Genie: 온톨로지 기반 시맨틱 웹 서비스 합성 시스템

(Genie: A Semantic Web Services Composition System based on Ontology)

오 지 훈 † 시 대 근 †† 정 영 식 ††† 한 성 국 †††
(Ji-Hoon Oh) (Dae-Keun Si) (Young-Sik Jeong) (Sung-Kook Han)

요 약 웹 서비스 자동화를 위해서는 웹 서비스 자동발견(Automatic Web Services Discovery) 기능, 자동 실행(Automatic Web Services Execution), 자동 구성과 상호운용(Automatic Web Services Composition and Interoperation)이 가능해야 한다. 본 연구에서는 웹 서비스 합성(Web Services Composition)에 중점을 둔다. 웹 서비스의 입력 및 출력 정보를 비롯하여 프리컨디션(precondition), 포스트컨디션(postcondition), 웹 서비스 제공자 정보, 웹 서비스 위치 정보등과 같은 웹 서비스 기본 사항들과 웹 서비스 합성을 의미 기반으로 수행하기 위한 웹 서비스의 수행(action)타입, 서비스 객체(object)와 같은 의미적 요소들을 온톨로지(ontology)를 이용하여 기술한다. 그러므로 기존의 WSDL(Web Service Description Language)이 한계점을 드러냈던 서비스에 대한 의미 정보 기술이 가능하다. 또한, 웹 서비스 간의 의미적 상호운용을 지원함으로써 자연스럽게 내부 또는 외부의 이질적인 어플리케이션간의 통합 서비스를 제공하고 새로운 비즈니스 시스템과의 통합도 자동적으로 이루어지게 하는 웹 서비스 합성 시스템 Genie를 개발한다.

키워드 : 시맨틱 웹, 온톨로지, 웹 서비스 합성, SOAP

Abstract To make Web Services the real applications, the efficient mechanisms for Web Services discovery, Web Services composition and Web Services execution must be provided. Among these issues, especially, Web Services composition plays the key roles in Web Services applications that are loosely coupled and composed applications consisted of primitive Web Service components.

In this paper we demonstrate a new Web Service composition approach using ontologies. We apply ontologies to describe Web Services information such as Web Services input/output parameters, pre conditions, post conditions and other necessary management information. In this paper, we also introduce Action ontology and Object ontology to describe the functional properties of Web Services. These ontologies offer semantic description of Web Services functionalities beyond the limitation of the current WSDL. We can achieve semantic interoperabilities between heterogeneous Web Services in terms of conceptual processing and realize semantic services composition. We implement semantic Web Services composition system called Genie based on service description ontologies.

Key words : Semantic Web, Ontology, Web Services Composition, SOAP

1. 서 론

현재의 웹 서비스 기술은 SOAP(Simple Object Access Protocol), WSDL(Web Service Description Language) 그리고 UDDI(Universal Description Discovery & Integration)를 중심으로 매우 기본적이고 비자동화된 상호작용 모델을 가지고 있다. 웹 서비스 제공자(provider)는 웹 서비스에 대한 정보를 UDDI 레지스트리에 공개(publish)하고 서비스 요청자(requester)에 대한 체계적인 접근을 제공한다. 웹 서비스 사용자는 전송 프로토콜 상의 XML(Extensible Markup Lan-

· 이 논문은 2002년도 원광대학교의 교비지원에 의해서 수행됨
† 정 회 원 : 이프로메디 연구소 연구원
opt@wonkwang.ac.kr
†† 학생회원 : 원광대학교 컴퓨터공학과
sdk124@wonkwang.ac.kr
††† 종신회원 : 원광대학교 컴퓨터및정보통신공학부 교수
ysjeong@wonkwang.ac.kr
skhan@wonkwang.ac.kr
논문접수 : 2003년 11월 6일
심사완료 : 2004년 6월 15일

guage)기반 메시지를 통해 위치정보를 갖고 있는 웹 서비스 레지스트리나 웹 서비스 제공자와 상호작용한다. 또한, 웹 서비스 표준 기술들은 단순히 단일 웹 서비스에 대한 정의와 비즈니스 레지스트리에 등록된 웹 서비스만이 이용 가능하다. 따라서 개발자들에 의한 단편적인 서비스 연결 및 서비스에 대한 사용은 가능하지만 서비스에 대한 상호작용이나 복합적인 해석이 불가능하며, 일반인들에 의한 해석이나 이용 또한 거의 불가능하다. 즉, 구문적인 확장은 가능하지만 의미적인 요소나 의미적인 확장이 거의 불가능한 형태인 현재의 웹 서비스는 한계성을 가지고 있다.

웹 서비스 자동화를 위해 특정 서비스를 제공하는 웹 서비스의 위치를 찾아내는 것과 서비스 이용자의 요구사항을 만족시키는 서비스를 자동으로 찾아내는 웹 서비스 자동발견(Automatic Web Services Discovery) 기능이 제공되어야 한다. 또한, 컴퓨터나 에이전트가 찾아낸 웹 서비스를 자동으로 실행하는 웹 서비스 자동 실행(Automatic Web Services Execution), 특정 작업을 위한 웹 서비스의 자동 선택, 조합, 및 상호 운용을 가능하게 하는 웹 서비스 자동 구성과 상호운용(Automatic Web Services Composition and Interoperation)이 가능해야 한다[1].

본 논문에서는 웹 서비스의 입력 및 출력 정보를 비롯하여 프리컨디션(precondition), 포스트컨디션(post-condition), 웹 서비스 제공자 정보, 웹 서비스 위치 정보등과 같은 웹 서비스 기본 사항들과 웹 서비스 합성을 의미 기반으로 수행하기 위한 웹 서비스의 수행(action)타입, 서비스 객체(object)와 같은 의미적 요소들을 서비스 도메인 온톨로지(ontology)로 기술하였다. 또한, 웹 서비스간의 의미적 상호운용을 지원하여 자연스럽게 내부 또는 외부의 이질적인 어플리케이션간의 통합 서비스를 제공하고 새로운 비즈니스 시스템과의 통합을 위해 D-Mediator(Data-Mediator)와 C-Mediator (Control-Mediator)등과 같은 중재자를 설계 및 구현하였다. 이 2가지 요소를 이용하여 웹 서비스 합성 시스템인 Genie를 구현함으로써 이질적인 웹 서비스들간의 통합을 자동화하여 기존에 비해 훨씬 빠르고, 유연하며, 효율적인 상호운용이 가능하게 하였다.

본 논문의 구성은 2장에서 웹 서비스 합성 시스템과 관련된 연구들에 대해 설명하고 3장에서 온톨로지 기반 의미 정보 모델링과 웹 서비스에 대해서 서술한다. 4장에서는 웹 서비스 합성시 고려사항들, 웹 서비스 온톨로지 구축방법, 그리고 본 논문에서 구현한 웹 서비스 합성 시스템 구조 등을 제안한다. 5장에서는 웹 서비스 합성기에 대한 시스템 요구사항, 핵심 알고리즘 및 적용사례를 제시한다. 마지막으로 6장에서는 결론 및 향후 연구과제에 대하여 기술한다.

2. 관련 연구

본 장에서는 시맨틱 웹 서비스 합성에 관련된 기술, 서비스 지향 구조(Service-Oriented Architecture: SOA) 및 기존 웹 서비스 합성 시스템과의 비교들에 대해서 소개한다.

먼저, 시맨틱 웹 서비스 합성관련 기술을 BPEL-4WS(Business Process Execution Language for Web Services)와 DAML-S(DAML-Service)로 구분하여 설명한다. BPEL4WS는 WSFL(Web Services Flow Language)과 XLANG 최고의 장점들을 패키지로 묶어 매우 자연스러운 방식으로 모든 종류의 비즈니스 프로세스를 구현할 수 있도록 지원한다. BPEL4WS는 구현 언어일 뿐만 아니라 추상 프로세스 개념을 사용하여 비즈니스 프로세스의 인터페이스를 설명하는데 사용된다. 웹 서비스 호출, 데이타 조작, 오류 보고, 프로세스 종료 같은 다른 작동들을 하나로 만들어서 복합적인 프로세스를 생성한다. 실행 가능한 프로세스 구현 언어로서 BPEL4WS의 역할은 기존 서비스들을 조합하여 새로운 서비스를 정의하는 것이다[2,3].

