UML 모델링 도구의 사용자 정의형 패턴 적용을 위한 설계 및 구현

이장우°, 이민규
㈜플라스틱소프트웨어 소프트웨어 기술연구소
{jwnara°, niklaus}@plasticsoftware.com

Design and Implementation for Applying User-Definable Pattern with UML Modeling Tools

Jangwoo Lee°, Minkyu Lee
Software Technology Laboratory, Plastic Software, Inc

요 약

소프트웨어 디자인 패턴(Design Pattern)은 좋은 설계나 아키텍처의 재사용을 도와주며, 이미 만든 시스템의 유지보수나 문서화도 개선해 준다. 패턴화를 통해서 클래스의 명세를 정확하게 하며, 객체간의 상호작용 또는 설계의 의도 등을 명확하게 정의할 수 있게 해준다. 사용자가 필요 시 디자인 패턴을 쉽고 편하게 UML 모델링 도구에 적용할 수 있다면 생산성 및 품질 개선에 크게 이바지할 것으로 판단된다. 본 논문은 UML 모델링 도구에 사용자 정의형 패턴을 쉽게 적용할 수 있도록 하는 기능을 설계하여 구현한 결과를 제시한다.

1. 서 론

소프트웨어의 개발은 점점 대형화되고 복잡해지면서 시스템 설계, 유지 보수 등의 중요성은 날로 증가하고 있다. 요즘들어 객체 지향 소프트웨어 개발 시 Unified Modeling Language(UML)[1] 모델링 도구들을 이용하여 시스템 설계를 하는 경우가 늘어나고 있다. 객체 지향 시스템 설계에 있어서 클래스들에 대한 책임 할당, 객체들의 상호작용 등은 매우 중요한 부분이다. 소프트웨어 디자인 패턴(Design Pattern)은 객체간의 상호작용 또는 설계의 의도 등을 명확하게 정의할 수 있게 해줌으로써 좋은 시스템 설계에 도움을 줄 수 있게 한다.

대부분의 UML 모델링 도구들을 보면 디자인 패턴이 도구에 포함되어 있기 때문에, 사용자가 필요에 의해서 새로운 패턴을 정의해서 쉽게 도구에 적용하기 어렵다. 예를 들어 특정 프로젝트에서 자주 사용되는 패턴이나 아키텍처 등을 쉽게 도구에 적용할 수 있게 된다면 시스템 설계 및 생산성에 크게 영향을 줄 것이라 생각된다.

본 연구는 UML 모델링 도구인 Agora Plastic[2]에 사용자 정의형 패턴을 쉽게 적용할 수 있는 기능을 추가로 설계하여 구현한 연구 결과를 제시한다. 본 논문의 2장에서는 관련 연구에 대하여 말하며 3장에서는 시스템 설계에 대하여 논하며 4장에서는 구현 결과를 제시하고 5장에서는 결론에 대하여 기술한다.

2. 관련 연구

2.1 디자인 패턴(Design Patterns)

소프트웨어 분야에 패턴 개념을 적용시키는 것에 대한 생각은 건축분야에서 시작되었다. 1977년과 1979년에 건축가인 Christopher Alexander는 " A Pattern Language: Towns, Buildings, Construction" 라는 책과 "The Timeless Way of Building"이라는 두 권의 책을 발표했다. 이 책들이 담고 있는 기본 생각은 건축물의 설계에 빈번화게 발생하는 동일 설계내용이 있으며 따라서 이런 것들을 하나의 패턴으로 보고 다른 건축물 설계에 재사용하는 것이 여러 가지 면에서 이득을 가져다 준다는 것이다.

1987년 Ward Cunningham과 Kent Beck은 Alexander의 생각을 사용해서 사용자 인터페이스(User interface)에 대한 다섯 가지의 패턴을 만들었다. 그리고 이 내용은"Using Pattern Languages for Object-Oriented Programs"라는 제목으로 정리되어 객체지향에 관한 세계적인 컨퍼런스인 OOPSLA(Object-Oriented Programming, Systems, Languages & Applications, 1987)에 논문으로 발표되어 소프트웨어 패턴에 대해서 학계에 공식적으로 알리는 계기가 되었다.

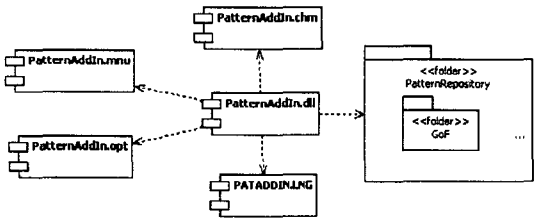
1990년대 초에는 Erich Gamma, Richard Helm, John Vlissides, Ralph Johnson이 90년대 가장 영향을 주었던 컴퓨터 책의 한 가지인 "Design Patterns: Element of Reusable Object-Oriented Software"[3]란 책 제작을 시작했다. 이것이 디자인 패턴에 대한 생각을 널리 알리는 계기를 만들게 되었다.

