시맨틱 웹에서 Preference를 위한 Annotation 기법

조 성 훈* 이 무 훈* 장 창 복* 최 의 인**

◈ 목 차 ◈

1. 연구배경

인터넷의 활성화로 인해 많은 기업들이 인터넷에 기반한 서비스에 관심을 갖기 시작했다. 이에 따라 국외의 주요 벤더들이 선도적으로 인터넷 기반의 서비스 지원을 위한 연구를 수행하여 많은 웹 서비스가 이루어지고 있다.

이러한 웹 서비스 기술은 W3C, OASIS 등의 표준화 단체 및 주요 IT 업체들의 주도하에 SOAP (Simple Object Access Protocol), WSDL(Web Service Definition Language), UDDI(Universal Description, Discovery and Integration) 등을 표준으로 자리 잡도록 하였다. 이와 같은 표준기술들은 웹 서비스를 사용자와 어플리케이션 간의 통신하도록 하기 때문에 웹 어플리케이션의 통합과 상호운용을 가능하게 하였다. 그러나 웹 서비스 검색이 단순 키워드 검색이나 서비스 분류정보만으로 서비스를 검색하기 때문에 현재의 사용자가 원하는 서비스를 정확히 찾기 위해서는 사용자의 개입이 필요하며 이를 통해 사용자의 요구사항(preference)을 바탕으로 일일이 검토해야 하는 문제를 가지고 있다. 이러한 문제를 해결하기 위하여 최근 시도되는 기술이 시맨틱 웹(Semantic Web)과의 연동이다.

시맨틱 웹은 의미 정의가 가능하기 때문에 자동화된 에이전트를 통해 정보의 의미와 정보간의 관계를 파악하고 이를 통해 정확한 정보의 검색과 추론을 통한 새로운 지식의 생성이 가능하다[1]. 이러한 시맨틱 웹과 시맨틱 웹을 이용한 웹 서비스를 현실화하기 위해서는 정보의 의미를 개념으로 정의하고 개념간의 관계를 정의 할 수 있는 온톨로지가 기반이 되어야 하며 사용자의 요구사항 또한 기계에 의해 의미적으로 해석될 수 있는 정보로 annotation될 필요가 있다[2,3].

따라서 본 논문에서는 사용자의 요구사항을 수집할 수 있는 인터페이스와 기계에 의해 의미적으로 해석될 수 있는 OWL 문서로 사용자의 요구사항을 annotation하기 위한 방법을 제시하고자 한다.

본 논문은 시맨틱 웹 서비스와 관련 사례를 살펴 본 뒤에 사용자 요구사항을 OWL을 이용하여 annotation할 수 있는 Preference Annotator의 설계하고 이에 대한 구성과 기능에 대해 기술한다. 그리고 기존 사례와의 비교하여 각각의 목적 및 적용범위에 대해 알아봄으로서 새로운 사용자 요구사항 annotation 기법에 대한 소개한다.

2. 시맨틱 웹 서비스

시맨틱 웹은 웹 자원에 대한 의미적(semantic) 처리를 가능하게 하는 기반 연구에 초점을 맞추고

* 한남대학교 컴퓨터공학과 박사과정
** 한남대학교 컴퓨터공학과 정교수

있으며, 웹 서비스는 웹 어플리케이션간의 자동화된 처리와 통합, 상호 운용성, 재활용성에 초점을 맞추고 있다. 즉 시맨틱 웹 서비스는 어휘에 포함되어 있듯이 의미와 서비스를 함께 처리할 수 있어야 한다. 의미의 처리는 웹 서비스에 대한 지식 기반 서비스 명세를 지원함을 의미한다. 시맨틱 웹 서비스 기반의 서비스 명세는 통합된 지식을 제공하는 서비스를 포함하며, 시맨틱 마크업 언어를 사용하여 콘텐트를 표현함으로써 웹 서비스의 자동화된 발견, 실행, 조합 모니터링을 가능하게 할 수 있다. 시맨틱 웹 서비스에서 서비스의 처리는 지식의 범위와 형태가 상이한 컨텐츠 사이의 상호운용과 추론을 지원할 수 있으며 지식의 통합을 위한 풍부한 기능을 제공할 수 있다. 대표적인 시맨틱 웹 서비스 연구로 미국의 DARPA 중심으로 진행중인 OWL-S(OWL Services)가 있다[4].

3. 시맨틱 웹 서비스 연구 사례

3.1 WSOM

WSOM(Web Services Outsourcing Manager)은 IBM의 AlphaWorks에서 진행한 프로젝트이며, 사용자 요구 사항을 기반으로 웹 서비스 플로우를 동적으로 조합할 수 있는 프레임워크이다. WSOM의 목적은 웹 서비스를 이용하여 비즈니스 프로세스를 자동으로 조합하는데 있다[5].

