시맨틱 웹 서비스를 위한 서비스 온톨로지의 자동 생성

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

Automatic Generation of Service Ontology for Semantic Web Services

Jin-Hyuk Yang, In-Jeong Chung
Dept. of Computer Science, Korea University

요 약

본 논문에서는 OWL-S 서비스 온톨로지를 자동으로 생성하는 방법에 대한 연구결과를 제공한다. 자동생성을 위하여 UML 클래스 다이어그램 및 상태차트 다이어그램을 XMI 파일들로 변환한 후 원자 서비스 및 속성들에 대한 정보와 복합 서비스 조합에 대한 정보를 각각 추출한다. 추출된 정보는 UML 상태차트 다이어그램 구성 요소들과 OWL-S 복합 서비스를 위한 구조물들 사이의 매핑 규칙들을 통하여 XSLT 응용에서 OWL-S 서비스 모델 온톨로지를 자동으로 생성시키는데 사용된다. 생성된 온톨로지의 타당성 검증을 위해서 이용 가능한 여럿의 유효성 검사를 수행하였다. 우리의 방법론은 자동적, 효과적 및 일반적일 뿐만 아니라 서비스 온톨로지 생성자인 개발자들에게 매우 친숙한 환경에서 수행된다는 장점들을 가진다.

1. 서론

인터넷의 역할이 정적인 자원뿐만 아니라 서비스와 같은 동적인 자원의 근원으로서의 역할이 중요하게 됨에 따라 웹 서비스와 같은 기술이 중요하게 되었다. 그러나 웹 서비스 근간 기술들인 WSDL(Web Service Description Language)[1], UDDI(Universal Description, Discovery, and Integration)[2], SOAP(Simple Object Access Protocol)[3] 들을 이용해서는 서비스의 지능적인 활용에 제한이 따른다. 예로 WSDL 은 서비스의 입출력에 대한 정보만을 나타내므로 복합 서비스의 기술을 표현할 수 없고, UDDI 는 키워드 검색형 인터페이스만을 제공하므로 지능형 웹 서비스(또는 시맨틱 웹 서비스[4]라고도 불림)를 실현하는데 많은 제약을 가진다. 이에 웹 서비스 커뮤니티에서는 BPEL4WS(Business Process Execution Language for Web Services)[5], W3C 에서는 WSCI(Web Service Choreography Interface)[6], 그리고 시맨틱 웹 커뮤니티에서는 OWL-S(OWL ontology for services)[7]와 같은 스펙과 표준들을 발표하고 있다. 그 중에서 우리는 OWL-S 를 시맨틱 웹 서비스 마크업으로 선택하였다. OWL-S 는 OWL(Web Ontology Language)[8]로 작성된 서비스 온톨로지로서 추론을 가능하게 할 뿐만 아니라 웹 서비스를 보다 지능적으로 활용하는 것을 허용한다. 기타 스펙들과의 관계에 대한 사항들은 [9]에서 더 자세히 살펴볼 수 있다.

본 논문에서는 OWL-S 서비스 온톨로지를 자동으로 생성하는 방법에 대한 연구결과를 제안한다. 특히 우리는 OWL-S 를 구성하고 있는 세 개의 온톨로지들(서비스 프로파일, 서비스 모델 및 서비스 그라운딩) 중 서비스 모델에 주안을 둔다. 왜냐하면 지능형 웹 서비스를 가능하게 하기 위해서 제일 중요한 정보는 서비스가 다른 서비스와 어떻게 상호작용할 수 있는지에 대한 정보인데 바로 이러한 정보가 서비스 모델 온톨로지에서 기술되기 때문이다.

자동적인 변환을 위하여 우리는 먼저 서비스에 대한 UML(Unified Modeling Language) 클래스 다이어그램 및 상태차트 다이어그램들에서 각각 원자 서비스 및 그것의 IOPE(Input, Output, Precondition and Effect)들에 대한 정보와 복합 서비스 조합에 대한 정보를 각각 추출한다. 우리가 UML 다이어그램들에서 필요한 정보를 추출하는 접근법을 선택하는 이유는 UML 이 개발자들에게 가장 친숙한 표준 설계수단이기 때문이다. 추출된 정보는 3 장에서 정의된 매핑들을 통하여 XSLT(Extensible Stylesheet Language Transformations) 응용에서 OWL-S 서비스 모델 온톨로지를 자동으로 생성시키는데 사용된다. 생성된 온톨로지의 타당성 검증을 위해서 W3C 및 기타 사이트에서 제공된 유효성 검사를 수행하였다.

