웹 온톨로지 구축을 위한 OWL 저작 시스템

OWL Authoring System for building Web Ontology

이무훈(Moohun Lee)*, 조현규(Hyunkyu Cho)**, 조현성(Hyeonsung Cho)**,
조성훈(Sunghoon Cho)*, 장창복(Changbok Jang)*, 최의인(Euiin Choi)***

초 록

현재의 웹 검색은 단순히 키워드 매칭만을 통해 필요한 정보들을 검색하기 때문에 그 결과가 사용자가 원하는 정보와는 의미적으로 상이한 결과들을 많이 포함하고 있다. 사용자가 원하는 정보와 의미적으로 정확히 일치하는 정보들을 추출하기 위해서는 웹 자원에 대한 정확한 의미 부여와 웹 자원들 사이의 의미적 연관성을 기술할 수 있는 지식 표현 수단인 온톨로지가 필요하다. 웹 기술 표준화 단체인 W3C에서는 이와 같은 웹 자원에 대한 의미 표현 기술로 OWL(Web Ontology Language)이라는 웹 온톨로지 언어를 발표하였으나 아직 이를 효과적으로 저작, 편집할 수 있는 전용 도구의 개발은 아직 미비한 실정이다. 따라서 본 논문은 OWL의 생성 및 편집을 효과적으로 제공할 수 있는 저작 시스템을 설계하고 구현하였다.

ABSTRACT

Current web search includes a lot of different results with information that user does not want, because it searches information using keyword mapping. Ontology can describe the correct meaning of web resource and relationships between web resources. And we can extract suitable information that user wants using Ontology. Accordingly, we need the ontology to represent knowledge. W3C announced OWL(Web Ontology Language), meaning description technology for such web resource. However, the development of a special tool that can effectively compose and edit OWL is inactive. In this paper, we designed and developed an OWL authoring system that can effectively provide the generation and edit about OWL.

키워드 : 시맨틱 웹, 온톨로지, OWL, 온톨로지 편집 도구

Semantic Web, Ontology, Web Ontlogy Language, Ontology Editing Tool

---

본 연구는 산업자원부 지역협력연구센터사업(R12-2003-004 -03002-0) 지원으로 수행되었음.

* 한남대학교 컴퓨터공학과

** 한국전자통신연구원 지능형로봇연구단 지능형서비스플랫폼연구팀

*** 한남대학교 컴퓨터공학과 교수

1. 서 론

현재 웹은 IT 인프라의 발달로 인해 다양한 정보원으로써 무한한 저장 능력을 갖춘 정보의 바다로 부각되었다. 웹에는 수많은 단체와 개인들이 서로 다른 목적으로 생성한 수많은 문서와 정보가 포함되어 있다. 이처럼 정보의 폭발적인 증가로 인해 유용한 정보를 효율적으로 찾는 것이 어렵게 되었다[1,4,10].

이와 같은 문제점은 현재의 웹이 HTML (Hyper Text Markup Language) 기반의 표현 위주의 웹으로 구성되어 있기 때문이다. 표현 위주의 웹은 단일 기종 내에서도 디자인의 변화에 따라 정보도 함께 표현해야 하므로 이는 생성된 정보들의 재사용성을 떨어뜨리고, 컴퓨터가 이해할 수 없어 사람에 의한 정보해석과 같은 추가 비용을 필요로 한다. 또한 정해진 태그들만을 사용해야 한다는 점과 문서의 구조적인 정보를 표현할 수 없으므로 정보 추출이 어렵다는 단점을 가지고 있다. 이러한 HTML의 단점을 해결하기 위해 XML(eXtensible Markup Language)이 제안되었다[2]. XML은 사용자가 임의적으로 태그를 정의할 수 있고 문서의 구조를 정의할 수 있다. 그러나 이러한 XML은 단지 문법 기술에 초점을 두고 개발되었기 때문에 정보들 간의 관계를 표현하기에 부족하고 의미가 상속되지 못한다. 이러한 HTML과 XML의 단점을 보완하기 위해 RDF가 제안되었다. RDF는 시맨틱 웹을 지원하기 위한 기반 구조로써 분산된 다양한 자원들을 기술하기 위한 것으로 상이한 메타데이터를 효율적으로 교환하고 이해할 수 있는 정보 교환 수단을 제공한다. 하지만 RDF는 정보간의 동치성과 같은 관계에 대한 표현 능력이 미약하고 추론을 위한 규칙을 표현할 수 없다는 단점을 가지고 있다. 이 문제점을 개선하기 위해 온톨로지, OIL(Ontology Inference Layer), DAML(DARPA Agent Markup Language)과 같은 언어가 개발되었으며 이러한 언어를 통해 정보의 표현력 부족을 해결하고, 효율적인 정보 교환을 통해 이기종 간의 정보 시스템 통합 뿐만 아니라 상호운용성을 보장하여 사용자가 원하는 자동화된 웹 서비스를 제공할 수 있다[5,6].

이러한 기반 기술들을 토대로 기존 웹의 문제점들을 해결하기 위해 제안된 개념이 시맨틱 웹이다[2]. 시맨틱 웹은 컴퓨터 스스로가 웹에 연결된 정보의 의미를 인식하고 사용자가 필요로하는 정보를 검색하여 검색된 정보에서 지식을 추론할 수 있는 기능을 제공한다. 즉, 사람이 웹 정보에 대하여 의미를 파악하고 의미에 따라 선택적으로 정보를 획득, 통합, 가공할 수 있는 웹 환경을 제공하는 것이다. 그러나 앞에서 기술한 바와 같이 XML과 RDF는 컴퓨터가 XML 문서에 포함된 태그의 의미를 정확하게 표현하고 절차적 추론 과정을 수행할 수 있게 하기 위해서는 온톨로지의 개념이 필요하다[5]. 온톨로지는 응용프로그램 사이에서 웹 기반 지식 처리, 공유, 재사용을 가능하게 하고, 전자상거래에서 구매자와 구입자간의 기계기반의 의사소통을 가능하도록 한다. 이러한 온톨로지는 XOL(XML-based Ontology Exchange Language), OML(Ontology Markup Language), SHOE, OIL, DAML+OIL 등이 있다.

