UML 및 OCL을 이용한 서비스 온톨로지 설계 방안에 관한 연구

이 윤 수[†] · 정 인 정[††]

요 약

지능형 웹 서비스는 시맨틱 웹과 에이전트 기술을 통하여 자동적인 서비스의 발견, 호출, 조합 및 상호운영, 실행 감시 및 복구를 수행하기 위한 목적으로 제안되었다. 이러한 지능형 웹 서비스의 목적을 실현하기 위해서는 컴퓨터가 지식을 추론하고 처리할 수 있게 하기 위한 온톨로지가 필수적으로 요구된다. 그러나 현재 지능형 웹 서비스를 위한 서비스 온톨로지의 생성은 서비스 개발자의 휴리스틱에 의존하여 많은 시간과 비용을 소모할 뿐만 아니라 서비스와 서비스 온톨로지간의 완벽한 매핑이 어렵다는 문제점을 가지고 있다. 또한 서비스 온톨로지를 기술하기 위한 마크업 언어를 서비스 개발자가 단기간 내에 학습하기에 많은 어려움이 있는 실정이다.

본 논문에서는 이러한 문제점들을 해결하기 위해 MDA 방법론을 사용하여 서비스 온톨로지를 효율적으로 설계 및 생성하기 위한 방안을 제안한다. 우리가 제안하는 방안은 MDA를 기반으로 UML을 사용하여 웹 서비스 모델을 설계하고 구축하는 과정에서 생성되는 모델을 재사용한다. 즉, 플랫폼 독립적인 웹 서비스 모델을 서비스 온톨로지 기술 언어인 OWL-S에 종속적인 모델로 변환한 후 XMI를 통해 OWL-S 서비스 온톨로지로 변환한다. 본 논문에서 제안하는 방안은 이미 널리 사용되는 UML과 같은 소프트웨어 공학적 방법을 사용하기 때문에 서비스 개발자들이 쉽게 서비스 온톨로지를 구축할 수 있으며 하나의 모델로부터 서비스와 서비스 온톨로지 모델을 동시에 이끌어 낼 수 있는 장점을 가진다. 또한 모델로부터 자동적으로 서비스 온톨로지를 생성함으로써 시간과 비용을 절감할 수 있는 효과를 얻을 수 있다. 그리고 플랫폼 변화와 같은 외부 환경 변화에 유연하게 대처할 수 있다. 끝으로 본 논문에서는 제안된 방안의 타당성을 검증하기 위해 실제로 웹 서비스 모델을 설계하고 서비스 온톨로지를 생성하는 예를 보인다. 또한 생성된 서비스 온톨로지가 올바르게 생성되었는지를 유효성 검사를 통해 검증한다.

키워드 : 지능형 웹 서비스, 온톨로지, OWL-S, MDA, UML

Study Service Ontology Design Scheme Using UML and OCL

Yun-Su Lee[†] · In-Jeong Chung[††]

ABSTRACT

The Intelligent Web Service is proposed for the purpose of automatic discovery, invocation, composition, inter-operation, execution monitoring and recovery web service through the Semantic Web and the Agent technology. To accomplish this Intelligent Web Service, the Ontology is a necessity for reasoning and processing the knowledge by the computer. However, creating service ontology, for the intelligent web service, has two problems not only consuming a lot of time and cost depended on heuristic of service developer, but also being hard to be mapping completely between service and service ontology. Moreover, the markup language to describe the service ontology is currently hard to be learned by the service developer in a short time.

This paper proposes the efficient way of designing and creating the service ontology using MDA methodology. This proposed solution reuses the creating model in terms of designing and constructing Web Service Model using UML based on MDA. After converting the Platform-Independent Web Service Model to the dependent model of OWL-S which is a Service Ontology description language, it converts to OWL-S Service Ontology using XMI. This proposed solution has profits, oneis able to be easily constructed the Service Ontology by Service Developers, the other is enable to be created the both service and Service Ontology from one model. Moreover, it can be effective to reduce the time and cost as creating Service Ontology automatically from a model, and calmly dealt with a change of outer environment like as the platform change. This paper cites an instance for the validity of designing Web Service model and creating the Service Ontology, and validates whether the created Service Ontology is valid or not.

Key Words : Intelligence Web Service, Ontology, OWL-S, MDA, UML

† 준 회 원 : 고려대학교 전산학과 석사과정
†† 종신회원 : 고려대학교 전산학과 교수
논문접수 : 2005년 4월 21일, 심사완료 : 2005년 6월 20일

1. 서 론

지능형 웹 서비스는 웹 서비스에 시맨틱 웹과 에이전트 기술을 접목하여 정보 공유 및 비즈니스 자동화 수준을 혁신적으로 향상시키기 위한 시도이다[19]. 지능형 웹 서비스는 시맨틱 웹과 에이전트 기술을 통하여 자동적인 서비스의 발견, 호출, 결합 및 상호운영, 실행 감시 및 복구를 수행할 수 있도록 하는 것을 목적으로 제안되었다. 이러한 기능을 실현하기 위해서는 컴퓨터가 지식을 추론하고 처리할 수 있게 하기 위한 온톨로지가 필수적으로 요구된다.

그러나 현재 지능형 웹 서비스를 위한 서비스 온톨로지는 서비스 구축과는 별개로 서비스 개발자의 휴리스틱에 의존하여 생성되고 있으며 이로 인한 많은 시간과 비용이 소모된다는 단점을 가지고 있다. 또한 서비스와 서비스 온톨로지가 별개로 구축되어 완벽한 매핑이 어려울 뿐만 아니라 서비스 온톨로지 기술을 위한 마크업 언어는 서비스 개발자가 단기간 내에 학습하기에 많은 어려움이 있는 실정이다[21]. 이러한 문제점들은 지능형 웹 서비스의 실현에 커다란 걸림돌로 작용한다.

따라서 우리는 이러한 문제점들을 해결하기 위해 MDA (Model Driven Architecture)[1] 방법론을 사용하여 서비스 온톨로지를 효율적으로 설계 및 생성하기 위한 방법을 제안한다. 우리가 제안하는 방안은 MDA를 기반으로 UML[2]을 사용하여 웹 서비스 모델을 설계하고 구축하는 과정에서 생성되는 모델을 재사용한다. 즉, 플랫폼 독립적인 웹 서비스 모델을 서비스 온톨로지 기술 언어인 OWL-S[3]에 종속적인 모델로 변환한 후 XMI[4]를 통해 OWL-S 서비스 온톨로지로 변환한다.

