디자인 패턴 컴포넌트의 명세와 조립에 관한 연구

하 성 민, 송 영 재

경희대학교 컴퓨터공학과
전화 : 031-201-2946

A Study on Specification and Composition of Design Pattern Component

Sung-Min Ha, Young-Jae Song
Dept. of Computer Engineering, KyungHee University
E-mail : sunnyob@cvs2.khu.ac.kr

Abstract

본 논문은 패턴 지향 설계를 함에 있어 필요한 구조적 디자인 패턴의 가시적 조립을 목적으로 하며, 재사용 가능한 패턴들을 명세 및 조립하는 방안을 제안함으로써 애플리케이션 설계의 복잡성을 감소시키고자 한다. 본 논문은 패턴 지향 설계를 함에 있어 필요한 구조적 디자인 패턴의 가시적 조립을 목적으로 한다.

디자인 패턴 컴포넌트의 명세에서 패턴 인터페이스들 사이의 관계를 명시적으로 정의하며 패턴의 내부와 인터페이스 사이의 관계를 기술한다. 디자인 패턴 컴포넌트의 조립은 패턴 타입과 인스턴스 네임으로 구성되며, 두 패턴 사이의 관계는 종속으로 지시되고 저장소로부터 패턴을 선택하여 종속을 정의하고 방향을 정해주게 된다.

I. 서론

객체지향 시스템의 설계단계에서의 반복적인 설계 문제에 대한 해결방안을 이론적 제시와 시스템 개발에 적용 가능하도록 구체적으로 규격화시킨 것이 디자인 패턴이다[1]. 하지만, 디자인 패턴을 적용함에 있어 대부분의 설계자들이 수작업을 통해 디자인 패턴을 설계에 적용함으로써 작업의 복잡성과 비효율성의 증대라는 문제점이 드러나고 있다. 이러한 변화와 아울러 빠르게 시스템을 구축할 수 있고, 컴포넌트 재사용이 가능하여 개발비용을 절감시킬 수 있으며, 소프트웨어의 증가하는 복잡도를 감소시킬 수 있는 컴포넌트 기반 소프트웨어 개발 방법론(CBSE)이 주목받고 있다[2].

따라서, 본 논문에서는 반복적 설계 문제를 해결하기 위한 구조를 갖고 있는 디자인 패턴을 기반으로 하는 컴포넌트의 명세 및 조립에 관한 연구를 제안하였다. 본 논문에서 제안하는 디자인 패턴 컴포넌트는 Gamma의 디자인 패턴 분류 중 구조 패턴만을 대상으로 하며, 인터페이스 정의와 조립을 통해 시스템 설계 시 디자인 패턴을 효율적으로 적용하고자 하였다.

II. 디자인 패턴 컴포넌트

2.1 디자인 패턴 컴포넌트의 정의

디자인 패턴을 설계단계에서 적용시 체계적인 분류 및 공유가 이루어지지 않아 적당한 패턴을 찾지 못하거나, 참조되지 못하는 경우가 많았다. 이를 디자인 패턴 라이브러리(Design Pattern Library)를 구축함으로써 해결하기 위한 연구가 최근 진행되고 있으며, 패턴 정보와 관련 패턴 구조를 컴포넌트화하여 제공하고 있다. 디자인 패턴이 CASE 도구간 호환성이 없는 구조를 갖고, 웹 기반에서 개발자나 개발 도구간에 공유할 수 없는 경우, 플랫폼에 따라 사용이 제한되는 경우 등

이 빈번하게 발생하므로 UML로 표현되는, 가시적으로 모델링한 디자인 패턴 구조를 표현하여 저장할 필요가 있다.

본 논문에서는 이처럼 UML로 표현된 가시적 모델링 구조를 갖는 디자인 패턴을 컴포넌트화하여 디자인 패턴 컴포넌트로 정의한다.

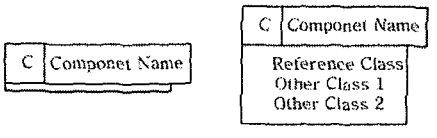
2.2 디자인 패턴 컴포넌트의 구조

(1) 구성 요소

디자인 패턴 컴포넌트의 핵심 구성요소들은 주로 기초 디자인 패턴의 구성요소에 의해 구성된다. 본 논문에서는 Gamma에 의해 제안된 구성 요소들의 목록을 적용하고 추가적으로 소스 코드, 실행 코드, 접근 권한, 설계 변화과정 등의 구성 요소들을 통합하여 디자인 패턴 컴포넌트가 이루어 진다.

