

ADL 모델 관리기 설계를 위한 XML기반 ADL 메타모델 정의

강미나^o 승현우
서울여자대학교 컴퓨터학과
{gmnmn^o, hwseung}@swu.ac.kr

Defining XML Based ADL Meta Model for the Design of ADL Model Manager

Mina Kang^o Hyonwoo Seung
Dept. of Computer Science, Seoul Women's University

요 약

소프트웨어의 재사용을 위한 방법 중에 최근 빠르게 확산되고 있는 방식으로 컴포넌트 기반의 소프트웨어 개발(CBD : Component-Based Development)방식이 있다. CBD에 있어 컴포넌트 시스템은 잘 정의된 아키텍처를 기반으로 개발되어야 한다. 컴포넌트 시스템의 아키텍처를 정확하고 엄밀하게 설계, 분석하기 위해서는 아키텍처 기술 언어(ADL)의 사용이 필요하다. 아키텍처 모델링에 ADL을 효과적으로 사용하기 위해서는 ADL로 기술된 아키텍처 모델을 효율적으로 분석, 처리, 관리할 수 있는 지원 환경이 필수적이다. 본 논문에서는 ADL 모델 지원 환경 시스템의 구조를 기술하고 본 연구팀에서 개발 중인 ADL 모델 관리기의 구조를 기술한다. 또한 ADL 모델 관리기 개발에 기본이 되는 XML 기반 ADL 메타모델을 정의한다.

1. 서 론

컴포넌트 기반 소프트웨어 개발(CBD : Component-Based Development)은 독립된 단위로 기능을 하는 소프트웨어 조각(Component)을 조립하여 소프트웨어를 개발하는 방식으로, 최근 소프트웨어 재사용을 위한 방법으로 빠르게 확산되고 있다.

CBD에 있어 컴포넌트 시스템은 잘 정의된 아키텍처를 기반으로 개발되어야 한다. 컴포넌트 시스템의 아키텍처를 정확하고 엄밀하게 설계, 분석하기 위해서는 아키텍처 기술 언어(ADL)의 사용이 필요하다. 아키텍처 모델링에 ADL을 효과적으로 사용하기 위해서는 ADL로 기술된 아키텍처 모델을 효율적으로 분석, 처리, 관리할 수 있는 지원 환경이 필수적이다.

본 연구는 ADL 모델 지원 환경 시스템 개발을 목표로 주관연구와 공동연구로 나누어 수행된다. 주관연구에서는 ADL을 정의하고 아키텍처 모델의 구조적 행위적 성질들을 분석하여 정의된 스타일 규칙들의 위반 여부를 검사하는 방법 및 이를 지원하는 ADL 모델 분석기를 개발한다. 공동연구에서는 아키텍처 모델 분석으로 생성된 아키텍처 정보들을 XML 형태로 변환, 저장, 지원하는 ADL 모델 관리기를 개발한다.

본 논문에서는 ADL 모델 지원 환경 시스템의 구조를 기술하고 공동연구 기관인 본 연구팀에서 개발 중인 ADL 모델 관리기의 구조를 기술한다. 또한 ADL 모델 관리기 개발에 기본이 되는 XML 기반 ADL 메타모델을 정의한다. ADL 메타모델은 XML Schema로 정의되며 이는 ADL 모델 관리기 내의 XML기반 ADL 모델 변환기, 저장소 개발에 기본이 된다.

2. C2스타일 기반의 아키텍처 기술 언어

주관연구에서는 C2스타일 기반의 ADL을 1)컴포넌트 인터페이스 명세 언어(IDN : Interface Definition Notation)와 2) IDN으로 정의된 컴포넌트들과 C2 커넥터들 사이의 바인딩들로 구성된 아키텍처 형세(topology)를 기술하는 아키텍처 명세 언어(ADN : Architecture Definition Notation)로 나누어 설계하고 각 언어의 구문(syntax) 및 의미(semantics)를 정의하였다 [1][2][3].

