한국산학기술학회논문지
Vol. 9, No. 2, pp. 415-421, 2008

# 효율적인 온톨로지 개발을 위한 UML의 변경

김영태[1], 임재현[1], 김치수[1*]

# The Modification of UML for Developing of the Efficient Ontology

Young-Tae Kim[1], Jae-Hyun Lim[1] and Chi-Su Kim[1*]

**요 약** 정보의 복잡도와 다양성의 증가뿐만 아니라 현재 이용 가능한 대용량의 정보로 인해 온톨로지에 대한 관심이 증가하고 있다. 이러한 경향은 전통적으로 수동으로 수행되던 많은 활동의 자동화에 대한 관심도 증가시켰다.
본 논문에서는 복잡한 OWL 온톨로지를 UML 클래스 다이어그램을 이용해서 개발하고 표현함으로써 생산성과 명료함을 향상시키기 위한 연구를 수행한다.
UML은 대부분의 온톨로지 언어에서 일반적으로 이용할 수 없는 프로파일, 대역 모듈성, 확장 메커니즘 등의 많은 특징을 갖고, 온톨로지 언어는 UML이 지원하지 않는 일부 특징을 갖는다.
본 논문에서는 UML과 온톨로지 언어 RDF, OWL 사이의 유사성과 차이점을 확인하고, 상당히 문제가 있는 차이점을 다루기 위해 UML 메타 모델의 변경을 제안한다.

**Abstract** The increasing interest in ontologies is driven by the large volumes of information now available as well as by the increasing complexity and diversity of this information. These trends have also increased interest in automating many activities that were traditionally performed manually
We are currently engaged in this paper that have realized benefits in productivity and clarity by utilizing UML class diagrams to develop and to display complex OWL ontologies.
UML has many features, such as profiles, global modularity and extension mechanisms that are not generally available in most ontology languages. However, ontology languages have some features that UML does not support.
Our paper identifies the similarities and differences between UML and the ontology languages RDF and OWL. To reconcile these differences, we propose a modification to the UML metamodel to address some of the most problematic differences.

**Key Words :** UML 변경(UML Modification), 온톨로지 개발(Ontology Development)

## 1. 서론

웹의 빠른 성장은 실행 시간에 지식과 서비스를 추리하고 동적으로 통합하는 에이전트를 가능하게 했다. 웹에서 사용가능한 에이전트는 이러한 요구를 처리하기 위한 하나의 기술이다[1]. 이러한 에이전트는 지식에 관해 추리할 수 있고, 실행 시간에 서비스를 동적으로 통합할 수 있다. 온톨로지는 이런 에이전트를 위한 기반이다. OWL은 에이전트 통신과 추론을 지원하기 위해 설계된다.

본 연구는 한국과학재단 특정기초연구
(R01-2006-000-10555-0)지원으로 수행되었음
[1]공주대학교 컴퓨터공학부
*교신저자 : 김치수(cskim@kongju.ac.kr)

복잡한 온톨로지를 시각화하고 온톨로지 개발 프로세스를 관리하기 위해 UML 기반의 온톨로지 개발을 하고자 한다. 따라서 UML과 OWL의 유사점과 차이점에 대하여 논하고, 차이를 조정할 수 있는 방법에 대하여 논하고자 한다.

UML과 OWL은 많은 공통점을 갖고 있다. 예를 들면, 둘 모두 인스턴스를 가질 수 있는 클래스의 개념을 갖는다. 그러나 상당히 문제가 있는 차이점도 있다. relation, predicate 또는 프로퍼티로 불리는 지식표현 개념이다. 이 지식표현 개념은 언뜻 보면 UML의 연관 혹은 애트리뷰트로 보인다. UML의 연관과 애트리뷰트는 최상위 클래스이지만 상응하는 지식표현 개념에서는 최상위 클래스가 아니기 때문에 오해의 소지가 있다. 더 정확하게 말하면 지식표현 시스템에서 relation은 관련된 어떤 클래스를

명시하지 않고도 존재할 수 있다.

