌는지를 평가하는 과정이다. 세 번째 단계는 새로운 응용을 위한 순공학으로의 재구축 단계로 형성된 설계 패턴은 가치 있는 재사용 대상으로 검색 시스템을 통해 필요시 검색되어 응용 생성기를 통해 새로운 시스템 구축을 위해 활용되며 마지막 단계로 라이브러리화 되어진 설계 패턴을 패턴 정보의 획득을 통한 검색과 응용 구축에 필요한 패턴의 재정의 및 응용 구축에 검색된 패턴을 사용한다.

1) 역공학기 : 기존 시스템의 원시 코드로부터 설계 정보를 추출하는 시스템으로 구조적이고 논리적인 데이터 구조와 클래스간의 상호 관련성, 동적인 호출 관계를 파악하여 저장소에 저장한다. 이것은 그림 9에서 보는 것과 같이 구문적 오류가 없는 코드임을 보증하기 위한 전처리기인 구문 오류 검사기와 객체지향 시스템에서 단일 클래스 및 클래스 간의 상호 관련성 정보 파악을 위한 클래스 정보 추출기, 객체지향 설계정보 추출기 그리고 클래스 간의 호출정보 획득을 위한 클래스 기능 종속성 추출기로 구성된다.

2) 설계 패턴 추출기 : 분석되어진 시스템의 분석 정보에서 설계 당시의 의도를 파악하여 클래스와 인스턴스 그리고 그것들의 역할과 협동 관계 및 책임 분산을 식별, 추상화함으로써 사용되어진 설계 패턴을 감지, 추출한다. 이것은 패턴 형식으로 구성하여 평가한 후 사용자에게 제시된다. 여기서 사용되는 추출 규칙은 3장에서 정의되어진 추출 알고리즘으로 그림 10과 같이 요약된다. 즉, 추출 정보 저장소의 클래스 분석 정보를 바탕으로

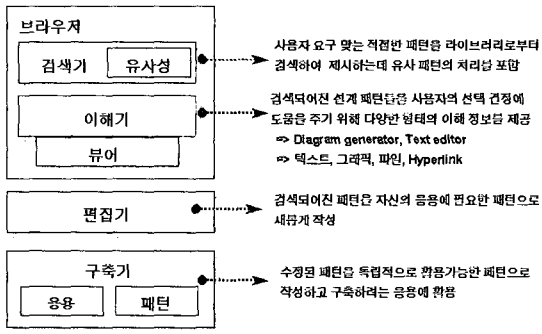



(그림 11) 설계 패턴 라이브러리

패턴 추출 규칙 적용을 통해 가능한 설계 패턴을 추출하며 식별되어진 패턴은 참조 패턴인 Gamma의 패턴을 기준으로 하여 구조화되어진다.

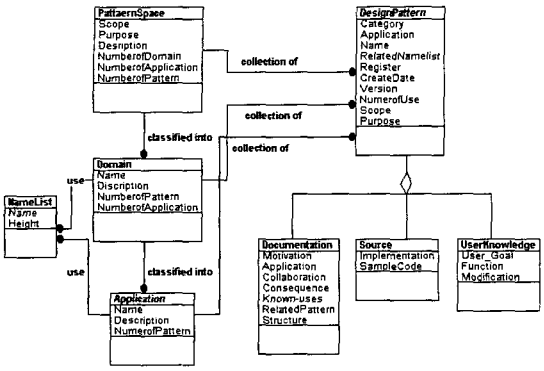
3) 패턴 라이브러리 시스템 : 추출되어진 설계 패턴들은 이미 그 유용성을 충분히 인정받은 참조 패턴의 집합과 매칭 관계를 분석함으로써 이후에 가치 있는 재사용 자원으로 사용될 수 있는지 평가 한 후 타당한 패턴들만 각 도메인별로 분류하고 검색의 인덱스를 부여한다. 추출기를 통해 일단 인식된 패턴은 참조 패턴과의 비교로 타당성을 확인받으며 또한 사용자는 자신의 의도하에서 반복적으로 도메인 의존적인 패턴의 정제를 통해 응용에 적합한 패턴 라이브러리를 생성한다. 이런 과정을 거친 후 재사용 시스템을 통해 검색되어 수정되어진 패턴은 버전관리 과정을 거쳐 기존 패턴과의 차별성을 부여받은 후 재사용 설계 패턴의 라이브러리에 저장한다. 세부 도구들은 재사용 시스템을 통해 수정되어진 패턴의 연계성을 유지하고 독립된 요소로 관리하기 위한 버전 제어기와 재사용 잠재성을 평가하는 품질 평가 도구 그리고 검색시의 효율성 증진을 위한 분류/인덱싱 도구가 있다. 이에 대한 세부 구조는 그림 11에 나타난다.