본 논문은 한국과학재단 목적기초과학연구 지원으로 수행되었음

2.1 컴포넌트 명세 언어 - IDN(Interface Definition Notation)

(그림 1)는 컴포넌트 명세 언어(IDN)의 구문이다. IDN은 컴포넌트의 포트 메시지와 메소드의 선언 부분, 그리고 선언된 메시지와 메소드를 통해 컴포넌트의 행위를 나타내는 부분으로 구성되어 있다.

Port 부분은 top 또는 bottom 포트를 통해 수신되거나 발송되는 컴포넌트 인터페이스의 메시지들을 선언한다. Methods 부분은 수신된 메시지에 대해 컴포넌트가 호출할 수 있는 메소드들을 선언한다. Behavior 부분은 컴포넌트의 startup(동작 시작), cleanup(종료), 또는 runtime(실행) 시의 행위를 나타낸다.

```
<<component> ::=
  <package>
  <import>
  component <comp_name> (
    <port top (
      out ( <msg_list> )
      in ( <msg_list> )
    )
    <port bottom (
      out ( <msg_list> )
      in ( <msg_list> )
    )
    <methods ( <method_decl_list> )
    <behavior (
      <startup>
      <cleanup>
      <received>
    )
    <notes ( <note_list> )
  )

<pack> ::= < t | package <id_list> ;
<import> ::= ( import <id_list> ; ) *
<id_list> ::= <id> ( , <id> ) *
<comp_name> ::= <id>
<msg_list> ::= ( <msg_signature_1> ; ) *
<msg_signature_1> ::= <id_list> ( <param_list_1> )
MSignature1
<param_list_1> ::= < t | <type><id> ( , <type><id> ) *
<type> ::= <id_list>
<method_decl_list> ::= ( <type> <msg_signature_1> ; ) *
MethodDeclList
<startup> ::= | startup( <invoked_methods> <generated_msgs> )
<cleanup> ::= | cleanup( <invoked_methods> <generated_msgs> )
.....
```

그림 1. IDN 구문

2.2 아키텍처 명세 언어 - ADN(Architecture Definition Notation)

(그림 2)는 아키텍처 명세 언어(ADN)의 구문이다. ADN은 아키텍처의 컴포넌트와 커넥터의 인스턴스 선언 부분, 컴포넌트들과 커넥터의 인스턴스들 간의 연결 관계들을 정의하는 부분으로 구성되어 있다.

Topology 부분은 컴포넌트와 커넥터의 연결 관계인 binding 부분들로 구성되며 아키텍처의 형태(configuration)를 나타낸다. Components 부분에서 나타나는 컴포넌트 인스턴스의 타입(<comp_name>)은 IDN 컴포넌트 명세에서 정의된다.

```

<architecture> ::=
  <package>
  <import>
  <architecture> <arch_name> (
    <components> { <component_list> }
    <connectors> { <connector_list> }
    <topology> { <binding_list> }
    <notes> { <note_list> }
  )
  )

<package> ::= <id> package <package_name> ;
<import> ::= ( import <comp_name> : ) *
<package_name> ::= <id> ( . <id> ) *
<arch_name> ::= <id>
<component_list> ::=
  ( <position> <comp_name> <brick_inst_list> : ) *
<position> ::=
  <id>
  | top_most
  | bottom_most
  | internal
  .....
  .....
    
```

그림 2. ADN 구문

3. ADL 모델 지원 환경 시스템

아키텍처 모델링에 ADL을 효과적으로 사용하기 위해서는 ADL로 기술된 아키텍처 모델을 효율적으로 분석, 처리, 관리할 수 있는 지원 환경이 필수적이다. ADL 모델 지원 환경 시스템 구조는 (그림 3)과 같이 크게 ADL 모델 분석과 ADL 모델 관리의 두 부분으로 구성된다.

