XML 스키마 메타모델에서 OCL 생성

Generate of OCL on XML Sechma Meta Model

이돈양*, 최한용**
세종대학교 컴퓨터공학부*, 한북대학교 컴퓨터공학과**

Don-Yang Lee(dylee11@sejong.ac.kr)*, Han-Yong Choi(hychoi@hanbuk.ac.kr)**

요약

XML이 인터넷에서 정보전송을 위한 메타언어의 표현방법으로 급속하게 이용되고 있다. 아울러 XML 스키마는 XML 데이터의 다양한 유형을 표현하는데 사용하는 빈도수가 늘고 있다. 본 논문에서는 UML을 이용한 XML 스키마의 simpleType형 메타모델에 대한 설계를 하였다. 그러나 XML 스키마의 구조가 복잡하고 다양한 데이터의 유형을 지원하기 때문에 UML에서 나타내고 있는 모델의 속성에 대한 사용자의 이해와 적용에 어려운 부분이 많이 발생하는 것을 알 수 있다. 이를 해결하는 방법으로 본 연구에서는 OCL의 기능을 적용하여 XML 스키마 메타모델에서 구조적인 표현을 명확하게 명시할 수 있도록 하였으며, 아울러 이를 바탕으로 컴파일단계에서 어휘분석과 구문분석을 위한 파스트리와 토큰생성에 대한 구체적인 설계방법을 제시하였다.

■ 중심어 : | XML | OCL | DTD | XML 스키마 | 파스 트리 |

Abstract

XML used rapid method of meta language representation in internet for information transmission. In addition to XML Schema used frequency specification to variety data type. This thesis designed to Simple Type meta model of XML schema using UML. But because structure of XML schema complicate and suppose variety data type we can recognize many difficult matter to user's apprehension and application of model properties that appeared UML. To way out of this matter this study could specified clearly to structured expression in XML schema meta model that is applied OCL specification and together, come up with method of detailed design to parse tree and token generation for lexical and symmentics analysis in compile step on this study foundation.

■ keyword : | XML | OCL | DTD | XML Schema | Parse Tree |

I. 서 론

UML은 소프트웨어개발에서 사실상의 표준으로 사용되고 있으며 XML 도큐먼트로 변환되는 웹 어플리케이션을 포함하고 있다. 그러므로 UML 기반 소프트웨어 개발에서 XML 스키마의 발전이 필요하게 되었다[1].

접수번호 : #060425-001
접수일자 : 2006년 04월 25일
심사완료일 : 2006년 06월 16일
교신저자 : 이돈양, e-mail : dylee11@sejong.ac.kr

XML이 웹에서 데이터교환의 표준으로 일반적으로 사용되며, DTD는 XML 인스턴스 문서의 구조를 표현하는데 가장 많이 사용하는 메타언어이다. 그러나 DTD 세부적인 데이터의 표현에 있어서 충분하지 못하다. 반면 XML 스키마는 XML 도큐먼트에서 데이터의 형식을 자세하게 정의할 수 있는 최고의 마크업언어이다. 따라서 본 연구에서는 DTD보다 더 표현력이 우수하고, 정확한 자료구조를 제공하는 구조정의 언어 인 XML 스키마언어를 선택하였다. XML 스키마에서 사용되는 데이터의 타입은 사용용도에 따른 분류와 정의되는 위치에 따른 분류로 나눠진다. 사용용도에 따른 분류에서 빌트인 심플타입과 사용자 정의 심플타입, 사용자 정의 콤플렉스 타입 등 이 세 가지의 종류가 있다.

그러나 기존연구에서는 XML 스키마에 대한 모델링에서 제한적인 문제들이 발생되었으며, 또한 XML 스키마의 복잡한 내용을 개념적인 수준의 표준화되지 않는 표기의 추가적인 소개 없이 UML로 표기하기에 어려움이 있고, XML 스키마에서는 다중상속을 지원하지 못하고 있다[2-4]. 따라서 본 논문에서는 XML 스키마언어에서 가장 일반적으로 사용되는 사용자 정의 심플타입의 형식을 가지고 OCL을 적용하고자 한다.