3. 시스템 설계

본 장에서는 UML 모델링 도구의 사용자 정의형 패턴 적용을 위한 설계 내용을 기술한다.

UML Modeling 도구인 Agora Plastic[2]은 모든 기능이 Microsoft의 COM 자동화(Automation)가 되어 있어 어떠한 COM 지원언어에서도 Agora Plastic을 제어할 수 있고, 또한 통합된 추가 기능을 쉽게 구현할 수 있게 설계 되어있다.

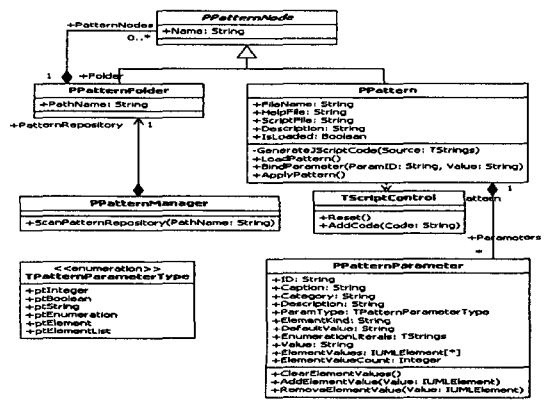
사용자 정의형 패턴 적용 기능의 추가를 위해 COM Automation을 통해 Agora Plastic[2]의 모델/메타모델 및 도구의 모든 기능에 접근하여 사용하도록 설계하였다.



(그림 1) 컴포넌트 다이어그램

(그림 1)은 Agora Plastic[2]에 사용자 정의형 패턴 적용 기능을 위한 컴포넌트 다이어그램이다. (그림 1)에 나타나 있는 컴포넌트들의 기능은 다음과 같다.

- PatternAddIn.mnu: 메뉴 파일로써, '.mnu' 확장자를 사용하는 유효한(valid) XML 파일이다. 도구의 메뉴 확장으로 보여지게 된다.
- PatternAddIn.chm: 사용자 정의형 패턴에 대한 도움말 파일이다.
- PatternAddIn.dll: 사용자 정의형 패턴 적용 기능을 가진 DLL 파일이다.
- PATADDIN.LNG: 언어 지원 파일이다.
- <<folder>> : 패턴 저장소(PatternRepository)로써, 본 폴더 하위로 폴더 구조가 계층적으로 구성될 수 있고 그 안에 관련 파일들이 위치하게 된다. 하위의 폴더들은 패턴의 분류가 된다.



(그림 2) 사용자 정의형 패턴 적용을 위한 클래스 설계

(그림 2)는 사용자 정의형 패턴 기능을 위한 클래스 설계 결과이다. 설계 구조는 패턴 저장소(Repository)의 사용자 정의형 패턴 파일을 분석하여 도구에 반영할 수 있도록 설계 되었다. (그림 2)의 클래스 구조를 보면 GoF 의 Composite 패턴[3]을 사용함으로써, 패턴 폴더가 다른 패턴 폴더를 포함할 수도 있고, 패턴 파일도 포함할 수 있도록 설계 되었다.

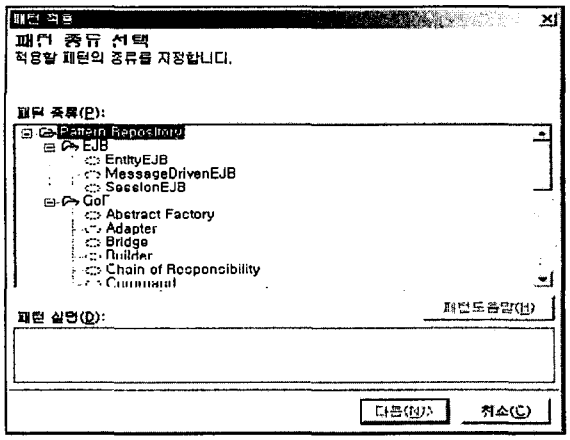
사용자 정의형 패턴 파일은 (표 1)의 DTD 구조로 XML 형식으로 작성해서 패턴 저장소(PatternRepository)의 하위 폴더를 생성해서 저장해야 하며, '.pat' 확장자를 사용하는 유효한(valid) XML 파일이어야 한다.

```
<!--========== Document Structure ==========-->
<!ELEMENT PATTERN(DESCRIPTION, PARAMETERS)>
<!ATTLIST PATTERN
  version      CDATA      #REQUIRED
  name         CDATA      #REQUIRED
  help         CDATA      #IMPLIED
>
<!--========== PATTERN Element ==========-->
<!ELEMENT DESCRIPTION (#PCDATA)>
<!ELEMENT PARAMETERS (PARAMETER+)>
<!--========== PARAMETER Element ==========-->
<!ELEMENT PARAMETER(DESCRIPTION, DEFAULTVALUE?,
LITERALS?)>
<!ATTLIST PARAMETER
  id           CDATA      #REQUIRED
  caption      CDATA      #REQUIRED
  type         CDATA      #REQUIRED
  elementKind  CDATA      #IMPLIED
>
<!--========== Description Element ==========-->
<!ELEMENT DESCRIPTION (#PCDATA)>
<!--========== DefaultValue Element =======-->
<!ELEMENT DEFAULTVALUE (#PCDATA)>
<!--========== LITERALS Element ==========-->
<!ELEMENT LITERALS (LITERAL*)>
<!--========== LITERAL Element ==========-->
<!ELEMENT LITERAL (#PCDATA)>
```