일부 지식표현 시스템에서 주된 모델링의 기본은 프로퍼티이고, 클래스는 두 번째 단계에서 실제적으로 관계된다. 이런 지식표현 시스템에서 전체는 프로퍼티를 갖는 인스턴스로 구성된다고 가정한다. 반면 UML에서 연관은 식별자와 연관되어야하는 연관 끝으로 정의된다.

본 논문에서는 복잡한 OWL 온톨로지를 UML 클래스 다이어그램을 이용해서 개발하고 표현함으로써 생산성과 명료함을 향상시키기 위하여 상당히 문제가 있는 문제를 다루기 위한 UML 메타 모델의 변경을 제안한다.

## 2. 관련연구

### 2.1 지식 표현 언어

지식의 표현은 모든 지식 기반 시스템의 중요한 부분이다. 특히, 모든 인공 지능 시스템은 어떤 종류든 지식 표현을 지원해야만 한다. 이것 때문에, 많은 지식표현 언어가 개발되었다[2].

기계가 읽을 수 있는 형식으로 지식을 나타내는 것은 그것이 데이터로 표현되는 것을 요구한다. 따라서 지식표현 언어와 데이터 언어가 많은 공통점을 갖으며, 서로의 아이디어와 개념을 사용했다. 객체 지향 프로그래밍과 모델링 언어의 상속은 지식표현 언어에 상응하는 개념부터 넓은 범위까지 유도되었다.

지식표현 언어는 대략 3개의 카테고리로 분류할 수 있다.

논리 언어 : 논리적 문장으로 지식을 나타낸다. 이런 지식표현 언어의 가장 유명한 예는 Knowledge Interchange Format(KIF)이다[3].

프레임 기반 언어 : 객체 지향 데이터베이스 언어와 유사하다.

**[표 1]** UML과 OWL의 매핑

| UML | OWL |
|---|---|
| Class | Class |
| "classifier" role | type |
| type of ModelElement | type |
| attribute | ObjectProperty or DatatypeProperty |
| association end | ObjectProperty |
| specialization of classes | subClassOf |
| specialization of associations | subPropertyOf |
| note | comment |
| name | label |
| "seeAlso" tagged value on a class and association | seeAlso |
| "isDefinedBy" tagged value on a class and association | isDefinedBy |
| Not supported | "Restriction" class |
| Not supported | "complementOf" property |
| Not supported | "unionOf" property |
| Not supported | "intersectionOf" property |
| "onProperty" association | onProperty |
| "toClass" association | oClass |
| class containing the attribute | "subClassOf" a property restriction |
| source class of anassociation | "subClassOf" a property restriction |
| attribute type | "toClass" on a property restriction |
| target class of anassociation | "toClass" on a property restriction |
| Not supported | equivalentTo |
| Not supported | sameClassAs |
| Not supported | samePropertyAs |
| Not supported | sameIndividualAs |
| Not supported | differentIndividualFrom |
| Package | Ontology |
| "versionInfo" tagged value on a package | versionInfo |
| import (dependency stereotype) | imports |
| multiplicity | cardinality |
| multiplicity range Y..Z | Y = minCardinality, Z = maxCardinality |
| association target with end multiplicity = 0..1 or 1 | UniqueProperty |
| association source with end multiplicity = 0..1 or 1 | UnambiguousProperty |
| relationship between the association ends of an association | inverseOf |
| Not supported | TransitiveProperty |

그래프 기반 언어 : 시멘틱 네트워크와 개념 그래프를 포함한다. 지식은 노드와 노드 사이의 링크를 이용해 표현된다. Sowa의 개념 그래프가 좋은 예이다[2].

지식표현 언어는 데이터 언어처럼 스키마에 따르는 데이터의 구조와 제약조건을 정의하는 스키마 표현 능력을 갖는다. 지식표현 언어의 스키마는 온톨로지로 불린다[4, 5, 6, 7].