첫 번째로 UML을 가지고 XML 스키마의 simpleType 형의 메타모델에 대한 설계를 하였다. 두 번째, XML 스키마의 구조가 복잡하고 다양한 데이터의 유형을 지원하기 때문에 UML에서 나타내고 있는 모델의 속성에 대한 사용자의 이해가 어려운 부분이 많이 발생하므로 OCL의 기능을 이용하여 XML 스키마 메타모델에서 구조적인 표현을 명확하게 할 수 있도록 하였다. 마지막으로 이를 바탕으로 컴파일단계에서 어휘분석과 구문분석을 위한 파스트리와 토큰생성에 대한 구체적인 설계방법을 제시하였다. [그림 1]은 본 연구에서 메타모델의 각각의 기능에 대한 세부적인 단계를 보여주고 있다.

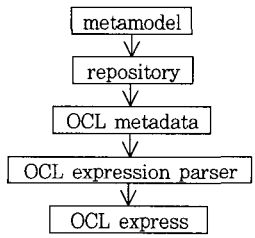


그림 1. 메타모델과 OCL 생성단계

따라서 본 논문에서는 XML 스키마를 이용한 마크업언어 생성에서 메타모델의 설계와 OCL 표기방법 그리고 이의 컴파일과정에서 어휘분석과 구문분석에 대한 세부적인 생성방법을 제시하였다.

그 결과 XML 스키마 마크업언어에 대한 메타모델설계뿐만 아니라 사용자가 이해하기 어려운 클래스간의 관계, 엘리먼트의 타입적용 등을 명확히 명세할 수 있었다.

II. 관련연구

1. UML과 XML 스키마

XML은 인터넷을 통한 정보의 전송에 급속히 표준적인 방법으로 이용되고 있으며, 특히 XML 스키마는 XML 데이터구조를 표현하는 방안으로 빠르게 선호되고 있다. 최근에는 텍스트기반의 구현에서의 기반이 부족한 상태에서 XML 스키마에 대한 비쥬얼화된 설계의 적용이 이루어지기도 하나 아직은 미흡한 상태이다. 따라서 참고문헌[5]에서는 XML 스키마의 생성에 대한 개념적인 언어의 표현인 "Object Role Modelling"을 제안하기도 하였다. 논문[6][7]은 UML과 XML 스키마의 관계를 전통적인 데이터베이스의 세 가지의 레벨을 사용하여 접근하는 방법을 찾고 있다. 이 세 개의 레벨은 개념(conceptional), 논리(logical), 물리(physical) 설계 레벨이다. 개념레벨에서는 최소한의 추가로 개념적인 제약을 이용한 표기의 표준 UML 클래스의 표기법을 사용하여 표현하였고 논리레벨에서는 UML의 스테레오 타입을 표기하고 있으며, 물리레벨에서는 XML 스키마를 사용하여 표현하였다. 이 세 가지 레벨을 설계한 목적은 UML 클래스의 개념레벨에서 논리레벨로 자동적으로 매핑이 되도록 하기 위함이다. 그리고 데이터 중복(redundancy)의 최소화와 정확도의 최대화에 바탕을 두고 있다.

XML 스키마와 UML과 관련된 수많은 접근 방법들은 다른 연구자들에 의해서 기술되고 있다[8][15]. 또한 현재의 솔루션들에는 수많은 문제들이 산재해 있다. 하나의 접근으로 sox에 의한 UML profile을 기술하는 것은 다른 시대에 뒤떨어지는 느낌을 갖는다[8]. 다른 논문[9]

들은 DTD에 의한 UML profile을 기술하고 있으며 이는 추가적인 구조와 제약되어 많은 실수들을 가질 수 있으나 XML 스키마에 의해 해결방법을 찾을 수 있다.

