DL기반에 의한 이질적 시스템간의 메타모델 매핑

홍현술*
*원광보건대학 정보컨텐츠과
e-mail:hshong@wkhc.ac.kr

Description Logic based Mapping of Meta Models between Heterogeneous Systems

Hyeun-Sool Hong*
*Dept of Information Contents, Wonkwang Health Science College

요 약

시맨틱 웹은 시스템이 더욱 효과적으로 정보를 액세스하고 이용이 가능하도록 하는 의미적 정보로서의 웹을 풍부하게 하는데 목적을 두며, 이는 온톨로지의 개념표현과 추론기능을 기반으로 한다. 온톨로지는 지식의 상호 커뮤니케이션을 위하여 개념적으로 명확하고 간결한 토대를 수립하기 위한 의미를 제공한다. 그런데 현재의 온톨로지 개발환경은 강력한 모델링 툴이나 경험이 풍부한 전문적인 온톨로지 구축 인력이 부족한 현실이다. 따라서 본 논문에서는 기존의 많은 개발자들에게 친숙해 있는 UML 또는 ER 도구를 이용하여 획득된 정보가 온톨로지 언어인 OWL의 정보와 커뮤니케이션이 가능하여 온톨로지 모델링 작업의 효율성을 높일 수 있도록 이들 사이의 메타모델 매핑변환을 시도하였다. 매핑의 기반에는 DL을 이용하였다.

1. 서론

오늘날 인터넷을 기반으로 하는 웹 기술이 의미정보를 이용하여 정보 이용의 효율성을 향상시키려는 연구로부터 의미 태그를 중심으로 한 메타데이터 정보 모델링 등이 출현하였고, 이를 개념 수준의 의미처리로 추상화한 온톨로지(ontology) 기술이 개발되게 되었다. 웹은 온톨로지의 개념 표현과 추론 기능을 기반으로 시맨틱 웹(Semantic Web)으로 전환되고 있으며, 온톨로지는 정보의 표현과 처리를 지식 표현과 추론 처리로 고도화하여 차세대 컴퓨터 기술을 실현하는 핵심 역할을 하고 있다.

메타데이터의 주석을 생성하고, 해석하고, 비교하기 위하여, 온톨로지가 필요하다. 메타데이터가 정보소스의 컨텐츠로부터 쉽게 생성될 수 있는 반면에, 온톨로지의 개발은 의미적 주석의 대규모적인 확대로 인하여 병목현상을 야기 시킨다. 그러나 현재의 상태는 이를 해결할 수 있는 강력한 모델링 툴이나 전문적인 온톨로지 구축 인력이 극히 부족하다. 온톨로지 언어의 중요한 목적중의 하나는 인간의 이해보다는 시스템의 이해에 중점을 둔다. 따라서 이러한 도구 들은 온톨로지 개발에 경험이 없는 사용자들에게는 용이하지 못하다. 그러나 UML이나 ER 도구 들은 인간의 작업 환경에 적합하도록 만들어진 도구이다. 따라서 기존에 사용되어온 어플리케이션 시스템을 유지하면서 온톨로지 모델링 작업을 한다면, 온톨로지 개발에 경험이 많지 않은 사용자들에게는 이용 가능한 도구와 전문성의 관점에서 효율성을 증대 시킬 수 있다.

본 논문에서는 기존의 개발자들에게 익숙해져 있는 UML, ER (Entity-Relationship) 모델, 그리고 현재 W3C의 온톨로지 개발 표준 언어인 OWL의 각기 다른 환경에서의 정보가 상호 커뮤니케이션이 가능하도록 모델의 매핑 변환을 시도하였다. 그리고 이들 이질적인 모델사이의 매핑을 용이하게 하기 위하여 OWL의 추론지원의 토대가 되고 있는 술어 로직(Description Logic : 이하 DL)을 이용하였다. 다양하고 복잡한 구조와 요소를 가지는 이질적인 시스템의 모델을 통일된 의미로 정의하는 것은 어려운 문제이다. 그러나 이러한 문제는 메타모델의 매핑 변환 접근법을 이용하면 의미적 정의가 가능할 수 있다.

2. DL과 OWL

Prolog, F-Logic, 또는 DL과 같은 로직 모델들은 규칙을 매핑이나 머징을 하여 서로 다른 데이터 소스를 연결할 수 있도록 돕는다. 또한 통합된 정보 베이스에 관한 연역적 비즈니스 로직을 쉽게 구축할 수 있게 하고, 지식베이스의 일관성 체크를 돕는다.

