2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

XML기반의 컴포넌트 명세화

김영미*, 임도연**,오수열*
*목포대학교 컴퓨터공학과 **전북대학교 컴퓨터과학과
e-mail:dudal@apollo.mokpo.ac.kr
syoh@chungkye.mokpo.ac.kr

XML-Based Component Specification

Young-Mi Kim*, Do-Yeon Yim**, Su-Yul Oh*
*Dept of Computer Engineering, Mokpo University
**Dept of Computer Science, Chonbuk University

요약

컴퓨터와 인터넷 사용의 대중화 시대가 열리면서 소프트웨어의 구조도 점차 복잡해짐은 물론 크기도 방대해지고 있으나 S/W 개발 기술 발전 속도는 매우 더디어 소프트웨어 개발과 유지보수 비용의 증가로 인해 소프트웨어 위기 문제가 발생하게 되었다. 이에 대한 대안으로 소프트웨어 생산성, 품질, 효율성을 높이기 위해서 부품화와 조립의 특성을 지닌 컴포넌트 기반 개발 방법이 제시되었다. 컴포넌트 명세에 있어 기존에 이미 많은 명세 언어가 사용되고 있으며, 이들은 나름대로의 장점을 가지고 있으나, 이들은 대부분 구현에 있어 특정 언어에 의존성이 강하며, 표현 능력에 한계를 가지고 있다. 이러한 한계를 해결하기 위해 문제 요구사항에 대한 설명, 컴포넌트의 기능, 그리고 컴포넌트 구조에 대한 명확한 정의를 제공하는 정형화된 명세 언어에 대한 연구가 현재 이루어지고 있으나, 대부분이 컴포넌트 명세 기법에 대한 표준이 없고, 명세서에 대한 데이터 모델링 방법 및 표현언어에 대한 기준이 없이 컴포넌트의 구문적 측면만을 고려한 개발을 하고 있는 실정이다. 본 논문에서는 컴포넌트 명세서를 규격화하고 규격화된 명세서를 표준 메타 언어인 XML 기반으로 모델링 하고자 한다.

1. 서론

컴퓨터와 인터넷 사용의 대중화 시대가 열리면서 소프트웨어에 대한 사용자의 요구사항은 매우 다양해지고 이를 만족시키는 소프트웨어의 구조도 점차 복잡해짐은 물론 크기도 방대해지고 있으나 S/W 개발 기술 발전 속도는 하드웨어 발전 속도에 비해 매우 낙후되어 있는 실정이다. 실제로 하드웨어는 잘 정의되어 있는 규격에 따라 여러 칩(Chip)등의 부품들을 결합하여 만들 수 있지만 소프트웨어는 하드웨어처럼 재사용할 수 있는 여러 모듈들로 조합 개발하지 못하고 대부분 새로 개발해야한다. 그로 인하여 대부분의 소프트웨어 프로젝트들은 개발 시간의 지연, 예산 초과, 고객이 원하는 고품질의 소프트웨어를 생산하지 못함으로 인해 실패하는 경우가 많다. 이처럼 소프트웨어 개발과 유지보수 비용의 증가로 인해 소프트웨어 위기 문제가 발생하게 되었다. [1][2]

이에 대한 대안으로 소프트웨어 생산성, 품질, 효율성을 높이기 위해서 부품화와 조립의 특성을 지닌 컴포넌트 기반 개발 방법이 제시되었다. 즉, 컴포넌트를 부품처럼 조립하고 기능이 개선된 부품을 재조립하는 재사용 방식을 사용하게 된 것이다.

컴포넌트 명세는 그 특성을 명확히 이해하기 위하여 컴포넌트의 의미(semantic), 그들 사이의 관계(relationship), 사용 문맥(context), 품질 속성(quality attribute)들을 포함한 정보가 기술되어야 한다. 그러나 대부분의 컴포넌트 명세 기법은 표준이 없고, 명세서에 대한 데이터 모델링 방법 및 표현언어에 대한 기준이 없이 컴포넌트의 구문적 측면만을 고려한 개발을 하고 있는 실정이다. 또한 공용 컴포넌트 부족 및 컴포넌트 공유체제의 미흡 등으로 인하여 개발된 컴포넌트를 쉽게 찾아 활용할 수 있는 공유체제가 구축되어 있지 않은 관계로 불필요한 중복 개발을 하고 있다. 본 논문에서는 컴포넌트 명세서는 규격화하고 규격화된 명세서를 표준 메타 언어인 XML 기반으로 모델링 하고자 한다.

