2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

XML을 이용한 디자인 패턴 활용구조

김운용*, 최영근*
*광운대학교 컴퓨터과학과

e-mail:wykim@cs.kwangwoon.ac.kr

A Framework for using Design Pattern with the XML

Woon-Yong Kim*, Young-Keun Choi*
*Dept of Computer Science, Kwang-Woon University

요약

디자인패턴은 디자인 경험을 표현하기 위한 새로운 메커니즘으로 미래의 유사 상황에서 다시 적용될 수 있는 과거에 잘 정의된 설계에 대한 정보를 기록하는 것이며 소프트웨어 개발 설계에서 재사용성을 증가시킨다. 그러나 광범위한 관심과 활용에도 불구하고, 패턴명세와 활용은 주로 개발자의 수작업을 통해 이루어지기 때문에 일관된 형태의 분석과 활용이 어렵다. 이로 인해 오류 발생 빈도를 높일 뿐 아니라 프로그램 개발이 어렵고 많은 시간을 필요로 한다. 따라서 본 논문에서는 디자인패턴을 구조화된 전자문서로 표현하기 위한 XML 표기방법과 디자인패턴 활용시스템 구조를 제시한다. 또한 이러한 표기법과 활용구조를 통해 소스코드 자동생성 지원 시스템을 제시하고, 적용 예를 보이고자한다. XML을 이용한 구조화된 문서활용은 소스코드 생성시 사용자들에게 더 작은 코드를 작성하게 만들고, 더 안정된 시스템을 구축할 수 있게 한다. 또한 XML의 다양한 기술에 접목함으로써 패턴 활용을 극대화할 수 있다.

1. 서론

디자인패턴은 객체지향 소프트에서 경험을 활용하기 위해 제안된 기술이다[3]. 이러한 것은 소프트웨어 시스템을 구성하는 빌딩블록으로서 크나큰 역할을 담당할 뿐 아니라 소프트웨어 개발 설계에서 재사용을 증가하기 위한 수단으로 활용된다[4]. 그러나 이러한 광범위한 관심과 활용에도 불구하고, 기존의 패턴명세와 활용은 주로 개발자의 수작업으로 이루어지므로, 오류발생 빈도를 높이고, 프로그램 개발에 많은 어려움과 시간이 필요하다. 효율적인 패턴 적용방법을 위한 여러 연구들이 진행되어왔는데 대부분은 주로 메타언어 및 패턴언어를 이용해 패턴을 구성하고 이를 프로그램에 적용시키는 방법에 관한 것들로서 대부분 추상적이며 실제 프로그램에 적용시키기 위해서는 많은 부가적인 작업이 요구된다. 또한 디자인 패턴의 활용 방법에 있어서 사용자에 대한 고려가 부족하며, 문서로서의 의미보다는 실제 구현 부분에 치중하고 있다. 본 논문에서는 산업 표준화 문서형태인 XML을 이용해 디자인 패턴의 표현방법을 제시하고 이를 바탕으로 소스코트 생성 지원 시스템과 적용 예를 보여준다. XML을 이용해 구조화된 패턴 문서는 컴퓨터에서 이용 가능한 구조적 문서역할과 프로그램 소스코드 생성에 쉽게 적용시킬 수 있으며 문서의 검증 및 다양한 XML기술에 활용될 수 있을 것이다.

본 논문의 구성은 다음과 같다. 우선 2장에서 관련연구를 소개하고, 3장에서 XML을 이용한 디자인 패턴의 활용구조와 디자인 패턴에 대한 XML 문서 표기법을 제시한다. 4장에서는 이러한 표기법을 통한 XML 기반의 패턴문서 생성을 보이고, 5장에서는 생성된 XML 문서형태의 패턴을 이용한 소스코드 생성 예를 보인다. 마지막 6장에서는 결론을 내린다.