DL은 지식표현의 제한된 형식을 가지며, 특히 자동 추론을 지원하는데 용이하다.[1] 따라서 DL은 이질적인 시스템을 결합하는데 공통의 중심에 사용하기에 적합하다. DL은 의미적 표현성의 풍부함을 유지하면서, 사용의 용이성에 목표를 둔 1차 술어 논리(first-order predicate logic)의 부분집합(subset)이라 볼 수 있다. 본 연구에서 제시한 메타모델의 결합은 일련의 양방향 매핑이다. 즉 메타모델 UML, ER, OWL의 각각의 어느 하나와 중심의 DL 메타모델 사이에 매핑이 이루어진다. 그렇게 함으로써 최악의 경우에 클래스의 N * N 매핑의 폭증(explosion)을 피할 수 있다.

DL은 지식표현 시스템의 핵심인 지식베이스를 정의하며, DL 지식베이스는 다음과 같이 세 가지 요소로 나누어진다. [2,3]

- 용어 또는 스키마(terminology or schema) : 'TBox'라 하며, 어플리케이션 도메인의 어휘이다.

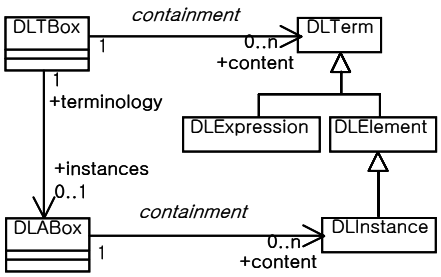
■ 단언(assertion) : 'ABox'라 하며, 어휘의 항(terms)으로 표현되어지는 명명된 개체들이다.

■ 술어 언어(description language) : 표현식(expression)을 구축하기 위하여 항(term)과 연산자(operator)를 정의한다.

TBox와 ABox 요소들은 어플리케이션 도메인에서 둘로 구분되는 메타레벨을 표현한다. 즉, TBox의 요소들은 ABox에 인스턴스화 되는 요소의 정의들이다.

대부분의 DL은 기초적인 용어로서 최소점(minimum), 원자 개념(atomic concepts), 원자 역할(atomic roles), 그리고 두 개의 특별한 개념인 최고점(Universal 또는 top)과 최저점(Empty 또는 bottom)을 정의한다.

그림 1은 DL 메타모델에서의 TBox와 ABox의 관계이다.



(그림 1) DL 메타모델의 TBox와 ABox

OWL(The Web Ontology Language : 이하 OWL)은 W3C를 중심으로 시맨틱 웹을 위한 표준 온톨로지 언어로 제안되고 있다. 온톨로지 표준 언어는 편집기, 스토리지, 추론엔진, 메타데이터 주석 도구, 웹기반 온톨로지 인프라구조의 개발을 지원하게 된다.[4]

일반적인 웹 구조 속으로 적합시키기 위하여, OWL은 XML 기반 인코딩과 그 배경이 되는 RDF(S)를 포함하고 있다. OWL 특성은 인간이 사용하는 것에 목적을 두기보다는 오히려 시스템에 의해서 다루어지도록 설계되었다. 잘 구축된 논리적 의미는 고품질의 온톨로지 생성의 선행요건이며 이는 OWL의 설계 원리이다. OWL은 논리적 추론을 가능하게 하는 특성들의 집합이라 할 수 있다.

OWL은 DL의 이론에 적응될 수 있는 의미구조를 갖는다. DL은 인터프리테이션을 추상적인 도메인으로 매핑한다. 따라서 DL의 강건하고 완전한 추론 절차가 OWL 모델을 위한 추론 지원을 제공하는데 사용될 수 있다.[5]

<표 1> 메타모델의 로직관련 요소 비교

	의미	DL	UML	OWL	ER
AL	Atomic Concepts, Universal Concept, Bottom Concept, Atomic Negation, Intersection, Value Restrictions and Limited Existential Quantification	○	△ Atomic Concepts, Value Restrictions and Limited Existential Qualification	○	△ Atomic Concepts, Value Restrictions and Limited Existential Qualification
C	Full Negation or Complement	○		○	
E	Full Existential Quantification	○		○	
H	Role Hierarchies	○	○	○	
I	Inverse Roles		○	○	
N	Unqualified Number Restrictions	○	○	○	○
O	Enumerated Classes		○	○	
R+	Transitive Role	○		○	
V	Union Constructor	○		○	
D	Datatypes	○	○	○	○

표 1은 선택된 메타모델에서 로직에 초점을 둔 모델간의 비교분석 결과이다. 표 1 에서 '○'은 완전일치를 의미하고, '△'은 부분일치를 의미한다. 그리고 '*VE*'는 의미적으로 '*C*'와 동치이다.

