디자인 패턴지향 소프트웨어 개발 지원 도구
(Tool for Supporting Design Pattern-Oriented Software Development)

김 운 용 † 최 영 근 ††
(Woon-Yong Kim) (Young-Keun Choi)

요 약 디자인 패턴은 과거에 잘 정의된 설계정보를 활용하기 위한 목적으로 사용되어진다. 이러한 디자인 패턴의 활용은 객체지향 패러다임에서 재사용성과 개발시간의 단축 그리고 소프트웨어 품질의 향상을 가져온다. 그러나 이러한 디자인 패턴의 광범위한 활용에도 불구하고 대부분의 디자인 패턴 정보는 수작업에 의해 활용됨으로써 일관성이 없고 활용능력이 떨어진다. 또한 설계자에 의해 적용된 디자인 패턴정보는 소프트웨어에서 나타나지 않기 때문에 디자인 패턴에 대한 추적성에 대한 문제를 가진다. 이에 본 논문에서는 디자인 패턴지향 소프트웨어 개발 지원 도구를 제시한다. 이 시스템은 디자인 패턴의 관리와 소프트웨어 설계 및 자동화 소스코드 생성기능을 지원한다. 디자인 패턴 관리 기능은 존재하는 디자인 패턴을 저장관리 및 분석과 새로운 디자인 패턴 등록할 수 있는 기능을 담고 있으며, 소프트웨어 설계 기능은 UML 형태의 소프트웨어 설계기능과 디자인 패턴요소의 자동생성기능을 지원한다. 또한 이러한 설계정보를 이용한 소스코드 자동생성기능을 지원하는 소스코드 관리 기능을 가진다. 그 결과 기존의 CASE 도구에서 제시하지 못한 디자인 패턴요소의 추적성을 설계정보에 포함시킴으로서 소프트웨어 분석의 용이성을 제공하고 디자인 패턴 관리와 자동 소스코드 생성기능의 제공을 통해 보다 안정되고 효율적인 시스템을 구축할 수 있다.

키워드 : 디자인패턴, 패턴지향분석설계, 소스코드자동화, 객체지향프로그래밍

Abstract Design patterns are used to utilize well-defined design information. As using these design patterns, we can get re-use in object-oriented paradigm, decrease the time of development and improvement the quality of software. Although these design patterns are widely used among practice, most of design patterns information is manually used, inconsistent and its utilization could be very low. Because the design patterns information that a designer applies does not appear in software, it is sometimes difficult to track them. In this paper, we propose a tool support for design pattern-oriented software development. This tool supports design pattern management, software design and automatic source code generation. The design pattern management has the function for storing, managing and analyzing the existing design pattern and registering new design pattern. The software design has the function for software design with UML and automatically generate design pattern elements. By using this design information, this system can automatically generate source code. In the result to include the tracking design pattern element that is not included in the existing CASE tools into design information, we can build the stable and efficient system that provides to analyse software, manage design pattern and automatically generate source code.

Key words : Design patterns, pattern-oriented analysis and design, automatic source code generation, object-oriented programming.

· 본 연구는 2002년 광운대학교 연구비 지원에 의해 수행되었음.
† 정 회 원 : 광운대학교 컴퓨터과학과
wykim@cs.kwangwoon.ac.kr
†† 비 회 원 : 광운대학교 컴퓨터과학과 교수
ygchoi@cs.kwangwoon.ac.kr
논문접수 : 2002년 1월 10일
심사완료 : 2002년 6월 11일

1. 서 론

객체 지향 소프트웨어 설계 및 구현의 최대 목표는 효율적인 재사용을 통한 개발시간의 단축 및 소프트웨어 품질의 향상과 유지보수성의 증대라 할 수 있다. 이러한 목적을 달성하기 위한 한 가지 방법으로 디자인

패턴이 있다. 이 디자인 패턴은 소프트웨어 개발 시 자주 발생되는 문제점에 대한 해결책을 표현한다[5]. 현재 패턴학회를 통해 다양한 패턴들이 소개되고 있다. 이중 객체 지향 시스템 개발에 유용한 디자인 패턴으로 GoF[5]패턴이 있다. 이러한 디자인 패턴에 대한 연구는 첫째 이들 패턴을 분류하고 표현하기 위한 방법으로 Gamma[5], Tichy[17],Buschman[8]등에 의해 이루어지고 있으며, 둘째 객체 지향 설계방법에 디자인 패턴을 포함하는 패턴 지향 설계방법에 대한 연구로 Sherif[16], Yacoub[13]에 의해 소개되고 있다. 셋째 이들 디자인 패턴의 응용에 대한 연구로 디자인 패턴을 컴포넌트 통합에 어떻게 활용할 것인가에 대한 연구[12][18]와 어플리케이션 개발에 소스코드 생성 및 지원을 위한 연구[1][2][3][6][13][14]등이 진행되고 있다.

디자인 패턴의 정보는 어플리케이션 개발 시 구체화되어야 한다. 그러나 이러한 디자인 패턴은 문제에 대한 추상적인 정보를 기술하고 있기 때문에 어플리케이션에 구체화될 때 개발자의 의도에 따라 다양한 형태로 변형되어진다. 또한 적용된 디자인 패턴은 더 이상 어플리케이션 내부에서 표현되지 않기 때문에 이들에 대한 분석 및 추적이 어렵다. 이에 본 연구에서는 객체지향 소프트웨어 설계 및 구현 시 디자인 패턴을 적용하고 활용하기 위한 설계 및 구현 지원 도구를 제시한다. 이 도구는 크게 3부분으로 나누어진다. 첫 번째는 디자인 패턴을 관리하고 분석할 수 있는 도구로 객체 지향 시스템 개발에 유용한 GoF패턴[5] 형태를 이용한다. 이 디자인 패턴 관리 및 분석도구는 디자인 패턴의 등록 및 관리 기능을 담고 있으며 등록된 디자인 패턴정보는 소프트웨어 설계 시 디자인 패턴 정보 자동 생성에 활용되어진다. 두 번째는 소프트웨어 설계 지원도구이다. 이 도구는 UML 기반의 설계 환경을 제공한다. 이 도구를 이용해 기존의 CASE 도구에서 제시하지 못한 디자인 패턴요소를 표현하고 자동 생성하는 기능을 가진다. 마지막으로 소스코드 관리 기능이 있다. 이 소스코드 관리기능은 소프트웨어 설계 기능을 통해 생성된 설계 정보를 이용해 소스코드를 자동 생성하는 역할을 담당한다. 추가적으로 이 자동 생성된 소스코드는 설계정보에 포함된 디자인 패턴에 대한 기능을 담고 있다. 또한 이러한 모든 정보 즉 디자인 패턴에 대한 정보와 소프트웨어 설계 정보는 XML형태의 정보로 저장 관리함으로서 다양한 XML기술에 활용될 수 있다. 이 도구를 통해 어플리케이션 내부에 구체화되어질 디자인 패턴 정보는 자동화 형태로 구체화될 수 있으며 적용된 디자인 패턴에 대한 분석 및 활용의 효율성을 증대시킬 수 있다.