DAML-S는 DARPA 에이전트 마크업 언어 프로그램의 일부로 개발된 서비스를 기술하기 위한 온톨로지이다. DAML-S는 사용자가 웹을 기반으로 서비스를 자동으로 발견, 선택, 채용, 합성 그리고 감독할 수 있는 온톨로지를 제공한다. 이후 에이전트들이 서비스에 대한 기술(description)을 의미 처리하여 서비스들에 자동으로 접근한다[4]. DAML-S는 작업의 목적에 대한 고수준 기술이 주어져 있을 때 특정 작업을 수행하기 위해 웹 서비스를 자동적으로 선택, 구성, 상호운용할 수 있도록 설계되었다. DAML-S는 자동 서비스 구성과 상호운용을 위해 필요한, 개별 서비스의 선행 조건과 효과에 대한 선언적 명세를 제공한다. DAML-S에서는 요구자가 적절한 제공자를 찾기 위한 레지스트리와 같이 동작하는 기반구조 컴포넌트에 의존한다. 이러한 레지스트리들은 제공되는 서비스들 중에서 어느 것이 요청된 서비스에 적당한가를 결정하기 위해 서비스 요청과 제공 사이의 매칭을 제공한다[4,5].

두 번째, SOA는 기업의 소프트웨어 인프라를 구축하는 방법으로 서로 다른 운영체계와 프로그래밍 언어와는 관계없이 어플리케이션간에 데이타와 프로세스를 교환할 때 유용하다. 기업 내에서 일상적인 업무 처리를 위한 팀과 팀간의 연결, 부서와 부서간의 연결뿐만 아니라, 파트너와의 연결, 심지어 경쟁사, 그리고 인터넷을 이용한 고객과의 연결 등 이러한 연결은 앞으로 점점 더 복잡하고 넓게 분포된다. 물론 성공적인 비즈니스를

위해 기업 내 구성원들은 유연하게 변화해야 하지만, 이에 못지 않게 IT 인프라 또한 기업 환경 변화에 유연하게 적응할 수 있어야 한다[14]. 이와 같이 기업 구성원간의 상호 연결의 필연성은 이들이 사용하는 개별(private) 어플리케이션 역시 끊임없이 상호 연결되어야 하는 필요성, 즉 통합에 대한 요구를 내포하게 된다. 뿐만 아니라 모든 상황에 가장 완벽한 단일한 플랫폼은 존재하지 않으므로 기업 내부의 시스템은 이질적인 환경에 노출될 수밖에 없다. 이질적인 플랫폼간의 어플리케이션 통합은 현대 기업 시스템의 필수적인 요구사항이 된다. 그러나 분석과 설계 단계를 거치면서 성공적으로 구축된 어플리케이션들을 필요에 따라 통합하는 것은 그리 쉬운 일이 아니다. 이를 힘들게 하는 가장 근본적인 이유는 해당 어플리케이션들의 분석과 설계 과정에서 통합에 대한 요구 사항이 결여되어 있기 때문이다. 또한 이것은 개별적인 단위 어플리케이션을 위해서는 비교적 잘 정의된 아키텍처가 있는 반면 어플리케이션과 어플리케이션 통합에 관한 전체적인 아키텍처가 결여되어 있기 때문이기도 하다. 서비스 지향 아키텍처는 바로 이러한 유연한 통합에 관한 대안을 제시한다[6,14].

MindSwap(http://www.mindswap.org)에서 개발한 기존 웹 서비스 합성 시스템인Web Service Composer는 웹 서비스를 동적으로 합성할 수 있는 메커니즘을 제공한다. 이 시스템의 대표적인 특징으로는 웹 서비스를 기술하기 위해 DAML 기반의 웹 서비스 온톨로지인 DAML-S를 사용한다는 것과 서비스 출력 결과값에 대한 필터링(filtering) 기능을 제공한다는 것이다. 이 시스템에서 서비스를 합성하기 위해서는 사용자가 직접 합성하기 위한 서비스를 선택한 후에 파라미터 매칭을 직접 입력해야 한다. 화면상에는 시스템에서 지원해주는 웹 서비스 목록을 사용자에게 보여주고, 합성하고자 하는 서비스들을 사용자가 선택한 후 직접 입력 파라미터 값들을 입력하여 웹 서비스를 실행할 수 있다. 내부적으로 이 시스템은 DAML-S 서비스들은 WSDL 그라운딩(grounding) 정보를 사용하여 실제 웹 서비스를 실행한다[7].

이 시스템과 본 연구를 통해 개발된 웹 서비스 합성 시스템인 Genie는 우선 웹 서비스 정보를 기술하는 방식에서 그 차이점이 있다. Web Service Composer는 웹 서비스 정보를 기술하기 위해 DAML 기반의 웹 서비스 온톨로지인 DAML-S를 사용하였지만, Genie는 웹 서비스 정보를 기술하기 위해 기존의 웹 서비스 기술체계인 WSDL과 온톨로지를 접목하여 웹 서비스 합성시 필요한 정보만이 기술된 웹 서비스 온톨로지를 사용한다. Web Service Composer는 웹 서비스를 기술하는 새로운 체계인 DAML-S를 사용함으로써 이를 이용하는 프로토타입(prototype)을 제시하여 주고, 추후 시맨틱 웹 서비스 환경에서 참조 모델이 될 수 있다는 것이다. 이에 반해 합성될 웹 서비스의 선택 및 합성이 의미 정보 기반이 아닌 미리 시스템상에 설정된 서비스 정보를 기반으로 한다는 것이다. 이는 서비스 선택 및 합성이 반 자동으로 수행됨을 의미한다. 그리고 현재 모든 웹 서비스 환경에서 웹 서비스를 기술하기 위해서 사용되는 기술체계가 WSDL이라는 것이다. 현재 몇몇 특수한 경우를 제외하고는 DAML-S로 기술된 웹 서비스는 전무한 상태이다.

Genie는 현재 웹 서비스 환경에서 웹 서비스를 기술하기 위해 사용되고 있는 WSDL 기반이기 때문에 현재의 웹 서비스 환경에 충분히 적용이 가능할 뿐만 아니라 WSDL과 서비스에 대한 의미 정보를 기술하기 위해 온톨로지를 접목하여 자체 웹 서비스 온톨로지를 두어 기존의 WSDL이 한계점을 드러냈던 서비스에 대한 의미 정보기술을 가능하게 한다. 또한, 서비스 선택 및 합성이 의미 정보 기반으로 런타임에 동적으로 수행 가능하다. 이렇게 하여 사용자는 서비스의 선택 및 합성을 의미기반으로 자유롭게 수행할 수 있다.

3. 온톨로지 기반 의미 정보 모델링과 웹 서비스

3.1 온톨로지와 정보 모델링

3.1.1 온톨로지의 필요성 및 역할

시맨틱 웹에서는 사람뿐만 아니라 사람에게 임무를 부여 받은 자동화된 프로그램, "에이전트(agent)[15]"가 사람을 대신하여 웹 정보를 읽고 작업하고 나아가 이를 가공하여 새로운 정보를 생성한다. 예를 들어, 가족휴가를 계획하기 위하여 웹 상에 있는 여행정보를 사용자가 일일이 직접 찾아서 비행기와 호텔을 예약하는 대신에 여행 대행사(Travel Agent)에게 대략적 휴가일정과 개인적 선호도만을 말해주면 세부일정과 여행에 필요한 예약이 이루어지는 것과 같은 원리이다. 이러한 여행 대행사 업무를 웹 상에 있는 정보를 이용하여 스스로 진행할 수 있는 자동화 프로그램 즉, 에이전트를 실현하기 위해서는 웹 정보가 사람 눈이 아니라 컴퓨터 프로그램이 이해할 수 있는 "의미(semantic)"을 가지고 있어야 한다.

정보검색에서 온톨로지를 이용할 경우 현재와 같은 검색어의 포함유무에 의한 검색이 아니라 '개념(concept)'을 이용한 검색과 이를 이용한 작업지시도 가능하다. 현재 이러한 정보가공은 많은 사람이 많은 시간을 들여서 행하여야 하는 어려운 작업이다. 일상의 예로 팩스로 전송된 주소록과 엑셀과 같은 스프레드 시트(spread sheet)로 전달된 주소록을 비교하면, 팩스 자료는 사람을 위한 자료로서 컴퓨터 처리를 위해서는 사람

의 손을 거쳐 다시 가공되어야 한다. 반면 스프레드 시트 자료는 컴퓨터 프로그램, 즉 에이전트에 의하여 표나 차트와 같은 다른 정보형태로 손쉽게 변환되거나 처리할 수 있다. 스프레드 시트 자료에는 컴퓨터 프로그램이 이해할 수 있도록 자료의 의미가 포함되어 있기 때문에 컴퓨터 프로그램에 의해서 손쉽게 처리될 수 있다. 이러한 의미를 표준화된 방법으로 체계적으로 표현하고자 하는 것이 시맨틱 웹(Semantic Web)의 주된 목적이다. 의미 정보가 추가되면 웹의 응용범위는 비교할 수 없을 정도로 확장 가능하다[16].

