UML을 이용한 효율적인 온톨로지 재사용에 관한 연구

이지홍 양진혁 손종수 정 인정

고려대학교 전산학 과
충청남도 연기군 조치원읍 서창 동 208 고려대학교 과학기술대학 202호
Tel: +82-41-860-1342, E-mail:{ljh78, grjinh, mis026, chung}@korea.ac.kr

요 약

차세대 웹의 중심기술인 시맨틱 웹을 구현하기 위해서는 컴퓨터 가 지식을 추 론하고 처리할 수 있게 하기 위한 지식 표 현 방법인 온톨로지가 필 수적으로 요구된다. 이러한 온톨로지 생성의 중요성이 점차 커져가는 실정에 따라, 생성된 온톨로지들의 재사용을 위한 방법이 관련 연구들 사이에 중요한 과제로 떠오르기 시작 되었다

본 논문에서는 온톨로지의 재사용을 효율적으로 하기 위한 방법을 제안한다. 우리가 제안하는 방법은 온톨로지를 사용자가 이해하기 쉽고 편집이 용이한 그래프 형태로 표현하는 방법으로 온톨로지를 UML로 변환하여 UML을 통한 온톨로지 재사용 방안을 제안한다. 우리가 제안하는 방법은 다음과 같다. OMG의 MDA개념을 기반으로 기존에 생성된 온톨로지를 XML 파서를 이용하는 방법을 통하여 XMI로 변환한다. XMI로 변환된 온톨로지는 UML 도구를 사용하여 재사용 할 수 있다. UML로 변환된 온톨로지는 위 과정을 역으로 다시 수행함으로써 온톨로지로 변환된다.

이렇게 UML로 변환된 온톨로지는 UML의 장점을 그대로 가지게 된다. 이미 널리 사용되고 가독성과 편집력 그리고 상호 운용성이 높은 UML을 이용하여 온톨로지의 재사용성을 높이고자 하는데 있다. 즉 사용자가 직관적으로 온톨로지의 전체 구조와 의미를 파악하는데 도움을 주며 편집 또한 용이하다.

이러한 방법을 바탕으로 UML을 통해 온톨로지를 쉽게 사용 자의 온 톨로지에 대 한 이해와 수정을 도와 온톨로지의 재사용성을 높이고 사용자간의 공유를 용이하게 만들 수 있다 .

키워드:
시맨틱웹, 온톨로지, XMI, UML, MDA, OMG

서론

정보 통신 기술의 발달에 따라 많은 사용자들이 웹을 보다 편리하고 쉽게 접근할 수 있게 되었고, 이에 따라 정보의 습득과 공유가 증가하게 되었다. 그러나 방대해진 데이터 가운데 원하는 데이터의 취득은 어려워 지고 있는 것이 사실이다. 이러한 문제의 해결책으로 Tim Berners-Lee에 의해 시맨틱 웹이 주창 되 었다. 시맨틱 웹[1]은 기계가 처리할 수 있는 형태로 데이터를 표현하여 인간의 개입 없이 데이터의 이해와 처리가 가능한 웹을 말한다. 이는 인간의 개입을 최소화하여 기계에 의해 정보의 처리가 가능케 하는 것을 말한다. 이러한 시맨틱 웹을 구현하기 위해서는 컴퓨터가 지식을 추론하고 처리할 수 있게 데이터를 표현하는 온톨로지[2]가 필수적이다. 이러한 온톨로지의 중요성이 커져가는 실정에 따라, 생성된 온톨로지들을 재사용하기 위한 방법이 관 련 연구들 사이에 중요한 과제로 떠오르기 시작 되었다.

그러나 온톨로지 언어는 그 사용이 난해하며 사용자에게 직관적으로 구조 및 관계 등의 형태를 보여 주지 못함으로써 사용자가 수정 및 공유를 하는데 불편을 주고 있다[3]. 또한 온톨로지를 재사용하기 위한 도구들의 인터페이스가 직관적이지 못하며 그 기능 또한 제한적으로 제공 되어 실상 사용자의 손으로 대부분의 작업을 하 여 많은 시간과 비용을 소비하는 단점이 있다.