4) 패턴 재사용기 : 생성된 설계 패턴들은 재사용의 단위로서 저장소에 저장되어지며 이것은

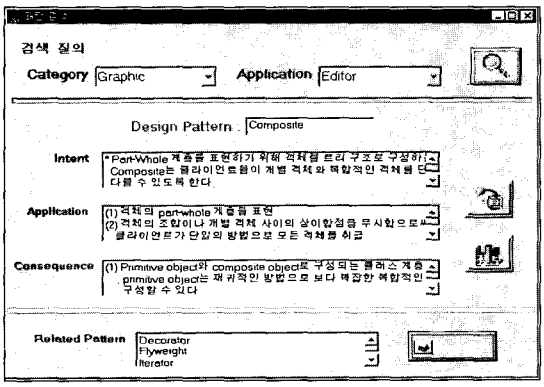



(그림 12) 설계 패턴 재사용기

새로운 시스템 구축을 위해 검색되어지고 응용 생성기를 통해 조합되어져 또 다른 시스템 구축에 이용된다. 이 시스템은 그림 12와 같이 클래스 기반 재사용 시스템과 유사하게 사용자 요구를 수용하여 가장 적당한 패턴을 반환하는 검색기와 재사용 패턴의 자세한 정보 확보를 통한 최상의 패턴 선택을 위한 이해기 그리고 사용자 자신의 필요에 맞는 컴포넌트로 수정하고 새롭게 구축하는 편집기와 구축기로 이루어진다. 패턴 재사용기는 이미 구축한 클래스 기반의 재사용 시스템인 MT-Views[10]의 확장 개념으로 유사한 아키텍쳐를 가진다. 즉, 식별, 분류되어진 패턴 라이브러리를 바탕으로 사용자가 원하는 사양을 받아들여 라이브러리에서 매칭되는 패턴을 사용자의 관여하에 쉽게 검색을 유도하는 브라우저와 제시된 정보 중 사용자가 선택한 정보를 사용자 자신이 적절히 수정하는 편집기 그리고 편집된 패턴을 자신의 응용 영역에 부합되도록 재구축하여 새로운 응용을 위해 사용하는 재구축기로 구성된다. 특히 설계 패턴은 구조적인 클래스 관련성의 추상적 모델링 정보가 매우 중요하므로 각 패턴의 관련성 표현을 위해 기존 모델링 도구를 포함하여 활용한다. 패턴의 분류는 Gamma의 카타로깅을 기초로 라이브러리화 된다. 즉, Gamma의 설계 패턴을 목적과 사용 영역에 따라 스키마를 정의하고 도메인별로 카탈로깅한 데이터베이스를



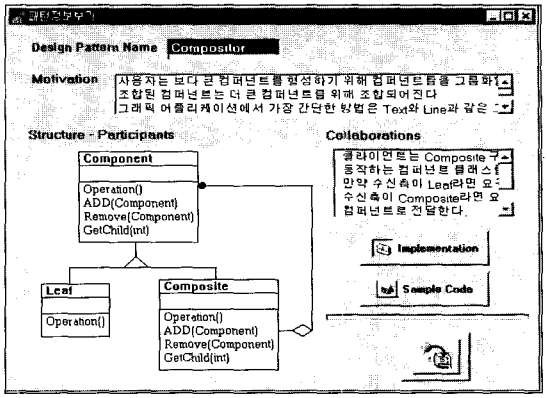

(그림 13) 패턴 라이브러리의 데이터베이스 스키마




(그림 14) 설계 패턴의 검색 인터페이스

구축한다. 여기에 적용되어진 도메인은 패턴이 추출된 원시 시스템으로 일반적이고 추상적인 설계 패턴들을 실제로 작업 가능하도록 응용에 한정적인 성격을 부여한다. 그림 13은 패턴 재사용 시스템을 위한 패턴 라이브러리의 스키마를 OMT의 객체 다이어그램으로 나타낸 것이다. 여기서 상위 클래스 Catalog와 Application은 원시 시스템으로부터의 패턴 추출을 통해 획득되어진 도메인 정보이며 Design Pattern은 검색 대상으로 Gamma에 의해 표현되어진 정보를 중심으로 구성된다.