본 논문의 구성은 다음과 같다. 2장 관련연구에서 객체 지향 소프트웨어 개발에 활용되는 디자인 패턴과 패턴 지향 개발 환경을 소개하고, 3장에서 패턴 지향 소프트웨어 개발 지원 도구를 소개한다. 4장에서 제시된 도구를 활용한 소프트웨어 개발 방법을 보이고, 5장에서는 기존의 연구와 비교 분석한다. 마지막으로 6장에서 결론을 제시한다.

2. 관련 연구

2.1 객체 지향 디자인 패턴

객체 지향 소프트웨어 구성은 여러 개의 작은 구조들의 집합으로 구성될 수 있다[7]. 이 작은 구조를 객체 지향 디자인 패턴이라 한다. 이 디자인 패턴은 디자인 문제에 대한 추상적인 해결방법을 제시하고 있으며 이 객체 지향 소프트웨어에 필요한 디자인 패턴요소를 기술하는 패턴으로 GoF패턴[5] 이 있다. 이 디자인 패턴의 내용은 패턴의 일반적인 구조, 역할, 상호관계 그리고 제약사항과 관련된 내용을 담고 있다. 이러한 디자인 패턴 은 (표1)에서 보인다.

표 1 객체지향 디자인 패턴 요소[5]

		Purpose		
		Creational	Structural	Behavioral
Scope	Class	Factory Method	Adapter(class)	Interpreter Template Method
	Object	Abstract Factory Builder Prototype Singleton	Adapter(object) Bridge Composite Decorator Façade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy *Visitor*

2.2 패턴의 분류

패턴들의 검색 및 관리의 효율성을 증가시키기 위한 패턴 분류법들에 대한 다양한 연구도 이루어지고 있다[5,8,17]. 먼저 Gamma[5]는 소프트웨어 디자인 패턴을 생성(creational), 구조(structural) 그리고 행위(behavioral)패턴으로 분류하고 있으며, Buschmann[8]은 패턴을 구조(architectural) 패턴, 디자인(design)패턴, 그리고 코드(idioms)패턴으로 분류하고 있다. 구조(architectural) 패턴은 소프트웨어의 뼈대를 형성하는 패턴에 대한 부분을 정의하고 있고, 디자인(design)패턴은 각 컴포넌트를 형성하는 패턴을 정의하고, 코드(idioms)패턴은 프로그램 언어를 이용한 코드 명세에 대한 패턴을 정의하고 있다. 또한 Tichy[17]의 표현에서는 패턴의 범주를 더 다양하게 분류하고 있다. 즉 어

떤 목적의 패턴들인가에 대해 Decoupling, Variant Management, State Handling, Control, Virtual Machines, Convenience Patterns, Compound Patterns, Concurrency, Distribution 과 같은 범주로 분류하고 있다. 이러한 분류방법들은 존재하는 패턴들을 좀더 효율적으로 구성하기 위한 시도이다.

2.3 패턴지향 분석 설계

패턴지향 분석 설계는 객체지향 분석 설계에 디자인 패턴 요소를 추가시켜 구성하는 형태이다. 이러한 형태의 패턴지향 분석설계방법은 Sherif[16]에서 제시되고 있다. 이 방법에서 디자인 패턴은 요구 분석단계에서 어플리케이션의 요구를 통해 사용 가능한 패턴을 식별하고, 이들을 설계단계에서 객체들과 함께 표현함으로써 설계 시 패턴정보를 유지관리 하기 위한 목적으로 이용된다. 일반적인 패턴지향 소프트웨어 개발과정은 (그림 1)과 같이 표현될 수 있다.

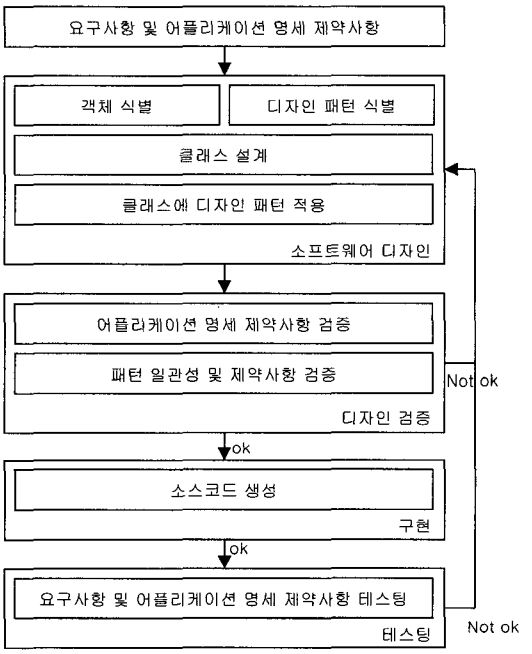

그림 1 패턴 지향 어플리케이션 개발 단계

(그림 1)은 패턴 지향 어플리케이션 개발의 개괄적인 단계를 보여준다. 이 과정을 살펴보면, 먼저 요구사항과 어플리케이션 명세 제약사항을 통해 필요한 객체와 사용 가능한 디자인 패턴을 식별한다. 이렇게 식별된 내용은 클래스의 설계에 이용되고 클래스들 간의 디자인 패턴요소가 추가된다. 이렇게 구성된 설계는 어플리케이션 명세와 디자인패턴의 일관성을 검증하여 정확하다면 패턴 기능을 포함한 소스코드를 생성한다. 이렇게 생성된 소스코드는 요구사항과 어플리케이션 명세 제약사항을 테스팅 함으로써 정확성을 검증한다. 본 연구는 이러한 패턴 지향 개발 환경에서 이루어지는 단계를 지원하는 도구를 제시한다.

