韓國 컴퓨터情報學會 論文誌
第10卷 第1號, 2005. 3.

2005-3-1-1-13

시멘틱 e-워크플로우 프로세스를 이용한 동적 웹 서비스 조합

이용주*

Dynamic Composition of Web Services using Semantic e-Workflow Processes

Yong-ju Lee*

요 약

최근에 동적으로, 즉 요구가 있는 즉시 서비스들을 조합하는 동적 웹 서비스 조합은 가장 큰 관심사 중 하나이다. 특히, 사용자가 현존하는 하나의 웹 서비스로는 요구사항들을 만족시켜줄 수 있는 기능이 없을 때 그러한 요구를 만족시키기 위해 몇 개의 서비스들을 동적으로 결합해야 하는 문제가 큰 이슈로 부각되고 있다. 본 논문에서는 비즈니스 분야에서 성공적으로 활용되고 있는 워크플로우 기법을 적용하여 동적 웹 서비스 조합을 구현한다. 워크플로우의 웹 서비스 적용은 아직까지는 새로운 분야로써 다음과 같은 두 가지 문제 해결을 요구한다. (1) 인터넷상에 흩어져 있는 수많은 웹 서비스들 중에서 어떻게 원하는 서비스를 효율적으로 탐색할 수 있는가, (2) 탐색된 다양한 이기종 웹 서비스들 간의 상호운용성 극대화 문제에 관한 해결이 필요하다. 본 연구에서는 (1)을 수행하기 위해 웹 서비스 매칭 알고리즘이 제안되고, (2)를 해결하기 위해 매칭 알고리즘에 온톨로지 개념이 적용된다. 그리고 어떻게 동적 웹 서비스 조합이 구현되는지 설명하기 위해 하나의 프로토타입 시스템을 구축한다.

Abstract

Recently, one of the most challenging problems is to compose web services dynamically, that is, on demand. In particular, when a functionality that cannot be directly realized by existing services is required, existing services may be combined together to fulfill the request. This paper is to implement the dynamic composition of web services using existing workflow technologies. Workflow systems play a major role in E-businesses. However, due to the composition of web services using workflows differs from the design of traditional workflow systems, two problems need to be solved: (1) how to efficiently discover web services and (2) how to facilitate the interoperability of heterogeneous web services. In this paper, we present a matching algorithm for web service discovery and propose an approach based on the use of ontologies to facilitate the integration of web services. Finally, we have built a prototype system to illustrate how to the dynamic composition of web services are achieved.

▸ Keyword : 웹 서비스 조합(web services composition), 워크플로우 프로세스(workflow process), 매칭 알고리즘(matching algorithm), OWL-S, 온톨로지(ontologies)

• 제1저자 : 이용주
• 접수일 : 2005.01.27, 심사완료일 : 2005.02.28
* 상주대학교 컴퓨터공학과 교수

Ⅰ. 서 론

웹 서비스(web services)는 표준적인 웹 프로토콜을 통해 웹 환경 분산 컴퓨팅을 가능케 하며, CORBA나 DCOM과 같은 기존의 분산 컴퓨팅 모델을 대체할 수 있는 새로운 기술이므로 최근에 많은 관심을 받고 있다. 웹 서비스가 점차 현실화되고 보편화됨에 따라 향후 응용프로그램들은 웹 서비스들의 집합으로 구축될 수도 있다. 즉, 개발자가 굳이 모든 것을 개발하지 않더라도 다양한 웹 서비스들을 조합하여 새로운 형태의 서비스를 재창출해 낼 수가 있다.

최근에 웹 서비스 조합(composition)에 대한 필요성이 점차 증가되고 있다[1]. 이는 단순히 하나의 웹 서비스 그 자체만으로는 다양하고 복잡한 사용자의 요구사항을 충분히 만족시켜줄 수 없기 때문에 서비스 조합을 통해 새롭고 유용한 솔루션을 얻기 위함이다. 예를 들면, 항공 예약서비스를 이용하는 경우 예약 및 결재 업무가 별도의 웹 서비스로 구현되어 있다면 좌석 예약이라는 업무를 수행하기 위해서는 예약과 결재를 수행하는 웹 서비스 간에 적절한 연결이 이루어져야만 한다.

현재 웹 서비스 사용은 사용자들이 이미 알고 있는 몇 개의 서비스들을 이용하거나, 키워드 기반 검색엔진(예, 야후) 또는 웹 서비스 레지스트리(예, UDDI) 탐색에 의해 적합한 서비스들을 발견하고 있다. 또한 발견한 서비스들의 조합이나 이들 사이의 데이터 흐름은 주로 사용자 판단에 의한 수동으로 이루어지고 있다[2]. 이는 매우 불편한 작업이며 특히 복잡한 상황에서는 많은 시간이 소비되고 지루한 일이 될 수 있다. 복잡한 상황에서는 보통 수백개 또는 수천개의 웹 서비스가 포함될 수 있으므로(예, 유전자 정보) 웹 서비스 조합의 자동화는 매우 유용하고 작업시간을 상당히 줄여 줄 수 있을 것이다.

본 논문은 웹 서비스 조합 자동화 기법에 초점을 맞추고 있다. 주된 아이디어는 시멘틱 웹을 실현하기 위한 웹 온톨로지(Web ontology)와 비즈니스 분야에서 성공적으로 활용되고 있는 워크플로우(workflow) 기법들을 적용하여 웹 서비스 조합 자동화 시스템을 구현하는 것이다. 워크플로우를 이용한 웹 서비스 조합에는 다음과 같은 두 가지 기술적인 문제 해결을 요구한다. (1) 웹 서비스의 발견(discovery): 인터넷 상에 흩어져 있는 수많은 웹 서비스들 중에서 어떻게 원하는 서비스를 효율적으로 탐색할 수 있겠는가, (2) 웹 서비스의 통합(integration): 앞 단계에서 발견된 다양한 이기종 웹 서비스들 간의 상호운용성 극대화 문제에 관한 해결이 필요하다. 기존 관련 논문들에서는 (1)에 관한 연구는 주로 이루어졌으나, (1)(2)가 결합된 연구는 거의 수행되지 않았다. 따라서 본 연구에서는 (1)을 위해 웹 서비스 매칭 알고리즘이 제안되고, (1)(2) 결합 메카니즘을 위해 웹 서비스 워크플로우가 새롭게 제안된다.

