DOI: 10.3745/KIPSTD.2008.15-D.4.569

반자동 웹 서비스 조합을 위한 WS-BPEL과 OWL-S의 융합 시스템

이 용 주†

요 약

웹 서비스는 현재 서비스 지향 아키텍처(SOA)를 구현하기 위한 가장 유망한 기술이다. 그렇지만 웹 서비스에 대한 많은 관심에도 불구하고 내부 통합 프로젝트에서만 사용되어지고, 파트너들이 '온 디맨드(on demand)' 방식으로 결합되는 가상 엔터프라이즈 환경에서는 아직 활용되지 못하고 있는 실정이다. 이러한 주된 이유는 현재의 웹 서비스 기술들이 동적인 웹 서비스 발견 및 통합에 대한 적절한 기법을 제공하지 못하기 때문이다. 본 논문에서는 반자동 웹 서비스 조합 시스템을 구현하기 위해 WS-BPEL 기법과 OWL-S 기법의 장점만을 채택한 새로운 SemanticBPEL 조합 기법을 기술한다. 특히, 동적 웹 서비스 발견 및 통합 문제를 해결하기 위해 다단계 웹 서비스 탐색 방법을 제안한다. 이 방법은 실험 분석을 통해 기존의 키워드 기반 검색 방법보다 성능이 우수함을 보인다.

키워드 : 웹 서비스 조합, 다단계 탐색 방법, SOA, OWL-S, WS-BPEL

A Fusion System of WS-BPEL and OWL-S for Semi-Automatic Composition of Web Services

Yong-Ju Lee†

ABSTRACT

Web services are the current most promising technology for service oriented architecture(SOA) implementations. However, in spite of the large scale acceptance of web services, they have been relegated to internal integration projects, and the grand vision of virtual enterprises where partners can be integrated on demand is yet to be realized. The main reason is that the current standards of web services are not very suitable for the dynamic web service discovery and integration. In this paper, we present a novel SemanticBPEL solution that merges the benefit of WS-BPEL, with the advantage of OWL-S for building a semi-automatic web service composition system. In particular, this work proposes a multi-phase search method for solving dynamic discovery and integration problems of web services. The proposed method is compared with the existing keyword based retrieval method. These comparisons show that our approach outperforms the existing method.

Key Words : Web Services Composition, Multi-Phase Search Method, SOA, OWL-S, WS-BPEL

1. 서 론

최근 IT 업계 전반에서 SOA(Service Oriented Architecture)는 큰 화두가 되고 있다. 이는 SOA가 현재의 비즈니스 환경변화에 가장 적절히 대응할 수 있는 최적의 IT 전략이라고 보고 있기 때문이다. SOA는 오늘날과 같은 급속한 환경변화에 적시에 적응할 수 있는 유연성 있는 IT 시스템으로써 조직 내·외부의 프로세스와 애플리케이션들을 각각 서비스라는 기본적인 기능들로 나누어 이들 서비스들을 조립하여 원하는 기능을 구성한다. 하지만 SOA는 소프트웨어 아키텍처로 이러한 아키텍처를 구현하기 위한 구체적인 기술들의 모음은 웹 서비스(web services)라 할 수 있다. 웹 서비스는 XML, SOAP, WSDL, UDDI, BPEL 등의 표준을 기본 구조로 하고 있으며 이러한 기술들은 SOA에서 요구되는 이기종 분산 플랫폼 상의 복잡한 애플리케이션 통합 문제를 유연하게 해결할 수 있는 다양한 인터페이스를 제공하고 있다.

그렇지만, 오늘날 웹 서비스의 눈부신 발전과 활발한 도입에도 불구하고 아직까지는 기업의 내부 통합 프로젝트에서만 이들이 사용되고, 외부 서비스들이 '온 디맨드(on demand)' 방식으로 연결되는 가상 엔터프라이즈 환경에서는 아직 활용되지 못하고 있는 실정이다[1]. 이러한 주된 이유는 현재

† 종신회원 : 경북대학교 이공대학 컴퓨터공학과 부교수
논문접수 : 2008년 3월 12일
수 정 일 : 1차 2008년 5월 22일
심사완료 : 2008년 6월 13일

의 웹 서비스 기술들이 동적인 웹 서비스 발견 및 통합, 즉 반자동화된 웹 서비스 조합에 대한 적절한 기법을 제공하지 못하기 때문이다. 현재의 기술들은 자동 또는 반자동화를 위한 시맨틱 개념이 부족하여 프로그래머에 의한 수동으로 모든 작업이 수행되고 있다.

현재 웹 서비스 조합에 관한 연구는 크게 두 가지 방향으로 추진되고 있다. 첫째는 IBM, Microsoft, BEA 등 대형 컴퓨터 업체들이 주도적으로 추진하고 있는 비즈니스 프로세스 관리(BPM: Business Process Management) 기법을 기반으로 한 웹 서비스 조합이다. WS-BPEL(Web Services Business Process Execution Language)[2]과 같은 프로세스 실행 언어를 사용하는 이러한 기법들은 예외 상황이나 트랜잭션 처리와 같은 비즈니스 환경에서 요구되는 실질적인 전체 기능을 지원하고 있다. 그렇지만 이러한 기법의 주된 단점은 정적 조합 기법으로써 서비스 선택 및 프로세스 관리가 사전에 수동으로 이루어져야만 한다. 이기종 분산 환경에서 수동으로 이러한 작업을 수행하는 것은 너무 시간이 많이 소비되고 복잡한 작업이 요구된다.

둘째는 주로 학계에서 추진되고 있는 OWL-S(Web Ontology Language-Services)[3]와 같은 시맨틱 웹을 기반으로 한 웹 서비스 조합이다. OWL-S는 자동적인 웹 서비스 발견 및 통합을 실현하기 위하여 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 서비스 실시간 탐색이 가능하다. 그러나 이러한 기법은 아직 성숙된 기술은 아니며, 실제 적용을 위해서는 BPWS4J[4]와 ActiveBPEL[5]과 같은 BPEL 스타일의 워크플로우 실행엔진이 필요하다.

본 연구에서는 반자동 웹 서비스 조합 시스템을 구현하기 위해 위에 기술한 WS-BPEL 기법과 OWL-S 기법의 장점만을 채택한 새로운 SemanticBPEL 조합 기법을 제안한다. WS-BPEL 기법은 실제 비즈니스에 널리 사용되고 있으나, 정적 기법이어서 수동으로 조합을 수행해야 하므로 작업하기 어렵고, 시간이 많이 소비되며, 에러가 발생되기 쉽다. 이를 보완하기 위해 OWL-S 기법을 적용하면 동적으로 웹 서비스 발견 및 통합이 가능할 수 있다.

본 연구와 유사한 반자동 웹 서비스 조합에 관한 연구는 METEOR-S 프로젝트[6, 7]에서 찾아볼 수 있다. METEOR-S는 워크플로우 기반 시맨틱 웹 서비스 조합 시스템으로서 시맨틱 템플릿을 작성하고 실행 프로세스를 만들어 이를 실행 엔진에서 수행하는 일련의 과정을 구현하고 있다. 그러나 이 시스템은 실제 비즈니스 적용보다는 이론에 더 많이 치중된 연구로서 엔터프라이즈 플랫폼에서는 아직 활용되지 못하는 실험실 수준인 단점이 있다. WS-Composer[8]는 OWL-S를 사용한 반자동 웹 서비스 조합 시스템으로써, 조합을 수행하는 동안에 자체 탐색 알고리즘을 적용하여 목표 지향적인 대화식 조합 기법을 제안하고 있다. 그러나 이 시스템은 WS-BPEL은 고려하지 않았기 때문에 비즈니스 전반적인 워크플로우 처리 능력은 제공하지 못하고 있다. 스탠포드 대학 KSL 연구실에서는 웹 서비스 탐색 중계자인 SDS(Semantic Discovery Service)를 구현하였다[9]. SDS에서는 동적 웹 서비스 발견을 위해 WS-BPEL를 중심으로 OWL-S 개념을 도입한 보텀-업(bottom-up) 방식을 제안하였다. 그러나 이 논문에서는 동적 웹 서비스 발견에 대한 방법론은 제시하고 있으나 본 연구에서 수행하고자 하는 웹 서비스 조합 기법에 대한 언급은 없다.