우리의 서비스 온톨로지 생성 방법론은 완전자동이고 일반적일 뿐만 아니라 서비스를 생성하는 개발자에게 OWL-S 의 이해를 강요하지 않는 매우 친숙한 환경을 제공한다. 또한 제약조건의 명세를 OCL(Object Constraint Language)과 같은 복잡한 언어를 이용하는 대신 GUI(Graphic User Interface)를 이용하여 조건을 명시할 수 있는 방법을 제공하였다.

본 논문의 구성은 다음과 같다. 먼저 2 장에서는 온톨로지 생성에 관한 기존 연구들을 살펴보고 3 장에서는 서비스 온톨로지의 자동생성 방안을 기술한다. 4 장에서는 예제 시나리오를 통한 구현을 보이고 마지막으로 5 장에서 결론을 기술한다.

2. 관련연구

2.1 시맨틱 웹 서비스와 OWL-S

지능형 웹 서비스라고도 종종 불리는 시맨틱 웹 서비스는 [4]에서 처음으로 소개된 것으로서 시맨틱 웹에서 중요한 역할을 수행하는 온톨로지를 이용하여 웹 서비스의 지능화를 가능하게 한다. 시맨틱 웹 서비스를 가능하게 하기 위한 메타데이터를 기술하는 언어들 중 OWL-S 는 OWL 로 작성되는 서비스 온톨로지이다. OWL-S 는 자동적인 서비스의 발견, 실행, 조합 및 상호운용을 목표로 한다. 이를 위하여 OWL-S 온톨로지는 3 개의 하부 온톨로지들을 이용하여 서비스에 대한 메타데이터를 기술한다. 먼저 서비스 프로파일 온톨로지는 서비스의 자동적인 서비스의 발견을 위한 것으로서 서비스의 IOPE 정보를 포함한다. 둘째로 서비스 모델 온톨로지는 서비스의 행동에 관한 조건들 및 제약사항들을 포함하고 있다. 마지막으로 서비스 그라운딩 온톨로지는 서비스에 접근하는 방법에 관한 정보를 기술한다.

2.2 온톨로지 생성 관련연구들

온톨로지 사용의 주요 아이디어는 자원에 대한 메타데이터를 이용하여 추론을 통한 지능화 및 자동화를 추구하는 것이다. 따라서 온톨로지를 생성하는 것은 주요한 연구분야이다. 그러나 온톨로지를 생성하는 작업은 매우 번거롭고 시간이 많이 소요되는 작업이므로[10] 온톨로지를 자동적으로 또는 효과적으로 생성하는 방안은 매우 중요하다.

여럿의 온톨로지 생성방법들 중 흥미로운 접근법은 UML 을 이용하여 온톨로지를 모델링 및 생성하는 것인데 [11], [12] 및 [13]과 같은 것들이 있다. 그러나 이러한 방법들은 모두 클래스 다이어그램을 이용하여 도메인 온톨로지를 설계 및 생성하는 것으로서 행동을 가지는 서비스 온톨로지를 모델링하는 데에는 부적합하다. 자세한 이유는 3.2 절에서 언급한다.

한편 서비스 온톨로지의 생성과 관련된 문헌은 WSDL 문서에 주석처리를 하여 OWL-S 온톨로지를 반자동으로 생성하는 [14]와 같은 연구가 존재한다. 그리고 [15]에서는 UML 액티브티 다이어그램을 이용하여 BPEL4WS 서비스를 생성하는 방법론을 언급한다. 그러나 [14]는 반자동이므로 개발자를 성가시게 하고 [15]는 우리가 UML 상태차트 다이어그램에서 OWL-S 서비스 온톨로지를 생성한 것과 같은 접근법을 취하고 있으나 UML 액티브티 다이어그램 구성요소들과 BPEL4WS 구성요소들 사이의 매핑정의가 기본적인 수준에서만 언급되었다.

