2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

XML기반의 디자인패턴 관리시스템 설계 및 구현

서영준*, 최한용, 송영재
경희대학교 전자계산공학과

e-mail:cionblue@case.kyunghee.ac.kr

Design and Implementation of Design Pattern Management System based on XML

Young-Jun Seo*, Han-Yong Choi, Young-Jae Song
Dept of Computer Engineering, KyungHee University

요 약

최근 반복적 설계에 대한 해결방안으로 제시되고 있는 디자인패턴을 체계적으로 분류, 공유하여 사용하려는 연구가 진행되고 있으나, 시스템 설계자 개개인이 제시한 패턴은 공유 대상에서 제외되고 있다. 또한, 특정 CASE 도구를 사용하여 모델링한 구조는 컴포넌트화되더라도 특정 도구나 플랫폼에 독립적으로 사용될 수 없었다. 본 논문에서는 기존 디자인패턴뿐만 아니라 사용자들에 의해 새로이 제시된 패턴을 모델링하는 패턴 에디터를 이용하여 설계정보를 도구와 플랫폼에 독립적인 XML코드로 컴포넌트화 할 수 있는 XML 기반의 디자인패턴 관리시스템을 설계, 구현하였으며, 효과적인 패턴 검색을 위해 Spreading Activation 검색 방법을 사용하였다.

1. 서 론

객체지향 패러다임을 적용한 시스템 설계시 반복적 설계가 자주 나타나게 되며, 이는 시간과 비용의 낭비와 유지보수의 어려움을 유발한다. 즉, 디자인 재사용성을 향상시키기 위한 방법이 요구되고 있으며, 이에 대한 해결방안으로 디자인패턴이 제시되고 있다. 그러나 패턴의 체계적인 분류, 공유가 이루어지지 않아 적당한 패턴을 찾지 못하거나, 참조되지 못하는 경우가 많았다. 최근 이에 대한 연구가 진행되고 있으며, 디자인패턴의 구조를 컴포넌트화하여 시스템 설계시 사용하기 위한 방안도 연구되고 있다[6]. 그러나, 기존 연구에서 제안한 시스템은 Gamma[1]의 디자인패턴과 같은 몇몇 검증된 패턴들만을 공유하여 사용하고 있으며, 개개인의 시스템 사용자들이 새로이 제안하는 디자인패턴들을 공유하여 사용하려는 시도가 이루어지지 않고 있다. 또한, 시스템 자체에서 디자인패턴을 모델링하지 못하고, 특정 CASE 도구를 사용하므로, 기존의 시스템에서 제공하고 있는 컴포넌트화된 디자인패턴 구조는 특정 도구나 플랫폼에 의존적일 수 밖에 없었다. 디자인패턴의 검색시에는 기존 검색 방법으로는 질의어와 직접 연결되어 있는 패턴만을 찾을 수 있으므로, 정확히 찾고자 하는 패턴을 모를 때는 정확한 결과를 얻을 수 없었다. 따라서, 질의어에는 연관되어 있지만 정확히 매치되지 않는 항목을 검색할 수 있는 방법을 사용해야 한다. 이와 같이, 시스템 설계자가 제안한 디자인패턴을 등록, 공유, 검색할 수 있으며, 시스템내에서 디자인패턴을 직접 모델링하여, 그 구조를 특정 도구와 플랫폼에 독립적인 포맷으로 컴포넌트화할 수 있는 연구가 필요하다.

본 논문에서는 이러한 디자인패턴들을 체계적으로 분류하여 데이터베이스화한 뒤 공유, 관리할 수 있는 시스템을 설계, 구현하였다. 또한, UML로 표기된 디자인패턴은 XMI 스펙에 의해 XML 파일로 컴포넌트화하여 DB에 저장되며, 이를 XMI를 지원하는 모든 CASE 도구에서 시스템 설계시 사용할 수 있도록 하였다. 즉, 컴포넌트화된 디자인패턴은 특정 도구와 플랫폼에 독립적으로 사용할 수 있으며, 개

발하는 시스템의 안정성과 성능향상에 기여하게 된다. 디자인패턴의 효율적인 검색을 위해, 질의어에는 연관되어 있지만 정확히 매치되지 않는 패턴을 검색할 수 있는 Spreading Activation 검색 방법[7]을 사용한다.