시맨틱 웹은 웹이 제공하고 있는 정보를 잘 정의된 온톨로지를 기반으로 하여 표현함으로써, 물리적 또는 논리적으로 분산되어 있는 어플리케이션 간의 상호운용성을 제공할 수 있다. 즉 시맨틱 웹에서는 서로 다른 데이타 구조를 갖고 있는 어플리케이션들도 온톨로지를 통해 상대방의 정보를 이해하고 처리가 가능하다. Tim Berners-Lee는 시맨틱 웹을 '잘 정의된 의미를 가진 정보를 제공하여 기계와 인간의 협력을 더욱 향상시키는 웹의 확장'이라고 정의하고 있다[8]. 그림 1은 Tim Berners-Lee가 제안한 시맨틱 웹 구현을 위한 기술구조이다.

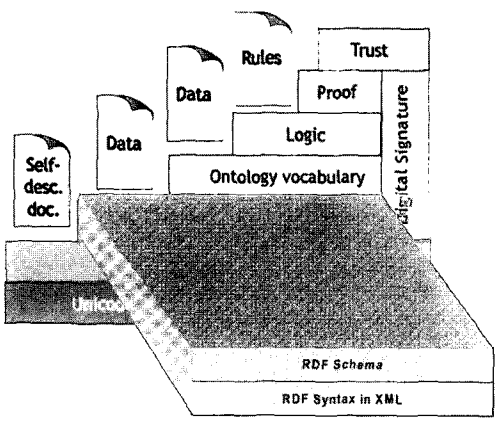


그림 1 시맨틱 웹 기술구조

이 기술구조의 가장 하위레벨에서 웹 프로토콜에서 자원을 지칭하기 위한 주소지정 방법인 URI가 기반이 되고, 이를 기반으로 XML과 Namespace, RDF(S), 온톨로지의 순서로 연구가 진행되고 있다. 그 위의 계층인 Logic 파트에서는 온톨로지 상에 기술된 규칙(rule)을 기반으로 요청된 의미정보들의 관계(relation)를 도출하고, 도출된 정보들간의 관계를 바탕으로 추론(inference) 메커니즘이 적용되어 새로운 의미정보를 산출해 낸다. Proof 계층에서는 하위 계층인 Logic 계층에서의 추론이 옳은지 그른지를 증명하는 단계이며, Trust 계층은 보안에 관련된 사항과 Proof 계층에서 증명된 결과가 과연 신뢰할 수 있는지 없는지를 판단하는 단계이다. 이를 위해서 증명된 결과에 정보에 대한 등급을 부여하거나 Digital Signature등의 기술을 사용할 수 있다. Trust 계층은 시맨틱 웹을 실현하기 위한 가장 최상위 단계이다.

온톨로지는 단어와 관계들로 구성된 사전으로서 어느 특정 도메인에 관련된 단어들을 계층적 구조로 표현하고 추가적으로 이를 확장할 수 있는 추론 규칙을 포함한다. 온톨로지 역할 중 하나는 서로 다른 데이타베이스가 같은 개념에 대해서 서로 다른 단어나 식별자를 사용할 경우에 이를 해결해주는 데 있다. 예를 들어, 주소를 포함하는 두 데이타베이스에서 postal code와 zip code는 같은 것을 의미하다. 이 두 데이타베이스의 정보를 비교하거나 통합하려는 프로그램이 있다면 이 두 단어가 같은 것을 지칭한다는 사실을 알아야 하며 이것이 바로 온톨로지를 통해서 이루어진다. 온톨로지는 웹 기반의 지식 처리나 응용 프로그램 사이의 지식 공유, 재사용들을 가능하게 하는 아주 중요한 요소로 자리잡고 있다[1].

3.1.2 온톨로지 기반 의미 정보 모델링

온톨로지 기반 의미 정보 모델링은 초안을 작성한 후에 반복적인 과정을 거쳐 정보 모델링을 평가하고 수정해 나아간다. 즉, 관련 분야의 어플리케이션이나 문제해결 방법에 온톨로지를 적용하면서 또는 그 분야의 전문가와 상의를 통해 세부적인 사항들을 채워가며 평가하고 수정해 나가는 것이다. 그 과정 중에 온톨로지 디자이너는 장단점과 다른 해결책 보완을 고려하여 모델링 방식을 결정해야 한다. 현재 알려진 온톨로지 모델링 방법론은 다양하고 온톨로지를 개발하는데 있어서 정석이란 없다. 도메인 모델링과 마찬가지로 온톨로지 구축은 고려 중에 있는 어플리케이션과 그 어플리케이션에서 예측되는 확장성에 달려있다. 다음은 일반적인 온톨로지 모델링 방법을 기술한다.

1단계 : 온톨로지가 적용될 도메인(domain)과 적용범위(scope)를 결정한다. 온톨로지 모델링은 구축될 온톨로지가 어느 도메인에서 적용될 것인지를 결정하는 것으로부터 시작된다. 도메인을 결정하지 않고 모델링한다는 것은 무리이며 자칫 잘못하면 온톨로지 구축시 방향성을 잃을 수도 있다. 온톨로지가 적용될 도메인을 결정된 후에는 도메인의 특성과 온톨로지의 사용목적을 고려하여 온톨로지가 적용될 범위(scope)를 정한다. 정확한 범위를 정하지 않고 온톨로지 모델링에 들어가면, 온톨로지 구축시 일관성을 잃게 되고, 시간이 지날수록 온톨로지의 크기가 커져 유지보수 및 평가하는데 문제점을 발생할 수 있다.

2단계 : 수행 질의 문서 작성과 기존에 구축된 온톨로지의 재사용 정도를 결정한다. 온톨로지가 이용되는 동안 요청되는 여러 질의에 대해 응답할 수 있도록 가능한 한 다양한 질의들을 나열함으로써 온톨로지 내에 기술되어야 할 내용의 세부 사항들을 정립하는데 도움이 되는 수행 질의 문서를 만든다. 수행 질의 문서를 만들어 가며 기존에 구축되어 공개된 온톨로지에서 자신이 구축하고자 하는 온톨로지 내에 어느 부분을 어느 정도까지 재사용할 것인지를 조사한다. 특정 온톨로지를 이미 사용하고 있는 어플리케이션과 상호적인 처리를 하기 위해서 기존 온톨로지의 활용도 매우 중요하다.

3단계 : 도메인 내에서 중요한 의미를 지니는 여러 용어(terms)들을 나열한다. 2단계에서 생성된 수행 질의 문서에 나타난 여러 용어들을 나열해 보면서 온톨로지 내에 기술될 여러 클래스(class)들과 속성(property)들을 도출한다.

4단계 : 3단계에서 도출된 여러 용어들을 기초로 온톨로지 내에 기술될 클래스들을 추출한다. 여기서 클래스는 유사한 속성들을 지닌 요소들의 집합을 의미한다. 또한 이렇게 추출된 클래스는 분류 계층(taxonomic hierarchy)을 구성하는 중요 요소가 된다.

5단계 : 3단계에서 도출된 여러 용어들을 기초로 온톨로지 내에 기술될 속성들을 추출한다. 여기서 속성(property)은 클래스에 대한 부가적 의미 정보를 정의하고, 다른 의미 정보와의 관계(relation)을 기술한다.

6단계 : 5단계에서 추출된 속성에 대한 제한조건(constraint)들을 기술한다. 이에 대한 예로써 "어떤 속성은 그 타입으로 'string' 값만 올 수 있고, 정의된 값 중에서 반드시 1개의 값을 가져야 한다. 기본값으로 'default'라는 문자열을 가진다."등이 될 수 있다.

7단계 : 클래스에 대한 인스턴스(instance)들을 생성한다.

이와 같은 온톨로지 기반의 의미 모델링은 다루고자 하는 도메인에서 사용되는 용어들을 정의하고 그들 사이의 관계를 규정하여 그 과정을 구현하는 것을 의미한다. 이 과정에 포함되는 것으로 개념 클래스화, 그 개념들의 관계를 클래스의 계층적인 구조로 정립, 클래스들의 속성과 그 속성에 존재하는 다양성과 제한요소 정의, 마지막으로 인스턴스들의 생성을 들 수 있다. 이를 통해 얻어진 공동 단어들과 공유된 이해를 바탕으로 상호운용시 의사 소통의 일관성을 제공한다.