우리가 제안하는 방법은 다음과 같은 장점을 가진다. 첫째, 서비스 온톨로지의 생성에 MDA를 사용함으로써 플랫폼에 독립적인 하나의 모델로부터 서비스 및 서비스 온톨로지 모델을 동시에 생성할 수 있기 때문에 시간과 비용이 절약될 수 있다. 또한 서비스와 서비스 온톨로지가 별개로 구축되어 완벽한 매핑을 보장할 수 없다는 문제를 해결할 수 있다. 둘째, 그리고 배우기 어려운 서비스 온톨로지 마크업 언어 대신 소프트웨어 개발에 널리 사용되는 UML과 같은 방법론을 통해 기존의 개발자들도 쉽게 서비스 온톨로지를 생성할 수 있다. 셋째, MDA를 기반으로 하고 있기 때문에 온톨로지 기술 언어가 바뀌거나 서비스 플랫폼이 변경되는 상황에서 유연하게 대처할 수 있다. 이러한 장점들은 지능형 웹 서비스의 발전에 걸림돌이었던 문제점들이 해결될 수 있다는 것을 의미한다.

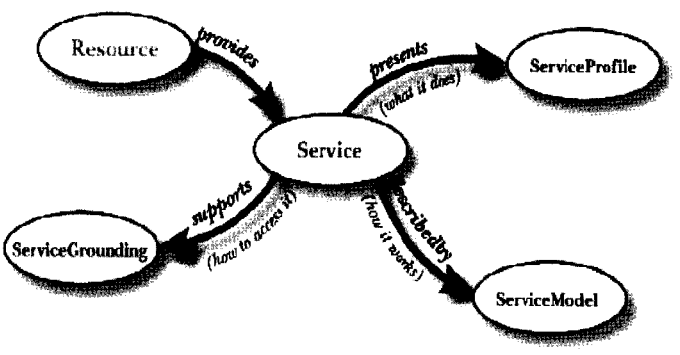
본 논문의 구성은 다음과 같다. 온톨로지를 효율적으로 생성하기 위한 관련연구지가 2장에서 소개된다. 3장에서는 지능형 웹 서비스에 대해 설명하며 4장에서는 MDA 방법론에 대한 배경 지식에 대해 언급한다. 그리고 5장에서는 본 논문에서 제안하는 방법에 대해 기술하며 6장에서는 실험을 통해 서비스 온톨로지를 생성하고 생성된 서비스 온톨로지를 검증한다. 그리고 본 논문에서 제안하는 방법을 사용하여 개선되는 사항에 대해 설명한다. 끝으로 7장에서 결론을 맺고 향후 연구 방향에 대해 논한다.

2. 관련 연구

2.1 지능형 웹 서비스

지능형 웹 서비스는 웹 서비스의 정적이고 구문적 접근 방법의 문제점을 해결하기위해 시맨틱 웹과 에이전트 기술을 웹 서비스와 접목하려는 시도이다. 지능형 웹 서비스는 온톨로지를 기술하기 위한 마크업 언어인 DAML[8]에서 파생된 DAML-S[9]라는 웹 서비스를 위한 온톨로지 기술언어의 제안에서부터 시작되었다.

지능형 웹 서비스에서 제안하는 주요 기능은 자동적인 웹 서비스의 발견(discovery), 호출(invocation), 조합 및 상호운영(Composition and Interoperation), 실행 감시 및 복구(Execution Monitoring and Recovery)이다. 이러한 기능들을 실현하기 위해서는 컴퓨터가 지식을 처리하고 추론할 수 있는 온톨로지가 필수이다. 현재 지능형 웹 서비스를 위한 서비스 온톨로지 기술 마크업 언어는 OWL-S가 표준이다. OWL-S는 서비스의 개요를 기술하는 Service Profile, 서비스의 기능을 기술하는 Service Model, 실제 웹 서비스의 WSDL[7] 기술 문서와의 매핑을 담당하는 Grounding으로 구성되어 있다. (그림 1)은 이러한 OWL-S의 온톨로지 구조를 나타낸다.



(그림 1) OWL-S의 구성요소

서비스 온톨로지의 생성은 현재로서는 수작업으로 이루어지고 있다. 서비스를 설계 및 구축한 후 파생되는 WSDL 문서를 기반으로 서비스 온톨로지를 수작업으로 생성한다. 이러한 현재의 서비스 온톨로지의 생성 방법은 몇 가지 문제점을 안고 있다. 첫 번째로 서비스와 서비스 온톨로지간의 불일치를 들 수 있다. 수작업으로 생성되는 만큼 예상치 못한 실수가 발생할 수도 있고 또한 실제 서비스와는 차이가 있는 엉성한 서비스 온톨로지가 생성될 우려도 있다. 두 번째, 수작업에 의한 생성으로 인해 많은 시간과 비용이 소모될 수 있다. 세 번째, 웹 서비스 개발자들이 서비스 온톨로지를 기술하는 마크업 언어를 쉽게 배우기 어렵다. 이러한 생성단계의 문제점들은 지능형 웹 서비스의 실현에 커다란 걸림돌로서 작용한다.

2.2 MDA(Model Driven Architecture)

MDA는 모델을 기반으로 소프트웨어를 구축하기 위해 OMG(Object Management Group)에 의해 제안된 소프트웨어 공학의 표준 패러다임이다. MDA는 플랫폼 독립적인 PIM(Platform Independent Model)을 구축하고 이를 플랫폼 종속적인 PSM(Platform Specific Model)으로 변환하여 최종 애플리케이션을 구축하는 프로세스를 갖는다. MDA는 그 목적을 위해 UML(Unified Modeling Language), MOF (Meta Object Facility)[10], XMI(XML Meta-data Interchange), CWM(Common Warehouse Metamodel)[11]과 같은 OMG의 다른 표준들을 사용한다.

모든 MDA 모델들은 MOF라는 추상적인 메타모델에 기초를 두고 있으며 따라서 MOF를 기반으로 하는 다른 모델들과 호환성을 가진다는 것을 보장한다. MDA에서 가장 중요한 부분은 UML Profile이다. 이는 특정 도메인에 대한 UML 모델을 작성하기 위한 일반 확장(generinc extension)과 메커니즘을 정의하기 위한 것으로 특정 도메인에 대한 모델링 문제를 기술하고 해당 도메인의 내용들을 모델링할 수 있도록 해준다. UML Profile은 UML을 확장하기 위한 것이므로 MOF와 밀접한 관계가 있으며 최근 새로운 표준으로 발표된 UML 2.0에서는 UML의 다양한 기능적인 사용 방법들을 기술할 수 있는 능력을 가진다. UML 2.0에서는 기존의 UML에서 보장되지 않았던 MOF와의 호환성 문제를 해결했기 때문에 MDA 모델의 구축에 있어서 가장 탁월한 능력을 제공할 수 있다.