3. 디자인 패턴지향 소프트웨어 개발 지원 도구

3.1 지원도구의 전체 시스템 구성 도

이 지원도구는 (그림 2)에서 개괄적으로 묘사하고 있다. 크게 3부분으로 구성된 시스템은 각 구성요소가 XML 문서형태의 저장관리를 통해 서로 정보를 활용함으로써 디자인 패턴요소와 설계정보 그리고 소스코드간의 일관된 형태의 구성이 가능하다.

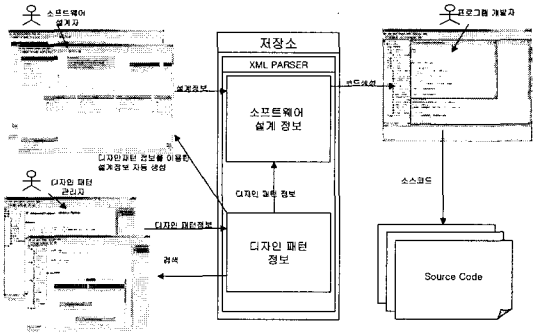

그림 2 디자인 패턴 지향 소프트웨어 개발 지원 도구

3.2 구현 환경

제시된 도구의 구현 환경은 플랫폼 독립적인 특징을 제공하기위해 JAVA를 이용해 구현되었다. 이 도구의 구현기술과 환경을 정리하면 다음과 같다.

- 운영체제 : JVM , Windows
- 적용기술 : XML, UML
- 사용언어 : JAVA 1.2 , Xerces(XML 파서)

현재 이 도구를 이용한 소프트웨어 설계 와 소스코드 생성은 JAVA 언어를 기준으로 하고 있다. 그러나 설계정보는 코드 변환 모듈의 추가를 통해 다른 객체지향 언어로 확장될 수 있다.

3.3 디자인 패턴 관리자

디자인 패턴 관리자는 GoF패턴의 요소를 담고 있으며 추가적으로 사용자의 패턴을 등록 수정 할 수 있고, 소프트웨어 설계시 자동적으로 적용시킬 수 있는 구조

를 가진다. 이 디자인 패턴 관리자는 3부분으로 나누어진다. 첫 번째는 디자인 패턴의 분석 및 시각화를 위한 부분으로 입력된 자료를 쉽게 분석할 수 있는 기능을 제공한다. 그 형태는 (그림 3)(그림4)과 같다.

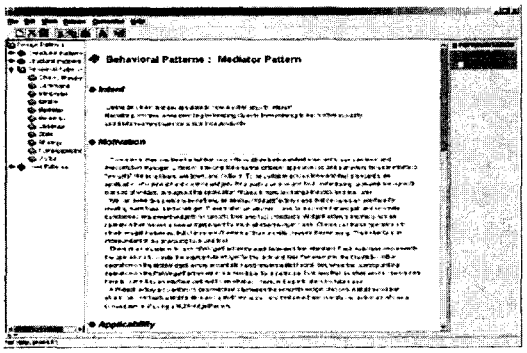

그림 3 디자인 패턴 시각적 요소(#1)

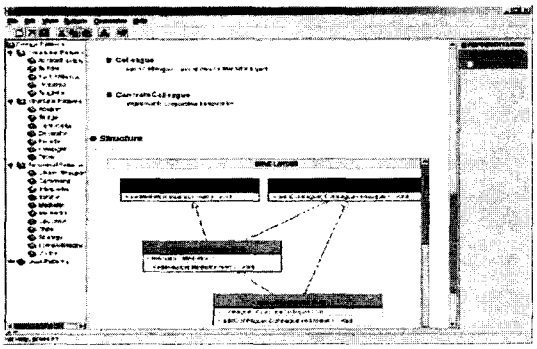

그림 4 디자인 패턴 시각적 요소(#1)

디자인 패턴 관리기의 시각적 요소는 입력된 디자인 정보를 효율적으로 분석 활용하기위한 부분이다. 패턴을 표현하는 시각적인 요소는 GoF 패턴에서 제시한 표현법을 따른다[5]. 제시된 구성형식은 Pattern Name and Classification, Intent, Also Known As, Motivation, Applicability, Structure, Participants, Collaborations, Consequences, Implementation, Sample Code, Known Uses, Related Patterns를 가진다. 본 관리도구에서는 이러한 내용을 기준으로 제 구성하였다. 두 번째는 디자인 패턴의 생성과 관리를 위한 입력 부분이 있다. 이곳에서는 필요한 디자인 패턴의 정보를 입력하고 관리하기 위한 부분이다. 이 입력 부분은 (그림 5)에서 보여준다.