2. OCL

OCL을 이용한 성능이 향상된 UML 다이어그램 표기법에서는 OCL의 제약 명세에 의해 모델의 애매모호한 점을 최소화할 수 있다[10][11]. 그리고 사전/사후형식을 사용하여 조건표시를 할 수 있어 subject가 다중으로 해석되지 않도록 객체들의 제약을 가지고 표현할 수 있는 표준 적인 방법을 제공한다.

UML 프로파일과 OCL 변환규칙을 살펴보면 UML 표기법에서 제약조건을 기술할 때는 괄호({})를 이용하며, 제약조건의 내용은 OCL을 사용하여 기술한다[12]. 각 오퍼레이션은 사전/사후 조건을 갖는다. 이것은 알고리즘이나 구현에 대한 설명이 아니라 오퍼레이션의 파급효과만을 명세화하며 오퍼레이션 사용자와의 간단한 계약서 구실을 한다. 또한 오퍼레이션이 해야 할 일을 상세하게 명세하며 항상 쌍으로 나타난다. 사후조건은 사전조건이 참일 경우에 오퍼레이션의 결과가 무엇이 될 것인지를 설명한다.

UML에서 사전조건의 의미는 특히, 규칙기반의 시스템(rule-based system)과 정형화된 액티비티 의존성 다이어그램(activity dependency 다이어그램)에서 잘 못 해석될 수 있기 때문에 여기서 강조할 필요가 있다. 사전조건은 오퍼레이션이 호출되기 위한 조건은 아니다. 오퍼레이션의 기동은 전적으로 이 사전조건의 값과는 무관하다. 사전조건은 사후조건이 참이 되기 위해 오퍼레이션이 보증해야 하는 조건이다. 오퍼레이션이 기동될 때 사전조건이 거짓이라면, 그 결과는 명시되지 않는다. 오퍼레이션결과에 대한 전제가 없다. 사전/사후조건을 기술하는 또 다른 방법은 오퍼레이션에 대한 가정(assumption)과 보증(guarantee)을 사용하는 것이다[10]. 사전조건은 올바른 가능을 수행하는 오퍼레이션을 기대하고 있는 가정을 나타내며, 사후조건은 그 가정적인 조건이 만족하였을 때 오퍼레이션이 보증해야하는 조건을 의미한다. UML에서는 이러한 계약적 조건을 OCL을 이용하여 정확하게 명세할 수 있다. OCL은 논리적인 표현을 기술하기 위한 서술적인 언어이다. OCL로 문장을 작성하는 것은 어려울 수 있다. 그러나 일단 사용하면 자연어에서 느꼈던 문장의 애매모호함을 배제할 수 있으며, 명확한 해석을 가능하게 한다.

OCL 표현식은 정확한 속성 값을 갖는 구조적인 존재를 강조함으로서 동일한 내용보다 명확하게 표현해준다.

III. XML 기반 클래스 설계 및 OCL 명세방법

본 연구에서는 UML 모델링을 XMI 메타모델로 생성하고, 다시 메타데이터로 분리하여 단일 클래스 형식으로 DB에 저장하는 방법에 대해서 제시하였다. 이렇게 DB에 저장된 클래스의 메타데이터는 관계형 데이터베이스형의 메타데이터나 일반형의 XML 메타데이터로 변환이 가능하다. DB로 저장된 클래스 테이블에는 모델내의 클래스 상속관계를 표시하는 model name, SuperClass, SubClass 등 필드가 있다. 여기서는 모델에서 클래스에 대한 관계뿐만 아니라 일반적인 단일 클래스의 메타데이터 생성에도 초점을 맞추어 XML 메타데이터 생성을 하였고, XML 메타데이터에 대한 마크업언어도 DTD 기반이 아닌 XML 스키마 기반으로 하고 있다.