3. 서비스 온톨로지의 자동생성

3.1 동기 및 설계 원칙들

지능형 웹 서비스의 실현을 위해서는 서비스 온톨로지들이 먼저 생성이 되어야만 한다. 그러나 서비스 온톨로지의 생성에 소요되는 시간 또는 비용이 높다면 온톨로지의 번창에 걸림돌이 될 것이다. 따라서 서비스 온톨로지의 생성은 자동적으로 그리고 효과적으로 생성되는 것이 바람직하다. 그러나 [16] 및 [17]과 같은 기존 도구들은 다양한 기능성을 제공하지만 서비스 온톨로지의 생성을 위하여 도구의 사용을 요구할 뿐만 아니라 그 도구의 사용은 OWL-S 의 이해를 전제로 하고 있다는 것에 주의할 필요가 있다. 이에 우리는 다음과 같은 두 가지 설계 원칙들을 결정하였다.

- 첫째 서비스 온톨로지는 자동으로 생성되어야만 한다.
- 둘째 서비스 개발자들에게 친숙하고 쉬운 방법이어야만 한다.

우리는 상기 두 가지 설계원칙들을 반영하기 위하여 다음과 같은 방법들을 구상하였다.

먼저 자동적인 변환을 위하여 XSLT 응용을 고려하였다. XSLT 는 XML 파일을 입력으로 받아 다양한 타입의 문서들을 생성할 수 있다. 우리가 서비스 온톨로지의 자동생성을 위하여 XSLT 를 결정한 이유는 두 번째 설계원칙에서 결정한 UML 방법론에서 생성되는 서비스 모델에 대한 정보가 XMI(XML Metadata Interchange)로 저장되는데 이것이 바로 XML 문서이기 때문이다. XMI 는 OMG 에서 제안한 메타데이터 교환 표준이다.

다음으로 개발자들에게 친숙하고 용이한 환경을 제공하기 위해서 우리는 소프트웨어 공학에서 가장 널리 사용되는 GUI 표준인 UML 을 결정하였다. 우리의 주요 생각은 서비스 분석 및 설계시 사용되는 여럿의 UML 다이어그램들에서 서비스 온톨로지에 필요한 정보를 추출하는 것이다. 이를 위하여 개발자들이 서비스의 생성과정에서 만든 UML 다이어그램들을 XMI 로 추출하고 이를 XSLT 응용을 이용하여 자동으로 서비스 온톨로지를 생성하는 것이 우리의 아이디어이다.

3.2 OWL-S 복합 서비스 구성을 위한 구조물들과 UML 상태차트 구조물들 사이의 매핑정의

본 절에서는 우리가 주안을 두고 있는 OWL-S 서비스 모델 온톨로지를 자동으로 그리고 효과적으로 생성하는데 사용되는 XSLT 응용에 필요한 규칙들을 살펴본다. 이러한 규칙들은 UML 다이어그램들에서 사용되는 구조물들과 OWL-S 의 구조물들 사이의 매핑들에 기반한다.

우리는 서비스 온톨로지를 생성함에 있어 IOPE 를 중심으로 하는 속성들을 생성하는 부분과 복합 서비스의 구성에 관한 정보를 추출하는 부분으로 나누어서 고려하였다. 다시 말해 UML 클래스 다이어그램에서 서비스의 속성들에 관련된 정보를 추출하고 UML 상태차트 다이어그램에서 복합 서비스의 구성에 관한 정보를 추출한다. 서로 다른 다이어그램들에서 각기 다른 정보를 추출하는 이유는 클래스 다이어그램이 속성들에 관한 정보와 다른 클래스와의 관계를 기술하기에 좋은 반면에 복합 서비스들로 구성된 서비스를 표현할 수 없는 문제점을 가지고, 상태차트 다이어그램은 서비스의 행동방법을 기술하기에 좋은 표현 수단일 뿐만 아니라 복합 서비스로 구성된 복합 서비스를 표현할 수 있는 장점을 가지지만 서비스의 속성들을 기술할 수 없는 문제점을 가지기 때문이다.

서비스에 대한 클래스 다이어그램에서 OWL-S 서비스 모델 온톨로지에 사용되는 정보는 단지 IOPE 와 같은 속성들에 대한 정보이므로 매핑은 간단하게 정의될 수 있다. 따라서 상태차트 다이어그램과 OWL-S 서비스 모델 온톨로지의 관계에 대한 매핑정의를 기술한다.