3. 매핑변환을 위한 분석

3.1 매핑의 문제점

매핑변환 연구에서 해결의 주요 문제점은 이질적인 환경의 모델간의 관련성이다. 구조나 의미, 또는 기술적인 차이를 이해하기 위해서 먼저 각 모델의 특성과 개념, 프로퍼티 등을 분석한다.

서로 다른 유형의 메타모델을 하나의 구조로 통합시키는 데는 다음과 같은 방식이 있다.

■ 우선 하나의 메타모델(목표모델)을 클래스로 지정하고, 그것의 서브클래스로서 또 다른 메타모델(소스모델)을 표현한다.

■ 우선 하나의 메타모델(목표모델을)을 지정하고, 그것의 내부로 또 다른 메타모델(소스모델)을 변환시키는 것이다.

■ 통합하고자 하는 각각의 메타모델(목표모델/소스모델)들을 별도로 유지하고, 중간 매개의 역할을 하는 모델(매개모델)안으로 양쪽의 메타모델들을 매핑시킨다.

처음 두 가지 방법은 바람직하지 않다. 왜냐하면 양자의 메타모델은 이질적인 시스템이기 때문에 변환된 결과가 목적하는 대로 나타나지 않고, 오히려 지정된 목표모델과는 다르게 왜곡되어 나타날 수 있고, 또한 목표모델의 구조를 아주 복잡하게 만들 수 있다. 그러나 세 번째의 매개모델을 이용한 매핑에서는 소스모델 요소들과 목표 모델 요소들 사이에 관계를 유지할 수 있기 때문에, 매핑의 결과는 무결성(integrity)을 유지할 수 있다. 그리고 일반적으로 모델간의 매핑은 N*N의 매핑과정을 수행해야 하지만, 매개모델을 이용하면 이러한 N*N 매핑 폭증을 피할 수 있다. 본 연구에서는 매개모델로서 DL메타모델을 이용한다.

이질적인 환경의 메타모델 사이에 변환의 근본 문제는 각 메타모델에 포함되어 있는 대응 요소들이 임의적이고 불규칙한데 있다. 즉, 메타모델간의 매핑시에 모델을 구성하는 각 요소들이 단일요소가 아니거나(1:1 대응이 아님.), 또는 한 모델에서는 두 가지 유형의 개념으로 표현되는 요소가 다른 모델에서는 한 개의 통합된 개념으로 존재할 수 있는 경우이다. 따라서 매핑 과정에서 요소들의 손실이 발생하지 않도록 하기 위한 방법이 필요하다.

본 연구에서는 소스모델의 요소들과 목표모델의 요소들 사이에 손실 방지를 위하여 해당 요소의 발생시 고유 모델의 정보를 파악할 수 있는 태그를 첨부하여 매핑을 수행한다. 이는 소스모델과 목표모델의 요소들의 이동경로를 추적할 수 있고, 후에 역추적의 기능도 가능하다.

표 2는 선택된 각 메타 모델의 개념을 의미적 구조적으로 분석하여 매핑에 적용시킬 객체와 프로퍼티의 추출된 결과이다.

<표 2> 메타모델의 매핑 요소 비교

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

3.2 이중 구조 프로퍼티의 매핑 처리

프로퍼티 개념을 갖는 UML의 Attribute와 Association 두 요소는 DL의 단일요소 Role과 매핑 한다. 그리고 더 나아가서 ER의 두 요소 Attribute와 Relationship과 매핑을 해야 한다면, 소스 요소의 정보를 참조해야만 정확한 매핑이 가능할 수 있다. 그런데 DL, ER, UML, 그리고 OWL 모두는 집합(set)과 관계(relationship)의 이론에 토대를 이룬다. 그래서 만약 어떠한 모델이 DL을 통하여 UML에서 ER로 매핑된다면 실제로 의미적인 손실은 없다. UML에서 DL에 매핑된 모든 프로퍼티가 다시 ER의

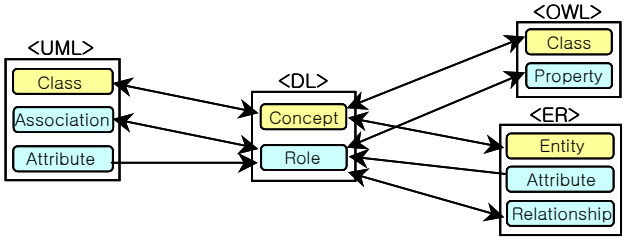
Relationship과 매핑한다면, 소스 모델을 확인하여 손실된 내용의 파악이 가능하다.