WSOM은 고객 요구사항을 분석 및 최적화하여 비즈니스 프로세스 아웃소싱을 위한 annotation 문서를 생성하며 서비스의 선택을 위해 검색 엔진과 최적화 알고리즘을 이용하여 고객을 위한 최적의 비즈니스 프로세스를 구성한다. 그 결과, 비즈니스 프로세스는 WSFL(Web Service Flow Language)이나 BPEL4WS(Business Process Execution Language for Web Service)와 같은 다른 웹 서비스 플로우 언어에 적합하게 구성한다.

WSOM은 고객의 비즈니스 요구사항을 얻기 위한 GUI 기반 요구사항 수집 툴과 Annotation을 위한 프로그램 기반 변환기를 이용한다. 이를 통해 획득된 요구사항 정보는 워크플로우, 선호도, 제약사항, 비즈니스 규칙에 대한 것이다. 요구사항 정보가 수집되면 웹 서비스 검색 엔진을 이용하여 검색 스크립트를 구성한다. 이 엔진은 UDDI 레지스트리와 WSIL(Web Service Inspection Language) 문서로부터 웹 서비스를 찾아 이용 가능한 서비스의 목록을 결과로서 제공한다. 그런 뒤에, 수집한 요구사항 정보는 최적의 비즈니스 프로세스를 조합하기 위해 비즈니스 프로세스 조합기를 유도하는데 사용되어진다. 일단 프로세스가 조합되었다면, 적절한 프로세스 플로우 모델링 언어로 변환되고 호출과 실행을 위해 사용자에게 전달된다.

그러나 WSOM은 비즈니스 프로세스를 생성하는데 필요한 웹 서비스를 단지 한가지로 한정하고 연구하였기 때문에 범용적인 사용에 있어서는 명확히 한계가 있다.

3.2 MWSAF

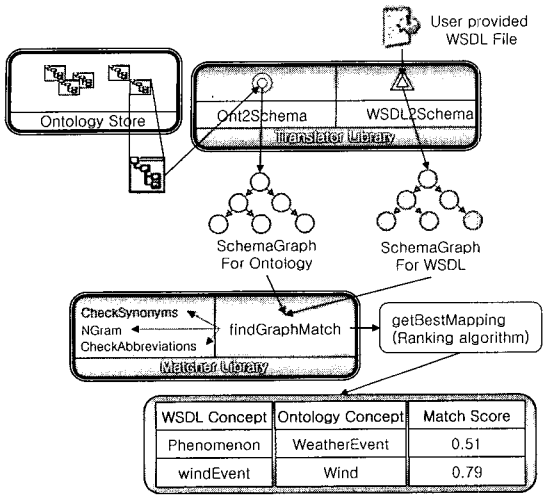
MWSAF는 Georgia 대학의 LSDIS Lab에서 진행 중인 METEOR-S 프로젝트로 일부로서 온톨로지(Ontologies)를 이용하여 기존의 웹 서비스 기술(Web service description)에 annotation하는 시맨틱 웹 기반의 어플리케이션이다[6,7].

MWSAF는 Ontology Store, Matcher Library, Translator Library로 구성된다. 그림 1은 MWSAF의 구조도이다.

Ontology Store는 이름과 같이 온토로지를 저장한다. 이 온톨로지들은 WSDL을 annotation하기 위해 시스템에 의해 이용될 수 있으며, 도메인들로 분류되어 있다.

Translator Library는 Schema Graph 표현을 생성하기 위해 사용되는 프로그램인 WSDL2 Grah와 Ontology2Grah로 구성되어 있다.

Matcher Libary는 두 가지 타입의 매칭 알고리즘(matching algorithm)을 통해서 WSDL과 온톨로지의 유사도를 비교하는데 사용된다. 그 중 하나는 element level matching algorithm으로 WSLD과 유사도, 동의어, 약어를 구분할 수 있다. 다른 하나



(그림 1) MWSAF의 구조

는 schema level matching algorithm으로 WSDL과 온톨로지의 개념들 간의 구조적 유사도(structural similarity)를 측정할 수 있다.

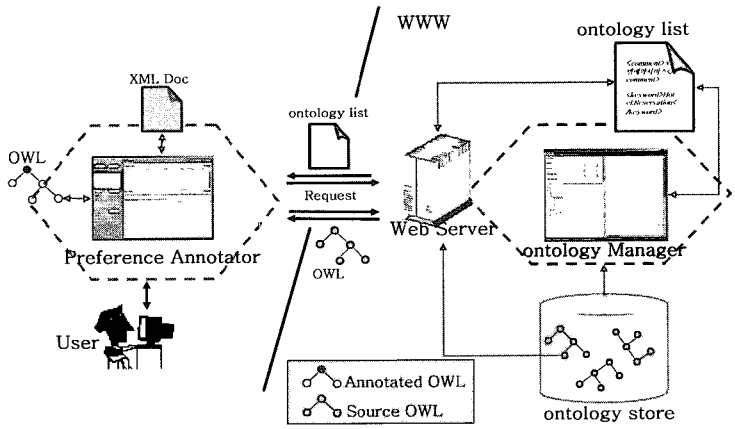
4. 사용자 요구사항 annotation 기법

웹 서비스를 이용하려는 사용자는 서비스에 대한 요구사항(preference)이 있다. 그러나 이러한 요구사항에 대한 의미 정의가 되지 않기 때문에 서비스 검색이 비효율적이라는 문제를 가지고 있다. 따라서 사용자 요구사항에 대한 의미 정의가 가능하다면 이러한 문제는 해결될 수 있을 것이다.