2. 관련 연구

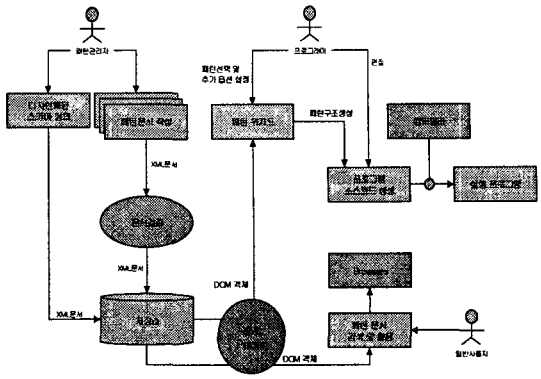
기존의 디자인 패턴 명세의 활용에 대한 연구로 Partha는 패턴에 의해 나타나는 행위자가 어떻게 각각의 협력자 작업에 의무를 부가하는 데에 필요한 정규화방법을 제시하였다[8]. 또한 Klarlund & Koistinen에 의한 또 다른 비슷한 작업으로 Parse Tree를 이용한 디자인 제약의 인증방법을 제시하였다[7]. 그러나 이러한 표현방법들은 구현에 대한 상세한 기술이나 접근이 어렵고, 각각 서로 다른 특수한 패턴 표현언어를 사용함으로써, 이해하거나 분석하기가 어렵다. 또한 문서로써의 활용능력을 담지 못한다. 또 다른 접근방법으로 디자이너에 의해 제공되는 정보를 이용해 패턴의 코드를 자동으로 생성하기 위한 도구의 설계이다[2,1,9,3]. 이러한 접근방법은 패턴을 이용한 소스코드의 생성을 기술하고있으나 패턴에 대한 정보가 도구에 의존해서 존재함으로 활용 성이 떨어진다. 디자인 패턴을 프로그램에 보다 효율적으로 활용하기 위해서는 작성된 문서가 쉽게 분석되고 적용될 수 있어야한다. 이러한 특징을 만족시키기 위해 본 논문에서는 산업표준화 문서인 XML을 기반으로 디자인패턴에 대한 구조화된 표기법을 제시하고 그 활용 예를 보인다.

3. 디자인 패턴에 대한 XML문서 표현

디자인 패턴의 표기방법은 자동화도구의 기본 틀을 형성하는 중요한 수단이며, 패턴 문서 작성 및 활용의 기본 요소가 된다. 본 장에서는 디자인 패턴을 소스코드에 활용하기 위한 디자인패턴의 활용구조를 제시하고, 이 구조에 이용되는 XML기반 패턴 문서 표기법을 제시한다.

3.1 XML을 이용한 디자인 패턴의 활용 구조

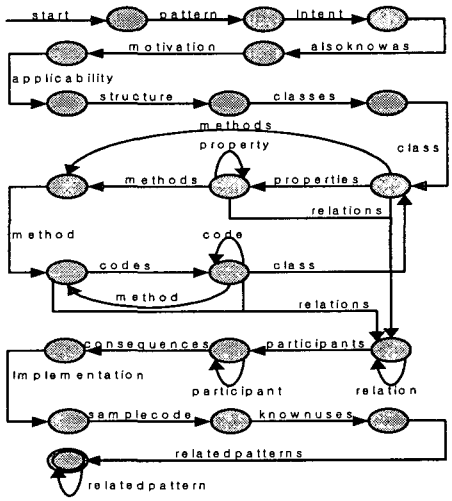
이 절에서는 XML형태의 디자인 패턴문서를 소스코드 자동생성에 활용하는 수행과정에 대한 전체구조를 제시한다. 디자인 패턴문서의 가장 중요한 역할중의 하나는, 사용자가 프로그램 개발 시 어느 정도 쉽게 적용할 수 있는가에 달려 있다. (그림 1)은 XML형태의 패턴에 대한 문서작성으로부터 그 활용방법까지에 대한 전체적인 구조를 나타낸다. 패턴문서의 작성 및 활용은 패턴관리자, 프로그래머 및 일반사용자로 구성될 수 있다. 그림에서 보듯이 작성된 패턴문서는 다양한 시각에서 활용될 수 있는 구조를 가진다.