소프트웨어 명세서 관점으로 볼 때 모델의 의미적 내용은 외부적으로 보일 수 있는 구체화된 행위자이다. 만약 한 명세서의 두 가지 표현이 구체화되어서 같은 의미를 갖는다면, 각각의 표현은 서로에게 정제된 의미(refinement)이다. 결과적으로 그것들은 동일(equivalent)하다고 볼 수 있다.

UML의 두 개의 프로퍼티 요소인 Attribute와 Association 중에서 어느 것을 선택할 것인가는 관계데이터베이스 시스템의 구현과 관련이 있다. 관계데이터베이스에서 일반적으로 Attribute는 테이블의 열(columns)로 구현되고, Association은 외부키(foreign key)에 의해서 연결되는 다중 테이블로 구현되므로 후자가 전자보다 훨씬 더 구현의 범위가 넓어진다. 따라서 어느 요소를 선택할 것인가는 구현의 계획에 달려있다. 이러한 구현 결정은 의미적 개념에 중심을 두는 온톨로지에서는 중요하지 않다.

또한 UML의 Attribute와 Association 중에서 선택하는 또 다른 이유는 사용자가 모델을 이해하는데 대한 기여도이다. 즉 모델을 이해하는데 부가적인 정보가 추가된다면, 훨씬 더 유리할 것이다. 따라서 모델을 다루는데 좀더 나은 이해를 원한다면 Attribute가 아닌 Association을 선택하여 표현한다.

그림 2는 메타모델 사이의 매핑 변환의 기본 구조이다. 각 모델의 프로퍼티의 매핑은 의미적으로는 동일한 모델을 생성하지만, 구조적으로는 다를 수 있다. 예를 들면, UML에서 DL로 매핑할 때 UML Association 와 Attribute 양쪽은 DL의 Role과 매핑 된다. 역으로 DL의 Role은 오직 UML Association과 매핑 된다.



(그림 2) UML, ER, OWL, 그리고 DL사이의 매핑 기본구조

4. 메타모델 매핑변환

4.1 단순 매핑

■ UML과 DL

UML과 DL 사이의 매핑을 위한 주요 규칙은 그림 3과 같다. 규칙 Class2Concept와 Concept2Class는 Class와 Concept 사이의 양방향 매핑을 나타낸다. 그리고 규칙 Assoc2Role와 Attr2Role는 UML의 Attribute와 Association에서 DL의 Role로의 매핑을 표현하고, 반면에 Role2Assoc는 DL의 Role에서 역으로 UML의 오직 Association으로만 단일 매핑하는 것을 표현한다.

LINKING 문장은 매핑될 때 모델들 사이에 대응되는 정보를 추적 기록하기 위하여 사용되는 트랙킹(tracking) 모델에서 클래스를 식별한다. 그림 3의 규칙에서 사용된 UML2DL 트래킹 모델은 클래스 RoleForAttr을 포함한다.

```
TRANSFORMATION UML2DL (uml, dl)

RULE Assoc2Role (a, r)
  FORALL UMLAssoc a
  MAKE DLRole r
  SETTING r.uniqueID = a.name;

RULE Attr2Role (a, r)
  FORALL UMLAttr a
  MAKE DLRole r
  SETTING r.uniqueID = a.name
  LINKING RoleForAttr
    WITH attr = a, role = r;

RULE Class2Concept (c1, c2)
  FORALL UMLClass c1
  MAKE DLConcept c2
  SETTING c2.uniqueID = c1.name;

TRANSFORMATION DL2UML (dl, uml)

RULE Role2Assoc (r, a)
  FORALL DLRole r
  MAKE UMLAssoc a
  SETTING a.name = r.uniqueID;

RULE Concept2Class (c1, c2)
  FORALL DLConcept c1
  MAKE UMLClass c2
  SETTING c2.name = c1.uniqueID;
```

