동적 웹 서비스 조합을 위한 시멘틱 웹 서비스 발견 및 실행 기법

이 용 주†

요 약

최근에 동적으로, 즉 요구가 있는 즉시 서비스들을 조합하는 동적 웹 서비스 조합은 가장 큰 관심사 중 하나이다. 지금까지 동적 웹 서비스 조합에 관한 많은 연구들이 수행되어졌는데 주로 사람이 개입된 반자동 대화식 조합 기법들이 제안되었다. 또한 이들 연구에서 탐색 알고리즘 및 워크플로우 설계 기법들이 부분적으로는 기술되어 있으나, 웹 서비스 조합 자동화를 위한 전반적인 워크플로우 처리과정을 기술한 연구는 아직 없다. 본 논문은 웹 서비스 조합 자동화 기법에 초점을 맞추고 있다. 주된 아이디어는 시멘틱 웹을 실현하기 위한 웹 온톨로지와 비즈니스 분야에서 성공적으로 활용되고 있는 워크플로우 기법들을 적용하여 웹 서비스 조합 자동화 시스템을 구현하는 것이다. 본 연구에서는 웹 서비스 발견을 위해 매칭 알고리즘이 제안되고, 웹 서비스들 간 상호연결을 지원하기 위해 매칭 알고리즘에 온톨로지 개념이 적용된다. 그리고 웹 서비스 조합 자동화를 지원하기 위한 워크플로우 실행계획이 기술된다. 마지막으로 실 데이터 실험을 통해 제안 시스템의 우수성을 보인다.

키워드 : 웹 서비스 조합, 웹 온톨로지, 매칭 알고리즘, 워크플로우 실행계획

Discovery and Execution Techniques of Semantic Web Services for Dynamic Web Services Composition

Yong-ju Lee†

ABSTRACT

Recently, one of the most challenging problems is to compose web services dynamically, that is, on demand. A number of researchers have been considerably interested in the dynamic web services composition. However, while most of them focused on the semi-automatic web services composition with a human controller, very little attention was devoted to the full automation of this process. This paper primarily focuses on the automatic web services composition techniques. The main idea is to implement an automatic web services composition system using web ontologies to realize the semantic web and workflow technologies to play a major role in E-businesses. In this paper, we propose a matching algorithm for web service discovery and present an approach based on the use of ontologies to facilitate the interoperability of web services. Finally, we describe a workflow execution plan to support the automatic web services composition, and an experimental study that shows the high performance of our system.

Key Words : Web Services Composition, Web Ontologies, Matching Algorithm, Workflow Execution Plan

1. 서 론

웹 서비스 기술(web services technologies)은 한 기업이나 다수 기업 간에 이기종 컴퓨터 프로그램들을 통합할 수 있는 하나의 강력한 수단으로써 최근에 많은 관심을 받고 있다. 특히, 웹 서비스 조합(composition)은 분산되어 있는 이질적이고 독자적인 응용프로그램들의 효율적인 통합 방법으로써 가장 관심의 초점이 되고 있다[1]. 웹 서비스 조합에서 각 응용프로그램들은 개별 웹 서비스로 캡슐화되고 그들 간의 상호작용은 워크플로우(workflow) 프로세스 모델로써 구현된다. 이 방법은 기존의 상용 프로그램을 이용한 분산 프로그래밍 방법을 대체할 수 있는 상당히 매력적인 방안으로써 대두되고 있다.

웹 서비스는 다른 웹 서비스들과 상호작용하기 위해 HTTP, XML, SOAP, WSDL, UDDI와 같은 표준적인 인터넷 프로토콜을 사용하는 플랫폼 독립적인 응용 프로그램 로직이다. 웹 서비스의 한 예는 항공기 예약 서비스가 될 수 있으며, 이 서비스는 독립적으로 사용될 수도 있고, 보다 복잡한 비즈니스 처리를 위하여 다른 웹 서비스들과 조합되어 사용될 수도 있다. 웹 서비스 조합의 예는 여행계획 시스템

※ 이 논문은 2004년도 한국학술진흥재단의 지원에 의하여 연구되었음(KRF-2004-002-D00339).
† 정 회 원 : 상주대학교 컴퓨터공학과 교수
논문접수 : 2005년 7월 19일, 심사완료 : 2005년 10월 26일

이 될 수 있는데, 이 시스템은 항공기 예약, 숙박예약, 차 임대, 그리고 관광일정 등 다수 웹 서비스들을 결합하고 이 서비스들을 순차 또는 병행 처리한다.

웹 서비스 조합은 크게 정적(static)과 동적(dynamic)으로 구분할 수 있으며, 정적은 서비스 조합이 설계 시에 결정되고, 동적은 실행 시에 결정된다. 복잡한 비즈니스 처리를 위해 결합되는 서비스들의 수는 상당히 많을 수도 있고 계속 변화될 수도 있다. 따라서 설계 시에 정확한 웹 서비스들을 결정지어야 하는 정적인 접근방법은 웹 서비스 조합을 위해서는 부적합하다[2]. 워크플로우 실행 시에 웹 서비스들을 선택하는 동적 접근방법은 최근에 큰 관심사이며, 지금까지 많은 연구들[2-4]이 수행되어졌지만 주로 사람이 개입된 반자동 대화식 조합 기법들이 제안되었다. 또한 이들 연구에서 탐색 알고리즘 및 워크플로우 설계 기법들이 부분적으로는 기술되어 있으나, 웹 서비스 조합 자동화를 위한 프로세스 작성 및 실행계획 등 전반적인 워크플로우 처리과정을 기술한 연구는 아직 없다.