그림 14는 추출된 설계 패턴 라이브러리를 재사용하기 위한 검색 인터페이스이다. Gamma의 일반화된 설계 패턴을 보다 도메인 한정적인 구분을 통해 정확한 재사용성을 확장시키기 위해서 검색




(그림 15) 설계 패턴의 이해 정보

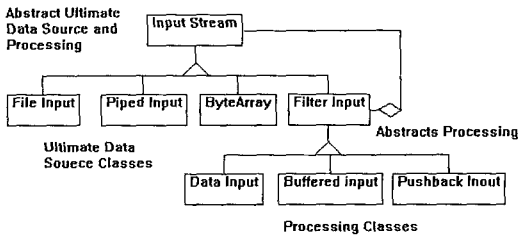
질의는 역공학을 통해 추출된 패턴의 도메인과 그 도메인 내에서 추출 패턴으로 구축되는 기능적 서브시스템으로 구성된다. 즉, Gamma에 의해 동일하게 분류된 패턴일지라도 그 적용 영역에 따라 상세함의 차이가 발생함으로 정확한 재사용 대상의 범위가 선택되어질 필요가 있다. 검색 인터페이스는 일차적인 기능성 정보인 Intent와 Application, Consequence를 제공함으로써 사용자의 재사용성 결정을 유도한다. 또 타당성 있는 패턴 선택을 위해 패턴의 상세한 정보를 제시한다(그림 15).

## 5. 패턴 추출의 적용 예

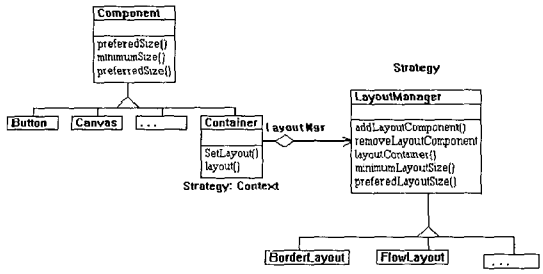
본 논문에서는 대상 시스템을 Java로 하여 역공학적 의미에서 시스템 설계시 적용 가능한 설계 패턴을 위에서 제시한 알고리즘을 통해 찾아보았다. 이를 통해 Java의 설계 구조를 파악하여 재사용 가능한 요소들을 식별하고 자동화 도구로의 실제 구현을 위한 타당성을 검증하였다[11,12,13].

### (1) Compositor, Decorator 패턴

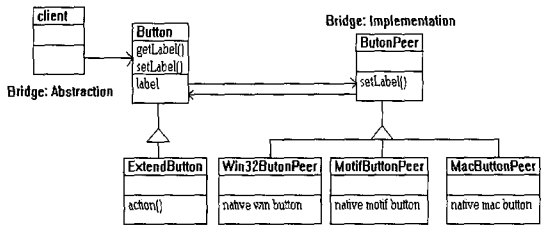
Composite 패턴의 기본 구조는 부모와 자식 클래스 사이의 part-whole 관련성을 요구한다. 이 aggregation 관련성을 composite 자식 클래스 내에서 부모 클래스까지의 참조로써 구현되어지므로 composite 패턴의 감지는 클래스 계층에서 이 루프의 감지에




(그림 16) Java의 입력 클래스




(그림 17) STRATEGY 패턴을 사용하는 AWT에서의 구현




(그림 18) Bridge 패턴을 사용하는 AWT 구현 예

달려있다. 즉, aggregation은 aggregate 객체 내부에 완벽하게 캡슐화되어진 컴포넌트 객체인 경우 물리적 containment 프로그램 실행시 초기화되어진 컴포넌트 타입의 참조/포인터로 구현되어질 수 있다. 따라서 aggregation은 다른 사용자가 정의한 클래스 타입인 클래스의 데이터 멤버를 조사함으로써 감지할 수 있다. 자바 I/O 패키지는 다양하고 정규적으로 순서화된 입력, 출력 그리고 프로세싱 알고리즘을 추상화한다. 자바 프로그램의 기본적으로 데이터 입력과 출력에 관계없이 독립적으로 InputStream과 OutputStream을 조작할 수 있다. 그러므로 이 경우 Compositor 및 Decorator 패턴이 적용 가능하다. 이 패턴은 동적으로 전체 클래스가 아닌 개별적인 객체에 부가적인 역할을 추가하는 것으로 기능성 확장시 서브 클래스가 유동적으로 대처할 수 있다.