DTD는 XML 문서구조를 표현하는 마크업언어를 작성하는 일반적인 방법으로 사용되어 왔으나, 세부적인 데이터구조를 표현할 수 있는 기능이 없다. W3C의 XML 스키마는 이와는 다르게 데이터의 구조에 대한 형식 및 제약사항등을 훨씬 효과적으로 표현할 수가 있어 XML 문서구조를 다양하게 정의하거나 검증하는데 사용이 일반화 되어가고 있다.

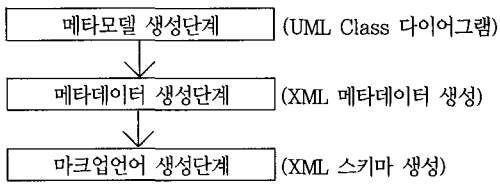


그림 2. 3단계 설계

[그림 2]는 UML의 클래스 다이어그램을 이용하여

XML 스키마를 모델링하고, 사용자들로 하여금 정확하고 효과적인 마크업언어를 작성할 수 있도록 하기 위해서 세 개의 단계로 나누어서 적용하고 있다. 첫 번째 단계는 UML 클래스 다이어그램을 이용한 메타모델 생성 단계이며, 두 번째 단계는 UML 클래스에 대한 메타데이터를 생성하는 단계이고, 마지막 세 번째 단계는 마크업 언어를 생성하는 단계이다.

1. XML 스키마언어 데이터타입 분류

앞에서 정의한 클래스의 XML 메타데이터의 마크업언어 정의에 사용되는 컴포넌트의 형식 중 본 논문에서는 심플타입(simple type)과 콤플렉스 타입(complex type)으로 분류하였다. XML 스키마에서 데이터타입은 이미 정의가 되어 사용되고 있는 내장형 심플타입(built-in simple type)과 사용자가 정의하여 사용하는 사용자 심플타입(user define simple type) 그리고 사용자 정의 콤플렉스 타입(user define complex type)으로 나눌 수 있다. 또한 데이터가 정의되는 위치에 따라 글로벌 데이터타입(global data type)과 로컬 데이터 타입(local data type)으로 분류할 수 있다.

```
〈simpleType  name = "simple type name"〉
   (restriction | list | union)
〈/simpleType〉
```

그림 3. 글로벌 심플타입 정의

```
〈simpleType〉
   (restriction | list | union)
〈/simpleType〉
```

그림 4. 로컬 심플타입 정의

심플타입에서 자식 엘리먼트로 올 수 있는 것은 "restriction", "list", "union" 가 있으며 여기서 하나를 선택하여 기술할 수 있다. "restriction"은 내장된 심플타입 또는 이미 정의되어 사용되는 사용자정의 심플타입을 제한하여 새로운 심플타입을 정의할 때 사용하고, "list"는 공백 문자열로 분리된 토큰(token)들의 리스트를 값으로 갖고자하는 타입을 정의할 때 사용한다. 그리고 "unicn"은 여러 개의 심플타입을 결합하여 여러 종류의 데이터 값을 갖는 경우 사용한다. 또 다른 타입으로 사용되는 콤플렉스 타입은 속성을 가지거나 자식 엘리먼트를 가지는 엘리먼트의 선언에 필요한 타입으로써 이 역시 글로벌 콤플렉스 타입과 로컬 콤플렉스 타입이 있다. 그리고 <sequence>를 사용한 자식 엘리먼트의 사용에서는 "순차적 자식 엘리먼트 콤플렉스 타입"과 "선택적 자식 엘리먼트 콤플렉스 타입" 정의를 사용할 수 있다.

```
〈complexType name="complex type name"〉
  〈sequence〉
    Element ...
  〈/sequence〉
〈/complexType〉
```