본 논문의 구성은 다음과 같다. 2장에서는 웹 서비스 조합에 관한 관련연구를 설명하고, 3장에서는 하나의 가상 시나리오를 설정하고 여기에서 야기되는 새로운 이슈들을 살펴본다. 4장에서는 웹 서비스 온톨로지 언어인 OWL-S를 간단히 살펴보고, 5장에서 온톨로지 개념을 이용한 탐색 및 통합 알고리즘을 제안하고 웹 서비스 워크플로우 처리과정을 설명한다. 6장에서 시스템을 구현하고 7장에서 결론을 내렸다.

Ⅱ. 관련연구

관련연구에서는 먼저 웹 서비스 조합을 위해 기존에 제안된 표준언어들을 살펴보고, 이를 활용하여 웹 서비스 조합을 구현하려고 노력한 연구 결과들을 살펴본다.

2.1 웹 서비스 조합 언어

웹 서비스 기술을 구성하는 주요 구성요소로는 SOAP(Simple Object Access Protocol), WSDL(Web Services Description Language), UDDI(Universal Description Discovery and Integrity)가 있다. SOAP[3]은 웹에서 구조화되고 타입이 있는 정보를 교환하는 표준 XML 프로토콜이며, WSDL[4]은 웹 서비스 이용에 필요한 인터페이스와 입·출력 메시지 형식 등을 기술하고 있으며, UDDI[5]는 웹 서비스를 등록하고 검색·발견하기 위한 하나의 메커니즘을 제공하고 있다. UDDI는 키워드에 의한 사전 분류 시스템이기 때문에 웹 서비스 조합 기능은 제공하지 않는다. 이러한 기능을 제공하기 위해서는 기존의 표준 웹 서비스 기술에 새로운 기술 요소들을 추가적으로 개발할 필요성이 있다.

IBM의 WSFL(Web Service Flow Language)[6]과 마이크로소프트사의 XLang[7]는 웹 서비스 조합을 위해 가장 초기에 제안된 언어이다. 둘 다 WSDL을 확장한 것으로써 웹 서비스의 구문적(syntactic) 관점을 묘사하기 위해 사용된 표준언어이다. 특히, WSFL과 XLang은 비즈니스 워크플로우 정의 언어를 이용하여 특정 비즈니스 플로우를 정적으로 정의하고 이때 요구되어지는 웹 서비스의 조합을 수행하고 있다. BPEL4WS(Business Process Execution Language for Web Services)[8]는 보다 최근에 제안된 스펙으로써 WSFL과 XLang의 장점을 취합한 새로운 비즈니스 프로세스 정의 언어이다. BPEL4WS는 그래프 중심으로 프로세스를 표현하는 WSFL과 구조적 중심으로 프로세스를 표현하는 XLang을 웹 서비스 조합을 위해 통합한 새로운 표준언어이지만 시멘틱 개념은 지원하고 있지 않다.

시멘틱 웹(Semantic Web)[9]은 정보의 의미를 개념으로 정의하고 개념간의 관계성을 표현함으로써 정보를 공유시킨다. 따라서 웹 상의 정보를 수집하고 처리하기 위해 더 이상 인간의 전적인 개입이 요구되지 않는다. 각종 자동화된 에이전트를 통해 정보의 의미와 정보간의 관계성이 파악되고 이를 통해 정확한 정보 검색, 새로운 지식의 생성, 최적의 서비스 제공 등이 가능해 진다. 이러한 시멘틱 웹을 실현하기 위한 웹 온톨로지는 웹 상에 존재하는 자원들에 대한 지식 표현방법으로서 W3C 표준인 RDF(Resource Description Framework)를 기반으로 정의된 온톨로지 언어를 통해 기술된다. 잘 알려진 웹 온톨로지 언어로는 DAML, OIL, SHOE 등이 있으며 DAML과 OIL의 기능을 합쳐서 만들어진 DAML+OIL이 있다. 현재는 DAML+OIL을 발전시킨 OWL(Web Ontology Language)[10]에 대한 표준화가 진행되고 있다. OWL-S(OWL-services language)[11]는 OWL를 기반으로 지능적인 웹 서비스의 발견, 실행, 조합, 모니터링이 가능하도록 한 기술이다. 본 연구에서는 시멘틱 e-워크플로우 기법을 이용한 동적 웹 서비스 조합을 구현하기 위해 OWL과 OWL-S를 사용한다. (그림 1)은 웹 및 웹 서비스 표준언어 계통도를 나타내고 있다.

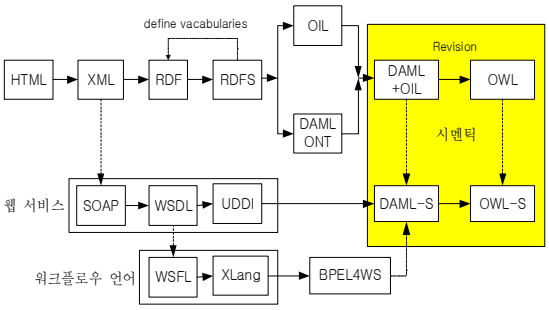

그림 1. 웹 및 웹 서비스 표준언어 계통도
Fig. 1. Diagram of the standard language for web and web services

2.2 웹 서비스 조합에 관한 연구

웹 서비스 조합은 크게 정적(static)과 동적(dynamic)으로 구분할 수 있는데, 정적은 서비스 조합이 설계 시에 결정되고, 동적은 실행 시에 결정된다. 현재 동적으로, 즉 요구가 있는 즉시 서비스들을 조합하는 동적 웹 서비스 조합이 큰 관심사인데, 특히, 사용자가 현존하는 하나의 웹 서비스로는 요구사항들을 만족시켜줄 수 있는 기능이 없을 때 그러한 요구를 만족시키기 위해 몇 개의 서비스들을 동적으로 결합해야 하는 문제가 큰 이슈로 부각되고 있다[12]. 동적 웹 서비스 조합에 관한 문제는 향후 수많은 웹 서비스가 매년 증가될 것으로 예측됨에 따라 동적 웹 서비스 조합 시스템은 웹 서비스들에 대한 활용도를 훨씬 증대시켜 줄 수 있을 것이다. eFlow[13]는 서비스 선택 룰(rules)을 지원하는 액티비티(activities)를 가지고 있다. eFlow 엔진이 액티비티를 실행할 때 서비스 선택 룰이 실행되고 우선 순위가 매겨진 서비스 리스트를 반환한다. eFlow 시스템에서 서비스 선택 룰은 XML 질의 언어인 XQL에 의해 표현된다. 이 시스템의 주된 단점으로는 시멘틱 개념을 지원하지 않고 단지 키워드 중심 탐색만이 가능하므로 그 활용도가 떨어진다.