이 입력요소는 각 패턴에 필요한 정보와 구조를 기술하는 부분으로 이 입력된 정보는 시각요소를 통해 문서

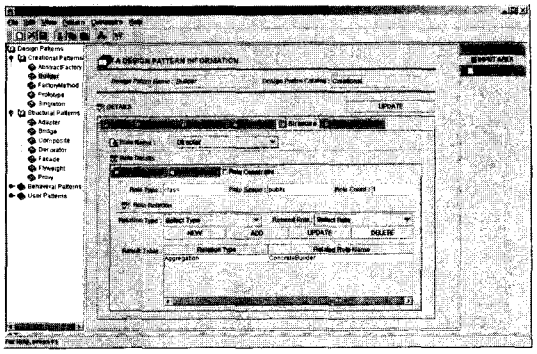

그림 5 디자인 패턴 관리기의 입력요소

형태로 보여질 수 있으며 소프트웨어 설계에 이용되어진다. 세 번째는 디자인 패턴에 대한 정보 뷰이다. 이 정보는 입력기를 통해 얻어진 정보를 이용해 XML문서 형태로 저장 관리하기 위한 뷰이다. 디자인 패턴에 대한 XML표현 방법에 대한 연구는 [19]에서 제시하였다. 이렇게 구성된 XML표현은 (그림 6)에서 보인다. 이러한 디자인 패턴 관리 도구를 이용하여 객체 지향 어플리케이션에 사용될 수 있는 디자인 패턴 정보를 입력하고 관리할 수 있는 기능을 담당한다. 또한 필요시 디자인 패턴 관리자는 언제든지 새로운 패턴을 추가 또는 수정할 수 있는 기능을 수행할 수 있다. 이 디자인 패턴 관리기에서 디자인 패턴을 구성하는 클래스들은 어플리케이션을 구성하는 클래스와의 구분을 위해 디자인 패턴 내부의 클래스들을 역할이라 정의한다. 이러한 역할 정보는 어플리케이션에 구체화 될 때 하나이상의 클래스에 역할이 포함되는 형태로 구성될 수 있다.

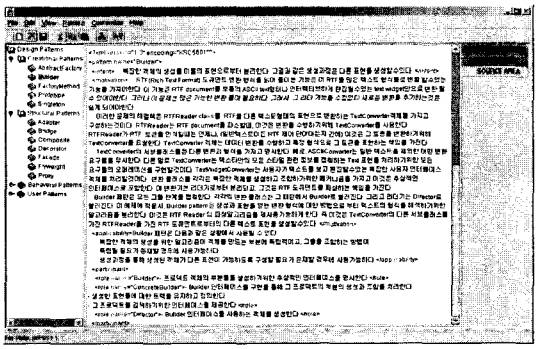

그림 6 디자인 패턴에 대한 XML 뷰

3.4 소프트웨어 설계 지원 도구

소프트웨어 설계 지원 도구는 UML기반의 소프트웨어

설계를 지원하는 도구이다. 그래픽한 환경에서 클래스 구조 다이어그램을 형성할 수 있다. 또한 패턴 지향 어플리케이션 개발에서 필요한 객체의 식별과 디자인 패턴 요소의 식별을 통해 필요한 클래스 구조를 구성할 수 있으며, 디자인 패턴 관리도구에서 생성한 디자인 패턴들을 이용해 디자인 패턴을 표현하는 기능을 자동으로 생성하고, 구성된 구조를 통해 소스코드를 자동 생성하는 기능을 제공한다. 이 설계도구의 구성은 (그림 7)과 같다.

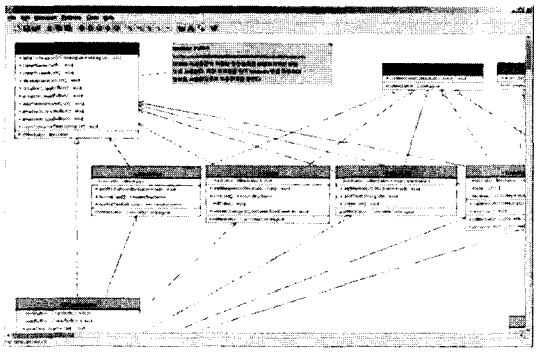

그림 7 설계 지원도구

(그림 7) 패턴 지향 소프트웨어 설계를 지원하는 개발 지원도구를 보여준다. 이 지원 도구에서의 객체 표현 구조는 (그림 8)과 같다.

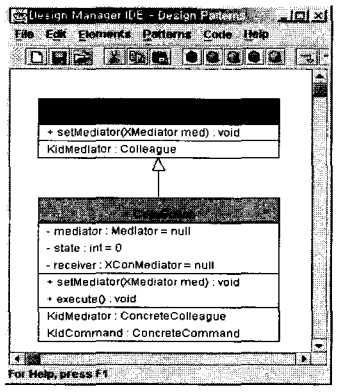

그림 8 객체 표현 구조

클래스 객체를 표현하는 구조는 크게 4부분인 클래스 이름, 프로퍼터 정보, 메소드 정보 그리고 적용된 디자인 패턴 정보로 구성된다. 또한 클래스의 종류로는 추상화 클래스, 인터페이스 클래스 그리고 구체화 클래스로 구성되고 클래스들 간의 관계표현은 연관관계(Association), 상속관계(Inheritance), 집단관계(Aggregation), 의존관계(Dependency)로 구성된다. (그림8)에서 XColleague 클래스는 Mediator 패턴의 Colleague 역할을 담당하고 있으며, CopyButton은 XColleague를 상속받고, Mediator 패턴의 ConcreteColleague 역할과 Command 패턴의 ConcreteCommand 역할을 담당한 형태를 보여주고 있다. 이러한 정보 역시 XML형태의 문서로 저장되어 소스코드 관리기와 협력하여 동작한다. 4장에서 이러한 소프트웨어 설계 도구를 이용해 어플리케이션 제작과정을 기술한다.

3.5 소스코드 관리 도구

소스코드 관리기 에서는 소프트웨어 설계를 통해 얻어진 정보를 이용해 자동으로 소스코드를 생성하고 어플리케이션 개발에 필요한 소스코드 추가를 통해 어플리케이션 제작을 완성할 수 있는 지원 도구이다. 이러한 지원 도구는 (그림 9)에서 나타낸다. 이 소스코드 관리 도구는 소프트웨어 설계에서 제공하는 정보를 기반으로 소스코드를 자동생성하고 적용된 디자인 패턴에 대한 추가적인 프로퍼티 및 메소드 정보 그리고 코드 정보를 얻어와 소스코드를 형성한다. 이러한 소스코드는 설계정보를 통해 일관되고 정확한 소스코드를 제공할 수 있으며, 적용된 디자인 패턴에 사용되는 프로퍼티 및 메소드 정보를 추가함으로써 디자인 패턴 사용의 효율성을 증가한다.