본 논문은 웹 서비스 조합 자동화 기법에 초점을 맞추고 있다. 주된 아이디어는 시멘틱 웹을 실현하기 위한 웹 온톨로지(ontology)와 비즈니스 분야에서 성공적으로 활용되고 있는 워크플로우 기법들을 적용하여 웹 서비스 조합 자동화 시스템을 구현하는 것이다. 웹 서비스 조합 자동화에 관한 연구는 Maryland 대학의 Mindswap과 Georgia 대학의 LSDIS 연구실에서 활발히 진행되고 있다. Maryland 대학에서 개발된 WS-Composer[2]는 OWL-S[5] 시멘틱 웹 서비스 언어를 사용하여 동적 웹 서비스 조합을 지원하는 시스템으로써, 조합을 수행하는 동안에 자체 탐색 알고리즘을 적용하여 목표 지향적이고 대화식 조합 기법을 제안하고 있다. 그러나 이 시스템은 워크플로우 e-비지니스 업무 프로세스는 고려하지 않았기 때문에 사람이 개입된 반자동 시스템으로만 작동되는 단점이 있다. Georgia 대학에서 개발된 METEOR-S[6]에서는 워크플로우 기반 시멘틱 웹 서비스 발견 및 실행 알고리즘들이 제안되고 있다. 그러나 이 시스템은 정적 웹 서비스 조합 접근방법이며 자동화를 위한 실행계획 메카니즘은 지원하지 못하고 있다.

본 논문에서는 동적 웹 서비스 조합을 지원하기 위해 Maryland 대학 방식의 순차적 목표지향 접근방법을 제안한다. 다만, 반자동으로 실행되는 이 방식의 단점을 해결하기 위해 Georgia 대학의 METEOR-S처럼 워크플로우 기반 웹 서비스 발견 및 실행 기법들이 새로 제안되고, 전체적인 웹 서비스 조합 자동화를 위해 워크플로우 실행계획이 적용된다.

본 논문의 구성은 다음과 같다. 2장에서는 동적 웹 서비스 조합 시스템 전체 구조를 설명하고, 3장에서는 웹 서비스 온톨로지 언어인 OWL-S를 간단히 살펴보고 온톨로지 개념을 이용한 웹 서비스 탐색 알고리즘을 제안하고 워크플로우 실행계획을 기술한다. 4장에서 실험 결과를 분석하고 5장에서 관련연구를 서술한다. 마지막으로 6장에서 결론을 내렸다.

2. 시스템 구조

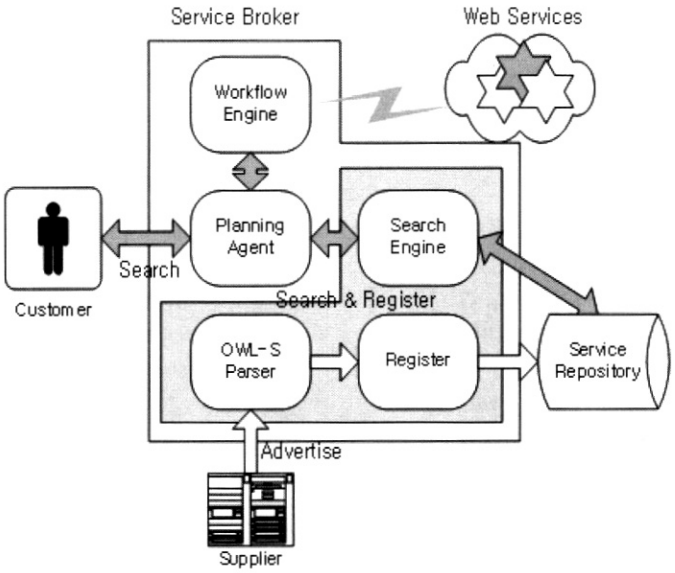
본 논문에서 구현된 동적 웹 서비스 조합(DWSC : Dynamic Web Services Composition) 시스템의 전체적인 구조는 (그림 1)과 같이 서비스 공급자(supplier)와 사용자(customer), 그리고 서비스 중개자(service broker)로 구성되어 있다. 서비스 중개자는 계획 에이전트(planning agent), 워크플로우 엔진(workflow engine), 그리고 웹 서비스 탐색(search) 및 등록(register) 모듈로 이루어져 있으며, 중개자는 DWSC 시스템의 미들웨어(middleware) 역할을 수행한다. DWSC 시스템은 본 연구실에서 이미 개발되어 있는 ASP(Application Service Provider)를 위한 워크플로우 기반 서비스 중개자 시스템인 IMP(Internet MarketPlaces)[7]를 확장 발전시켰다. 구현 시스템은 기존의 IMP 시스템과 개념 및 구조적인 면에서 큰 차이가 없으나 웹 서비스 탐색 및 등록 모듈이 추가 개발되었다. 이 모듈에서 OWL-S 파서는 Maryland 대학의 OWL-S API 모듈[8]을 이용하고 있다. (그림 1)에 대한 자세한 설명은 다음과 같다.

- 서비스 공급자 : 웹 서비스 메타 정보를 등록한다. OWL-S로 작성된 웹 서비스 객체를 등록하기 위해 등록 모듈을 엑세스 한다.
- 사용자 : 계획 에이전트를 접속하여 웹 서비스 조합을 수행한다.
- 계획 에이전트 : 프로세스 템플릿을 이용하여 실행 계획을 작성하고, 워크플로우 엔진과 상호 작용한다.
- 워크플로우 엔진 : 워크플로우 실행 계획을 실행한다.
- 웹 서비스 탐색 및 등록 모듈 : OWL-S 파서, 웹 서비스 등록기, 탐색엔진으로 구성되어 있으며, 웹 서비스 등록 및 탐색을 수행한다.