이를 위해 본 논문에서는 사용자 요구사항에 대한 의미 정의를 위해 온톨로지 기술언어인 OWL을 기반으로 요구사항을 annotation하는 기법을 적용한 Preference Annotator와 Preference Annotator에서 사용할 온톨로지를 관리할 수 있는 Ontology Manager를 개발하게 되었다.

4.1 Preference Annotator와 Ontology Manager의 구성

Preference Annotator는 OWL로 기술된 각각의 웹 서비스 온톨로지를 이용하여 사용자의 요구 사항 정보를 annotation한다. 여기서 웹 서비스 온톨로지는 웹 서비스에 대한 온톨로지로서 해당 서비스에 대한 정의가 기술되어 있다. 웹 서비스 온톨로지는 웹 서비스뿐만이 아니라 웹 서비스에 관련된 개념들을 모두 포함하기 때문에 방대하다. 또한 웹 서비스 온토롤지가 변경되거나 추가될 경우 이를 관리하기가 어렵다. 따라서 본 논문에서는 온톨로지를 위한 서버로 Ontology Manager를 설계하였다. 이 Ontology Manager는 웹 서비스의 온톨로지 문서인 OWL 문서에 대한 메타데이타(metadata)를 관리하고 사용자 요구 사항을 annotation할 때 해당 온톨로지를 제공하기 위한 것이다. 그림 2는 Preference Annotator와 Ontology Manager의



(그림 2) Preference Annotator와 Ontology Manager의 개념 구조

개념 구조를 나타낸다.

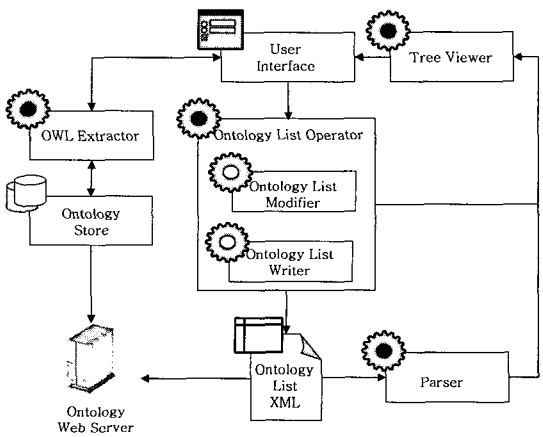
그림 2에 제시된 것처럼 Preference Annotator는 Ontology Manager의 클라이언트와 같이 동작하는데 이는 사용자 요구 사항을 annotation할 때 사용되는 OWL 문서를 Ontology Manager에 요청하고, 응답으로 받은 OWL 문서를 기반으로 사용자 요구 사항을 기술하기 때문이다.

4.2 Ontology Manager의 구성과 기능

Ontology Manager는 5개의 하위 모듈과 온톨로지 저장소, Ontology List 정보가 기록된 XML 파일 그리고 Ontology Web Server로 구성되며 이를 통해 OWL 문서를 관리하고 운영할 수 있도록 설계하였다. 각 모듈은 기능에 따라 분리되어 있지만, OWL 문서 관리를 위해 상호 작용하며, 웹 서비스에 대한 온톨로지 문서의 정보를 등록, 수정, 삭제할 수 있도록 설계하였다. 먼저 Ontology Manager의 구성에 대해 기술하고 각 모듈에 대해 알아보도록 하겠다. 그림 3은 개념적 구조를 표현한 구성도이다.

가. Ontology List Operator 모듈

웹 서비스를 위한 OWL 문서를 카테고리별로 분류 지정하고 해당 문서의 서비스에 관련된 키워드 및 문서의 URL 정보를 등록하는 기능을 수행하는 모듈이다.



(그림 3) Ontology Manager의 구조

사실상 가장 시맨틱 웹 서비스에 근접하기 위해서는 웹 크롤러(web crawler)처럼 웹 사이트 또는 웹 페이지를 방문하여 웹 서비스와 관련된 OWL 문서를 자동으로 검색, 파싱, 분류하여 해당 문서에 대한 온톨로지 레지스트리를 구축하여야 하지만, 아직까지 웹 서비스의 정보를 의미적으로 추출 및 분류하는 기술에 대한 한계로 인해 지원하지 못하고 있다.

나. Tree Viewer 모듈

사용자 인터페이스에 트리 인터페이스를 지원하기 위한 모듈이다. 이 모듈에서 사용하는 파서는 XML DOM 파서로서 각 Element 즉, Ontology_List, ontology, description, keyword, location, classification Element에 있는 데이터를 파싱하여 획득한다. 이 모듈은 사용자가 Tree View 인터페이스 상에서 선택한 ontology정보를 Tree로 제공함과 동시에 연관된 입력 텍스트 필드에 데이터를 출력한다. 이를 통해 Ontology Manager의 운영자가 쉽게 등록된 온톨로지 정보를 변경할 수 있게 된다.