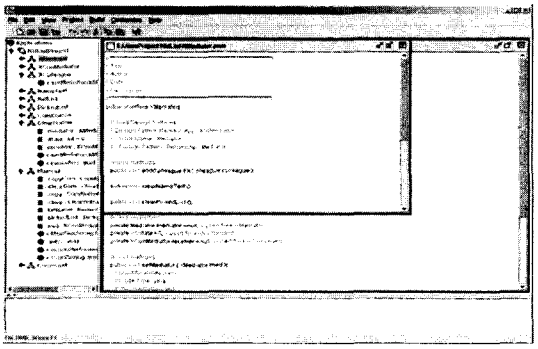

그림 9 소스코드 관리기

4. 제시된 도구를 이용한 소프트웨어 개발 적용 및 활용

제시된 소프트웨어를 이용한 소프트웨어 설계 및 활용방법을 POS(Point of Sale)시스템의 예를 들어 설명한다. 또한 이러한 소프트웨어 개발을 위한 방법론으로 UP(Unified Process)방법론을 이용한다[10]. UP 개발

방법론은 점진적이고 반복적인 개발 방법론으로 객체지향 방법론으로 많이 활용되고 있는 분야이다. 이 방법론은 그림 10에서 보여주는 것처럼 위험성과 활용도가 높은 부분들을 점진적으로 해결해 나감으로써 소프트웨어를 완성해 나가는 방법이다. 각 반복단계의 결과는 다음 반복단계의 입력으로 활용되고 다음 반복단계에서 새로운 요구사항을 포함하거나 이전 반복단계에서 지적된 내용을 수정 보안함으로써 보다 안정된 시스템을 구축할 수 있는 방법을 제시하고 있다.

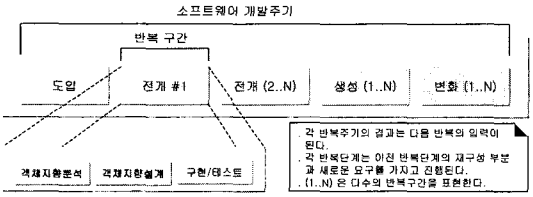

그림 10 UP의 반복적이고 점진적인 개발방법론

객체지향 소프트웨어 개발의 주된 관심은 객체에 책임을 할당하는 정책일 것이다. 이러한 책임 할당에 대한 기본 원칙을 정리하는 패턴이 GRASP(Patterns of General Principles in Assigning Responsibilities) 패턴이다[9]. 이 GRASP는 객체에 책임을 할당하기 위한 기본 원리를 분류하고 이들의 해결책을 제시하고 있다. GRASP에서 책임할당 원리로 "Information Expert", "Creator", "High Cohesion", "Low Coupling", "Controller", "Polymorphism", "Indirection", "Pure Fabrication", "Protected Variations" 등과 같은 내용을 포함하고 있으며 이들은 소프트웨어 객체에 책임을 할당하기 위한 기본 원칙을 제시한다. 또한 이들의 특성을 고려해 특정 상황에서 발생되는 문제를 해결하기 위한 패턴으로 GoF 디자인 패턴이 있다[5]. 그러므로 이들 패턴의 효율적인 적용과 분석은 프로그램의 성능을 향상시키는데 크게 공헌한다. 본 논문에서 이러한 목적을 달성하기 위해 패턴정보를 소프트웨어 설계 시 보다 편리한 방법으로 제공할 수 있는 방법을 제시하고 있다. 본장에서는 실제 적용 과정을 통해 이러한 내용이 적용되는 과정과 효율성을 분석한다

4.1 POS 시스템 의 Use Case 모델

UP 방법론은 Use Case 지향적 접근 방법을 통해 이루어진다. POS 시스템을 구성하기 위한 Use Case 다이어그램은 그림 11에서 보여준다. 이 POS 시스템에서 발견된 Use Case를 기반으로 각 반복단계에서는 그 단계에서 처리할 부분과 대상을 선정함으로써 진행된다. 물론 POS 시스템에서 가장 위험성과 활용도가 높은 부분을 선정하는 작업이 필요하다. 여기에서는 Process Sale Use Case 부분을 선택한 반복단계를 고려한다. 이 단계에서 처리하고자하는 SSD(System Sequence Diagram)는 그림 12에서 보여준다.

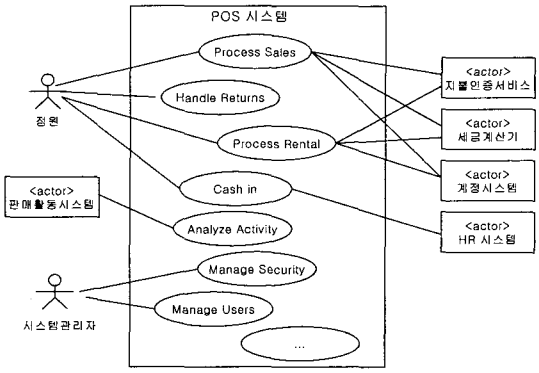

그림 11 POS 시스템 Use Case 다이어그램

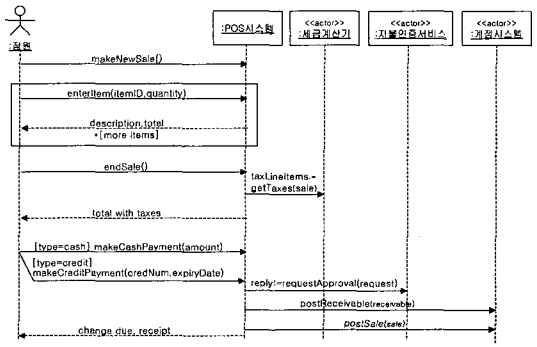

그림 12 Process Sale을 위한 시스템 순서 다이어그램

이 반복단계에서 Process Sale Use Case의 분석을 통해 도메인 모델에 존재하는 객체를 선택할 수 있다. 이렇게 선택된 객체는 그림 13에서 보여준다.

