제25회 한국정보처리학회 춘계학술발표대회 논문집 제13권 제1호 (2006. 5)

도메인 온톨로지 구축을 위한 UML 모델의 OWL 변환 연구

나홍석
한국디지털대학교 디지털정보학과
e-mail : hsna99@kdu.edu

A Research for Converting UML model to Domain Ontology with OWL

Hong-Seok Na
Dept. of Computer and Information Science, Korea Digital University

요 약

독립적으로 구축되고 운영되어온 정보 자원의 공유와 통합된 서비스를 위해서 공통 온톨로지를 구축하고 활용하는 방법에 대한 연구가 활발히 진행되고 있지만, 온톨로지의 구축에 소요되는 많은 시간과 높은 비용은 온톨로지를 이용한 정보 통합의 있어 큰 장애가 된다.

본 논문에서는 UML 로 작성된 시스템 모델로부터 온톨로지 구축에 필요한 지식을 유도하여 초기 단계 도메인 온톨로지를 구축하는 방법을 제시하였다. 특히, UML 모델을 온톨로지 언어인 OWL 로 변환할 수 있는 구체적인 방향을 제시함으로써, 향후 자동화된 도구 개발의 이론적 기반을 될 것으로 기대한다.

1. 서론

온톨로지는 "개념화에 대한 외부적인 명세"[1]로 정의되며, 독립적으로 구축되어 운영되고 되는 정보 자원에 컴퓨터가 이해할 수 있는 명시적인 언어로 그 의미를 서술해 줌으로써, 정보의 통합과 공유를 가능하게 한다.

온톨로지를 사용한 정보자원의 통합은 참여하는 시스템에 대한 도메인 온톨로지를 구성하고 최소한의 공통 표준(상위 온톨로지)를 사용하여 각각의 도메인 온톨로지를 연결하는 형태로 진행된다[2].

하지만, 온톨로지를 구축하는 일은 많은 시간과 노력이 드는 매우 어려운 작업이며, 여러 다양한 분야의 전문가들의 참여를 필요로 하기 때문에[3], 효과적인 온톨로지 구축 방법에 대한 연구가 필요하다.

본 논문에서는 효과적인 도메인 온톨로지 구축을 위한 하나의 방법으로 시스템을 구축할 때 생성했던 설계모델(UML 모델)로부터 도메인 온톨로지를 유도할 수 있는 방법을 제안한다.

시스템 설계의 표준으로 자리잡고 있는 UML 모델은 그 자체에 시스템에서 필요로 하는 핵심 도메인 지식이 포함되어 있으며, 모델 개발과정에서 많은 전문가의 검증을 거치기 때문에, 도메인 온톨로지 구축을 위한 좋은 기반 자료가 된다.

또한, 서비스 지향 성격을 가지고 있는 온톨로지의 구축이 실제 서비스 시스템으로부터 출발함으로써 구축된 도메인 온톨로지와 응용 시스템 사이의 의미적 완성도를 높일 수 있는 장점이 있다.

구체적으로, 본 논문에서는 소프트웨어 설계 언어인 UML 과 온톨로지 언어의 표준으로 자리잡고 있는 OWL 을 구성하고 있는 모델 요소들을 비교하고 이들 사이의 변환 방법을 제시함으로써, 향후 변환 도구 구현을 위한 이론적 기반을 제공하는 것으로 목표로 한다.

제안된 방법 및 도구는 구축에 대한 초기부담을 줄여주면서도 도메인에 대한 정확한 지식을 표현하고 있는 온톨로지 구축을 위한 이론적 기반이 될 것이다.

2. 관련연구

UML, ER 등은 목표 시스템을 설계하기 위한 대표적인 모델링 언어(메타모델)이며, 대부분의 중규모 이

상 시스템들은 필수적으로 문제 도메인에 대한 모델링 과정을 거친 후에 구현 과정에 들어가게 된다. 즉, UML, ER 등의 모델은 완벽하지는 않더라도 현재 운영되는 시스템의 도메인에 대한 기본적인 구조에 대한 정보를 가지고 있다. 이 정보는 도메인 온톨로지 구축을 위한 좋은 출발점이 되며, 기 구축된 시스템의 모델로부터 온톨로지를 구축을 위한 기반 정보를 추출하려는 연구들이 진행되고 있다.