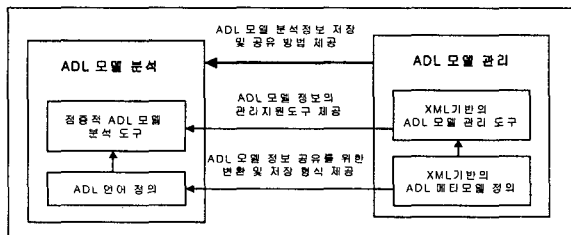


그림 3. ADL 모델 지원 환경 시스템 구조

ADL 모델 분석에서는 ADL 언어를 정의하고 ADL 모델 분석 도구를 개발한다. ADL 분석의 범위는 ADL로 기술된 아키텍처 모델에 대한 구문 분석, 의미 분석, 일관성 분석으로 구문 오류(syntax error)를 검사하고 파스 트리를 생성하는 ADL 파서 개발을 포함한다.

ADL 모델 관리에서는 ADL 모델을 XML 형태로 변환 저장 관리하는 것을 목적으로 XML 기반 ADL 메타모델을 정의하고 ADL 모델 관리 도구를 개발한다. ADL 메타 모델은 XML Schema로 정의하며 ADL 모델 관리 도구는 XML 기반 ADL 모델 변환기, 변환 모델 분석 정보 저장을 위한 저장소, 저장된 분석 정보 검색을 위한 검색기 개발을 포함한다.

3.1 ADL 모델 관리기

ADL 모델 관리기는 아키텍처 모델 분석 과정 중에 생성된 아키텍처 모델 정보들을 서로 다른 ADL 지원 도구들이 공유할 수 있도록 XML 형태로 변환, 통합 관리하기 위한 도구이다. ADL 모델 관리기는 (그림4)와 같이 XML 기반 ADL 모델 변환기, 저장소, 모델 검색기로 구성되어 있다.

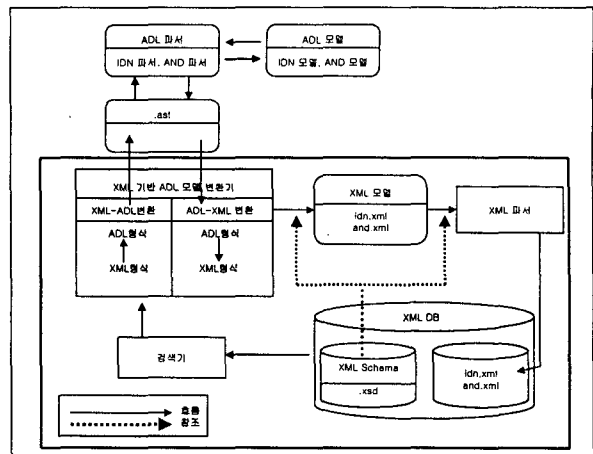


그림 4. ADL 모델 관리기 구조

XML 기반 ADL 모델 변환기는 ADL-XML 변환, XML-ADL 변환을 수행한다. ADL 파서에 의해 파싱된 ADL 모델 정보들은 ADL-XML 변환을 통해 XML 형태의 모델로 변환되며 변환된 XML 형태의 모델 정보들은 DB에 저장된다. 저장된 모델 정보들은 특정 요구에 의해 검색기를 통해 검색되어 XML-ADL 변환을 통해 다시 ADL 모델 정보들로 복원된다.

4. XML 기반 ADL 메타모델 정의

본 장에서는 ADL 모델 관리기 설계에 있어 기본이 되는 XML 기반 ADL 메타모델 정의에 대해 기술한다.

ADL 메타모델은 ADL 언어로 기술된 모델들의 구문 정보를 담고 있는 모델로 본 연구팀에서는 XML Schema로 이를 정의한다. Schema로 정의된 ADL 메타모델은 ADL 모델 관리기에서 XML 기반 ADL 모델 변환기 및 저장소 개발과 변환된 XML 모델 파싱에 참조된다.