(그림 3) UML과 DL의 매핑

■ ER과 DL

그림 4에서 보는 바와 같이 ER과 DL사이의 매핑은 UML과 DL간의 매핑규칙과 유사하다. Entity와 Concept, 그리고 Relationship과 Role은 그림 3의 단일 매핑과 같이 동등하게 수행하며, ER의 Attributes는 규칙 Attribute2Role을 통해서 DL의 Role과 매핑한다.

```
TRANSFORMATION ER2DL (er, dl)

RULE Relationship2Role (r1, r2)
  FORALL ERRelationship r1
  MAKE DLRole r2
  SETTING r2.uniqueID = r1.name;

RULE Attribute2Role (a, r)
  FORALL ERAttribute a
  MAKE DLRole r
  SETTING r.uniqueID = a.name;
  LINKING RoleForAttribute
    WITH attribute = a, role = r;

RULE Entity2Concept (e, c)
  FORALL EREntity e
  MAKE DLConcept c
  SETTING c.uniqueID = e.name;

TRANSFORMATION DL2ER (dl, er)

RULE Role2Relationship (r1, r2)
  FORALL DLRole r1
  MAKE ERRelationship r2
  SETTING r2.name = r1.uniqueID;

RULE Concept2Entity (c, e)
  FORALL DLConcept c
  MAKE EREntity e
  SETTING e.name = c.uniqueID;
```

(그림 4) ER과 DL의 매핑

■ OWL과 DL

그림 5와 같이 OWL과 DL간의 매핑은 더욱 간단하다. Class는 Concept과 대응하고, Property는 Role과 대응한다.

```
TRANSFORMATION OWL2DL (owl, dl)

RULE Property2Role (p, r)
  FORALL OWLProperty p
  MAKE DLRole r, r.uniqueID = p.localname
  LINKING PropForRole
    WITH prop = p, role = r;

RULE Class2Concept (c1, c2)
  FORALL OWLClass c1
  MAKE DLConcept c2, c2.uniqueID = c1.localname
  LINKING ClassForConcept
    WITH cls = c1, con = c1;

TRANSFORMATION DL2OWL (dl, owl)
```

```
RULE Role2Property (r, p)
  FORALL DLRole r
  MAKE OWLProperty p
  SETTING p.localname = r.uniqueID,
  LINKING propForRole
    WITH prop = p, role = r;

RULE Concept2Class (c1, c2)
  FORALL DLConcept c1
  MAKE OWLClass c2
  SETTING c2.localname = c1.uniqueID;
```

(그림 5) OWL과 DL의 매핑

4.2 매핑의 확장

단순 매핑은 다중 변환으로 그 범위가 확장될 수 있는데, 이때 모델 안에서의 요소들의 손실을 막기 위하여 추가적인 정보가 필요하다. DL을 매개로한 UML과 ER 사이의 매핑(UML-DL-ER)이라면, 모델러가 Association이나 Relationship보다는 오히려 Attribute를 사용했을때, 특별한 상황을 식별하고자 하는 소스 모델의 패턴을 감지하기 위하여, 목표 모델에 이러한 구별을 보존하기 위한 추적 정보를 태그로서 첨부시켜야 한다.

```
TRANSFORMATION DL2ER-fromUML
    (dl, uml2dltrace, er)
EXTENDS DL2ER (dl, er)

RULE Role2Attribute-fromUML (r, a)
  SUPERSEDES Role2Relationship (r, r2)
  FORALL DLRole r,
RoleForAttr@uml2dltrace ra
  WHERE ra.role = r
  MAKE ERAttribute a FROM r
  SETTING a.name = r.uniqueID;
```

(그림 6) 소스모델 UML에서 ER Attribute로의 확장 매핑

그림 6은 DL과 ER 사이의 매핑을 확장하는 예를 보인다. Role이 UML Attribute로부터 본래 매핑 되었다면, 그 규칙은 ER의 Relationship 대신 Attribute와 매핑된다.

DL2ER의 확장 매핑을 거치면 규칙 Role2Relationship을 오버라이드(override) 하기 위하여 규칙 교체(SUPERSEDES)를 이용하며 이때 추가 매개인자를 갖는다. 소스 UML 모델이 DL과 매핑 되었을 때 생성되었던 추적 모델의 정보를 관찰하여 각각의 Role이 UML의 Association이나 Attribute의 어느 것으로부터 매핑되는지를 확인할 수 있다. 만약 추적의 내용에 RoleForAttr 클래스가 있다면 그 Role은 UML Attribute로부터 선행된 매핑이 된다. 유사한 매핑 확장으로, 반대의 상황인 ER Attribute로부터 매핑되었던 각각의 DL Role에 대해서도 추적 모델의 정보를 관찰하여 DL2UML 매핑의 결과가 UML의 Attribute 쪽으로 결정될 수 있다.