본 논문에서는 이러한 문제를 해결하기 위하여 UML[4]을 이용한 온톨로지의 효율적인 재사용 방안을 제안한다. 우리가 제안하는 방법은 기존에 많은 연구가 이루어진 UML을 이용한 온톨로지 생성에 관한 연구를 기반으로 XML 파서를 이용하여 생성된 온톨로지를 XMI[5]로 변환한다. 이렇게 변환된 XMI는 UML및 여타 OMG표준의 MDA(Model Driven Architecture) [6] 모델로 나타 내는

것이 가능하다. 본 논문에서는 그 중 가장 일반적으로 사용되는 UML을 이용한 온톨로지의 재사용 방안을 제시한다.

본 논문에서는 온톨로지와 XMI간의 변환을 위한 기반 기술에 대해 간단히 살펴본 후 기존 연구를 토대로 XMI를 통한 온톨로지와 UML간 양방향 변환 방법 및 그 매핑 관계에 대하여 살펴본다. 다음으로 정립된 매핑 관계에 의거하여 간단한 실험 예제를 보이고 마지막으로 결론 및 향후과제를 논한다.

기반 기술

MDA(Model Driven Architecture)는 설계 명세를 통해서 시스템을 UML모델의 형태로 설계 및 명세하여 개발하기 위한 방법이다. MDA의 기본개념은 MOF(Meta Object Facility)[7]라는 메타모델 정의 언어를 기반으로 UML을 메타모델 기술언어로 이용해 시스템의 설계 및 명세를 정의한 모델로 실제 구현과 관련된 모델을 모델링 언어 와 구현 언어들 사이의 메타모델간의 매핑을 통하여 변환하는 설계수준 표준이다. 정의된 매핑 관계를 통해 상호 변환이 가능하도록 지원함으로써 다양한 플랫폼과 구현언어에서 편리하게 소프트웨어 어플리케이션을 개발하기 위해 사용된다. 그림 1 즉, 플랫폼에 독립적인 모델링 언어로 모델을 설계하고 이를 특정 플랫폼에 기반한 모델로 혹은 언어로 변환하여 보다 쉽게 애플리케이션을 개발 할 수 있는 능력을 제공한다[8].

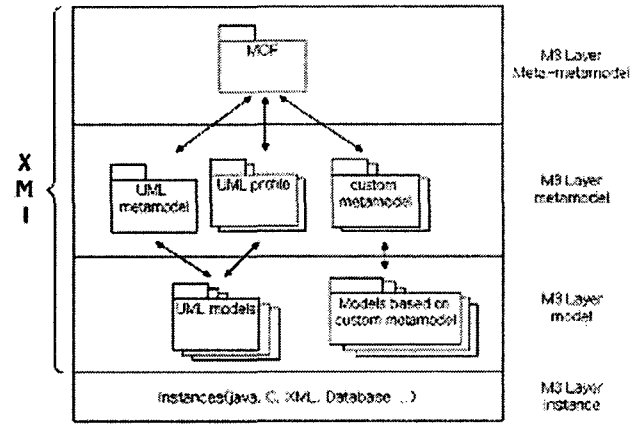

그림 1 - MDA 4단계 개념도

온톨로지를 UML로 변환하기 위해서는 XMI로 변환하는 과정을 거친다. 즉, 온톨로지를 UML기술 언어로 변환하고 이를 다시 온톨로지 기술 언어로 변환하는 방법을 사용할 때 XMI를 이용한 메타모델 레벨에서 매핑이 된다[9]. 따라서 추후에 온톨로지 기술언어가 변화하더라도 메타모델간의 매핑 관계에 약간의 수정만으로 거의 그대로 사용할 수 있는 장점을 가진다. 따라서 기술 변화 상황 또는 시스템 인프라 변화에 효율적으로 대처할 수 있다[10].

