제22회 한국정보처리학회 추계학술발표대회 논문집 제11권 제2호 (2004. 11)

소프트웨어 공학적 방법을 이용한 온톨로지의 효율적인 설계 및 생성에 관한 연구

이윤수 김태석 양진혁 정인정
고려대학교 전산학과
e-mail: {arzhna, cstkts, grjinh, chung}@korea.ac.kr

A Research on Efficient Design and Creation of Ontology Using Software Engineering Method

Yun-Su Lee Tae-Suk Kim Jin-Hyuk Yang In-Jeong Chung
Dept. of Computer Science, Korea University

요 약

현재 웹 데이터는 폭발적으로 증가하고 있지만 이를 통해 유용한 정보를 추출한 지식 표현 위주의 웹 데이터는 부족한 실정이다. 이러한 문제점을 해결하기 위해 제안된 시맨틱 웹은 온톨로지를 기반구조로 하고 있다. 그러나 현재 온톨로지의 생성에 관한 기존 연구들은 대부분 도메인 전문가들의 휴리스틱에 의존하는 수작업 형태를 띠고 있다. 이러한 방법은 많은 시간과 노력이 소요될 뿐만 아니라 뜻하지 않은 오류가 나타날 수 있다.

이와 같은 문제를 해결하기 위해 우리는 소프트웨어 공학적인 접근방법을 통하여 온톨로지를 효과적으로 설계 및 생성하는 방법을 제안한다. 우리는 기존의 UML과 OWL을 단순 매핑하는 방법에 MDA 접근법의 장점들을 취하여 통합, 확장하였다. 즉, MDA를 기반으로 UML을 이용하여 온톨로지를 설계한 후 설계된 온톨로지 모델을 XMI를 통해 온톨로지 기술 언어인 OWL로 변환하는 과정을 거쳐 온톨로지를 생성한다. 끝으로 구체적인 보기를 통해 본 논문에서 제안한 방법의 타당성을 보인다.

1. 서론

현재 웹 데이터는 폭발적으로 증가하고 있지만 이를 통해 유용한 정보를 추출한 지식 표현 위주의 웹 데이터는 부족한 실정이다. 이러한 문제점을 해결하기 위해 시맨틱 웹[16]이 Tim Berners-Lee에 의해 제안되었다. 시맨틱 웹은 사람과 기계에 의해 데이터의 의미를 처리할 수 있고, 도메인 내의 개념들과 개념들 간의 관계를 정형적으로 기술하고 있는 온톨로지[15]를 기반구조로 가진다. 온톨로지를 기반으로 우리는 기존 표현 위주의 웹 언어들이 가지는 기계에 의해 인식될 수 없고, 공유 및 재사용이 어렵다는 등의 단점들을 보완할 수 있다.

그러나 시맨틱 웹의 기반구조인 온톨로지의 생성에 관한 기존 관련 연구들의 상당 부분은 도메인 전문가들의 휴리스틱에 의존하는 수작업 형태를 띠고 있다. 이러한 연구들은 온톨로지의 생성에 많은 시간과 노력이 소요될 뿐만 아니라 뜻하지 않은 오류가 나타날 수 있기 때문에 시맨틱 웹의 발전에 저해요소가 되고 있다[9].

이러한 문제를 해결하기 위한 기존의 연구들은 UML과 온톨로지 기술 언어를 단순히 매핑하는 것에 불과했다. 그러나 [12]는 MDA[1]를 기반으로 ODM(Ontology Definition Model)이라는 메타모델을 통해 OWL로의 변환을 제안하고 있다. 우리는 [12]에서 기반으로 삼고 있는 MDA의 장점 및 잠재성을 취하여 기존의 방법과 통합, 확장하는 방법을 제안한다. 우리가 제안하는 방법은 MDA 접근법을 기반으로 UML[2]을 이용하여 온톨로지를 설계하고 XMI[4]를 통해 온톨로지 기술 언어인 OWL[3]로 변환하는 과정을 거쳐 온톨로지를 생성하는 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 우리가 제안

하는 접근법에서 사용하는 소프트웨어 공학의 개념들에 대해 간단히 언급한다. 3장에서는 우리의 접근법을 자세히 기술하며 4장에서는 실제로 온톨로지 모델을 설계하고 온톨로지를 생성하는 구체적인 예제를 보인다. 그리고 5장에서 결론 및 향후과제에 대해 언급한다.

