BDI 에이전트 구조 기반의 시맨틱 웹서비스 실행기

진 훈[O] 최재혁 김인철
경기대학교 전자계산학과
{bioagent[O], 01choi, kic }@kyonggi.ac.kr

A Semantic Web Service Execution System Based on BDI Agent Architecture

Hoon Jin[O], Jaehyuk Choi, Incheol Kim
Dept. of Computer Science, Kyonggi University

요 약

시맨틱 웹서비스와 관련하여 서비스 검색 및 조합에 관하여 많은 연구들이 이루어지고 있으나 인터넷의 특성을 고려할 때 서비스 실행에 관한 연구는 웹서비스의 안정적인 제공이라는 측면에서 고려되어야 한다. 이를 위해서는 실행과정에 의미적 접근이 시도되어야 하는데, BDI 구조 기술이 적절한 대안 기술이 될 수 있다. BDI 구조를 이용한 웹서비스 실행기는 특성상 실시간으로 데이터 타입과 의미의 일관성을 추론해 내고, 해당 서비스의 실행을 위한 조건과 영향을 고려하여, 제약조건의 변화 시에도 능동적으로 대처할 수 있는 장점을 갖게 된다. 우리는 웹서비스 실행기로서 SWEEP 시스템을 개발하고 이를 진료예약 서비스에 적용함으로써 그 효용성을 밝히고자 하였다.

1. 서론

시맨틱 웹서비스 기술에 관한 연구들을 살펴보면, 서비스 발견(discovery), 선택(selection), 협상(negotiation), 조정(coordination), 조합(composition), 검증(validation), 실행(execution), 모의 실행(simulation) 및 보안(security) 등이 존재하지만, 아직까지는 서비스들의 조합을 위한 발견, 선택, 연결(matchmaking) 등에 관한 연구가 주류를 이루고 있다[2]. 하지만 이와 관련하여 간과되어서는 안 되는 것이 바로 서비스 실행과 관련된 연구이다. 실제로 존재하는 수많은 웹서비스들을 대상으로 시맨틱 기술을 부여하여 조합하는 것은 가능하나, 그러한 통신을 가능하게 하는 기반기술이 인터넷임을 감안할 때, 실제로 서비스 조합단계에서 잘 작성된 서비스라고 할지라도 웹서비스의 특성상 공표된 서비스는 미리 등록되는 과정을 겪게 된다. 이를 통해 검색된 서비스들이 서비스 조합에 이용되는 상황은 실제로 서비스를 실행하는 상황과는 다를 수 있다. 그러므로 의미적 관점에서 이와 같은 문제를 해결하기 위한 연구과정이 요구된다. 본 연구에서는 에이전트 기술에 사용된 BDI 구조 기술을 웹서비스를 실행하는 과정에 적용함으로써 제시된 문제점을 해결하고자 하였다. BDI 구조 기술은 그 특성상 추론의 기능과 함께 환경의 변화에 동적으로 반응함으로써 시맨틱 웹서비스의 실행을 위해 적합한 기술로 이용될 수 있다.

2. 관련 연구

2.1 시맨틱 서비스 조합과 실행

Systems	Base Technique	Choreography						Orchestration	Invocation	Independency
		Online-Planning	Control Constructs	Data Constructs	Runtime Service Binding	Fault-Tolerating	Monitoring			
OWL-S VM	[illegible]	x	o	o	△	x	x	o	o	o
WSMX	[illegible]	x	o	o	o	x	o	o	o	x
IRS-III	[illegible]	x	o	o	o	x	x	o	o	x
METEOR-S	[illegible]	o	o	o	o	o	x	o	o	x
SWEEP	[illegible]	△	o	o	o	o	o	o	o	o

그림 1 시맨틱 웹서비스 플랫폼들의 실행기능에 관한 비교 분석

본 연구는 보건복지부 보건의료기술진흥사업의 지원에 의하여 이루어진 것임. (0412-MI02-0404-0002)