(표 1) 사용자 정의형 패턴 파일의 DTD 구조

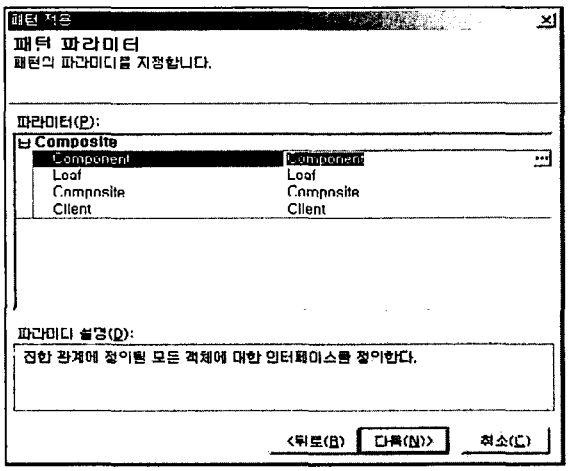
4 구현결과

본 장에서는 구현 결과를 실행 예를 중심으로 기술한다. (그림 3)는 도구에 적용할 패턴을 선택 하기 위한 화면이다. 패턴 종류는 트리 뷰 형태로 보여진다. GoF 디자인 패턴과 EJB 패턴[4]이 미리 정의 되어 있는 것을 알 수 있다. 사용자 정의형 패턴을 정의하게 되면 (그림 3)에서 보이는 것처럼 패턴 목록이 생성되어서 도구에 사용할 수 있게 된다.



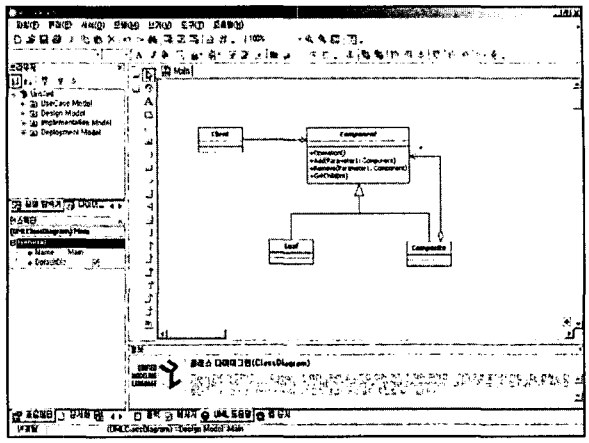
(그림 3) 패턴 종류 선택 대화 상자

(그림 4)은 사용자 정의형 패턴을 도구에 적용하기 위해서 미리 정의된 패턴 파라미터의 값을 수정하거나 다른 클래스들을 연결하기 위한 대화 상자이다. 예를 들어 패턴을 사용해 리팩토링을 할 때 기존 클래스나 더 나은 패턴으로 대체할 수도 있다.



(그림 4) 패턴 파라미터 대화 상자

(그림 5)는 패턴이 도구에 적용된 화면 이다. 사용자는 사용자 정의형 패턴 파일만 정의해서 패턴 저장소(PatternRepository)에 저장만 하면 쉽게 도구에서 적용할 수 있게 된다.



(그림 5) 도구에 패턴이 적용된 화면

5. 결론

본 논문에서는 UML 모델링 도구에 사용자 정의형 패턴을 쉽게 적용할 수 있도록 하는 기능을 설계하여 구현한 결과에 대하여 논하였다. 사용자 정의형 패턴의 적용을 보다 쉽고 편리하게 도구에 적용함으로써 더 나은 객체 설계를 얻기 위한 원리(principle)와 패턴(pattern) 등을 사용할 수 있게 되었다.

참고문헌

[1] Martin Fowler, UML Distilled: Applying the Standard Object Modeling Language, Addison-Wesley, 1997.

[2] Plastic Software Inc, http://www.plasticsoftware.com.

[3] Erich Gamma, Richard Hel, Ralph Johnson, John Vlissides. "Design Patterns : Elements of Reusable Object-Oriented Software", Addison Wesley, 2002.

[4] Richard Monson-Haefel, "Enterprise JavaBeans, 2nd Edition", O'Relly, 2000.