DWSC 시스템에서 공급자는 메타 데이터를 사용하여 웹 서비스를 등록할 수 있어야 하고 사용자는 프로세스 템플릿을 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 전달되며, 등록기에서는 SQL(Structured Query Language) 입력문을 생성하고 실행하여 메타 데이터를 서비스 저장소(service repository)에 저장한다. 한편, 사용자는 계획 에이전트로부터[1] 실행 계획을 작성한 후 워크플로우 엔진을 통해 웹 서비스 조합을 수행한다.

사용자 처리과정을 좀 더 자세히 살펴보면, 사용자는 먼저 WfMS 프로세스 디자이너를 사용하여 워크플로우 및 프로세스 템플릿을 작성하는데, 템플릿은 XML로 표현된 액티비티(activity)와 제어 구성자(control construct)로 구성된다. 액티비티는 입력과 출력 매개변수를 가지고 있으며, 제어

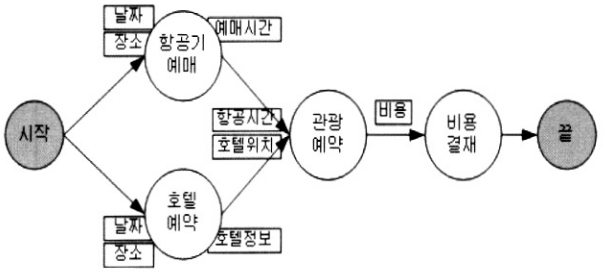
1) 계획 에이전트의 역할 및 기능에 관한 사항은 지면상 생략하고 자세한 내용은 참고문헌 [7]을 참조하기 바람.



(그림 1) DWSC 시스템의 구조

구성자는 4개의 형태 즉, 순차(sequence), 반복(while), 병행(flow), 선택(switch)을 가지고 있다. (그림 2)와 (그림 3)은 워크플로우 및 프로세스 템플릿의 예를 보여주고 있다. 템플릿이 완성되면 워크플로우 프로세스를 실행하기 위한 사전 작업으로 웹 서비스 탐색 및 입출력 인터페이스 연결을 수행한다. 즉 템플릿의 각 액티비티는 탐색 엔진의 웹 서비스 탐색 알고리즘에 의해 관련 웹 서비스가 검색되고, 선택된 웹 서비스는 프로세스 전후 관계에 따라 입출력 인터페이스가 연결되며 최종적으로 실행계획이 작성된다. 이 실행 계획은 워크플로우 엔진에 의해 웹 서비스 프로세스가 실행된다.

이러한 처리과정에서 DWSC 시스템에서는 기존 워크플로우 시스템에서는 볼 수 없었던 새로운 웹 서비스 탐색 문제의 해결을 요구한다. DWSC 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. 더구나 인터넷상의 웹 서비스들은 입력과 출력만 기술된 거의 블랙박스처럼 보이기 때문에 이 작업을 더욱 어렵게 만든다. 따라서 수동으로 이러한 작업을 수행하는 것은 거의 불가능하게 보이며, 이를 효율적으로 지원할 수 있는 새로운 웹 서비스 탐색 메카니즘이 제공되어야만 한다.



(그림 2) 여행계획 워크플로우

```
<process-template name="여행계획">
  <flow>
    <activity name="항공기예매">
      <parameter = "parameter-1"/>
    </activity>
    <activity name="호텔예약">
      <parameter = "parameter-2"/>
    </activity>
  </flow>
  <sequence>
    <activity name="관광예약">
      <parameter = "parameter-3"/>
    </activity>
    <activity name="비용결재">
      <parameter = "parameter-4"/>
    </activity>
  </sequence>

  <parameter name="parameter-1">
      <input name="날짜" type="xsd:date"/>
      <input name="장소" type="xsd:string"/>
      <output name="예매시간" type="xsd:date"/>
  </parameter>
  <parameter name="parameter-2">
      <input name="날짜" type="xsd:date"/>
      <input name="장소" type="xsd:string"/>
      <output name="호텔정보" type="xsd:string"/>
  </parameter>
  <parameter name="parameter-3">
      <input name="항공시간" type="xsd:date"/>
      <input name="호텔위치" type="xsd:string"/>
      <output name="비용" type="xsd:string"/>
  </parameter>
  <parameter name="parameter-4">
      <input name="비용" type="xsd:string"/>
  </parameter>
</process-template>
```

(그림 3) 프로세스 템플릿의 예

3. 웹 서비스 탐색 및 실행 기법

본 장에서는 웹 서비스 탐색 및 실행 기법을 기술하기 위해 먼저 OWL-S를 간단히 살펴보고, 온톨로지 개념을 이용한 웹 서비스 탐색 알고리즘과 워크플로우 실행계획을 기술한다.

3.1 OWL-S

OWL-S[5] 온톨로지는 웹 서비스를 기술하기 위해 presents, describedBy, supports의 세 가지 속성을 선언하고 있으며, 서비스 프로파일(Service Profile), 서비스 모델(Service Model), 그리고 서비스 그라운딩(Service Grounding)의 세 가지 클래스로 구성된다.