다. Parser 모듈

Parser 모듈은 JAXP 1.0의 DOM 파서로서 주된 기능은 3 가지이다. 먼저 Ontology List Operator 모듈에서 Ontology List 파일을 생성한다. Tree Viewer 모듈에서 Tree 인터페이스를 구성하는데 필요한 데이터를 획득한다. 끝으로 Ontology Manager의 운영자가 온톨로지 정보를 변경할 때 XML 문서의 변경 사항을 적용하는 기능을 수행한다.

라. OWL Extractor 모듈

OWL Extractor 모듈은 파일 시스템을 검색하여 OWL 파일만을 필터링한 결과를 사용자에게 보여주고 사용자가 새로운 웹 서비스 온톨로지 파일을 등록할 때 파일시스템에서 선택한 OWL 파일을 파싱하여 웹 서비스 온톨로지에 대한 메타정보를 추출한다.

온톨로지에 대한 메타정보의 추출을 위해 Jena 2.1에서 지원하는 OntModel 클래스를 이용하였다.

OWL Extractor 모듈에서 필요한 정보를 추출하는 방법은 Ontology List에 keyword정보는 owl:Class, description정보는 owl:Ontology에 있는 rdfs:comment에 기술된 정보를 통해 추출한다. 그리고 classification정보는 기본 네임스페이스 정보를 기반으로 기준클래스를 지정한다. 기준클래스와 동일한 클래스가 owl:Class로 선언되어 있는지 검사하여 존재할 경우 rdfs:subClassOf에 rdf:resource 정보를 추출하여 classification 정보로 이용하게 된다. 만일 존재하지 않으면 기준클래스로 classification정보를 반환한다.

마. Ontology Web Server 모듈

이 모듈은 웹을 통해 두 가지 역할을 수행한다. 첫째는 Preference Annotator가 실행되면 Ontology List에 대한 XML 파일을 전송한다. 그리고 둘째는 Preference Annotator에서 요청한 웹 서비스 온톨로지에 대한 OWL 파일을 HTTP를 이용하여 전송한다.

바. Ontology Store

Ontology Store는 웹 서비스 온톨로지 파일이 classification 정보로 분류되어 저장되는 파일 시스템이다.

사. Ontology List

Ontology List는 XML을 기반으로 웹 서비스 온톨로지에 대한 메타데이타를 기술한다. 이를 위한 XML Schema는 그림 4와 같다.

```
<?xml version="1.0" encoding="euc-kr" ?>
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <xsd:element name="ontoloty_List" minOccurs="1" maxOccurs="1">
    - <xsd:complexType>
      - <xsd:sequence>
        - <xsd:element name="ontology" minOccurs="0" maxOccurs="unbounded">
          - <xsd:complexType>
            - <xsd:sequence>
                <xsd:element name="description" type="xsd:string" />
                <xsd:element name="keyword" type="xsd:string" />
                <xsd:element name="location" type="xsd:string" />
                <xsd:element name="classification" type="xsd:string" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" />
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
```

(그림 4) 웹 서비스 온톨로지의 메타데이타 기술을 위한 XML Schema 구조

4.3 Preference Annotator의 설계

4.3.1 Preference Annotator의 개요

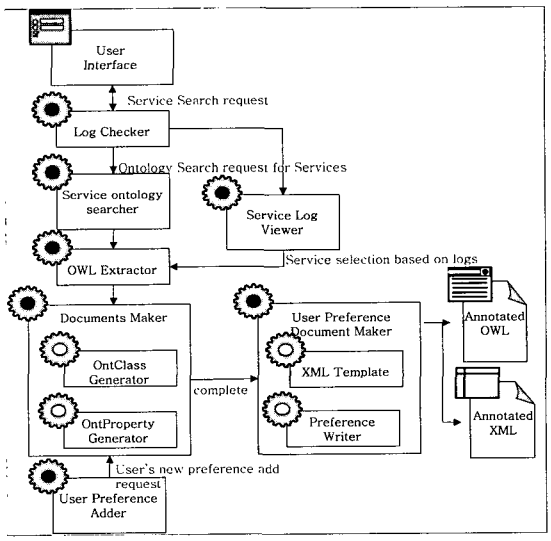
웹 서비스를 사용하려는 사용자는 자신이 원하는 서비스를 찾기 위해 자신의 의도가 포함된 정보를 이용하여 웹 서비스를 검색하게 된다. 그리고 사용자의 의도는 요구사항(preference)임과 동시에 서비스 검색의 기준이 된다. 이러한 요구사항은 특정 제품의 이름이나 서비스의 비용 등이 될 수 있다. 그러나 사용자가 원하는 선호사항은 언어가 가진 다의적인 특성과 범주로 인해 명확히 구분되지 못한다. 이러한 문제가 발생하는 것은 의미 정보가 고려되지 못하고 단어의 형태만 검색하였기 때문이다. 따라서 이 문제의 해결책은 검색 시 의미 정보를 포함시키는 것이다. 이를 위해 본 논문에서는 OWL을 기반으로 작성된 온톨로지를 통해 의미 정보를 기록하고 전달하기 위한 기법을 적용하여 Preference Annotator를 설계하게 되었다.