UML(Unified Model Language)은 소프트웨어 설계에 있어서 강한 표현력을 제공하는 모델링 언어이다. UML은 직관적이고 표현력이 강한 시각적 모델링 방법을 제공한다. 따라서 사용자에게 온톨로지 전체를 시각적으로 이해하기 쉽게 표현하여 효율적으로 모델들을 개발할 수 있게 하며 설계된 모델을 상호 운용 할 수 있게 한다[4]. UML을 이용한 온톨로지의 표현은 객체지향적 방법으로 쉽게 편집하며 이해할 수 있고 상호운용이 가능하다 이는 온톨로지의 재사용성을 높인다.

XMI(XML Metadata Interchange)는 MOF와 호환성 이 있는 모델들을 XML 문서 형태로 표현하기 위한 표준이다[5]. 즉, UML로 설계된 모델을 다른 포맷의 모델로 변환하고 교환하기 위해 사용되는 표준이다. 이를 통하여 사용자들 사이에 metadata에 관한 정보를 교환, 공유할 수 있다. 본 논문은 온톨로지와 UML간의 변환의 위해 XMI를 이용하였다.

본 논문에서는 여러 가지 온톨로지 기술 언어 가운데 OWL을 이용하여 UML을 이용한 온톨로지 재사용에 대해 논하겠다. OWL(Web Ontology Language)은 XML 기반의 온톨로지 기술을 위한 W3C 표준 마크업 언어로써 단지 사람에게 정보를 표현하는데 그치지 않고 기계가 정보의 내용을 직접 처리할 수 있는 어플리케이션을 구현하는데 활용될 수 있도록 설계되었다[11].

OWL은 풍부한 어휘와 형식적 의미론을 포함하고 있기 때문에 기계 해석이 가능한 웹 컨텐트를 저작하는데 있어 XML, RDF 및 RDF-S보다 뛰어 나며 필요에 따라 RDF-S를 채용하여 사용할 수 있다. OWL은 표현력에 따라 OWL Lite, OWL DL(Description Logic), OWL Full 3가지 형태로 구성되어 있다. 후자로 갈수록 표현력이 증가하며 이는 사용자의 표현력 요구에 따라 선택하여 사용이 가능하다[12].

온톨로지 변환 및 재사용

XMI를 매개로 온톨로지와 UML사이의 변환을 위해서는 MDA 접근 방식을 이용하여 메타모델간 매핑이 이루어져야 한다. 온톨로지와 UML 사이의 매핑에 있어서 Package 또는 Class와 같은 정적인 구조물들과 상속관계와 같은 연관관계를 표현하는 구조물들을 표 1과 같이 매핑 시킬 수 있다[9, 13].

표 1 - UML과 OWL 매핑 관계

UML	OWL
UML:Package	owl:Ontology
UML:Class	owl:Class
UML:Attribute	owl:DatatypeProperty
UML:Generalization	owl:subClassOf
UML:Association	owl:objectProperty, owl:onProperty, owl:cardinality, owl:restriction

다음 그림 2는 위의 매핑 관계를 토대로 하여 만든 온톨로지/UML 표현의 간단한 예 이다.

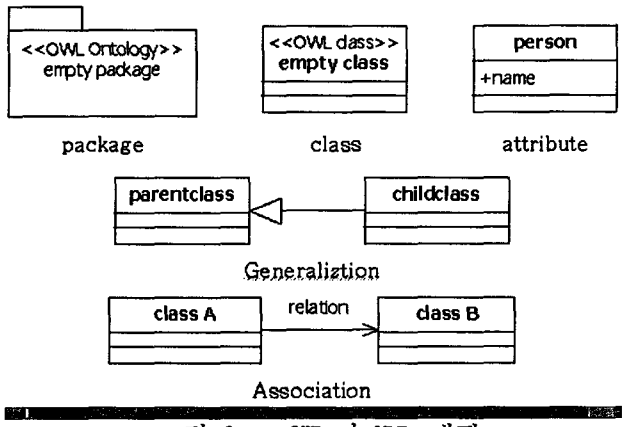

그림 2 - OWL과 UML 매핑

위 예제와 같이 대부분의 관계를 표현 할 수 있으나 위 예제에서 보이듯이 OWL모델임을 나타내거나 UML에서 나타낼 수 없는 개념 및 구조물 등 추가적인 표현은 스테레오 타입을 이용하여 정의 해야 한다[9].