ERONTO[4] 시스템은 ER 모델을 OWL 형식의 온톨로지로 변환한다. 기본적으로 ER 모델의 개체와 속성 그리고 관계를 OWL 의 클래스와 프로퍼티로 매핑하며, 관계성의 차수와 같은 제약조건을 OWL 로 표현하는 방법을 제시하고 변환 도구 프로토타입을 개발하였다.

Robert M. Comb[5,6]는 DL(Description Logic)을 중심으로 UML, ER 그리고 OWL 사이의 개념적인 매핑을 위한 기본구조를 제시한다. 이를 위해서 <표 1>에서와 같이 메타 모델의 매핑 요소를 비교하고 이들 모델을 개념적인 수준에서 QVT(Query, View, Transformation)을 사용해서 직접 변환하는 함수 프로토타입을 정의한다.

하지만, DL 을 중심으로 UML, ER, OWL 사이의 변환이 이루어지기 때문에, 변환 과정에서 의미 정보의 손실이 발생한다. 예를 들어, UML 의 연관(Association)과 속성(Attribute)은 의미적으로 다른 요소이지만 무조건 DL 의 롤(Role)로 대응되며, OWL 의 프로퍼티(Property)로 변환된다.

<표 1> 메타모델 매핑 요소 비교

메타모델 유형	UML	ER	OWL	DL
객체	Class	Entity	Class	Concept
프로퍼티	Association	Relationship	Property	Role
	Attribute	Attribute		

Java2OWL[7]은 자바 클래스와 인스턴스를 카디널러티(Cardinality) 제약조건과 XML 스키마 데이터 타입을 이용하여 OWL 로 변환하는 프로그램으로 UML 모델의 OWL 변환에 대한 접근 방향을 제시한다. Falkovych[8]은 UML 클래스 모델을 DAML+OIL 로 변환하는 방법을 제시하고 있으며, 특히, UML 의 연관 관계를 세분화하여 연관의 정확한 의미를 온톨로지로 표현하는 방법을 제시하고 있다.

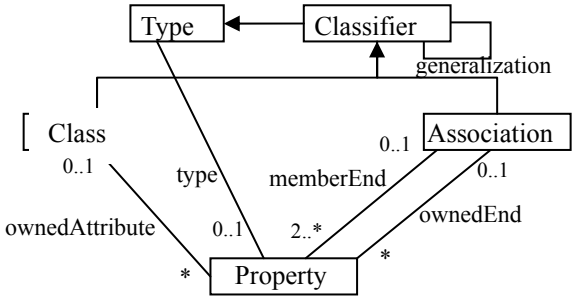
기 구축된 시스템 모델이 가지고 있는 도메인 지식을 완전하게 변환하기 위해서는 클래스-프로퍼티 수준의 개념적 매핑 보다는, UML 모델의 각 구성요소에 대한 구체적인 변환 방법이 요구된다.

3. UML 모델의 OWL 사상

3.1 UML 모델 기본 요소

<그림 1>에서 보듯이 UML 모델을 구성하는 모델 요소는 클래스와 클래스에 속해있는 프로퍼티, 그리고, 클래스 사이의 관계를 표현하는 연관(Association)과 같은 기본 요소와 함께 일반화, 타입의 개념을 표현한다. 또한, 그림에는 표현하지 않고 있지만, 같은 요소라도 제약조건이나 문맥에 따라 여러 가지 의미로 사용될 수 있으므로, 세부적인 적용 용도에 따라 OWL 의 대응 요소가 달라져야 한다.

UML 메타모델을 기반으로 클래스, 프로퍼티, 연관, 서브클래스, 일반화, 종속 등 UML 모델을 구성하는 요소를 유도할 수 있으며, <표 2>의 왼쪽 열에 나열해 놓았다.

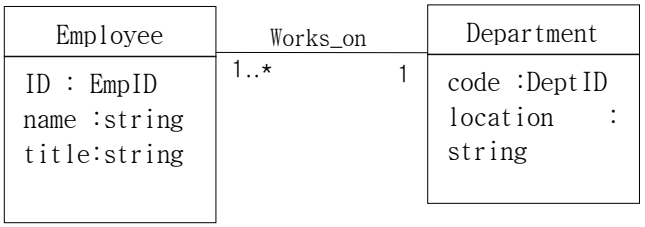


<그림 1> UML 메타모델의 일부

3.2 UML 모델 요소의 OWL 대응

<그림 2>는 회사도메인에서 부서(Department)와 직원(Employee)클래스 그리고 이들 사이의 관계를 표현하고 있는 UML 클래스 다이어그램의 일부이다.