시멘틱 기반 동적 웹 서비스 조합에 관한 연구는 WS-Composer[12]에서 수행되었다. 이 시스템은 OWL-S 시멘틱 서비스 언어를 사용하여 동적 웹 서비스 조합을 지원하는 시스템으로써, 조합을 수행하는 동안에 자체 알고리즘을 적용하여 목표 지향적이고 대화식 조합 기법을 제안하고 있다. 따라서 사람이 개입된 반자동 시스템으로 작동되며 워크플로우 기반 e-비즈니스 업무 프로세스는 고려되지 않았다.

한편, 정적 웹 서비스 조합에 관한 연구는 SWORD 프로젝트[14]에서 수행되어 졌다. SWORD에서는 룰 기반 전

문가 시스템이 자동적으로 프로세스 조합을 결정하는데 사용되고, 이러한 조합을 실현하기 위한 하나의 프로세스 계획을 반환한다. 그렇지만, 이러한 조합 구성에서는 다양한 방법들이 존재할 수 있으나 SWORD에서는 단지 하나의 임의적인 계획만 반환한다. 왜냐하면, 이 시스템에서는 계획들을 평가할 수 있는 비용 모델을 가지고 있지 않기 때문이다.

시멘틱 기반 정적 웹 서비스 조합 연구로써 eWorkFlow 시스템[15]이 있다. eWorkFlow에서는 3차원(즉, syntax, operational matrix, semantic)을 기반으로 한 웹 서비스 발견 알고리즘 및 방법론을 제안하였다. 제안된 알고리즘은 웹 서비스 인터페이스 간의 유사성을 발견하기 위해 특징(feature) 기반 모델을 적용하였다. 이 시스템에서는 기존의 워크플로우 시스템의 기능은 확장시켰으나 동적인 웹 서비스 조합 문제는 다루지 않고, 웹 서비스 워크플우 실행 메카니즘을 지원하지 못하고 있다. 지금까지 논의된 웹 서비스 조합에 관한 연구들을 <표 1>에 분류, 정리하였다.

표 1. 웹 서비스 조합에 관한 연구
Table 1. Study on the composition of web services

	키워드 기반 탐색		시멘틱 탐색	
	언어	시스템	언어	시스템
동적 웹 서비스 조합	WSFL, BPEL4WS	eFlow[13]	OWL-S	WS-Composer[12]
정적 웹 서비스 조합	WSFL, XLang, BPEL4WS	SWORD [14]	DAML-S	eWorkFlow [15]

III. 시나리오

본 연구를 위해 먼저 현실적으로 실제 일어날 수 있는 하나의 가상 시나리오를 설정하고 여기에서 야기되는 새로운 이슈들을 살펴본다. [예 1]은 전통적인 워크플로우 프로세스 예이고, [예 2]는 워크플로우 기반 동적 웹 서비스 조합의 예를 보여주고 있다.

워크플로우란 비즈니스의 자동화를 의미하며, 워크플로우 관리시스템(WfMS: Workflow Management System)은 소프트웨어를 이용하여 워크플로우의 수행을 정의, 생성, 관리하는 시스템을 의미한다[16]. 대부분의 WfMS는 비슷한 구조를 가지고 있는데 일반적으로 프로세스 디자이너, 워크플로우 엔진, 그리고 워크플로우 클라이언트로 구성되어 있다[17].

[예 1] "휴가를 위한 여행계획을 작성하라"와 같은 질의에 대해 WfMS에서는 먼저 설계자가 프로세스 디자이너를 통해 (그림 2)와 같은 워크플로우를 작성한다. 전통적인 워크플로우 프로세스 구조에는 순차, 병행, 선택, 반복 등이 있다. 먼저 항공기 예매 및 호텔예약을 병행 처리한 후 일정에 따른 관광을 예약하고, 일정이 확정되어지면 마지막으로 비용을 결재한다.

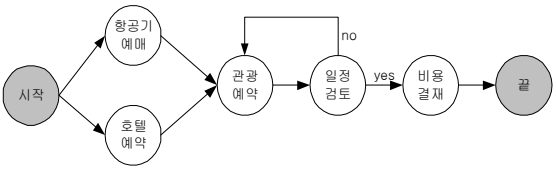

그림 2. 여행계획 워크플로우
Fig. 2. Workflow for a trip plan

본 논문을 위한 사전연구의 하나인 [18]에서는 전통적인 워크플로우 개념을 응용하여 인터넷 서비스 임대를 위한 워크플로우 기반 서비스 중개자 구현기법이 제안되었다. 여기에서 서비스 중개자는 인터넷상에 분산되어 있는 소프트웨어 서비스들을 결합하고 자동적으로 실행되기 위한 계획을 만들어서 사용자가 본 시스템을 효율적으로 활용할 수 있도록 지원해 준다. 그러나 이 시스템은 키워드 기반 검색방법을 사용하고 입력 매개변수와 워크플로우를 수동으로 작성하여야 한다. (그림 2)의 각 태스크(원으로 표시)는 웹 서비스로 대체될 수는 있으나 웹 서비스 조합을 구현하기 위해서는 여기에서 사용된 워크플로우 플랫폼의 기능 확장이 요구된다. [예 2]는 웹 서비스 응용 예를 보여주고 있다.

[예 2] 먼저 여행 날짜와 장소가 정해져 있을 때, 사용자는 "항공기 예매"와 "호텔예약" 웹 서비스는 기존의 UDDI를 통해 쉽게 발견할 수 있었다(그림 3참조). 그렇지만, "관광예약" 웹 서비스는 적절한 웹 서비스를 찾지 못하였다. 이를 완성하기 위해 사용자는 적합한 관광예약 웹 서비스를 찾아서, 이 웹 서비스를 기존의 워크플로우에 결합시켜야만 한다.