(2) 표기

의도, 동기, 결과등의 간략한 구성요소를 묘사하기 위해서, 간단한 텍스트나 그림을 이용할 수 있다. 본 논문에서는 디자인 패턴 컴포넌트의 표기로써 UML 패키지 표현을 적용하고 확장한다. 디자인 패턴 컴포넌트의 이름에 "C"로 주석을 단다. 디자인 패턴 컴포넌트의 구성 클래스는 컴포넌트 기호로 보여질 수 있으며, 대부분의 디자인 패턴 컴포넌트는 하나의 우세한 클래스를 가지며, 그 클래스는 일반적으로 추상 명세를 제공한다. 이 참조 클래스는 (그림 1)에서와 같이 핵심 기호로 주석을 단다. 디자인 패턴 컴포넌트를 묘사하는 서로 다른 구성 요소들은 상세 뷰에서 확대되어 보여줄 수 있도록 핵심 기호로 표시된다.



(그림 1) 디자인 패턴 컴포넌트 기호

(3) Services

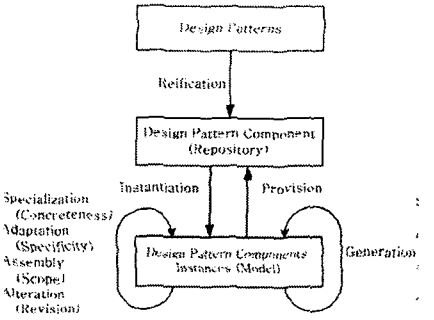
디자인 패턴 컴포넌트는 영속성(Persistence), 검사(Inspection), 이벤트 핸들링 (Event Handling), Customization, 보안(Security), 코드 생성(Code generation), 통합(Integration)의 7개 항목을 제공한다.

(4) 인스턴스화 방식

디자인 패턴 컴포넌트는 주어진 요구사항을 받아 인스턴스화 되어 만들어질 수 있다. 인터페이스들은 네트워크 컴퓨터 환경과 비네트워크 환경 모두에서 전개될 수 있으므로, 인스턴스화에서의 두가지 경우(원격, 지역)가 정의되어 왔다.

2.3 디자인 패턴 컴포넌트 개발 프로세스

본 논문에서 제안하는 개발 프로세스의 개략적인 모습은 (그림 2)에 나타나 있다. 설계 조합은 저장소에 유지되는 디자인 패턴의 집합으로 부터 출발하며, 디자인 패턴 컴포넌트로 구체화된다. 인스턴스화는 특수화, 변경, 적용, 다른 인터페이들과의 조합 등을 위해서 디자인 패턴 컴포넌트의 새로운 인스턴스를 제공한다. 제공된 새로운 인터페이스는 인스턴스화 된 디자인 패턴 컴포넌트를 위한 컨테이너로서 하나의 모델을 생성한다. 생성된 모델은 디자인 패턴 컴포넌트로 시스템에서 정의된 인스턴스로 실현된다. 이 때, 생성된 모델은 보통의 디자인 패턴 컴포넌트와 동일한 함수성을 가지며 특수화를 통해 추상화 계층과 디자인 패턴 컴포넌트의 구체적 개념이 생성된다.



(그림 2) 디자인 패턴 컴포넌트 개발 프로세스 개요

구체화(Reification)는 디자인 패턴을 명확한 결과물로 전환하는 활동으로 패턴의 구성 요소들을 묘사하고, 텍스트나 그래픽 다큐먼트와 같은 물리적 다큐먼트의 형태로 제공한다.

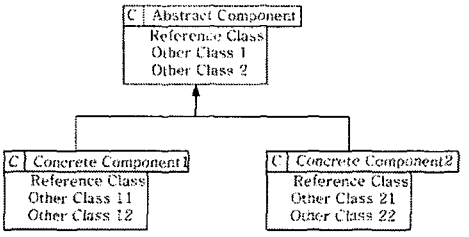
디자인 패턴 컴포넌트들은 객체지향 프로그래밍 언어와 마찬가지로 인스턴스화(Instantiation) 되어 컴포넌트의 특정한 예를 만든다. 클래스와 달리 컴포넌트 인스턴스화는 구조적이고 행위적인 것 뿐 아니라 문맥에 맞게 조정되고 확장된 문서화를 포함한다.

특성화(Specialization)는 명세 요구에 대해 디자인 패턴 컴포넌트를 조정하는 동작이며, (그림 3)에 개략적인 모습을 나타내었다.

적응화(Adaptation)는 특정 어플리케이션이나 영역 요구사항에 적합하게 만들기 위한 조정 행위이며, 명세화를 설명하고 디자인 패턴 컴포넌트를 특정 어플리케이션 영역에 적합하도록 조정하여 그 디자인 패턴 컴포넌트를 도메인 명세로 만들어 준다.

