제26회 한국정보처리학회 추계학술발표대회 논문집 제13권 제2호 (2006. 11)

SemanticBPEL: 시멘틱과 워크플로우의 혼합 시스템

이용주
상주대학교 컴퓨터공학과
e-mail:yongju@sangju.ac.kr

SemanticBPEL: A Hybrid System of Semantic and Workflow

Yong-Ju Lee

Dept of Computer Engineering, Sangju National University

요 약

본 논문에서는 자동화된 웹 서비스 조합을 구현하기 위해 BPEL 기법에 OWL-S 기법을 도입하는 혼합 조합기법을 제안한다. BPEL 조합기법은 에러 처리나 트랜잭션 관리와 같은 비즈니스 환경에서 요구되는 실질적인 전체 기능을 지원하고 있으나, 주된 단점은 정적 조합 기법으로써 서비스 선택 및 워크플로우 관리가 사전에 수동으로 이루어져야만 한다. 반면에, OWL-S 조합기법은 기계 가독형으로 웹 서비스 기능을 묘사할 수 있는 온톨로지를 사용함에 따라 호환 가능한 웹 서비스들 간의 동적 통합 및 웹 서비스 발견이 가능하나, 이러한 기법은 아직 연구 중에 있으며 실제 적용을 위해서는 BPEL의 상용기법이 요구된다. 따라서 본 연구에서는 BPEL4WS와 OWL-S 혼합 시스템인 SemanticBPEL의 구조를 설계하고, 웹 서비스 탐색 알고리즘을 제안한다. 그리고 SemanticBPEL의 프로토타입 시스템을 실제 구현한다.

1. 서론

최근 웹 서비스 기술의 출현으로 분산 애플리케이션 통합 문제에 대한 차세대 솔루션인 서비스 지향 아키텍처(SOA: Service Oriented Architectures)의 구현이 가능하게 되었다. SOAP, WSDL, UDDI, 그리고 BPEL4WS와 같은 표준 웹 서비스 기술들은 SOA에서 요구되는 이기종 분산 플랫폼 상의 복잡한 애플리케이션 통합 문제를 유연하게 해결할 수 있는 다양한 인터페이스 기술들을 제공하고 있다.

그렇지만, 웹 서비스 기술의 활발한 도입과 관심에도 불구하고 아직까지는 기업의 내부 통합 프로젝트에서만 이들이 사용되고, 외부 서비스들이 '온 디맨드(on demand)' 방식으로 연결되는 엔터프라이즈급 IT 플랫폼에서는 아직 활용되지 못하고 있는 실정이다. 이러한 주된 이유는 현재의 웹 서비스 기술들이 동적인 웹 서비스 발견 및 통합, 즉 자동화된 웹 서비스 조합에 대한 적절한 기법을 제공하지 못하기 때문이다. 현재의 기술들은 자동화를 위한 시멘틱 개념이 부족하여 프로그래머에 의한 수동으로 모든 작업을 수행하고 있다.

웹 서비스 조합에 관한 연구는 크게 두 가지 방향으로 추진되고 있다. 첫째는 IBM, 마이크로소프트, BEA 등 컴퓨터 대형 업체들이 주도적으로 추진하고 있는 비즈니스 워크플로우 관리 이론을 기반으로 한 웹 서비스 조합기법이다. 비즈니스 프로세스 실행 언어(BPEL: Business Process Execution Language)[1]를 사용하는 이러한 기법들은 에러 처리나 트랜잭션 관리와 같은 비즈니스 환경에서 요구되는 실질적인 전체 기능을 지원하고 있다. 그렇지만 이러한 기법의 주된 단점은 정적 조합 기법으로써 서비스 선택 및 워크플로우 관리가 사전에 수동으로 이루어져야만 한다.