서비스 프로파일은 서비스의 역할을 기술하고 있다. 즉, 사용자가 서비스 탐색 에이전트를 이용하여 원하는 서비스를 찾을 때 이를 위한 필요한 정보를 제공하고 있다. 서비스 모델은 서비스 작업 프로세스를 기술하고 있다. 프로세스는 구성 개수에 따라 원자 프로세스와 복합 프로세스로 구분된다. 마지막으로 서비스 그라운딩은 에이전트가 서비스를 엑세스할 수 있도록 자세한 사항을 기술하고 있다. 예를 들면, 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, 서비스를 처리하기 위한 기능에 대한 정보, 그리고 서비스의 특성을 구체화한 특징에 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, 즉 전제조건(preconditions)과 서비스를 통한 효과(effects)에 대한 정보를 기술하고 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[9]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.

3.2 웹 서비스 탐색 알고리즘

본 논문에서는 동적 웹 서비스 조합을 위해 순차적 목표지향 접근방법을 제안한다. 워크플로우의 시작 단계에서부터 점차적으로 하나씩 새로운 웹 서비스가 탐색된다. 각 단계에서 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.

질의 템플릿 Q는 워크플로우가 최종적인 목표에 가깝게 도달될 수 있도록 그 단계에서 요구되는 웹 서비스의 특성을 표현한 하나의 청사진이다. Q는 다음과 같이 표현된다.

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

여기서, N은 웹 서비스 이름, D는 텍스트 설명, Is는 입력항목, 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.

정의: 질의 템플릿 Q의 모든 출력항목들이 웹 서비스 S의 출력항목들과 매치가 되고, 이 S의 모든 입력항목들이 Q의 입력항목들과 매치가 될 때, 이 S와 Q는 매치된다고 할 수 있다.

[정의]는 매치되는 S가 Q의 모든 요구사항을 만족할 수 있고, 이 S를 올바로 작동시키기 위해 필요한 모든 입력항목들은 Q가 제공할 수 있다는 것을 보장한다. [정의]를 기반으로 웹 서비스 탐색 알고리즘을 작성하면 (그림 4)와 같다. (그림 4)에서 discovery( ) 함수는 질의 템플릿 Q를 서비스 저장소에 있는 모든 웹 서비스들과 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 Repository) 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

위 알고리즘에서 어떤 서비스들은 정확하게 매치되어 최종 결과로 선택되어 질 수도 있지만, 정확하지 않더라도 무

조건 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.

OWL-S의 입출력은 내장 데이타 타입이나 온톨로지에서 정의된 개념으로 기술될 수 있다(3.1 참조). 따라서 두개의 입력항목(또는 출력항목)[2] 사이의 유사성은 두 가지 경우, 즉 내장 데이타 타입과 온톨로지 개념으로 구분하여 측정할 수 있다.

① 검색 항목이 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 입출력 정보들이 내장 데이타 타입으로 선언된 경우가 많기 때문에 간단히 이들을 먼저 계산하는 것이 전체 성능 향상에 도움을 줄 수 있다.

내장 데이타 타입과 관련된 유사도(Similarity) Ω 함수는 WfMS(Workflow Management System) 데이타 타입 변환 기능과 밀접한 연관이 있다. 일반적으로 WfMS에서는 액티비티 사이에 데이타 전달을 할 때 데이타 타입 변환 기능을 지원하고 있다. 즉, 질의 템플릿 입력항목이 integer로 선언되어 있으나 웹 서비스 입력항목은 decimal인 경우, WfMS가 integer를 decimal로 데이타 타입 변환을 할 수 있다면 이의 유사도 Ω 값은 1로 결정된다. 값 1은 유사도가 최대임을 의미한다. 그러나 WfMS에서 데이타 타입 변환이 불가능한 경우에는 웹 서비스 실행이 불가능하므로 Ω 값은 0으로 되며, 이는 유사도가 전혀 없음을 의미한다. 본 논문에서 유사도 Ω 측정값은 XML 스키마 데이타 타입 계층도[9]를 기반으로 하고 있으며 웹 서비스 상호연결을 위해 0과 1 사이의 유사도 값을 산출한다. <표 1>은 (그림 5)와 같은 3

〈표 1〉 3가지 경우에 대한 유사도

경우	조건	Ω
1	Q{I}와 S{I}가 같은 경우	1
2	Q{I}가 S{I}의 하위 개념인 경우	1
3	Q{I}가 S{I}의 상위 개념이거나 관계가 없는 경우	α

〈표 2〉 유사도(Ω) 측정치

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

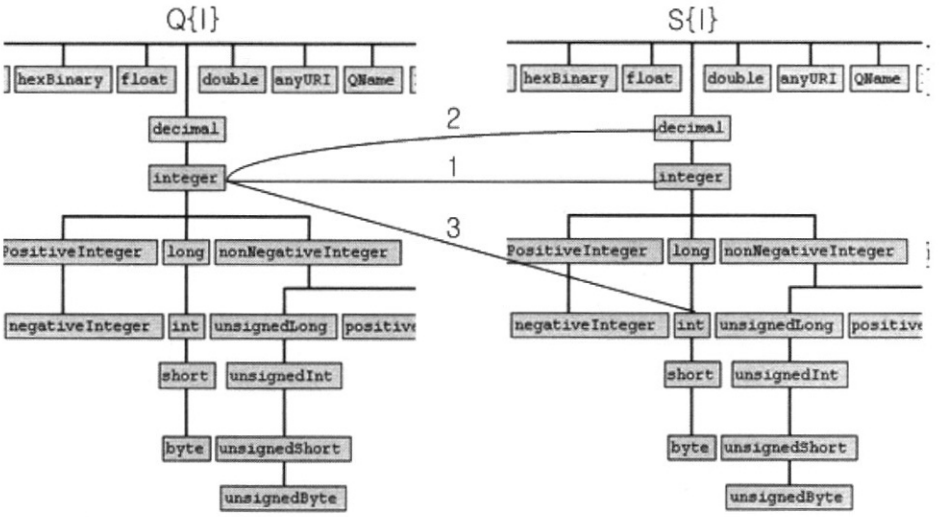
Q{I}		S{I}	Ω
long	⇨	int	2/3
string	⇨	float	0
date	⇨	time	0
date	⇨	gYear	1/3

가지 경우에 대한 유사도를 계산한 것이다.