(그림 1) XML을 이용한 디자인 패턴 활용 구조

3.2 디자인 패턴문서에 대한 XML문서 구조

디자인패턴은 디자인 문제에 대한 추상적인 해결책이다. 이 패턴은 클래스의 일반적인 구조, 그들의 역할, 협력과정 그리고 관계를 표현한다. 아직까지 패턴에 대한 표준적인 정의는 없고, 패턴에 대한 몇 가지 범주가 존재한다[4,5,10]. 이 논문에서는 [4]의 표기법을 따른다. [4]에서 디자인 패턴을 표현하기 위해 13가지의 요소를 가진다. ***Name, Intent, Also Know As, Motivation, Applicability, Structure, Participants, Collaborations, Consequences, Implementation, Sample Code, Known Uses, Related Pattern.*** 이러한 요소들은 크게 패턴의 사용용도 및 특징을 설명하는 텍스트와 패턴의 구조와 행위를 나타내는 부분으로 나누어질 수 있다. 이러한 형태를 DFA(Deterministic Finite Automation)을 통해 제시한면 (그림 2)와 같다.



(그림 2) 디자인패턴을 위한 DFA

이 DFA의 구성은 순차적으로 [4]에서 표현한 내용을 순서적으로 적용한 결과이다. 여기에서 실제 소스코드에 적용될 부분은 structure 시작되는 부분이다. 디자인패턴의 구조 및 행위에 대한 DFA는 크게 classes 부분과 relations부분으로 나누어진다. classes부분에서는 properties와 methods 들과 패턴의 기능을 수행하기 위한 codes로 구성되고, relations부분에서는 패턴에서 이용되는 클래스의 관계를 설정한다. 이 관계는 크게 생성, 참조, 집합, 상속관계를 나타낼 수 있다. 이러한 DFA와 패턴을 이용해 DTD(Document Type Definition)을 정의하면 (그림 3)과 같다.

```
<!ELEMENT pattern (intent,alsoknowas,motivation,applicability,structure,classes,
    relations,participant,consequences,samplecode,knownuses,relatedpatterns)
<!ATTLIST pattern name CDATA #REQUIRED>
<!ELEMENT intent(#PCDATA)>
<!ELEMENT alsoknowas(#PCDATA)>
...
<!ELEMENT structure(classes,relations)>
<!ELEMENT classes(class+)>
<!ELEMENT class(properties,methods)>
<!ATTLIST class name CDATA #REQUIRED>
<!ATTLIST class scope(public|protected|private) "public">
<!ATTLIST class abstract(true|false) #REQUIRED>
<!ELEMENT properties(property+)>
<!ELEMENT property (#PCDATA)>
<!ATTLIST property scope(public|protected|private) "public">
<!ATTLIST property abstract(true|false) #REQUIRED>
<!ATTLIST property type CDATA #REQUIRED>
<!ELEMENT methods(method+)>
<!ELEMENT method(codes?)>
<!ATTLIST method name CDATA #REQUIRED>
<!ATTLIST method scope(public|protected|private) "public">
<!ATTLIST method abstract(true|false) #REQUIRED>
<!ATTLIST method arguments CDATA >
<!ATTLIST method return_type CDATA >
<!ELEMENT codes(code+)>
<!ELEMENT code(CDATA)>
<!ATTLIST code language CDATA #REQUIRED>
<!ELEMENT relations(relation+)>
<!ELEMENT relation EMPTY>
<!ATTLIST relation type(create|reference|aggregate|inheritance) #REQUIRED>
<!ATTLIST relation from CDATA #REQUIRED>
<!ATTLIST relation to CDATA #REQUIRED>
<!ATTLIST relation from_cnt(one|more_then_one | more_then_zero) "one">
<!ATTLIST relation to_cnt(one|more_then_one | more_then_zero) "one">
<!ELEMENT participants(participant+)>
<!ELEMENT participant(#PCDATA)>
<!ATTLIST participant name CDATA #REQUIRED>
...
<!ELEMENT relatedpatterns(relatedpattern+)>
<!ELEMENT relatedpattern EMPTY>
<!ATTLIST participant name CDATA #REQUIRED>
```