XMI는 MOF와 호환성이 있는 모델들을 XML 문서 형태로 표현하고 이를 MOF 호환 데이터베이스에 저장하고 교환할 수 있도록 하는 표준이다. 그러므로 XMI 문서는 곧 MOF XML 문서라고 할 수 있다. XMI는 XML 문법을 따르고 있기 때문에 XML Parser를 이용하여 다른 XML 문서로 쉽게 변환 할 수 있다. CWM은 OMG에서 표준화한 데이터 웨어하우스를 관리하는 메타 데이터 모델이다.

MDA 방법론을 사용하면 이미 널리 사용되고 있는 UML을 이용하여 소프트웨어를 쉽게 설계하고 구축할 수 있으며 이를 여러 가지 다른 도메인에 쉽게 적용시킬 수 있다. 이러한 특징에 의해 MDA 방법론은 적은 시간과 비용으로 높은 생산성을 보장하며 기술 변화 또는 시스템 인프라 변화에 효율적으로 대처 할 수 있다는 장점을 가진다.[12]

2.3 소프트웨어 공학적인 방법을 이용한 온톨로지 생성

소프트웨어 공학적인 방법, 특히 UML을 이용하여 온톨로지를 설계하고 생성하고자 하는 연구들이 상당수 진행되었다. 가장 대표적인 예로 [15]는 UML과 온톨로지 기술 언어 사이의 매핑관계를 정의하고 XMI를 통하여 변환하는 방법을 내세운 최초의 논문이다. 이후에 [14], [16]과 같은 비슷한 방법의 연구들이 있었으며, [13]에서는 UML과 온톨로지 기술언어 사이의 매핑을 좀 더 자세하게 다룬바 있다. [17]은 MDA를 기반으로 ODM(Ontology Definition Model)를 정의하고 이를 통하여 UML과 온톨로지 기술 언어 사이의 매핑을 좀 더 유연하게 하고 있다. 그리고 우리의 지난 연구인 [20]에서는 MDA를 기반으로 UML의 프로파일을 적극 활용하여 온톨로지를 생성하는 방안을 제안하였다.

그러나 UML을 이용하여 도메인 온톨로지를 생성하기 위한 연구는 많이 진행되었지만 서비스 온톨로지를 생성하기 위한 연구는 별로 활발하지 않았다. 서비스 온톨로지는 서비스의 행동의 기술하기 위한 온톨로지이기 때문에 기존에 시도되었던 UML의 Class Diagram을 통한 방법은 적합하지 않다. [22]에서는 State Chart를 이용하여 서비스를 모델링하는 방안을 제시하고 있지만 이는 여러 개의 Composite Process들로 구성된 복잡한 서비스를 나타내기에는 한계가 있다. 이 외에 [21]에서는 MDA와 Product Line을 기반으로 하는 방안을 제안하고 있으나 실제 생성방안에 대해서는 언급하고 있지 않다.

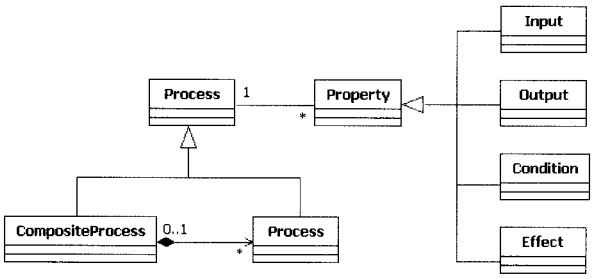
3. 서비스 온톨로지의 설계 및 생성 방안

이 장에서는 UML을 사용하여 서비스 온톨로지를 모델링하기 위하여 OWL-S와 UML 메타 모델간의 매핑에 대해 기술한다. 또한 정의 된 매핑 정보를 기반으로 모델로부터 OWL-S 모델로 변환되는 과정에 대해 기술한다.

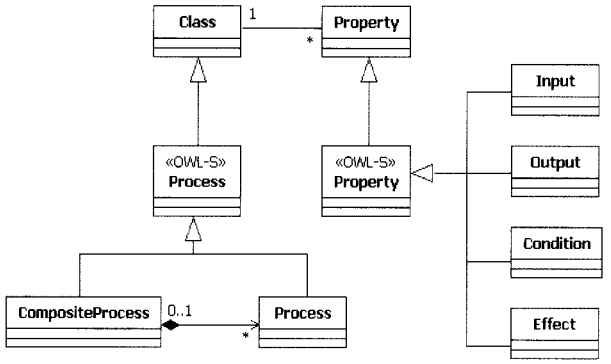
3.1 프로세스의 표현

서비스 온톨로지는 프로세스들의 행동양식을 기술함으로써 서비스가 어떻게 사용되는지를 기술한다. 즉, 서비스 온톨로지의 핵심은 프로세스이다. 이 프로세스는 IOPE(Inputs, Outputs, Preconditions, Effects)[3]로 기술된다. IOPE는 다음과 같은 의미를 가진다.

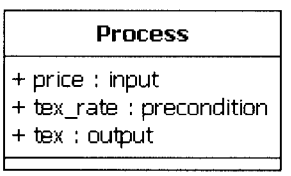
- Inputs : 프로세스의 실행을 위해 요구되는 정보
- Outputs : 프로세스에 의해 출력되는 정보
- Precondition : 프로세스의 올바른 실행을 위해 만족되어야하는 조건
- Effects : 출력에 의해 발생되는 조건



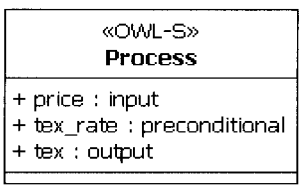
(그림 2) OWL-S 프로세스의 메타모델



(그림 3) OWL-S 프로세스의 메타모델과 UML 메타모델의 매핑



(그림 4) PIM으로 표현된 프로세스



(그림 5) PSM으로 변환된 프로세스

다음 (그림 2)는 OWL-S의 프로세스의 메타모델을 나타낸다.

프로세스는 서비스를 나타내는 최소단위이며 UML의 클래스와 매핑될 수 있다. 따라서 프로세스가 가질 수 있는 속성들은 클래스와 연관된 속성들과 연결될 수 있다. 다음의 (그림 3)은 UML의 메타모델상의 클래스와 매핑된 프로세스의 메타모델을 나타낸다.