Preference Annotator의 구성과 처리도는 그림 5와 같다.

4.3.2 Preference Annotator의 구성과 기능

가. Log Checker 모듈

기존에 사용자가 사용한 요구사항 정보가 있는



(그림 5) Preference Annotator의 구성 및 처리도

지 검사하고 정보가 존재할 경우에 이를 바탕으로 새 서비스의 요구사항을 자동으로 구성해 주는 모듈이다. 기존에 사용한 요구사항 정보는 Preference Annotator로 요구 사항을 annotation할 때마다 기록된다. 본 논문에서는 이를 서비스 로그라 하며 XML을 기반으로 로그를 작성한다. 작성되는 로그에는 Service, Date, ServiceName, ServiceOWL가 있다. Service는 해당 로그가 하나의 서비스임을 나타내는 구분자로 볼 수 있다. Service 엘리먼트 내의 Date는 사용자가 서비스를 구성한 작성일자이며, ServiceName은 사용자가 선택한 서비스 명, ServiceOWL은 사용자의 서비스를 OWL로 annotated한 OWL 파일명을 나타낸다.

나. Service Log Viewer 모듈

Service Log Viewer 모듈은 Log Checker에서 기존에 사용자가 사용한 요구사항 정보가 있을 때 이를 사용자 인터페이스에 제시해 주는 역할을 수행한다. 기존 요구사항 정보는 사용자에 의해 채택될 수도 있고 새롭게 작성될 수도 있다.

다. Service Ontology Searcher

Service Ontology Searcher는 사용자가 서비스 받고자하는 서비스를 Ontology List 파일에서 검색하여 그 결과를 사용자 인터페이스의 에 전달하는 모듈이다. 이 모듈에서 사용하는 Ontology List 파일은 Ontology Manager의 운영자에 의해 작성된 OWL 정보 파일이다. 그림 6은 Ontology List 파일의 예이다.

그림 6에 제시된 바와 같이 Ontology List 파일은 ontology 엘리먼트가 하나의 OWL 파일을 기술한 것이다. 각 엘리먼트의 의미는 다음과 같다. description은 등록된 온톨로지에 대한 주석이고, keyword는 온톨로지에 대한 키워드이다. location은 등록된 온톨로지가 있는 URL 정보를 기술한다. classification는 온톨로지의 분류 정보를 의미한다.

```
<ontology name="Transport.owl">
  <description>Transport reservation ontology</description>
  <keyword>Train, AirPlane, Transport, Grade, Taxi, AirPlanSeperate</keyword>
  <location>http://khkim.hannam.ac.kr/OntologyStore/Transport/Transport.owl</location>
  <classification>Transport</classification>
</ontology>
```

(그림 6) Ontology 리스트 파일

라. OWL Extractor 모듈

온톨로지화된 웹 서비스 정보는 OWL Concept으로 정의된다. 즉, owl:Class나 Property로 정의되는 것이다. 따라서 서비스는 owl:Class로 정의된다.

본 논문에서는 이렇게 서비스를 owl:Class로 정의하는 점에 착안하여 owl:Class를 검색하고 owl:Class에 기술된 서비스 정보를 추출하여 사용자가 선택할 수 있도록 하기 위해 OWL Extractor 모듈을 설계하였다.

이 모듈에서 처리하는 것은 OWL 문서이기 때문에 OWL 문서를 read하고 구조를 검색할 수 있는 Parser와 Reader가 필요하다. 이에 따라 본 논문에서는 Jena 2.1에서 지원하는 OntModel 클래스를 사용한다.

OWLExtractor 모듈에서 수행하는 owl:Class 추출 처리는, 먼저 OntModel을 생성하고 처리할 OWL 문서를 Ontology Manager에 질의하여 다운로드한다. 다운로드에 필요한 URL 정보는 사용자 인터페이스에서 선택한 OWL 문서의 location 엘리먼트의 text를 서비스 개념 추출 요청 시 이 모듈에 전달된다. 이를 통해 해당 URL에 HTTP 프로토콜을 이용하여 접속하고, 접속이 성공적으로 이루어지면 해당 OWL 문서를 OntModel로 read한다. 그런 뒤에 읽혀진 OWL 문서를 Preference Annotator가 있는 시스템에 OWL 문서로 저장한다. 저장된 OWL 문서는 다시 OntModel로 read하여 사용자가 선택한 온톨로지 정보를 가지게 한다. 그런 뒤에 OntModel에 정의된 모든 owl:Class 선언 정보에 대해 질의를 수행한다. 질의문을 통해 OntModel에서 owl:Class 선언 구문에 대한 리스트를 받게 되고 그 결과는 사용자 인터페이스에 전달되게 된다.