이와 같은 다양한 온톨로지 언어를 바탕으로 온톨로지 관련 브라우저, 편집기, 분석기, 추론 엔진 등의 소프트웨어가 존재하고 있지만 기존의 온톨로지 저작 시스템은 각기 다른 온톨로지 언어를 기반으로 하고 있고, 각기 다른 방식으로 개발되었기 때문에 온톨로지를 저작하고 탐색함에 있어 서로 연동하기가 어렵다는 문제점이 있다. 이러한 이유로 W3C를 중심으로 온톨로지 언어에 대한 표준화 작업이 진행되어 새로운 온톨로지 언어인 OWL을 발표하였다. 이러한 상황에서 웹 자원에 의미 정보를 추가하기 위해서는 온톨로지에 대한 전반적인 지식 없이도 온톨로지 문서를 손쉽게 생성할 수 있는 온톨로지 저작 시스템이 필요하다. 국외에서 온톨로지 저작 시스템에 대한 연구가 활발히 진행 중이지만, OWL을 위한 전용 시스템이 미흡한 실정이며 기존의 시스템들은 다양한 인터페이스를 제공하고 있지 못하여 온톨로지 전문가들조차도 사용하기 어렵다는 문제점을 가지고 있다. 따라서 본 논문에서는 이와 같은 문제점을 해결하기 위해 온톨로지 문서를 그래픽 인터페이스를 통하여 쉽게 생성 및 편집할 수 있고 복잡한 온톨로지 문서를 효과적으로 분석할 수 있는 OWL 저작 시스템을 설계하고 구현하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로서 기존 온톨로지 편집 도구에 대해 살펴보고, 3장에서는 제안한 OWL 저작 시스템의 각 기능의 설계 및 구현에 관하여 설명한다. 4장에서는 기존 온톨로지 편집 도구와 제안 시스템을 비교 분석하고, 5장에서는 결론 및 향후 연구과제를 제시한다.

2. 관련 연구

2.1 Protégé OWL Plugin과 ezOWL Plugin

Protégé는 국립의료도서관(National Library of Medicine), NSF(National Science Foundation), DARPA(Defense Advanced Research Projects Agency)의 후원을 받아 스탠포드 의과대학의 의료정보학과(Stanford University School of Medicine, Stanford Medical Informatics)에서 지식 기반의 구조를 작성하기 위한 시스템으로 15년간의 연구 기간을 거쳐 개발되었다. Protégé는 다른 지식 표현 언어와 호환이 가능하며, 온톨로지의 생성 및 수정을 위한 확장이 용이하여 현재 가장 광범위하게 활용되고 있는 온톨로지 개발/관리도구이다. 그리고 다양한 온톨로지 언어의 import/export 기능, 지식기반 데이터베이스 구축을 통한 질의 기능, 추론 기능 등의 풍부한 플러그인을 통한 확장성을 제공하고 있다. 최근 스탠포드에서 개발된 OWL Plugin과 국내에서 개발된 ezOWL Plugin을 포함하고 있는 Protégé 3.0 버전은 온톨로지 언어의 표준으로 자리 잡고 있는 OWL에 대한 편집 기능을 제공하고 있다[9,14]. Protégé의 경우 오랜 개발 기간만큼 완성도가 높은 시스템이지만 OWL이 표준화 되기 이전부터 개발되었던 도구이기 때문에 Protégé의 내부 온톨로지 모델은 OWL에서 제공하는 표준 온톨로지의 개념(concept)을 사용하지 않고, 기존에 Protégé 내에서 자체적으로 정의된 개념들을 사용하고 있어 OWL을 위한 전용 도구로써 적합하지 않다[3,7,14]. 따라서 Protégé의 경우 다양

한 온톨로지 표현을 내부 온톨로지 모델로 변환함으로써 범용적인 온톨로지를 개발/관리하는 시스템으로 적합하지만, 표준 OWL을 작성하기 위한 전용 편집 도구로 활용하기에는 부적합한 실정이다.

2.2 OilEd

맨체스터 대학에서 개발된 OilEd는 DAML+OIL에 기반으로 트리(tree) 방식으로 온톨로지를 편집하는 도구이다. 초기 OilEd는 OIL을 기반으로 하는 간단한 프리웨어 개발을 목표로 하였기 때문에 전반적인 온톨로지 구축 환경을 제공하지는 않는다. OilEd는 FaCT(Fast Classification of Terminologies) 추론 기관과 연결하여 DAML+OIL 온톨로지를 구축하는 과정에서 FaCT의 추론 기능을 활용하고 있다. 이 시스템은 DAML+OIL의 풍부한 표현력과 다양한 제약조건을 모두 지원하기 위해 노력하였으나, 시각적인 온톨로지 편집 환경을 제공하지 못함으로 대규모의 온톨로지 개발에 부적합하다[3,8,12].