이러한 매핑관계를 따르는 예로써 (그림 4)와 같은 프로세스의 PIM을 만들고 (그림 5)와 같이 PSM으로 변환할 수 있다.

(그림 4)의 PIM은 OWL-S뿐만 아니라 다른 구현언어, 즉 자바 또는 C++와 같은 언어에 최적화된 모델로도 변환될 수 있다. 즉, 실제 서비스를 수행하는 애플리케이션과 그 서비스를 기술하는 서비스 온톨로지를 하나의 모델로부터 이끌어 낼 수 있다. 이는 모델을 재사용하여 개발효율을 높이는 MDA의 특성을 웹 서비스의 개발에 적용할 수 있음을 나타낸다. 또한 서비스와 서비스 온톨로지를 하나의 모델에서 이끌어내기 때문에 서비스와 서비스 온톨로지간의 불일치 문제를 자연스럽게 해결할 수 있다.

3.2 OWL-S의 각 구성요소와 UML 메타 모델간의 매핑

OWL-S는 서비스를 기술하기 위해 서비스 프로파일, 서비스 모델, 그라운딩이라는 세 가지 구성요소를 사용한다. 각 구성요소의 역할이 다르기 때문에 각기 다른 기술을 필요로 한다. 그러나 서비스 온톨로지에서 서비스의 행동을 기술하는 구성 요소는 서비스 모델이며 우리는 서비스 모델의 설계와 생성에 중점을 두었다. 서비스 모델은 객체의 행동을 기술하는 활동도(Activity Diagram)로 표현이 가능하다. 서비스 프로파일은 서비스 모델의 설계 과정에서 정의된 프로세스들로부터 변환하여 생성할 수 있으며 그라운딩은 별도로 서비스가 구축된 후 구현 도구로부터 얻을 수 있는 WSDL과의 매핑을 통해 추출한다.

3.2.1 Service Model

서비스 모델은 웹 서비스의 행동을 기술하는 온톨로지이다. 이러한 서비스 모델의 설계를 위해 우리는 UML의 활동도를 사용한다. 활동도는 객체의 행동을 묘사하기 위한 UML의 표현 도구로써 서비스를 구성하는 프로세스들의 실행 순서와 데이터의 흐름 등을 효과적으로 설계할 수 있다. 또한 여러 개의 복합 프로세스들로 구성된 복잡한 서비스를 쉽게 표현할 수 있으며, 나아가 설계 단계에서 다른 서비스와의 조합을 고려할 수 있다는 장점을 가진다[21].

서비스 모델은 프로세스들의 조합으로 서비스의 행동을 기술한다. 프로세스는 단일 프로세스(Atomic Process)와 단일 프로세스들로 구성된 복합 프로세스로 구분되며 단일 프로세스는 서비스의 행동을 기술하는 최소 단위이다[3]. 본 논문에서는 서비스 온톨로지를 생성하기 위해 서비스를 모델로 설계한다. 따라서 단일 프로세스는 활동도에서 모델의 행동을 기술하는 최소 단위인 활동 상태(Action State)와 매핑할 수 있다.

복합 프로세스는 단일 프로세스들과 이들의 실행 순서를 결정하기 위한 제어 구조물들로 구성된다. 이러한 복합 프로세스는 활동도의 Swimlane과 매핑할 수 있다[21]. Swimlane은 활동도에서 논리적(혹은 물리적)으로 유사한 개체들에 의해 실행되는 활동(Activity)들을 포함한다[2]. 즉, 활동 상태와 활동 상태의 흐름을 제어하는 제어구조물들을 포함한다.

서비스의 행동을 제어하기 위한 OWL-S의 제어 구조물들은 각각 특성에 따라 다음과 같이 성격에 따라 UML 활동도의 구조물들과 1 : 1 매핑될 수 있다. 이 매핑관계는 OWL-S의 제어 구조와 UML 활동도 사이의 유사성[21]에서 기인한다.

〈표 1〉 서비스 모델과 활동도 간의 매핑

Service Model	Activity Diagram
CopositeProcess	Swimlane
AtomicProcess	Action State
Sequence	Transition
Repeat	Self Transition
Choice	Decision
Split & Join	Synchronization

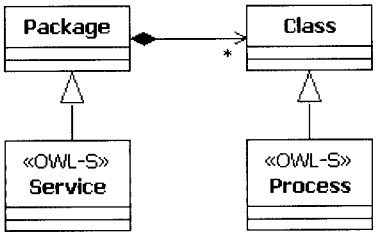
- Sequence → Transition : 상태 변화에 의한 제어의 흐름을 기술
- Split+Join → Synchronization : 상태 변화의 흐름을 분할
- Choice → Decision : 조건에 의해 상태 변화의 흐름을 결정
- Repeat → Self-Transition : 스테이트의 반복 실행

UML의 활동도와 OWL-S의 서비스 모델의 구조물들 사이의 매핑 관계는 <표 1>과 같이 정리 할 수 있다.

3.2.2 Service Profile

서비스 프로파일은 웹 서비스의 능력을 표현하고, 웹 서비스의 설명을 위한 추가적인 특징을 기술하는 온톨로지이다. 서비스 프로파일은 UDDI 레지스트리의 UDDI entry와 유사한 개념이다. 따라서 웹 서비스가 하는 일을 설명하는 것 이외에 서비스 제공자의 정보가 기술된다. 서비스 제공자에 대한 정보는 구체적이고 서비스 제공자에 종속적이므로 모델링에서는 제외한다.

서비스 프로파일의 설계는 서비스 모델의 설계 단계에서 만들어진 프로세스들의 모델을 이용한다. 서비스 프로파일은 서비스가 어떠한 프로세스들을 가지고 있는지, 그 프로세스들이 어떠한 IOPE들로 구성되어 있는지가 주요 내용이며 이외에 서비스의 이름이나 서비스 제공자에 대한 추가적인 정보들이 기술된다. 프로세스들의 모델들을 묶어 하나의 추상적인 모델을 제공하는 것이 바로 서비스 프로파일이다. 따라서 클래스로 표현된 프로세스들을 패키지로 묶어 사용할 수 있다. (그림 6)은 서비스 프로파일을 나타내기 위해 OWL-S의 구조물인 Service와 UML의 패키지와의 매핑을 나타낸다.