3.2 웹 서비스

웹이라는 인프라가 풍부하고 확장성이 좋다는 것이 알려지면서 기존의 서비스들을 웹 환경으로 이전하려고 하는 노력이 계속 있어왔으나 이는 쉬운 일이 아니었다. 무엇보다도 현재의 컴퓨팅 환경은 이기종이 복합적으로 존재하고 다양한 종류의 어플리케이션, 프로토콜, 포맷들이 혼재하는 매우 복잡한 것이기 때문에 이들을 하나로 통합하기 위해서는 어느 정도 정형화된 규약이 필요하다. 더불어 보안 문제가 지속적으로 제기되는 환경에서 방화벽을 경유해 데이타와 메시지를 전달할 방법도 찾아야 한다. 또한 경우에 따라서는 원격지에 있는 서비스 객체나 API를 사용할 방법도 강구해야 한다. 그리고 어떤 서비스가 어떤 서버에 위치하는지를 알 수 있는 디렉터리 서비스도 매우 중요한 요소이다. 웹 서비스는 표준 인터넷 프로토콜을 사용하여 접속할 수 있는 프로그램화가 가능한 어플리케이션 로직(programmable application logic)이며, 컴포넌트 기반 개발기법과 웹의 장점만을 활용한다. 컴포넌트처럼 웹 서비스는 서비스의 구현 결과에 대한 걱정 없이 재사용이 가능한 블랙박스 기능을 가지고 있다. 웹 서비스 인터페이스는 전적으로 웹 서비스가 수용하고 생성할 수 있는 메시지에 의하여 정의된다. 사용자가 웹 서비스 인터페이스가 가능한 메시지를 생성하고 활용할 수만 있다면, 웹 서비스는 모든 프로그래밍 언어와 플랫폼에서 구현된다.

그림 2는 웹 서비스 이용 절차를 보여주는데, 서비스 제공자는 공개할 서비스들을 UDDI 레지스트리에 등록한다. UDDI 레지스트리는 웹 서비스에 대한 정보와 이 서비스에 대한 추가 정보의 저장소를 의미한다. 클라이언트가 자신이 이용하고자 하는 웹 서비스를 UDDI에서 검색하면, UDDI는 검색된 웹 서비스의 위치 정보를 클라이언트에 리턴한다. 이후 클라이언트는 리턴된 서비스의 위치정보를 참조하여 WSDL 파일을 요청한다. 이 요청에 대해 서비스 제공자는 서비스의 상세 정보가 기술되어 있는 WSDL 파일을 서비스 클라이언트에게 반환한다. 이러한 일련의 과정을 거친 후에 클라이언트가 자신이 이용하고자 하는 웹 서비스를 SOAP 메시징을 통해 요청하면, 서비스 제공자는 요청된 서비스를 처리

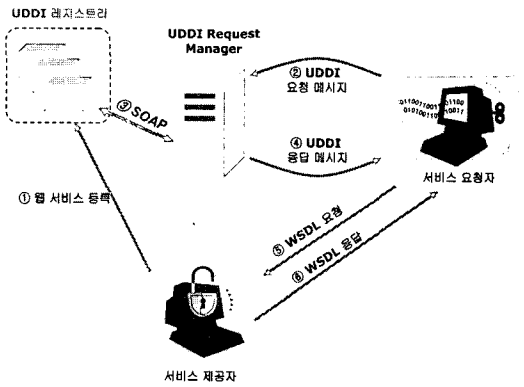


그림 2 웹 서비스 이용 절차

한 후 결과를 SOAP 메시징을 통해 서비스 클라이언트로 리턴한다.

4. Genie 설계 및 구조

4.1 웹 서비스 온톨로지 설계

본 논문에서는 현재 웹 상에 공개되어 실제 운영되고 있는 웹 서비스들에 대한 Action과 Object을 조사하여 웹 서비스에 대한 의미 정보를 추출하여 온톨로지화 한다. 현재 운영되고 있는 웹 서비스를 조사하기 위해 여러 웹 서비스 제공 포탈 사이트, IBM UDDI Business Registry[9], Microsoft UDDI Business Registry[10]를 그 대상으로 조사하였다. 조사된 대표적인 웹 서비스들로는 Amazon Web Service[11]들과 BabelFish 번역 서비스[12], CapeScience의 다양한 웹 서비스들[13]등이 있다. 조사된 서비스 스펙상에는 해당 서비스의 Action과 Object 타입, 입출력 파라미터, 입력 파라미터에 대한 프리컨디션 정보, 출력 파라미터에 대한 포스트컨디션 정보, 그리고 WSDL 위치 정보 등을 기술하였다.

그림 3은 본 논문에서 구축한 웹 서비스 온톨로지에 대한 관계 구성을 나타낸 것이다. 이렇게 구축된 웹 서비스 온톨로지는 행위 요소를 의미하는 Action과 행위 요소가 적용되는 도메인을 의미하는 Object와 같은 웹 서비스에 대한 의미적 기술과 입출력 파라미터, 입력 파라미터에 대한 프리컨디션과 출력 파라미터에 대한 포스트컨디션등과 같은 웹 서비스에 대한 기능적 기술, 이외의 웹 서비스를 기술하기 위해 요구되는 기본 정보들로 구성되어 있다. 예를 들면, 이메일 전송 서비스를 온톨로지를 이용하여 그림 3과 같이 의미 정보 모델링을 한다면, 서비스에 대한 동작 행위 요소인 Action 타입으로써 '전송하다' 또는 '보내다'의 의미를 지니는 'Send'를 취하게 되고 서비스 대상이 되는 객체요소인 Object 타입로써 'Email'을 취하게 된다. 이외에 기능적 기술은 이메일 전송 서비스가 서비스되고 있는 위치정보(WSDL 위치), 서비스를 제공하고 있는 서비스 제공자 정보, 그리고 서비스 실행을 위한 입출력 파라미터들의 정보등이 온톨로지 상에 기술된다.

본 논문에서는 위와 같이 모델링된 웹 서비스 온톨로지를 기술하기 위해 온톨로지 기술언어로써 DAML+OIL을 이용하였고, 온톨로지 입력도구로는 스탠포드 대학에서 개발된 Protégé-2000을 이용하였다. 그림 3에서 사각형 요소는 클래스(class)로, 화살표는 속성(property)을 의미한다. 이를 바탕으로 Protégé-2000을 이용하여 추출된 클래스들과 속성들, 이에 따른 인스턴스들을 입력하였다. 그림 4는 Protégé-2000 상에서 웹 서비스 온톨로지를 구축하는 과정을 보여주고 있다.

본 논문에서는 웹 서비스 온톨로지를 이용하여 의미기반 웹 서비스 선택(selection)과 합성(composition)을 가능하게 하였을 뿐만 아니라 웹 서비스에 대한 기능적 기술을 통하여 입출력정보, 입출력 조건사항, 서비스 위