2. 관련연구

2.1 컴포넌트

컴포넌트는 바라보는 관점에 따라 그 정의가 다양하다. 하지만 컴포넌트에 대한 정의들의 공통점을 요약하여 보면 다음과 같은 내용으로 정리 할 수 있다. 첫째, 컴포넌트는 모듈화 되어 있는 단위이다. 둘째, 컴포넌트는 인터페이스를 가져야 한다. 즉, 컴

포넌트는 재사용을 높이기 위한 수단 가운데 하나로 인터페이스만을 제공함으로써 세부적인 구현 사항들을 외부에 숨기겠다는 것이다. 셋째, 컴포넌트는 구현되어 있는 단위이다. 즉, 컴포넌트는 단지 논리적인 모델이 아니라 실제 구동될 수 있도록 만들어진 모듈이라는 것이다. 넷째, 컴포넌트는 아키텍처를 기반으로 하다. 다섯째, 컴포넌트는 커스터마이즈가 가능해야 한다. 즉, 컴포넌트는 컴포넌트내의 세부 구현 사항들을 외부에 노출시키지는 않지만, 컴포넌트 사용자들이 자신의 목적에 맞도록 컴포넌트가 지니는 속성이나 메소드들을 변경할 수 있도록 지원되어야 한다는 것이다

2.2 XML

XML이란 'Extensible Markup Language'의 약자로 HTML과 같이 고정된 포맷을 사용하지 않고 확장가능하기 때문에 'Extensible(확장가능한)이라는 말을 사용한다. XML은 웹에서 SGML을 널리 사용하도록 하기 위해 설계되었다. XML이란 단순한 마크업 언어(Markup Language)가 아니고 자신만의 마크업 언어를 설계하게 해주는 메타언어이다. 일반적으로 마크업 언어는 HTML과 같이 문서들의 특정 클래스들 안에 정보를 표현하는 방법이지만, XML은 문서의 많은 클래스들을 위하여 사용자의 필요성에 따라 마크업 언어를 재 정의할 수 있도록 한다.

XML의 목표는 웹에서 HTML이 처리되는 방식으로 SGML이 서비스되도록 하는 것이다. XML은 SGML이 제공하는 중요한 장점인 정보제공자가 임의로 태그 집합과 속성을 정의하고 문서의 구조적 표현에 제약이 없는 것을 포함하고 있다. 또한 인간과 기계 모두에게 뛰어난 유연성과 단순성, 가독성을 제공하며 계단형 스타일시트(*Cascading Style Sheet*)와도 매끄럽게 작동하기 때문에 개발자들은 웹페이지들이 구조화된 것만큼 ptfus되게 표현될 수 있는 웹페이지를 생성할 수 있고 데이터를 교환하기 위한 훌륭한 포맷으로 사용할 수 있다.

2.3. 컴포넌트 명세기법

2.3.1 약정(Contract)기반 기술

소프트웨어 컴포넌트 세계의 4단계 약정으로서 기본 약정, 행위 약정, 동기화 약정, 양적 약정이 있다. 이러한 약정 이행에 대한 책임은 클라이언트와 서비스 제공자 모두에게 있다. 이러한 약정이 충분한 설명을 포함하고 있으면 관리하기 쉬우나, 일부 약정은 잠재하는 외부 요인에 의존하는 경우가 많다. 특히 서비스 질(Quality of Service) 같은 양적 약정은 시스템 계층에 의존하기 때문에 컴포넌트 외부 요인에 좌우된다. [6]

2.3.2 UML 기술

UML은 소프트웨어를 모델링하는 언어이다. UML로 컴포넌트를 기술하는 것은 크게 컴포넌트 구현 입장과 컴포넌트 사용 입장으로 나누어 볼 수 있다. 컴포넌트는 주어진 요구 기능들을 구현하기 위하여 하나 이상의 클래스들이 상호작용 하는 개발 단위를 구성하므로, UML에서의 패키지에 해당된다. 따라서 패키지의 스테레오 타입으로 컴포넌트를 정의할 수 있다. 컴포넌트를 구성하는 측에서는 다른 개발 단위 혹은 다른 컴포넌트 공급 업체로부터의 컴포넌트를 시스템 상에 표현하기 위해 패키지 다이어그램을 사용할 수 있다. 이 경우에 컴포넌트의 구현 세부사항을 표현하지 않고, 컴포넌트의 인터페이스 명세만을 사용한다.