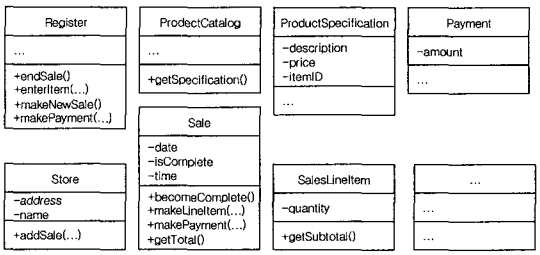

그림 13 POS 시스템에 사용되는 객체들

이렇게 검색된 객체들은 GRASP 패턴 정책에 따라 책임 할당이 이루어진다. 또한 성능 향상을 위해 새로운 시스템 클래스들이 추가되어 연관관계를 구성할 수 있다. 이때 특정 목적을 달성하기 위해 GoF 디자인 패턴이 사용될 수 있다. 이러한 패턴들은 제안된 도구의 디자인 패턴 관리시스템에 의해 관리되고 설계 시 설계 관리시스템에 정보를 제공함으로써 효율적인 추가 및 수정이 가능하다. 또한 이러한 디자인 패턴정보를 포함한 설계문서는 소프트웨어 개발에 필요한 소스코드를 자동 생성한다. 이러한 응용을 보기 위해 하나의 적용 예를 살펴본다.

4.2 디자인 패턴 활용 예

적용 예로 Process Sale에서 지불처리가 이루어질 때 각 특성에 따라 지불정책을 가지고 있다고 가정하자. 적용 규칙은 다음과 같다.

1. 월요일에 구입하는 상품에 대해서는 전체금액의 2%를 할인한다.
2. 구입총금액이 50만원 이상일 경우 5만원을 할인한다.
3. 고객의 종류에 따라 할인규칙이 적용된다.
   3.1 고객이 직원일 경우 구입금액의 10%를 할인한다.
   3.2 고객이 우수고객일 경우 5%를 할인한다.
   3.3 60세 이상의 고객일 경우 3%를 할인한다.
4. 여러 가지 할인정책이 중복될 경우 각 판매상의 규칙에 따라 변경될 수 있다.
   4.1 고객을 위한 할인율 적용
      4.1.1 지불정책이 동시에 존재할 경우 가장 높은 할인정책 적용
   4.2 판매수익을 증가시키기 위한 경우
      4.2.1 지불정책이 동시에 포함될 경우 가장 낮은 할인정책을 적용

이러한 적용규칙을 효율적으로 적용하기 위해 디자인 패턴을 활용할 수 있다. 먼저 다양한 지불정책을 다루기 위해 Strategy 패턴을 고려할 수 있다. Strategy는 알고리즘의 집합을 정의하고 그 알고리즘들이 클라이언트에 독립적으로 제공됨으로써 알고리즘의 의존성을 줄이는 목적으로 사용된다. 이 그림 14와 같은 구성을 가진다.

이 Strategy 패턴은 위에서 제시한 여러 가지 지불정책을 다루는데 적용될 수 있다. 그림 15는 디자인 패턴과 소프트웨어 객체와의 관계를 관리하기 위한 목적으로 제시된다. 이러한 문서는 해당 패턴과 소프트웨어 객체의 적용관계를 명시하고 있으며 이들은 소프트웨어 설계 시 객체들에 패턴을 할당할 때 활용된다.

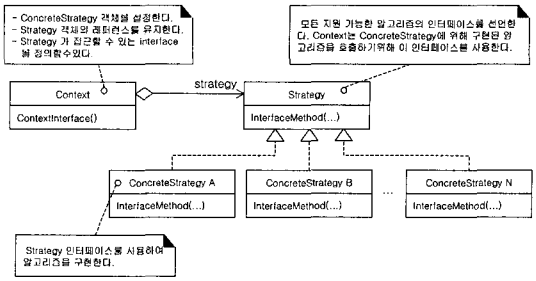

그림 14 Strategy 패턴 구조

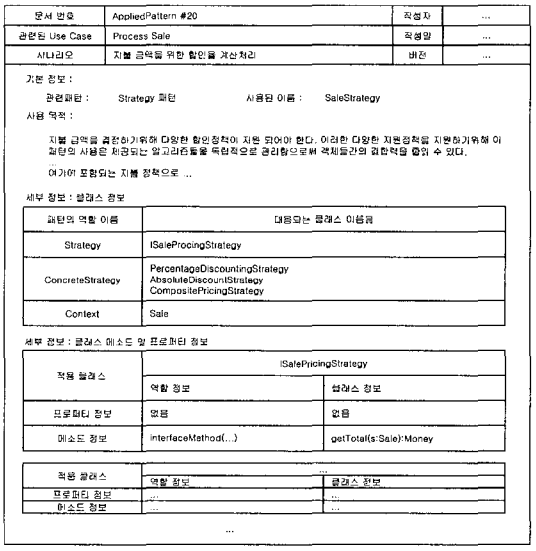

그림 15 적용패턴과 클래스 정보 관련 문서

그림 15의 표현 문서는 소프트웨어 설계 시 소프트웨어 클래스에 정보를 할당하기 위해 사용된다. 실제 이 구성 시스템에서 이 내용은 XML 문서화 되어서 저장되고 이를 정보와 패턴 관리기에서 관리되는 정보 그리고 소프트웨어 설계 관리기에서 관리되는 설계정보와 조합하여 해당 클래스들에 책임을 할당하고 연관관계를 구성한다. 이렇게 구성된 소프트웨어 디자인 구조 다이어그램은 그림 16에서 보여준다.

이 구조는 Strategy와 Composite 패턴이 적용된 구조이다. 클래스의 네 번째 사각형박스는 클래스에 부여된 객체의 역할을 표현한다. 예로 ISaleStrategy 클래스는 Strategy 패턴의 Strategy 역할과 Composite 패턴의 Component 역할을 수행하는 것을 알 수 있다. 또한 이러한 객체들 간의 책임할당과 연관관계는 디자인 패턴 정보를 통해 생성될 수 있다.