(그림 6) Service와 Package의 매핑

〈표 2〉 WSDL의 매핑을 위한 Grounding Class

wsdlVersion	WSDL문서의 버전정보를 기술
wsdlDocument	서비스를 설명
wsdlOperation	서비스에서 제공하는 Operation을 사용하기 위한 방법을 기술
wsdlInputMessage	서비스에 접근하기 위한 WSDL 인터페이스를 기술
wsdlInput	
wsdlOuputMessage	
wsdlOutput	

3.2.3 Grounding

그라운딩은 웹 서비스의 실체화, 즉 실제 웹 서비스의 WSDL 문서와의 매핑을 담당하는 온톨로지로써 프로세스 모델에서 표현된 웹 서비스와 실제 존재하는 WSDL로 표현된 웹 서비스간의 관계를 기술한다. 그라운딩 온톨로지는 WSDL 문서에 대한 매핑 정보를 나타내는 일련의 Grounding Class들로 구성되어 있기 때문에 WSDL 문서로부터 쉽게 추출이 가능하다. 따라서 본 논문에서는 그라운딩 온톨로지를 WSDL 문서로부터 변환하는 방법을 채택했다. OWL-S에서 지원하는 WSDL과의 매핑을 위한 Grounding Class는 <표 2>[3]과 같다.

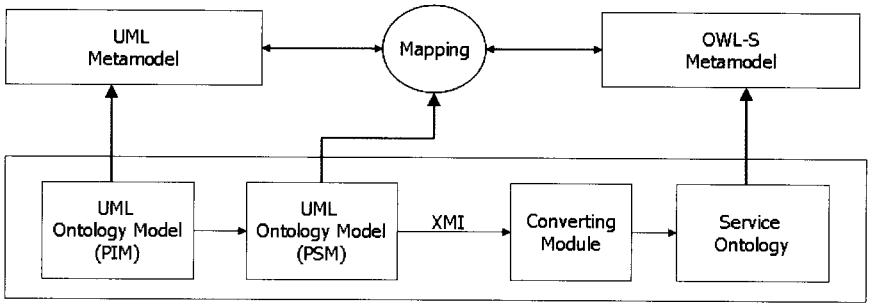
그러나 이 방법은 WSDL 에서 조건부 입출력(conditional input/output)을 표현할 수 없기 때문에 문제가 발생한다[19]. 따라서 이를 해결하기 위해서 추출된 그라운딩 온톨로지를 보완할 필요가 있다.

3.3 변환 프로세스

설계된 모델을 정의된 매핑 관계를 기반으로 실제 서비스 온톨로지로 변환하기 위해 우리는 (그림 7)과 같은 구조의 변환 프로세스를 정의하고 이를 수행하는 애플리케이션을 개발하였다.

이 변환 프로세스는 다음과 같은 순서로 수행된다. 우선 UML 모델링 도구를 통해 웹 서비스의 PIM을 설계한다. 설계된 PIM은 MDA 도구에 의해 PSM으로 변환되며 변환된 PSM은 XMI 문서로 저장된다. XMI는 XML 문서이므로 XML Parser인 DOM[5]을 통해 실제 OWL-S 모델로 변환될 수 있다. 이때 DOM Parser를 이용하는 것은 XSLT를 사용하여 변환할 때의 모호성 문제[13]를 막기 위함이다. DOM Parser는 변수를 이용하여 이 모호성 문제를 쉽게 해결할 수 있다.

현재 이 변환 프로세스는 서비스 온톨로지의 생성에만 초점이 맞춰져 있지만 MDA를 기반으로 하고 있기 때문에 다른 개발 플랫폼으로의 변환이 가능하다[1]. 즉, UML로 모델링된 PIM을 실제 웹 서비스를 구현하기 위한 구현 플랫폼인 JAVA 또는 .NET 플랫폼에 맞는 모델로 변환하여 하나의 모델로 여러 가지 플랫폼에 맞는 서비스 모델로 변환할 수 있다.



(그림 7) 변환 프로세스의 블록 다이어그램

4. 실험 결과 및 검증

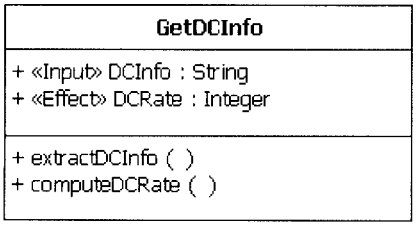
이 장에서는 본 논문에서 제안하는 방법의 타당성을 검증하기 위해 실제로 웹 서비스를 설계하고 설계된 모델을 바탕으로 서비스 온톨로지를 생성하는 실험의 결과에 대해 기술한다.

4.1 서비스 온톨로지의 설계 및 생성

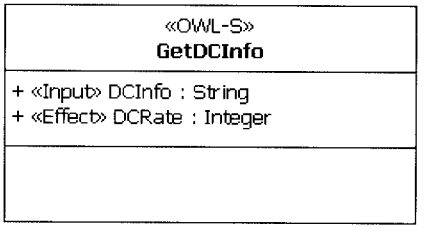
본 논문에서는 실험을 위한 예로써 우리의 이전 연구[18]에서 예로 들었던 레스토랑의 예약 서비스를 사용한다. 이 서비스는 레스토랑의 예약을 위해 빈 테이블이 있는지를 확인하는 FindEmptyTable, 예약 정보를 처리하기 위한 GetReservationInfo, 고객이 원하는 테이블을 선택하기 위한 SelectTable, 고객이 할인자격이 있을 경우를 처리하기 위한 GetDCInfo, 예약을 처리하기 위한 ConformReservation의 5개의 단일 프로세스로 구성되어 있으며 그 기능에 따라 크게 SpagettiaProcess, ReservationProcess, DiscountProcess라는 세 가지의 Composite Process로 묶여 있다.

각 프로세스는 (그림 8)과 같은 PIM으로 설계되어 MDA 도구에 의해 (그림 9)와 같은 PSM으로 변환된다. 3장에서 기술한 바와 같이 OWL-S에 종속적인 모델은 프로세스의 세부 구조와 같은 내용보다 프로세스의 IOPE 속성에 의해 기술되므로 프로세스 클래스의 오퍼레이션들은 은폐되고 IOPE 정보만 남게 된다.