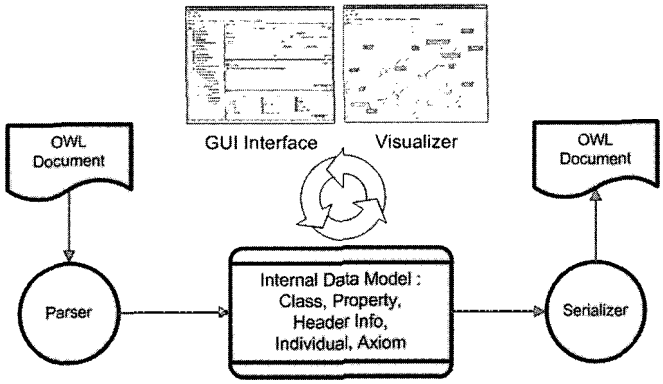
2.3 OntoEdit

OntoEdit은 독일의 ontoprise GmbH 사에서 개발하였으며, 그래픽을 이용하여 온톨로지의 생성과 유지보수을 제공하는 온톨로지 공학 환경(Ontology Engineering Environment)을 구축하고 있다. OntoEdit은 강력한 내부 온톨로지 모델을 가지고 있으며 이 모델은 개념(concept), 관계(relation), 원칙(axioms)이 가능한 중립적 모델링 표현 언어를 제공하고 있다. 또한 직관적인 인터페이스를 통하여 온톨로지에 대한 시각화 및 탐색을 지원하고 있다. 하지만 OntoEdit에서 사용하고 있는 내부 온톨로지 모델은 표현력에 있어 제약조건 및 동치관계를 명시할 수 없다는 단점을 가지고 있다[3,13,16].

3. OWL 저작 시스템의 설계 및 구현

본 논문에서 제안한 OWL 저작 시스템은 현재 W3C를 중심으로 온톨로지의 표준화를 위해 제안된 웹 온톨로지 언어인 OWL을 기반으로 GUI 인터페이스와 시각화 모듈을 통하여 OWL 문서를 직관적으로 편집하고 분석할 수 있다. 전체 시스템 구조는 <그림 1>과 같이 구성하였다.

OWL 저작 시스템은 텍스트 파일 형태의 OWL 문서를 입력 받아 내부 데이터 모델로 변환하는 파서(Parser), 온톨로지 의미 구성 요소들의 정보를 저장하기 위한 내부 데이터 모델, 내부 데이터 모델을 사용자에게 시각적으로 표시하고 생성할 수 있도록 하기 위한 GUI 인터페이스, 구축된 온톨로지의 정보들을 시각화하여 확인하고 탐색할 수 있도록 하기 위한 시각화 모듈(Visualizer), 그리고 최종적으로 생성된 정보들을 바탕으로 다시 OWL 문서를 생성해 주는 직렬화 모듈(Serializer)로 구성하였다.

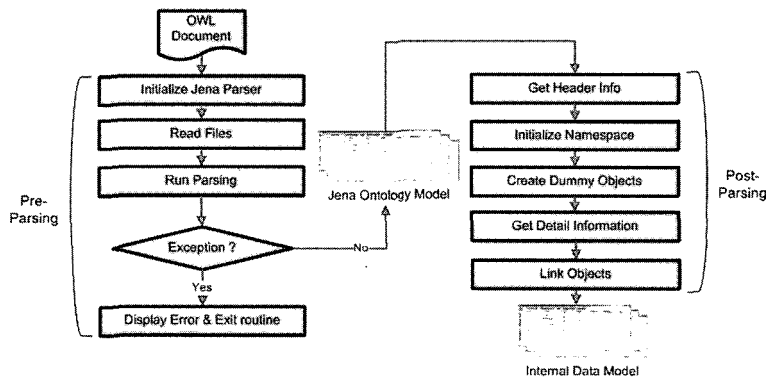


〈그림 1〉 OWL 저작 시스템의 구조

3.1 파서 설계 및 구현

파서는 OWL 문서의 구문을 검사하고, OWL 문서에 포함된 의미표현 요소들을 추출하여 내부 데이터 모델을 구축한다. 파싱(parsing)은 OWL 문서를 입력받아 먼저 Jena[11] 온톨로지 모델을 구축한 후, 내부 데이터 모델로 변환한다. 파서는 OWL 의미 요소마다 Jena 모델을 내부 데이터 모델로 변환하기 위한 메소드를 제공하도록 하였다.

파싱은 〈그림 2〉와 같은 순서로 진행되도록 설계하였다. 입력된 텍스트 형태의 OWL 문서를 먼저 pre-parsing 단계에서 Jena API를 통해 Jena Ontology 모델을 생성한다. post-parsing 단계에서 Jena Ontology Model을 참조하여 실제 시스템에서 사용될 Internal Data Model로 변환한다. 변환 단계는 먼저 온톨로지의 헤더 정보를 확인하여 온톨로지에서 사