온톨로지는 명백하고 형식에 맞는 시멘틱 모델이다. 전형적으로 일부 자연 언어 텍스트나 일반적인 하이퍼텍스트 문서를 요약하거나 주석을 달게 되면 온톨로지에 적합한 데이터는 주석이나 마크업으로 자주 참조된다. 온톨로지는 어휘 조건, 분류, 관계, 규칙/주장을 포함할 수 있으며, 인스턴스/주석을 포함해서는 안 된다.

### 2.2 RDF(Resource Description Framework)

하이퍼텍스트와 시멘틱 네트워크는 유사할 수밖에 없다. 만일 보편적인 자원식별자나 URI로 식별되는 웹 자원으로 시멘틱 네트워크 노드를 식별하고, 하이퍼텍스트 링크로 시멘틱 네트워크 링크를 식별하면, 그 결과는 전체 웹으로 확장할 수 있는 지식 표현을 나타내는 토대를 형성한다. 이것은 RDF의 본질이다[8].

RDF는 W3C의 주도로 개발된 XML 표준 내의 권고안이다. RDF는 데이터에 대한 어떤 것을 말하기 위한 메커니즘이다. 이것의 이름은 어떠한 것을 지시한다. 언어가 아니고 어떠한 것에 관한 데이터를 표현하기 위한 모델이다. RDF 스키마 언어가 있고 RDF와 RDF 스키마를 처리할 수 있는 많은 도구와 제품이 있다.

RDF 스키마는 RDF를 위한 간단한 타입 시스템이다. RDF 스키마는 도메인 특정적인 내용을 정의하기 위한 메커니즘을 제공하고, 속성에 적용할 수 있는 자원 클래스이다.

### 2.3 OWL(Web Ontology Language)

OWL은 논리적인 추론을 지원할 뿐 아니라, 보다 풍부하고 다양한 제약조건이 표현 가능한 RDF와 RDF 스키마(RDF-S) 기반 언어이다.

OWL은 소프트웨어 에이전트가 웹 정보의 의미를 해석할 수 있고, 설명할 수 있도록 하기 위해서 웹상의 정보에 주석을 달아서 기계 해석이 가능한 지식을 만들 수 있다. OWL은 풍부한 어휘와 시멘틱을 포함하고 있기 때문에 기계 해석이 가능한 웹 컨텐트를 저작하는데 있어 XML, RDF 및 RDF 스키마보다 뛰어나다[9].

OWL은 단지 사람에게 정보를 표시하는데 그치지 않고 정보의 내용을 직접 처리할 수 있는 어플리케이션을 구현하는데 활용될 수 있도록 설계된 언어이다. OWL은 표현력이 서로 다른 세 개의 하위 언어 OWL Lite, OWL DL, OWL Full로 구성되어 있으며, 후자로 갈수록 표현력이 더 크다.

이런 엔진과 도구는 다중 웹 페이지 검색이 가능하고, 프로그래머가 콘텐츠를 식별하기 위해 사용하는 언어의 변화나 부정확함으로 인해 현재는 관련이 없어 보이는 내용을 개념적으로 관련시킬 수 있을 것이다.

## 3. OWL과 UML의 매핑

UML과 OWL 사이의 유사성을 논하기 위해 언어 사이의 매핑을 생성했다. [표 1]은 UML과 OWL의 고수준 개념을 설명하고, 언어 사이의 초기의 매핑에 필요한 구체적인 확장의 일부를 나타낸다.

매핑은 UML 클래스 다이어그램이 명확히 OWL 온톨로지를 설계할 목적으로 만들어진다는 조건으로 생성된다. 애초에 OWL 어플리케이션에 사용할 목적이 아니었던 클래스 다이어그램은 OWL 목적을 위해 사용은 가능하지만 OWL 능력을 충분히 이용하기 위해서는 수정이 필요하다. [표 1]에서 "Not supported"로 표기된 구성요소는 UML의 적당한 스테레오타입을 도입하여 추가된다.