조립(Assembly)은 기본 컴포넌트를 연결함으로써 합

성 컴포넌트의 생성하는 행위이다. Push 버튼, 텍스트 박스, 이미지 등의 부품 조립을 고려하는 그래픽 유저 인터페이스 설계 도구와 유사하게 디자인 패턴 컴포넌트를 지원하는 도구는 결합 기법을 제공한다.



(그림 3) 디자인 패턴 컴포넌트의 특성화 표기

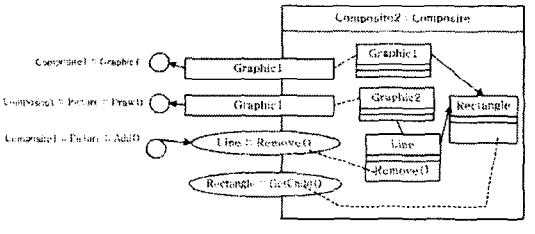
변경(Alteration)은 디자인 패턴 컴포넌트를 변화하는 요구사항에 맞도록 변경하는 동작이다. 이러한 변경은 특수화, 적응화, 조립을 통해 생성된 변경된 디자인 컴포넌트 및 그 도출물의 일관성을 보장하는 제어 기법을 요구한다.

제공(Provision)은 분해 처리에서 재사용을 위한 디자인 패턴 컴포넌트로써 컴포넌트 기반 모델을 공급하는 행위이다[3,4].

III. 디자인 패턴 컴포넌트의 명세 및 조립

3.1 디자인 패턴 컴포넌트의 명세

디자인 패턴 컴포넌트의 명세는 패턴 인터페이스들 사이의 관계를 명시적으로 정의하며 패턴의 내부와 인터페이스 사이의 관계를 기술한다. 명세에 있어서 클래스와 오퍼레이션 간에는 인터페이스 클래스가 다른 패턴의 오퍼레이션을 호출함으로써 이루어진다.



(그림 4) Composite 패턴의 상세 패턴 레벨 뷰

디자인 패턴 컴포넌트의 명세는 상세 패턴 레벨 뷰에서 표현되며 상세 패턴 레벨 다이어그램은 패턴 레벨 다이어그램에서의 모든 패턴들을 대상으로 생성된다. 이 다이어그램은 패턴 내부의 클래스들이 패턴의 내부와 패턴 인터페이스들과 어떻게 관계지어 졌는가를 나타낸다. (그림 4)는 GoF의 디자인 패턴 분류중 구조패턴의 하나인 Composite 패턴을 대상으로 패턴 내부가 어떻게 패턴 인터페이스로 이루어 졌는가를 보여준다.

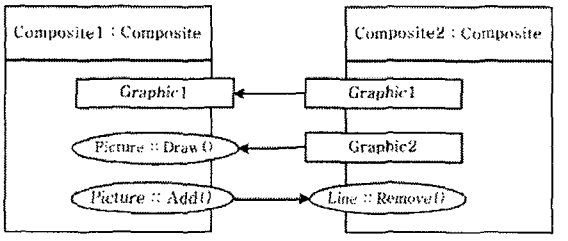
상세 패턴 레벨 다이어그램은 패턴에 참여한 패턴의 이름을 선택하고 어플리케이션 명세를 정의함으로써 패턴 어플리케이션 명세 성질을 추가할 수 있다.

3.2 디자인 패턴 컴포넌트의 인스턴스화

패턴 인터페이스 뷰에서는 패턴 인터페이스가 선언되어 디자인 패턴 컴포넌트를 인스턴스화 한다. 설계자는 패턴 인터페이스들 사이의 관계를 갖는 패턴 레벨 다이어그램을 사용하여 의존성 관계를 파악하고, 재정의한다.

의존 관계에 참여하는 두 패턴의 인터페이스는 인터페이스 클래스들과 인터페이스 오퍼레이션들로 표현된다. 인터페이스 클래스는 패턴 프레임에 패턴의 이름을 라벨화 된 탭으로 표현하며, 화살표의 방향은 클래스의 역할에 의해 결정된다. 인터페이스 오퍼레이션은 "Class Name :: Operation Name"의 형식으로 라벨화 된 타원으로 표현되며, 각각의 오퍼레이션은 그것이 속한 클래스의 이름에 의해 구분된다.

(그림 5)는 패턴 인터페이스뷰의 간략한 다이어그램을 나타낸다.