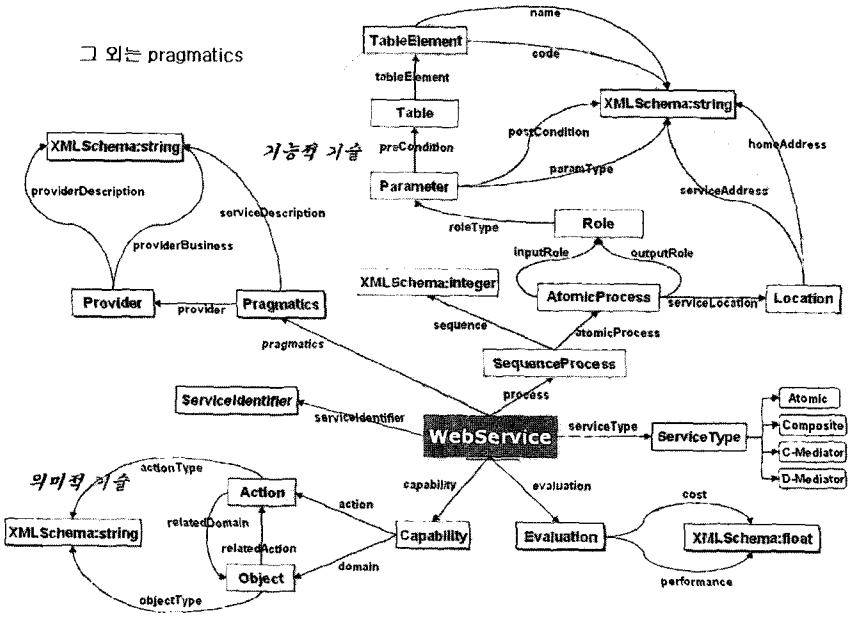


그림 3 웹 서비스 온톨로지 관계 구성

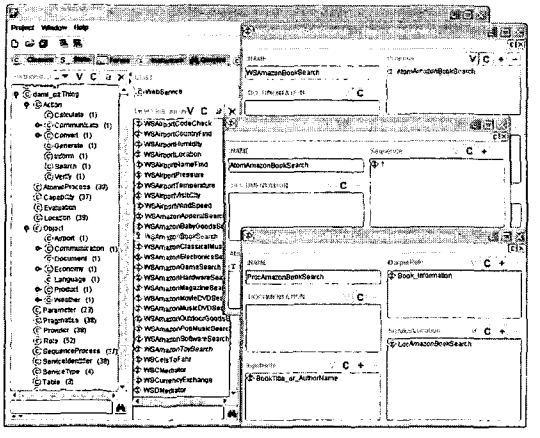


그림 4 웹 서비스 온톨로지 구축화면

치정보도 구축된 온톨로지를 기반으로 추출 가능하게 구축한다. 그림 5는 그림 4에서와 같이 Protégé-2000을 사용하여 구축한 실제 DAML+OIL로 기술(description)

```
<WSOntology_daml:Capability rdf:ID="CapAmazonMagazineSearch">
  <WSOntology_daml:domain rdf:resource="#InsMagazine"/>
  <WSOntology_daml:action rdf:resource="#InsSearch"/>
</WSOntology_daml:Capability>
        ⋮
<daml_oil:ObjectProperty rdf:ID="serviceLocation">
  <rdfs:comment>Web service address with optional home address</rdfs:comment>
  <daml_oil:domain rdf:resource="#AtomicProcess"/>
  <daml_oil:range rdf:resource="#Location"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
</daml_oil:ObjectProperty>
```

그림 5 웹 서비스 온톨로지

된 온톨로지의 일부분을 보여주고 있다.

4.2 웹 서비스 합성 시스템 구조

본 논문에서 구현한 웹 서비스 합성 시스템인 Genie는 Service Ontology, Service Composer, System Manager, WSDL Crawler 이렇게 4개의 모듈로 구성되어 있다. Genie 구조는 그림 6과 같다.

Service Ontology는 웹 서비스 합성할 때 제공되는 웹 서비스에 대한 의미적 기술 정보와 기능적 기술 정보를 제공하여 주는 역할을 한다. 웹 서비스 온톨로지에서 제공되는 정보들과 WSDL Repository에 저장되어 있는 웹 서비스 위치정보를 조합하여 Service Annotator가 추후 Service Composer 파트에서 웹 서비스 합성시 요구되는 여러 의미 정보를 제공하여 준다.

Service Composer 모듈은 Service Annotator에서 넘어온 해당 서비스 정보들을 바탕으로 실제 서비스 합성을 하여 주는 가장 핵심적인 부분이다. Service Composer 모듈 내부에는 서비스 실행의 제어자 역할을 수행하는 Execution Controller, 서비스 합성시 합성될 서비스 간에 파라미터 매칭시 파라미터 타입들을 관리 및 변환하는 Data Manager, Execution Controller에서의 서비스 실행 정보를 바탕으로 실제 서비스들을 합성하기 위한 Composite Service Generator, Data Manager에서 합성될 서비스들의 파라미터 매칭 가능 여부를 기초로 실제 서비스들간의 파라미터 매칭을 수행하는 Parameter Match Maker 그리고 실제 서비스들 간의 합성 작업을 수행하기 전에 서비스 합성이 가능한 조합인지를 먼저 모의적으로 테스트해보기 위한 Service Composition Simulator등을 두어 효율적으로 서비스를

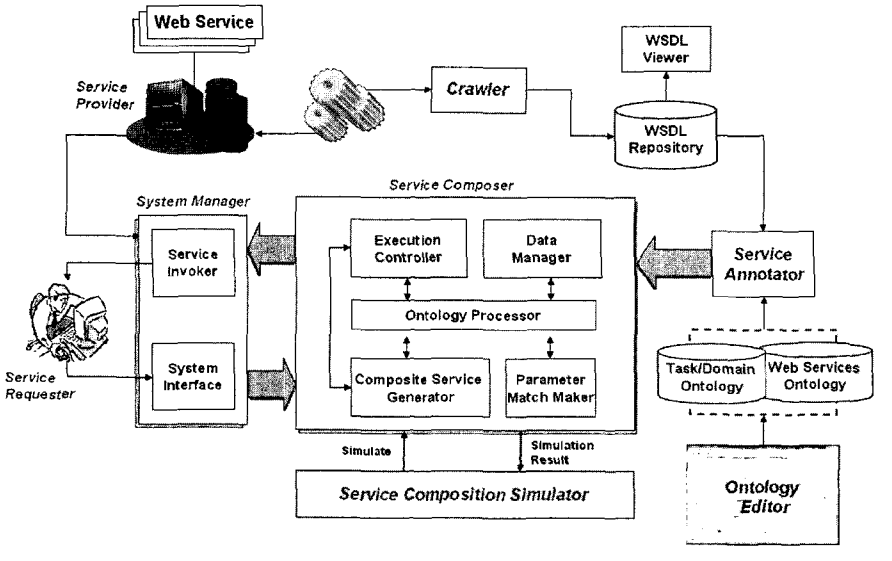


그림 6 Genie 구조

```
using BBN.DAML;
using BBN.RDF;
using BBN.RDF.Common;
namespace Genie {
  public class InputParameter : System.Windows.Forms.Form {
    // ParameterMatch.cs에서 넘어온 인자값들
    DummyPanel  dp;          // 작업 히스토리를 기록하기 위한 객체
    String  serviceName;     // 현재 서비스명
    int  serviceKey;         // 현재 서비스의 키값
    ArrayList  siList;       // 현재 서비스 입력인자 목록
    ArrayList  serviceMatchInfos;  // 현재 서비스의 파라미터 매칭 정보 리스트
    ArrayList  arrIsMatch;         // 현재 서비스 입력인자들의 매칭여부 정보 리스트
      // 해당 클래스 변수들
      bool      isMatch;  // 현재서비스의 입력에 대한 파라메터 매칭여부
      ServiceExecute  se;       ArrayList      siData;
      static Hashtable hashResult = new Hashtable();
            // ServiceExecute.cs에서 넘어오는 서비스 실행 결과값.
            object     executeResult;       string    strInput;
            bool       isPrecondition = false;
            ArrayList  arrCode = new ArrayList();
            ArrayList  arrValue = new ArrayList();
      DAMLModel model;
      <!---------------------- 중 략 ---------------------->
      public void getPrecondition(string strInput){
        int indexA = strInput.IndexOf('(');
        string sString = strInput.Substring(0, indexA);
        this.model = frmOntoHierarchy.model;
        string strName = null;  string strNameID = null;
        IEnumerator stmt = model.ListStatements();
        while(stmt.MoveNext() ) {
            Statement st = (Statement)stmt.Current;
            Resource subject = st.GetSubject();
            Property predicate = st.GetPredicate();
            RDFNode obj = st.GetObject();
            if(predicate.GetLocalName() == "roleType" && subject.GetLocalName() == sString) {
              strName = obj.ToString();
              strNameID = strName.Remove(0, strName.LastIndexOf('#')+1);
            }
        }
        <!---------------------- 중 략 ---------------------->
        stmt.Reset();
    } }
}
```

그림 7 파라미터 매칭정보를 이용한 입력 파라미터 추출 알고리즘

합성시킨다. 그림 7은 Parameter Match Maker에서 생성한 서비스들간의 입력 파라미터와 출력 파라미터의 매칭 정보를 바탕으로 서비스들간의 입력 파라미터를 추출하는 알고리즘을 보여주고 있다.

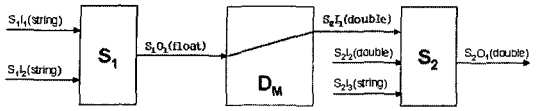
System Manager 모듈은 Service Annotator에서 넘어온 서비스 위치 정보와 Service Composer 상에서의 서비스 매칭 정보를 바탕으로 실제 서비스를 실행한다. 이는 서비스를 호출해 주기 위한 Service Invoker와 호출될 서비스로 넘겨줄 서비스 입력 파라미터를 입력하는 System Interface로 구성되어 있다.