### 3.1 서브클래스 관계 표현

UML의 일반화 개념과 OWL의 subClassOf 개념의 차이는 일관성과 정당성의 좋은 예이다. UML의 특수화/일반화는 가장 일반적으로 객체 지향 프로그래밍 언어에서 서브 클래스/슈퍼 클래스 관계를 표현하기 위해 사용된다. OWL의 subClassOf 프로퍼티가 집합론으로 정의되는 반면, 특수화/일반화 관계는 기본적으로 행동적이다. 이 두 관점은 공식적으로 양립할 수 없다. 이 두 관점이 서로 양립할 수 없는 이유를 보기 위해, 다음과 같이 정의된 클래스 Square와 Rectangle을 고려해보자.

```
class Square {
        double length;
        Square(double l) { length = l; }
        double area() { return length * length; }
        void magnify(double scale) {
                length = scale * length; }
}
class Rectangle {
        double length, width;
        Rectangle(double l, double w) {
```

```
        length = l; width = w; }
    double area() { return length * width; }
    void magnify(double scale) {
        length *= scale; width *= scale; }
    void stretchLength(double s) { length *= s; }
    void stretchWidth(double s) { width *= s; }
}
```

집합론적인 관점에서 정사각형 집합이 직사각형 집합의 부분 집합이기 때문에 square는 rectangle의 서브 클래스일 수 있다. 그러나 이 관점은 정사각형과 직사각형의 행동에 관한 관점을 무시한다. 실제 대부분의 객체 지향 프로그래밍에 언어에서 Square 클래스를 Rectangle의 서브클래스로 쉽게 설정할 수 없다. 이러한 시도는 대용가능성의 원칙과 충돌할 것이다. 정사각형은 확대될 수 없기 때문에 정사각형 객체를 직사각형으로 간주할 수 없다.

반면에, 행동에 관한 관점에서 직사각형은 정사각형의 서브클래스일 수 있다. 실제로, 다음과 같이 정의할 수 있다.

```
class Rectangle extends Square {
    double width;
    Rectangle(double l, double w) {
        super(l); width = w; }
    double area() { return length * width; }
    void magnify(double scale) {
        length *= scale; width *= scale; }
    void stretchLength(double s) { length *= s; }
    void stretchWidth(double s) { width *= s; }
}
```

이 예로부터, UML의 일반화는 의미적으로 OWL의 subClassOf 프로퍼티와 아주 다른 것이 명백해진다. 그럼에도 불구하고 일반화 관계를 사용하여 subClassOf 프로퍼티가 의미적인 동치를 유지하도록 명세한다. 왜냐하면, 하나가 다른 하나의 특수화 관계인 두 클래스는 OWL의 subClassOf 관계의 두 클래스로 매핑 될 것이기 때문이다. 따라서 UML 일반화를 OWL subClassOf로 매핑하는 것은 모순이 아니다.

### 3.2 OWL 프로퍼티

[그림 1]은 Person 클래스와 Woman 클래스 사이에 존재하는 mother 관계를 보여준다. UML에서 이것은 두 클래스 사이의 연관으로 표현된다.

UML에서 이것은 두 클래스 사이에서 레이블이 붙은 연관으로 표현된다. OWL에서 mother 프로퍼티는 두 클래스에 독립적으로 존재한다. 그리고 만약 적당한 제약이 강요되지 않으면 어떤 객체라도 mother일 수 있다. [그림 1]에서 이것은 Person 클래스의 프로퍼티 mother에 대해 Woman 클래스로의 제약으로 표현된다. 동물, 회사 등에서 어머니 개념을 표현하기 위한 다른 제약 또한 사용될 수 있다.