본 연구에서는 기존의 연구들과는 달리 워크플로우 구성 및 질의 템플릿 작성, 동적 웹 서비스 발견 및 통합, 그리고 워크플로우 자동실행 등 비즈니스 프로세스 수명주기(life cycle) 전체 과정을 다루고 있다. 특히, 반자동 웹 서비스 조합 시스템을 구현하기 위해 본 논문에서는 다단계(multi-phase) 웹 서비스 탐색 방법을 새로 제안한다. 이 방법에서는 1단계로 분류코드에 따라 후보 웹 서비스 집합을 필터링(filtering)하고, 2단계로 구문 및 시맨틱 매칭 방법에 의해 우선순위별 서비스 리스트를 출력하며, 3단계로 QoS(Quality of Service) 정보를 이용하여 이들을 정제(refinement)하여 최종 결과물을 산출한다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 살펴보고 3장에서 하나의 가상 시나리오를 설명한다. 4장에서 시스템 구조를 소개하고 5장에서 웹 서비스 발견 및 통합 기법을 기술한다. 그리고 6장에서 워크플로우 실행 과정을 서술하고 7장에서 실험 및 분석을 수행한다. 마지막으로 8장에서 결론을 내린다.

2. 관련연구

웹 서비스 조합에 관한 연구는 두 가지 방향, 즉 BPM 기반과 시맨틱 기반 웹 서비스 조합 기법으로 진행되고 있다. 이 장에서는 이들에 대한 기술 동향을 먼저 기술하고, 본 시스템과 유사한 관련연구들을 살펴본다.

2.1 BPM 기반 웹 서비스 조합 기법

BPM은 실시간으로 비즈니스 프로세스를 변경하고, 개별적으로 관리되었던 비즈니스 프로세스를 수명주기 전체에 걸쳐서 일관성 있게 관리하기 위한 경영관리 기법이다[10]. 전통적인 BPM은 프로세스 가시화와 자동화 측면에서는 매우 매력적이었지만, 조직 내·외부의 다양한 자원에 대한 유연성은 떨어졌고 재사용성도 비교적 약한 모습을 보여줬다. 그러나 최근의 SOA 기반 BPM에서는 프로세스 가시성과 자동화를 재고하면서도 유연하게 자원들을 사용할 수 있고 재사용의 장점도 제공하고 있다.

현재의 BPM 표준들은 BPM 핵심기능이라 할 수 있는 프로세스 모델링(예, BPMN, UML), 실행(예, WS-BPEL, XPDL, WSCI), 그리고 모니터링(예, BPQL, BAML)에 초점이 맞추어져 있다. 이들 중 웹 서비스와 관련이 있는 표준은 WS-BPEL이며, WS-BPEL은 기본적으로 비즈니스 프로세스를 구현하는 언어이다. SOA 기반 BPM에서는 그래픽 모델링 툴에서 비즈니스 프로세스를 모델링하면 이 그래픽 데이터는 웹 서비스들의 조합인 WS-BPEL로 변환되고 BPM

실행엔진에 의해 프로세스가 실행된다.

WS-BPEL이 비즈니스 커뮤니티에서 성공할 수 있었던 가장 큰 요소는 첫째, 비즈니스 분야에서 성공적으로 활용되고 있는 기존의 워크플로우 관리 기법을 바탕으로 개발되었기 때문에 비즈니스 프로세스의 상호 관계를 모델링하기가 매우 쉬웠으며, 둘째로는 IBM, Microsoft, BEA, SAP 등 주요 IT 벤더들이 자사의 많은 관련 제품들을 WS-BPEL 기반으로 구성함에 따라 다른 기술들보다 산업 전반에 빠르게 전파될 수 있었다. WS-BPEL이 SOA 오케스트레이션(orchestration) 언어가 되기에는 아직 부족한 점이 많음에도 불구하고 이젠 많은 유명 연구기관들로부터 확고한 표준으로 보고되고 있다.

2.2 시맨틱 기반 웹 서비스 조합 기법

OWL-S는 가장 최근에 발표된 시맨틱 기반 웹 서비스 언어이다. OWL-S 온톨로지는 기계 가독형으로 웹 서비스 기능을 묘사할 수 있는 메카니즘을 제공한다. 이 메카니즘은 자동화된 웹 서비스 발견 및 통합을 가능하게 만든다.

OWL-S는 서비스를 기술하기 위해 presents, describedBy, supports의 세 가지 속성을 선언하고 있으며, 이는 서비스 프로파일(Profile), 서비스 모델(Model), 서비스 그라운딩(Grounding)의 세 가지 클래스로 구성된다. 즉 OWL-S는 기계가 처리할 수 있는 형태로 서비스를 표현하기 하기 위해 세 가지 클래스를 이용하여 서비스 온톨로지를 기술한다. 서비스 프로파일은 에이전트가 서비스를 탐색하는데 필요한 정보를 제공하고 있으며, 서비스 모델 및 그라운딩은 에이전트가 서비스를 사용할 수 있도록 충분한 정보를 제공하고 있다. 본 연구에서는 서비스 탐색 및 통합 문제를 해결하기 위해 주로 서비스 프로파일을 이용한다.

서비스 프로파일 클래스를 자세히 살펴보면 서비스 제공자에 대한 정보(what organization provides the service), 서비스를 처리하기 위한 기능에 대한 정보(what function the service computes), 그리고 서비스의 특성을 구체화한 특징(features that specify characteristics of the service)에 관한 정보를 포함하고 있다. 서비스 제공자에 대한 정보는 서비스를 제공하는 조직의 연락처 및 관련된 정보를 포함한다. 기능에 대한 정보는 서비스에 의해 처리되는 변환 내용을 담고 있다. 구체적으로 설명하면, 서비스의 실행을 위해 요구되는 입력(input)과 생성된 출력(output), 그리고 서비스 실행 전에 요구되는 외부적 조건 및 서비스를 통한 효과에 대한 정보를 기술하고 있으며 프로파일에서는 이러한 정보를 전제조건(preconditions)과 효과(effects)로 명시하고 있다(이를 IOPE라고 한다). 마지막으로 특징에 관한 정보는 제공된 서비스의 카테고리(category), 품질등급(quality rating), 서비스 매개변수(parameter)를 포함하고 있다. <표 1>는 서비스 프로파일 온톨로지 내용을 기술한 것이다.

2.3 반자동화된 웹 서비스 조합 시스템

본 연구와 가장 유사한 시스템으로는 조지아 대학 LSDIS 연구실에서 진행되고 있는 METEOR-S 프로젝트[6, 7]에서 찾아볼 수 있다. METEOR-S는 본 연구와 비슷하게 시맨틱 템플릿을 사용하여 이를 워크플로우 실행계획으로 변환한 후 실행엔진에서 웹 서비스 조합을 실행하는 일련의 과정을 기술하고 있다. 그러나 이 시스템은 실제적인 적용에 중점을 둔 것보다는 이론에 더 많이 치중된 연구로서 각 프로세스 수행 과정이 본 연구와 전혀 다르다. 웹 서비스 매칭 알고리즘은 [11]에서 제안한 알고리즘을 확장한 것으로서 본 논문에서와 같은 OWL-S 온톨로지 처리 메카니즘은 고려하지 못하고 있

〈표 1〉 서비스 프로파일 온톨로지 내용

구성	요소	설명	
organization	serviceName	서비스 이름, 서비스 식별자로 사용	
	textDescription	서비스 내용을 요약한 텍스트	
	contactInformation	서비스 관련 연락처	
function	hasInput	입력항목	
	hasOutput	출력항목	
	hasPrecondition	서비스 사용 전에 만족해야할 조건	
	hasEffect	서비스 실행 후 변화	
features	serviceCategory	categoryName	분류명칭
		taxonomy	분류체계
		value	분류체계에서의 값
		code	분류체계 코드
	qualityRating	품질등급	
	serviceParameter	serviceParameterName	파라메트 이름
		sParameter	파라메트 값

으며, QoS 처리의 ILP(Integer Linear Programming) 모듈은 너무 이론적이어서 실제 비즈니스 프로세스에 적용하기에는 비현실적이다. 또한 본 연구에서는 METEOR-S와는 달리 엔진 프록시를 자체 개발하여 독자적인 워크플로우 실행환경을 구축하고 있다.