경우 3의 α값은 <표 2>와 같이 유사도 측정치를 사전에 정의하여 검색항목이 내장 데이타 타입으로 선언된 경우 관련되는 Ω 값을 산출한다. 예를 들면, double에서 int의 Ω 값은 1/3로 정해져 있는데, 이런 경우에는 WfMS에서 데이타 타입 변환은 허락되지만 어떤 경우 데이타 손실이 발생될 수도 있기 때문에 별로 선호되지 않는 경우로써 Ω 값도 비교적 낮은 편으로 정의된다.

② 검색항목이 온톨로지 개념으로 선언된 경우

질의 템플릿 입력항목이 온톨로지 개념으로 선언된 경우에는 객체지향 모델과 같은 계층 관계에 따라 유사도가 결정된다. 예를 들면, (그림 6)의 Time 온톨로지에서 부/자 관계에 있는 Date와 CalendarDate는 유사성이 높지만 관계 설정이 없는 Date와 Time 간에는 유사성이 없다. 온톨로지 개념 간 유사성 관계는 <표 3>과 같이 5가지 경우가 발생될 수 있다.

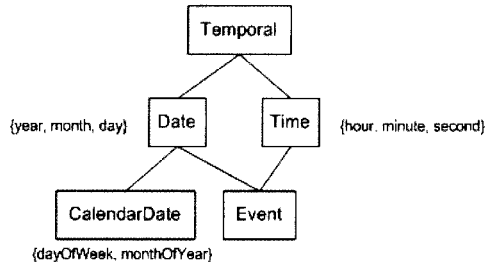


(그림 5) XML 스키마 데이타 타입 계층도

2) 본 논문에서는 편의상 입력 항목만 예시한다.

〈표 3〉 온톨로지 개념 간 유사성 관계

경우	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})



(그림 6) Time 온톨로지

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[3],

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

경우 1은 가장 간단하다. 두개의 개념이 같으면 Exact가 리턴되고 그 값은 1이다. 2의 경우, S는 Q의 상위 개념에 있으므로 S는 Q에 펄럭 인(Plug In) 된다고 할 수 있고 결과 값은 아직도 1로 평가된다. 실 예로, (그림 6)에서 Date에 관한 웹 서비스는 CalendarDate에 관한 사항도 지원될 수 있다. 3의 경우에는 Q가 S의 상위 개념에 있다. 이 경우 선택된 S는 Q의 모든 사항을 다 만족시켜주지 못한다. S가 사용될 수도 있으나 목표를 완전히 수행하기 위해서는 수정되거나 다른 대안을 찾아야 한다. 유사도는 Subsumes가 리턴되고 그 값은 개념 Q 속성의 개수 ‖p(Q)‖ 대 개념 S 속성의 개수 ‖p(S)‖ 비율로 평가된다. 4의 경우는 Q와 S 사이에 부/자 관계가 아닌 일부분 공통된 속성이 존재하는 경우이다. 이런 Intersect인 경우 유사도는 양쪽의 공통된 속성 개수를 발견하여(즉, $\partial = \| p(Q) \cap p(S) \|$) 개념 Q에만 있는 속성($\| p(Q) \| - \partial$), 개념 S에만 있는 속성($\| p(S) \| - \partial$), 그리고 양쪽 공통된 속성(∂)을 모두 더한 개수의 비율로 평가된다. 이는 Tversky 특징기반 유사도 모델[10]을 응용한 것으로써, 유사도는 두 개념 사이의 차이점 보다는 공통된 속성에 더 많은 영향을 받는다는 이론에 그 바탕을 두고 있다. 마지막으로 5는 Q와 S 사이에 전혀 공통된 속성이 없는 경우이므로 Fail이 반환되고 그 값은 0이 된다.

지금까지는 편의상 설명을 간단히 하기 위해 입력항목이 한 개인 경우만을 다루었다. 만일 입력항목이 여러 개 존재하는 경우에는 전체의 평균값을 유사도로 설정한다. 즉 전체 유사도 Ω′ 함수는 다음과 같이 표현된다.

$$\Omega' = \frac{\sum_{i=1}^{n} \Omega_i}{n}$$

예를 들면, 경우 1과 3 두개의 입력항목이 존재한다면,

$$\Omega' = \frac{1 + \dfrac{\| p(Q) \|}{\| p(S) \|}}{2}$$

로 평가된다. 또한, 입력항목과 출력항목이 모두 존재하는 경우에도 같은 방법으로 전체 평균값이 유사도로 평가된다.