2. 관련 연구

2.1 디자인패턴

디자인패턴은 시스템 설계시에 자주 발생하는 문제들에 대한 "재사용 가능한 해결책"이라 할 수 있다[1]. 경험 많은 소프트웨어 엔지니어의 해법들을 정리해서 이름을 부여하고 디자인적인 추상성을 주어 패턴화 시키면 시스템 디자인시에 디자인 재사용성을 제공할 수 있다. 현재 디자인패턴은 표준이 없으나 Gamma[1]가 제안한 디자인패턴이 가장 기본적인 디자인패턴으로 인정받고 있으며, 매 년 컨퍼런스를 통해 발표, 검증되고 있다. 따라서 본 논문에서는 위에 언급된 디자인패턴들을 표준 패턴인 공유패턴(Public Pattern)이라 정의하여 사용하며, 사용목적에 따라 기본, 생성, 분류, 구조화, 행위 패턴으로 분류하였다[3]. 또한, 시스템사용자 개개인이 제안한 패턴인 개인 패턴(Private Pattern)도 같은 분류 방식을 택했다.

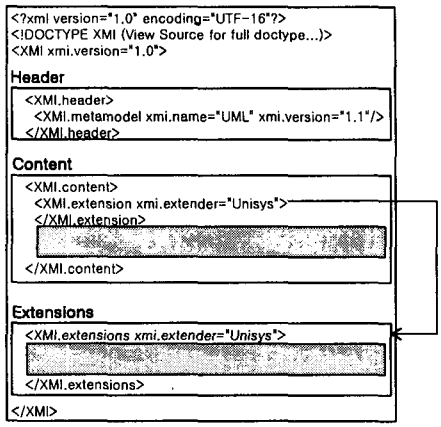
2.2 XML(eXtensible Markup Language)

XML[4]은 1996년 W3C에서 제안된 웹 상에서 구조화된 문서를 전송 가능하도록 설계된 표준 마크업 언어이며, 모든 플랫폼, 운영체제 환경에서 실행할 수 있다. XML로 저장된 데이터의 검증은 특정 컨텐트를 표현하는 태그와 속성을 정의한 DTD(Document Type Definition)를 통해 할 수 있다. DTD는 많은 XML 문서에서 반드시 필요한 것은 아니지만, XML 문서가 규칙에 맞는지 확인하기 위해서는 DTD를 반드시 포함해야 한다.

2.3 XMI(XML MetaData Interchange)

XMI[5]는 1999년 3월 OMG에서 웹기반의 XML과 OO기반의 UML의 장점을 결합시킨 새로운 open산업의 표준으로 채택되었다.

XMI는 크게 (그림 1)과 같이 세 부분으로 나뉘어진다. Header 요소에는 전송되는 메타데이터에 대한 다양한 정보와 메타데이터를 위한 메타모델(Class, Attribute, Operation 등)을 확인하는 XML요소를 기술하며, Content 항목에는 전송되는 실제 메타데이터를 포함하는 XML요소를 기술한다. 마지막으로, Extensions 요소에는 메타모델을 확장하는 메타데이터를 포함하는 XML요소를 기술한다.