2.3.3 정형 명세 기술

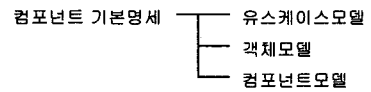
컴포넌트를 정형적으로 표현하는 가장 중요한 이유는 컴포넌트 명세를 최대한 정확하게 기술하여, 구현상의 모호함을 제거하는데 있다. 컴포넌트 명세를 정확하게 기술함으로써 시스템 통합시에 생길 수 있는 불일치를 없앨 수 있다. 더 나아가 정의된 명세들을 검증하고, 기술된 명세들의 일치성 및 완전성을 증명할 수 있다.

컴포넌트 표준에 정형 명세를 적용하는 것과 마찬가지로, 컴포넌트의 기능을 명료하게 제시하기 위해 정형명세를 사용할 수 있다. 컴포넌트 프레임워크나 모델링 언어에서의 모호성은 이들을 적용하는 많은 컴포넌트 및 설계에 영향을 주기 때문에 정형 명세화 하는 것이 바람직하다. 또한 안전성 또는 성능이 시스템 구현에 중요한 경우에도 정형 명세를 사용할 수 있다.[6]

3. 컴포넌트 명세서 구조

컴포넌트 명세서에는 컴포넌트를 인터넷상에 개방하면서 그 컴포넌트에 대한 검색을 하거나 컴포넌트를 이해하고자 하는 사용자들에게 도움이 되는 정보들이 상세하게 기술되어야 한다.

본 명세서는 ETRI에서 정의한 컴포넌트 명세 방법 V2.0과 1999년에 작성된 컴포넌트들에 대한 명세서와 정보과학회 논문지를 결합 분석한 것이다. 컴포넌트 명세는 <그림1>과 같이 3가지 영역으로 구분할 수 있다. [5][8]



<그림1> 컴포넌트 명세서 트리구조

3.1 유스케이스 모델(Use case Model)

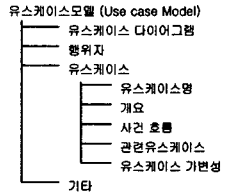
유스케이스 모델은 특정 도메인 내에서 컴포넌트를 개발하는 대상 응용 시스템들에 대한 기능들을 정의한다.

3.2 객체 모델(Object Model)

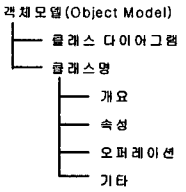
컴포넌트를 식별×추출하려는 대상 도메인의 응용 시스템들의 부분 집합에 대해 시스템의 객체(클래스)를 모형화 한다. 여기에서 식별된 클래스들을 기반으로 컴포넌트를 추출하다.

3.3 컴포넌트 모델(Component Model)

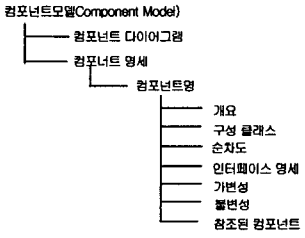
객체 모델을 바탕으로 컴포넌트를 식별하고 컴포넌트 간의 의존성(dependency)을 보여준다. 컴포넌트 간에 cycle 혹은 chain 의존성이 있을 경우, 개발 과정에서나 유지보수와 관련하여 여러 가지 문제가 있을 수 있으므로 가능한 한 단방향의 의존성만 가지도록 한다.



<그림2> 유스케이스모델



<그림3> 객체 모델



<그림4> 컴포넌트 모델

4. XML기반 컴포넌트 명세서 문서형 정의

<그림1>과 같이 컴포넌트 기본 명세를 세 부분으로 구분하고 이를 XML 문서형 정의(DTD)로 표현하면 아래와 같다.

```
<!ELEMENT Component_Description_Document
 (Usecase_Model+, Object_Model+,
  Component_Model+,)>
```

4.1 유스케이스 모델

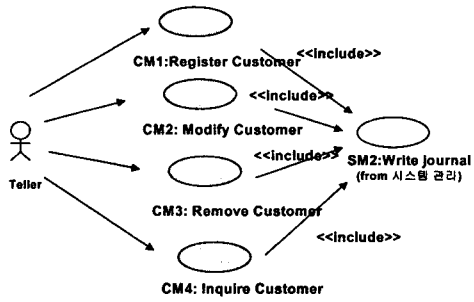
유스케이스 모델은 <그림2>와 같이 유스케이스 다이어그램 (UM_Usecase_Diagram), 행위자 (UM_Actor), 유스케이스 (UM_Usecase), 그리고 기타 (UM_etc)로 나누어 표현한다.

유스케이스 모델을 DTD로 표현하면 다음과 같다.