두 번째 접근 방법은 주로 학계에서 추진되고 있는 OWL-S와 같은 시멘틱 웹을 기반으로 한 웹 서비스 조합기법이다. OWL-S[2]는 자동적인 웹 서비스 발견 및 통합을 실현하기 위해 기계 가독형으로

웹 서비스 기능을 묘사할 수 있는 메카니즘, 즉 온톨로지(ontology)를 사용한다. 이에 따라 호환 가능한 웹 서비스들 간의 동적 통합이 가능하고, 웹 서비스 조합 실행 시에 웹 서비스 발견이 가능하다. 그러나 이러한 기법은 아직 연구 중에 있으며, 실제 적용을 위해서는 BPEL4WS 스타일의 워크플로우 실행계획 및 엔진이 필요하다.

본 연구에서는 자동화된 웹 서비스 조합을 구현하기 위해 BPEL 기법에 OWL-S 기법을 도입하는 하나의 혼합(hybrid) 조합시스템인 SemanticBPRL 시스템을 제안한다. BPEL 조합기법(예, BPEL4WS)은 실제 비즈니스에 널리 사용되고 있으나, 정적 기법이어서 수동으로 조합을 수행해야 하므로 작업하기 어렵고, 시간이 많이 소비되며, 에러가 발생되기 쉽다. 이를 보완하기 위해 OWL-S 조합기법을 적용하면 동적으로 웹 서비스 발견 및 통합이 가능할 수 있다.

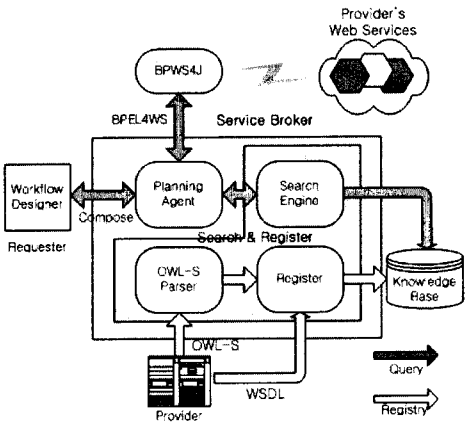
본 논문의 구성은 다음과 같다. 2장에서 본 시스템의 전체 구조 및 온톨로지 개념을 이용한 웹 서비스 탐색 알고리즘을 제안한다. 3장에서 시스템 구현에 관해 서술하고, 4장에서 결론을 내렸다.

2. BPEL4WS와 OWL-S의 혼합 시스템

2.1 시스템 구조

BPEL4WS와 OWL-S 혼합기법을 적용한 SemanticBPEL 시스템의 전체적인 구조는 (그림 1)와 같으며, 서비스 공급자(provider), 서비스 사용자(requester), 그리고 서비스 중개자(service broker)로 구성되어 있다. 서비스 중개자는 계획 에이전트(planning agent)와 웹 서비스 탐색(search) 및 등록(register) 모듈로 이루어져 있으며, 중개자는 SemanticBPEL 시스템의 미들웨어(middleware) 역할을 수행한다. SemanticBPEL 시스템은 본 연구실에서 이미 개발되어 있는 ASP(Application Service Provider)를 위한 워크플로우 기반 서비스 중개자 시스템인 IMP(Internet MarketPlaces)[3]를 기반으로 확장 발전시켰다. (그림 1)에 대한 자세한 설명은 다음과 같다.

- 서비스 공급자: 웹 서비스 메타 정보를 등록한다. OWL-S와 WSDL로 작성된 웹 서비스 객체를 등록하기 위해 등록 모듈을 엑세스 한다.
- 서비스 사용자: 계획 에이전트를 접속하여 워크플로우를 작성하고 이를 실행한다.
- 계획 에이전트: 탐색엔진을 이용하여 웹 서비스를 검색하고, BPEL4WS 스팩을 생성하여 BPWS4J 엔진에 의해 이들 워크플로우 문서를 등록(deploy)한다.
- 웹 서비스 탐색 및 등록 모듈: OWL-S 파서, 웹 서비스 등록기, 탐색엔진으로 구성되어 있으며, 웹 서비스 등록 및 탐색을 수행한다.
- BPWS4J 엔진[4]: IBM alpahWorks로부터 이용 가능한 BPEL4WS 구현 엔진이다.