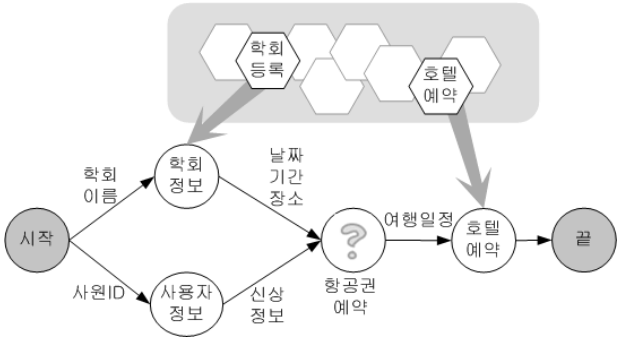
매릴랜드 대학 Mindswap 연구실에서 개발한 WS-Composer[8]는 OWL-S 기반 반자동 웹 서비스 조합 시스템으로서, 조합을 수행하는 동안에 웹 서비스 필터링 및 선택을 지원하는 matchmaking 알고리즘을 사용하여 목표 지향적이고 대화식 조합 기법을 제안하고 있다. 그러나 이 시스템에서는 BPM 기법은 고려하지 않았기 때문에 비즈니스 환경에서 요구되는 전체 기능을 제공하지 못하고, 워크플로우 실행엔진이 미비한 아직 성숙된 기술이 아닌 단점이 있다. 스탠포드 대학 KSL 연구실에서는 동적 웹 서비스 발견을 위해 WS-BPEL를 중심으로 OWL-S 개념을 도입한 보텀-업(bottom-up) 방식을 제안하였다[9]. 이 연구실에서는 WS-BPEL 구현 엔진인 BPWS4J가 확장 가능하지 못하기 때문에 웹 서비스 탐색을 위한 중개자로써 SDS를 구현하였다. SDS는 비록 동적 웹 서비스 발견은 지원하고 있으나, 본 논문에서 수행하고자 하는 웹 서비스 조합 전반적인 내용에 대해서는 고려하고 있지 않다.

3. 시나리오

본 연구를 위해 먼저 현실적으로 실제 일어날 수 있는 하나의 가상 시나리오를 설명하고 여기에서 야기되는 새로운 이슈들을 살펴본다.

“학회 참석을 위한 여행 계획을 작성하라”와 같은 질의에 대해 먼저 설계자는 프로세스 디자이너(process designer)를 통해 (그림 1)과 같은 워크플로우(workflow)를 작성한다. 전통적인 워크플로우 구조에는 순차, 병행, 선택, 반복 등이 있다.

(그림 1)에서 회사 직원이 학회 참석을 원할 때 먼저 내부 결재 과정을 거친다. 그림에서 이 부분은 ‘시작’으로 표시되어 있으며, 워크플로우의 복잡성을 피하기 위해 이 부분은 자세히 표시하지 않았다. 일단 승인이 완료되면 학회정보와 사용자정보를 얻고 항공권 및 호텔예약이 이루어진다. 학회정보 태스크는 학회이름으로 날짜, 기간, 개최장소 등에 대한 정보를 얻는다. 이 정보를 얻기 위해 학회등록 웹 서비스가 선택되고 워크플로우에 링크된다. 사용자정보 태스크는 사내 웹 서비스를 사용하여 사원 ID로 사원 신상정보를 검색한다. 항공권예약 태스크는 학회날짜, 기간, 장소, 그리고 신상정보를 이용하여 항공권예약을 수행한다. 마지막으로 호텔예약 태스크는 호텔예약 웹 서비스를 사용하여 여행일정에 따라 필요한 호텔예약이 이루어진다. 일단 항공권과 호텔예약이 끝나면 확인과 조정 작업이 이루어지는데, 이 부분은 ‘끝’으로 표시되어 있다.



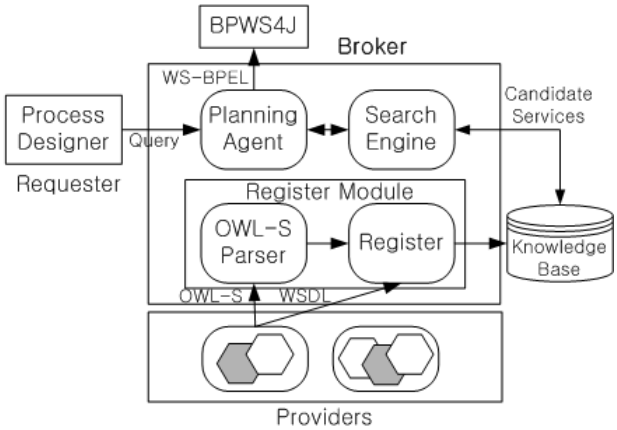
(그림 1) 여행계획 워크플로우

설계자는 (그림 1)과 같은 워크플로우를 거의 완성하였으나, 다만 항공권예약 태스크는 적절한 웹 서비스를 찾지 못하였다. 이를 완성하기 위해 설계자는 적합한 항공권예약 웹 서비스를 찾아서, 기존 워크플로우에 이 웹 서비스를 결합(binding)시켜야만 한다. 여기에서 설계자에게 두 가지 큰 문제가 발생하게 된다. 첫째, 탐색 문제: 기존 시스템과는 달리 인터넷 상에는 수없이 많은 웹 서비스들이 존재하므로 이들 중에서 가장 적당한 것을 찾는 일은 너무 시간이 많이 소비되고 지루한 일이 된다. 이러한 작업을 수동으로 수행하는 것은 거의 불가능하게 보이며 이를 효율적으로 지원할 수 있는 웹 서비스 탐색 메카니즘이 제공되어야 한다. 둘째, 통합 문제: 일단 웹 서비스가 발견되었더라도 완벽하게 워크플로우에 일치되고 상호 운용 가능하리라고는 보장할 수 없다. (그림 1)에서 설계자는 발견된 항공권예약 웹 서비스의 입력부분을 학회등록과 사용자정보 웹 서비스의 출력부분과 연결시켜야 하고, 출력부분을 호텔예약 웹 서비스의 입력부분과 연결되어야 한다. 하지만 발견된 웹 서비스가 입출력 인터페이스 부분에서 구조적으로 또는 의미적으로 상이할 수 있으므로 이 작업을 효율적으로 지원할 수 있는 통합 메카니즘이 필요하다.

4. 시스템 구조

WS-BPEL 기법과 OWL-S 기법을 융합한 SemanticBPEL 조합 시스템의 전체적인 구조는 (그림 2)와 같으며, 서비스 공급자(provider), 서비스 사용자(requester), 그리고 서비스 중개자(broker)로 구성되어 있다. 이는 SOA의 기본 구조를 따르고 있다. 서비스 중개자는 계획 에이전트(planning agent), 탐색 엔진(search engine), 그리고 등록 모듈(register module)로 이루어져 있으며, SOA의 ESB(Enterprise Service Bus) 역할을 수행한다.

SemanticBPEL 조합 시스템은 처음부터 완전히 새롭게 개발된 것이 아니라 본 연구실에서 이미 개발되어 있는 관련 시스템들을 기반으로 추가 기능들이 확장·발전되었다. 본 연구실에서는 이전에 ASP(Application Service Provider)를 위한 워크플로우 기반 서비스 중개자 시스템인 IMP(Internet MarketPlaces)[12]를 개발하였다. 이후, 웹 서비스 기술이 이기종 분산 컴퓨터 프로그램들을 통합할 수 있는 하나의 강력한 수단으로 많은 관심을 받게 됨에 따라 IMP를 동적 웹 서비스 조합을 위한 DWSC(Dynamic Web Service Composition)[13]



(그림 2) SemanticBPEL 시스템 구조

시스템으로 그 기능을 확장·발전시켰다. 그렇지만 DWSC 시스템에서는 비즈니스 워크플로우 전체 과정을 다룬 것이 아니라 웹 서비스 매칭 알고리즘만 소개되었다. DWSC의 다음 버전인 시맨틱과 워크플로우 혼합(hybrid) 시스템[14]에서는 DWSC 자체 워크플로우 실행엔진의 기능적 한계 때문에 BPWS4J 엔진으로 대치되었고, GUI 기반 워크플로우 디자이너가 새로 개발되었다. 한편, 본 연구의 SemanticBPEL 조합 시스템에서는 기존의 DWSC 기능과는 달리 비즈니스 프로세스 수명 주기 전체 과정의 반자동화를 다루게 되며, 동적 웹 서비스 발견 및 통합 문제를 해결하기 위해 다단계 웹 서비스 탐색 방법이 제안된다.