앞서 정의한 UML개념과의 매핑 관계에 따라 온톨로지를 XMI 문서로 변환한다. 이때 온톨로지는 XML기반 언어 이므로 XML 파서를 거쳐 XMI로 쉽게 변환 될 수 있다. 변환된 XMI 문서는 UML도구에서 import하여 UML로 표현이 가능하다. 역으로 UML에서 XML로의 변환은 위 과정을 역으로 수행한다. 즉, UML도구를 사용하여 XMI로 모델을 export한 후 XML파서를 사용하여 온톨로지로 변환 한다. 기본적으로 온톨로지는 프레임구조에 기반을 두고 있기 때문에 객체지향 패러다임을 지원하는 UML로 잘 표현될 수 있다[14].

실험 예제

본 논문에서는 위에서 제안한 방법에 의거하여 기존의 온톨로지를 매핑 관계에 의거하여 UML로 변환 하여 나타낸 후 UML을 통한 온톨로지를 재사용 방법을 보인다. 이 프로세스의 기본적인 구조는 그림 3과 같다.

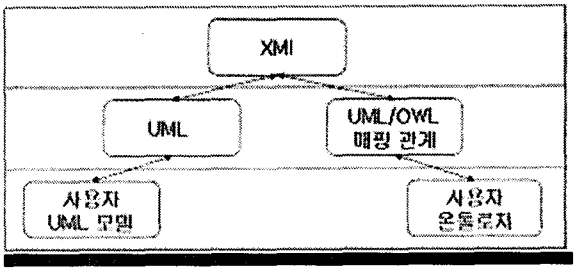

그림 3 - 온톨로지/UML 변환 구조

간단한 온톨로지를 실험예제로 온톨로지/UML간 변환을 간단히 보이고 이를 이용한 온톨로지의 효율적인 재사용 방안에 대하여 살펴 보겠다. 표 2는 [9]에서 생성한 wine온톨로지의 일부 이다.

표 2 - 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>
.....
```

주어진wine 온톨로지를 본 논문에서 제안한 방법을 토대로 하여 XMI로 변환한다. 이때 앞서 언급한 온톨로지/UML 매핑 관계에 따라 위 wine 온톨로지를 XML 파서를 이용하여 XMI로 변환한다. 아래 표 3은 XML 파서를 사용하여 XMI로 변환된 문서의 일부이다.

표 3 - XMI로 변환된 wine 온톨로지