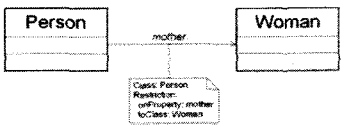



**그림 1.** OWL 프로퍼티 제약

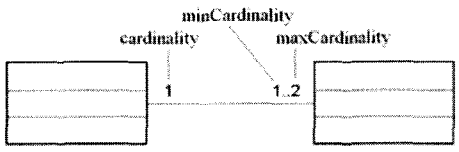
OWL 변환은 제안된 OWL과 UML의 매핑을 적용함으로써 생성될 수 있다. [그림 2]는 [그림 1]로부터 만들어진 온톨로지의 일부를 보여준다.

```
<owl:Property rdf:ID="mother"/>
<owl:Class rdf:ID="Person">
  <owl:label>Person</owl:label>
  <owl:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#mother"/>
      <owl:toClass rdf:resource="#Woman"/>
    </owl:Restriction>
  </owl:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Woman">
  <owl:label>Woman</owl:label>
</owl:Class>
```

**[그림 2]** OWL 변환

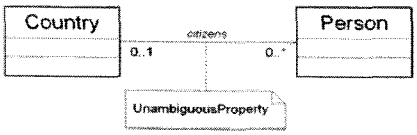
### 3.3 프로퍼티의 표현

[그림 3]은 UML의 다중도와 OWL의 카디널리티 사이의 매핑을 보여주기 위해 연관 끝의 다중도에 상응하는 OWL의 카디널리티를 보여준다. 단일 값을 포함하는 연관 끝은 명확한 카디널리티 값에 매핑하고, 범위 값을 포함하는 연관 끝은 허용되는 최소와 최대 카디널리티에 매핑한다.



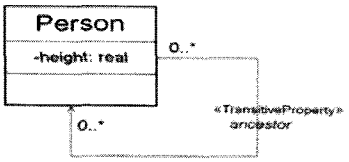

**[그림 3]** OWL 카디널리티

[그림 4]와 [그림 5]는 미리 정의된 카디널리티 제약과 함께 OWL 프로퍼티의 특수한 경우를 묘사한다. [그림 4]의 명확한 프로퍼티는 OWL에서 항상 같은 소스 요소로부터 시작되어 정해진 대상 요소로 전달되는 관계로서 정의된다.




**[그림 4]** 명확한 프로퍼티의 예

[그림 5]는 OWL의 Transitive 프로퍼티 개념에 대한 UML 표기법을 나타낸다. Transitive 프로퍼티로 간주되려면 첫 번째와 두 번째 요소에 대해 유효하고, 두 번째와 세 번째 요소에 대해 유효한 프로퍼티는 첫 번째와 세 번째 요소에 대해서도 유효해야만 한다. 즉, A가 B의 조상이고, B가 C의 조상이면 A가 C의 조상이어야 한다.

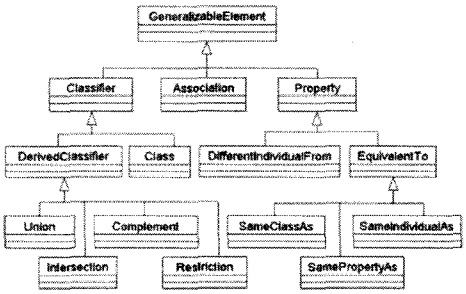



**[그림 5]** Transitive 프로퍼티의 예

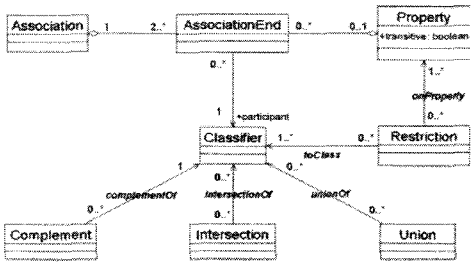
## 4. 제안하는 UML 변경

본 논문에서는 불 연산과 수량화를 사용하여 식별자를 구축할 뿐만 아니라, 첫 번째 클래스 프로퍼티를 사용하여 모델링을 가능하게 하는 UML 메타모델을 변경한다. 본 논문의 제안을 위한 메타 모델은 [그림 6]과 [그림 7]에 나타나 있다. 그림에서 Property, Restriction, Union, Intersection, Complement를 제외한 메타 클래스는 이미 UML의 일부이다. 먼저 프로퍼티 메타 클래스의 정당성과 의미를 보이고, 그 다음 식별자 구성요소를 고려한다.