3.3 워크플로우 실행계획

워크플로우 시작에서부터 점차적으로 하나씩 새로운 웹 서비스가 첨가되어 웹 서비스 워크플로우, 즉 워크플로우 실행계획(WEP : Workflow Execution Plan)이 완성되고 나면 워크플로우 엔진에 의해 이들 프로세스들이 자동으로 수행되어야 한다. WEP은 BPEL4WS[11] 스펙을 따르는 워크플로우 프로세스로써 프로세스 템플릿과 같이 액티비티와 제어 구성자의 집합으로 구성된다. 액티비티는 <receive>, <invoke>, <reply>가 있으며, 파트너(partner), WSDL 포트 타입(port type), 작업(operation), 그리고 입출력 컨테이너(container)가 기술되어 있고, 최종적으로 WSDL 파일과 바인딩하게 된다. 제어 구성자는 순차, 반복, 병행, 선택 블록을 가지고 있는데, 순차 블록에서는 액티비티들이 하나씩 순차적으로 수행되고, 반복 블록은 리스트의 각 멤버들을 반복적으로 수행한다. 그리고 병행은 액티비티가 병렬로 수행되며 선택은 여러 개 중 하나가 선택되어 수행된다. (그림 7)은 (그림 3)을 기반으로 계획 에이전트에서 작성된 WEP을 XML로 기술한 것이다. WEP을 XML로 작성한 이유는 이기종 시스템 간에 플랫폼 독립적인 데이타 교환 및 처리가 가능하게 하고, XML은 시스템 확장성 및 상호운용성을 보장하기 때문이다.

작성된 WEP을 이용하여 워크플로우 실행을 요청하면 계획 에이전트는 먼저 WEP 문서를 파싱하여 실행계획 알고리즘을 작성한다. WEP XML 문서 파싱을 위해서 본 연구에서는 W3C(World Wide Web Consortium)에서 정한 DOM(Document Object Model) 인터페이스를 사용한다.

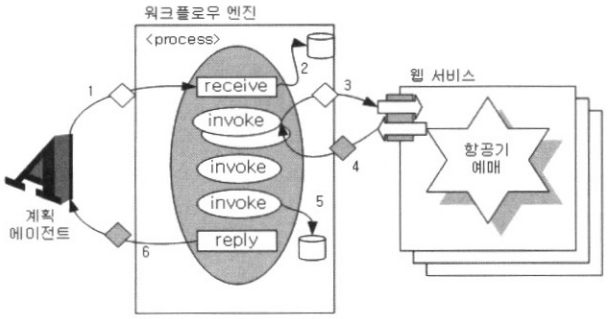
실행계획 알고리즘에 의해 워크플로우 엔진에서 웹 서비

3) $\partial = \| p(Q) \cap p(S) \|$

```
<process name="관광예약 실행계획"
         xmlns:air="http://www.daml.org/services/BrovoAir.wsdl"
         xmlns:hotel="http://www.daml.org/services/HotelInfo.wsdl"
         xmlns:tour="http://www.daml.org/services/TourGuide.wsdl"
         xmlns:pay="http://www.daml.org/services/CostPay.wsdl">
  <partners>
    <partner name="고객" serviceLinkType="air:CustomerLinkType"/>
    <partner name="항공기예매" serviceLinkType="air:ReserveLinkType"/>
    <partner name="호텔예약" serviceLinkType="hotel:ReserveLinkType"/>
    <partner name="관광예약" serviceLinkType="tour:ReserveLinkType"/>
    <partner name="비용결재" serviceLinkType="pay:PayLinkType"/>
  </partners>
  <containers>
    <container name="request" messageType="air:RequestMessage"/>
    <container name="response" messageType="air:ResponseMessage"/>
    <container name="input-1" messageType="air:InputMessage"/>
    <container name="output-1" messageType="air:OutputMessage"/>
    <container name="input-2" messageType="hotel:InputMessage"/>
    <container name="output-2" messageType="hotel:OutputMessage"/>
    <container name="input-3" messageType="tour:InputMessage"/>
    <container name="output-3" messageType="tour:OutputMessage"/>
    <container name="input-4" messageType="pay:InputMessage"/>
  </containers>

<sequence>
  <receive name="receive" partner="고객" portType="air:RequestPT"
           operation="Request" container="request">
  </receive>
  <flow>
    <invoke name="invoke-1" partner="항공기예매" portType="air:ReservePT"
            operation="Reserve" inputContainer="input-1" outputContainer="output-1">
    </invoke>
    <invoke name="invoke-2" partner="호텔예약" portType="hotel:ReservePT"
            operation="Reserve" inputContainer="input-2" outputContainer="output-2">
    </invoke>
  </flow>
  <sequence>
    <invoke name="invoke-3" partner="관광예약" portType="tour:ReservePT"
            operation="Reserve" inputContainer="input-3" outputContainer="output-3">
    </invoke>
    <invoke name="invoke-4" partner="비용결재" portType="pay:ReservePT"
            operation="Reserve" inputContainer="input-4">
    </invoke>
    <reply name="reply" partner="고객" portType="air:ResponsePT"
           operation="Response" container="response">
    </reply>
  </sequence>
</sequence>
</process>
```

(그림 7) 워크플로우 실행계획(WEP) (계속)



(그림 8) 웹 서비스 실행 내부 흐름

스가 실행되는 내부 과정을 살펴보면, 먼저 입력 매개변수를 받고(receive), 관련 웹 서비스들을 호출(invoke)하여, 에이전트에게 응답(reply)하는 3단계로 이루어진다. 예를 들면, (그림 8)에서와 같이 계획 에이전트에서 날짜와 장소 입력 매개변수를 워크플로우 엔진에게 보내면 하나의 프로세스가 생성되고 receive 엑티비티가 시작된다. 이 매개변수가 입력 컨테이너에게 넣어진 후, 곧바로 원격지에 있는 "항공기예매"와 "호텔예약" 웹 서비스가 병렬로 호출된다. 다음에 "관광예약", "비용결재" 웹 서비스가 차례로 호출된 후 그 결과가 출력 컨테이너에 넣어지고, 마지막으로 reply 엑티비티가 수행되어 계획 에이전트에게 그 결과가 보내진다.