그림 5. 순차적 자식 엘리먼트 콤플렉스 타입

<sequence>의 자식 엘리먼트로 대개 여러 개의 엘리먼트가 오지만 한 개의 엘리먼트만 사용되는 경우에도 반드시 <sequence> 엘리먼트를 삽입해야 한다. 그리고 "선택적 자식 엘리먼트 콤플렉스 타입"에서는 <choice>를 사용하여 선택적으로 엘리먼트를 사용할 수 있다.

```
〈complexType name="complex type name"〉
  〈sequence〉
    Elements ...
    〈choice  minOccurs="minimum  count"
maxOccurs="maximum count"
        Elements ...
    〈/choice〉
    Elements ...
  〈/sequence〉
〈/complexType〉
```

그림 6. 선택적 자식 엘리먼트 콤플렉스 타입

2. 변환규칙과 UML profile

메타모델에서 제시하고 있는 제약사항들에 대해서 UML에서 정확히 표현하는 데에는 상당한 어려움을 가질 수 있다. 그래서 UML은 복잡한 제약들이나 메타모델에서 추가적인 요구사항들을 지원하기 위해서 OCL를 이용하여 기능적인 표현을 하기도 한다. 현재 사용되고 있는 대부분의 모델설계 도구들이 OCL의 기능을 조금 지원하거나 아니면 거의 지원하지 않고 있다. 특히 지원하는 도구들 역시 사용상에 제한적이고 어려움이 있어 불편함을 감수 하여야 한다. 본 연구에서는 UML 이용하

여 XML 스키마의 마크업언어 생성에 대한 메타모델 설계와 OCL 명세를 위한 기반을 확립하였다.

일반적으로 DTD가 마크업언어로 많이 사용되고 있지만 데이터형식 표현에 대한 제약으로 인한 문제점들이 발생되어 이를 보완할 수 있는 것으로 여기서는 XML 스키마를 기반으로 적용하여 연구하였다.

XML 스키마 언어에서는 빌트인 심플타입이나 사용자 정의 심플타입의 기반으로 데이터의 형태를 정의한다. 본 논문에서는 아래의 그림과 같이 사용자 정의 심플타입의 메타모델에서 메타데이터를 생성하기 위해서 분류하였다. 여기서는 클래스간의 상속관계를 표시하고 있으나 세부적인 데이터의 사용이나 선택에 대한 구체적인 방법들을 나타내지 않아 이해하기 힘든 부분이 많다.

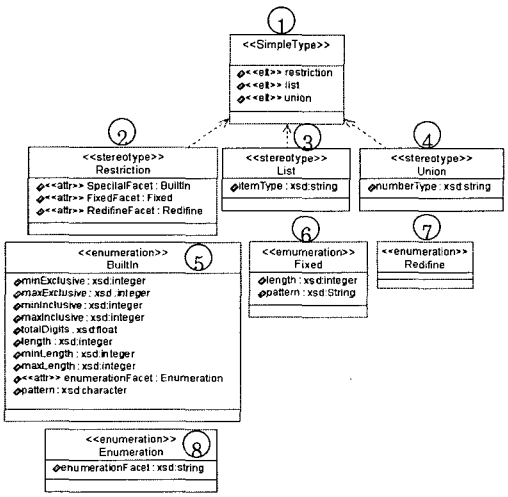


그림 7. simpleType 모델설계

[그림 7]에서 simpleType 해당하는 클래스의 개체에 대한 OCL으로 표기[12]에서 다음과 같이 프로그램을 작성하였다. 여기서 세 개의 엘리먼트에 대한 선택에서는 조건문을 사용하여 하나의 엘리먼트만이 적용이 가능하도록 한 것이다.

[그림 7]에서 표현된 메타모델의 상관관계나 메타데이터의 사용에 정확하고 쉬운 방법을 적용하기 위해서 OCL을 위한 파스 트리(parse tree)를 도식하는 방법을 생성하였다. 본 연구에 사용된 메타모델에서 OCL 표현은 context name inv : 형식을 사용하였다.

```
①
Context SimpleType inv:
if Element.type.BuiltInType then
  Element = self.expression -> select(restriction) -> notEmpty()
else if Element.type.BlankTokenType then
  Element = self.expression -> select(list) -> notEmpty()
else
  Element = self.expression -> select(union) -> notEmpty()
end if
```

그림 8. simpleType 클래스 OCL 명세

그리고 이를 파스 트리를 이용하여 표현하였으며 점선으로 되어있는 부분은 지정된 언어의 문장을 토큰으로 나눈 것이고 실선으로 된 사각형은 메타데이터의 임의의 프로그램 작성에 대한 토큰이다.