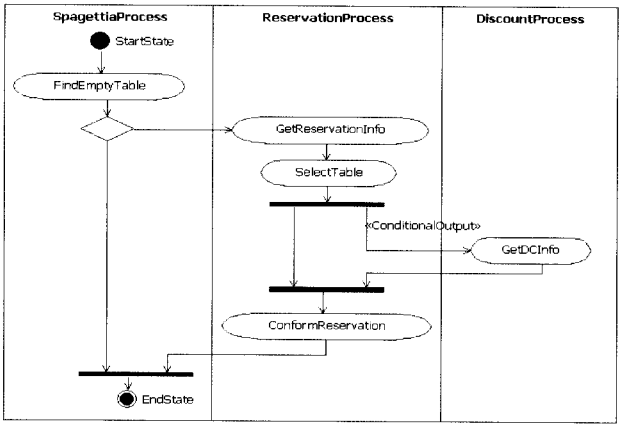
웹 서비스의 서비스 모델 온톨로지를 만들기 위해서는 서비스의 흐름을 정의해야한다. 예제로 사용할 서비스 흐름은 다음과 같다. 예약을 위해 빈 테이블을 검색하고 빈 테이블이 없을 경우 전체 서비스를 종료한다. 그러나 빈 테이블이 있을 경우에는 고객으로부터 예약에 필요한 정보를 입력받아 그에 부합하는 테이블을 선택한다. 또한 고객이 할인에



(그림 8) GetDCInfo의 PIM



(그림 9) GetDCInfo의 PSM

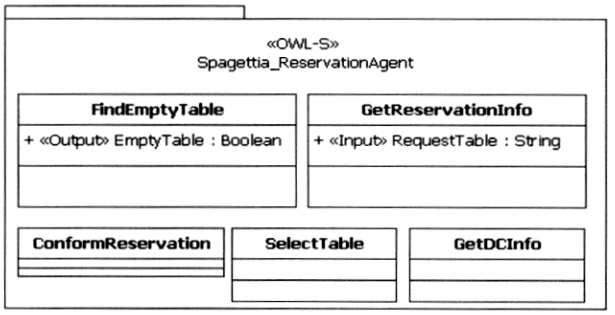


(그림 10) Spagettia Reservation Service Model

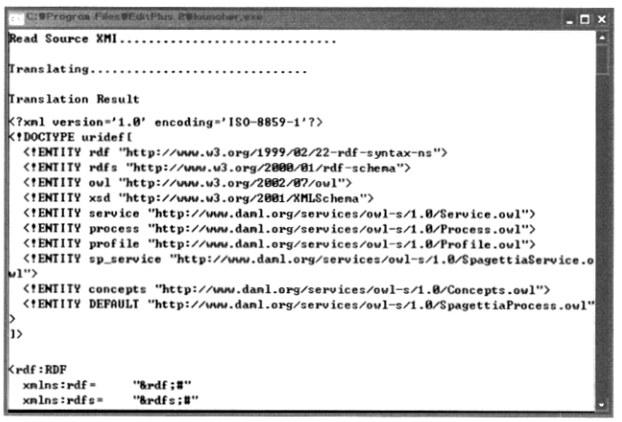
대한 정보를 가지고 있을 경우 그에 대해 조건적으로 처리할 수 있는 별도의 프로세스를 실행할 수 있다. 그리고 마지막으로 예약을 처리하고 서비스를 종료하게 된다. (그림 10)은 이러한 서비스의 흐름을 UML의 활동도로 모델링한 것이다.

본 논문에서는 서비스 프로파일 온톨로지를 만들기 위한 방법으로 패키지를 이용한다. 따라서 개별적으로 정의된 프로세스들을 (그림 11)과 같이 패키지로 모델링하였다. 이 패키지에서는 각 프로세스의 IOPE 정보와 프로세스 이름, 서비스 이름과 같은 정보를 이용하여 서비스 프로파일 온톨로지를 생성할 수 있다.

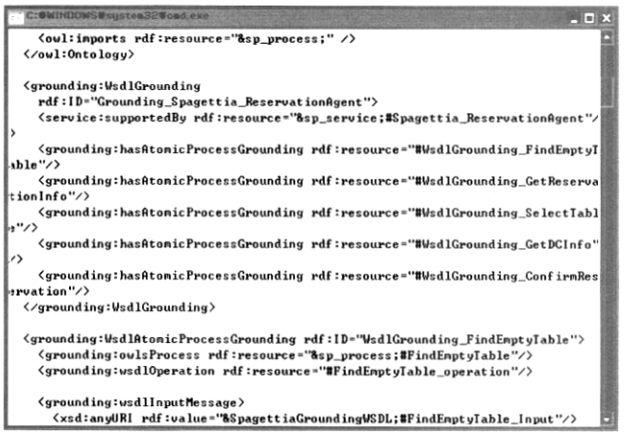
이렇게 정의된 서비스 온톨로지의 모델은 UML 도구에 의해 XMI 문서로 저장되며 저장된 XMI 문서는 변환 모듈을 통해 서비스 모델 온톨로지로 변환된다. (그림 12)는 서비스 모델의 XMI 문서가 실제 서비스 온톨로지로 변환되는 결과를 보여준다.



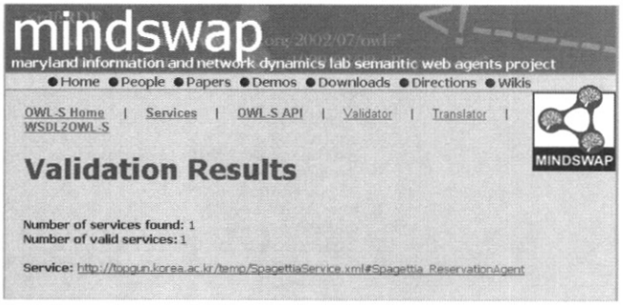
(그림 11) Profile Spagettia__ReservationAgent



(그림 12) XMI 문서로부터 서비스 온톨로지로의 변환



(그림 13) 그라운딩 온톨로지의 추출



(그림 14) 생성된 서비스 온톨로지의 검증 결과

본 논문에서 제안하는 방법에서는 서비스 온톨로지의 그라운딩을 추출하기 위해서 설계된 모델을 플랫폼에 맞는 모델로 변환하고 이를 실제 서비스로 구축함으로써 얻을 수 있는 WSDL 문서를 사용한다. 이를 위해 우리는 레스토랑의 예약 서비스를 .NET 플랫폼으로 구축하였으며 서비스 개발 도구로부터 WSDL 문서를 얻을 수 있었다. 우리는 미리 정의된 OWL-S Grounding Class와 WSDL 간의 매핑을 통해 서비스 개발 도구로부터 얻은 WSDL 문서에서 (그림 13)과 같이 그라운딩 온톨로지를 추출했다.

전술한 바와 같이 WSDL에서는 조건부 입출력을 표현할 수가 없기 때문에 부분적인 수정이 필요하다. 이 문제는 차후 나올 WSDL의 새로운 버전에서 보완될 필요가 있다.