이것이 완전한 명세라고는 할 수 없다. 본 논문의 의도는 문제를 제기하고, 해결책을 제시하는 것이다. 충분한 명세는 다른 모델 요소의 집합을 고려해야만 한다. 예를 들면 프로퍼티를 사용하여 연관 끝이 모여지는 것과 같은 방법으로 애트리뷰트와 메서드를 모을 수 있다.




**[그림 6]** UML 변경 : 메타클래스




**[그림 7]** UML 변경 : 메타 연관

### 4.1 프로퍼티

OWL은 수학적인 그래프 또는 네트워크 개념에 의거하는 많은 다른 지식표현 언어와 유사하다. OWL에서 술부는 프로퍼티에 의해 표현된다. 그러나 그것이 사용될 수 있는 문맥이라도 프로퍼티의 OWL 개념은 독립적으로 정의된다.

프로퍼티가 모델링 작업에 적합하지 않더라도 특별한 정지를 하지 않는다. 이 결정은 모델러가 해야 한다. 그렇게 하는 것이 편리하고 기존의 어떤 모델도 파손시키지 않는다면 이것이 모델링 기술을 지원하는 이치에 맞는다. 프로그래밍 언어 커뮤니티는 크로스 커팅 관점이라 불리는 첫 번째 클래스 프로퍼티의 개념을 소개했다[10]. 모델링 관점에 대한 지원 역시 기술되어 있다.

본 논문에서는 UML 메타 모델에 약간의 추가적인 모델 요소를 추가함으로써 관점 지향적 프로그램 작성을 지원할 뿐만 아니라, UML을 지식 표현 언어와 호환성이 있도록 하고자 한다. 이를 위해 [그림 6], [그림 7]과 같이 프로퍼티라고 불리는 메타 클래스를 추가함으로써 UML을 변경한다. 다이어그램에서와 같이 프로퍼티는 많은 연관 끝의 집합이다. 프로퍼티의 개념은 많은 연관 끝을 모으는 역할을 한다.

프로퍼티가 첫 번째 클래스 개념이라는 사실은 프로퍼티가 어떤 클래스와도 관련되지 않고 존재할 수 있다는 사실에 의해 증명된다. 프로퍼티는 제한의 사용에 의해

제약될 수 있다.

UML에서 메타 모델 요소에 대한 의미는 OCL을 사용하여 명시된다. 프로퍼티는 연관 끝들의 그룹화이다. OCL에서 이것을 표현하는 것은 연관의 모든 연관 끝의 집합인 allConnections를 사용한다. 그리고 연관의 일반화 가능한 요소의 모든 프로퍼티 집합인 allPropConnections를 사용한다. 만일 T가 allConnections와 allPropConnections 집합의 논리곱이라면 T는 많아야 1의 카디널리티를 갖는다. 보다 형식적으로 보면 다음과 같다.

```
allConnections: Set(AssociationEnd);
allPropConnections: Set(Property);
self.allConnections->intersection(self.allPropConnections
:Set(T)):Set(T);
size(#T)<=1
```

또한, 추가적으로 프로퍼티가 다른 프로퍼티의 특수화나 일반화가 가능한지 명시해야 한다.