SemanticBPEL 조합 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 등록기(register)에 전달되고, WSDL은 바로 웹 서비스 등록기에 전달된 후 지식 베이스(knowledge base)에 저장된다. 서비스 사용자는 프로세스 디자이너를 사용하여 워크플로우를 완성하고 계획 에이전트로부터 이를 WS-BPEL 스팩으로 변환한 후 BPWS4J 엔진에 워크플로우를 배치(deploy)한 후 웹 서비스 조합을 실행한다.

사용자 처리과정을 좀 더 자세히 살펴보면, 사용자는 먼저 프로세스 디자이너를 사용하여 비즈니스 프로세스를 작성하는데, 탐색이 요구되는 태스크는 지식베이스로부터 적합한 후보 서비스들이 발견되도록 먼저 질의 템플릿을 작성한다. 그리고 난 후 탐색 엔진은 질의 템플릿을 기반으로 웹 서비스 발견 과정을 수행한다. 특히 본 논문에서는 다단계 웹 서비스 탐색 방법을 통해 유사 값에 의한 우선순위별 후보 서비스들을 리턴한다. 이들 중 사용자가 가장 적합한 웹 서비스를 선택하면 프로세스 디자이너는 프로세스 전후 관계에 따라 새로운 웹 서비스를 기존 워크플로우에 결합한다. 최종적으로 이 워크플로우는 계획 에이전트에 의해 WS-BPEL로 변환되고 이 실행 계획은 BPWS4J 엔진에 의해 프로세스가 실행된다.

5. 웹 서비스 발견 및 통합 기법

5.1 질의 템플릿

새로운 웹 서비스를 탐색할 때 사용자는 먼저 질의 템플릿(query template) 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.

질의 템플릿 Q는 워크플로우가 최종적인 목표에 가깝게 도달될 수 있도록 그 단계에서 요구되는 웹 서비스의 특성을 표현한 하나의 청사진이다. Q는 서비스 메타데이타(SM: Service Metadata), 기능 시맨틱(FS: Function Semantic), 그리고 QoS(Quality of Service) 3개의 튜플로 표현된다. 즉,

$$Q = \langle SM, FS, QoS \rangle$$

여기서, *SM*은 다음과 같이 4개의 튜플로 구성되며,

$$SM = \langle ServiceName, Description, ServiceCategory, Location \rangle$$

*ServiceCategory*는 NAICS와 UNSPSC와 같은 분류시스템을, *Location*은 ISO 3166 Geographic 코드시스템을 사용한다. 특히, OWL-S의 serviceParameter 요소는 다양하게 이용될 수 있는데 본 논문에서는 이를 *Location*으로 구체화시켰다. *FS*는 다음과 같이 4개의 튜플로 구성된다.

$$FS = \langle Is, Os, Ps, Es \rangle$$

여기서, *Is*는 입력항목, *Os*는 출력항목, *Ps*는 전제조건, *Es*는 효과를 표시한다.[1)]

QoS는 아직 표준화 되어 있지 않으나 본 논문에서는 관련연구[6, 7, 15]에서 정의한 웹 서비스 품질모델을 참조하여 다음과 같이 4개의 튜플로 구성한다. 이것은 OWL-S의 qualityRating 요소를 확장한 개념이다[16].

$$QoS = \langle Time, Cost, Availability, Reliability \rangle$$

QoS의 각 요소는 다음과 같이 4개의 튜플로 구성되며,

$$QoS_i = \langle pName, Op, Val, Unit \rangle$$

여기서, *pName*은 파라메터 이름, *Op*는 비교 연산자, *Val*는 값, *Unit*는 단위이다.

(그림 1)에서 찾고자 하는 항공권예약 웹 서비스에 대한 질의 템플릿 Q의 예를 들면 <표 2>와 같이 표현된다.

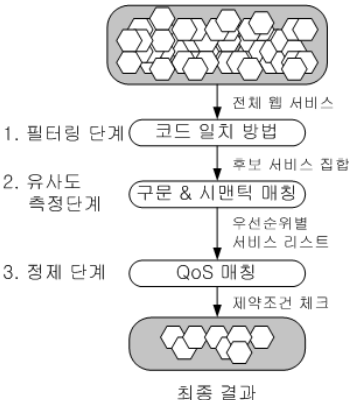
5.2 다단계 웹 서비스 탐색 방법

본 연구에서 웹 서비스 발견 및 통합 과정은 다단계 매칭 방법에 의해 수행된다. 각 매칭 단계에서 질의 템플릿 Q와

1) OWL-S 1.1 버전에서는 *Ps*와 *Es*를 어떻게 표현할지 아직 공식적인 스펙이 정해지지 않았다.

〈표 2〉 질의 템플릿(Q)

항목	설명	값
ServiceName	서비스 이름	항공권 예약
ServiceCategory	NAICS 분류코드	721215(항공예약)
Location	지리적 범위	KOR(한국)
Is	입력항목	날짜∪시간∪장소∪신상정보
Os	출력항목	여행일정
Time	반응시간	< 50 seconds
Cost	비용	< 50000 dollars



(그림 3) 다단계 웹 서비스 탐색 방법

지식베이스 내의 웹 서비스 S 사이의 일치 여부가 판단되는데, 이를 위해 먼저 사용자로부터 작성된 질의 템플릿에 대해 어떠한 순서로 수행할지에 대한 질의 계획을 세운다. 질의 계획을 세울 때는 질의 처리의 효율을 고려해야 하는데, 매칭 요소의 특성상 매칭 속도가 빠른 항목이 있고 매칭 속도가 느린 항목이 있으며, 코드별 1차 필터링 단계와 제약조건에 따른 2차 정제 단계가 있기 때문이다. 따라서 계산량이 많은 시맨틱 매칭과 정제 단계인 QoS 항목은 다른 항목들 보다 늦게 처리하고, 카테고리와 문자열 처리의 경우 적은 계산량으로 높은 질의 처리 효율을 가지고 있으므로 다른 항목들 보다 먼저 처리한다.

본 연구에서는 (그림 3)과 같이 1단계로 전체 웹 서비스들 중에서 코드 일치 방법에 의해 후보 서비스 집합을 필터링한 후, 2단계로 구문 및 시맨틱 매칭 방법에 따라 우선순위별 서비스 리스트를 구한다. 그리고 마지막 3단계에서 QoS 매칭 방법에 의해 제약조건에 위배되는 서비스들을 정제시켜 최종 결과를 산출하는 다단계 웹 서비스 탐색 방법을 제안한다. 각 매칭 방법에 대한 자세한 내용은 아래에 같다.

5.2.1 코드 일치 방법

질의 템플릿 Q에 카테고리(*ServiceCategory*)와 위치(*Location*) 정보가 있으면 이는 분류코드로 이루어져 있으므로 코드 일치(exact) 방법을 사용하여 매칭을 수행한다. 코드 일치 방법은 분류 코드에 속하는 서비스들만 빠르고 쉽게 필터링하여 다음 매칭 단계가 효율적으로 수행되도록 탐색 범위를 한정시킨다.

5.2.2 구문(syntactic) 매칭

서비스 이름(*ServiceName*)과 설명(*Description*)은 문자열로 구성되어 있다. 문자열에 대해 문자 일치 방법만 사용하면 너무 제한된 탐색이 이루어질 수 있기 때문에 좀더 유연한 매칭을 위해 정보검색 분야에서 널리 사용되고 있는 TF-IDF (Term Frequency-Inverse Document Frequency) 방법[17]을 사용하여 매칭의 유사도를 구한다.