(그림 5) Composite 패턴의 패턴 인터페이스 다이어그램

(그림 5)에서 Composite1이라는 디자인 패턴 컴포넌트 인터페이스의 Picture::Draw()는 오퍼레이션 Draw()가 Composite1의 인터페이스이며, 그것이 Composite1의 내부 클래스인 Picuture에 포함된다는 것을 나타낸다.

이 단계에서의 관계는 3가지 형태로 구분할 수 있다. Class/Class의 관계는 UML 클래스 관계이며, Class/Operation의 관계에서는 인터페이스 클래스가 다른 패턴의 오퍼레이션을 호출할 수 있다. 이 관계의 형태는 인터페이스 클래스에서의 오퍼레이션과 이 오퍼레이션과의 상호 작용이 확실하지 않을 때 설계자에게

나타내 준다. (그림 5)에서 Composite2의 Graphic2는 Composite1의 Picture::Draw()를 호출한다.

Operation/Operation 관계는 설계자가 고려한 상세한 설계의 인식을 반영하는 상호 작용 모델 관계의 형태이다.

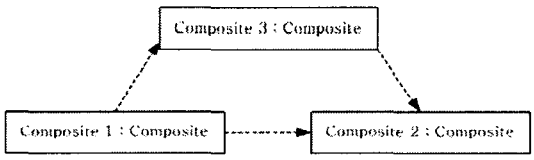
3.3 디자인 패턴 컴포넌트의 조립

디자인 패턴 컴포넌트의 조립은 패턴 타입과 인스턴스 네임으로 이루어 지는데 타입은 잘 알려진 패턴의 이름을 말하며 패턴 인스턴스 네임은 동일한 몇 개의 패턴을 실체화하게 된다.

디자인 패턴 컴포넌트의 조립에서 각각의 패턴들은 패턴 인스턴스 네임과 타입을 포함하는 사각형으로 표현된다. 조립 단계에서는 협력 패턴들에 의해 설계를 표현하고, 패턴의 의존성을 확인한 후, 선택된 패턴을 연결한다. 각각의 패턴 인스턴스는 자신만의 타입을 가지며 Observer, Factory, Strategy 등과 같이 널리 알려지고 문서화된 이름을 타입으로 정한다.

이 단계에서 원시 패턴의 변경은 목적 패턴의 변화를 필요로 하므로, 하나의 패턴은 다른 패턴의 신뢰성에 의존하게 된다. 따라서, 패턴 의존성이라 함은 두 패턴 사이의 의미론적 관계를 나타내며, UML 패키지의 'access'나 'generalization' 관계와는 다른 'use'관계이다. 패턴 의존성은 두 패턴의 인터페이스 클래스들간의 조합으로 패턴 인터페이스 뷰와 상세 패턴 레벨 뷰에 상세하게 재정의되어 있다.

이 단계에서는 패턴 저장소로부터 디자인 패턴을 선택하고, 패턴 의존성과 그들의 방향을 정의해 주어야 한다. 대략적으로, 패턴 레벨 뷰의 다이어그램은 UML 패키지 다이어그램과 닮았지만 의미론적으로 이들 두 다이어그램은 타입과 패키지의 성질에 있어서 차이가 있다.



(그림 6) Composite 패턴의 패턴 레벨 뷰

Ⅳ. 결론

디자인 패턴을 컴포넌트화 하는데에 따른 디자인 패턴 컴포넌트의 명세와 인스턴스화, 조립을 가시적으로 표현해 주는 과정은 애플리케이션 설계의 복잡성을 감소시킬 수 있다. 따라서 본 논문에서는 패턴의 인터페이스를 인터페이스 클래스와 인터페이스 오퍼레이션으로 정의하여 패턴 인터페이스들을 명세화하고, 명세화 단계에서 패턴 인터페이스들 사이의 관계를 명시적으로 정의하였으며, 디자인 패턴 컴포넌트의 조립은 두 패턴 사이의 관계를 종속 관계로 처리하여 저장소로부터 패턴을 선택하여 종속을 정의하고 종속의 방향을 정해주는 방법으로 조립의 방법을 선택하였다.

따라서 본 논문은 디자인 패턴을 컴퍼넌트화 하여 설계 적용함으로써 디자인 패턴 적용의 자동화와 재사용성의 증대시킬 수 있을 것으로 사료된다.

Reference

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

[2] T. Heineman, T. Councill "Component-Based Software Engineering" Addison-Wesley, 2001

[3] Sherif M.Yacoub, Hengyi Xue, Hany H.Ammar "Automating the Development of Pattern-Oriented Designs for Application Sepcific Software Systems" 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, 2000

[4] Rudolf K.Keller, Reinhard Schauer "Design Components:Towards Software Composition at the Design Level" ICSE'98, 1998.