```
.....
<UML:Class xmi.id="UMLClass.4" name="ConsumbleThing"
visibility="public" isSpecification="false"
namespace="UMLModel.3" isRoot="false" isLeaf="false"
isAbstract="false"
specialization="UMLGeneralization.14
UMLGeneralization.24" isActive="false"/>
<UML:Class xmi.id="UMLClass.5" name="PotableLiquid"
visibility="public" isSpecification="false"
namespace="UMLModel.3" isRoot="false" isLeaf="false"
isAbstract="false"
generalization="UMLGeneralization.14
UMLGeneralization.24"
specialization="UMLGeneralization.16
UMLGeneralization.17 UMLGeneralization.25"
isActive="false"/>
<UML:Class xmi.id="UMLClass.6" name="Winecolor"
visibility="public" isSpecification="false"
namespace="UMLModel.3" isRoot="false" isLeaf="false"
isAbstract="false"
generalization="UMLGeneralization.15
UMLGeneralization.26"
participant="UMLAssociationEnd.19" isActive="false"/>
<UML:Class xmi.id="UMLClass.7" name="WineGrape"
visibility="public" isSpecification="false"
namespace="UMLModel.3" isRoot="false" isLeaf="false"
isAbstract="false" participant="UMLAssociationEnd.22"
isActive="false"/>
<UML:Class xmi.id="UMLClass.8" name="Wine"
visibility="public" isSpecification="false"
namespace="UMLModel.3" isRoot="false" isLeaf="false"
isAbstract="false"
generalization="UMLGeneralization.16
UMLGeneralization.25"
participant="UMLAssociationEnd.20
UMLAssociationEnd.23" isActive="false"/>
<UML:Class xmi.id="UMLClass.9" name="WineDescriptor"
visibility="public" isSpecification="false"
namespace="UMLModel.3" isRoot="false" isLeaf="false"
isAbstract="false"
specialization="UMLGeneralization.15
UMLGeneralization.26" isActive="false"/>
.....
```

위와 같이 XMI로 변환된 온톨로지는 일반적으로 사용하는 UML도구에서 import하여 사용할 수 있다. 다음 그림 4는 변환된 XMI를 UML도구에서 import하여 UML로 표현 된 것이다.

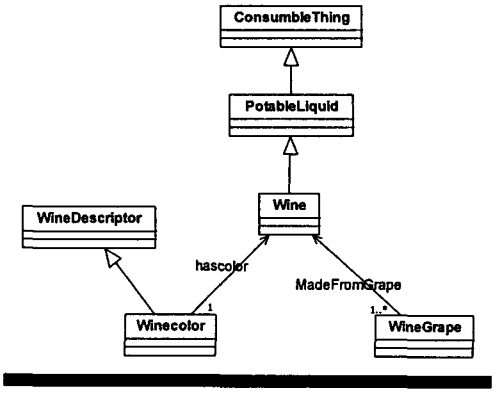

그림 3 - UML로 표현된 wine 온톨로지

위 그림에서 보여지듯이 복잡한 장문의 온톨로지가 UML로 변환된 후 그 가독성이 월등히 높아짐을 알 수 있다. 그간 존재 하였던 온톨로지 편집 도구들은 온 톨로지 전 체에 걸 친 시각 적 표현이 미흡하기 때문에 특정 개체의 이해나 온톨로지 전반에 걸친 이해에 문제가 있었으나. 이처럼 UML을 이용하여 표현함으로써 그 문제를 해결할 수 있다.

위와 같이 UML로 변환된 온톨로지는 기존에 있던 온톨로지를 토대로 하여 간단히 새로운 온톨로지로 확장하거나 사용 자에게 필요한 부분만을 수정하여 사용하게 한다. 또한 UML도구을 이용하여 각 클래스의 속성 및 연관 관계까지 한눈에 확인이 가능하고 UML에서 변경된 내용은 다시 XMI로 export 하여 앞서 설명한 과정의 역과정 즉 XML 파서를 이용하여 다시 온톨로지로 변환하면 변경된 내용이 적용된 온톨로지가 생성된다. 다시 말해 그림 4와 같이 온톨로지를 XMI로 변환하고 UML로 모델로 표현하여 수정된 결과를 다시 온톨로지로 변환하는 방식을 제안한 것이다.