TF-IDF 방법에서 각 단어의 가중치(term weight) w_{ik}는 해당 문서에서 각 단어의 빈도(TF)와 역문헌빈도(IDF)의 곱으로 나타낸다. 즉

$$w_{ik} = \frac{tf_{ik} log(N/n_k)}{\sqrt{\sum_{k=1}^{t}(tf_{ik})^2[log(N/n_k)]^2}}$$

여기서, tf_{ik}는 i번째 문서에서 단어 k의 빈도이며, N은 전체 문서의 수, n_k는 k가 출현한 문서의 수이다. 이를 이용한 질의와 문서 간의 유사도 $sim(Q, D_i)$ 측정은 cosine 방법으로 측정된다. 즉,

$$sim(Q, D_i) = \frac{\sum_{j=1}^{t} w_{qj} \times w_{ij}}{\sqrt{\sum_{j=1}^{t}(w_{qj})^2 \times \sum_{j=1}^{t}(w_{ij})^2}}$$

여기서, $Q = w_{q1}, w_{q2}, \ldots w_{qt}$이고 $D_i = w_{i1}, w_{i2}, \ldots. w_{it}$이다.

위 식을 이용하여 서비스 이름과 설명 유사도를 각각 계산할 수 있으며, 전체적으로 질의 템플릿 Q와 지식베이스에 있는 웹 서비스 S 간의 구문 유사도(syntactic similarity) *SynSim*은 다음과 같이 계산된다.

$$SynSim = \frac{w_1 NamSim + w_2 DesSim}{w_1 + w_2}$$

여기서, *NamSim*은 서비스 이름 유사도, *DesSim* 설명 유사도이며, w_1과 w_2는 각각의 가중치이다. 결과 값은 0과 1 사이의 실수값을 리턴한다.

5.2.3 시맨틱 매칭

단순한 키워드 기반 구문 매칭 방법은 의미(semantic) 있는 검색을 할 수가 없다. 전통적인 이 방식은 웹 서비스 발견 및 조합 과정에 반드시 사람의 개입을 필요로 함으로써 복잡한 비즈니스 환경에 적용하기 어렵다. 따라서 웹 서비스의 능력(capability)에 관한 의미 정보를 취급함으로써 이러한 제약을 완화하고 개방형 환경에서 동적 웹 서비스의 발견과 조합, 그리고 실행이 가능하게 할 수 있다.