서비스 조합과정은 사용자 입장에서 볼 때, 실제 서비스 호출 이전에 미리 작성된 서비스 명세를 가져야 함을 의미한다. 미리 작성된 서비스 명세는 UDDI와 같은 레지스트리를 통해 사전에 등록된 서비스들을 대상으로 하며, 실시간 상황에서 조합에 참여할 서비스를 찾는 과정은 제외하는 것이 일반적이다. 이와 같이 작성된 서비스 명세는 구성된 조합원리에 의해 차례로 해당 서비스를 호출하는 과정을 통해 최종 목표를 달성하게 된다. 이에 반해 시맨틱 서비스 실행이란 사전에 작성된 서비스 명세의 존재유무는 별개로 하고, 서비스를 호출하는 차원에서 조합된 서비스를 필요로 하는 경우, 단계별로 요구되는 목표를 달성시킬 수 있는 서비스를 찾고, 이를 바로 수행하는 과정을 통해서 최종 목표를 이루게 된다. 이렇게 함으로써 실시간적으로 수행될 서비스의 상태 및 실행 상황의 변화 시에도 능동적으로 서비스를 수행할 수 있는 장점을 갖게 된다. [그림 1]에서는 서비스의 조합과 실행을 통합한 시스템들에 관하여 비교 분석한 자료이다.

2.2 BDI 에이전트 구조

BDI 구조란 [그림 2]와 같이 내부에 기호로 명시된 믿음(belief), 소망(desire), 그리고 의도(intention)들의 집합을 포함한다. 믿음들은 외부 세계나 에이전트 내부 상태에 관한 사실(fact)들이다. 이 사실들은 전통적인 일차 서술 논리로 표현된다. 소망은 에이전트의 목표를 나타낸다. 에이전트는 믿음들과 소망들을 바탕으로 주어진 목표를 달성하거나 특수한 상황에 반응하기 위해 어떤 일련의 동작과 테스트를 하여야 하는가를 나타내는 추상계획인 의도(intention)들를 지닌다.

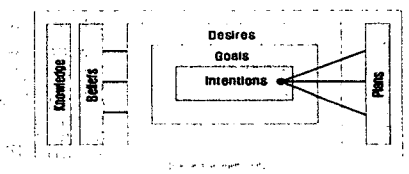

그림 2 BDI 구조

그리고 이들 중에 현재의 지식(믿음과 소망)을 기반으로 활성 가능한 계획(plan)들을 선택하게 되고, 이들을 BDI 구조 내에서 실행기(interpreter)를 통해 실행하게 된다. BDI는 실행 중인 지식영역들에 대해 명시된 바대로 문맥이 지켜지고 있는 지를 수시로 점검하여 문맥이 만족되지 않으면 곧바로 해당 지식영역의 실행을 중단함으로써 환경변화에 신속히 반응할 수 있다. 이와 같은 점들을 고려할 때 BDI 구조는 시맨틱 웹서비스의 조합된 서비스를 실행상황에 잘 맞는 구조임을 알 수 있으며, 특정한 서비스의 오류 시에도 그 당시의 지식을 기반으로 유연하게 작성된 계획 리스트를 통해 대체 가능한 서비스를 선택, 실행할 수 있다는 장점을 갖게 된다[4].

3. 시맨틱 웹서비스 실행기

웹서비스 실행기는 우선 기능적으로 점차 웹서비스의 표준으로 자리를 잡아가고 있는 OWL-S 명세를 지원하고, 실행 시에도 입-출력 데이터에 대하여 온톨로지를 이용하여 추론을 위한 일관성 유지 기능을 지원해야 한다. 그리고 실행 시 동작 상태 등을 감시하여 변화 시 이를 수용할 수 있어야 한다. 또한 OWL-S에서 제공하는 대부분의 제어구조를 지원할 수 있어야 한다.

3.1 시스템 설계

SWEEP(Semantic Web Execution Environment based Plan)은 시맨틱 웹서비스의 동적 서비스 실행을 위해 실행 모듈에 의미적 접근을 시도하고자 한 최초의 연구라 할 수 있다. SWEEP은 에이전트의 내부 구조인 BDI 구조 기술을 적용함으로써 오프라인(off-line) 방식의 계획이 갖는 한계에 대해 제한적이나마 온라인(on-line) 방식의 계획처리 기능을 제공하여 이의 단점을 개선하고자 하였다[1]. 또한 서비스의 계획과 실행을 분리하여 접근함으로써 조합된 서비스들의 검증과 감시 기능까지도 제공할 수 있다. SWEEP 개발을 위해 우리는 BDI 구조의 장점을 가지면서 개발 시 용이함을 위해 지능형 에이전트 구조인 JAM을 사용하기로 하였다. JAM은 계획 기반의 검색기능을 가지며, 계층적인 구조의 하부 계획들을 수행함으로써 전체의 목표를 달성할 수 있는 이점을 지닌다[4].