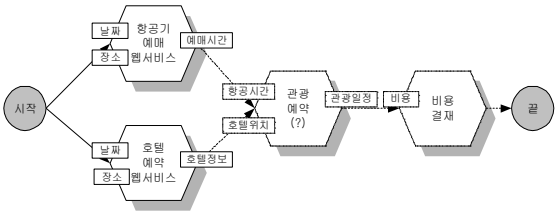

그림 3. 웹 서비스 탐색 및 조합
Fig. 3. Searching and composing of web services

여기에서 워크플로우 사용자에게 두 가지 큰 문제가 발생하게 된다.

① **웹 서비스 탐색 문제**: 전통적인 워크플로우 시스템과는 달리 인터넷상에는 수 없이 많은 웹 서비스들이 존재하므로 이들 중에서 가장 적당한 것을 찾는 일은 너무 시간이 많이 소비되고 지루한 일이 될 것이다. 더구나 인터넷상의 웹 서비스들은 입력과 출력만 기술된 거의 블랙박스처럼 보이기 때문에 더욱 이 작업을 어렵게 만든다. 따라서 수동으로 이러한 작업을 수행하는 것은 거의 불가능하게 보이며, 이를 효율적으로 지원할 수 있는 웹 서비스 탐색 메카니즘이 제공되어야만 한다.

② **웹 서비스 통합 문제**: 일단 웹 서비스가 발견되었더라도 완벽하게 워크플로우에 일치되고 상호 운용 가능하리라고는 보장할 수 없다. (그림 3)에서 사용자는 발견된 관광예약 웹 서비스의 입력부분을 항공기 예매, 호텔예약 웹 서비스 출력부분과 연결시켜야하고, 출력부분은 앞으로 탐색될 비용결재 웹 서비스의 입력부분에 연결되어야 할 것이다. 그러나 발견된 웹 서비스가 입출력 인터페이스 부분에서 구조적 또는 의미적으로 상이할 수 있으므로 이 작업을 효율적으로 지원할 수 있는 통합 메카니즘의 제공이 필요하다.

본 논문에서는 시멘틱 웹 온톨로지 개념 및 워크플로우 기법을 이용하여 이 두 가지 문제를 해결하려고 한다. 웹 서비스 탐색 및 통합 기법을 기술하기 위해 먼저 다음 장에서 웹 서비스 온톨로지 언어인 OWL-S를 간단히 살펴본다.

IV. 웹 서비스 온톨로지 언어: OWL-S

OWL-S 온톨로지[11]는 서비스를 기술하기 위해 presents, describedBy, supports의 세 가지 속성을 선언하고 있으며, 이는 ServiceProfile, ServiceModel, Service Grounding의 세 가지 클래스로 구성된다(그림 4참조). 즉, OWL-S는 기계가 처리할 수 있는 형태로 서비스를 표현하기 위해 세 가지 클래스를 이용하여 서비스 온톨로지를 기술한다.

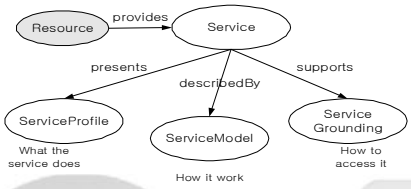

그림 4. OWL-S의 Upper Ontology
Fig. 4. Upper ontology of OWL-S

서비스 프로파일(Service Profile)은 서비스의 역할을 기술하고 있다. 즉, 사용자가 서비스 탐색 에이전트를 이용하여 원하는 서비스를 찾을 때 이를 위한 필요한 정보를 제공하고 있다. 서비스 모델(Service Model)은 서비스 작업 프로세스를 기술하고 있다. 프로세스는 구성 개수에 따라 원자 프로세스와 복합 프로세스로 구분된다. 마지막으로 서비스 그라운딩(Service Grounding)은 에이전트가 서비스를 엑세스할 수 있도록 자세한 사항을 기술하고 있다. 예를 들어, 통신 프로토콜, 메시지 형식, 또는 포트 번호 등을 명시하고 있다. 요약하면, 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. 마지막으로 특징에 관한 정보는 제공된 서비스의 카테고리(category)와 서비스의 품질 등급(quality rating)을 포함하고 있다. 예를 들어, 카테고리와 관련된 정보는 UNSPSC와 NAICS와 같은 분류 시스템을 포함할 수 있으며, 품질 등급은 "Good" 또는 "Poor" 등을 포함할 수 있다. 또한 최대 응답 시간, 지리적 범위와 같은 서비스 매개변수(parameters)를 제공하고 있다.

서비스 프로파일 내에서 위에서 설명한 각 정보들은 숫자/문자와 같은 간단한 내장 데이타 타입(built-in primitive type)이나 온톨로지에서 정의된 개념(클래스)으로 기술된다. 내장 데이타 타입은 XML 스키마 스펙[19]에서 정의된 데이타 타입을 사용하므로 string, decimal, non-negative integer, short, unsigned byte, base 64 binary 등을 포함한다. 온톨로지 개념은 객체지향 모델과 비슷한 상속이 허용된 계층 형태가 된다. 한편, 서비스 프로파일에서는 IOPE (Inputs, Outputs, Preconditions, Effects) 인스턴스를 묘사하는 스키마는 제공되지 않는다. 대신 이 스키마는 서비스 프로세스에 나타난다. 즉, 서비스 프로파일에 나타나는 IOPE는 서비스 프로세스 IOPE의 부분집합으로 볼 수 있다. 따라서 서비스 프로세스에서는 모든 IOPE 인스턴스가 묘사되고 서비스 프로파일에서는 단지 이에 대한 포인트만 존재한다. 프로세스 IOPE 역시 내장 데이타 타입이나 온톨로지 개념으로 기술될 수 있는데, 특히 온톨로지 개념들은 개념 온톨로지 파일과 연결되어 있다. 즉, 어떤 입력이나 출력 인스턴스는 온톨로지 개념 클래스와 1:1 매핑이 이루어진다. 예를 들면, 표 2에서 서비스 프로파일의 입력 RoundTrip_In에 관한 스키마는 서비스 프로세스 파일에 나타나고, 서비스 프로세스의 입력 RoundTrip은 개념 온톨로지 Concepts.owl에 정의된 RoundTrip 클래스를 참조한다.