4. 실험 및 분석

시멘틱 웹 서비스 조합 문제는 현재 연구가 활발히 진행되고 있는 분야이므로 아직까지 명확한 기능 설정이나 표준 스펙이 완료되지 않은 상황이다. 따라서 규모나 개발 범위가 다른 유사 시스템 간 성능 분석은 큰 의미가 없으므로 본 연구에서는 <표 4>와 같이 DWSC 시스템을 기존 Maryland

〈표 4〉 시스템 간 기능 비교

기능	DWSC	WS-Composer	METEOR-S
매핑 알고리즘	내장 데이터 타입과 온톨로지 개념을 분리	온톨로지 개념만 처리	같은 온톨로지 및 다른 온톨로지에 따라 구분
유사성 측정	Exact, PlugIn, Subsume, Intersect, Fail, WfMS 타입	DAML-S Matchmaker[12]	DAML-S Matchmaker[12]
실행계획	Workflow Execution Plan	없음	없음
OWL-S 파서	Maryland API 모듈 이용	자체개발	간단한 파서 개발
QoS 모델	없음	없음	시간, 가격, 신뢰도, 가용도
WfMS	자체개발	없음	자체개발
웹 서비스 조합	동적	동적	정적
웹 서비스 언어	OWL-S	OWL-S	DAML-S

대학의 WS-Composer 및 Georgia 대학의 METEOR-S 시스템과 기능 스펙을 먼저 비교하였다.

<표 4>에서 DWSC 시스템이 다른 시스템과 크게 다른 점은 위의 3가지 즉, 매핑 알고리즘, 유사성 측정, 실행계획이 될 수 있다. 이에 대한 특성을 분석하기 위해 각각에 대한 실험 분석을 수행하였다. 본 연구에서는 실험 분석을 위해 Maryland 대학의 Mindswap 연구실[8]에서 제공하고 있는 OWL-S 파일들을 이용하였다. 이 사이트에서는 약 50개 정도의 OWL-S 파일들을 제공하고 있는데 이들 중 WSDL 파일과 관련된 31개 웹 서비스를 추출하여 관련 정보를 데이터베이스에 저장하였다.

DWSC의 매핑 방식은 간단한 내장 데이터 타입을 먼저 계산함으로써 전체적인 성능 향상을 꾀할 수 있는데, 실 데이터를 분석한 결과 내장 데이터 타입과 온톨로지 개념으로 선언된 비율이 68:32로 내장 데이터 타입이 압도적으로 많았다. 따라서 이들을 먼저 분리하여 처리하는 DWSC 방식은 상당한 이점을 얻을 수 있다.

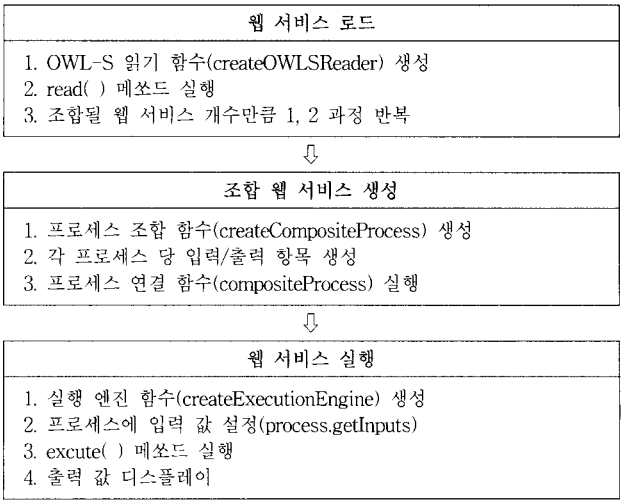
두 번째 유사성 측정을 실험하기 위해 본 장에서는 3.2에서 살펴본 5가지 경우에 대해 임의의 질의를 수행하였다. 한 예를 들면, (그림 6)의 Time 온톨로지를 이용하는 경우 <표 5>와 같은 유사도 Ω가 각각 계산된다. 입출력 항목이 여러 개 존재하는 경우에는 전체의 평균값이 유사도로 계산된다.

구축된 웹 서비스 입출력 정보는 대부분 간단한 온톨로지를 사용하고 있으므로 주로 Exact나 Fail이 많이 일어났다. 문제는 하나라도 Fail이 발생되면 이 서비스는 작동 불능이지만 DWSC 방식에서는 평균값이 유사도로 리턴된다. 예를 들면, 경우 1과 5 두개의 항목이 존재한다면 유사도는 (1+0)/2=0.5로 평가된다. 이를 해결하기 위해 HA(Human-Assisted) 기법[3]의 도입이 요구된다.