OWL-S 의 Sequence 구조물은 서비스들 사이의 순서를 명시하는 구조물이므로 상태차트 다이어그램 구조물 중 변이를 사용하면 쉽게 모델링이 가능하다. Split 및 Split+Join 은 Fork/Join 구조물을 이용하여 모델링 될 수 있다. AnyOrder 및 Choice 구조물은 Choice 구조물과 변이의 응용을 통하여 모델링될 수 있다. If-Then-Else 구조물은 Choice 구조물과 클래스 및 의존관계를 이용하여 모델링 될 수 있다. Iterate 와 이것의 하부 클래스들인 Repeat-While 및 Repeat-Until 은 If-Then-Else 구조물 매핑에서 사용되는 방법을 이용하여 모델링될 수 있다(상기 매핑에 대한 보기들은 지면 관계상 생략하였다).

상기 매핑정의들에서 주목할 만한 것은 If-Then-Else 구조물의 모델링이다. 우리가 If-Then-Else 구조물을 모델링하기 위하여 상태차트 다이어그램에서 지원되는 변이나 상태와 같은 구조물들 외

에 클래스와 의존관계를 사용한 이유는 조건을 GUI 로서 명시할 수 있는 방법을 선택하였기 때문이다. OWL-S 에서 조건명시를 위해 사용할 수 있는 표현문(expression)에는 SWRL(Semantic Web Rule Language) [18], RDF(Resource Description Framework)[19], KIF(Knowledge Interchange Format)[20] 및 PDDL(Planning Domain Definition Language)[21]과 같은 언어를 이용할 수 있다. 이것들 중 우리는 SWRL 을 선택하였다. 그 이유는 SWRL 은 OWL 위에 놓여지는 DAML.org 에서 규칙표현으로 고려하고 있는 표준이기 때문이다. SWRL 의 아톰들은 unary predicate 들(클래스들), binary predicate 들(속성들), equalities 및 inequalities 들로부터 생성될 수 있고 이러한 아톰들은 RuleML[22]의 구성요소인 ruleml:_imp 를 부모로 가지는 ruleml:_body 및 ruleml:_head 의 자식들이다. SWRL 에서는 많은 아톰들의 종류들을 위한 구조물들이 존재한다. 우리는 여기서 classAtom, individualPropertyAtom 및 buitinAtom 을 우선하여 고려하였으나 기타 아톰들도 우리가 제안한 방법을 통하여 유사하게 모델링 될 수 있다.

우리가 OWL-S 의 조건명시를 위하여 SWRL 을 선택하였고 이의 모델링을 위해 클래스 및 의존관계를 선택한 이유는 다음과 같다. 우선 선택한 상기 세 개의 SWRL 의 아톰들을 이용하여 조건을 명시하기 위해서는 조건의 이름과 조건평가에 사용되는 인수들 및 타입들에 관한 정보가 필요하다. 이러한 정보를 UML 상태차트 다이어그램에서 기술할 수 있는 방법들로 우리가 고려한 것들 중 몇몇은 OCL 을 이용하는 것과 조건에 관한 정보를 노트로 처리하여 그 안에서 SWRL 표현문을 직접기술 하는 것이었다. 그러나 OCL 및 노트처리를 통한 SWRL 표현문을 직접 입력하는 것은 개발자로 하여금 OCL 및 SWRL 문법의 이해를 강요할 뿐만 아니라 제공된 표현문을 다시 스캔하여(XMI 안에서 노트 엘리먼트 안에 조건이 명시되기 때문에) 처리해야 하므로 자동변환의 의미를 찾을 수 없다. 이 문제를 해결하기 위하여 우리는 클래스와 의존관계를 사용하기로 하였다. 의존관계의 이름을 스테레오 타입인 SWRL 아톰타입으로 명시하고, 의존관계의 소스인 클래스 이름에서 아톰타입의 조건 이름을 명시한다. 그리고 그 클래스의 속성들에서 명시할 조건의 인수들과 타입들을 나타낸다. 이렇게 함으로써 GUI 를 이용하여 손쉽게 조건을 명시할 수 있을 뿐만 아니라 자동변환 시 간단하게 OWL-S 조건 표현문을 나타낼 수 있다.