4.2 생성된 서비스 온톨로지의 검증

본 논문에서는 제안한 방법을 통해 생성된 서비스 온톨로지가 정확한지 여부를 검증하기 위해 [23]에서 제공하는 OWL-S Validator를 사용하였다. 이 도구는 [28]에서 제시한 온톨로지 검증 기준에 맞춰 OWL-S로 기술된 서비스 온톨로지를 파싱하여 문법적 오류가 없는지를 검증하고 서비스의 내용이 올바르게 기술되어 실제로 서비스가 가능한지를 검증한다. [28]에서 제시한 검증 기준은 다음과 같다.

- 문법적 오류는 없는가?
- 관련된 온톨로지와 서비스의 위치는 올바른가?
- 관련어휘가 관련된 도메인 온톨로지에 정의되어 있는가?
- 카디널리티(cardinality)가 변화하지는 않는가?

(그림 14)는 실험 결과로 생성된 서비스 온톨로지를 OWL-S Validator를 통해 검증한 결과를 보여준다.

본 논문에서는 OWL-S Validator를 이용한 검증과 함께 DARPA에서 제공하는 예제와의 비교를 통한 검증도 수행하였다. DARPA에서 제공하는 예제들 중 BravoAir의 서비스 온톨로지 예제[24]는 본 논문에서 생성한 서비스 온톨로지와 유사한 구조를 가지고 있으며 온톨로지의 내용상으로도 상당한 유사성을 지니고 있다. 즉, 실험의 결과로써 반자동적으로 생성된 온톨로지가 올바르게 구성되었음을 보여준다. (그림 15)는 DARPA에서 제공하는 BravoAir의 서비스 온톨로지의 일부이며 (그림 16)은 본 논문의 실험 결과로 얻어진 서비스 온톨로지의 일부이다.

우리는 본 논문에서 제안한 방법이 서비스 온톨로지를 정확히 설계하고 생성할 수 있는지를 검증하기 위하여 간단한 웹 서비스를 UML로 모델링하고 실제 서비스 온톨로지로 변환하는 실험을 수행하였다. 또한 OWL-S Validator를 이용하여 검증했으며 DARPA에서 제공하는 예제와의 비교를 통해 수작업 검증을 수행했다. 검증의 결과는 실험으로 산출되는 서비스 온톨로지는 예제로 사용한 레스토랑 예약에 관련된 웹 서비스를 비교적 정확히 기술하고 있음을 보여준다.

```
<process:AtomicProcess rdf:ID="GetDesiredFlightDetails">
  <rdfs:comment>Get details such as airports, prefered time, roundtrip etc</rdfs:comment>
  <process:hasInput>
    <process:Input rdf:ID="GetDesiredFlightDetails_DepartureAirport">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Airport</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="GetDesiredFlightDetails_ArrivalAirport">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Airport</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="GetDesiredFlightDetails_OutboundDate">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#FlightDate</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="GetDesiredFlightDetails_InboundDate">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#FlightDate</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="GetDesiredFlightDetails_RoundTrip">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#RoundTrip</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasOutput>
    <process:Output rdf:ID="GetDesiredFlightDetails_FlightsFound">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#FlightList</process:parameterType>
    </process:Output>
  </process:hasOutput>
</process:AtomicProcess>
```

(그림 15) BravoAir Service Ontology

```
<process:AtomicProcess rdf:ID="GetReservationInfo">
  <process:hasInput>
    <process:Input rdf:ID="RequestTable">
      <process:parameterType rdf:resource="&concepts;#String">&concepts;#Restaurant</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="UserID">
      <process:parameterType rdf:resource="&concepts;#String"/>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="UserPass">
      <process:parameterType rdf:resource="&concepts;#String"/>
    </process:Input>
  </process:hasInput>
  <process:hasOutput>
    <process:Output rdf:ID="TableInfo">
      <process:parameterType rdf:resource="&concepts;#Integer">&concepts;#Restaurant</process:parameterType>
    </process:Output>
  </process:hasOutput>
  <process:hasOutput>
    <process:ConditionalOutput rdf:ID="DCInfo">
      <process:parameterType rdf:resource="&concepts;#String">&concepts;#Restaurant</process:parameterType>
    </process:ConditionalOutput>
  </process:hasOutput>
</process:AtomicProcess>
```

(그림 16) Spagettia Service Ontology

4.3 제안된 방법에 의한 개선 사항

현재 서비스 온톨로지는 개발자의 휴리스틱에 의존하여 Protege 또는 OWL-S IDE와 같은 온톨로지 생성 도구를 이용하여 만들어지고 있다. 이러한 방법은 도구를 이용하므로 편리하긴 하지만 사람이 직접 필요한 내용을 입력하는 형태이다. 그리고 이러한 도구를 이용하기 위해서는 도구를 사용하는 방법이나 서비스 온톨로지 기술 언어에 대해 미리 숙지하고 있어야 한다. 그러나 서비스 개발자가 이러한 새로운 지식들을 습득하기 위해서는 적지 않은 시간과 비용이 소모된다[21]. 또한 현재의 방법으로는 우선 서비스를 개발하고 서비스 개발도구로부터 얻을 수 있는 WSDL 문서를 기반으로 온톨로지 기술 도구를 이용하여 서비스 온톨로지를 생성한다. WSDL 문서는 서비스 온톨로지에 비해 서비스를 설명하는 표현력이 약하기 때문에 이를 기반으로 하는 서비스 온톨로지는 빈약할 수밖에 없다. 또한 개발자의 휴리스틱에 의존하기 때문에 얘기치 않은 실수를 야기할 수도 있다[21].

이러한 문제점을 해결하기 위해 우리는 서비스 개발자들이 이미 널리 사용해온 UML과 같은 소프트웨어 공학적인 방법을 사용하여 서비스를 설계하고 설계된 모델로부터 서비스 온톨로지로 변환하는 방법을 제안했다. 따라서 서비스 개발자가 많은 시간과 비용을 들여 새로운 지식을 습득하지 않아도 UML을 이용하여 설계된 모델로부터 쉽게 서비스 온톨로지를 얻을 수 있다. 또한 모델로부터 자동적으로 서비스 온톨로지를 얻어낼 수 있기 때문에 생성에 필요한 시간을 대폭 감소시킬 수 있다. 또한 우리가 제안하는 방법은 서비스 온톨로지가 미리 정의된 매핑관계에 의해 자동적으로 변환되어 생성되며 서비스를 설계한 모델 자체를 기반으로 하기 때문에 서비스를 충분히 표현할 수 있다. 즉, 서비스의 설계가 잘 되어 있다면 서비스 온톨로지 역시 제대로 생성할 수 있다. 이는 앞 절에서 실험과 검증으로 이미 입증하였다.