마지막으로 WEP을 기반으로 한 웹 서비스 조합을 실험하기 위해 본 연구에서는 다수(예, 2개)의 웹 서비스를 조합하여 실행하는 테스트 프로그램을 작성하였다. 먼저 2개의

〈표 5〉 Time 온톨로지를 이용한 5가지 예

경우	Q{I}		S{I}	관계	Ω
1	Date	⇨	Date	Exact	1
2	CalendarDate	⇨	Date	PlugIn	1
3	Date	⇨	CalendarDate	Subsumes	0.6
4	CalendarDate	⇨	Event	Intersect	0.375
5	Date	⇨	Time	Fail	0



(그림 9) 프로그램 처리 과정

웹 서비스를 읽어서 하나의 웹 서비스 조합을 만든다. 이를 실행 엔진으로부터 실행 시킨 후 그 결과를 화면에 디스플레이 한다. 이에 대한 자세한 프로그램 처리 과정은 (그림 9)와 같다. DWSC 시스템은 비록 아직까지 완벽한 사용자 인터페이스를 제공하는 것은 아니지만 기존의 WS-Composer나 METEOR-S 시스템보다는 진일보된 실행계획에 의한 웹 서비스 조합 자동화 프로세스를 제공하고 있다.

5. 관련 연구

웹 서비스 기술을 구성하는 주요 구성요소로는 SOAP, WSDL, UDDI가 있다. SOAP은 웹에서 구조화되고 타입이 있는 정보를 교환하는 표준 XML 프로토콜이며, WSDL은 웹 서비스 이용에 필요한 인터페이스와 입출력 메시지 형식 등을 기술하고 있으며, UDDI는 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[13]과 마이크로소프트사의 XLang[14]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[11]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[15]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 OWL에 대한 표준화가 진행되고 있다. OWL-S[5]는 OWL를 기반으로 지능적인 웹 서비스의 발견, 실행, 조합, 모니터링이 가능하도록 한 기술이다.

프로세스 템플릿을 이용한 동적 웹 서비스 조합에 관한 연구는 METEOR-S 프로젝트[16]에서 찾아볼 수 있다. METEOR-S에서는 본 논문과 비슷하게 시멘틱 프로세스 템플릿을 작성하고 실행 프로세스로 변환한 후 이를 프로세스 실행 엔진에서 수행하는 일련의 과정을 설명하고 있다. 그렇지만 이 연구에서는 온톨로지 기반 웹 서비스 탐색 및 실행 기법은 기술하고 있지 않으며 웹 서비스 조합 자동화를 위한 전반적인 워크플로우 처리 과정은 취급하고 있지 않다. 시멘틱 기반 동적 웹 서비스 조합에 관한 연구는 WS-Composer[2]에서 수행되었다. 이 시스템은 OWL-S 시멘틱 서비스 언어를 사용한 동적 웹 서비스 조합 시스템으로써, 조합을 수행하는 동안에 Matching-Filtering 알고리즘을 적용하여 목표 지향적인 대화식 조합 기법을 제안하고 있다. 따라서 사람이 개입된 반자동 시스템으로 작동되며 워크플로우 기반 e-비즈니스 업무 프로세스는 고려되지 않고 있다.

6. 결 론

본 논문에서는 비즈니스 분야에서 성공적으로 활용되고 있는 워크플로우 기법을 적용하여 동적 웹 서비스 조합을 구현하였다. 워크플로우의 웹 서비스 적용은 아직까지는 새로운 분야로써 기존 워크플로우 시스템에서는 볼 수 없었던 인터넷상의 수많은 웹 서비스들 중에서 가장 적당한 것을 찾아야하는 웹 서비스 탐색 문제의 해결을 요구한다. 본 연구에서는 이러한 탐색 문제를 해결하기 위해 시멘틱 웹 온톨로지 개념을 이용한 새로운 웹 서비스 탐색 알고리즘이 제안되었다. 그리고 어떻게 동적 웹 서비스 조합이 실행되는지 설명하기 위해 웹 서비스 워크플로우 실행 계획을 제안하였고, 계획 에이전트와 워크플로우 엔진에 의한 이의 실행 과정을 보였다. 전체적인 구현 시스템은 본 연구실에서 이미 개발되어 있는 워크플로우 기반 서비스 중개자 시스템 IMP를 확장 발전시켰다. 본 시스템은 기존의 IMP 시스템과 개념 및 구조적인 면에서는 큰 차이가 없으나 웹 서비스 등록 및 탐색 모듈이 추가 개발되었다.

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 운영되고 있는 상용 웹 서비스 검색 엔진에서의 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.

참 고 문 헌

[1] J. Cardoso, A, Sheth, "Introduction to Semantic Web Services and Web Process Composition," Semantic Web Process : powering next generation of processes with Semantics and Web Services, Lecture Notes in Computer Science, Springer, 2005.

[2] 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.

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

[4] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Sheng, "Quality Driven Web Services Composition", ACM WWW 2003, Budapest, Hungary, May, 2003.

[5] 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

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

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

[8] E. Sirin, B. Parsia, and J. Hendler, "Composition-driven Filtering and Selection of Semantic Web Service," In AAAI Spring Symposium on Semantic Web Services, 2004.

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

[10] J. Bradshaw, "Introduction to Tversky Similarity Measure," MUG'97 : 11th Annual Daylight User Group Meeting, Feb., 1997.

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

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

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

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

[15] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," Scientific American 284(5) : 34-43, 2001.

[16] K. Sivashanmugam, J. Miller, A. Sheth, and K. Verma, "Framework for Semantic Web Process Composition," Technical Report 03-008, LSDIS Lab, Dept of Computer Science, University of Georgia, June, 2003.

이 용 주

e-mail : yongju@sangju.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~현재 상주대학교 컴퓨터공학과 부교수

2004년~현재 국립상주대학교 전산정보원 원장

관심분야 : 웹 데이터베이스, 정보검색, 공간 데이터베이스