3.2 SWEEP 구성 요소

SWEEP의 구성요소들을 살펴 보면, 크게 질의부, 실행부, 호출부로 나뉠 수 있다. 질의부에는 GUI 를 갖고, 사용자로부터의 요구가 명세된 OWL-S파일을 읽어 들여서 서비스 수행 목표를 삼는 과정과 최종 모의 실행과정을 통해서 알게 되는 사용자의 정보를 입력 받는 과정, 그리고 최종 서비스 수행결과를 제공하는 과정으로 나뉘며, 이를 수행할 질의 처리기(Query Processor)가 존재한다.

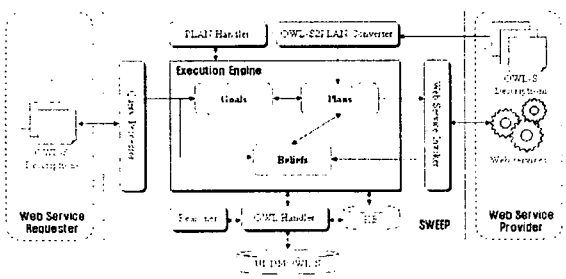

그림 3 SWEEP 구조

실행부에는 사전에 UDDI4OWL-S에 등록된 서비스 명세들에 대해 서비스-계획 변환기에 의해 변환하여 저장소에 보유중인 계획들과, 질의 처리기로부터 입력된 목표 계획을 대상으로 계획들의 조합 및 실행을 담당하는 실행 엔진과 서비스-계획 변환기(OWL-S2Plan Converter), 실제 수행 가능한 계획들에 대한 목록 정보를 담당하는 계획 조정기(Plan Handler), UDDI4OWL-S와의 중개역할 및 엔진에 의해 수행 예정된 OWL 파일들을 관리하고 WSDL 정보를 추출하는 역할을 담당하는 OWL 조정기(OWL Handler), 그리고 온톨로지를 사용하는 경우 추론과정을 담당하는 추론기(Reasoner)와 저장소(KB)가 포함된다. 호출부의 경우, OWL 조정기로부터 넘겨받은 WSDL 정보들을 실제로 호출하여 서비스를 수행하는 서비스 호출기(Web Service Invoker)가 존재한다[그림 3].

3.3 서비스-계획 변환

입력된 서비스 명세를 계획으로 변환하는 과정은 [표 1,2]에서 나타내었다. OWL-S의 경우 인수로서 IOPE를 사용하게 되는데, 이들은 계획에서 헤더영역의 지식변수(FACT), 목표(GOAL), 조건(CONTEXT, PRECONDITION), 영향(EFFECTS)로 변환될 수 있다. 또한 제어구조(Sequence, Split, Split-Join, Any-Order, Choice, If-Then-Else, Iterate, Repeat-Until, Repeat-While)는 Spit와 Split-Join 만을 제외하고 JAM 계획에서 제공되는 행위제어(action-control)들로 표현이 가능하다. [표 1]에서는 실제로 질병 검색 기능을 제공하는 서비스를 나타내었다.

표 1 질병명 검색을 위한 OWL-S 명세

```
<owl:imports rdf:resource=  <-- (가) -->
   "http://127.0.0.1/resources/ontology/Hospital.owl"/>
... <process:Local rdf:ID="UserMethod"> ...
   </process:Local>
 <profile:Profile rdf:ID="DiseaseSearchProfile"> ...
<process:Input rdf:ID="Pain">
    <process:parameterType rdf:datatype=
        "http://www.w3.org/2001/XMLSchema#
         anyURI">http://127.0.0.1/resources/ontology/
         Hospital.owl#Illness</process:parameterType>
   </process:Input> ...
<process:AtomicProcess rdf:ID="DiseaseSearchProcess">
 ... <process:hasOutput>
       <process:Output rdf:ID="Disease"> ...
```

입력된 OWL-S명세를 JAM 계획으로 변환하게 되면 [표 2]와 같이 나타낼 수 있다.