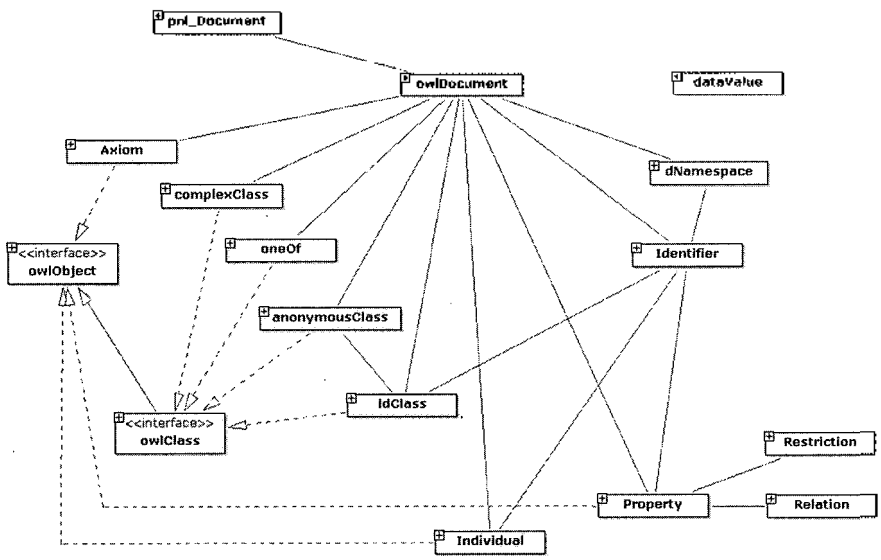
〈그림 2〉 파싱 처리 과정

용된 Namespace를 내부 모델에 등록하고, named Class, Property, Individual에 대한 Dummy 객체를 생성한다. 생성된 Dummy 객체가 생성된 후에는 모든 객체에 대해 해당 객체가 갖고 있는 세부 의미 요소들을 추출하여 내부 모델에 반영하고 이들 간의 참조관계를 연결하도록 구성하였다.

3.2 내부 데이터 모델 설계 및 구현

파싱 단계에서 추출된 요소들은 내부 데이터 모델로 변환하여 메모리에 저장함으로써 신속하고 빠른 참조를 통해 효율적으로 OWL 문서를 처리하도록 하였다. 내부 데이터 모델은 OWL 스펙에서 요구하는 의미 표현을 충실하게 제공할 수 있도록 owlDocument, owlObject, owlClass, idClass, anonymousClass, oneOf, complexClass, Axiom, Property, Individual, dNamespace, dataValue, Identifier, Restriction, Relation의 15개의 클래스를 정의하였다. owlDocument는 내부 데이터 모델의 최상위 객체로 온톨로지의 모든 의미요소들을 포함하며, Class, Property, Individual 등 자원 식별자를 통해 접근하는 객체에 대한 HashTable을 유지함으로써 보다 빠른 객체 참조가 가능하도록 구성하였다. 또한 추가적으로 사용자 인터페이스에서 빈번하게 이용되는 Class와 Property의 트리 모델도 내부 데이터 모델에서 관리하도록 하였다. 〈그림 3〉은 내부 데이터 모델의 클래스 다이어그램으로 각각의 클래스들 간의 관계를 보여준다.

내부 데이터 모델에 정의된 각각의 객체는 OWL 스펙에서 정의한 의미 요소들과 1:1로 대응되도록 하였으며, 의미 요소들을 저장하기 위한 멤버 변수들과 이들을 조작하기 위한 메소드, 그리고 사용자 인터페이스를 통해 발



〈그림 3〉 내부 데이터 모델의 클래스 다이어그램

생된 Action을 처리하기 위한 메소드로 구성하였다.

3.3 사용자 인터페이스 설계 및 구현

OWL은 온톨로지 생성을 위한 풍부한 표현력을 제공하고 있기 때문에 인터페이스의 설계에서도 모든 표현의 처리가 가능하도록 설계하였다. 따라서 본 논문에서는 OWL 명세서 분석을 통해 이를 효과적으로 편집할 수 있도록 Class, Property, Individual, Axiom, Header, Namespace 6개의 모듈로 나누어 설계하였다. Class, Property, Individual 모듈은 기본적인 OWL 요소를 정의하며, Axiom 모듈에서는 하나의 Class나 Individual 상에서 정의할 수 없는 DisjointWith, AllDifferent 요소를 정의하도록 하였다. Header에서는 OWL 온톨로지의 헤더 정보인 versionInfo, priorVersion, import, backCompatibleWith, inCompatibleWith를 정의한다. Namespace 인터페이스에서는 온톨로지 내부에서 사용하는 Namespace에 대해 수정을 할 수 있도록 하였다. 〈표 1〉은 각 인터페이스 모듈별 기능을 나타내고 있다.

3.3.1 Class 정의 인터페이스

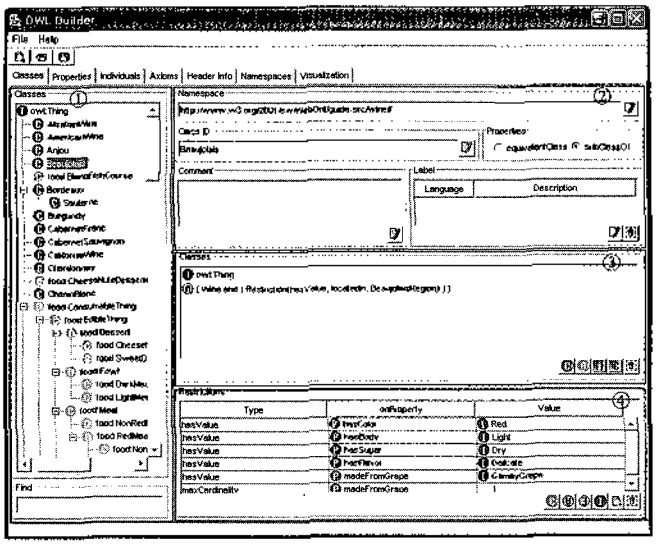
Class를 크게 named Class, anonymous Class, one-of, union Class, intersection Class, complement Class로 구분하여 구현하였으며, 일반적인 named Class를 정의하는 사용자 인