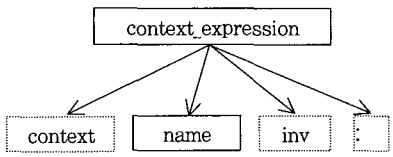


그림 9. context parse tree

①에 해당되는 사용자정의 심플타입의 데이터형을 선택하는 OCL의 파스 트리에서 simpleType의 형태로 정의가 된다.

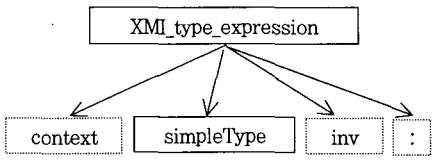


그림 10. simpleType parse tree

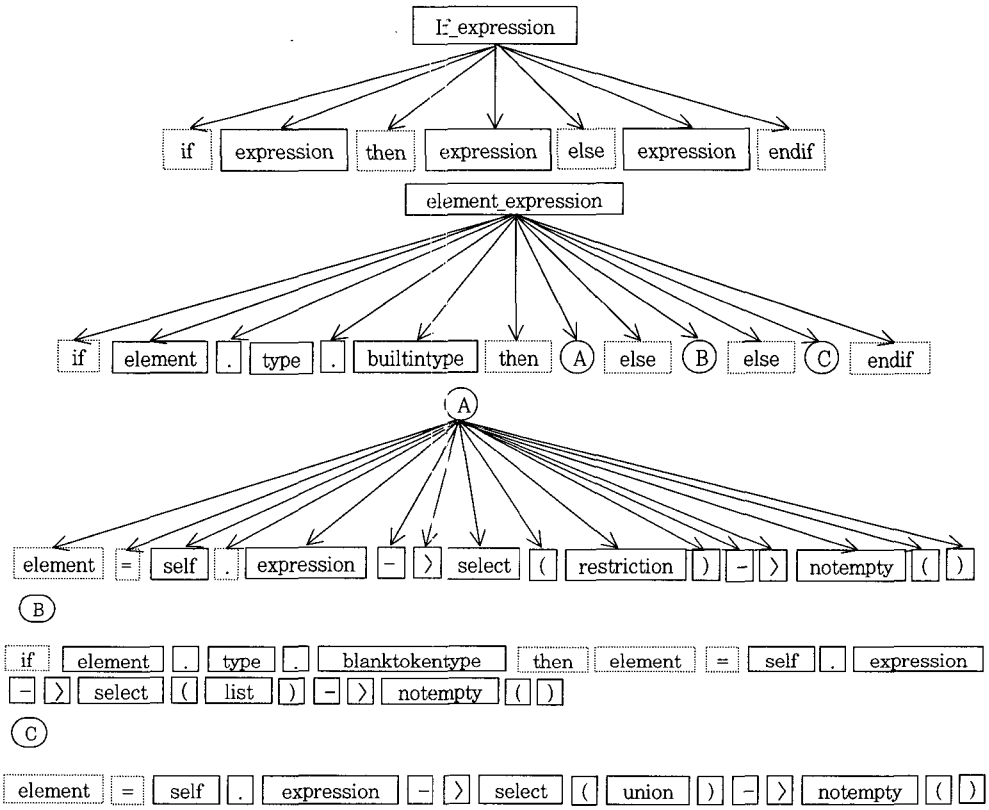


그림 11. 엘리먼트 선택 파스 트리

그리고 simpleType 클래스에서 3개의 엘리먼트 중 하나를 선택할 수 있다. 그러므로 파스 트리에서 엘리먼트 선택에 대한 조건문을 생성할 수 있다. 조건문을 선택할 수 있는 파스 트리에서 실선으로 된 사각형에 대한 OCL의 명세에 대해 세부적인 구문표기에 대한 각각의 토큰을 생성할 수 가 있다.