(그림 1) XMI 스펙 구조

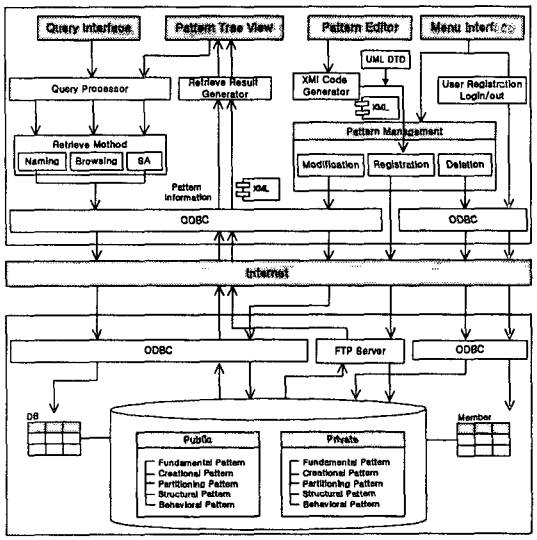
XMI 스펙에서는 XMI 문서의 검증을 위해 "UML Semantics ver 1.1"[2]에서 정의한 UML 구문과 의미를 바탕으로 UML DTD를 구성하였으며, 이에 맞추어 UML 메타모델을 기술하였다.

3. 디자인패턴 관리시스템 설계 및 구현

3.1 시스템 설계

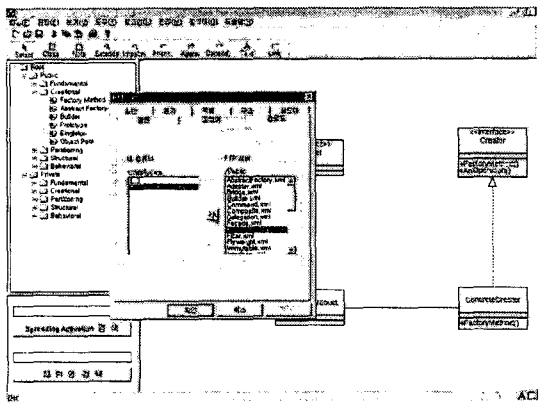
본 논문에서 구현한 디자인패턴 관리시스템(DPMS)은 하부 저장시스템으로 MS-SQL Server 6.5를 사용하였으며, MFC를 사용해 구현하였다. (그림 2)는 구현한 디자인패턴 관리시스템의 전체 시스템 구조를 보여준다. 디자인패턴 관리시스템은 PE(Pattern Editor), QI(Query Interface), PTV(Pattern Tree View), PM(Pattern Management)으로 구성되어 있으며, 각각의 역할은 다음과 같다. PE는 CASE 도구의 클래스 다이어그램을 구현하였으며, 제공되는 클래스, 관계, 노트, 텍스트 툴을 사용해 디자인패턴을 설계하게 되며 XML코드로 변환 저장한다. QI에서는 디자인패턴을 이름 검색, Spreading Activation 방법으로 검색이 가능하며, 그 결과는 PTV에서 재구성되어 보여진다. PTV는 분류된 디자인패턴을 트리 형태로 보여주며, 디자인패턴들의 세부정보 확인과 함께 브라우징 검색을 지원한다. 또한, XML 컴포넌트 형태의 패턴 구조도 다운받을 수 있다. PM은 새로운 디자인패턴을 등록, 수정, 삭제할 수 있으며, (그림 3)은 디자인패턴을 PE에서 UML로 모델링한 후 서버의 DB에 등록하는 화면이다. 관리할 수 있는 디자인패턴의 범위는 DB의 개인패턴 영역

의 로그인한 사용자 영역으로 국한한다. 공유패턴 영역의 경우에는 관리자가 직접 등록, 관리하도록 하였다. (그림 4)는 검색을 통해 찾은 디자인패턴을 그 세부정보를 확인하는 화면이며, 그 구조를 다운로드 받을 수 있다.