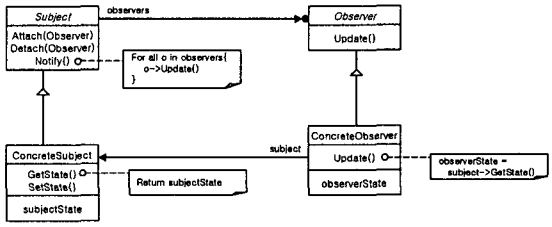
(그림 3) 디자인패턴을 표현하기 위한 DTD

4. 정규화 디자인 패턴 XML문서 활용

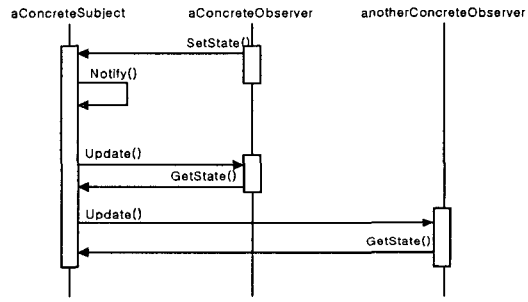
디자인 패턴 활용 적용 과정을 다루기 위해, 본 논문에서는 옵저버(Observer) 패턴[4]을 이용하여 제시된 DTD을 활용한 적용 예를 보인다. 옵저버 패턴은 객체가 1 대 다의 관계를 가지고서 한 객체의 상태 변화에 다른 모든 객체가 영향을 받는 기능을 가진 패턴이다. XML문서로 이 패턴을 표현하기 위해 먼저 text로 기록된 문서는 그 내용을 활용하여 다음 형태처럼 구성할 수 있다.

```
<intent>  <!-- intent item -->
  Define a one-to many dependency ...
</indent>
```

또한 소스코드생성과 관련된 Structure 와 Collaborations 부분은 패턴문서[4]에 제시된 클래스 다이어그램과 상호 협동다이어그램을 통해 얻을 수 있다. 이 옵저버 패턴과 관련된 클래스다이어그램과 협동다이어그램은 (그림 4)(그림5)에서 볼 수 있다.



(그림 4) 옵저버 패턴 클래스 다이어그램



(그림 5) 옵저버 패턴의 협동다이어그램

전체 구성된 옵저버 패턴에 대한 XML문서는 (그림 6)과 같다.

```
<?xml version = "1.0"?>
<pattern name = "Observer">
 <intent>  Define a one-to-many dependency between objects ...  </intent>
 ...
 <structure>
  <classes>
   <class name= "Subject" scope="public" abstract="true">
    <properties>
     <property scope="public" type="Vector">
      observers
     </property>
    </properties>
    <methods>
     <method name="Attach" scope="public" abstract="true"
                    arguments="Observers _obs"> </method>
     ...
     <codes>
      <code language="java">
        for (Enumeration e = observers.elements() ; e.hasMoreElements() ;){
          ((Observer)e.nextElement()).Update();
        }
      </code>
     ...
  </classes>
  <relations>
   <relation type="aggregate" from="Observer" from_cnt="more_then_one"
                                  to="Subject" to_cnt="one"> </relation>
    ...
 </structure>
 <participants>
  <participant name="Subject"> knows its observers. ... </participant>
 ...
 </participants>
 ...
 </pattern>
```