3.3 변환 알고리즘

3.2 절에서 언급된 매핑정의들을 이용하여 우리는 XSLT 응용 두 개를 작성하였다. XSLT 응용 두 개는 서비스에 대한 클래스 다이어그램과 상태차트 다이어그램 각각에 대한 XMI 를 입력으로 하여 OWL-S 서비스 온톨로지를 생성한다. 클래스 다이어그램에서 원자 서비스 및 서비스의 속성들 정보(예로, IOPE 들)를 추출하는 것은 클래스의 이름과 클래스의 속성들 정보에서 쉽게 추출될 수 있으므로 별도로 기술하지 않는다. 다음은 서비스 상태차트 다이어그램에서 OWL-S 의 복합 구조물에 관한 정보를 추출하는 방법을 기반으로 한 XSLT 응용의 알고리즘이다.

1: 전체 서비스의 이름을 추출하여 명시한다.
2: OWL-S 의 각 구조물들에 대해서 다음을 수행한다.
  2-1: Sequence 의 경우
    XMI 트리 구조에서 Sequence 변이의 소스와 타겟 id 를 식별하고, 대응되는 상태의 이름을 출력한다.
  2-2 Split 및 Split+Join 의 경우
    XMI 트리 구조에서 Split(Split+Join) 변이의 타겟을 소스로 가지는 상태(상태차트 다이어그램의 Fork/Join 구조물에 대응됨)의 id 를 소스로 가지는 변이들의 타겟 상태들의 이름을 출력한다.
  2-3 AnyOrder 및 Choice 의 경우
    XMI 트리 구조에서 AnyOrder(Choice) 변이의 타겟을 소스로 가지는 상태(상태차트 다이어그램의 choice 구조물에 대응됨)의 id 를 소스로 가지는 변이들의 타겟 상태들의 이름을 출력한다.
  2-4 If-Then-Else 의 경우
    XML 트리 구조에서 If-Then-Else 변이의 타겟을 소스로 가지는 상태(상태차트 다이어그램의 choice 구조물에 대응됨)의 id 를 임시변수에 저장한다.
    임시변수에 저장된 id 를 가지는 상태는 두 개의 변이(Then 및 Else 변이)와 한 개의 의존관계를 가진다.
    // if 조건문 출력
    의존관계 이름을 출력한다.
    의존관계의 타겟 클래스의 이름을 출력하고, 그 클래스 안의 속성들 이름과 타입을 출력한다.
    // then 부분 출력
    Then 으로 명명된 변이의 타겟 상태의 이름을 출력한다.
    // else 부분 출력
    Else 로 명명된 변이의 타겟 상태의 이름을 출력한다.
  2-5: Repeat-While 및 Repeat-Until 의 경우
    XML 트리 구조에서 Repeat-While(Repeat-Until) 변이의 타겟 상태(복합 서비스)의 id 를 찾아 임시변수에 저장한다.
    임시변수값을 id 로 가지는 If-Then-Else 변이를 찾는다.
    // 이러한 이유는 If-Then-Else 가
    // Repeat-While(Repeat-Until) 구조물의 모델링에 사용되는
    // If-Then-Else 임을 보장하기 위함이다.
    // While(Until) 조건문을 표현
    If-Then-Else 변이의 타겟 상태의 id 를 소스로 가지는 Then 및 Else 변이를 식별한다.
    // 반복될 서비스를 출력
    Then 변이의 타겟 상태 이름을 출력한다.