클래스 Employee 의 프로퍼티(ownedAttribute)는 ID, name, title 이고, 클래스 Department 의 프로퍼티는 code 와 location 이다. 또한, 연관 Works_on 은 다중성 제약조건을 갖는다.



<그림 2> 부서-직원 관계에 대한 클래스 모델

- 클래스(Class)와 인스턴스(Instance)

클래스는 인스턴스의 집합으로 정의되며, UML 과 OWL 은 클래스를 사용하여 도메인의 개념(Concept)을 표현한다. 따라서, UML 클래스는 owl:Class 로 대응되며, 다음과 같이 표현할 수 있다.

```
<owl:Class rdf:ID="Employee"/>
<owl:Class rdf:ID="Department"/>
```

UML 클래스의 인스턴스는 OWL 의 개체(Individual)로 대응할 수 있는데, OWL 은 모든 클래스가 "Thing"이라는 최상위 클래스에서 상속되도록 규정하고 있으므로, 식별자가 "EMP0001"인 클래스 인스턴스는 아래와 같이 표현할 수 있다.

```
<owl:Thing rdf:ID="EMP0001" />
    <owl:Thing rdf:about="#EMP0001">
```

```
    <rdf:type rdf:resource="#Employee"/>
</owl:Thing>
```

여기서, rdf:type 은 RDF 에서 정의하는 프로퍼티로 지정한 OWL 개체가 클래스의 멤버임을 표현한다.

- 프로퍼티(ownedAttribute, Property)

UML 클래스 다이어그램의 프로퍼티(속성, owned Attribute)에 대응하는 OWL 프로퍼티는 ObjectProperty 와 DatatypeProperty 가 존재하는데, 프로퍼티 타입이 UML 클래스 일 경우에는 ObjectProperty 로 변환되고, 나머지 경우에는 DatatypeProperty 로 변환한다.

하지만, UML 에서 클래스를 타입으로 가지는 프로퍼티는 연관(association)에서 다루게 되므로, 여기서는 DatatypeProperty 로 변환하는 경우만 고려한다. 이때, 프로퍼티의 도메인(domain)과 레인지(range)속성을 이용하여, 프로퍼티가 속한 클래스와 프로퍼티의 타입을 표현한다.

예를 들어, <그림 2>의 Employee 클래스의 ID 속성과 code 속성은 owl:ObjectProperty 로 변환되며(ID 타입에 대한 클래스 정의가 있다고 가정), name, title, string 속성은 owl:DatatypeProperty 로 변환된다.

```
<owl:DatatypeProperty rdf:ID="name">
  <rdfs:domain rdf:resource="#Employee"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

또한, OWL 에서는 프로퍼티가 가지는 특성에 따라, Transitive, Symmetric, Functional, inverseOf, InverseFunctional 프로퍼티 타입을 추가하는 것도 고려해 볼 수 있다.

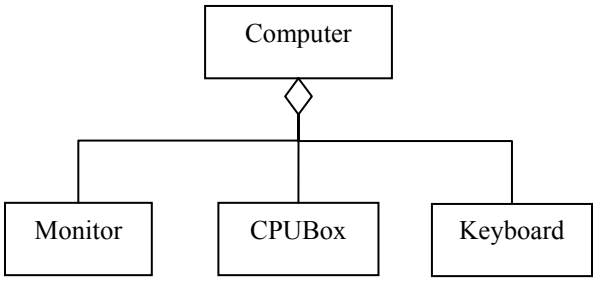
- 연관(Association)

UML 의 연관은 클래스와 클래스 사이의 관계를 표현하는 것으로 owl:ObjectProperty 로 대응하며, 도메인과 레인지 속성을 연관에 참여하는 클래스를 나타낸다. 또한, UML 연관은 기호의 형태(association end)에 따라, 바이너리(binary, 양방향) 연관, 단방향(unidirectional) 연관, 집합(Aggregation) 연관, 합성(Composition) 연관의 하위 타입 연관이 존재하므로 각각의 경우를 고려해 주어야 한다.

바이너리 연관의 경우에는 양방향의 추론이 가능해야 하므로, 두 개의 프로퍼티를 정의하고, 이 두 개의 프로퍼티들이 역(Inverse) 관계가 있음을 표현한다. 물론, 프로퍼티의 이름은 미리 정의된 접두어 등을 붙여서 구별해 준다.