그림 16은 자바의 입력 클래스에 Decorator 패턴이 적용된 것으로 서브 FilterInput가 루트 클래스 InputStream까지의 원시정보 참조를 보여준다.

### (2) Strategy 패턴

AWT(Abstract Window Toolkit)는 윈도우, 대화상자, 체크 박스, 리스트, 메뉴, 버튼, 패널 등의 사용자 인터페이스를 관리하는 클래스와 인터페이스를 제공함으로써 플랫폼에 대한 컴포넌트 독립성과 이에 따른 위치 지정 문제를 해결한다. LayoutManager는 모든 Layout 정책을 위한 공통인터페이스를 정의하며 Container는 LayoutManager에 의해서 정의된 인터페이스에만 독립적이고 이 인터페이스는 구현하는 모든 Layout들과 같이 동동한다. 여기서 Strategy 패턴을 식별할 수 있다. Strategy는 StrategyContext, 즉 Container를 조작하고 StrategyContext는 클라이언트 요청을 Strategy인 LayoutManager로 전송하며 클라이언트들은 StrategyContext와의 상호작용만으로 실제 구현 알고리즘을 수행한다. 그림 17은 AWT 내에서의 Strategy 패턴의 적용 예이다.

### (3) Bridge 패턴

자바 API는 플랫폼에는 독립적인 반면 내부적으로는 플랫폼 한정적인 위젯을 사용하는데 이는 Bridge 패턴을 이용할 수 있다. Bridge 패턴은 구현으로부터 추상화를 분리함으로써 분리된 2개의 추상화는 구현에 독립적이며 동적으로 변화할 수 있다. 컴포넌트 객체들은 "Peer" 객체를 가지며 이것에 대한 인터페이스는 Peer 인터페이스에 의해 정의되어지고 한정적인 위젯에 연결 관계를 차례로 제공한다. Bridge 패턴에서 Peer는 구현에, 컴포넌트는 추상화에 대응되며 Peer 인터페이스는 지원된 모든 플랫폼에 대해 구현되어야 한다. 그림 18은 AWT에서 Bridge 패턴의 사용을 나타낸 것이다.

## 6. 결 론

본 논문은 기존 시스템에서 설계 패턴 추출을 위한 역공학 도구와 추출된 패턴의 재사용을 위한 자동화 도구에 관한 연구 중 초기 단계로 기존 코드에서 설계 패턴 추출을 위한 역공학 적용의 타당성과 패턴들의 기본적인 속성을 파악하여 몇몇 패턴들의 자동 추출을 위한 알고리즘을 살펴보고 이를 Java 시스템에 적용시켜 보았다. 그리고 실제적인 자동 도구로 구현하기 위해 패턴 추출 역공학 도구 및 패턴 재사용 도구를 포함하는 재공학 도구의 아키텍쳐를 제시했다. 역공학 프로세스는 원시 코드에 포함되어진 구조적인 정보는 물론 설계 수준의 정보를 파악하여 본 논문에서 정의한 정보 구조로 저장하고 기본 패턴을 식별하는 기본 패턴 식별 과정과 사용자 고유의 경험 지식을 부가시킴으로써 도메인 한정적인 패턴으로 식별하는 도메인 패턴 추출 과정으로 구분된다.

Composite 패턴은 상속받은 클래스에서 포함 관련성을 찾음으로써, Template Method 패턴을 감지하는 것은 추상 클래스에서 메소드 타입 각각의 분류를 통해 그리고 Chain of Responsibility 패턴은 동일한 메시지 선택기를 가지는 메시지 전송의 순차적인 연결을 발견함으로써 추출이 가능하다. 또한 Strategy 패턴은 캡슐화된 각 객체들에 대한 대표 객체의 파라미터로 구분할 수 있으며 Iterator 패턴은 연속적인 자료 구조 엑세스를 위한 객체를 식별함으로써 찾을 수 있었다.