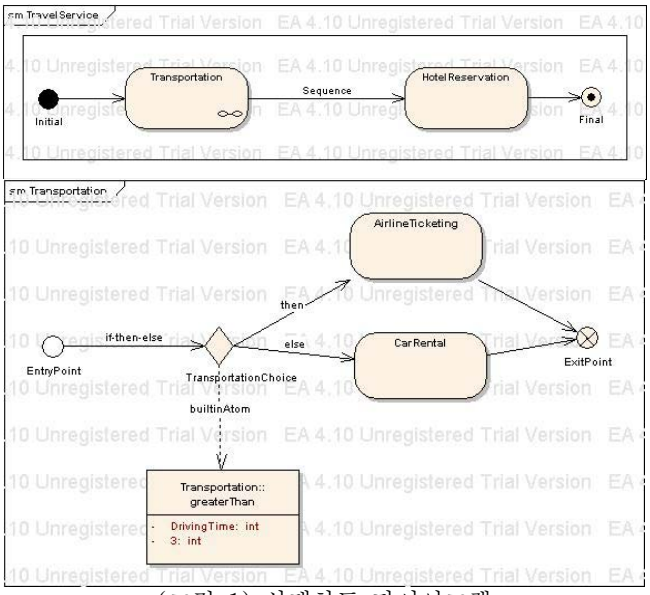
4. 구현

4.1 시나리오

4 장에서 사용되는 시나리오는 시맨틱 웹 서비스를 처음으로 언급한 [4]에서 소개된 여행 서비스 시나리오의 응용으로서 다음과 같다. 어떤 사람은 어떤 곳에서(예로, 샌프란시스코) 또 다른 어떤 곳으로(예로, 모니테리) 여행을 하길 원한다. 그 사람은 비행시간이 3 시간이 넘지 않을 경우에는 자가운전을 희망한다는 제약조건을 가진다. 나아가 전체 여행서비스는 다음과 같은 일련의 단계들로 구성된다고 가정한다. 먼저 여행수단을 결정하고 난 후 숙박을 결정한다. 여행수단은 비행기와 자가용 대여 두 가지 경우가 가능하다.

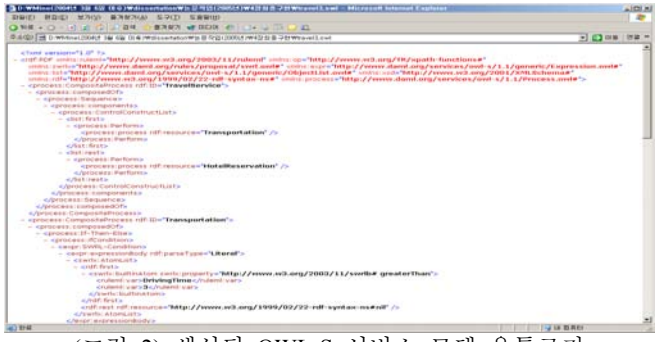
4.2 설계 및 변환

우리는 4.1 절에서 언급한 시나리오를 클래스 다이어그램 및 상태차트 다이어그램을 이용하여 설계하였다. 아래 (그림 1)의 상단은 전체 시나리오가 Transportation 복합 서비스와 HotelResrvation 이라는 원자 서비스로 Sequence 구조물을 통하여 구성되었다는 것을 나타낸다. (그림 1)의 하단은 Transportation 복합 서비스는 If-Then-Else 구조물을 이용하여 조합된 복합 서비스임을 기술한다. 조건은 4.1 절에서 기술한 DrivingTime 이 3 시간보다 크면(greaterThan(DrivingTime, 3)으로 표현될 수 있음) 자동차를 대여한다는 것을 표현한다.



(그림 1) 상태차트 다이어그램

다음 (그림 2)는 3 장에서 기술된 알고리즘을 기반으로 만들어진 XSLT 응용들에서 생성된 두 개의 서비스 온톨로지를 합쳐서 만든 최종의 OWL-S 서비스 모델 온톨로지이다.



(그림 2) 생성된 OWL-S 서비스 모델 온톨로지

4.3 검증

생성된 온톨로지의 유효성 검증을 위하여 우리는 먼저 W3C 에서 제공하고 있는 RDF 유효성 검사 사이트[23]를 이용하였다. W3C 에서는 RDF 까지만 유효성 검사를 수행하고 있기 때문에 OWL 유효성 검사를 위해서는 [24]에서 제공되는 OWL consistency 체크를 이용하였다. OWL-S 온톨로지 유효성 검사기가 OWL-S 버전 1.0 까지 [25]에서 제공하고 있지만 현재까지 Sequence, Unordered 및 Split 구조물들에 한해서 적용가능 하다. 따라서 If-Then-Else 의 구조물을 포함하고 있는 OWL-S 버전 1.1 을 사용한 우리의 서비스 온톨로지의 유효성 검사는 수행할 수 없었다(지면 관계상 검사결과에 대한 화면은 생략하였다).