마. Documents Maker 모듈

Preference Annotator는 사용자의 요구사항을 입력으로 하여 OWL문서를 생성한다. 온톨로지 내에 개념과 개념 간 관계, 속성이 기술되기 때문에

```
<?xml version="1.0" encoding="euc-kr" ?>
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <xsd:element name="ServiceLog" minOccurs="1" maxOccurs="1">
    - <xsd:complexType>
      - <xsd:element name="service" minOccurs="0" maxOccusr="unbounded">
        - <xsd:sequence>
            <xsd:element name="Date" minOccurs="1" maxOccurs="1" type="string" />
            <xsd:element name="ServiceName" minOccurs="1" maxOccurs="1" type="string" />
            <xsd:element name="ServiceOWL" minOccurs="1" maxOccurs="1" type="string" />
          - <xsd:element name="Preference" minOccurs="1" maxOccurs="1">
            - <xsd:element name="PreferenceType" minOccurs="1" maxOccurs="unbounded">
              - <xsd:complexType>
                - <xsd:sequence>
                    <xsd:element name="PreferenceTypeName" type="xsd:string" />
                    <xsd:element name="PreferenceTypeValue" type="xsd:string" />
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:element>
          </xsd:sequence>
        </xsd:element>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
```

(그림 7) 사용자 요구사항 정보를 표시하기 위한 XML Schema

OWL 문서의 생성은 각각을 구분하여 생성할 수 있어야 한다. 따라서 Document Maker 모듈도 각각을 구분하여 처리할 수 있도록 Class Generator, Property Generator 모듈을 설계하였다. Class Generator 모듈은 한 서비스에 대해 하나의 OWL 문서를 생성한다. 하나의 서비스는 대개의 경우 한 owl:Class로만 구성되지 않는다. 대부분 여러 owl:Class와 연결되어 owl:Class가 정의되기 때문에 한 owl:Class로만 되는 경우는 드물다. 실제 온톨로지의 정의를 보면, 개념을 정의하기 위해서 다른 개념과의 관계를 통해 정의된다. 이에 따라 본 논문에서는 사용자가 선택한 서비스의 의미를 보존하기 위하여 관련된 owl:Class도 같이 OWL 문서에 추가하도록 구성하였다. 따라서 사용자가 선택한 사용자 요구사항의 의미는 보존되게 된다. Property Generator 모듈은 OWL로 기술되는 서비스의 제약사항들을 OWL에 포함된 속성 표현으로 변환하여 OWL 구문으로 생성하는 모듈이다.

바. User Preference Adder 모듈

이 모듈은 사용자가 원하는 서비스 요구사항을 OWL 문서에서 찾을 수 없을 때 사용자가 직접 자신의 요구사항을 생성하는데 사용되는 모듈이다.

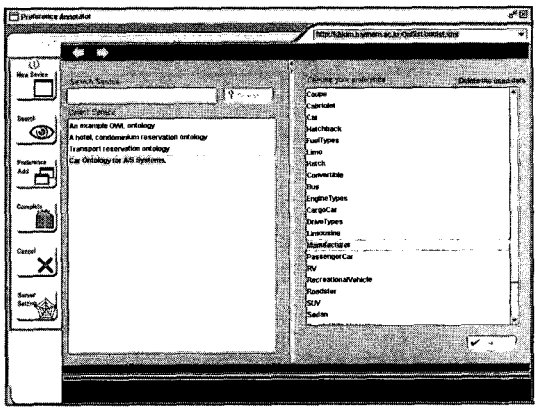
User Preference Adder는 Annotated OWL 문서에 DatatypeProperty 속성으로 사용자의 추가적인 요구사항을 생성한다.

사. User Preference Document Maker 모듈

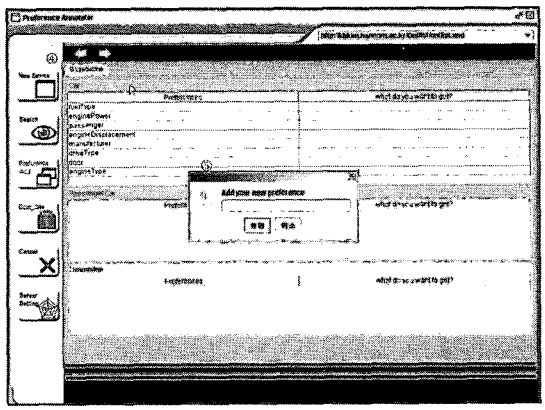
이 모듈은 사용자 요구사항을 Annotated OWL과 XML 파일로 생성하는 모듈이다.