```
<owl:ObjectProperty rdf:ID="Works_on">
  <rdfs:domain rdf:resource="#Employee"/>
  <rdfs:range rdf:resource="#Department"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="i_Works_on">
  <owl:inverseOf rdf:resource="#hasMaker" />
</owl:ObjectProperty>
```

UML 의 집합연관과 합성연관 모두 전체-부분관계를 표현한다. OWL 에서는 이러한 관계를 표현하는 기능을 지원하지 않기 때문에, 전체-부분 관계를 표현하는 키워드(is_part_of, is_whole_of)를 정의하고 owl:unionOf 를 이용하여 합집합을 표현해야 한다.



<그림 3> UML 의 집합연관 표현

전체-부분 관계에 참여하는 두 클래스 사이의 연관 바이너리 연관을 표현하는 OWL 프로퍼티로 변환하고, 전체-부분 관계를 추가한다. 물론, 이러한 추가적인 정의는 OWL 의 추론 엔진에서 별도로 처리되어야 한다. <그림 3>의 컴퓨터와 모니터 사이의 연관을 표현하면 다음과 같다.

```
<owl:ObjectProperty rdf:ID="is_part_of_P1">
  <rdfs:domain rdf:resource="#Monitor"/>
  <rdfs:range rdf:resource="#Computer "/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="is_whole_of_P1">
  <rdfs:domain rdf:resource="#Computer"/>
  <rdfs:range rdf:resource="#Monitor "/>
</owl:ObjectProperty>
```

- 일반화(Generalization)

UML 과 OWL 모두 상속(Inheritance) 개념을 지원한다. UML 에서 상속의 대상이 되는 것은 클래스와 연관이며(속성의 상속은 지원하지 않음), OWL 에서는 클래스에 대한 상속과 프로퍼티에 대한 상속을 구분해서 지원한다. 따라서, UML 클래스 상속은 아래 예와 같이 OWL 의 rdfs:subClassOf 로 대응하며, 연관에 대한 상속은 rdfs:subPropertyOf 로 대응된다.

```
<owl:Class rdf:ID="WomenEmployee">
  <rdfs:subClassOf rdf:resource="#Employee" />
  ...
</owl:Class>
<owl:ObjectProperty rdf:ID="Works_daily_on">
  <rdfs:subPropertyOf rdf:resource="#Works_on" />
  ...
</owl:ObjectProperty>
```

- 제약조건(Constraint)

카디널러티(Cardinality)등 UML 에서 사용되는 제약조건들은 OWL 클래스 정의 부분에서 owl:Restriction 을 사용하여 표현한다. 이때, 제약조건이 프로퍼티와 관련이 있는 경우, 해당 프로퍼티를 명시하고 제약조건을 기술한다. 아래의 예는 직원은 하나의 부서에 속

해있다는 카디널러티 조건을 표시한다.

```
<owl:Class rdf:ID="Employee">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#Works_on "/>
      <owl:minCardinality rdf:datatype="&xsd;nonnega
        tiveeeInteger " > 1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

또한, 프로퍼티의 도메인 카디널러티가 1 인 경우, owl:FunctionalPropery 타입으로, 레인지 카디널러티가 1 인 경우에는 owl:InverseFuncionalProperty 로 별도의 카디널러티 명세 없이 사용할 수 있다.

4. 비교 및 논의

<표 2>에서 UML 과 OWL 의 요소들 사이의 변환 관계를 비교 정리해 놓았다. UML 의 핵심적인 요소들은 대부분 OWL 로 변환이 가능하지만, 서로 호환하기 어려운 요소들도 존재한다.

UML 클래스 모델에서 표현하는 중요한 요소들의 대부분은 온톨로지 표준 언어인 OWL 로 변환이 가능함을 알 수 있다. 다만, 완성도 높은 온톨로지 구축을 위해서는 클래스와 같은 온톨로지의 개념 요소를 추출하는 작업과 함께 개념에 속하는 온톨로지 개체(Individual)가 정의되어야 하는데, UML 모델에서 인스턴스는 모델 외적으로 존재하는 경우가 많기 때문에, 실제 구현되어 운영되는 시스템의 데이터를 분석하여 추출하는 연구가 병행되어야 한다.