4.3 웹 서비스 합성과 Mediator

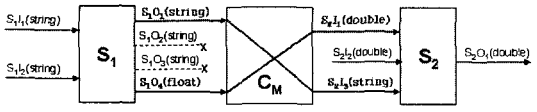
서로 이질적인 2개 이상의 서비스를 합성할 경우 2가지 사항을 고려하여야 한다. 첫째, 서로 상이한 데이타 타입간의 매칭을 시도하려 할 경우 발생하는 타입충돌 문제이다. 예를 들면, 임의의 서비스의 출력 파라미터의 타입은 'float' 타입인데, 이를 'string' 타입이 요구되는 입력 파라미터에 파라미터 매칭을 할 경우 이는 타입충돌의 문제를 발생한다. 둘째, 어떤 서비스의 출력 결과가 1개 이상일 경우 입력 파라미터로 매칭할 파라미터를 추출하는 문제이다. 서비스의 모든 결과를 매칭할 경우는 이 사항을 고려할 필요는 없지만, 만약 반환된 서비스의 출력 결과의 필요한 일부분만을 매칭할 경우는 이를 추출하는 프로세스를 추가할 필요가 있다. 그러므로 본 논문에서는 전자의 타입 충돌 문제를 해결하기 위해 서로 상이한 두 타입간의 형변환을 작업을 수행하는 D-Mediator(Data-Mediator)와 후자의 필요한 출력 파라미터 추출을 하기 위해 C-Mediator(Control-Mediator)를 구현하였다.

그림 8(a)는 서로 다른 두 서비스간의 파라미터 매칭 시 D-Mediator에 의해 서비스 S_1의 'float' 타입인 출력 파라미터 S_1O_1을 서비스 S_2의 'double' 타입인 입력 파라미터 S_2I_1으로 형변환하는 메커니즘을 보여주고 있다. 예들 들어, 두 국가의 통화간의 환율을 계산하는 서비스 S_1의 출력 파라미터인 환율(float 타입)과 환율(double 타입), 환전금액(double 타입)등을 입력 파라미터로 받아 환전하는 환전 서비스 S_2를 합성을 수행할 경우, 그림 8(a)와 같이 타입 충돌 문제가 발생한다. 이때 D-Mediator가 S_1의 S_1O_1(float 타입)을 S_2의 S_2I_1 (double 타입)으로 형변환을 수행하여 줌으로써 이런 문제를 해결하여 준다. 만약 숫자형 string 타입이 아닌 일반 string 타입을 int나 float으로 형변환을 시도할 경우, 이는 허용되지 않는 형변환이므로 적절한 예외처리가 수행된다.

반면, 그림 8(b)는 C-Mediator에 의해 서비스 S_1의 출력 파라미터 S_1O_1, S_1O_2, S_1O_3, S_1O_4 중에서 S_1O_1과 S_1O_4만을 추출하여 서비스 S_2의 입력 파라미터 S_2I_1과 S_2I_3로 각각 매칭되는 메커니즘을 보여주고 있다. 예들 들어, 저자명(string 타입)과 서명(string 타입)을 입력 파라미터를 입력받아 도서를 검색하여 출판일(string 타입), 출판사(string 타입), 도서가격(float 타입)등을 출력 파라미터로 지니는 아마존 도서검색 서비스 S_1과 환율(double 타입), 환전금액(double 타입)등을 입력 파라미터로 받아 환전하는 환전 서비스 S_2를 합성을 수행할 경우, 타입 충돌 문제뿐만 아니라 파라미터 추출문제도 발생한다. 이때는 그림 8(a)와 같이 D-Mediator가 수행되기 이전에 출력 파라미터들 중 S_2의 입력 파라미터들과 매칭할 파라미터들을 S_1에서 추출하는 C-Mediator가 수행된다. 그림 8(b)에서 아마존 도서검색 서비스(S_1)의 S_1O_1과 S_1O_4만을 환전서비스(S_2)의 S_2I_1과 S_2I_3로 각각 매칭을 수행한다. 이때 4개의 출력 파라미터들 중



(a) D-Mediator



(b) C-Mediator

그림 8 웹 서비스 합성 Mediator 타입

에서 S_1O_1과 S_1O_4만을 추출해 내기 위해 C-Mediator가 수행된다. 이후 S_1O_1과 S_2I_3를 매칭할 경우 동일한 타입이므로 D-Mediator가 수행될 필요가 없지만, S_1O_4(도서가격: float 타입)와 S_2I_3(환전금액: double 타입)을 매칭할 경우 상이한 두 타입간의 적절한 형변환을 위해서 D-Mediator가 수행된다. D-Mediator는 앞에서 언급하였듯이 C-Mediator가 수행된 후에 실행된다.

본 절에서 설명한 D-Mediator와 C-Mediator는 향후 합성된 서비스의 프로세스 기술의 일관성을 기하고, 합성된 서비스를 재사용할 때 좀더 용이 할 수 있도록 웹 서비스 형태로 구현하였다. D-Mediator와 C-Mediator가 웹 서비스 합성 프로세스에 기술될 때 'serviceType' 속성으로 각각 D-Mediator와 C-Mediator 타입을 지닌다.

5. Genie 구현

5.1 시스템 요구사항

본 논문에서 구현한 웹 서비스 합성 시스템인 Genie는 Microsoft .NET Framework와 구현 언어로는 C#을 사용하여 개발되었다. 이외에 DAML+OIL로 기술된 웹 서비스 온톨로지를 처리하기 위해 HP LABs에서 개발한 JENA 2 API를 사용하였으며, 이외에 기본적인 XML 데이타를 처리하기 위해서 MSXML Parser를 사용하였다. 본 논문에서 구현한 웹 서비스 합성 시스템은 Microsoft .NET Framework가 설치되어 있고 인터넷이 연결된 환경이라면 언제 어디서나 운영체제에 상관없이 구동 가능하다.

5.2 Genie 구현 및 적용사례

웹 서비스를 합성하기 위해서는 합성할 서비스를 선택해야 한다. 그림 9와 같이 로딩된 메인화면 상에서 합성할 서비스의 Action과 해당 Object를 선택한 후 'Add' 버튼을 클릭하면 웹 서비스 합성기는 선택된 Action과 Object에 해당되는 웹 서비스들을 웹 서비스 온톨로지를 분석한 후 이를 추출하여 합성 서비스 목록에 추가한다. '합성'이라는 의미가 최소 2개의 서비스를 조합할 때를 의미하기 때문에 서비스를 합성하기 위해

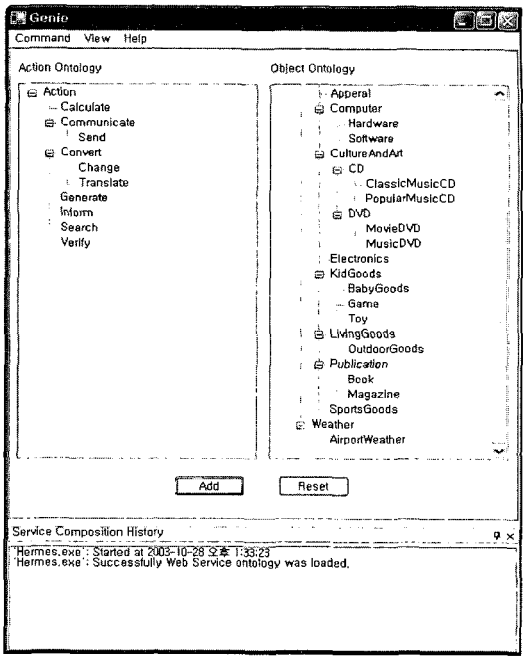


그림 9 Genie 메인 화면

서는 합성 서비스 목록에 최소 2개 이상의 서비스가 추가되어야 한다. 위의 과정을 거쳐 합성 서비스 목록에 추가된 화면은 그림 11과 같다.

그림 10은 그림 9와 같이 Genie 상으로 온톨로지를 로딩하여, 컨셉 리스트를 추출한후 이를 트리뷰 객체에 나타내어 주는 알고리즘이다.

합성될 서비스를 계속 추가하기 위해서는 'Continue' 버튼을 클릭한다. 만약 합성될 서비스들을 모두 추가하였다면 'Composition' 버튼을 클릭하여 추가된 서비스들에 대한 서비스 합성작업을 시작한다. 그림 11은 의미적으로 'Inform' Action과 'AirportWeather' Object에 해당하는 서비스, 'Search' Action과 'Game' Object에 해당하는 서비스, 그리고 'Send' Action과 'Email' Object에 해당하는 서비스들을 합성하는 예를 나타낸 것이다. 화면상의 컴보박스를 클릭하여 보면 웹 서비스 온톨로지에서 추출된 서비스 목록을 확인할 수 있다.