Annotated OWL 파일은 사용자가 원하는 서비스가 포함된 OWL 파일에서 사용자에 의해 선택된 owl:Class와 그 owl:Class의 의미(semantic)를 유지하는데 필요한 관련 owl:Class와 Property에 대한 OWL 구문을 포함하고 있다. 다시 말하자면, 서비스의 의미를 보장하기 위한 구문들을 기록하고 있다는 것이다. 따라서 사용자의 서비스를 포함했던 초기의 OWL 파일과는 다르다. 특이한 사항으로는 Annotated OWL가 owl:Class의 individual을 포함하지 않는다는 점이다. 이는 owl:Class에 사용자가 User Preference Adder를 이용하여 새로운 요구사항을 추가하였을 경우에 owl:Class 내에 owl:minCardinality라는 제약 사항이 추가되게 된다. 따라서 owl:Class의 individual은 이 제약 사항을 만족하여야만 individual이 되게 된다. 이는 기존의 individual들은 추가된 제약 사항으로 인해 모두 유효하지 않게 된다는 의미이다. 그러므로 Annotated OWL 파일에는 individual이 포함될 수 없다.

그러나 사용자의 요구사항 정보의 이름만을 기록할 수 있지 실제 데이터는 유지하지 못한다. 이를 위해 데이터 유지를 위한 XML 파일을 설계하였다. 이 XML 파일은 추가사항을 정확히 기술하기 위해 XML Schema에 기반하여 작성된다. XML



(그림 8) 사용자 인터페이스 (1)



(그림 9) 사용자 인터페이스 (2)

Schema는 그림 7과 같으며 각 엘리먼트의 의미는 살펴보면, ServiceLog는 사용자 로그의 Root 엘리먼트, service는 서비스를 구분하는 엘리먼트, Date는 사용자 요구사항의 작성일자, ServiceName은 사용자가 선택한 서비스명, ServiceOWL은 사용자가 선택한 서비스와 해당 서비스에 대한 요구사항을 annotated한 OWL 문서, Preference는 사용자의 요구사항을 지정하기 위한 엘리먼트, PreferenceType은 사용자 요구사항의 이름과 속성값을 포함하는 엘리먼트, PreferenceTypeName는 사용자 요구사항의 이름, PreferenceTypeValue는 사용자 요구사항의 실제 값을 의미한다.

4.3.3 사용자 인터페이스

사용자 인터페이스는 사용자가 서비스를 검색하여 해당 서비스의 요구사항을 쉽게 지정할 수 있도록 구성하였다. 그림 8과 그림 9는 사용자 인터페이스이며 ① 메뉴, ② 검색부, ③ 서비스 선택부, ④ 요구사항 제시부, ⑤ 요구사항 추가부로 구성되어 있다.

메뉴는 사용자의 서비스 요구사항을 입력하는데 필요한 메뉴를 구성하여 놓은 것이다. 메뉴의 [New Service]는 새로운 서비스를 생성한다. 이 메뉴와 연결된 인터페이스는 그림 8의 ② 검색부이다.

검색부는 사용자가 원하는 서비스명 또는 유사한 키워드를 입력받아 Ontology Manager에서 초기 다운로드한 Ontology List 파일에서 매칭되는 서비스를 찾는다. 검색된 결과는 그림 8의 ③ 서비스 선택부에 선택 가능한 온톨로지로 제공된다. 매칭되는 서비스가 없으면 어떤 정보도 표시하지 않는다. 서비스 선택부는 검색부는 검색된 결과를 받아 리스팅한다. 이때 리스팅된 온톨로지는 OWL 문서의 이름이 아닌 OWL 문서의 설명으로 제공된다. 사용자가 원하는 서비스 온톨로지가 있을 경우, 해당하는 정보를 클릭하면 그 온톨로지에 저장된 owl:Class 정보를 추출하여 서비스 선택부로 전송한다. 사용자가 특정 서비스 항목을 선택하고 ③의 [Select] 버튼을 선택하면 해당 서비스의 하위 속성 즉, 설정 가능한 요구사항의 속성명이 그림 9의 ④ 요구사항 제시부에 리스트로 추가된다.

요구사항 제시부는 위에서 기술한 것처럼 ③ 서비스 선택부에서 선택한 서비스와 관련된 owl:Class를 리스팅한다. 이 인터페이스의 경우 OWL Concept을 추출할 때 해당 owl:Class의 인스턴스를 같이 추출하여 사용자가 팝업(popup) 메뉴를 통해 쉽게 입력할 수 있도록 한다.

Preference Adder Part는 그림 9의 ⑤에 해당하고 사용자가 추가적인 사용자 요구사항을 입력하고자 할 경우에 추가사항에 대한 항목을 추가하여 요구사항을 Annotated OWL 문서에 적용할 수 있도록 지원하는 인터페이스이다.

5. 기존 연구와의 비교

본 논문의 Preference Annotator는 사용자의 요

(표 1) 기존 연구와의 비교

	목적	적용대상	annotation 기법
PA	사용자 요구사항 annotation	사용자	온톨로지
WSOM	비지니스 프로세스의 자동 구성	비지니스 프로세스	온톨로지
MWSAF	웹 서비스 기술	서비스	온톨로지

구사항을 annotation함으로서 웹 서비스 검색 시 사용자가 원하는 서비스를 효율적으로 결정할 수 있도록 하기 위한 것으로서 기존의 WSOM과 MWSAF와는 다소의 차이점을 가진다. 표 1은 WSOM, MWSAF, Preference Annotator 간의 차이점을 비교한 것이다.