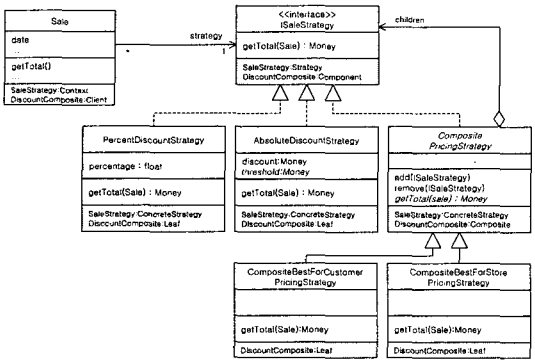

그림 16 디자인 패턴의 역할을 포함한 소프트웨어 구조 다이어그램

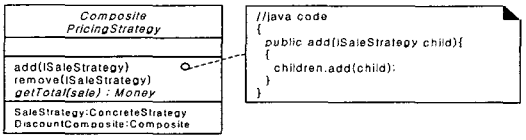

그림 17 디자인 패턴에 포함된 메소드 내용

그림 17은 디자인 패턴에 포함된 메소드가 클래스에 할당될 때의 관계를 표현한다. 패턴과 소프트웨어 객체들 간의 관계를 명시한 정보를 통해 클래스에 메소드가 추가되고 이들 메소드 중 동적인 행위를 표현하는 구현 내용이 존재할 경우 UML의 노트 표현을 이용하여 클래스 정보에 포함시킬 수 있다. 이 코드 정보는 소프트웨어 설계 내용이 코드로 생성될 때 자동적으로 포함되어 나타난다.

제시된 도구는 이러한 XML 형태의 전자 문서화된 패턴 정보를 소프트웨어 설계에 효율적으로 적용할 수 있으며, 적용된 패턴정보 역시 클래스 대 역할의 관계로 관리함으로써 각 클래스의 특성과 패턴을 쉽게 파악하고 활용할 수 있다. 특히 반복적인 단계를 통한 소프트웨어 개발은 지속적인 객체들의 수정과 변환이 이루어진다. 즉 초기 반복단계에서 적용한 패턴이 다른 요구들에 의해 다른 패턴들로 변환될 필요성이 존재한다. 이때 해당 패턴을 적용한 클래스들의 빠른 검색과 이들 관계의 분석은 보다 견고한 시스템을 구축하는데 크나큰 영향을 미칠 수 있다.

5. 디자인 패턴 지향 소프트웨어 설계 지원 도구에 대한 비교 분석

디자인 패턴을 지원하기 위한 몇 가지 연구들이 존재한다. 먼저 Sefika[15]는 디자인 패턴을 포함한 설계정보가 구현내용과 일치하는지를 검사하는 방법을 연구하였다. 이러한 방법은 각각의 디자인 패턴을 위해 하나의 규칙을 정의하고 그 규칙을 통해 구현 내용을 검증하는 과정을 통해 이루어진다. 이 방법은 패턴으로부터 개발된 시스템을 검증하기 위한 목적으로 사용되지만 패턴 정보를 이용하여 어플리케이션 개발에 적용하는 방법에 대해서는 지적하지 않고 있다. Stephen[12]는 디자인 패턴을 이용해 소프트웨어 컴포넌트를 통합시키기 위한 방법을 연구하였다. 이 방법은 디자인 패턴 정보를 이용하여 어플리케이션에 사용되는 컴포넌트들을 효율적으로 통합하고자 하는 목적으로 연구되었다. 그러나 디자인패턴을 이용해 컴포넌트 웹퍼 클래스를 생성하는데

표 2 디자인 패턴 활용을 위한 기존의 연구와의 비교

관련연구	접근방법	문제점	제안된 시스템
Sefika[15]	디자인 패턴에 규칙정의 하여 설계 정보가 그 규칙을 만족하는지를 검증하기위한 목적	패턴에 대한 규칙 관리	XML기반의 패턴 정보 관리
		설계 정보에 디자인 패턴 자동화나 적용 방법에 대한 제시 부족	자동화를 통한 디자인 패턴 문서 적용
		사용된 패턴에 대한 추적성 부족	설계문서에 패턴 정보를 추가하고 각 적용 패턴별 검색지원
Stephen[12]	컴포넌트들 간의 효율적인 통합을 위한 디자인 패턴 활용 방안제시	웹퍼클래스의 생성을 통한 컴포넌트 간의 통합을 제공하지만 이들간의 정보제공관리 미흡	객체들간의 책임할당을 통한 소프트웨어 설계 지원하고 이들 객체간의 패턴정보 유지
Budinsky[6]	디자인 패턴으로부터 코드 자동생성기법 제안	적용할 패턴 정보를 통한 소스코드 자동 생성은 제공하지만 설계정보에는 반영되지 않는다.	적용할 패턴 정보는 디자인 설계 정보에 자동적으로 적용되고 이를 통해 소스코드 자동생성 함으로써 설계와 구현 정보에 반영
Eden[2]	메터언어를 이용한 어플리케이션 소스코드 생성기법 제안	메터언어에 대한 추가 지식이 필요하고 이 방법 역시 설계 정보의 활용방법은 제시되지 않는다.	문서기반의 자동적인 패턴 적용과 설계정보 구성.
기타 CASE 도구	설계문서를 작성하기위한 효율적인 기능제공	설계에 필요한 다이어그램 작성 및 문서 관리에 목적을 두고 있으며 디자인 패턴 활용 방법 제시못함	설계에 필요한 다이어그램 생성시 적용시킬 패턴이 존재 시 자동적으로 패턴 요소 추가 및 관리

목적을 가지고 있기 때문에 생성된 디자인 패턴들 간의 관계나 관리측면에서 미흡하다. Budinsky[6]은 디자인 패턴으로부터 코드생성 기법을 제시하고 있다. 이 방법은 선택된 패턴에 대해 명시적인 제공된 어플리케이션 명세를 이용해 하나의 패턴을 자동적으로 생성하는 기능을 가진다. 그러나 이 방법 역시 코드 생성에 중점을 두고 있기 때문에 디자인 패턴들 간의 관계 및 디자인 패턴 정보 관리문제를 지적하지 않고 있다. Eden[2]는 메터언어를 이용해 어플리케이션 소스로 디자인 패턴 기능을 포함시키는 방법을 제시하고 있다. 그러나 이 방법은 디자인패턴을 적용하기 위해 메터언어에 대한 추가적인 지식이 필요하고 메터언어를 통해 만들어진 정보는 다시 프로그램에 적용시키는 단계를 거쳐야한다. 또한 일반적인 CASE 도구에서는 아직까지 디자인 패턴을 쉽게 적용할 수 있는 적절한 방법을 제시하기 못하고 있는 듯 하다.