〈표 1〉 인터페이스 모듈별 기능

Class	Property	Individual

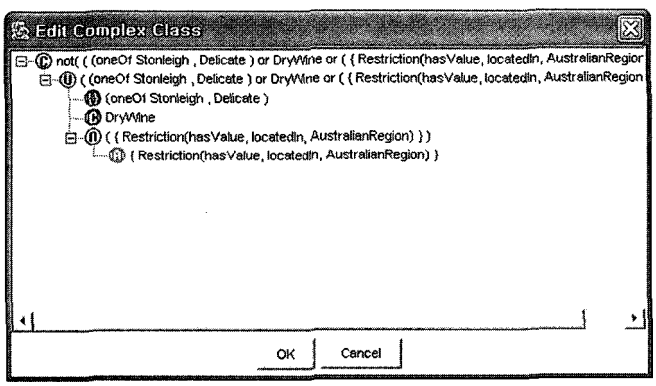
· Annotation 정의
  - ID, Label, Comment
· subClassOf
· equivalentClass
· Restriction 정의
  - Cardinality
  - Property Restriction
· Class Combination
  - intersectionOf
  - unionOf
  - complementOf
	· Annotation 정의
	  - ID, Label, Comment
	· subPropertyOf
	· equivalentProperty
	· inverseProperty
	· Domain
	· Range
	· Property 특성 정의
	  - Functional
	  - inverseFunctional
	  - Symmetric
	  - Transitive
		· Annotation 정의
		  - ID, Label, Comment
		· Individual간 관계
		  - sameIndividualAs
		  - differentFrom
		· Relation 정의

Axiom	**Header**	**Namespace**
· AllDifferent · disjointWith	· versionInfo · priorVersion · backwardCompatibleWith · inCompatibleWith · import	· Namespace 추가/삭제 · Default Namespace 설정



〈그림 4〉 named Class 생성 인터페이스

터페이스는 〈그림 4〉와 같다. ① Class Tree, ② Annotation 정의부, ③ Super Class(Equivalent Class) 정의부, ④ Restriction 정의부로 구성하였으며, ①은 온톨로지에 정의된 모든 Class들의 Sub Class 관계를 트리 형태로 표시한다. 선택된 Class에 대해 우측 세부 정보 패널(②, ③, ④)에서 수정할 수 있도록 구성하였다. ②에서는 해당 클래스에 대한 메타데이터(Namespace, ID, Comment, Label)를 정의할 수 있도록 하였다. ③은 선택된 Class의 Super Class(혹은 equivalent Class)를 정의하며, 다양한 형태의 anonymous



〈그림 5〉 Complex Class 편집 인터페이스

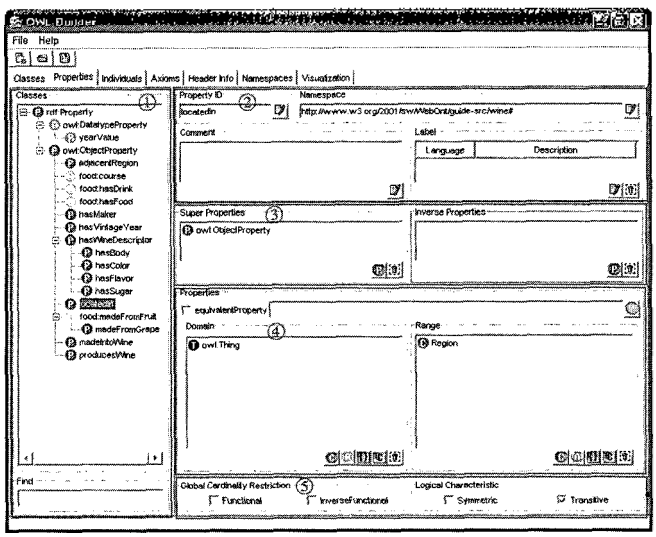
Class를 정의 할 수 있다. ④는 Restriction을 정의하는 부분으로 Cardinality, maxCardinality, minCardinality 등의 Cardinality 정보와 AllvaluesFrom, someValuesFrom, hasValue 등의 Property Restriction을 정의할 수 있도록 구성하였다.

union Class, intersection Class, complement Class 등의 Complex Class는 필요시 Complex Class 편집창을 통해 생성하게 된다. 〈그림 5〉는 Complex Class를 편집하기 위한 인터페이스이다. Complex Class 편집 인터페이스는 Tree 형태로 내부 요소들을 표현하여 세부 요소들의 편집이 용이하도록 구성하였다.

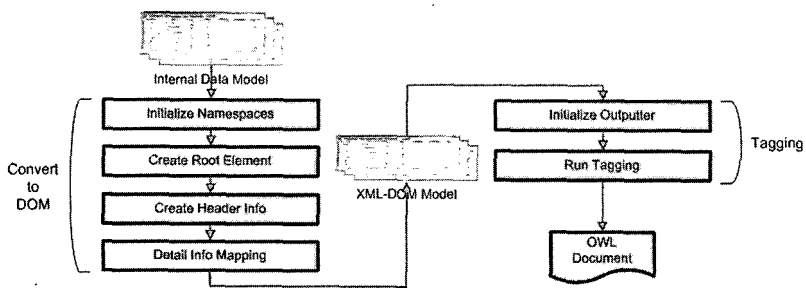
3.3.2 Property 정의 인터페이스

OWL의 Property는 기본적으로 rdf:Property의 Sub Property이며, 크게 owl:Datatype Property와 owl:ObjectProperty로 구분할 수 있다. owl:Datatype-Property는 속성 값으로 리터럴(Literal) 값을 가지며, owl:Object Property는 속성 값으로 Individual을 갖는다는 큰 차이점이 있다. 따라서 본 논문에서는 이를 이용하여 사용자 인터페이스에서 owl:Datatype Property는 Ⓓ로, owl:Object Property는 Ⓟ로, 그리고 다른 온톨로지에서 정의된 Property를 참조하여 사용하는 Imported Property는 Ⓘ로 표시하였다.