5. 결론 및 향후 연구 과제

본 논문에서 우리는 MDA 방법론을 기반으로 지능형 웹 서비스를 위한 서비스 온톨로지의 설계 및 생성 방법을 제안하였다. 제안된 방법은 웹 서비스를 플랫폼에 독립적인 모델로 설계한 후 이 모델을 MDA 도구를 이용하여 플랫폼에 종속적인 모델로 변환한다. 그리고 이 모델들을 XMI를 통하여 실제 서비스 온톨로지로 변환한다. 우리는 제안된 방법을 검증하기 위해 [10]에서 예제로써 사용되었던 레스토랑 예약 서비스를 실제로 모델링하고 서비스 온톨로지를 생성하였다. 그리고 이 서비스 온톨로지를 검증 도구를 이용하여 검증함으로써 타당성을 입증하였다.

우리가 제안하는 방법은 서비스 온톨로지의 생성에 MDA를 사용함으로써 하나의 모델로부터 서비스 및 서비스 온톨로지를 동시에 생성할 수 있기 때문에 시간과 비용을 절약할 수 있다는 장점을 가진다. 또한 서비스와 서비스 온톨로지가 별개로 구축되어 완벽한 매핑을 보장할 수 없다는 문제를 해결할 수 있으며 배우기 어려운 서비스 온톨로지 마크업 언어 대신 소프트웨어 개발에 널리 사용되는 UML과 같은 방법론을 통해 기존의 개발자들도 쉽게 서비스 온톨로지를 생성할 수 있다. 그리고 플랫폼의 변화와 같은 외부적 변화에 보다 유연하게 대처할 수 있다.

향후 연구로서 다음의 두 가지 과제가 있다. 첫 번째로는 그라운딩 온톨로지의 생성 방법에서 발생하는 조건부 입출력을 처리할 수 없는 문제를 해결하기 위한 방안을 모색할 계획이다. 현재 WSDL에서는 조건부 입출력을 표현할 수 없다[11]. 향후 WSDL이 조건부 입출력의 표현을 지원하게 된다면 자연히 해결될 수 있겠지만 현재로써는 생성된 그라운딩 온톨로지를 직접 수정하는 방법보다 효율적인 방법을 찾아야 할 것이다. 두 번째로는 규칙기반 웹 서비스 프레임워크[18]를 위해 서비스 온톨로지의 설계 단계에서 OCL(Object Constraint Language)[6]을 이용하여 온톨로지 모델에 규칙을 부여하는 방안을 연구할 계획이다. 이 프레임워크는 서비스 온톨로지에 규칙을 부여하여 자동적인 검색 및 조합을 가능케 하기 위한 목적을 가지고 있다. OCL로서 온톨로지 모델에 조건적인 제약사항을 기술할 수 있다면 규칙이 부여된 서비스 온톨로지를 만들 수 있는 가능성이 충분하다.

참 고 문 헌

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

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

[3] OWL-S : http://w3.org/2004/OWL-S/

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

[5] DOM : http://www.w3.org/DOM/

[6] OCL : http://www.uml.org/

[7] WSDL : http://www.w3.org/TR/2004/WD-wsdl20-primer-20041221/

[8] DAML : http://www.daml.org/2001/03/daml+oil-index.html

[9] DAML-S : http://www.daml.org/services/daml-s/0.9/

[10] MOF : http://www.omg.org/cgi-bin/doc?formal/00-04-03

[11] CWM : http://www.omg.org/technology/documents/modeling_spec_catalog.htm#CWM

[12] 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.

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

[14] 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.

[15] Stephen Cranefield and Martin Purvis, UML as an ontology modelling language. In Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99), 1999.

[16] 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.

[17] 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.

[18] 양진혁, 민재홍, 이윤수, 김태석, 정인정, 지능형 e-비즈니스를 위한 플랫폼에 관한 연구: 시맨틱 웹 서비스 아키텍처, 21회 한국정보처리학회 춘계학술발표대회 논문집 11권 1호, pp. 369-372 2004. 5.

[19] 지능형 웹 서비스 표준 기술 동향 및 국내 도입 방안 연구 보고서, 한국 전산원, 2004. 4.

[20] 이윤수, 김태석, 양진혁, 정인정, 소프트웨어 공학적 방법을 이용한 온톨로지의 효율적인 설계 및 생성에 관한 연구, 22회 한국정보처리학회 추계학술발표대회 논문집 11권 2호, pp. 645-648, 2004. 11.

[21] Gerald C. Gannod, John T. E. Timm, An MDA-based Approach for Facilitating Adoption of Semantic Web Service Technology, Proceedings of the 8th IEEE Enterprise Distributed Object Computing Conference Workshop on Model-Driven Semantic Web, September, 2004.

[22] Zakaria Maamar, Boualem Benatallah, Wathiq mansoor, Service Chart Diagrams - Description & Application, The Twelfth International World Wide Web Conference, pp. 43-49, 2003.

[23] OWL-S Validator : http://www.mindswap.org/2004/owl-s/validator/

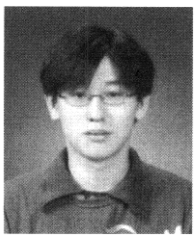
[24] Bravo Air Example : http://www.daml.org/services/owl-s/1.1/examples.html/BravoAirService.owl

[25] DARPA : http://www.daml.org

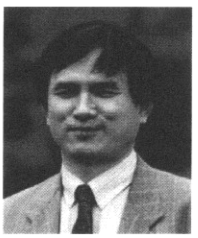
[26] Protege : http://protege.stanford.edu/

[27] OWL-S IDE : http://projects.semwebcentral.org/projects/

owl-s-ide/
[28] Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Debugging owl ontologies. In The 14th International World Wide Web Conference, Chiba, Japan, May, 2005.



이 윤 수
e-mail : arzhna@korea.ac.kr
2003년 고려대학교 전산학과(이학사)
현재 고려대학교 전산학과 석사과정
관심분야 : 웹 서비스, 임베디드 시스템



정 인 정
e-mail : chung@korea.ac.kr
1978년 서울대학교 계산통계학과(이학사)
1980년 한국과학기술원 전산학과(공학석사)
1989년 미국 Univ. of Iowa 전산학과 (전산학박사)
1980년~1983년 삼성전자(주) 컴퓨터사업부 연구원
1983년~1984년 이화여자대학교 전산학과 전임강사 및 학과장
1990년~1997년 고려대학교 전산학과 조교수 및 부교수
1997년~현재 고려대학교 전산학과 정교수
관심분야 : 시맨틱 웹, 웹 서비스