표 2. 온톨로지 개념으로 기술된 입력 인스턴스의 예
Table 2. A example of the input instance described ontology concepts

서비스 프로파일(ServiceProfile.owl)
<profile:hasInput rdf:resource="http://www.daml.org/services/ owl-s/1.0/AirProcess.owl#RoundTrip_In" />
서비스 프로세스(ServiceProcess.owl)
<process:Input rdf:ID=RoundTrip_In"> <procee:parameterType rdf:resource="http://www.daml.org/ services/owl-s/1.0/Concepts.owl#RoundTrip" /> </process:Input>
개념 온톨로지(Concepts.owl)
<owl:Class rdf:ID="RoundTrip"> <rdfs:subClassOf rdf:resource="&xsd:#boolean" /> </owl:Class>

V. 웹 서비스 탐색 및 통합 기법

본 논문에서는 동적 웹 서비스 조합을 위해 순차적 목표 지향 접근방법을 제안한다. 워크플로우의 시작 단계에서부터 점차적으로 하나씩 새로운 웹 서비스가 첨가된다. 각 단계에서 새로운 웹 서비스를 워크플로우에 첨가시킬 때 사용자는 먼저 질의 템플릿 T를 생성한다. 일단 T가 생성되면 이는 웹 서비스 탐색 모듈로 보내지고, T와 기존 웹 서비스들 간의 유사도 측정에 따라 우선순위가 매겨진 웹 서비스들이 반환된다. 이때 사용자는 그가 의도한 바를 잘 이룰 수 있는 가장 적합한 웹 서비스를 선택한다.

5.1 질의 템플릿

사용자가 워크플로우에 첨가될 웹 서비스를 탐색할 필요가 있을 때 하나의 질의 템플릿을 작성한다. 질의 템플릿은 워크플로우가 최종적인 목표에 가깝게 도달될 수 있도록 그 단계에서 요구되는 웹 서비스의 특성을 표현한 하나의 청사진이다. 질의 템플릿 T는 다음과 같이 표현된다.

$$T = <N, D, Is, Os, [Ps, Es, C, QR, SP]>$$

여기서, N은 웹 서비스 이름, D는 텍스트 설명, Is는 입력 항목, Os는 출력 항목, Ps는 전제조건, Es는 효과, C는

카테고리, QR은 품질 등급, SP는 서비스 매개변수, [ ]는 선택사항을 표시한다. 예를 들면, [예 2]에서 찾고자 하는 관광예약 웹 서비스를 위한 질의 템플릿은 다음과 같이 표현된다. 여기서 여행사에 관한 UNSPSC 카테고리 코드는 "90121500"이라 한다.

T = <"관광예약", "일정에 따른 관광 안내 서비스", {"항공시간"}∪{"호텔위치"}, {"관광일정"}, {회원가입}, {일정 온라인 송부}, {90121500}, {"Good"}, {"한국"}>[2]

5.2 매칭(matching) 알고리즘

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.

> [정의 1] 질의 템플릿 T의 모든 출력항목들이 웹 서비스 S의 출력항목들과 매치가 되고, 이 S의 모든 입력항목들이 T의 입력항목들과 매치가 될 때, 이 S와 T는 매치된다고 할 수 있다.

[정의 1]은 매치되는 S가 T의 모든 요구사항을 만족할 수 있고, 이 S를 올바로 작동시키기 위해 필요한 모든 입력항목들은 T가 제공할 수 있다는 것을 보장한다. [정의 1]을 기반으로 한 웹 서비스 매칭 알고리즘은 그림 5와 같다. 그림 5에서 main( ) 함수는 질의 템플릿 T를 서비스 저장소(service repository)에 있는 모든 웹 서비스들과 비교한다. 만일 매치가 발견되면 기록되고 우선순위에 의해 정렬된다. match( ) 함수는 먼저 템플릿 출력항목을 웹 서비스 출력항목과 비교하여 유사도를 계산하고, 매치가 실패하지 않는다면 반대로 웹 서비스 입력항목과 템플릿 입력항목을 비교한다.

이 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.

```
Algorithm 1: Matching
  main(T) {
        record = empty;
        for each S ∈ serviceRepository) {
            if match(T, S) then record.append(S) }
        return sort(record)
  }
  match(T, S) {
        outputMatch(T{Os}, S{Os})
        inputMatch(S{Is}, T{Is})
  }
```

그림 5. 웹 서비스 매칭 알고리즘
Fig. 5. Matching algorithm for web services

OWL-S의 IOPE는 내장 데이타 타입이나 온톨로지에서 정의된 개념으로 기술될 수 있다(4장 참조). 따라서 두개의 입력항목(또는 출력항목)[3] 사이의 유사성은 두가지 경우, 즉 내장 데이타 타입과 온톨로지 개념으로 구분하여 측정할 수 있다.

5.2.1 내장 데이타 타입으로 선언된 경우

질의 템플릿 입력항목이 내장 데이타 타입으로 선언된 경우에는 서비스 저장소에 있는 웹 서비스들 중에서 입력항목이 내장 데이타 타입으로 선언된 것들만 간단하게 비교하면 된다. 실제로 웹 온톨로지 파일에서 IOPE 정보들이 내장 데이타 타입으로 선언된 경우가 많으므로 이에 관한 유사도 측정은 최종 결과에 매우 중요한 영향을 미칠 수 있다.

내장 데이타 타입과 관련된 유사도(Similarity) SM 함수는 WfMS 데이타 타입 변환 기능과 밀접한 연관이 있다. 일반적으로 WfMS에서는 태스크 사이에 데이타 전달을 할 때 데이타 타입 변환을 지원하는 능력을 가지고 있다. 이는 시스템 내에서 기본적으로 정의될 수도 있으나 요구사항에 맞게 사용자에 의해 정의될 수도 있다. 예를 들면, 질의 템플릿 입력항목이 integer로 선언되어 있으나 웹 서비스 입력항목은 decimal인 경우, WfMS가 integer를 decimal로 데이타 타입 변환을 할 수 있다면 이의 유사도 SM 값은 1로 결정된다. 값 1은 유사도가 최대임을 의미한다. 그러나 WfMS에서 데이타 타입 변환이 불가능한 경우에는 웹 서비스 실행이 불가능하므로 SM 값은 0으로 되며, 이는 유사도가 전혀 없음을 의미한다. 본 논문에서 유사도 SM 측정값은 XML 스키마 데이타 타입 계층도[18]를 기반으로 하고

2) OWL-S 버전 1.0에서는 전제조건(Ps)과 효과(Es)를 어떻게 표현할지 아직까지 공식적인 스펙이 정해지지 않았다.

3) 본 논문에서는 편의상 입력 항목만 예시한다.