(그림 1) SemanticBPEL 시스템

SemanticBPEL 시스템에서 공급자는 메타 데이터를 사용하여 웹 서비스를 등록할 수 있어야 하고 사용자는 계획 에이전트를 이용하여 웹 서비스 조합 및 실행을 수행할 수 있어야 한다. 공급자는 자신의 웹 서비스를 등록하기 위해 등록 모듈을 접속한다. OWL-S로 작성된 웹 서비스 객체는 OWL-S 파서에 의해 파싱된 후 웹 서비스 등록기에 전달되고, WSDL은 바로 웹 서비스 등록기에 전달된 후 지식베이스(Knowledge Base)에 저장된다. 한편, 사용자는 워크플로우 디자이너를 사용하여 워크플로우를 작성하고 계획 에이전트로부터 이를 BPEL4WS 스팩으로 변환한 후 BPWS4J 엔진을 통해 워크플로우를 등록하고, 최종적으로 웹 서비스 조합을 실행한다.

이러한 처리과정에서 SemanticBPEL 시스템에서는 기존 워크플로우 시스템에서는 볼 수 없었던 새로운 웹 서비스 탐색 및 통합 문제의 해결을 요구한다. SemanticBPEL 시스템에서는 전통적인 워크플로우 시스템과는 달리 인터넷상의 수 없이 많은 웹 서비스들 중에서 가장 적당한 것을 찾아야 하는데 이는 너무 시간이 많이 소비되고 지루한 일이 된다.

따라서 수동으로 이러한 작업을 수행하는 것은 거의 불가능하게 보이며, 이를 효율적으로 지원할 수 있는 새로운 웹 서비스 탐색 메카니즘이 제공되어야만 한다.

2.2 웹 서비스 탐색 알고리즘

새로운 웹 서비스를 탐색할 때 사용자는 먼저 질의 템플릿 Q를 작성한다. 일단 Q가 생성되면 이는 웹 서비스 탐색 모듈로 보내지고, Q와 기존 웹 서비스들 간의 유사도 측정에 따라 우선순위가 매겨진 웹 서비스들이 반환된다. 이때 사용자는 그가 의도한 바를 가장 잘 이룰 수 있는 적합한 웹 서비스를 선택한다. Q는 다음과 같이 표현된다.

$Q = \{ N, D, Is, Os, [Ps, Es, C, R, V] \}$

여기서, N은 웹 서비스 이름, D는 텍스트 설명, Is는 입력 항목, Os는 출력 항목, Ps는 전제조건, Es는 효과, C는 카테고리, R은 품질 등급, V는 서비스 매개변수를 표시한다. [ ]는 선택사항이다.

웹 서비스 탐색 알고리즘의 기본적인 원리는 질의 템플릿의 입력항목을 사용하여 원하는 출력을 산출해 낼 수 있는 웹 서비스들을 찾는 것이다. 이를 위해서 선택되는 웹 서비스는 반드시 템플릿의 출력항목을 포함하고 있어야만 하고, 이 서비스의 입력항목들은 템플릿의 입력항목에 포함되어 있어야만 한다. 이러한 원리를 기반으로 웹 서비스 탐색 알고리즘을 작성하면 (그림 2)과 같다. (그림 2)에서 discovery( ) 함수는 질의 템플릿 Q를 지식베이스에 있는 모든 웹 서비스 S들과 비교한다. 만일 매치가 발견되면 기록되고 우선순위에 의해 정렬된다. matching( ) 함수는 먼저 템플릿 출력항목을 웹 서비스 출력항목과 비교하여 유사도를 계산하고, 매치가 실패하지 않는다면 반대로 웹 서비스 입력항목과 템플릿 입력항목을 비교한다.

```
discovery(Q) {
    for all (S in KnowledgeBase) do {
        if matching(Q, S) then result.append(S) }
    return sort(result)
}
matching(Q, S) {
    outputMatch(Q{Os}, S{Os})
    inputMatch(S{Is}, Q{Is})
}
```