기존의 시맨틱 매칭 방법들[11, 18]은 웹 서비스의 입·출력항목에 대한 의미 표현만을 기반에 두고 탐색을 수행하였다. 그러나 입·출력항목의 의미 특성이 동일하다고 하더라

```
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 4) 웹 서비스 탐색 알고리즘

도 실제 서비스 내용에는 차이가 있을 수 있으므로 서비스 입·출력의 의미 일치만으로는 요구되는 기능을 발견할 수 없다. 좀 더 효과적으로 사용자가 원하는 서비스를 발견하기 위해서는 웹 온톨로지에 기반을 둔 지능적인 서비스 매칭 방법이 필요하다.

본 논문에서 제안하는 시맨틱 매칭 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 4)와 같다. (그림 4)에서 Discovery( ) 함수는 질의 템플릿 Q를 지식베이스에 있는 모든 웹 서비스 S와 비교한다. 만일 매치가 발견되면 기록되고 우선순위에 의해 정렬된다. Matching( ) 함수는 먼저 템플릿 출력항목 Q.Os를 웹 서비스 출력항목 S.Os와 비교하여 유사도를 계산하고, 매치가 실패하지 않는다면 반대로 웹 서비스 입력항목 S.Is와 템플릿 입력항목 Q.Is를 비교한다.

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 연구에서는 웹 온톨로지 기반 유사성 측정기법을 적용하여 시맨틱 유사도를 구한다. OWL-S의 IOPE는 내장 데이타 타입이나 온톨로지에서 정의된 개념으로 기술될 수 있다. 따라서 입·출력항목 사이의 유사성은 두 가지 경우, 즉 내장 데이터 타입과 온톨로지 개념으로 구분하여 측정한다.

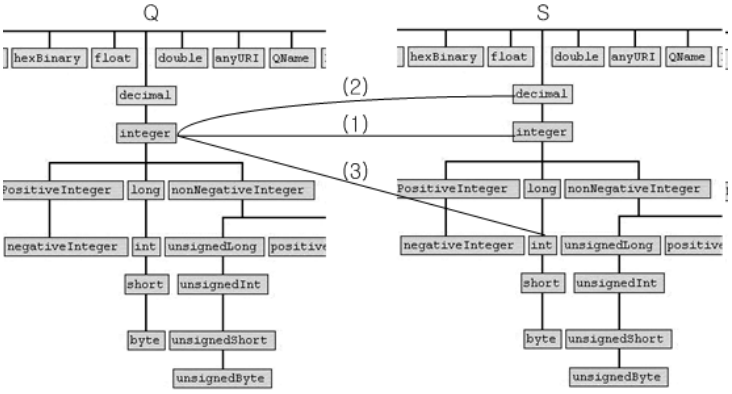
5.2.3.1 내장 데이터 타입으로 선언된 경우

질의 템플릿 입·출력항목이 내장 데이터 타입으로 선언된 경우에는 지식베이스에 있는 웹 서비스들 중에서 입·출력항목이 내장 데이터 타입으로 선언된 것들만 간단하게 비교하면 된다. 내장 데이터 타입과 관련된 시맨틱 유사도(semantic similarity) *SemSim* 함수는 (그림 5)의 XML 스키마 데이터 타입 계층도[19]를 기반으로 하고 있으며 0과 1 사이의 유사도 값을 산출한다. 그림에서 (1)은 Q와 S가 같은 경우, (2)는 Q가 S의 하위 개념인 경우, 그리고 (3)은 Q가 S의 상위 개념이거나 관계가 없는 경우를 나타낸다.

위의 3가지 경우를 고려한 시맨틱 유사도 *SemSim*의 결과 값은 다음과 같다.

$$SemSim = \begin{cases} 1 & \text{if} \quad case(1) \\ 1 & \text{if} \quad case(2) \\ \Omega & \text{if} \quad case(3) \end{cases}$$

Ω 값은 WfMS(Workflow Management System) 데이터 타입 변환 기능과 밀접한 연관이 있다. 일반적으로 WfMS에서는 태스크 사이에 데이터 전달을 할 때 데이터 타입 변



(그림 5) XML 스키마 데이터 타입 계층도

〈표 3〉 유사도(Ω) 측정치

Q		S	Ω
integer	⇨	long	2/3
integer	⇨	int	2/3
double	⇨	int	1/3
int	⇨	string	1

Q		S	Ω
long	⇨	int	2/3
string	⇨	float	0
date	⇨	time	0
date	⇨	gYear	1/3

〈표 4〉 온톨로지 개념 간 상호 포함관계

결과 값	조건	설명
Exact	Q = S	Q와 S가 같은 경우
PlugIn	S subsumes Q	S가 Q의 상위 개념인 경우
Subsumes	Q subsumes S	Q가 S의 상위 개념인 경우
Intersect	Q intersect S	Q와 S 사이에 일부분 공통된 속성이 있는 경우
Fail	Q ≠ S	Q와 S 사이에 전혀 관계가 없는 경우

환을 지원하는 능력을 가지고 있다. 예를 들면, 질의 템플릿이 integer로 선언되어 있으나 웹 서비스가 decimal인 경우, WfMS가 integer를 decimal로 데이터 타입 변환을 할 수 있다면 이의 유사도 *SemSim* 값은 1로 결정된다. 그러나 WfMS에서 데이터 타입 변환이 불가능한 경우에는 웹 서비스 실행이 불가능하므로 *SemSim* 값은 0으로 된다.

Ω 값은 <표 3>에서와 같이 유사도 측정치를 사전에 정의하여 검색항목이 내장 데이터 타입으로 선언된 경우 관련되는 Ω 값을 산출한다. 예를 들면, double에서 int의 Ω 값은 1/3로 정해져 있는데, 이런 경우에는 WfMS에서 데이터 타입 변환은 허락되지만 어떤 경우 데이터 손실이 발생될 수도 있기 때문에 별로 선호되지 않는 경우로써 Ω 값도 비교적 낮은 편으로 결정된다.

5.2.3.2 온톨로지 개념으로 선언된 경우

질의 템플릿 입·출력항목이 온톨로지 개념으로 선언된 경우에는 객체지향 모델과 같은 계층 관계에 따라 유사도가 결정된다. 온톨로지 개념 간 상호 포함관계는 <표 4>과 같이 5가지 경우가 발생될 수 있다.

위의 5가지 경우를 고려한 시맨틱 유사도 *SemSim*의 결과 값은 다음과 같다.

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

여기서, $|p(Q)|$는 Q 속성의 개수, $|p(S)|$는 S 속성의 개수, $\partial = |p(Q) \cap p(S)|$는 양쪽의 공통된 속성 개수이다. 결과 값은 0과 1 사이의 실수 값을 리턴한다.

지금까지는 편의상 설명을 간단히 하기 위해 입·출력항목이 한 개인 경우만을 다루었다. 만일 입·출력항목이 여러 개(n개) 존재하는 경우에는 전체의 평균값을 유사도로 설정한다. 즉 전체 유사도 *SemSim*은 다음과 같이 표현된다.

$$SemSim = \frac{\sum_{i=1}^{n} SemSim_i}{n}$$

한편, 웹 서비스의 통합 문제는 수많은 웹 서비스들에 대한 복잡한 입·출력 온톨로지 등으로 인해 기존의 정보시스템 통합과는 크게 다르다. 선택되어지는 웹 서비스의 입력부분은 이전 태스크의 출력부분과 연결되어야 하고, 출력부분은 다음 테스크의 입력부분과 연결되어야 하므로 입·출력 항목에 대한 유사도 값만 가지고 사용자가 수동으로 상호간 매핑 작업을 수행하는 것은 비현실적이다. 따라서 웹 서비스 통합을 위해 사람의 개입 없이 자동적으로 웹 서비스들의 우선순위를 계산해주는 메카니즘이 필요하다.

본 논문에서는 유사도 값을 기반으로 통합 정도(degree of integration)를 계산하여 우선순위별로 재정렬한 후 결과 값을 리턴해 주는 메카니즘을 제안한다. 통합 정도 *Integration*은 웹 서비스의 통합 정도를 리턴하는 이진함수로서 그 결과 값은 0과 1 사이이다. *Integration*을 구하기 위해서는 구문 및 시맨틱 매칭 방법에 따라 구해진 각각의 유사도를 결과 처리시 통합 계산될 필요가 있다. 즉 *Integration*은 다음 식과 같이 유사도 *SynSim*과 *SemSim*의 제곱근으로 계산된다.

$$Integration = \sqrt{SynSim \times SemSim}$$

5.2.4 QoS 매칭

구문 및 시맨틱 유사도 측정 방법을 통해 우선순위별 후보 서비스들을 발견하였다면, 이들 중에서 QoS 제약조건에 위배되는 서비스들을 배제하여 사용자가 만족할 수 있는 품질이 좋은 것만 정제할 필요가 있다. QoS 매칭 방법은 각 웹 서비스에 대한 품질요소를 매트릭스화 하고(이를 QoS 매트릭스라 한다), 사용자의 품질에 대한 요구사항을 Q에 매트릭스화한 후 두 매트릭스 간의 비교를 통해 결과를 정제한다.

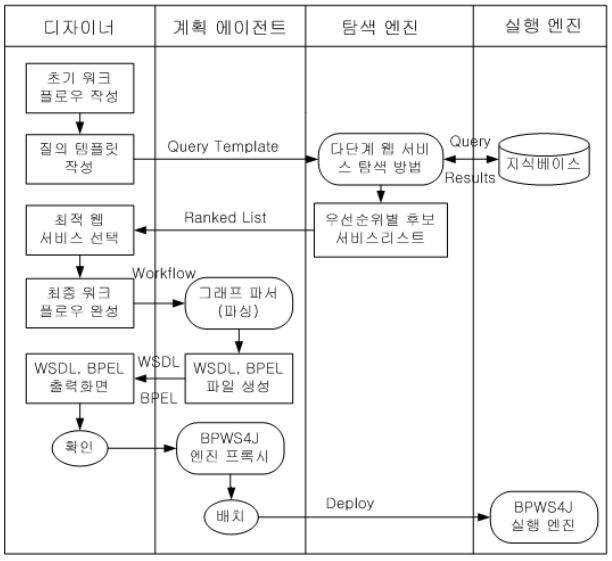
QoS 매트릭스는 웹 서비스들에 대한 정량적 또는 정성적 품질요소들을 정규화된 형태로 표현해야 하는데, 저장소에 있는 모든 서비스들에 대한 이러한 작업은 그리 용이하지는 않다. 웹 서비스의 특성상 시간이 흐름에 따라 품질요소의 수준이 계속 변할 수 있으며, 이를 지속적으로 품질 매트릭스에 반영한다는 것은 쉬운 일이 아니다. 따라서 QoS 품질요소를 쉽게 측정할 수 있는 공식이 먼저 제시되어야 한다. 현재 OWL-S에서는 QoS 요소로 qualityRating으로 간단히 표시되어 있으나 본 연구에서는 이를 확장하여 시간(Time), 비용(Cost), 가용성(Avalability), 그리고 신뢰성(Reliability)으로 구체화 시켰다.

5.2.4.1 시간

시간은 가장 보편적인 성능 측정치이다. 서비스 응답시간(response time)은 요청(request)을 보내고 응답(response)을 받는데 걸리는 시간을 의미한다. 이러한 응답시간은 실제 웹 서비스 호출 시 아래의 공식을 적용하여 시간을 산출한다.

$$Time = FT(t) - RT(t)$$