5. 결론

WSDL 이 가지는 서비스의 인터페이스 정보만으로 지능형 웹 서비스를 실현하는 데에는 한계점들이 많기 때문에 서비스에 대해 보다 풍부한 메타정보가 필요하게 되었다. 지능형 웹 서비스의 실현을 위한 노력들 중의 하나가 OWL-S 이고, 시맨틱 웹 서비스의 핵심은 서비스 온톨로지의 생성 및 번창이다. 따라서 서비스 온톨로지를 자동적으로 그리고 효과적으로 생성하는 방법은 매우 중요하다. 우리는 본 논문에서 OWL-S 서비스 온톨로지를 구성하고 있는 세 개의 온톨로지들 중 서비스의 행동방법에 대한 정보를 가지고 있는 서비스 모델 온톨로지를 자동적으로 생성하는 방법을 제안하였다. UML 클래스 다이어그램 및 상태차트 다이어그램으로부터 원자 서비스 및 속성들에 대한 정보와 서비스 조합에 관한 정보를 각각 추출하여 이를 XMI 파일 형태로 저장한다. OWL-S 의 복합 서비스 모델링을 위한 구조물들과 UML 상태차트 다이어그램 구성요소들과의 매핑정의를 통하여 생성된 XSLT 응용은 XMI 파일을 입력으로 받아 OWL-S 서비스 모델 온톨로지를 생성한다. 우리의 방법은 자동적이고 일반적일 뿐만 아니라 OWL-S 의 문법적인 이해를 강요하지 않는, 실제 개발자들에게 가장 친숙한 방법이다. 우리가 강조하는 또 다른 공헌은 조건의 명시에 있어 OCL 또는 KIF 와 같은 문법을 사용한 것이 아니라 GUI 를 이용하여 조건을 명시하고 이를 자동 변환할 수 있는 방법론을 제안하였다는 것이다. 생성된 온톨로지의 타당성 검증을 위해 우리는 RDF 및 OWL 유효성 검사를 이용하였다.

참고문헌

[1] http://www.w3.org/TR/2004/WD-wsdl20-primer-20041221/
[2] http://www.uddi.org/
[3] http://www.w3.org/TR/2003/REC-soap12-part0-20030624/
[4] Sheila A. Mcllraith, Tran Cao Son, Honglei Zeng, Semantic Web Services, IEEE Intelligent Systems, pp.46-53, 2001.
[5] http://www-128.ibm.com/developerworks/library/specification/ws-bpel/
[6] http://www.w3.org/TR/wsci/
[7] http://www.daml.org/services/owl-s/1.1/
[8] http://www.w3.org/TR/owl-features/
[9] http://www.daml.org/services/owl-s/1.1/related.html
[10] Michele Missikoff, Roberto Navigli, Paola Velardi, The Usable Ontology: An Environment for Building and Assessing a Domain Ontology, ISWC 2002, LNCS 2342, pp.39-53, 2002.
[11] Cranefield S., Purvis M., UML as a Ontology Modeling Language, Proc. Of the Workshop on Intelligent Information Integration, 16th Int. Joint Conference on AI(IJCAI-99), 1999.
[12] K. Baclawski, M. Kokar, P. Kogut, L. Hart, J. Smith, W. Holmes, J. Letkowski, M. Aronson, P. Emery, Extending the UML for Ontology Development, SOSYM 2002, Software System Model(2002) Vol.1, pp.1-15, 2002.
[13] K. Falkovych, M. Sabou, H. Stuchenschmidt, UML for the Semantic Web: Transformation-Based Approaches, Knowledge Transformation for the Semantic Web, IOS Press, pp.92-106, 2003.
[14] Andreas H., Eddie J., and Nicholas K., ASSAM: A Tool for Semi-automatically Annotating Semantic Web Services, ISWC 2004, LNCS 3298, pp. 320-334, 2004.
[15] Keith Mantell, From UML to BPEL: Model Driven Architecture in a Web Services world, http://www-128.ibm.com/developerworks/webservices/library/ws-uml2bpel/
[16] http://protege.stanford.edu
[17] http://staff.um.edu.mt/cabe2/supervising/undergraduate/owlseditFYP/OwlSEdit.html
[18] http://www.daml.org/2004/04/swrl/
[19] http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/
[20] KIFormat: Draft proposed American National Standard (dpans). Technical Report 2/98-004, ANS, 1998.
[21] M.Ghallab et al., Technical Report, report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.
[22] http://www.ruleml.org/
[23] http://www.w3.org/RDF/Validator/
[24] http://www.mindswap.org/2003/pellet/demo.shtml
[25] http://www.mindswap.org/2004/owl-s/validator/