(그림 2) 웹 서비스 탐색 알고리즘

위 알고리즘에서 어떤 서비스들은 정확하게 매치되어 최종 결과로 선택되어 질 수도 있지만, 정확하지 않더라도 무조건 탈락되지 않고 상호 포함관계에 따라 우선순위가 정해질 수도 있다. 따라서 본 논문에서는 유사성 측정기법을 적용하여 우선순위별로 정렬하여 가장 높은 점수를 얻은 웹 서비스를 최종 결과로 선택하도록 한다. 온톨로지 개념 간 유사성 관계는 <표 1>과 같이 5가지 경우가 발생될 수 있다.

<표 1> 온톨로지 개념 간 유사성 관계

	조건
1	같은 경우(Q{I} = S{I})
2	S{I}가 Q{I}의 상위 개념인 경우(S{I} subsumes Q{I})
3	Q{I}가 S{I}의 상위 개념인 경우(Q{I} subsumes S{I})
4	Q{I}와 S{I} 사이에 일부분 공통된 속성이 있는 경우 (Q{I} intersect S{I})
5	Q{I}와 S{I} 사이에 전혀 관계가 없는 경우(Q{I} ≠ S{I})

위의 5가지 경우를 고려한 유사성 측정 알고리즘은 다음과 같이 표현되고 각 경우에 따라 유사도 Ω 값이 반환된다.

```
Similarity(Q, S) {
    If Q{I} = S{I} then return Exact
    If S{I} subsumes Q{I} then return PlugIn
    If Q{I} subsumes S{I} then return Subsumes
    If Q{I} intersect S{I} then return Intersect
    otherwise return Fail
}
```

여기서,

$$\Omega = \begin{cases} 1 & \text{if } Exact \\ 1 & \text{if } PlugIn \\ \dfrac{|p(Q)|}{|p(S)|} & \text{if } Subsumes \\ \dfrac{\partial}{\{|p(Q)|-\partial\}+\{|p(S)|-\partial\}+\partial} & \text{if } Intersect \\ 0 & \text{if } Fail \end{cases}$$

| p(Q) | 는 Q 속성의 개수, | p(S) | 는 S 속성의 개수, $\partial = |p(Q) \cap p(S)|$ 를 의미한다.

3. 시스템 구현

SemanticBPEL 프로토타입 시스템은 자바 언어(JDK1.5)로 구현되었으며, 지원 툴들은 모두 오픈소스를 선택하였다. 서버 쪽 웹 컨테이너로 아파치 톰캣 5.5를 이용하였고, 웹 서비스 컨테이너로는 아파치 Axis 1.3을 사용하였다. 또한 BPEL4WS 실행을 위해서 IBM의 BPWS4J 엔진이 사용되었으며, 워크플로우 작성 시 요구되는 그래프 구현을 처리하기 위해 opengraph graphing 팩키지[5]가 사용되었다. 마지막으로 OWL-S 파서 및 온톨로지 등록 및 탐색을 처리하기 위해 Maryland 대학의 OWL-S API 모듈[6]을 이용하고 있다. 이런 툴들을 기반으로 한 시스템 구현 작업은 IBM의 Eclipse 개발 환

경에서 수행되었다.