패턴 활용을 위한 기존 연구의 특징과 비교되는 본 연구의 결과는 표 2와 같이 구성된다.

6. 결 론

디자인패턴은 문제해결을 위한 추상화된 해결책이다. 이러한 정보는 어플리케이션에서 이용되기 위해서는 디자인 패턴에 대한 구체화 작업에 필요하다. 이를 위해 본 연구에서는 객체 지향 프로그램에서 효율적으로 디자인 패턴을 적용할 수 있는 지원 도구를 제시하였다. 이 디자인 패턴 지향 소프트웨어 개발 지원도구는 크게 디자인 패턴 관리부와 소프트웨어 설계부 그리고 소스코드 관리부로 크게 나누어져있으며, 패턴지향 설계에 필요한 역할을 수행한다. 이를 통해 어플리케이션 개발에 디자인 패턴을 보다 쉽게 그리고 명확하게 적용할 수 있도록 했으며, 개발 사이클의 내에서 적용된 자동화 기능을 통해 보다 일관되고 신뢰할 수 있는 소프트웨어를 개발할 수 있는 지원 도구를 제시하였다. 또한 어플리케이션 내에 적용된 디자인 패턴의 정보를 유지 관리함으로써 디자인 패턴의 관계와 각 구성요소간의 분석 및 활용 성을 증가시켜 개발된 어플리케이션의 유지보수의 효율성을 증가 시킬 수 있다. 또한 모든 관련데이타를 XML형태의 자료로 관리할 수 있도록 함으로써 자료의 활용능력을 증가시켰다. 이러한 개발 지원 도구는 소프트웨어 개발 시 보다 효율적인 패턴지향환경을 제공할 수 있을 것이다.

참 고 문 헌

[1] A. Cornils and G. Hedin. Statically checked documentation with design patterns. Technology of Object-Oriented Languages, 2000. TOOLS 33. Proceedings. 33rd International Conference on, 2000

[2] A. Eden, A. Yehudai and J. Gil. Precise specification and automatic application of design patterns. Automated Software Engineering, 1997. Proceedings., 12th IEEE International Conference, 1997

[3] B. Schulz, T. Genssler, B. Mohr and W. Zimmer. On the computer aided introduction of design patterns into object-oriented systems. Technology of Object-Oriented Languages, 1998. TOOLS 27. Proceedings , 1998

[4] C. Marcos, M. Compos, and A. Pirotte, Reifying Design Patterns as Metalevel Constructs, Electronic Journal of Sadio , 2(1) Page(s) 17-19, 1999

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

[6] F. Budinsky, M. Finnie, J. Vlissides, and P. Yu. Automatic Code Generating from Design Patterns. IBM Systems Journal, 35(2), 1996

[7] F. Buschmann and R. Menuier, "A System of Patterns," Pattern Languages of Program Design, Coplien and Schmidt, eds., Addison-Wesley, 1995

[8] F. Buschman, R. Meunier, H. Rohnert, P. Sommerlad, and Stal Michael. Pattern-Objected Software Architecture - A System of Patterns. John Wiley&Sons, 1996

[9] J. Cooper. Java Design Patterns, Addison-Wesley, 2000

[10] R. Wirfs-Brock, B. Wilkerson, and L. Widner, Designing Object-Oriented Software, englewool Cliffs, NJ.:Prentice-Hall, 1990

[11] S. Ambler, The Unified Process - Elaboration Phase, Lawrence, KA.: R&D Books, 2000

[12] S. Stephen, S. Yau and Ning Dong, Integration in component-based software development using design patterns, Computer Software and Applications Conference, 2000. COMPSAC 2000. The 24th Annual International , 2000

[13] S. Yacoub, Xue, H. and Ammar, H.H. Automating the development of pattern-oriented designs for application specific software systems, Page(s): 163 -170, Application-Specific Systems and Software Engineering Technology, 2000. Proceedings. 3rd IEEE Symposium on, 2000

[14] M. Claudia, C. Marcelo, P. Alain. Reifying Design Patterns as Metalevel Constructs. Electronic

Journal of SADIO vol.2,no. 1, pp.17-29, 1999
[15] M. Sefika, A. Sane, and R. Campbell. Monitoring Compliance of a Software System with its high-Level Design Models. Proceedings of the 18th International Conference of Software Engineering, ICSE'96, Berlin, Germany, March, 1996.
[16] M. Sherif, Yacoub and Heany H. Ammar Pattern-oriented analysis and design(POAD): a structural composition approach to glue design patterns, Technology of Object-Oriented Languages and Systems, 2000. TOOLS 34. Proceedings. 34th International Conference on, 2000
[17] W. Tichy, Essential Software Design Patterns. University of Karsruhe. http://wwwipd.ira.uka.de/~tichy/patterns/overview.html, 1997
[18] W. Vanderperren, B. Wydaeghe, Towards a new component composition process, Engineering of Computer Based Systems, 2001. ECBS 2001. Proceedings. Eighth Annual IEEE International Conference and Workshop on the , 2001
[19] 김운용, 김영철, 주복규, 최영근. 코드 자동 생성을 위한 XML기반의 효율적인 디자인 패턴구조. 한국정보처리학회 논문지. 제8-D권 6호, 2001

김 운 용
1996년 독학사 전자계산학과(이학사). 1999년 광운대학교 정보과학기술대학원(이학석사). 1999년 ~ 현재 광운대학교 컴퓨터과학과 박사과정. 관심분야는 객체지향프로그래밍 언어, 객체 모델링, 디자인패턴, 재사용기법, 컴포넌트 개발방법, 분산컴퓨팅기술

최 영 근
1980년 서울대학교 수학교육과(이학사). 1982년 서울대학교 계산통계학과(이학석사). 1989년 서울대학교 계산통계학과(이학박사). 1992년 ~ 현재 광운대학교 컴퓨터과학과 교수. 1992년 ~ 2000년 광운대학교 전산정보원 원장. 2001년 ~ 2002년 광운대학교 정보통신연구원장. 2002년 ~ 현재 광운대학교 교무연구처장. 관심분야는 프로그래밍 언어, 병렬 프로그래밍언어, 객체지향 설계 및 분석, 분산컴퓨팅기술, Mobile agent