여기서, 서비스 완료시간(finish time) $FT(t)$는 시스템의 응답이 사용자에게 도착한 시간이며, 사용자 요청시간(request time) $RT(t)$는 사용자가 요청을 보낸 시간을 의미한다. 대개



(그림 6) 웹 서비스 조합 실행 과정

〈표 5〉 QoS 매트릭스

요소	설명	공식
Time	요청을 보내고 응답을 받는데 걸린 시간	FT(t) - RT(t)
Cost	웹 서비스를 실행하는데 드는 비용	RC + EC
Availability	웹 서비스가 이용 가능한지를 나타내는 확률	$1-\frac{DT}{\triangle t}$
Reliability	웹 서비스가 시스템의 실패 없이 얼마나 잘 수행했는지를 나타내는 확률	$1-\frac{MTBE}{\triangle t}$

의 경우 응답시간은 일정시간 동안의 평균값으로 구한다.

5.2.4.2 비용

비용은 웹 서비스의 크기, 종류, 그리고 서비스의 질 등에 따라 사용자에게 정해진 이용가격을 말한다. 이는 단순히 서비스 실행비용(realization cost) *RC* 뿐만 아니라 운영/관리 비용(enactment cost) *EC*를 포함한 가치를 말한다.

$$Cost = RC + EC$$

5.2.4.3 가용성

어떤 특정한 순간에 웹 서비스가 이용 가능한지를 나타내는 확률을 나타낸다. 시스템이 사용 가능하지 않는 시간을 가동불가시간(down time) *DT*라 하고, 시스템이 사용 가능한 시간을 가동가능시간(up time) *UT*라 할 때 이용 가능성이란 가동가능시간의 평균값을 의미한다. 이용 가능성을 구하기 위해서는 가동가능시간을 계속 모니터링하는 대신에 가동불가시간을 구해서 역으로 계산하는 방법을 사용한다. 가동불가시간은 시스템 이벤트를 분석해서 시스템 다운 시간을 얻어낸다.

$$Availability = 1-\frac{DT}{\triangle t}$$

여기서, 단위시간 $\triangle t = t_2 - t_1$이고, $DT = \sum_{t=i}^{j} dt(t)$이다. $(i, i+1, \ldots j) \in [t_1 \ldots t_2]$

5.2.4.4 신뢰성

주어진 기간 동안 웹 서비스가 실패(failure) 없이 얼마나 잘 수행했는지를 나타내는 확률이다. 신뢰성을 측정하기 위해서는 단위시간 동안 시스템 다운 뿐만 아니라 서비스 요청 실패에 대한 로그 분석을 통해 에러간 평균시간(mean time between error) *MTBE*를 구해 계산한다.

$$Reliability = 1-\frac{MTBE}{\triangle t}$$

여기서, $MTBE = \sum_{t=i}^{j} mtbe(t)$이다.

이상 설명한 QoS 매트릭스를 <표 5>에서 요약 정리하였다.

6. 워크플로우 실행

워크플로우 시작에서부터 점차적으로 하나씩 새로운 서비스가 첨가되어 웹 서비스 프로세스가 완성되고 나면 워크플로우 실행계획, 즉 WS-BPEL 스펙으로 자동 변환되어 실행엔진에 배치되고 워크플로우가 실행된다. OWL-S는 Sequence, If-Then-Else, Repeat-Until과 같은 프로세스 제어 구성자는 지원하고 있으나 실시간 환경에서 비지니스 프로세스의 조합 실행 기능은 제공하지 못하고 있다[20]. 이러한 필요성에 따라 최근엔 WS-BPEL, WSCI, BPSS 등 여러 가지 웹 서비스 조합언어가 발표되었으나 이들 언어들은 시맨틱 개념을 지원해 주지 못하고, 수동으로 작업을 해야 하는 단점이 있다. 따라서 본 연구에서는 OWL-S 기반 동적 웹 서비스 조합 기법을 제안하였고, 완성된 워크플로우는 WS-BPEL 스타일의 실행계획으로 변환되어 BPEL 구현 엔진인 BPWS4J에서 웹 서비스 조합이 실행되는 모듈을 구현하였다.

SemanticBPEL 조합 시스템은 클라이언트-서버 프로그래밍 구조로 되어 있다. 클라이언트 측 프로세스 디자이너는 "드래그 & 드롭" 인터페이스 방식으로 워크플로우를 작성한다[14]. 사용자로부터 질의 템플릿이 주어지면 다단계 웹 서비스 탐색 방법을 적용하여 우선순위별 후보 서비스 리스트를 보여주고, 이들 중 가장 적합한 웹 서비스를 선택한다. 이러한 방식으로 점차적으로 하나씩 새로운 웹 서비스가 첨가되고 나면 최종 워크플로우가 완성된다.

일단 워크플로우가 완성되고 나면 이를 서버 쪽에 전송하여 WS-BPEL 스팩으로 변환하고 BPWS4J 엔진에 배치시켜야 한다. 이를 위해 프로세스 디자이너에서는 완성된 워크플로우를 그래픽 형태로 서버에 전달한다. 서버는 이 그래프를 받아서 그래프 파서(Graph Parser)를 호출한다. 그래프 파서는 워크플로우를 파싱하여 관련된 WSDL과 WS-BPEL 파일을 만들어 클라이언트 측에 출력한다. 여기서 사용자가 '확인' 버튼을 클릭하면 서버 쪽의 BPEL 배치함수를

부른다.

서버 쪽의 BPEL 배치함수는 IBM BPWS4J 엔진을 대신해서 기동된다. 이는 현재의 BPWS4J 엔진이 단지 웹 기반 관리자 툴을 통해서만 배치를 허용하기 때문에 엔진 그 자체로는 자동적인 BPEL 배치가 불가능하다. 따라서 프로그래밍 인터페이스가 지원되는 엔진 프록시를 만들어야 한다. 처리방법은 IBM 웹 서비스 Gateway 컴포넌트를 초기화한 후 SOAP RPC(Remote Procedure Call) 채널을 통해 WS-BPEL 문서를 BPWS4J 엔진에 배치한다.

지금까지 설명한 워크플로우 실행 내용은 (그림 6)에서 보여주고 있다. 각 구획에서 SemanticBPEL 구성원들을 구분하고 구성원들 간의 상호 작용을 통한 웹 서비스 조합 실행 과정을 액티비티 다이어그램을 사용하여 기술하였다.

7. 실험 및 분석

실험의 목적은 본 논문에서 제안하는 다단계 웹 서비스 탐색 방법의 우수성을 보이는 것이다. 전통적인 웹 서비스 탐색 방법은 UDDI를 이용한 키워드 기반 검색만 지원하고 있다. 이러한 키워드 기반 검색 방법에 비해 다단계 웹 서비스 탐색 방법이 얼마나 효율적으로 수행되는지 두 방법을 비교·분석하였다.

실험은 MS Window Server 2003 버전에 Apache Tomcat 5.5와 Axis 1.3이 설치된 3.20GHz Intel CPU에서 수행되었다. WS-BPEL로 작성된 워크플로우 실행은 IBM BPWS4J 엔진에서 수행되었고, 프로세스 디자이너 구현을 위한 그래픽 툴은 opengraph graphing 팩키지를 사용하였다. 그리고 OWL-S 파서와 등록기는 매릴랜드 대학의 OWL-S API 모듈[8]을 이용하였으며, 이런 툴들을 기반으로 한 시스템 구현 작업은 IBM Eclipse 개발 환경에서 수행되었다.

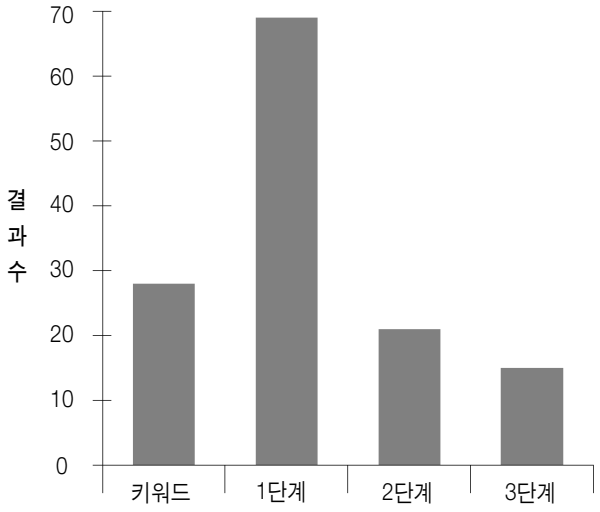
시맨틱 웹 서비스들은 현재 연구가 활발히 진행되고 있는 분야이므로 아직까지 적합한 실세계 테스트 데이터들이 이용 가능하지 않다. 따라서 다단계 웹 서비스 탐색 방법의 성능을 분석하기 위해 xmethods.net에서 제공하고 있는 기존의 웹 서비스들을 이용하였다. 이 사이트로부터 508개의 WSDL 파일을 다운로드 받아 OWL-S 온톨로지와 NAICS 분류체계에 맞게 수정하여 지식베이스에 저장하였다.

평가 방법은 정보검색에서 가장 보편적으로 사용되고 있는 재현율(recall)과 정확률(precision)을 사용한다. 본 실험의 경우는 서비스 집합과 질의 모두 웹 서비스이므로 재현율과 정확률은 다음과 같이 정의된다.

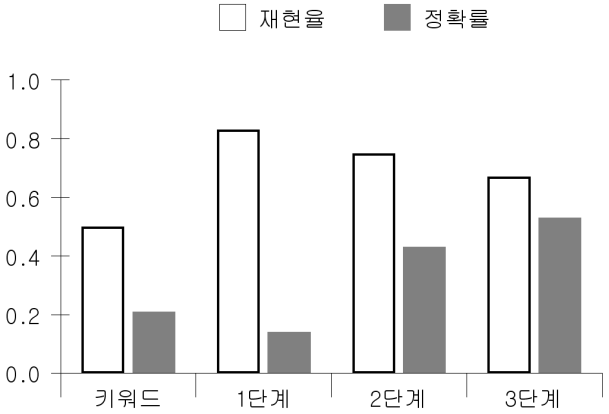
$$\text{재현율} = \frac{\text{검색된 서비스들 중 같은 부류에 속하는 웹 서비스의 수}}{\text{같은 부류에 속하는 웹 서비스의 수}}$$

$$\text{정확률} = \frac{\text{검색된 서비스들 중 같은 부류에 속하는 웹 서비스의 수}}{\text{검색된 웹 서비스의 수}}$$

실험은 기존의 키워드 기반 검색 방법과 제안된 다단계 탐색 방법 중 1단계(필터링), 2단계(유사도 측정), 그리고 3단계(정제)에 대해 각각 검색된 웹 서비스의 수와 재현율·정확률을 측정하였다. 이에 대한 결과는 (그림 7), (그림 8)과 같다.



(그림 7) 검색된 웹 서비스의 수



(그림 8) 재현율과 정확률

재현율과 정확률은 모두 높을수록 성능이 좋다. 하지만 이들은 서로 반비례의 관계가 있어 한쪽을 높이면 다른 한쪽이 내려가는 것이 보통이다. 1단계는 탐색 효율을 향상시키기 위해 NAICS 분류체계에 따라 전체 웹 서비스들 중에서 후보 웹 서비스 집합을 단순히 필터링시키는 것이다. 실험 결과 1단계에서 검색된 웹 서비스의 수는 키워드 검색보다 훨씬 많다. 이는 xmethods.net에 등록된 웹 서비스들이 아직까지는 다양한 산업분야에 걸쳐서 개발된 것이 아니라 주로 컴퓨터 분야에 치우쳐 있어서 이러한 내용을 위주로 검색을 수행했기 때문이다. 향후 웹 서비스 개발이 다양한 분야로 확산되면 필터링은 큰 효과를 볼 수 있을 것이다. 1단계에서는 최종 결과 가능성이 있는 후보 서비스들을 임시 저장소에 여과시키는 과정이므로 재현율은 상당히 높은 반면 정확률은 상당히 낮은 경향을 보이고 있다.