〈그림 6〉은 Property 생성을 위한 인터페이스를 보여준다. 인터페이스는 크게 ① Property Tree, ② Annotation 정의부, ③ Relation 정의부, ④ Domain & Range 정의부, ⑤ Property 속성 정의부로 구성하였다. ①은 모든 Property의 Sub Class 관계를 트리 형태로 표시하며, Property를 선택하여 우측 패널(②, ③, ④, ⑤)에서 세부정보를 수정할 수 있도록 구성하였다. ②는 Class 인터페이



〈그림 6〉 Property 생성 인터페이스



〈그림 7〉 직렬화 모듈의 처리 과정

스와 동일하며. ③은 선택된 Property의 Super Property, inverse Property를 정의할 수 있도록 하였다. ④는 해당 Property가 사용될 수 있는 Domain과 취할 수 있는 값을 제한하는 Range를 정의하며, Equivalent Property를 선택한 경우 선택된 Property와 동일한 Domain 및 Range를 갖는다. ⑤는 Property의 논리적인 특성(Functional, inverse Functional, Symmetric, Transitive)을 정의하며, 선택된 Property가 owl:ObjectProperty일 경우에만 가능하도록 하였다.

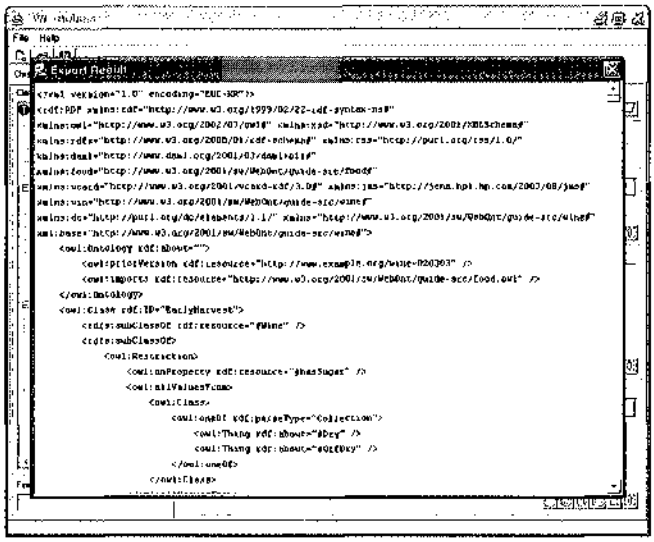
3.4 직렬화 모듈 설계 및 구현

직렬화 모듈은 편집된 내부 데이터 모델을 다시 텍스트 형태의 OWL 문서로 변환한다. 직렬화 모듈은 내부 데이터 모델의 각 요소들을 XML-DOM 모델로 변환하기 위한 메소드를 제공하도록 하였으며, 모든 요소들이 XML-DOM 모델로 변환되면 JDOM의 XMLOutputter 객체를 이용하여 최종적으로 텍스트 형식의 OWL 문서를 반환하도록 구현하였다. 〈그림 7〉은 직렬화 모듈의 처리 과정을 표현하고 있다.

직렬화 모듈은 내부 모델의 Class, Property, Individual, Axiom 등 각각의 요소들에 대해 XML 형태로 변화하기 위해 메소드를 제공하도록 구현하였다. 사용자의 Action에 의해 수정된 내부 데이터 모델을 저장하기 위해서는 직렬화 모듈을 통해 텍스트 형태의 OWL 문서로 변환해야 한다. 직렬화 모듈은 매칭 규칙을 통해 내부 데이터 모델을 XML-DOM 모델로 변환한다. 〈그림 8〉은 직렬화 모듈에 의해 변환된 OWL 문서를 export한 화면을 보여주고 있다.

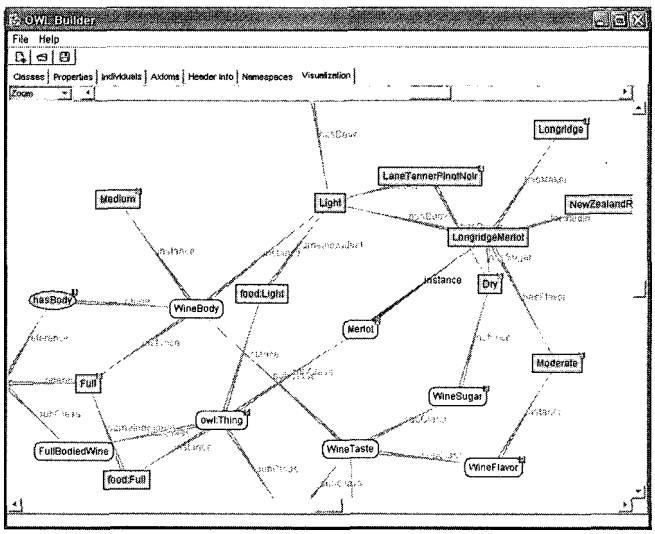
3.5 시각화 및 탐색 모듈 설계 및 구현

본 논문에서 제안한 저작 시스템의 시각화 및 탐색 모듈은 구축된 의미 정보들을 직관적으로 판독할 수 있도록 그래프 형태로 표현하고 객체들을 선택하여 탐색할 수 있는 기능을 제공해줌으로써 보다 효율적으로 온톨로지를 관리하고 분석할 수 있도록 하였다. 시각화 및 탐색 모듈에서 온톨로지의 의미 요소인 Class, Property, Individual을 각각의 노드



〈그림 8〉직렬화 모듈의 처리 결과

(Node)로 표현하며, 이들 간의 관계는 아크(Arc)로 표현한다. 시각화 및 탐색 모듈은 Graph Layout을 사용하여 구현하였다[15]. Visualization 탭이 선택되면 온톨로지에 구축된 의미 요소들을 이용하여 그래프를 초기화하며, 탐색 과정에서 선택된 노드와 인접한 노드를 중심으로 화면을 재구성하여 표시하도록 구성하였다. 또한 탐색 과정에서 자유로



〈그림 9〉 시각화 및 탐색 모듈의 실행 화면

운 Zoom-in, Zoom-out이 가능하도록 하였고, 전체 노드의 수에 따라 사용자가 직접 선택 노드와 몇 단계까지 연결된 노드를 표시할 것인지에 대한 locality를 선택할 수 있도록 하였다. 〈그림 9〉는 시각화 및 탐색 모듈의 실행 화면을 보여준다.