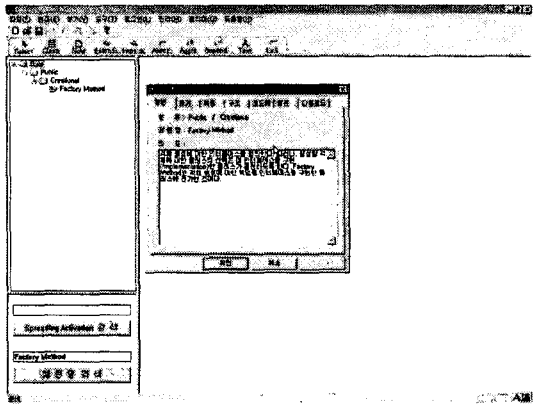


(그림 2) 시스템 구성도

서버에서는 디자인패턴들을 사용목적에 의해 분류하여 DB를 구성하게 되며, 클라이언트에서는 서버의 DB에서 검색을 통해 원하는 디자인패턴을 찾아 그 세부 정보를 알 수 있다. 선택된 디자인패턴의 구조는 클라이언트로 다운받아 CASE 도구를 사용한 시스템 설계시 사용할 수 있다. 또한, 개개인의 시스템 사용자가 제안한 디자인패턴은 UML로 모델링하여 컴포넌트화한 후 개인 패턴으로 서버의 DB에 등록, 관리된다.



(그림 3) 패턴의 등록 및 XML파일의 업로드



(그림 4) 패턴의 검색 및 패턴정보 확인

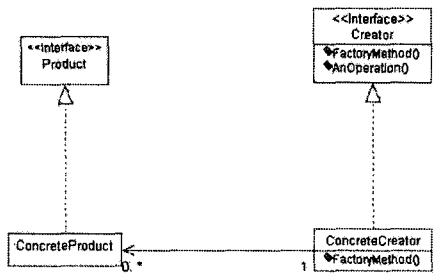
3.2 검 색

세 가지의 검색방법을 제공하고 있으며, 정확한 디자인패턴명을 이용한 검색 방법인 이름검색과 디자인패턴들을 트리 형태로 브라우징하면서 검색하는 브라우징 검색, 그리고 Spreading Activation 검색 방법으로 나뉘어 진다.

Spreading Activation 검색 방법[7]은 사용자가 원하는 디자인패턴을 분명히 표현하지 못하거나, 정확히 매치되지 않는 경우에 사용한다. 각각의 디자인패턴을 등록시 연관된 질의어들을 설정하고, 질의어는 디자인패턴들 중 공통된 특성을 갖는 어휘로 선택한다. 이를 위해 활성값을 이용하여 유사도를 측정하게 되며, 활성값이 큰 항목순으로 확장 검색을 한다.

3.3 XML코드 생성

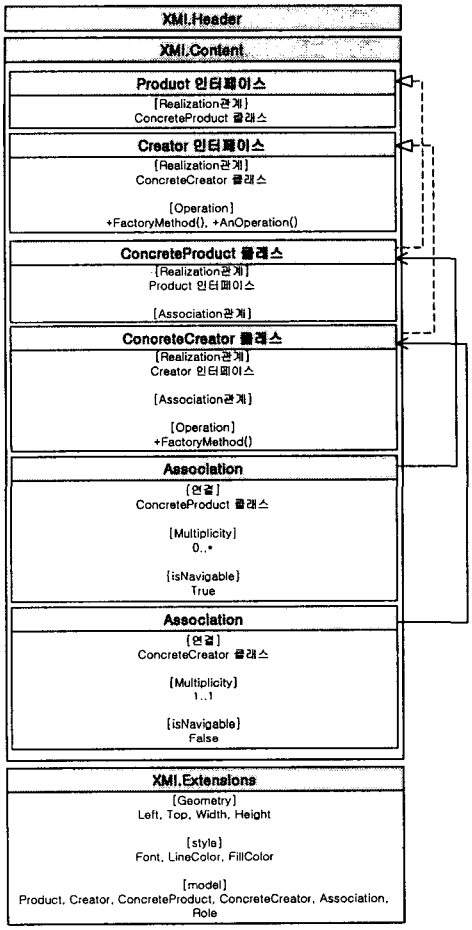
(그림 5)는 객체 생성에 관련된 디자인 패턴 중 하나인 "Factory Method 패턴"을 나타내고 있다.



(그림 5) Factory Method 패턴

이 패턴은 객체 생성에 대한 책임을 인터페이스를

구현한 클래스에 전가함으로서 객체 생성에 대한 의존성을 없애 유지, 보수가 높은 시스템을 설계할 수 있다.