SemanticBPEL 프로토타입 시스템은 클라이언트-서버 프로그래밍 구조로 구현되어 있다. 클라이언트 측 워크플로우 디자이너는 "드래그 & 드롭" 인터페이스 방식으로 워크플로우를 생성한다. 이를 구현하기 위한 클라이언트 프로그래밍 기술은 자바 애플릿을 선택하였다. 자바 애플릿은 사용자가 웹 브라우저로 웹 사이트를 접속할 때 사용자의 PC로 프로그램이 자동으로 다운로드된 후 실행되기 때문에 서버의 로드를 줄여줄 수 있다. 먼저 질의 템플릿의 입력항목이 주어지면 (그림 2)의 웹 서비스 탐색 알고리즘을 적용하여 이용 가능한 웹 서비스 리스트를 보여주고, 이들 중 가장 적합한 웹 서비스를 선택한다. 이러한 방식으로 시작 단계로부터 점차적으로 하나씩 새로운 웹 서비스가 첨가되고 나면 최종적으로 웹 서비스 워크플로우가 완성된다. 이때 워크플로우 디자이너는 워크플로우를 쉽게 생성할 수 있도록 사용자 편의성 인터페이스를 지원한다. 일단 워크플로우가 완성되고 나면 이를 서버 쪽에 전송하여 BPEL4WS 스팩으로 변환하고 BPWS4J 엔진에 의해 등록된다. 그러고 난 후 최종적으로 클라이언트에서 워크플로우 프로세스 실행을 서버 측에 요청한다.

서버 측 계획 에이전트는 자바 서블릿으로 구현된었다. 서블릿은 플랫폼 독립적인 서버 측 자바 프로그램이다. 계획 에이전트는 2.2장에서 기술된 웹 서비스 탐색 알고리즘을 수행하고 클라이언트에서 작성된 워크플로우를 BPEL4WS 스팩으로 변환하는 책임을 가지고 있다. 일단 BPEL4WS 문서가 생성되고 나면 BPWS4J 엔진에 의해 이들 문서가 등록된다. 하지만 IBM의 BPWS4J 엔진은 현재 관리자 툴을 통해서만 등록을 허용하기 때문에 프로그래밍 인터페이스가 가능하도록 엔진 proxy를 만들어야만 한다. 이상과 같은 클라이언트-서버 프로그램의 전체 처리과정을 정리하면 (그림 3)과 같다.

```
ClientApplet {
    Display main menu;
    Enter input parameter;
    Retrieve available web services (by server);
    Construct a workflow;
    Convert from a workflow into BPEL4WS (by server);
    Submit the result;
}

AgentServlet {
    case ("retrieving") {
        Perform searching algorithm;
    }
    case ("converting") {
        Transform a workflow into BPEL4WS;
        Deployed by BPWS4J;
    }
}
```

(그림 3) 클라이언트-서버 프로그래밍 처리 과정

4. 결론

본 논문에서는 자동화된 웹 서비스 조합을 지원하기 위해 BPEL4WS에 OWL-S 기법을 도입하는 혼합 조합기법을 제안하였다. BPEL4WS와 OWL-S 혼합 시스템인 SemanticBPEL의 전체적인 구조를 설계하고, 웹 서비스 탐색 알고리즘을 제안하였다. 특히, SemanticBPEL 시스템을 구현하기 위해 하나의 프로토타입 시스템을 실제 개발하였으며, 본 프로토타입 시스템은 동적 워크플로우 생성 및 등록 기능을 시연하고 사용하기 쉽고 재사용 가능한 기능을 보여주고 있다.

참고문헌

[1] Curbera F., Goland Y., Klein J., Leymann F., Roller D., Thatte S., and Weerawarana S., Business Process Execution Language for Web Services, Version 1.0, http://www-106.ibm.com/developerworks/webservices/library/wsbpel/, 2001

[2] OWL Services Coalition, OWL-S: Semantic Markup for Web Services, OWL-S White Paper, http://www.daml.org/services/owl-s/1.0/owl-s.pdf, 2004

[3] 이용주, "동적 웹 서비스 조합을 위한 시멘틱 웹 서비스 발견 및 실행 기법," 한국정보처리학회논문지D, 제12-D권 제6호, 889-898, 2005

[4] IBM, BPWS4J, http://www.alphaWorks.ibm.com/tech/bpws4j

[5] Openjgraph, http://sourceforge.net/projects/openjgraph

[6] E. Sirin, J. Hendler, and B. Parsia, "Semi-automatic Composition of Web Services using Semantic Description," Web Services: Modeling, Architecture and Infrastructure Workshop in Conjuction with ICEIS, 2003