4. 비교분석

본 논문을 통해 구현된 OWL 저작 시스템은 OWL 온톨로지 문서를 처리하기 위한 기본적인 파서, 인터페이스, 직렬화 모듈을 제공할 뿐만 아니라, 구축된 온톨로지를 쉽게 관리, 분석 할 수 있도록 시각화 및 탐색 모듈을 제공하고 있다. 이러한 OWL 저작 도구의 기능을 바탕으로 기존 저작 도구와 비교 분석하였다. 〈표 2〉는 OWL 저작과 온톨로지 관리에 필요한 기능 명세와 그 기능 지원 여부를 정리한 것이다.

Protégé는 처음 의료분야의 지식 정보에 대한 온톨로지를 구축하기 위하여 시작된 온톨로지 저작 시스템으로 개발 초기에는 자체 모델을 사용하여 호환성이 떨어진다는 단점이 있었으나, 현재는 다양한 플러그인을 통하여 확장성을 제공하고 있다. 하지만 이러한 플러그인을 통한 확장성은 Protégé에서 사용하고 있는 내부 데이터 모델을 기반으로 하고 있고 Protégé의 온톨로지 모델에 대한 전문지식을 가지고 있어야 이를 활용할 수 있기 때문에 OWL을 위한 전용 편집 도구로 활용하기에 부적절하다. 또한 Protégé ezOWL Plugin의 경우 시각화 기능을 제공하고 있지만 Protégé OWL Plugin의 경우 시각화 기능을 제공하고 있지 않아서 OWL의 편집 및 분석이 용이하지 못하다. OilEd는 DAML+OIL을 기반을 두고 있어 DAML+OIL에 적합한 인터페이스로 고정되어져 있다. 그리고 온톨로지에 대한 시각화를 제공하고 있지 않아 복잡하고 거대한 온톨로지를 생성하는데 많은 어려움을 가지고 있다. OWL과 DAML+OIL의 문법이 유사하지만, OWL의 표현력이 훨씬 풍부하고 다양한 제약조건을 지정할 수 있어 OilEd를 확장하는데 많은 문제점이 따른다. 하지만 본

〈표 2〉 온톨로지 저작 시스템의 지원 기능 비교

	Protété OWL Plugin	Protété ezOWL Plugin	OilEd	OntoEdit	제안 시스템
모델 언어	OWL	OWL	DAML+OIL	자체 온톨로지	OWL
온톨로지 데이터 모델	기존 Protété 데이터 모델	기존 Protété 데이터 모델	DAML+OIL spec.	자체 온톨로지	OWL spec.
표현력	중	중	상	중	상
시각화	×	O	×	O	O
탐색 기능	×	×	×	O	O

논문에서 제안한 OWL 생성 시스템은 새로운 표준인 OWL을 위한 전용 저작 시스템으로써 효율적으로 온톨로지를 작성/편집할 수 있다. 또한 다른 도구와 비교하여 OWL의 표기법과 제약조건을 모두 수용하고 있어 온톨로지 표현력이 우수하고, 시각화 및 탐색 모듈을 지원함으로써 복잡한 온톨로지를 효과적으로 분석할 수 있다.

5. 결 론

현재 국내외에서 온톨로지를 구축하고 활용하기 위한 많은 노력들이 진행 중에 있으며, 시맨틱 웹을 위한 많은 프로젝트와 제품을 개발 중이다. 하지만 W3C에서 온톨로지 표현 언어로 제정하고 있는 OWL 기반의 온톨로지를 효율적으로 작성하고 편집할 수 있는 전용 저작 시스템은 아직 미흡한 실정이다.

이에 본 논문에서는 OWL 기반의 온톨로지 문서를 파싱하여 내부 데이터 내부 데이터 모델로 구축하고, 시각적인 GUI 인터페이스를 통해서 효율적으로 OWL을 생성/편집할 수 있는 저작 시스템을 설계 및 구현하였다. 제안된 시스템은 OWL에서 지원하는 모든 표현들을 수용하기 위해 Class, Property, Individual, Axiom, Header Info, Namespace에 따른 분리된 인터페이스를 제공하며, 이를 위해 각각에 대한 내부 데이터 모델을 설계하고 구현하였다. OWL 저작 시스템에서 사용하고 있는 내부 데이터 모델은 W3C의 온톨로지 표현 언어인 OWL 스펙의 분석을 통하여 설계하였기 때문에 풍부한 표현력과 호환성을 제공하는 장점을 가지고 있다. 또한 본 논문에서 제안한 그래프 기반의 시각화 및 탐색모듈은 구축된 의미요소들을 노드 앤 아크 다이어그램으로 표현함으로써 요소들 간의 관계를 보다 직관적으로 분석할 수 있으며, 탐색기능을 통해 의미들을 추적함으로써 보다 효율적인 온톨로지 관리가 가능하다. 그리고 W3C의 표준인 OWL을 기반으로 하였기 때문에 시맨틱 웹뿐만 아니라 다양한 온톨로지 분야에서 폭넓게 활용할 수 있을 것이다.