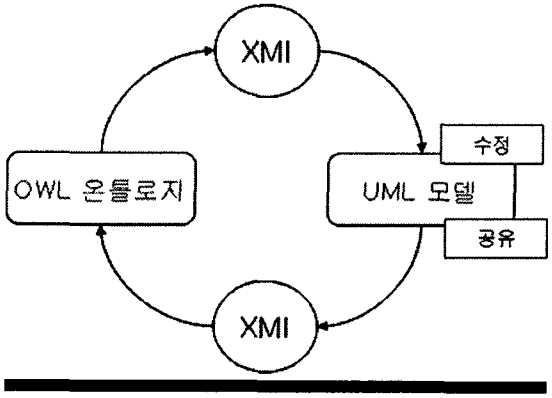

그림 4 - 온톨로지 재사용 변환 모델

이러한 방법은 후에 온톨로지 기술 방법이

진보하거나 바뀌더라도 매핑 관계만을 약간 수정하는 것으로 재사용이 가능하기 때문에 기술 변화에 효율적으로 대응이 가능하다.

또한 이미 널리 알려진 UML을 이용한 모델링 방법을 기반으로 온톨로지를 편집하기 때문에 사용자가 온톨로지 기술언어를 숙달하지 않은 상태에서도 온톨로지의 확장 및 수정이 가능하다.

결론

본 논문에서 기존의 온톨로지를 재사용하기 위하여, UML을 이용한 방법을 사용하였다. 이는 MDA를 이용한 방법으로 온톨로지를 UML로 변환하여 수정 및 확장이 용이하게 하며 다시 그 역과정을 통해 온톨로지로 변환 하는 것 이다. 기존의 UML을 통한 온톨로지 설계를 기반으로 그 역 과정을 생성함으로써 온톨로지의 자유로운 변환 및 공유가 가능함을 보였다. 이는 온톨로지를 상호운용이 가능한 모델레벨로 변환하여 UML로 쉽게 재사용 가능하고 미래에 있을 기술 및 시스템 인프라 변화에 효율적으로 대처할 수 있으며[15] 높은 가독성과 신뢰도를 지니는 온톨로지를 만들 수 있음을 의미한다.

기존의 도구들 또한 사용자 편의를 위해 개발되었지만 온톨로지 전체에 걸친 표현력이 부족하여 사용자가 이해하기 어렵고 사용 방법 또한 생소하고 온톨로지 기술언어를 알지 못한다면 사용법을 배우기도 어렵다. 또한 그 기능 역시 제한적인 상황에서 밖에 사용할 수 없으므로 그 효용성이 떨어진다. 하지만 본 논문에서 제안한 방법은 사용자가 배우기 힘들고 오랜 시간을 소모해야 하는 온톨로지 언어를 배우지 않고 사용자에게 널리 알려진 모델링 언어인 UML을 이용하여 손쉽게 온톨로지 구조 및 논리를 이해하게 하며 그 수정 방법 또한 용이하다.

XMI는 기본적으로 MDA에 근거하여 UML로 설계된 모델을 다른 포맷의 모델로 변환하고 교환하기 위해 사용되는 표준이다[15]. 따라서 본 연구를 토대로 다른 온톨로지 포맷과의 매핑을 정의한다면 온톨로지 포맷간의 전환 또한 이루어 질수 있을 것이다. 또한 잘 정의된 데이터베이스 역시 XMI로 표현이 가능하기 때문에 데이터간 연결관계가 잘 정의 되어 있다면 데이터베이스의 온톨로지 변환도 가능하다. 이러한 차후 본 논문에서 제안한 방법을 기반으로 위와 같은 과제를 해결한다면, 시맨틱 웹의 발전의 걸림돌이 된 문제를 해결함으로써 보다 효율적인 시맨틱 웹의 구현 가능성이 높아질 것이다.

참조문헌

[1] Berners-Lee. Tim, Hendler. J, Lassila. O (2001) "The Semantic Web," *Scientific American*, pp.28-37

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

[3] 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.

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

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

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

[7] MOF:http://www.omg.org/mof/

[8] 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,

[9] 이윤수, 김태석, 양진혁, 정인정, 소프트웨어 공학적 방법을 이용한 온톨로지의 효율적인 설계 및 생성에 관한 연구, 22회 한국정보처리학회 추계학술발표대회 논문집 11권 2호, pp645-648, 2004. 11

[10] 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.

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

[12] OWL:http://www.w3.org/TR/2004/REC-owl-features-20040210/

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

[14] 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.

[15] Dragan Đurić MDA-based Ontology Infrastructure ComSIS Vol. 1, No. 1, February 2004