이러한 접근법의 이점은 모든 것이 규칙의 공통부분으로부터 확장하므로, 많은 변환과정에서 똑같이 반복되는 규칙의 중복을 피하고, 확장하고자 하는 모델의 적합한 상황에 맞는 확장 변환 부분만을 선별하여 적용할 수 있는 점이다. 규칙의 교체와 확장은 그림 7에서와 같이 규칙의 행위자를 변경시킬 수도 있고, 또는 특정 조건하에서의 규칙을 중지시킬 수도 있다. 그림 7은 DL의 Role을 UML의 Attribute로 매핑시킨다.

WHERE 절은 적용시킬 조건을 부여한다. 그림 7의 예에서 Role2Attr는 오직 Role2Assoc로 교체되며, DL의 Role에 대응하는 OWL의 Property는 OWLDataTypeProperty이다. 트랙킹 클래스는 PropForRole이다.

```
TRANSFORMATION DL2UML-fromOWL
            (dl, owl2dltrace, uml)
EXTENDS DL2UML (dl, uml)
```

```
RULE Role2Attr (r, a)
  SUPERSEDES Role2Assoc (r, a)
  FORALL DLRole r,
PropForRole@owl2dltrace pr
  WHERE pr.role = r AND pr.prop = p
    AND OWLDataTypeProperty p
  MAKE UMLAttribute a FROM r
  SETTING a.name = r.uniqueID;
  LINKING RoleForAttr
    WITH attr = a, role = r;
```

(그림 7) 소스모델 OWL에서 UML Attribute로의 확장 매핑

5. 결론

시맨틱 웹에 사용되지 않았던 기존의 지식표현 언어를 이용한 온톨로지 모델링에 관한 연구는 사용자들의 기능성과 전문성의 효과를 발휘할 수 있고, 온톨로지 이용의 중요 특성 중의 하나인 재사용 가능성과 확장 가능성을 실현하는데 의의가 있다고 본다.

따라서 본 논문에서는 이질적인 시스템 모델 간에 상호 커뮤니케이션이 가능할 수 있도록 메타모델의 매핑 변환 접근을 시도하였다. 본 논문에서 선택된 이질적인 환경의 메타모델은 UML, ER, OWL, DL 등이다. 이질적인 환경의 메타모델 사이에 변환 연구에서 해결해야할 근본적인 문제는 각 메타모델에 포함되어 있는 대응 요소들이 임의적이고 불규칙한 점에 있다. 따라서 매핑 과정에서 요소들의 손실이 발생할 수 있는데 이를 방지하기 위하여 소스 모델에서 목표모델로의 매핑 과정을 추적할 수 있는 트레킹 시스템을 적용시키는 것이다.

본 연구는 온톨로지 모델링 작업의 효율성 제고에 목적을 두고 이질적 시스템간의 개념적 표현 모델의 매핑에 근거를 두었다. N*N 매핑의 증폭을 방지하기 위하여 DL을 중간 매개모델로 쌍방향 매핑의 모델을 적용하였다. 매핑과정 중에 대응 요소들의 추적 시스템을 적용하여 요소 및 구조 손실을 방지하였다. 기존의 매핑연구는 의미보다는 구조적인 측면에 무게를 두었고, 정보지향 관점 보다는 통합 프로세스의 트랜잭션지향을 추구하였지만, 본 연구는 구조적인 점 뿐만 아니라 의미적인 측면에서 정보지향 관점의 매핑 변환을 수행하였다.

참고문헌

[1] David Frankel, Pat Hayes, Elisa Kendall, Deborah McGuinness, A Model-Driven Semantic Web Reinforcing Complementary Strengths, To be published in *MDA Journal*, July 2004, v00-07.

[2] F.Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider Editors, The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003, p.574.

[3] I. Horrocks, U. Sattler and S. Tobies; Practical Reasoning for Very Expressive Description Logics; *Ligic Journal of the IGPL*, Volume 8, Issue 3: May 2000.

[4] M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. Web Ontology Language(OWL) Reference Version 1.0. Working draft, W3C, November 2002.

[5] F.M. Domini, M. Lenzerini, D. Nardi, and A. Schaerf, Reasoning in description logics, In Gerhard Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pp.193-238, CSLI Publications, 1996.