있으며 통합 메카니즘을 위해 0과 1 사이의 유사도 값을 산출한다. <표 3>은 (그림 6)을 이용한 3가지 [예case1~3]에 대한 유사도를 계산한 것이다.

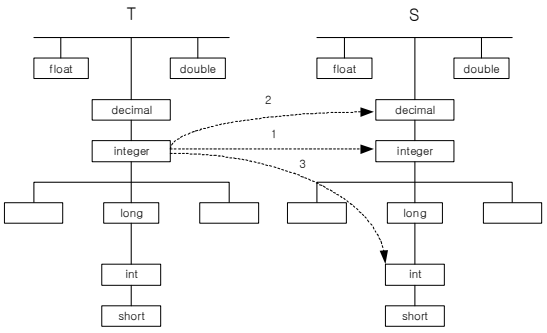

그림 6. XML 스키마 데이타 타입 계층도
Fig. 6. Hierarchy of XML schema data types

표 3. 세가지 경우에 있어서 유사도 값
Table 3. Similarity values in three cases

	Condition	SM
case 1	T{I}와S{I}가 같은 경우	1
case 2	T{I}가 S{I}의 하위 개념인 경우	1
case 3	T{I}가 S{I}의 상위 개념이거나 관계가 없는 경우	α

α값은 <표 4>에서와 같이 유사도 측정치를 사전에 정의하여 검색항목이 내장 데이타 타입으로 선언된 경우 관련되는 SM 값을 산출한다. 예를 들면, double에서 int의 SM 값은 1/3로 정해져 있는데, 이런 경우에는 WfMS에서 데이타 타입 변환은 허락되지만 어떤 경우 데이타 손실이 발생될 수도 있기 때문에 별로 선호되지 않는 경우로써 SM 값도 비교적 낮은 편으로 정의된다.

표 4. 유사도 측정치 예
Table 4. Examples of similarity measurements

T{I}	S{I}	SM	T{I}	S{I}	SM
integer	long	2/3	long	int	2/3
integer	int	2/3	string	float	0
double	int	1/3	date	time	0
int	string	1	date	gYear	1/3

5.2.2 온톨로지 개념으로 선언된 경우

질의 템플릿 입력항목이 온톨로지 개념으로 선언된 경우에는 객체지향 모델과 같은 계층 관계에 따라 유사도가 결정된다. 예를 들면, (그림 7)의 Vehicle 온톨로지에서 부/자 관계에 있는 Vehicle과 Car는 유사성이 높지만 관계 설정이 없는 Vehicle과 Price 간에는 유사성이 없다. 온톨로지 개념 간 유사성 관계는 다음과 같이 5가지 경우가 발생될 수 있다.

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

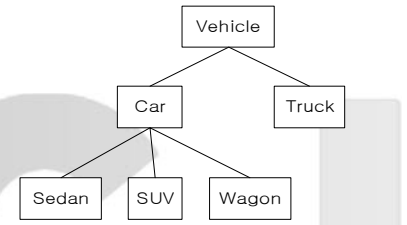

그림 7. Vehicle 온톨로지
Fig. 7. Vehicle ontology

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

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

여기서[4],

4) ∂ = | T.Ps ∩ S.Ps |

$$SM = \begin{cases} 1 & \text{if } Exact \\ 1 & \text{if } PlugIn \\ \frac{|T.Ps|}{|S.Ps|} & \text{if } Subsumes \\ \frac{\partial}{\{|T.Ps|-\partial\}+\{|S.Ps|-\partial\}+\partial} & \text{if } Intersect \\ 0 & \text{if } Fail \end{cases}$$

[case 1]은 가장 간단한 경우이다. 두개의 개념이 같으면 Exact가 리턴되고 그 값은 1이다. [case 2]의 경우, S는 T의 상위 개념에 있으므로 S는 T에 펄럭 인(Plug In)된다고 할 수 있고 결과 값은 아직도 1로 평가된다. 실 예로, (그림 7)에서 Car에 관한 웹 서비스는 Sedan, SUV, Wagon에 관한 사항도 지원될 수 있다. [case 3]의 경우에는 T가 S의 상위 개념에 있다. 이 경우 선택된 S는 T의 모든 사항을 다 만족시켜주지 못한다. S가 사용될 수도 있으나 목표를 완전히 수행하기 위해서는 수정되거나 다른 대안을 찾아야 한다. 유사도는 Subsumes가 리턴되고 그 값은 개념 T 속성의 개수 |T.Ps| 대 개념 S 속성의 개수 |S.Ps| 비율로 평가된다. case 4는 T와 S 사이에 부/자 관계가 아닌 일부분 공통된 속성이 존재하는 경우이다. 이런 Intersect인 경우 유사도는 양쪽의 공통된 속성 개수를 발견하여(즉, $\partial = |T.Ps \cap S.Ps|$) 개념 T에만 있는 속성($|T.Ps| - \partial$), 개념 S에만 있는 속성($|S.Ps| - \partial$), 그리고 양쪽 공통된 속성($\partial$)을 모두 더한 개수의 비율로 평가된다. 이는 Tversky 특징기반 유사도 모델[20]을 응용한 것으로써, 유사도는 두 개념 사이의 차이점 보다는 공통된 속성에 더 많은 영향을 받는다는 이론에 그 바탕을 두고 있다. 마지막으로 case 5는 T와 S 사이에 전혀 공통된 속성이 없는 경우이므로 Fail이 반환되고 그 값은 0이 된다.

지금까지는 편의상 설명을 간단히 하기 위해 입력항목이 한개인 경우만을 다루었다. 만일 입력항목이 여러 개 존재하는 경우에는 전체의 평균값을 유사도로 설정한다. 즉 전체 유사도 Similarity는 다음과 같이 표현된다.

$$Similarity = \frac{\Sigma\, SM_i}{n}$$

예를 들면, case1과 case3 두개의 입력항목이 존재한다면,

$$Similarity = \frac{1 + \frac{|T.Ps|}{|S.Ps|}}{2}$$

로 평가된다. 또한, 입력항목과 출력항목이 모두 존재하는 경우에도 같은 방법으로 전체 평균값이 유사도로 평가된다. 전체 평균값으로 표현된 유사도 값은 웹 서비스 통합 문제 해결을 위한 하나의 평가 기준치를 제공한다.