표 2 질병명 검색을 위한 JAM 계획 명세

```
plan: { NAME: "Disease Search"
GOAL: ACHIEVE disease_known "True";
PRECONDITION: RETRIEVE disease_known $known;
          (== $known "False");
BODY: ACHIEVE pain_known "True";
       RETRIEVE pain $pain_name;
       ASSIGN $service_name "DiseaseSearch";
       ASSIGN $servicvice_WSDL
            "http://203.249.22.58/DiseaseSearch?WSDL;
       EXECUTE com.irs.jam.userfunctions.ServiceCall//(나)
            $service_name $service_WSDL $pain_name
            $disease_names;
       WHEN TEST (!= $disease_names "")
{ UPDATE (disease_known) (disease_known "True"); }; }
```

이전 서비스로부터 자동으로 주어지는 변수인 UserMethod는 그 값에 대하여 미리 PRECONDITION부를 통해 현재 계획의

수행가능여부를 판단받게 된다. 그리고 OWL-S로부터 사용된 입력변수인 Pain은 JAM 계획에서 변수로 선언되어 지식 영역(world model)에 놓이게 된다. 계획 수행 결과로서 구해져야 하는 값은 임의의 변수 $disease_name을 통해 실제 원격 프로시져 호출을 담당하는 (나)에서 이용된다. [표 1]의 (가)는 JAM 계획상에서 직접 표현할 방법이 없으므로 OWL API를 이용한 별도의 처리를 통해 이용되어야 한다.

3.4 실행 과정

SWEEP 시스템을 통해 서비스들을 탐색하고 실행해 나가는 과정을 설명하면, 사전에 UDDI4OWL-S에 등록된 기본(atomic) 서비스들은 OWL-S2Plan 변환기를 통해 계획(plan)으로 변환되어 지식베이스에 저장되게 된다. 이때 사용자로부터 OWL-S 명세가 입력되면 역시 변환기를 통해 수행해야 할 목표 계획으로 설정되며, 이를 수행할 수 있는 저장된 계획들의 목록 정보를 읽어 JAM 스크립트 파일이 작성된다. 이와 같이 작성된 스크립트 파일은 BDI 엔진에 의해 전개되면서 목표 달성을 위한 하부 계획들을 호출하고 최종적으로 수행과정까지 담당하게 된다. 이때 오류가 발생한 계획에 대해서는 대체 가능한 계획이 존재하는 경우 대체된다. BDI 구조의 JAM 계획을 이용하여 웹서비스들을 실행할 때 최종 결정되어 수행된 계획, 즉 OWL-S 명세를 얻을 수 없다는 단점이 존재하게 되는데 이는 앞으로 해결해야 할 과제이다.

4. 구현 및 응용

현재 개발 중인 SWEEP의 주요 기능을 살펴보면, 사용자 화면을 통해 달성하고자 하는 서비스 목표를 직접 입력하도록 하였고, UDDI4OWL-S를 통해 갱신된 서비스 목록을 가져와서 사용자가 현재 수행 가능한 서비스 정보들을 확인할 수 있도록 하였으며, JAM 계획으로 바꾸는 기능을 제공한다. 또한 입력된 목표는 자동화된 서비스 탐색 및 온톨로지 추론 과정을 통해서 적절한 계획들을 선택하고 이들을 연결하며 또한 실행 상태를 GUI를 통해 확인할 수 있다[그림 4]. 우리는 이와 같이 개발중인 SWEEP을 진료예약 서비스에 적용하고자 하였다.