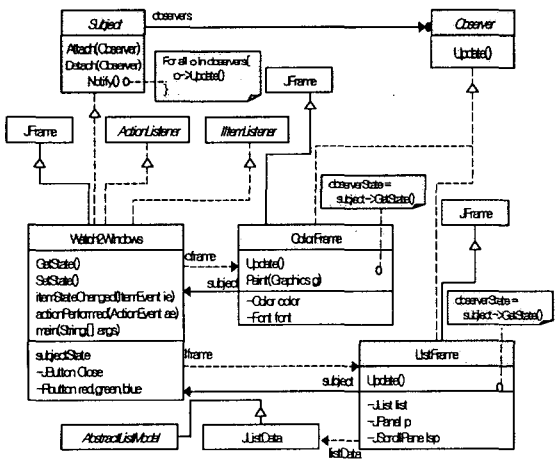
(그림 6) 옵저버 패턴에 대한 XML문서

5. XML문서형태의 디자인 패턴 활용 예

이 장에서는 XML문서형태의 디자인 패턴 문서를 활용하여 자동화된 소스코드 생성방법과 활용 구조에 대해 기술한다. 패턴 활용 및 적용 구조는 (그림 1)에서 보았다. 그림에서 보듯이 구성된 스키마와 패턴문서는 문서검증을 통해 저장소에 저장되어지고, 저장된 문서는 XML Parser를 이용해 패턴 위저드에서 사용자의 선택사항과 조합하여 프로그램 소스코드를 생성하고 활용되어진다.

5.1 XML기반 디자인 패턴을 이용한 어플리케이션 개발

어플리케이션 개발시 사용자가 디자인 패턴을 적용시키고자한다면, 그때 사용자는 패턴위저드를 통해 어플리케이션 구성에 필요한 정보입력을 통해 해당 어플리케이션에 원하는 패턴을 추가시킬 수 있다. 패턴을 어플리케이션에 적용시키기 위해 필요한 것은 어플리케이션에 필요한 클래스를 옵저버 패턴에 정의된 클래스와 연관시킴으로써 어플리케이션 클래스에 원하는 패턴기능을 추가시킬 수 있다.



(그림 7) 어플리케이션 적용 예

(그림 7)은 옵저버 디자인패턴이 적용된 어플리케이션을 보여준다. XML문서화된 패턴문서의 정보를 이용해 소스코드 자동생성부분을 통해 그림에서 진하게 표시된 글과 클래스들이 생성된다. 나머지부분의 사용자들에 의해 완성된다.

6. 결론

본 논문에서는 디자인 패턴의 활용 성을 높이기 위한 방법으로 디자인패턴문서에 대한 XML 표현 방법과 이 XML로 구성된 패턴 문서를 이용한 자동화 소스코드 생성 구조를 제시하고, 이를 이용한 활용 예를 보였다. XML문서 기반의 패턴문서가 가지는 장점은 패턴에 대한 구조화되고 일관된 형태의 구성을 제공함으로써 분석 및 유지 보수의 효율성을 증대시키고 사용자들이 보다 쉽게 어플리케이션에 패턴을 활용할 수 있도록 해주는 것이다. 또한 프로그램 개발의 효율성을 증가시키고 오류발생요인을 줄일 수 있게 함으로 비용 절감효과를 가져올 수 있다. 이러한 XML기반 패턴문서는 전자문서로써의 기능을 담고있기 때문에 다양한 XML기술에 활용될 수 있을 것이다.

참고문헌

[1] A. Eden, J. Gil and A. Yehudai. Automating the Application of design patterns. page 44-46. Report on Object Analysis and Design ROAD. May 1997.

[2] A. Eden, J. Gil and A. Yehudai. Precise specification and automatic application of design patterns. In Automated Software Engineering, 1997.

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

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

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

[6] J. Coplien and D. Schmidt, editors. Pattern Languages of Program Design Addison-Wesley, 1995

[7] N. Klarlund and J. Koistinen "Formal Design Constraints". Proceedings of OOPSLA. 1996.

[8] P. Pal, "Law-Governed Support for Realizing Design Patterns". Proceedings of TOOLS USA 17. 1995.

[9] S. Yacoub, H, Xue, and H. Ammar, 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

[10] W. Tichy, Essential Software Design Patterns. University of Karsruhe. 1997
http://wwwipd.ira.uka.de/~tichy/patterns/overview.html,