2단계는 1단계의 결과를 가지고 구문 및 시맨틱 매칭을

수행하여 유사도가 0인 서비스들을 제외시킨 우선순위별 리스트를 구하는 것이다. 유사도가 0인 것은 질의와 전혀 관계없는 서비스들이므로 이들을 제거하면 정확률을 향상시킬 수 있다. 실험 결과 2단계에서 검색된 웹 서비스의 수는 상당히 많이 줄어들었으며, 재현율과 정확률도 키워드 검색에 비해 각각 25%, 22% 개선된 것을 알 수 있다.

한편, 3단계는 QoS 제약조건에 위배되는 서비스들을 제거하여 품질이 좋은 서비스만 정제하는 단계이므로 재현율 및 정확률 향상과는 큰 관련이 없다. 측정된 QoS 매트릭스 값을 기반으로 질의 제약조건을 적용한 결과 검색된 웹 서비스의 수는 예측된 바와 같이 2단계보다 약간 줄어들었다. 이로 인해 재현율과 정확률을 다시 계산한 결과 재현율은 조금 하락하고, 정확률은 조금 상승되었다. 이러한 실험 분석 결과는 다단계 웹 서비스 탐색 방법이 기존의 키워드 기반 검색 방법보다 성능이 우수함을 보여주고 있다. 검색 시간도 수없이 많은 웹 서비스들이 존재할 경우 빠른 필터링 기법에 따라 상당한 성능 향상이 기대된다.

8. 결 론

본 논문에서는 반자동 웹 서비스 조합 시스템을 구현하기 위해 WS-BPEL 기법과 OWL-S 기법을 융합한 SemanticBPEL 조합 기법을 제안하였다. 본 연구에서는 기존의 연구들과는 달리 워크플로우 및 질의 템플릿 작성, 동적 웹 서비스 발견 및 통합, 그리고 워크플로우 실행 등 비즈니스 프로세스 수명주기 전체 과정을 다루고 있다. 특히, 웹 서비스 발견 및 통합을 위해 다단계 웹 서비스 탐색 방법을 새로 제안하고 있다. 본 기법은 실험 분석을 통해 기존의 키워드 기반 검색 방법보다 성능이 우수함을 보였다.

SemanticBPEL 조합 기법에 대한 보다 객관적인 성능 우수성을 증명하기 위해서는 WS-BPEL 기법과 OWL-S 기법과의 상호 비교·분석이 요구된다. 하지만, 이들 분야들도 현재 연구가 초기 단계이므로 테스트를 위한 적합한 데이터가 아직까지는 존재하지 않는다. 향후 이에 대한 충분한 실험 및 분석이 요구된다. 또한, 본 시스템은 상용화되어 있는 타 시스템들에 비해 완전성이 부족하여 이미 나와 있는 WS-BPEL 2.0 표준 스펙을 모두 만족시키기 위해서는 많은 테스트 및 추가 모듈의 개발이 필요하다.

참 고 문 헌

[1] Papazoglou M., Traverso P., Dustdar S., and Leymann F., Service-Oriented Computing Research Roadmap, Technical Report, Tilburg University, Netherlands, http://infolab.uvt/pub/papazogloump-2006-96.pdf, 2006.

[2] OASIS, The Web Service Business Process Execution Language Version 2.0 Working draft, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel, 2007.

[3] OWL Services Coalition, OWL-S: Semantic Markup for Web Services, OWL-S White Paper, http://www.daml.org/services/owl-s/1.2/overview/, 2006.

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

[5] Active Endpoints, http://www.active-endpoints.com/

[6] Sivashanmugam K., Miller J., Sheth A., and Verma K., Framework for Semantic Web Process Composition, International Journal of Electronic Commerce, Vol.9(2), pp. 71-106, 2005.

[7] Verma K., Gomadam K., Sheth A., Miller J., and Wu Z., The METEOR-S Approach for Configuring and Executing Dynamic Web Processes, Technical Report, LSDIS Lab, University of Georgia, 2005.

[8] Sirin E., Parsia B., and Hendler J., Composition-driven Filtering and Selection of Semantic Web Services, AAAI spring symposium on semantic web services, 2004.

[9] Mandell D. and Mcllraith S., Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation, Proceeding of the 2nd International Semantic Web Conference(ISWC2003), Sanibel Island, Florida, 2003.

[10] 김훈태, BPM 도입에 의한 e-비즈니스 환경 고도화에 대한 연구, 한국정보사회진흥원, 2006.

[11] Paolucci M., Kawamura T., Payne T. R., and Sycara K., Semantic Matching of Web Services Capabilities, Proceedings of the 1st International Semantic Web Conference(ISWC), 2002.

[12] 이용주, 인터넷 서비스 임대를 위한 워크플로우 기반 서비스 중개자 구현기법, 정보처리학회논문지D, 제9-D권 제2호, pp.277-288, 2002.

[13] 이용주, 동적 웹 서비스 조합을 위한 시맨틱 웹 서비스 발견 및 실행 기법, 정보처리학회논문지D, 제12-D권 제6호, pp. 889-898, 2005.

[14] 이용주, 시멘틱과 워크플로우 혼합 기법에 의한 자동화된 웹 서비스 조합 시스템, 정보처리학회논문지D, 제14-D권 제2호, pp.265-272, 2007.

[15] Cardoso J. and Sheth A., Semantic e-Workflow Composition, Journal of Intelligent Information Systems(JIIS), 2003.

[16] Aversano L., Canfora G., and Ciampi A., An algorithm for Web service discovery through their composition, Proceedings of the IEEE International Conference on Web Services(ICWS), 2004.

[17] Salton G. and C. Buckley, Term weighting approaches in automatic text retrieval, Information Processing and Management, 24(5), pp.513-523, 1988.

[18] Pistore M., Marconi A., Bertoli P., and Traverso P., Automated Composition of Web Services by Planning at the Knowledge Level, International Joint Conference on Artificial Intelligence(IJCAI), 2005.

[19] XML Schema Part2, hppt://www.w3.org/TR /xmlschema-2/

[20] Sirin E., Parsia B., and Hendler J., Template-based Composition of Semantic Web Services, American Association for Artificial Intelligence, 2005.

이 용 주

e-mail : yongju@knu.ac.kr

1985년 한국과학기술원 산업공학과 정보검색전공(석사)

1997년 한국과학기술원 정보및통신공학과 컴퓨터공학전공(박사)

1985년~1989년 시스템공학연구소 연구원

1987년~1988년 일본 IBM TRL 연구소 연구원

1989년~1994년 삼보컴퓨터 근무

1998년~2007년 상주대학교 컴퓨터공학과 부교수

2008년~현 재 경북대학교 이공대학 컴퓨터공학과 부교수

관심분야 : 웹 데이터베이스, 정보검색, 공간 데이터베이스