2. 기반 개념

2.1 MDA

MDA(Model Driven Architecture)는 UML을 통해서 시스템을 모델의 형태로 설계 및 명세하기 위한 새로운 소프트웨어 공학의 패러다임이다[1]. 이 접근방법은 모델링 언어 및 구현 언어들의 메타모델간의 매핑을 통하여 상호 변환이 가능하도록 지원함으로써 거대한 소프트웨어 애플리케이션을 효율적으로 개발하기위해 사용된다. 즉, 플랫폼에 독립적인 모델링 언어로 모델을 설계하고 이를 특정 플랫폼에 기반한 구현에 가까운 모델로 변환하여 보다 쉽게 애플리케이션을 개발 할 수 있는 능력을 제공한다.

온톨로지를 UML로 모델링하고 이를 온톨로지 기술 언어로 변환하는 방법을 사용할 때 MDA는 메타모델 레벨에서 매핑이 되기 때문에 온톨로지 기술언어가 진화 또는 변화하더라도 메타모델간의 매핑관계는 약간의 수정만으로 거의 그대로 사용할 수 있는 장점을 가진다. 따라서 기술 변화 상황 또는 시스템 인프라 변화에 효율적으로 대처할 수 있다[7].

2.2 UML

UML(Unified Model Language)은 소프트웨어 설계에 있어서 강력한 표현력을 제공하는 모델링 언어이다. UML은 직관적이고 표현력이 강한 시각적 모델링 방법을 제공함으로써 정확하고 신뢰성 있는 모델들을 개발하고 상호 운용 할 수 있게 한다[2].

UML을 이용한 온톨로지 설계는 객체지향적 설계방법으로 쉽게 재사용가능하고 상호운용이 가능하다. 이는 온톨로지의 특징들과 부합된다.

2.3 XMI

XMI(XML-based Metadata Interchange)는 UML로 기술된 모델정보의 XML 표현을 위한 OMG의 표준이다.[4] 즉, UML로 설계된 모델을 다른 포맷의 모델로 변환하고 교환하기 위해 사용되는 표준이다. 우리는 UML로 설계된 모델을 OWL 온톨로지로 변환하기 위해 XMI를 사용하였다.

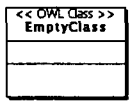
2.4 OWL

OWL(Web Ontology Language)은 온톨로지 기술을 위한 표준 마크업 언어로써 단지 사람에게 정보를 표시하는데 그치지 않고 정보의 내용을 직접 처리할 수 있는 어플리케이션을 구현하는데 활용될 수 있도록 설계되었다. OWL은 풍부한 어휘(vocabulary)와 형식적 의미론(formal semantics)을 포함하고 있기 때문에 기계 해석이 가능한 웹 컨텐트를 저작하는데 있어 XML, RDF 및 RDF-S보다 뛰어나다. OWL은 표현력이 서로 다른 세 개의 하위 언어(OWL Lite, OWL DL, OWL Full)로 구성되어 있다. 후자로 갈수록 표현력이 더 크다[3].

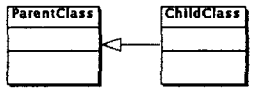
3. 온톨로지의 설계 및 생성 방안

3.1 매핑 관계의 정립

MDA 접근방법은 메타모델간의 매핑을 기반으로 이루어지기 때문에 먼저 UML과 OWL의 메타모델 간의 매핑 관계를 정립할 필요가 있다. 두 언어 사이의 매핑에 있어서 Package 또는 Class와 같은 정적인 구조물들과 상속관계와 같은 연관관계를 표현하는 일부 구조물들은 (그림 1), (그림 2)와 같이 쉽게 매핑 시킬 수 있다[8].