<표 2> UML 과 OWL 의 대응요소 비교

UML	OWL
클래스	클래스(owl:Class)
연관 클래스	클래스(owl:Class)
나열형 클래스 (enumeration)	나열형 클래스(owl:oneOf)
인스턴스	개체(Individual)
속성	프로퍼티(owl:DatatypeProperty)
연관(바이너리)	프로퍼티의 쌍(inverseOf 관계)
연관(단방향)	프로퍼티
연관(집합,합성)	대응요소 없음(필요시, is_part_of, is_whole_of 키워드 사용)
일반화(클래스)	서브클래스(rdfs:subClassOf)
일반화(연관)	서브프로퍼티(rdfs:subPropertyOf)
서브클래스집합	합집합(owl:unionOf)
다중상속	다중상속(rdfs:subClassOf)
다중성	카디널러티(owl:cardinality, mincardninality, maxcardinality,) 프로퍼티(owl:FunctionalProperty, owl:InverseFunctionalProperty)
패키지	온톨로지(owl:Ontology)
항해가능성 (navigable)	도메인(rdfs:domain) 레인지(rdfs:range)

5. 결론

본 논문에서는 도메인 온톨로지 구축을 위하여 기 구축된 시스템 모델로부터 온톨로지 구축에 필요한 지식을 추출하여 초기 온톨로지 모델을 작성하는 방법을 제시하였다.

어플리케이션 설계 언어인 UML 의 클래스 모델과 온톨로지 언어인 OWL 의 주요 구성요소 사이의 대응 과정을 분석하고 비교하였으며, 구체적인 변환 방법을 제시하였다.

UML 클래스 모델 자체가 도메인에 대한 지식과 개념을 어느 정도 표현하고 있으므로, 대부분의 핵심 요소들의 OWL 로 변환이 가능함을 알 수 있으며, 변환된 OWL 문서는 도메인에 대한 초기 온톨로지 모델로 온톨로직 구축을 위한 중요한 역할을 할 것으로 기대된다.

하지만, 도메인 온톨로지 변환에 대한 완성도를 높이기 위해서는 클래스 모델 뿐만 아니라, 시스템의 동적인 측면을 다루는 시퀀스, 액티비티 다이어그램의 메시지와 매개변수 등으로부터 클래스와 객체의 책임(responsibility)를 유도하는 방법에 대한 연구가 필요하며, 인스턴스 레벨까지 온톨로지를 확장하기 위해서는 운영되는 인스턴스로부터 온톨로지 개체를 유도하는 방법에 관한 연구가 필요하다. 추후, 논문의 이론을 기반으로 UML 모델을 온톨로지로 변환하는 자동화된 도구를 개발할 예정이다.

참고문헌

[1] Tom Grumber, “A translation approach to protable ontology specifications”, Knowledge Acquisition, Vol. 5, No. 2, pp.199-220, 1993.

[2] Michael Uschold, Machael Gruninger, “Ontologies and Semantics for Seamless Connectivity”, SIGMOD Record, Vol.33, No.4, pp.58-64, December 2004.

[3] H.Wache, T.Vögele, U.Visser, H.Stuckenschmidt, G.Schuster, H.Neumann, S.Hübner, “Ontology-Based Integration of Information – A Survey of Existing Approaches”, in The Seventeenth International Joint Conference on Artificial Intelligence, USA, 2001

[4] Sujatha R Upadhyaya, P Sreenivasa Kumar, “ERONTO: A Tool for Extracting Ontologies from Extended E/R Diagrams”, SAC05, pp.666-670, March 2005.

[5] Robert M. Colomb, Anna Gerber, Michael Lawley, “Issues in Mapping Metamodels in the Ontology Development Metamodel Using QVT”, in The 1st International Workshop on the Model-Driven Semantic Web, 2004.

[6] 홍현술, Robert M. Colomb, “시멘틱 웹을 위한 이기종 시스템 간의 온톨로지 매핑”, 한국정보과학회 논문지:기술교육, 제 1 권 제 1 호, 2004. 12

[7] Dean, M. Java2OWL, DAML.org, 2003.

[8] K. Falkovych, M. Sabou and H. Stuckenschmidt, “UML for the Semantic Web:Transformation-Based Approaches”, In Knowledge Transformation for the Semantic Web, B. Omelayenko and M. Klein editors. IOS Press, 2003.