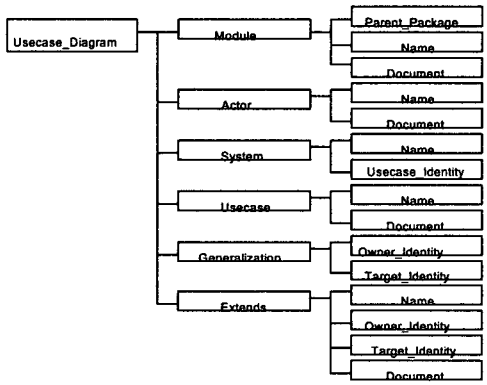
```
<!ELEMENT Usecase_Model
 (UM_Usecase_Diagram,
 UM_Actors+, UM_Usecase+, UM_etc?)>
```

4.1.1 유스케이스 다이어그램

유스케이스 다이어그램은 컴퓨터 시스템과 사용자의 상호 작용을 표현한 다이어그램이다. 유스케이스 다이어그램의 예인 <그림5>를 보고 <그림6 유스케이스 다이어그램 트리 구조>를 만들었다.



<그림5> 유스케이스 다이어그램 예



<그림6> 유스케이스다이어그램 트리 구조

유스케이스 다이어그램 트리 구조를 UML 기반의 다이어그램 지원 요소인 모듈(Module), 행위자(Actor), 시스템(System), 유스케이스(Usecase), 일반화(Generalization) 관계, 확장(Extends)관계로 분류할 수 있고, 이를 DTD로 표현하면 다음과 같다.

```
<!ELEMENT UD_Usecase_Diagram (Usecase_Module, Usecase_Actor+, Usecase_System*,
        Usecase_Usecase+, Usecase_Generalization*, Usecase_Extends*>
    <!ELEMENT Usecase_Usecase_Module (Usecase_Module_Name,
            Usecase_Module_Document)>
        <!ELEMENT Usecase_Module_Name (#PCDATA)>
        <!ELEMENT Usecase_Module_Document (#PCDATA)>
    <!ELEMENT Usecase_Actor (Usecase_Actor_Name, Usecase_Actor_Document)>
        <!ELEMENT Usecase_Actor_Name (#PCDATA)>
        <!ELEMENT Usecase_Actor_Document (#PCDATA)>
    <!ELEMENT Usecase_System (Usecase_System_Name,
                Usecase_System_Usecase_Identity)>
        <!ELEMENT Usecase_System_Name (#PCDATA)>
        <!ELEMENT Usecase_System_Usecase_Identity (#PCDATA)>
    <!ELEMENT Usecase_Usecase (Usecase_Usecase_Name,
                Usecase_Usecase_Document)>
        <!ELEMENT Usecase_Usecase_Name (#PCDATA)>
        <!ELEMENT Usecase_Usecase_Usecase_Document (#PCDATA)>
    <!ELEMENT Usecase_Generalization (Usecase_Generalization_Owner_Identity,
            Usecase_Generalization_Target_Identity)>
        <!ELEMENT Usecase_Generalization_Owner_Identity (#PCDATA)>
        <!ELEMENT Usecase_Generalization_Target_Identity (#PCDATA)>
    <!ELEMENT Usecase_Extends (Usecase_Extends_Name,
                Usecase_Extends_Owner_Identity,
            Usecase_Extends_Target_Identity, Usecase_Extends_Document)>
        <!ELEMENT Usecase_Extends_Name (#PCDATA)>
        <!ELEMENT Usecase_Extends_Owner_Identity (#PCDATA)>
        <!ELEMENT Usecase_Extends_Target_Identity (#PCDATA)>
        <!ELEMENT Usecase_Extends_Document (#PCDATA)>
```

<표 1> 유스케이스다이어그램 DTD 표현

4.2 객체 모델

객체 모델은 <그림3>과 같이 클래스 다이어그램(Class Diagram)과 클래스 명(Class Name)으로 구분하며 이를 DTD로 표현하면 다음과 같다.

```
<!ELEMENT Object_Model (Class_Diagram,
  Class_s+)>
```

4.2.1 클래스 다이어그램(OM_Class_Diagram)

클래스 다이어그램은 식별된 클래스들의 관계(relation), 기수성(multiplicity), 속성(attributes), 오퍼레이션(operation)을 기술한다. 클래스 다이어그램의 효과적인 표현을 위해 필요한 경우, 관련된 클래스 별로 패키지를 이용한 그루핑이 가능하다.