### 4.2 클래스 구성요소

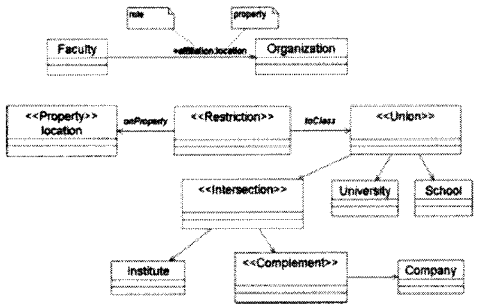
본 논문에서는 UML에 OWL에서처럼 불 연산과 한정 기호를 사용하여 식별자를 구축하는 메커니즘을 제안한다. 본 논문의 제안을 위한 메타 모델은 [그림 6]과 [그림 7]에 나타나 있다.

메타 클래스 Intersection, Union, Complement는 각각 OWL의 프로퍼티 intersectionOf, unionOf, complementOf의 역할을 한다. 메타클래스는 목록 순서화이지만 OWL은 목록 순서화를 조금도 이용하지 않는다. Restriction 메타 클래스는 OWL의 메타 클래스 개념과 거의 같다.

[그림 7]에서는 toClass 제한의 경우를 보였다. 제한은 객체를 위한 식별자다. 제한의 인스턴스는 제한과 관련된 하나 이상의 프로퍼티에 대해서 조건을 만족시키는 객체다. 제한은 클래스가 제한 식별자의 특수화인 것을 명시하도록 강요한다. 제한 식별자가 onProperty에 의해 프로퍼티와 연결되거나 toClass 메타 연관에 의해 클래스와 연결되면 제한 식별자의 인스턴스는 명시된 클래스 중 하나의 객체와만 연결될 수 있다. 프로퍼티와 같이 제한 식별자는 다른 제한 식별자의 일반화나 특수화가 될 수 있다.

### 4.3 구성요소의 사용 예

[그림 8]은 제안하는 구성요소의 사용 예이다. [그림8]에서 구성요소는 Faculty와 Organization 사이의 연관이고, 연관 끝의 역할은 affiliation으로 불린다. 콜론으로 분리된 프로퍼티 이름을 연관 끝에 추가함으로써 이 연관 끝이 location 프로퍼티에 속하는 것을 보여준다. 프로퍼티 자체는 다른 곳에 정의된다. [그림 8]에서는 클래스 그래픽 심볼을 사용하여 보여주고, 프로퍼티 스테레오타입을 사용하여 수정한다.




**[그림 8]** UML 변경을 사용한 모델의 예

식별자 구성요소 또한 같은 클래스 그래픽 심볼을 사용하여 보여주고, 적당한 스테레오타입을 사용하여 변경된다. Union 식별자는 school, university, 그리고 company가 아닌 institute의 합집합을 명시한다. Restriction은 객체의 location이 school, university, 그리고 company가 아닌 institute 식별자를 명시한다. Faculty 클래스가 Restriction의 특수화이기 때문에, 교직원의 입회는 school, university, 그리고 company가 아닌 institute의 조직이어야 한다.

## 5. 결론

본 논문에서는 온톨로지 개발에 UML을 사용하기 위한 UML의 변경을 제시했다. UML과 지식표현 언어 사이의 유사점과 차이점을 확인하고 OWL의 경우 서로 어떻게 매핑 될 수 있는지 설명하였으며, UML 개념을 OWL 개념으로 변환하는 규칙을 만드는 시도를 했다.

OWL 프로퍼티의 개념이 UML 연관 개념과 다소 유사하지만 쉽게 매핑 될 수 없다. 이것이 OWL 기반 온톨로지 개발을 위해 UML과 UML 도구를 사용하는데 큰 장애물이지만 UML 메타 모델의 변경에 의해 조정 될 수 있다.

또한 본 논문에서는 불 연산과 수량화를 사용하여 식별자를 구축하는 방법을 소개했다. 이것은 많은 연관에 모두 같이 강요될 수 있는 크로스 커팅 제약 조건을 명시하는데 유용하다. UML에 대한 변경의 동기는 UML을 지

식표현 언어와 관점 지향적 프로그램 작성에 호환성이 있게 하기 위한 것이다. 본 논문의 결과로 OWL과 같은 지식표현 언어의 그래픽 표기법으로 UML의 사용을 선도할지도 모른다.