소스프로그램을 읽어 들여 컴파일러를 이용하여 다른 목적 프로그램으로 변환시키는 과정에서 어휘분석을 통하여 소스 프로그램의 character와 symbol들을 토큰으로 변환한다. 그리고 구문에 대한 규칙을 분석하는데 일반적으로 context-free 문법을 사용하여 표현한다. 이 부분은 파서라고 하며 어휘분석 단계에서 토큰을 받으며, 이 토큰을 규칙적인 파스트리로 생성한다. context-free 문법은 일반적으로 BNF를 사용하며, 본 연구에서는 아래의 그림과 같이 EBNF 표기법을 확장하여 적용하였다.

```
②
Context Restriction inv:
if attribute.type.SpecialFacet then
  attribute = self.Facet -> select(BuiltIn) ->
notEmpty()
else if attribute.type.fixedFacet then
  attribute = self.Facet -> select(Fixed) ->
notEmpty()
else
  attribute = self.Facet -> select(Redefine)
-> notEmpty()
end if
```

그림 12. Restriction 클래스 OCL 명세

IV. 평가 및 결론

본 논문에서 제시하고 있는 방법은 UML 모델설계에서 표현하기 어려운 클래스의 다중상속과 조건부 엘리먼트의 선택적 사용에 대한 모호한 표현에 대해 정확한 명세를 지원할 수 있도록 하였다. 그러나 OCL이 UML 메타모델의 합성에서 정형화된 정의에 어려움을 가지고 있어서 메타모델설계에 포함되는 경우가 드물다. 몇몇의 UML과 관련된 툴 중에는 OCL의 가능을 다소 지원하는 경우를 보면 현재 Oclarity 등의 툴들이 Rational Rose에서 사용을 지원하고 있다. 그리고 Dresden OCL 툴킷은 라이브러리로 OCL를 제공하는 연구를 하고 있다 [13-15]. 이와 관련된 논문들을 보면 메타모델에 대한 초보수준의 OCL을 에디터를 통하여 입력하여 이를 컴파일하여 오류의 유무를 확인하는 기능을 갖추고 있을 뿐 메타데이터를 생성하기 위한 마크업언어 설계와 관련된 연구는 미진한 부분이 많다.

또한 본 연구에서는 이러한 기존의 연구부분에 추가적으로 XML 스키마를 이용한 마크업언어 생성을 위한 메타모델의 설계와 이와 관련되어 OCL로 표기할 수 있는 방법을 제시하면서 이를 컴파일의 과정에서 어휘분석과 구문분석에 대한 세부적인 설계방법을 제안하였다.

그 결과 XML 스키마 마크업언어에 대한 메타모델설계 뿐만 아니라 사용자가 이해하기 어려운 클래스간의 관계, 엘리먼트의 타입 적용 등을 명확히 할 수 있었다. 그러나 ⑥의 EBNF를 이용한 표기법에서 pattern value = "\d{6} - \d{7}"나 "[a-o]x"의 식에 대한 OCL 표기법에는 문제점들이 나타나고 있다. 이와 같이 OCL은 아직도 초보단계의 미완성부분을 많이 가지고 있는 언어이다. 앞으로도 꾸준히 비중 있는 연구가 이루어져야 할 것이다.

그리고 아울러 추가적인 연구로는 마크업언어 생성에서 메타모델설계에 관련된 컨트롤개체를 이용한 OCL 표기법에 대한 자동 테스트생성과 메타모델에서 클래스 개체간의 관계를 표현하고 테스트하면서 어려운 OCL의 문법적인 기능을 사용자가 쉽게 사용할 수 있는 추가 연구가 많이 요구된다.