(그림 1) UML:Class ➡ owl:Class



(그림 2) UML:Generalization ➡ owl:subClassOf

그러나 대부분의 관계를 표현하는 동적인 구조물들이나 속성들을 표현하는 구조물들은 UML로 쉽게 표현할 수 없다[8]. 따라서 이러한 구조물들의 매핑관계를 정립하기 위해 UML로 표현하기 어려운 각 구조물들을 스테레오타입으로 정의해야한다.

3.2 온톨로지 모델의 설계 및 변환

UML의 기본 구조물과 앞에서 정의된 스테레오타입들을 이용하여 온톨로지 모델을 설계한다. 온톨로지는 프레임구조에 기반을 두고 있기 때문에 객체지향 패러다임을 지원하는 UML로 설계하기가 용이하다.

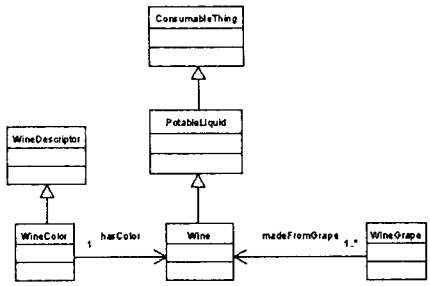
UML을 이용하여 설계된 온톨로지 모델을 XMI 문서로 변환한다. XMI 형태로 변환된 온톨로지 모델은 XML 형태이므로 XML 파서 또는 XSLT 등을 거쳐 OWL 온톨로지로 쉽게 변환 될 수 있다.

4. 실험 결과

우리는 우리가 제안한 온톨로지 설계 및 생성 방안의 타당성을 검증하기 위해 실제로 간단한 온톨로지를 설계하고 생성하는 실험을 수행했다.

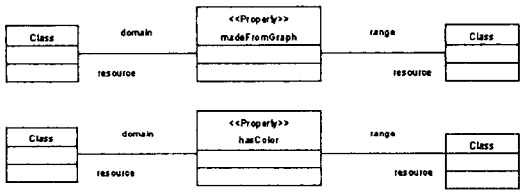
4.1 온톨로지 모델 설계

우리는 UML을 이용하여 (그림 3)과 같이 온톨로지 모델을 설계하였다. 이 온톨로지는 [13]에서 정의하고 있는 Wine에 대한 온톨로지를 기반으로 설계한 것이다.



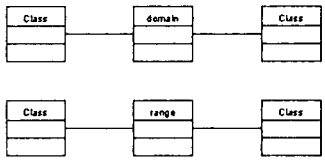
(그림 3) UML 클래스 다이어그램으로 설계된 Wine 온톨로지

그러나 UML 만으로 설계된 온톨로지 모델은 사용자 정의에 의한 관계, 즉 hasColor와 madeFromGrape와 같은 연관관계를 제대로 표현할 수 없다. 따라서 우리는 이러한 연관관계의 표현을 위해 (그림 4)와 같이 스테레오 타입으로 정의하였다.



(그림 4) 연관관계를 정의한 메타 데이터

UML에서는 연관관계를 연결선으로 나타내지만 온톨로지에서는 연관관계 역시 클래스의 개념을 가지므로 (그림 4)와 같이 클래스 다이어그램으로 정의할 수 있다. 각 연관관계의 속성인 domain, range 또한 같은 방법으로 (그림 5)와 같이 정의할 수 있다.



(그림 5) domain, range를 정의한 메타 데이터

이와 같은 방법으로 UML의 기본 구조물들로 표현하기 어려운 OWL의 구조물들을 UML의 스테레오 타입으로 정의하여 온톨로지 모델을 설계할 수 있다.

4.2 설계된 온톨로지 모델의 변환

UML로 설계된 온톨로지 모델을 OWL 온톨로지로 변환하기 위해서는 UML 모델을 XML 형태로 나타낼 필요가 있다. 이를 위해 우리는 XMI를 사용하였다. UML 모델의 설계를 지원하는 많은 도구들은 자체적으로 UML 모델을 XMI 문서로 변환하는 기능을 제공하고 있으며 우리는 그 중 ArgoUML[6]이라는 도구를 사용하였다. <표 1>은 ArgoUML을 이용하여 설계된 온톨로지 모델을 변환한 XMI 문서의 일부이다.