5.3 웹 서비스 워크플로우

워크플로우 시작에서부터 점차적으로 하나씩 새로운 서비스가 첨가되어 웹 서비스 워크플로우가 완성되고 나면 워크플로우 엔진에 의해 이들 프로세스들이 자동으로 수행되어야 한다. 웹 서비스 워크플로우는 전통적인 워크플로우와는 달리 인터넷상에 수 많은 공급자와 수요자가 관련되어 있기 때문에 여러 사이트에 분산 저장되어 있는 이질적 서비스들을 통합·관리 할 필요성이 있다.

OWL-S는 웹 서비스의 지능적 탐색 및 프로세스 실행은 지원하고 있으나 이질적 분산 환경에서 비지니스 프로세스의 조합을 위한 기능은 제공하지 못하고 있다[11]. 이러한 필요성에 의해 BPEL4WS, WSFL, XLang, WSCI, WS-CDL 등 여러 가지 웹 서비스 조합언어가 발표되었으나 아직까지는 표준화가 정해져 있지 왔고, 이들 언어들은 시멘틱 개념을 지원해 주지 못하고 있다[21]. 따라서 본 연구에서는 BPEL4WS 스펙을 기반으로 자체 웹 서비스 워크플로우 실행계획 SEP(Service Execution Plan)을 개발하였고 향후 OWL-S 등이 활용가능하면 이를 대체할 수 있도록 하였다.

SEP은 XML로 표현된 워크플로우 프로세스로써 태스크(task)와 제어 구성자(control construct)의 집합으로 구성된다. 태스크는 서비스 그라운딩 파일로부터 추출된 포트 타입과 작업(operation), 그리고 입력과 출력 매개변수를 가지고 있으며, 최종적으로 WSDL 파일과 바인딩하게 된다. 제어 구성자는 4개의 형태 즉, 순차(sequence), 반복(while), 병행(flow), 선택(switch)을 가지고 있다. 순차 블록에서는 단위 업무(activity)들이 하나씩 순차적으로 수행되고, 반복 블록은 리스트의 각 멤버들을 반복적으로 수행한다. 그리고 병행과 선택 블록에서는 단위 업무가 병렬로 수행되거나 여러 개 중 하나가 선택되어 수행된다. 그림 8은 [예 2]의 워크플로우를 SEP XML 문서로 표현한 것이다.

작성된 SEP을 이용하여 워크플로우 실행을 요청하면 계획 에이전트는 먼저 SEP XML 문서를 파싱하여 실행계획 알고리즘을 작성한다. SEP XML 문서 파싱을 위해서 본 연구에서는 W3C(World Wide Web Consortium)에서 정한 DOM(Document Object Model) 인터페이스를 사용하였다(자세한 내용은 [18] 참조).

```
<process name="관광예약 실행계획">
<flow>
 <task name="항공기예매"
   location="http://www.daml.org/services/BrovoAir">
  <portType="ReservePT" />
  <operation="Reserve" />
  <input><parameter name="날짜" type="date"></parameter>
          <parameter name="장소" type="string">
          </parameter>
  </input>
  <output><parameter name="예매시간" type="date">
  </parameter></output>
 </task>
 <task name="호텔예약"
   location="http://www.daml.org/services/HotelInfo">
  <portType="ReservePT" />
  <operation="Reserve" />
  <input><parameter name="날짜" type="date"></parameter>
          <parameter name="장소" type="string">
          </parameter>
  </input>
  <output><parameter name="호텔정보"><type>string</type>
  </parameter></output>
 </task>
</flow>
<sequence>
 <task name="관광예약"
   location="http://www.daml.org/services/TourGuide">
  <portType="ReservePT" />
  <operation="Reserve" />
  <input><parameter name="항공시간" type="date">
          </parameter>
          <parameter name="호텔위치" type="string">
          </parameter></input>
  <output><parameter name="관광일정"><type>string</type>
           </parameter></output>
 </task>
 <task name="비용결재"
   location="http://www.daml.org/services/CostPay">
  <portType="#PayPT" />
  <operation="Pay" />
  <input><parameter name="관광일정" type="string">
          </parameter></input>
 </task>
</sequence>
</process>
```

그림 8. 웹 서비스 워크플로우 실행계획
Fig. 8. Execution plan of web service workflow

이 알고리즘에 의해 워크플로우 엔진에서 웹 서비스가 실행되는 내부 과정을 살펴보면, 먼저 입력 매개변수를 받고(receive), 관련 웹 서비스를 호출(invoke)하여, 에이전트에게 응답(reply)하는 3단계로 이루어진다. 예를 들면, 그림 9에서와 같이 워크플로우 클라이언트에서 날짜와 장소 입력 매개변수를 워크플로우 엔진에게 보내면 태스크 클래스가 생성되고 receive 메쏘드가 시작된다. 이 매개변수가 입력 컨테이너에게 넣어진 후, 곧바로 원격지에 있는 "항공기예매" 웹 서비스가 호출(invoke)된다. 웹 서비스가 실행된 후 그 결과가 출력 컨테이너에 넣어지고, 마지막으로 reply 메쏘드가 수행되어 계획 에이전트에게 그 결과가 보내진다. 한편, 태스크가 여러개 존재하는 복잡한 워크플로우인 경우에도 이러한 과정이 연속적으로 진행된다.

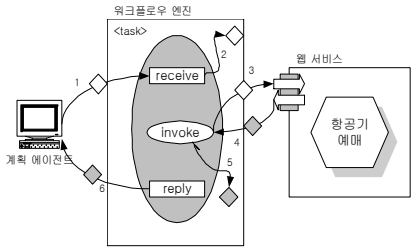

그림 9. 웹 서비스 실행 내부 흐름
Fig. 9. Internal flow for executing web services

VI. 시스템 구현

구현 시스템은 본 연구실에서 이미 개발되어 있는 워크플로우 기반 서비스 중개자 시스템 IMP(Internet MarketPlaces)[18]를 확장 발전시켰다. IMP의 전체적인 구조는 그림 10과 같이 서비스 공급자(supplier)와 사용자(customer), 그리고 서비스 중개자(service broker)로 구성되어 있다. 서비스 중개자는 계획 에이전트(planning agent), 워크플로우 엔진(workflow engine), 그리고 웹 서비스 탐색(search) 및 등록(register) 모듈로 이루어져 있으며, 중개자는 IMP의 미들웨어(middleware) 역할을 수행한다. 이에 대한 자세한 설명은 다음과 같다.