참 고 문 헌

[1] F. Finger, *Design and Implementation of a modular OCL compiler*, Master Thesis Dresden University of Technology. Mar., 2000.

[2] S. Gaito, S. Kent, and N. Ross, "A Meta-Model Semantics for Structural Constraints in UML. Behavioural Specifications for Business and Systems," pp.123-141, Sep., 1999.

[3] M. Gogolla and M. Richters, "Development of UML Descriptions with USE. In First Eurasian Conference on Information and Communication Technology," LNCS 2510, pp.228-238, 2002.

[4] H. Hussmann, B. Demuch, and F. Finger, "Modular Architecture for a Toolset Supporting OCL," LNCS, pp.278-293, 2000.

[5] K. Baclawski, M. K. Kokar, P. A. Kogut, L. Hart, J. Smith, W. S. Holmes III, J. Letkowski, and M. L, Aronson, "Extending UML to support Ontology Engineering for the Semantic Web," LNCS 2185, pp.342-360, Springer, 2001.

[6] R. CONRAD, D. SCHEfFNER, and J. C. FREYTAG, "XML Conceptual Modeling Using UML. Proc," International Conceptual Modeling Conference, LNCS, pp.558-557, 2004.

[7] N. Routledge, L. Bird, and A. Goodchild, "UML and Schema," Thirteenth Australasian Database Conference, Vol.5, 2002.

[8] http://www.rational.com.

[9] http://www.rational.com/media/whitepapers /TP189draft.pdf

[10] G. D. Pollet, Yves le Traon, and Jean-Marc Jezequel, "Refactoring uml models," Proceedings of UML 2001, volumn 2185 of LNCS, pp.134-148, 2001.

[11] M. Casanova, T. Wallet, and M. D. Hondt, "Ensuring Quality of Geographic Data with UML and OCL," LNCS 1939, pp.225-239, 2000.

[12] M. Richters and J. G. Warner, "Object Modeling with the OCL," LNCS 2263, pp.42-68, 2002.

[13] A. S. Kent and B. Selic, "UML 2000-The Unified Modeling Language. Advancing the Standard," Proceedings volume 1939 of LNCS, pp.440-450, 2000.

[14] M. M. Gogolla, M. R. Richters, and B. Rumpe, "UML'99-The Unified Modeling Language. Beyond the standard," LNCS 1723, pp.156-171, Oct., 1999.

[15] G. C. Gannod and J. T. E. Timm, An MDA-based Approach for Facilitating Adoption of Semantic Web Service Technology, Proceedings of the 8th IEEE Enterprise Distributed Object Computing Conference Workshop on Model-Driven Semantic Web, Sep., 2004.

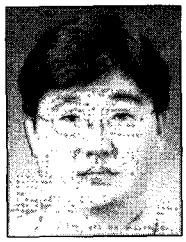
저 자 소 개

이 돈 양(Don-Yang Lee) 정회원

- 1987년 2월 : 대구대학교 통계학과(이학사)
- 1993년 2월 : 경희대학교 전자계산공학과(공학석사)
- 2004년 9월 : 경희대학교 전자계산공학과(공학박사)
- 1995년 3월~2002년 8월 : 대한상공회의소
- 2003년 3월~2006년 2월 : 경인여대 겸임교수
- 2006년 3월~현재 : 세종대학교 컴퓨터공학과 초빙교수

<관심분야> : Design Pattern, AOP, OCL, XMI, MDA

최 한 용(Han-Yong Choi) 정회원



- 1994년 2월 : 경희대학교 전자계산공학과(공학사)
- 1998년 2월 : 경희대학교 전자계산공학과(공학석사)
- 2002년 8월 : 경희대학교 전자계산공학과(공학박사)
- 2002년 9월~2004년 8월 : 경희대학교 전자정보학부 강의교수
- 2004년 3월~현재 : 한북대학교 컴퓨터공학과 교수

<관심분야> : Component Design, Design Pattern, XMI, MDA