<표 1> XMI로 변환된 온톨로지 모델

```
                    ……
<Foundation.Core.Class xmi.id="xmi.6"
        xmi.uuid="-93--104--101--89-afa68a:fe94e962f8:-7ffc">
    <Foundation.Core.ModelElement.name>
            Wine
    </Foundation.Core.ModelElement.name>
    <Foundation.Core.ModelElement.visibility
            xmi.value="public"/>
    <Foundation.Core.ModelElement.isSpecification
 xmi.value="false"/>
    <Foundation.Core.GeneralizableElement.isRoot
            xmi.value="false"/>
    <Foundation.Core.GeneralizableElement.isLeaf
            xmi.value="false"/>
    <Foundation.Core.GeneralizableElement.isAbstract
            xmi.value="false"/>
    <Foundation.Core.Class.isActive xmi.value="false"/>
    <Foundation.Core.ModelElement.namespace>
        <Foundation.Core.Namespace xmi.idref="xmi.1"/>
    </Foundation.Core.ModelElement.namespace>
    <Foundation.Core.GeneralizableElement.generalization>
        <Foundation.Core.Generalization xmi.idref="xmi.5"/>
    </Foundation.Core.GeneralizableElement.generalization>
</Foundation.Core.Class>
                    ……
```

우리는 온톨로지 모델을 변환한 XMI문서를 OWL 온톨로지로 변화하기 위해 XML DOM(Document Object Model) Parser[5]를 사용하였다. DOM 파서는 기존의 여러 연구에서 사용했던 XSLT를 사용할 경우 발생할 수 있는 모호성문제[8]를 변수를 사용하여 해결 할 수 있다. 또한 XSLT보다 빠르고 유연하게 변환할 수 있는 장점을 지닌다. 우리는 DOM 파서를 이용하여 XMI 문서를 OWL 온톨로지로 변환하는 애플리케이션을 Java로 개발하여 사용하였다. <표 2>는 DOM 파서를 통해 변환된 OWL 온톨로지이다.

5. 결론 및 향후 연구과제

우리는 기존의 UML을 이용하여 온톨로지를 설계하고 생성하는 방법에 MDA를 이용한 방법의 장점들을 취합하여 통합, 확장하였다. 확장된 방안은 MDA를 기반으로 UML로 설계된 온톨로지 모델을 OWL 온톨로지로 변환하는 것이다. 우리는 UML로 표현하기 어려운 온톨로지의 구조물들을 스테레오타입으로 정의하였고 이를 바탕으로 온톨로지 모델을 설계하였다. 설계된 온톨로지 모델은 XMI 형태로 변환되며 이 XMI 문서는 DOM 파서를 거쳐 OWL 온톨로지로 변환된다. 이러한 일련의 과정을 통해 UML로 설계한 Wine 온톨로지 모델의 실제 온톨로지를 추출할 수 있다.

우리의 접근방법은 UML을 사용하여 쉽게 재사용가능하고 상호운용이 가능한 모델 레벨의 온톨로지를 설계할

수 있으며 이를 통해 실제 온톨로지를 생성할 수 있다. 또한 MDA가 가지는 장점을 취할 수 있다. 즉, 기술 변화 상황 또는 시스템 인프라 변화에 효율적으로 대처할 수 있고 적은 시간과 비용으로 높은 신뢰도를 가지는 온톨로지를 만들 수 있다.