표 1에서 알 수 있듯이 본 연구와 WSOM, MWSAF 모두 온톨로지를 기반으로 annotation을 한다. 그러나 각각의 적용 대상과 목적 면에서 차이가 있다. Preference Annotator는 사용자의 요구사항을 annotation하는 것이 목적이면서 적용대상이지만 WSOM은 비즈니스 프로세스, MWSAF는 웹 서비스 차제를 annotation한다. 각각의 적용 대상과 제시된 목적은 다르지만 궁극적인 목표는 동일하다고 볼 수 있다. 즉, 기존 웹 서비스에 시맨틱을 부여함으로서 개선된 웹 서비스 환경의 구축이 그 목표인 것이다.

6. 결 론

현재의 웹 서비스는 서비스 각각의 의미 부재라는 문제로 한계에 봉착해 있는 실정이다.

이를 위해 본 논문에서는 효과적으로 웹 서비스 검색이 가능토록 사용자 요구사항을 웹 온톨로지 표준인 OWL로 annotation하여 의미를 부여하고 이를 통해 웹 서비스를 이용하는 방법을 제안하였다.

본 논문에서 개발한 Ontology Manager는 웹 서비스 온톨로지에 대한 관리와 분류를 수행할 수 있으며 Preference Annotator는 전문 지식없이도 사용자의 요구사항을 annotation 할 수 있다. 그러나 웹 서비스에 대한 온톨로지화가 보편적이지 않은 상황이기 때문에 이에 대한 활용은 아직 활성화될 수 없다는 문제를 가지고 있다. 웹 서비스의 온토로지화가 보편화된다면 본 논문의 Preference Annotator는 사용자 측면에서 유용하게 활용될 수 있을 것이다.

참 고 문 헌

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

[2] Deborah L. McGuinness and Frank van Harmelen, "OWL Web Ontology Language Overview," 2004, http://www.w3.org/TR/owl-features/.

[3] Peter F. Patel-Schneider, et al., "OWL Web Ontology Language Semantics and Abstract Syntax," 2004, http://www.w3.org/TR/ owl-semantics/.

[4] David Martin, et al., "OWL-S 1.0 Release", 2004, http://www.daml.org/services/owl-s/1.1B/owl-s/owl-s.html.

[5] Abhijit Patil, et al., "METTOR-S Web service Annotation Framework", 2004, http://lsdis.cs.uga.edu/Projects/METEOR-S

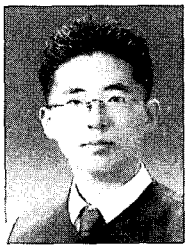
[6] Kunal Verma, et al., "METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web", http://lsdis.cs.uga.edu/Projects/METEOR-S

[7] http://alphaworks.ibm.com/tech/WSOM

◐ 저 자 소 개 ◑

조 성 훈

2001년 한남대학교 컴퓨터공학과 졸업(학사)
2003년 한남대학교 컴퓨터공학과 졸업(석사)
2004년~현재 한남대학교 컴퓨터공학과 박사과정
19○○~현재 △△대학 △△학과 교수
관심분야 : 데이터베이스, 시맨틱 웹, 온톨로지, 정보 검색
E-mail : shcho@dblab.hannam.ac.kr

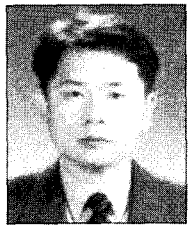


이 무 훈

2002년 한남대학교 컴퓨터공학과 졸업(학사)
2004년 한남대학교 컴퓨터공학과 졸업(석사)
2004년~현재 한남대학교 컴퓨터공학과 박사과정
관심분야 : 시맨틱 웹, 웹 서비스, 정보 검색
E-mail : mhlee@dblab.hannam.ac.kr

장 창 복

2001년 한남대학교 컴퓨터공학과 졸업(학사)
2003년 한남대학교 컴퓨터공학과 졸업(석사)
2003년~현재 한남대학교 컴퓨터공학과 박사과정
관심분야 : 데이터베이스, 시맨틱 웹, 정보 검색, 전자상거래
E-mail : chbjang@dblab.hannam.ac.kr



최 의 인

1982년 숭전대학교 계산통계학과(학사)
1984년 홍익대학교 전자계산학과(석사)
1995년 홍익대학교 전자계산학과(박사)
1985년~1988년 공군 교육사 전산실장
1992년~1996년 명지전문대학 전자계산과 조교수
1996년~현재 한남대학교 컴퓨터공학과 정교수
관심분야 : 실시간 데이타베이스, 주기억 데이타베이스, 클라이언트/서버 데이터베이스
E-mail : eichoi@dblab.hannam.ac.kr