향후 연구로는 hasValue, hasClass, minCardinality와 같이 본 논문에서 명시되지 않은 제한을 포함시켜야 하며, 제안된 UML의 변경을 실무 프로젝트에 적용하여 UML에서 OWL로의 변환을 통한 온톨로지 구축을 이루고자 한다.

## 참고문헌


[1] McGuinness, D.: Ontologies and online commerce. IEEE Intelligent Systems, 2001

[2] Sowa, J. (ed.): Knowledge Representation: Logical, Philosophical, and Computational Foundations. PWS Publishing, 2000

[3] Genesereth, M.: Knowledge Interchange Format draft proposed American National Standard (dpANS) NCITS.T2/98-004, 1998. Available at: logic.stanford.edu/kif/dpans.html

[4] Falkenberg, E.D., Lyytinen, K., Verrijn-Stuart, A.A.(eds.): Ontological Evaluation of the OML Metamodel, IFIP Conference Proceedings, 2000

[5] Heflin, J., Hendler, J., Luke, S.: Coping with changing ontologies in a distributed environment. In: AAAI-99 Workshop on Ontology Management. MIT Press, 1999

[6] Heflin, J., Hendler, J., Luke, S.: SHOE: A knowledge representation language for Internet applications. Technical Report www.cs.umd.edu/projects/plus/SHOE, Institute for Advanced Studies, University of Maryland, 2000

[7] McGuinness, D.L., Fikes, R., Rice, J., Wilde, S.: An environment for merging and testing large ontologies. In: Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning, 2000

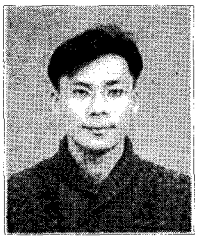
[8] Lassila, O., Swick, R.: Resource description framework (RDF) model and syntax specification, www.w3.org

[9] OWL. Ontology Web Language Web Site, www.w3.org

[10] Elrad, T., Filman, R., Bader, A.: Aspect-oriented programming: Introduction. Comm. ACM, 44(10): 29.32, October 2001



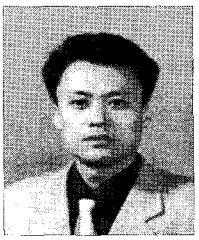
**김 영 태(Tae-Young Kim)** [정회원]



- 2000 공주대학교 전자계산학과 졸업(학사)
- 2002 공주대학교 전자계산학과 졸업(석사)
- 2004년 공주대학교 컴퓨터공학과 박사과정 수료

<관심분야>
데이터통합, UML, 온톨로지

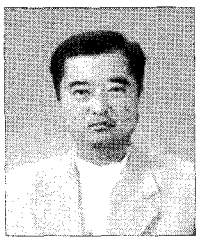
**임 재 현(Jae-Hyun Lim)** [정회원]



- 1986년 중앙대학교 전자계산학과 졸업(학사)
- 1988년 중앙대학교 대학원 전자계산학과 졸업(석사)
- 1998년 중앙대학교 대학원 컴퓨터공학과 졸업(박사)
- 1998년 ~ 현재 공주대학교 컴퓨터공학부 부교수

<관심분야>
상황인식, RFID/USN, 온톨로지, 인터넷 기술

**김 치 수(Chi-Su Kim)** [정회원]



- 1984년 중앙대학교 전자계산학과 졸업(학사)
- 1986년 중앙대학교 대학원 전자계산학과 졸업(석사)
- 1990년 중앙대학교 대학원 전자계산학과 졸업(박사)
- 1990.9.~1992.8 공주교육대학교 전임강사
- 1992년 ~ 현재 공주대학교 컴퓨터공학부 교수

<관심분야>
소프트웨어 개발 방법론, 온톨로지