<표 2> 생성된 Wine 온톨로지

```
<owl:Class rdf:ID="ConsumableThing"></owl:Class>
<owl:Class rdf:ID="PotableLiquid">
    <rdfs:subClassOf rdf:resource="#ConsumableThing">
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Wine">
    <rdfs:subClassOf rdf:resource="#PotableLiquid">
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasColor">
                <owl:Cardinality>1</owl:Cardinality>
            </owl:onProperty>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#madeFromGrape">
                <owl:minCardinality>1</owl:minCardinality>
            </owl:onProperty>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="WineColor">
    <rdfs:subClassOf rdf:resource="#WineDescriptor">
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="WineDescriptor"></owl:Class>
<owl:Class rdf:ID="WineGrape"></owl:Class>
<owl:ObjectProperty rdf:ID="hasColor">
    <rdfs:range rdf:resource="#WineColor">
    </rdfs:subClassOf>
    <rdfs:domain rdf:resource="#Wine">
    </rdfs:subClassOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="madeFromGrape">
    <rdfs:domain rdf:resource="#Wine">
    </rdfs:subClassOf>
    <rdfs:range rdf:resource="#WineGrape">
    </rdfs:subClassOf>
</owl:ObjectProperty>
```

향후 연구과제로 우리는 본 논문에서 제안한 방법을 OWL-S, 즉 서비스 온톨로지를 설계하고 생성하는 방안으로 확장시킬 것이다. 우리가 제안한 방법을 서비스 온톨로지 설계 및 생성에 적용하면 현재 웹 서비스를 구축하는데 있어서 서비스와 서비스 온톨로지를 따로 설계하는 비효율성을 해결할 수 있다. UML로 하나의 서비스 모델을 설계하고 이를 서비스와 서비스 온톨로지로 동시에 변환할 수 있는 가능성이 있다. 따라서 우리가 제안한 방법을 사용한다면 보다 효율적으로 웹 서비스를 구축 할 수 있다. 나아가 OCL(Object Constraint Language)[14]을 사용하여 서비스 모델에 규칙을 부여함으로써 시맨틱 웹 서비스까지 확장시킬 수 있는 가능성이 충분하다.

참조 문헌

[1] MDA : http://www.omg.org/mda/

[2] UML : http://www.uml.org/

[3] OWL : http://w3.org/2004/OWL/

[4] XMI : http://www.omg.org/technology /documents/formal/xmi.htm

[5] DOM : http://www.w3.org/DOM/

[6] ArgoUML : http://argouml.tigris.org/

[7] Jean Bezivin, Slimane Hammoudi, Denivaldo Lopes and Frederic Jouault, An Experiment in Mapping Web Services to Implementation Platforms, ICCS 2004: 4th Int. Conf. pp. 164 - 173, June, 2004.

[8] Stefan Wendler, Mapping XMI / UML to DAML+OIL, http://www.jdev.de/html/projects /uml2daml/mapping/uml2daml_mapping.html, 2002

[9] Stephen Cranfield, Stefan Haustein and Martin Purvis, UML-Based Ontology Modeling for Software Agents, Proc. of Ontologies in Agent Systems Workshop, pp. 21-28, 2001

[10] Stephen Cranefield and Martin Purvis, UML as an ontology modelling language. In Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99), 1999.

[11] Jernnj Kovse and Theo Harder, Generic XMI-Based UML Model Transformations, in: Proc. 8th Int. Conf. on Object-Oriented Information Systems (OOIS'02), pp. 192-198, Sept. 2002,

[12] Dragan Duric, Dragan Gasevic and Vladan Devdzic, A MDA-based Approach to the Ontology Definition Metamodel, In Proc. of the 6th Int. Conf. on Information Technology, pp. 193-196, 2003,

[13] 오삼균, Web Ontology Language와 그 활용에 대한 고찰, 데이터베이스 연구회 18권 3호, pp. 64-79, 2002.

[14] OCL : http://www.uml.org/

[15] Asuncion Gomez-Perez, Oscar Corcho, Ontology languages for the Semantic Web, IEEE, Vol. 17, pp. 54-60, Jan-Feb 2002

[16] Stefan Decker, Sergey Melnik, Frank van Harmelen, Dieter Fensel, Michael Klein, Jeen Broekstra, Michael Erdmann, Ian Horrocks, The Semantic Web : the roles of XML and RDF, IEEE, Vol. 4, pp. 63-70, Sept-Oct, 2000