향후 연구과제로는 Topic Map과 같은 다양한 분야에서 서로 다른 언어로 작성된 온톨로지를 OWL로 변환하여 통합 할 수 있는 방안에 대한 연구가 진행되어야 할 것이다.

참 고 문 헌

[1] 이재호, "시맨틱 웹의 온톨로지 언어", 한국정보과학회 정보과학회지, 제21권 제3호 pp.18-27, 2003

[2] 김홍기, 김학래, 이강찬, 정지훈, 이재호 외, "월드와이드웹에서 시맨틱 웹으로", 마이크로소프트웨어 시맨틱 웹 특집, pp. 242-301, 2002

[3] Asunción Gómez-Pérez, Oscar Corcho, "Deliverable 1.3: A survey on ontology tools," OntoWeb, May, 2002.

[4] Berners-Lee, t., et al., "The Semantic Web," Scientific American, 2001.

[5] Comez-Perez, a. and Corcho, O., "Ontology languages for the Semantic Web," IEEE Intellignet Systems, Vol.17, No.1, pp.54-60, 2002

[6] D. Fensel, et al., "Semantic Web Application Areas," the 7th International Workshop on Applications of Natural Languages to Information system, Stockholm, Sweden, June, 2002.

[7] D. Fensel, F. Harmelen, M. Klein, H. Akkermans, "On-To-Knowledge: Ontology -based Tools for Knowledge Management," Proceeding of eBusiness and eWork 2000 Conference(EMMSEC 2000), 2000.

[8] Frank van Harmelen, Peter F. Patel-Schneider, and Ian Horrocks, Reference Description of the DAML+OIL (March 2001) Ontology Markuk Language, DAML +OIL Document, URL http://www.daml.org/2000/12/ reference.html, March, 2001.

[9] ezOWL, http://iweb.etri.re.kr/ezowl/

[10] Jeff Z. Pan, Ian Horrocks, "Metamodeling Architecture of Web Ontology Languages," In Proceedings of the Semantic Web Working Symposium, pages 131-149, Stanford, July, 2001.

[11] Jena, http://www.hpl.hp.com/semweb/jena2.htm

[12] OilEd, http://oiled.man.ac.uk/

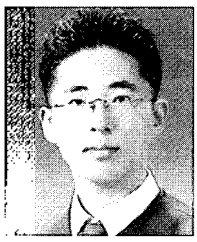
[13] OntoEdit, http://www.ontoprise.de/products/ontoedit_en

[14] Protégé OWL Plugin, http://protégé.stanford.edu/plugins/owl/

[15] TouchGraph, http://www.touchgraph.com/

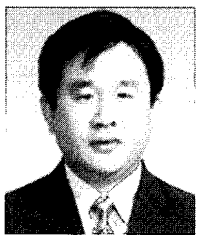
[16] York Sure, et al., "OntoEdit: Collaborative Ontology Development for the Semantic Web," Proceedings of the First International Semantic Web Conference on The Semantic Web, p.221-235, June 09-12, 2002.

저 자 소 개



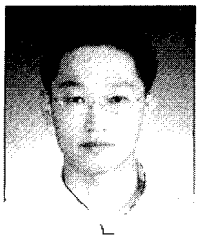
이무훈 (email : mhlee@dblab.hannam.ac.kr)
2002. 한남대학교 컴퓨터공학과(공학사)
2004. 한남대학교 컴퓨터공학과(공학석사)
2004 ~ 현재 한남대학교 컴퓨터공학과 박사과정
관심 분야 Semantic Web, Web Service, Web search engine, Data Stream Management System

조현규 (email : hkcho@etri.re.kr)
1988. 한국외국어대학교(학사)
1990. 고려대학교 대학원(경영학석사)
1997. 한남대학교 대학원(경영학박사)
1990 ~ 현재 한국전자통신연구원(ETRI) 지능형로봇연구단 지능형서비스플랫폼연구팀 책임연구원
관심 분야 Semantic Web, ebXML, Web Service, Intelligent Robot, Ubiquitous Computing



조현성 (email : hsc@etri.re.kr)
1995. 충남대학교 컴퓨터공학과(공학사)
1997. 충남대학교 컴퓨터공학과(공학석사)
1997 ~ 현재 한국전자통신연구원(ETRI) 지능형로봇연구단 지능형서비스플랫폼연구팀 선임연구원
관심 분야 Semantic Web, Web Service, Web Search, Intelligent Robot, Ubiquitous Computing

조성훈 (email : shcho@dblab.hannam.ac.kr)
2001. 한남대학교 컴퓨터공학과(공학사)
2003. 한남대학교 컴퓨터공학과(공학석사)
2004 ~ 현재 한남대학교 컴퓨터공학과 박사과정
관심 분야 RDF, Semantic Web, Ontology, ebXML, Web Service



장창복 (email : shcho@dblab.hannam.ac.kr)
2000. 한남대학교 컴퓨터공학과(공학사)
2002. 한남대학교 컴퓨터공학과(공학석사)
2002 ~ 현재 한남대학교 컴퓨터공학과 박사과정
관심 분야 Middleware, Active network, Context Modeling, Semantic Web, Software Security, OVPN

최의인 (email : eichoi@dblab.hannam.ac.kr)
1982. 한남대학교 계산통계학과(학사)
1984. 홍익대학교 전자계산학과(이학석사)
1995. 홍익대학교 전자계산학과(이학박사)
1985. ~ 1988. 공군 교육사 전산실장
1992. ~ 1996. 명지전문대학 전자계산과 조교수
1996 ~ 현재 한남대학교 컴퓨터공학과 교수
2003. UCLA visiting scholar
관심 분야 Web Service, Semantic Web, Web search engine, Ubiquitous Computing, Context-Aware Retrieval, Context Modeling, Data Stream Management System