그림 11에서 'Composition' 버튼을 클릭하면, 그림 12와 같은 서비스 파라미터 매칭화면이 나타난다. 그림 12에서는 공항의 습도정보를 제공하는 서비스인 AirportHumidity 서비스, Amazon 사이트에서 게임 제품을 찾아주는 AmazonGameSearch 서비스, 그리고 e-mail을 전송하는 서비스인 EmailSend 서비스의 입력 파라미터와 출력 파라미터를 매칭시키기 위한 화면이다. 여기서는 EmailSend 서비스의 'EmailBody'라는 입력 파라미터에 AirportHumidity 서비스의 출력 파라미터인

```
/// <summary>/각각의 DAML+OIL 문서를 로딩후 클래스 리스트를 생성/// </summary>
/// <param name="damlFileURI">DAML+OIL URI</param>
/// <param name="RootClass">DAML+OIL 문서 루트 클래스(컨셉)명</param>
/// <param name="tvDAMLTree">컨셉 Hierarchy를 받아오기 위한 TreeView객체</param>
private void damlDocLoad(string strRootClass, System.Windows.Forms.TreeView trvDAMLTree) {
    string strDamlClassID = null;   // 클래스(컨셉)명을 위한 문자열변수 선언
    // DAML+OIL 문서를 컨트롤 위한 DAMLModel 객체 생성후 DAML+OIL 문서로딩
    model = new DAMLModelImpl();
    model.Read(GlobalVariables.ONTOLOGY_URI);
    // 로딩된 클래스(컨셉)들에 대한 리스트 생성
    IEnumerator damlClassList = model.ListDAMLClasses();
    // 생성된 클래스(컨셉) 리스트 루핑처리부
    while(damlClassList.MoveNext())  {
      // 현재 처리되고 있는 리스트 요소를 클래스(컨셉)화한 후,
      // 그에 대한 클래스명을 추출
      DAMLClass damlClass = (DAMLClass)damlClassList.Current;
      strDamlClassID = damlClass.GetLocalName();
      // 현재 클래스가 인자로 넘어온 strRootClass 일 경우
      if(strDamlClassID == strRootClass) {
          // 현재 클래스명으로 TreeNode 객체 생성한 후,
          // 생성된 TreeNode 객체를 TreeView에 추가
          TreeNode rootNode = new TreeNode(strDamlClassID);
          trvDAMLTree.Nodes.Add(rootNode);
          // 현재 클래스(컨셉)에 대한 서브클래스(컨셉) 처리부 호출
          processSubClass(rootNode, damlClass);
      } else  // strRootClass가 아니면 다음 차례로 계속
      {  continue;  }
  }
  trvDAMLTree.ItemHeight = 18;
  trvDAMLTree.ExpandAll(); // 모든 트리를 펼친다.
}
```

그림 10 온톨로지 로딩 및 컨셉 추출 알고리즘

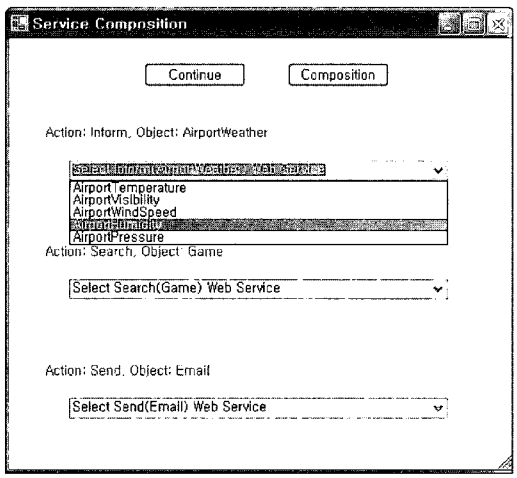


그림 11 합성될 서비스 선택화면

'Weather_Message'를 매칭시킨다. 추후 서비스 입력창에서 위의 'EmailBody'를 입력하는 텍스트 박스에는 AirportHumidity 서비스의 출력 파라미터인 'Weather_Message' 결과가 자동으로 입력된다. 'Simulate' 버튼을 클릭하면 위에서 설정된 파라미터 매칭정보가 적절한지에 대해서 테스트가 가능하다. 파라미터 매칭 설정이 끝

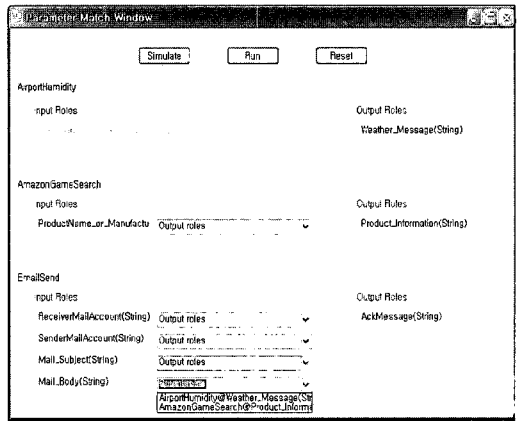


그림 12 파라미터 매칭화면

났다면, 'Run' 버튼을 눌러 실제 서비스 제공자측에서 제공하고 있는 원격지의 웹 서비스를 실행할 수 있다.

그림 13은 서비스에 대한 입출력 파라미터 분석 및 파라미터 매칭정보 생성을 위한 알고리즘이다.

```
private void btnRun_Click(object sender, System.EventArgs e) {
  string strMatchInfo = null;    this.searchAllService();
  this.addServiceListToHash() // 추출된 서비스에 키값을 부여해서 Hashtable화
  siList = new ArrayList();
  // 생성자로 넘어왔던 파라미터 리스트를 루핑(서비스 갯수만큼 루핑)
  for(int i=0; i < sParamList.Count; i++) {
    // 이전 서비스에 대한 정보 초기화
    strMatchInfo = null;   cboText.Clear();   serviceMatchInfos.Clear();
    ParamSetup mp = (ParamSetup)sParamList[i];   strService = "WS" + mp.SERVICENAME;
    // 각각의 서비스에 대한 입력인자들을 가져온후 이를 ArrayList에 저장
    Params[] ip = (Params[])mp.IPLIST[i];
    for(int u=0; u < gbxIn[i].Controls.Count; u+=2) { cboText.Add(gbxIn[i].Controls[u+1].Text); }
      // 서비스 매칭리스트 저장하는 구문
      for(int k=0; k < ip.Length; k++){
        // 서비스 입력정보를 ArrayList에 저장
        siList.Add(ip[k].ToString() );
        strMatchInfo = mp.SERVICENAME + "$" + ip[k].ToString();
        strMatchInfo += "#" + cboText[k].ToString();
        // 파라미터 매칭정보 리스트
        serviceMatchInfos.Add(strMatchInfo);
      }
      for(int w=0; w < sParamList.Count; w++){
        cboText1.Clear();     ParamSetup mp1 = (ParamSetup)sParamList[w];
        Params[] ip1 = (Params[])mp1.IPLIST[w];
        for(int h=0; h < ip1.Length; h++)   {
          int cntExtract = 0;
          for(int f=0; f < gbxIn[w].Controls.Count; f+=2) {
            cboText1.Add(gbxIn[w].Controls[f+1].Text);
          }   }
      <!---------------------------- 중 략 ---------------------------->
      // InputParameter.cs로 서비스 키값과 해당 서비스의 입력 파라미터 정보를 넘겨준다.
      InputParameter inp =
          new InputParameter(dp, mp.SERVICENAME, serviceKey, siList, serviceMatchInfos, arrIsMatch);
      if(inp.ShowDialog(this) == DialogResult.OK) {
      // 서비스명과 서비스키값, 입력 파라미터 정보를 초기화 한후 다음 서비스 실행
      strService = "";   serviceKey = 0;   siList.Clear();   continue;
    }   } this.Close();
}
```

그림 13 입출력 파라미터 분석 및 파라미터 매칭정보 생성 알고리즘

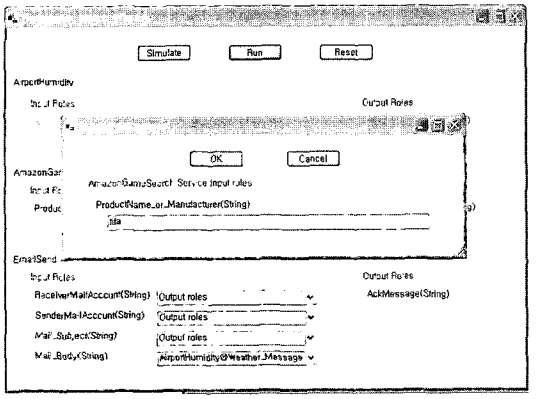


그림 14 파라미터 매칭 정보가 없는 파라미터 입력화면

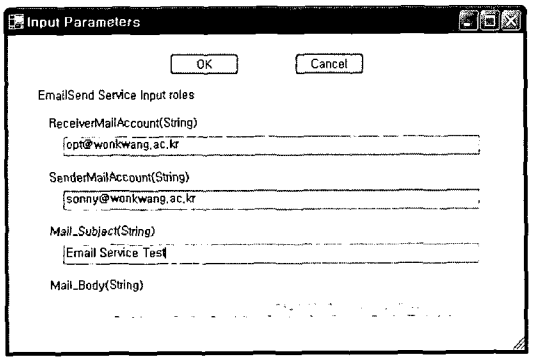


그림 15 파라미터 매칭 정보가 있는 파라미터 입력화면

그림 14와 그림 15는 각각 파라미터 매칭 정보의 유무에 따라서 웹 서비스를 실행하기 위한 실제 파라미터 입력단계를 보여주고 있다. 이 단계에서 데이타를 입력하면, 입력된 데이타를 서비스가 위치한 원격지로 SOAP 메시지 형태로 전송한다. 서비스 호스트에서는 요청된 서비스를 실행하여 그 결과 값을 다시 SOAP 메시지 형태로 서비스를 호출한 클라이언트로 반환한다.