클래스 다이어그램의 세부적인 내용으로는 모듈(Module), 클래스(Class), 오퍼레이션(Operation), 속성(Attribute), 일반화(Generalization), 의존 관계(Dependency), 연관 관계(Association), 집합연관 관계(Aggregation), 복합연관 관계(Composition), 연관관계(Role) 등으로 분류·표현할 수 있다.

4.2.2 클래스 명(OM_Class_Name)

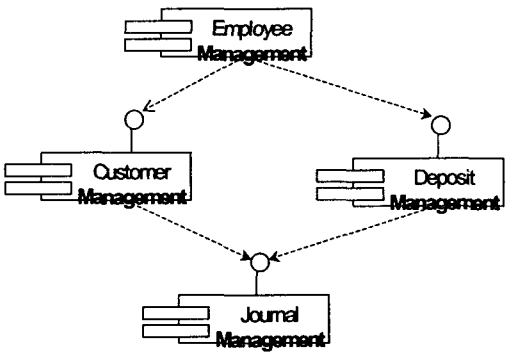
클래스 명은 클래스 다이어그램에 있는 각 클래스에 대해 개별적으로 기술하는 부분으로 크게 네 부분으로 분류되는데 클래스의 개요(Agreememt), 속성(Attribute), 오퍼레이션(Operation), 기타(Etc)등으로 분류·표현할 수 있다.

4.3 컴포넌트 모델 (Component Model)

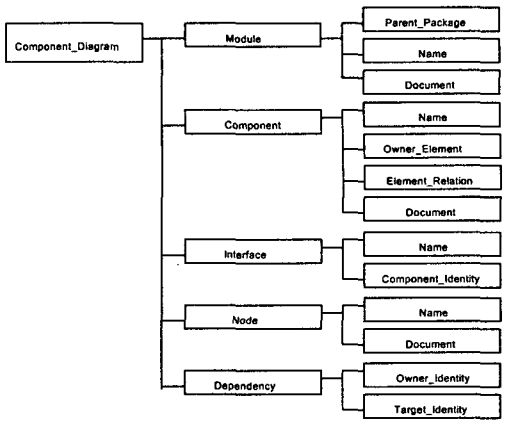
컴포넌트 모델은 소프트웨어의 물리적 단위(exe, dll 등 기타 library)의 구성과 연결상태를 나타내는 곳으로서, <그림4>와 같은 구조로 분류할 수 있고, 이를 DTD로 표현하면 아래와 같다.

```
<!ELEMENT Component (Component_Diagram,
          Component_Specifications+)>
```

4.3.1 컴포넌트 다이어그램(Component_Diagram)



<그림7> 컴포넌트 다이어그램 예



<그림8> 컴포넌트 다이어그램 트리 구조

5. 결론

본 논문에서는 첫째, 컴포넌트의 정의와 컴포넌트 명세를 위한 컴포넌트 기술동향을 검토하였다. 둘째, 컴포넌트 명세를 구성하는 정보의 모델링 측면에서 컴포넌트 요소들을 도출하였으며 컴포넌트 명세서를 위한 세부 정보를 식별하였다. 셋째, 컴포넌트의 규격화를 위해서 XML기반 컴포넌트 문서형 정의(DTD)를 작성하였다.

다음엔 작성된 문서형 정의를 기반으로 표준 메타언어인 XML 기반으로 모델링 하고자 한다

참고문헌

[1] Wojtek Kazaczynski and G.Booch, "Component Based Software Engineering", IEEE Software, pp34-36, Sept./Oct. 1998.

[2] A.W.Brown and K.C.Wallnau, "The Current State of CBSE", IEEE Software, pp37-46, Sept./Oct. 1998.

[3] Desmon F. D'souza and A. C. Wills, Objects, Components, and Components with UML,Addison-Wesley,1998.

[4] Sun Microsystems, Enterprise JavaBens Specification 1.1 at URL:http://www.javasoft.com

[5] 박서영, 신영길, 우치수, XML컴포넌트 명세서 기반의 컴포넌트 검색 기법, 정보과학회논문지 : 소프트웨어 및 응용 제 27권 제 2호(2000.2)

[6] 박찬진, 이명정, 우치수, 컴포넌트 명세 기법, 소프트웨어공학회지 12권 3호(1999.9)

[7] 이상덕,정효택,신석규, 공용컴포넌트 개발 및 기술개발 전략, 정보처리 제7권 제4호(2000.7)

[8] 한국전자통신연구원, 소프트웨어공학연구부, "컴포넌트 명세 작성 지침서(안)", 컴포넌트 명세 방법 V2.0

[9] 최한석, 김동욱, "XML 구조 문서 편집기 설계 및 구현" 한국정보과학회 제26권 1호