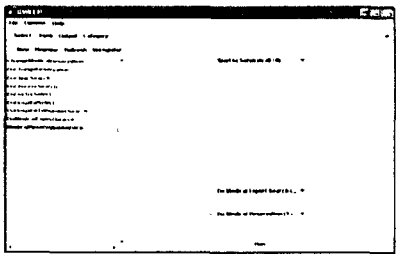

그림 4 SWEEP 실행 화면

진료예약 서비스는 인터넷을 기반으로 하는 프로세스로 한정한다. 이를 단계별로 나눠보면, 질병검색을 통한 진료 예약, 사용자의 직접 선택에 의한 진료 예약, 전화를 통한 진료 예약, 3가지로 구별될 수 있다. 이들 3가지를 각각 수행해야 할 목표로 가정할 때 이들을 이루기 위한 세부적인 기본 서비스들 및 서비스 연결을 위한 IOPE를 표현하면 [그림 5]와 같다. 질병검색을 통한 진료 예약 서비스를 사용자가 선택한 경우, 시스템은 질병 검색 서비스를 이용하는 진료예약이라는 목표를 설정하게 되고, 목표 달성을 위해 요구되는 IOPE를 고려하여 의료전문가 및 해당 전문가가 근무하는 병원을 출력으로 제공하는 서비스를 찾게 된다. 이 서비스를 위해서 질병 이름과 해당 진료과를 출력을 제공하는 서비스를 찾게 되며, 이는 다시 질병명을 출력으로 하는 서비스를 찾고, 최종적으로 환자의 증상을 입력으로 하는 서비스를 검색해냄으로써, 사용자에게 자신의 증상 정보만을 입력하도록 요구하고 이를 통해 진료예약 서비스를 위한 각 절차 프로세스들을 수행하게 된다. 각 서비스들은 수행 이전에 사용자에게 서비스 수행을 위한 입력 정보를 줄 것인지를 질의함으로써 계속적으로 하부 서비스로 분기하는 과정을 단축할 수 있다. 또한 조합된 서비스의 실행 오류 시, 대체 서비스를 구동시킴으로써 안정적으로 사용자의 요구를 충족시킬 수 있다.

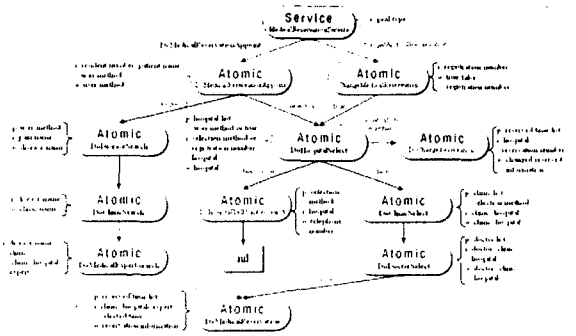

그림 5 진료 예약 서비스 제어 흐름도

우리는 SWEEP이 진료예약 시뮬레이션 과정을 통해 에러 상황에서 안정적으로 동작하는 것을 확인할 수 있었다.

5. 결론

지금까지 우리는 안정적인 시맨틱 웹서비스 실행을 위한 방법론적 접근을 시도하였고, 이를 위한 실행기(SWEEP) 개발에 관하여 기술하였다. SWEEP은 시뮬레이션 과정을 통해 유용하게 사용될 수 있음을 알 수 있었다. 이를 토대로 차후의 시맨틱 웹서비스 통합 시스템을 위한 보다 구체적인 구성 요소 및 기능을 설계할 수 있을 것으로 기대된다. 현재 SWEEP 시스템은 JAVA언어를 이용해 프로토타입 수준으로 개발 중이며 향후 계획 중인 자동화된 실시간 웹서비스 조합기와의 연동을 통해, 현재 참여중인 과제인 온톨로지 기반의 의료 웹서비스 에이전트 통합 플랫폼 개발을 위한 기초 연구로 이용될 예정이다.

참고 문헌

[1] Hoon Jin, In-Cheol Kim, "Plan-based Coordination of Multi-Agent System for Protein Structure Prediction, *LNAI 3397*, pp. 224-232, AIS 2004.

[2] L. Cabral, J. Domingue, E. Motta, T. Payne and F. Hakimpour, " Approaches to Semantic Web Services: An Overview and Comparisons" , *Proc. of the First European Semantic Web Symposium (ESWS2004)*, Heraklion, Crete, Greece. 10-12 May 2004.

[3] M. Paolucci, A. Ankolekar, N. Srinivasan, and K. Sycara, " The DAML-S Virtual Machine" , *In 2nd ISWC2003*, Sep 2003.

[4] Marcus J. Huber, " JAM: A BDI-theoretic Mobile Agent Architecture" , *Proc. of the Third International Conference on Autonomous Agents (Agents'99)*, pp236-243, Seattle, WA, May 1999.

[5] M. Zaremba, C. Bussler: Towards Dynamic Execution Semantics in Semantic Web Services, *Proc. of the Workshop on Web Service Semantics, WWW2005*, Chiba, Japan, 2005.