파라미터 매칭이 설정되지 않은 서비스 경우는 데이타를 입력 후 서비스 실행 결과를 바로 확인해 볼 수 있지만, 파라미터 매칭이 설정된 서비스 경우에는 출력 파라미터가 후에 실행될 서비스의 입력 파라미터로 매칭이 되었기 때문에 다음 서비스 과정에서 파라미터 매칭이 적용된 입력 파라미터 입력 텍스트 박스에 매칭된 서비스의 출력 파라미터 결과값이 자동으로 입력된 후 데이타 수정을 할 수 없도록 비활성화 처리된다.

지금까지 본 논문에서 구현한 온톨로지 기반의 웹 서비스 합성 시스템인 Genie의 개괄적인 구조 및 동작 절차에 대해 기술하였다. 본 논문에서는 웹 서비의 의미적 요소들을 온톨로지(ontology)로 기술함으로써 기존의 서비스 기술방식이 한계를 드러냈던 서비스에 대한 의미정보기술을 가능하게 하였다. 또한, 웹 서비스간의 의미적 상호운용을 지원하여 자연스럽게 내부 또는 외부의 이질적인 어플리케이션간의 통합 서비스를 제공하고 새로운 비즈니스 시스템과의 통합을 위해 D-Mediator (Data-Mediator)와 C-Mediator(Control-Mediator)와 같은 중재자를 설계 및 구축하였다. 이를 통해 이질적인 웹 서비스들간의 통합을 자동화하여 기존에 비해 훨씬 빠르고, 유연하며, 효율적인 상호운용이 가능하게 하였다.

6. 결 론

웹 서비스는 XML, SOAP, WSDL등의 표준 인터넷 프로토콜을 사용하여 액세스할 수 있고 시스템 간의 통신과 응용 프로그램 간 통신을 투명하게 하기 위한 웹 지원 표준의 구현이다. 이런 웹 서비스 자동화를 위해서는 특정 서비스를 제공하는 웹 서비스의 위치를 찾아내는 것과 서비스 이용자의 요구 사항을 만족시키는 서비스를 자동으로 찾아내는 웹 서비스 자동발견, 컴퓨터나 에이전트가 찾아낸 웹 서비스를 자동으로 실행하는 웹 서비스 자동 실행, 특정 작업을 위한 웹 서비스의 자동 선택과 조합, 그리고 상호 운용을 가능하게 하는 웹 서비스 자동 구성과 상호운용이 가능해야 한다.

본 논문에서는 온톨로지 기반의 의미 정보 모델링과 웹 서비스에 대해서 기술하였고, 웹 서비스 합성시 고려사항들, 웹 서비스 온톨로지 구축방법, 그리고 본 논문에서 구현한 웹 서비스 합성 시스템 구조를 제시하였다. 웹 서비스 합성 시스템에 대한 시스템 요구사항과 시스템 구현 및 적용사례를 제시하였다.

또한 향후 연구로는 현재 구축된 웹 서비스 합성 시스템 대부분이 서비스 호스트에서 서비스의 연결이 끊어지거나 서비스 운영을 중지할 경우 시스템 상에서는 사용자가 원하는 서비스 정보를 리턴할 수 없다. 이는 서비스 호스트에서 운영되고 있는 서비스들의 상태에 의존적이 될 수 밖에 없다는 것이다. 이를 위해서 요청 서비스가 불시에 끊어 졌을 경우, 서비스 정보를 기술할 때 서비스 합성 시스템의 신뢰도를 높이기 위한 유사 서비스로의 자동적인 연계 기능을 추가를 위한 연구가 되어야 한다.

참 고 문 헌

[1] McIlraith, Sheila, "Semantic Enabled Web Services," XML-Web Services ONE Conference, 2002.6.

[2] Sanjiva Weerawarana, Francisco (Paco) Curbera, "Business Process with BPEL4WS," http://www-903.ibm.com/developerworks/kr/preout.jsp?url=http://www-106.ibm.com/developerworks/webservices/library/wsbpelcol2/&origin =ws, 2002.

[3] Curbera, F., Goland, Y., Klein, J., Leyman, F., Soller, D., Thatte, S., Weerawarana, S., "Business Process Execution Language for Web Services," BEA Systems & IBM Corporation & Microsoft Corporation, http://www-106.ibm.com/developer-works/library/ws-bpelwp, 2002.
[4] The DAML Service Coalition, "DAML-S: Semantic Markup for Web Services," http://www.daml.org/services/damls/0.9/, 2003.
[5] Denker, G., Kagal, L., "Security annotation for DAML Web Services," 2003.
[6] Microsoft MSDN, XML Web Service, http://msdn.microsoft.com/library/default.asp?url=/nhp/Default.asp?contentid=28000442
[7] MindSwap Web Service Composer, http://www.mindswap.org/~evren/comp oser
[8] Tim Berners-Lee, James Hendler, Ora Lassila, "The Semantic Web," Scientific American, 2001.5.
[9] IBM UDDI Business Registry, https://uddi.ibm.com/ubr/registry.html
[10] Microsoft UDDI Business Registry, http://uddi.microsoft.com
[11] Amazon.com Web Services, http://www.amazon.com/gp/browse.html/002-3527402-0547209?node=3435361
[12] AltaVista's Babel Fish Translation Service, http://babelfish.altavista.com/
[13] CapeScience: Live Web Services, http://www.capescience.com/webservice s/index.shtml
[14] 홍영준, "서비스 지향 아키텍처", 시사컴퓨터 Tech Report - SOA, 2003.8.
[15] "특집: 에이전트 시스템", 정보과학회지 제15권 제3호, 1997.3.
[16] 이재호, 양정진, "시맨틱 웹 : 차세대 지능형 웹 기술", 한국정보통신기술협회 TTA 저널 제81호, 2002.6.

오 지 훈

2002년 원광대학교 컴퓨터및정보통신공학부(공학사). 2004년 원광대학교 컴퓨터공학과(공학석사). 1999년~현재 원광대학교 시맨틱 웹 연구실 연구원. 2004년~현재 (주)이프로메디 연구소 선임연구원. 관심분야는 온톨로지, 시맨틱 웹 서비스, 지능형 e-비즈니스

시 대 근

1997년 원광대학교 컴퓨터공학과(공학사) 2002년 원광대학교 교육대학원 전자계산교육전공(교육학석사). 2004년~현재 원광대학교 시맨틱 웹 연구실 선임연구원 2004년~현재 원광대학교 컴퓨터공학과 박사과정. 관심분야는 시맨틱 웹 서비스, 온톨로지, 지식표현 및 추론

정 영 식

1989년 고려대학교 전산학 석사. 1993년 고려대학교 전산학 박사. 1997년~1998년 미시간 주립대학교 전산학과 객원교수. 1993년~현재 원광대학교 컴퓨터 및 정보통신공학부 교수. 관심분야는 분산병렬처리, 그리드컴퓨팅, LBS

한 성 국

1979년 인하대학교 전자공학과(공학사) 1983년 인하대학교 전자공학과(공학석사) 1988년 인하대학교 전자공학과(공학박사) 1990년~1992년 University of Pennsylvania 방문교수. 2003년~2004년 University of Innsbruck와 DERI 연구교수 1984년~현재 원광대학교 컴퓨터 및 정보통신공학부 교수 관심분야는 온톨로지, 시맨틱 웹 서비스, 지식표현 및 추론, 자연언어처리