(그림 6) Factory Method패턴의 XML 표현 구조

Factory Method 패턴은 (그림 6)과 같이 XML문서로 코드화할 수 있는데, <XMI.Content> 항목에서는 Factory Method 패턴의 메타모델들 즉, Product, Creator 인터페이스, ConcreteProduct, ConcreteCreator 클래스, Realization, Association 관계에 대한 메타데이터를 기술하게 되며, Relationship관계인 경우, 상호연결된 메타모델에 그 관계를 덧붙여 기술한다. <XMI.Extensions>에서는 <XMI.Content>에서 기술한 각각의 메타모델들에 대한 CASE 도구에서의 Presentation 정보, 즉 위치, 스타일을 기술한다. (그림 7)은 (그림 6) 구조에서 ConcreteCreator 클래스를 UML DTD를 사용해 XMI로 코드화한 것이다.

```
<Foundation.Core.Class xmi.id = 'S.10006'>
  <Foundation.Core.ModelElement.name>ConcreteCreator</Foundation.Core.ModelElement.name>
  <Foundation.Core.ModelElement.visibility xmi.value = 'private'/>
  <Foundation.Core.Classifier.specification>
    <Foundation.Core.Interface xmi.idref = 'S.10002'/> <!-- Creator -->
  </Foundation.Core.Classifier.specification>
  <Foundation.Core.Classifier.associationEnd>
    <Foundation.Core.AssociationEnd xmi.idref = 'G.6'/> <!-- {39A4DD250008} -->
  </Foundation.Core.Classifier.associationEnd>
  <Foundation.Core.Classifier.feature>
    <Foundation.Core.Operation xmi.id = 'S.10007'>
      <Foundation.Core.ModelElement.name>FactoryMethod</Foundation.Core.ModelElement.name>
      <Foundation.Core.ModelElement.visibility xmi.value = 'public'/>
      <Foundation.Core.Feature.ownerScope xmi.value = 'instance'/>
    </Foundation.Core.Operation>
  </Foundation.Core.Classifier.feature>
</Foundation.Core.Class>
```

(그림 7) ConcreteCreator 클래스의 XML표현 예

4. 결 론

본 논문에서는 시스템 설계시 재사용성을 향상시키는 디자인패턴을 공유, 관리할 수 있는 DPMS를 설계, 구현하였다. 공유된 디자인패턴은 Spreading Activation 검색방법을 통해, 사용하고자 하는 디자인패턴이 질의어에 직접 연결되어 있지 않아도 찾을 수 있을 뿐만 아니라, UML로 표현된 디자인패턴을 XMI스펙에 의해 XML코드로 변환하여 컴포넌트화함으로서, 특정 도구와 플랫폼에 무관하게 XMI를 지원하는 CASE 도구를 사용한 설계에서 사용할 수 있다. 향후 연구과제로는 개인패턴 영역의 패턴 검증과 공유패턴 영역으로의 이전에 대한 명확한 기준을 설정하여야겠다.

5. 참고문헌

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

[2] OMG, UML Semantics version 1.1, http://www.omg.org/uml/, 1997.

[3] Mark Grand, "Pattern in Java", Volume 2, Wiley, 1998.

[4] W3C, Extensible Markup Language(XML) 1.0, REC-xml-19980210, http://www.w3.org/TR/REC-xml/, 1998.

[5] OMG, XML MetaData Interchange(XMI) 1.0, http://www.omg.org/, 1999.

[6] 김행곤, 차정은, 김지영, "웹 상에서 설계 패턴 라이브러리에 기반한 재사용 시스템 구현", 한국정보과학회 추계학술발표논문집, 제26권 2호, pp.551-553, 1999.

[7] 한정수, 송영재, "개선된 SARM을 이용한 객체지향 부품 재사용 시스템", 정보처리 논문지, 제7권, 제4호, pp.1092-1102, 2000, 4.