설계 패턴 추출을 위한 시스템은 클래스 자체 정보와 객체지향 설계시 필요한 클래스 간의 관련성(inheritance, association, aggregation, 메시지 패싱) 정보를 추출하는 역공학 도구와 적절한 패턴을 감지, 생성하는 패턴 생성기, 그리고 재사용 자원으로서 타당한 설계 패턴을 저장, 분류하는 패턴 라이브러리 시스템과 패턴을 검색하고 편집하며 새로운 응용 구축에 활용하도록 하는 패턴 재사용 시스템으로 구성된다. 여기서 추출되어진 패턴의 적합성을 평가하기 위해 이미 그 유용성이 널리 인정된 Gamma가 정의한 패턴을 참조패턴으로 하여 일치성을 조사하며 사용자는 자신의 의도하에서 반복적으로 도메인 의존적인 패턴의 정제를 통해 응용에 적합한 패턴 라이브러리를 생성한다. 역공학을 통한 설계 패턴 추출과 이를 재사용하는 자동화 도구를 통해 시스템 이해성 증진은 물론 소프트웨어 재사용 효과의 극대화를 객체지향 시스템 (재)개발에 최상으로 적용시킬 수 있을 것이라 기대된다.

현재, 추출 대상 시스템의 선정과 자동화 지원 도구 활용의 최대화, 추출 패턴의 적절한 평가 요소의 설정 등이 실제 구현을 위해 요구되어지는 현안이다. 그리고 패턴을 패키지화하고 저장, 관리하고 추론하며 검색하기 위해 지식 기반의 소프트웨어공학 데이터베이스를 포함하는 패턴 식별 알고리즘의 정의와 이를 통한 자동화된 역/재공학 도구 구축 및 재사용 시스템의 프레임워크로의 조합으로의 연구가 필요하다.

## 참고문헌


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

[2] Carma McClure, The Three Rs Of Software Automation : Re-Engineering, Repository, Reusability, Prentice Hall, 1992.

[3] Hafedh Mili, et al, "Reusing Software : Issue and Research Directions", IEEE Trans, On SE, Vol. 21, N0. 6, pp. 528~562, 1995.

[4] Ted Lewins et al. "Object-Oriented Application Frameworks", Manning Publications Co, 1995.

[5] Harald C. Gall et al., "Object-Oriented Re-Architecturing", ESEC '95, 1995.

[6] 김 행곤 외, "자바 프로그램 개발을 위한 설계 패턴에 대한 연구", 한국정보처리학회 춘계 학술발표회, pp. 520~523, 1997, 4.

[7] 김 행곤 외, "설계 패턴 자동 추출을 위한 역공학에 관한 연구", 한국정보과학회 추계 학술발표회, pp. 868~872, 1997, 10.

[8] Wolfgang Free, Design Patterns for Object-oriented Software Revelopment, Addison-Wesley, 1995.

[9] Forrest Shull, et al, "An Inductive Method For Discovering Design Patterns From Object-Oriented Software Systems", Technical Report Universuty of MaryLand, Computer Science Department, College Park, MD 20742 USA, Oct., 1996.

[10] 김 행곤 외, "프로토타이핑 지원 재사용 시스템 개발에 관한 연구", 최종보고서, 한국전자통신연구원, 1997.

[11] Iseult White, "Patterns and Java Class Libraries", Java Report, pp. 41~45, Oct., 1996.

[12] Kyle Brown, "Design Reverse-Engineering and Automated Design Pattern Detrction in Smalltalk", http://www2.ncsu.edu/eos/info/tasug/kbrown/thesys2.htm.

[13] Harald C. Gall et al., "Application Patterns in Re-engineering: Identifying and using Reusable Concepts, IMPU '96, Vol. 3, pp. 1099~1106, July, 1996.

[14] 김국보 외, "러프집합에 기반한 불완전 정보의 정보 이론적 척도에 관한 연구", 한국멀티미디어학회 논문지, 제3권, 제5호, pp. 550~556, 2000, 10.

[15] 김국보 외, "Design of the Integrated Incomplete Information Processing System based on Rough Set", 한국퍼지 및 지능시스템학회 논문지, 제11권, 제5호, pp. 441~447, 2001, 10.

[16] Guk-Boh Kim et al., "A Study on Component Customization Concept for Component-Based Reuse Environment", JECS, Vol. 4, No. 2, pp. 13~20, Winter, 2002.

## ◐ 저 자 소 개 ◑

**김 국 보**

1984년 서울산업대학교 전자계산학과 졸업(학사)
1986년 연세대학교 공학대학원 전자계산학과 졸업(석사)
1997년 대구가톨릭대학교 전산통계학과 졸업(박사)
1988년~1990년 해군 중앙전산소장
1990년~1993년 부경대학교 교수
1993년~현재 : 대진대학교 컴퓨터공학과 교수
관심분야 : 소프트웨어공학, 시스템 분석 및 설계, e-Biz 시스템
E-mail : kgb@daejin.ac.kr