4.1 XML Schema

XML Schema는 XML 형식에 따라 문서 내의 데이터 타입들의 의미와 사용, 관계 등을 제안하고 기술한다. 이것은 XML 문서의 구조를 표현하기 위해 기존에 사용했던 DTD(Document Type Definition)의 문제점을 해결하기 위해 W3C에서 개발한 것으로 다음과 같은 특징이 있다.

첫째, XML Schema는 XML과 같은 문법 형식을 취한다. 기존의 DTD는 XML 문서 구조를 정의하는데 있어 XML과 다른 문법인 EBNF 형식을 취하므로 XML 문법과 EBNF 문법을 따로 익혀야 하며 응용 어플리케이션 측면에서도 두개의 해석기를 지녀야만 하는 어려움이 있었다. 이에 반해 XML Schema의 경우 문서 정의 부분과 실제 사용 부분들이 XML로 표현되므로 XML 문서 교환 시 큰 장점이 된다.

둘째, XML Schema는 다양한 데이터 타입을 지원하고 사용자 정의를 통해 무한히 확장 가능하다. 기존의 DTD의 경우 제한적인 데이터 타입만을 지원하는 반면 Schema의 경우 미리 만들어진 원시 타입과 그에서 유도된 타입이 존재하며 이들을 사용자 정의로 확장이 가능해 데이터 타입이 무한대라고 할 수 있다.

셋째, XML Schema는 네임스페이스를 지원한다. XML Schema의 경우 기존 DTD에서는 지원하지 못한 네임스페이스를 활용해 다양한 스키마들을 참조해 하나의 스키마를 구성함으로써 재사용을 극대화 할 수 있으며 여러 문서상의 용어의 충돌을 막을 수 있다.

4.2 ADN 구문의 XML Schema로의 변환

본 절에서는 ADN 구문 중 component에 해당하는 구문을 XML Schema로 정의하는 과정을 기술하며, ADN 구문 전체에 대한 각각의 ADN, IDN Schema를 기술한다.

component 구문은 먼저 component의 전체적인 구조를 정의하고 세부적으로 component의 구성 요소들을 정의하는 과정에 의해 XML Schema로 정의된다. 이때 사용되는 Schema 문법들은 다음과 같다[4][5].

루트 엘리먼트 및 자식 엘리먼트 타입의 정의는 전체적 component 구조를 정의하고, (그림 5)에서와 같이 글로벌 심플 타입의 정의는 component 구성요소 중 position을 정의한다. 글로벌 콤플렉스 타입의 정의는 (그림 6)에서와 같이 component 구성요소 중 brick_inst-list를 정의한다.

```
<simpleType name="심플 타입 이름">
  (restriction | list | union)
</simpleType>
```

그림 5. 글로벌 심플 타입 정의 문법

```
<complexType name="콤플렉스 타입 이름">
  <choice minOccurs="최소선택횟수" maxOccurs="최대선택횟수">
    엘리먼트 선언들
  </choice>
</complexType>
```

그림 6. 글로벌 콤플렉스 타입 정의 문법

위에 정의된 각각의 Schema 문법을 적용하여 component 구문을 Schema로 정의하면 다음과 같다.