- **서비스 공급자**: 제공되는 웹 서비스를 저장하고 있다. OWL-S로 작성된 웹 서비스 객체를 등록하기 위해 레지스터에 억세스 한다.
- **사용자**: 웹 브라우저를 통해 웹 서비스 탐색 및 워크플로우 실행을 요구한다.
- **계획 에이전트**: 사용자의 요청을 받아들여서 실행 계획을 작성하고 워크플로우 엔진과 상호 작용한다.

- **워크플로우 엔진**: 프로세스 자동화를 위해 워크플로우의 수행을 정의·생성·관리하는 시스템이다.
- **웹 서비스 탐색 및 등록 모듈**: OWL-S 파서, 웹 서비스 등록기, 탐색엔진으로 구성되어 있으며, 웹 서비스 등록 및 탐색을 수행한다.

본 연구에서는 웹(Web) 프로그래밍을 위해 Microsoft의 ASP.NET을 사용하였고, 워크플로우 엔진을 구현하기 위해 COM+(Component Object Model plus)를 사용하였다. COM+ 컴포넌트 서비스는 다층(multi-tier) 구조 분산시스템 구현을 위한 미들티어(middle-tier) 트랜잭션 관리 메커니즘 역할을 수행한다. 사용자와 서비스 중계자 사이의 자료 교환은 XML을 통해 이루어진다.

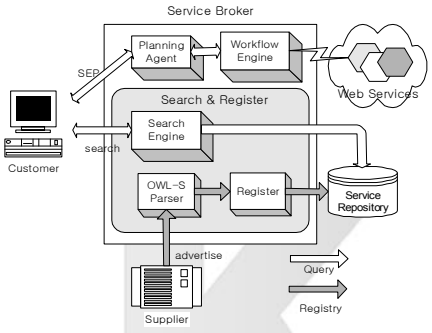

그림 10. IMP 시스템 구조
Fig. 10. IMP system architecture

VII. 결론

본 논문에서는 비즈니스 분야에서 성공적으로 활용되고 있는 워크플로우 기법을 적용하여 동적 웹 서비스 조합을 구현하였다. 워크플로우의 웹 서비스 적용은 아직까지는 새로운 분야로써 웹 서비스 탐색 및 웹 서비스 통합 문제 두 가지 문제 해결을 요구한다. 본 연구에서는 탐색 문제를 위해 웹 서비스 매칭 알고리즘이 제안되고, 통합 문제를 위해 시멘틱 웹 온톨로지 개념이 적용된다. 그리고 어떻게 동적 웹 서비스 조합이 구현되는지 설명하기 위해 웹 서비스 워크플로우 실행 계획을 개발하였고, 계획 에이전트와 워크플로우 엔진에 의한 이의 실행 과정을 보였다.

시멘틱 웹 서비스 조합 문제는 현재 연구가 활발히 진행되고 있는 분야이므로 아직까지 명확한 개념 설정이나 표준화 작업이 완료되어 있지 않은 상황이다. 따라서 본 논문에서도 실제 웹 서비스 저장소 구축 및 이에 대한 성능분석은 수행되지 못하였다. 또한 제안 알고리즘이 적용되었을 때 사용자의 최종 요구사항을 얼마만큼 만족시켜 줄 수 있는지 분석될 수 있는 QoS(Quality of Service)의 측정이 이루어지지 않았다. 이들 분야에 대한 향후 연구가 필요하다.

참고문헌

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

[2] Arpinar I.B., Aleman-Meza B., Zhang R., and Maduko A., Ontology-Driven Web Services Composition Platform, IEEE Conference on E-Commerce Technology(CEC 2004), San Diego, California, July 2004

[3] W3C, SOAP Version 1.2 Part0: Primer, http://www.w3.org/TR/soap12-part0/, 2003

[4] W3C, Web Services Description Language (WSDL) Version 2.0 Part1: Core Language, "http://www.w3c.org/TR/wsdl20/, 2003"

[5] UDDI, UDDI Technical White Paper, http://www.uddi.org/, 2000

[6] Leymann P., Web Service Flow Language(WSFL) 1.0, http://www-4.ibm/software/solutions/webservices/pdf/WSFL.pdf, 2002

[7] Thatte S., XLang: Web Services for Business Process Design, http://www.gotdotnetcom/team/xml_wsspecs/xlang-c/default.htm, 2002

[8] 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
[9] Berners-Lee T., Hendler J., and Lassila O., The Semantic Web, Scientific American 284(5):34-43, 2001
[10] Dean M., et al., Web Ontology Language(OWL) Reference Version 1.0, http://www.w3.org/TR/2002 /WD-owl-ref-20021112/, 2002
[11] 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
[12] Sirin E., Parsia B., and Hendler J., Composition-driven Filtering and Selection of Semantic Web Service, In AAAI Spring Symposium on Semantic Web Services, 2004
[13] Casati F., Ilnicki S., Jin L., Krishnamoorthy V., and Shan M., Adaptive and Dynamic Service Composition in eFlow, In Proceedings of the International Conference on Advanced Information Systems Engineering, 2000
[14] Shankar P., Fox A., SWORD: A Developer Toolkit for Web Service Composition, In Proceedings of the Eleventh International World Wide Web Conference, 2002
[15] Cardoso J., Sheth A., Semantic e-Workflow Composition, Journal of Intelligent Information Systems(JIIS), 2003
[16] WfMC, WfMC(Workflow Management Coalition) Strand Documents, Technical Report, 1998
[17] 오종태, DB에이전트를 이용한 전자상거래 워크플로우 모델링 도구 설계, 한국컴퓨터정보학회논문지, 제8권 제3호, 2003
[18] 이용주, 인터넷 서비스 임대를 위한 워크플로우 기반 서비스 중개자 구현기법, 정보처리학회논문지D, 제9-D권 제2호, 277-288, 2002
[19] XML Schema Part2, hppt://www.w3.org/TR /xmlschema-2/
[20] Bradshaw J., Introduction to Tversky Similarity Measure, MUG'97: 11th Annual Daylight User Group Meeting, Feb 1997
[21] Cheng Yushi, Lee Eng Wah, Dilip K. Limbu, Web Services Composition: An Overview of Standards, http://www.itsc.org.sg/synthesis/2004/4_WS.pdf.

저 자 소 개

이용주

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~현재 상주대학교 컴퓨터공학과 부교수