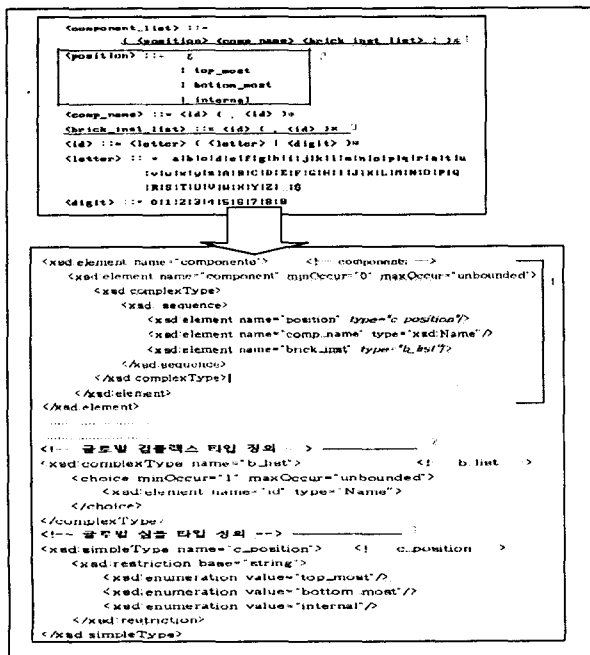


그림 7. component 구문의 Schema 변환 과정

4.3 ADN, IDN Schema 정의

본 절에서는 Schema의 타입 정의 문법에 의해 정의된 ADN Schema와 IDN Schema를 기술한다.

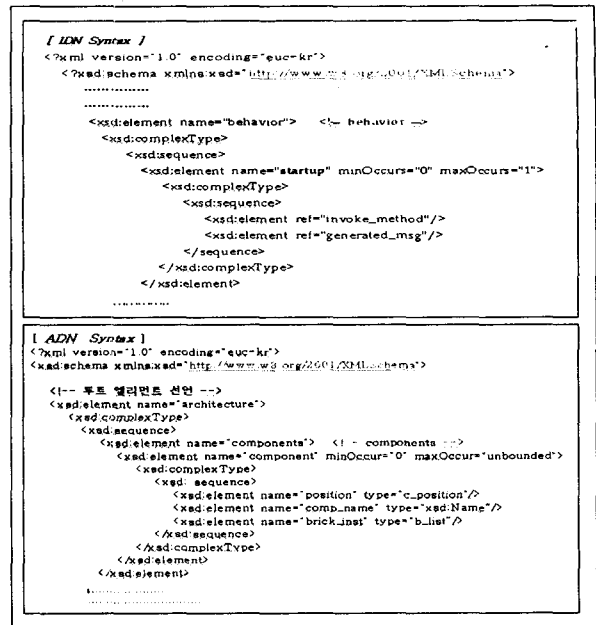


그림 8. ADN, IDN Schema

5. 결론 및 향후 연구

본 논문에서는 아키텍처 기반의 소프트웨어 개발(CBD)에 효과적으로 사용될 수 있는 ADL 모델 지원 환경 시스템의 구조를 제안 하였으며 현재 개발 중인 ADL 모델 지원 시스템 내의 ADL 모델 관리기의 구조를 기술하였다. 또한 ADL 모델 관리기 개발에 기본이 되는 XML 기반 ADL 메타모델을 XML Schema로 정의하였다. ADL 메타모델은 ADL로 기술된 아키텍처 모델의 구문 정보를 담고 있는 것으로 현재 개발 중인 모델 관리기 내의 변환기 및 저장소 개발에 기본이 된다.

향후 본 논문에서 제시한 XML Schema로 정의한 ADL 메타모델을 기반으로 ADL 모델 변환기, 저장소, 검색기의 기능을 가지는 XML기반 ADL 모델 관리기를 개발할 계획이다.

6. 참고문헌

- [1] 노성환, 신동익, 최재각, 전태웅, "C2스타일의 아키텍처 기술을 지원하는 ADL 정의" 한국정보과학회 가을 학술발표논문집 Vol. 28, No. 2, 2001. 10
- [2] 노성환, 신동익, 최재각, 전태웅, 이승연, 권오천, 신규상, "C2스타일의 아키텍처 기술을 지원하는 ADL 지원도구의 개발" 정보처리학회 논문지D 제8-D권 제6호, 2001. 12
- [3] The C2 Style, <http://www.ics.uci.edu/pub/arch/c2.html>, Information and Computer Science, University of California, Irvine
- [4] jon Duckett의 8인 공저, "Professional XML Schemas" 2002. 6
- [5] W3C, Extensible Markup Language(XML)1.0 <http://www.w3.org/TR/2000/REC-xml-20001006.html>, Oct, 2000