Journal of Digital Contents Society Vol. 8 No. 3 Sept. 2007(pp. 269-277)

UML과 OWL-S를 사용한 웹 서비스 진화 프레임워크

김진한*, 이창호**, 이재정***, 이병정****

요 약

최근에 웹 서비스는 비즈니스 간 e-commerce 응용을 위한 중요한 기술이 되었다. 시장적시성이 요구되는 비즈니스 관점에서, 동적 진화는 예측 불가능하고 자주 변화하는 비즈니스 요구사항에 대응할 수 있는 유연성을 제공한다. OWL-S는 OWL에 기반 한 서비스 온톨로지 언어이다. OWL에 의해 제공되는 의미성은 서비스의 발견, 수행, 조합의 자동화를 제공한다. 본 논문에서는 OWL-S를 이용하여 서비스지향 애플리케이션의 동적 진화를 지원하기 위한 프레임워크를 제안한다. 본 프레임워크에서는 표준 요구사항 분석 방법인 유즈케이스를 확장한 요구사항 개념과 활동도로의 매핑을 정의한다. 그리고 프로토타입을 구현하여 프레임워크의 유효성을 보인다.

A Framework For Web Service Evolution using UML and OWL-S

Jinhan Kim*, Changho Lee**, Jaejeong Lee***, Byungjeong Lee****

Abstract

Web service is an important technology to develop business to business e-commerce application. From a business perspective of time to market, dynamic evolution offers flexibility that software can adapt to unforeseen and fluctuating business requirements. OWL-S, based on OWL, is a service ontology language. The semantics provided by OWL support automation of service discovery, invocation, and service composition. In this paper we propose a framework to support dynamic evolution of service-oriented applications. We extend use-case analysis method to derive service description by defining requirements concept and mapping from requirement concept to activity diagram. A prototype is provided to show the validity of this framework.

Keywords : Software Evolution, Web Service Discovery and Reconfiguration, UML, OWL-S

1. 서론

최근의 비즈니스 환경은 과거 독립적인 조직 및 프로세스에 의해 주도되는 수직적 통합에서 고객, 공급자, 파트너 등 다수 기업과의 관계적 협업관계가 중시되는 수평적 통합 환경으로 변화하고 있다. 이것을 위한 기업 IT 아키텍처로 대표되는 SOC(Service-Oriented Computing)는 분산되고 상호 운용적인 비즈니스 구축을 위한 널리 퍼진 패러다임으로서 발전하였다[1,2]. SOC는 기본적으로 SOA(Service-Oriented Architecture)[3]에 의존한다. SOA에서 말하는 서비스는 비즈니스 프로세스를 수행하는 소프트웨어 단위이며 각각의 서비스는 위치투명성과 상호연동을 보장하는 메시지 프로토콜을 통해 느슨하게 결합(loosely-coupled) 한다. 전체 애플리케이션을 구성하는 각각의 서비스들은 그들이 변경되거나 또는 추가, 삭제가 되더라도 서로에게 영향을 미치지 않아야 한다. 이러한 웹 서비스의 가용성(availability)과 확장성(scalability)을 위하여 동적으로 적응하는 것이 필요하다. 이를 위하여 의미 정보를 사용하여 서비스를 검색하고[4], 검색된 서비스를 이용하여 시스템을 중지시키지 않고 재구성되어야한다[5,6]. 그러나 이전 연구에서는 의미 정보를 사용한 검색에 한계가 있었고,

※ 제일저자(First Author) : 김진한
접수일자:2007년06월29일, 심사완료:2007년07월13일
* 서울시립대학교 컴퓨터과학부
kimjinhan@uos.ac.kr
** 서울시립대학교 컴퓨터과학부
*** 서울시립대학교 컴퓨터과학부
**** 서울시립대학교 컴퓨터과학부(교신저자)
▣ 이 논문은 2006년도 서울시립대학교 학술연구조성비에 의하여 연구되었음.

동적 재구성이 특정 구현 기술에 종속되는 경향이 있다. 따라서 소프트웨어 기능에 대한 좀 더 세밀한 발견 및 랭킹 방법과 구현 기술에 독립적인 재구성 기법이 요구된다.

본 논문에서는 사용자에 의해서 새로운 요구사항이 발생했을 때 시맨틱 웹 서비스를 이용하여 필요한 서비스를 실행시간에 발견하고 재구성하는 프레임워크를 제안한다. 본 프레임워크는 실행시간 재구성 대상이 되는 소프트웨어를 서비스로 표현하고, 온톨로지[4]를 이용하여 서비스를 의미적으로 발견, 랭킹하고, 동적으로 서비스를 재구성하는 것을 지원한다.

본 논문의 구성은 다음과 같다. 먼저 2장에서 시맨틱 서비스 요구사항, 서비스 발견 기법, 소프트웨어의 동적 재구성을 위한 연구를 소개한다. 3장에서는 시맨틱 웹 서비스를 이용한 서비스 발견과 재구성을 위한 프레임워크를 설명한다. 4장에서는 프레임워크 프로토타입을 기술하고, 마지막으로 5장에서 결론을 맺는다.

2. 관련연구

2.1 서비스 요구사항

IEEE(The Institute of Electrical and Electronics Engineers)는 좋은 소프트웨어를 위한 요구사항 명세방법을 정의한다. 그 방법은 정확하고, 모호하지 않으며, 완전하고, 검증가능하며, 수정가능 하고 추적성을 제공해야한다[7]. 유즈케이스 분석 방법은 소프트웨어의 요구사항을 정의하는 표준적인 방법이다. 서비스들의 요구사항들은 의미적인 서비스 발견을 위해서 정확하게 기술되어야한다. 또한 그 요구사항들은 컴퓨터에 의해 빠르게 해석되어야만 한다. 이러한 문제들을 해결하기 위해 사용자 요구사항과 구현과의 차이를 줄이기 위한 방법이 제안되었다[8]. 그 방법은 목적을 달성하기위해서 달성하기 위한 목적들과 그들의 관계, 제약사항들을 기술한다. 자연어로 표현된 유즈케이스의 시나리오는 애매모호하며 컴퓨터가 이해하지 못한다는 단점을 가진다. 정형적인 언어들은 컴퓨터의 행동을 모호하지 않게 기술할 수 있지만 효과적으로 비즈니스 문맥을 표현할 수 없다. 이러한 문제를 해결하기 위해 유즈케이스 기술을 위한 최소한의 단어들의 집합인 ML(Minimal Language)이 제안되었다[9]. 여섯 개의 동사들을 사용하여 요구사항을 기술하는 이 방법은 쉽게 정형화된 언어로 변환된다. 그러나 그 방법은 정확하게 사용자들의 요구사항들을 기술할 수 없다는 한계를 가지고 있다. 한편 UML에 OWL-S(Web Ontology Language for Services)[10] 온톨로지를 포함하여 요구사항을 표현하는 방법도 제안되었다[11, 12]. 그렇지만 그 연구에서는 개발자가 사용자의 의도가 반영된 요구사항을 직관적으로 표현하지 못하는 문제가 존재한다.

2.2 서비스 발견

웹 서비스의 분산 컴퓨팅 모델과 시맨틱 웹의 의미 모델을 통합한 차세대 분산 서비스 환경이 시맨틱 웹 서비스이다. OWL-S는 시맨틱 웹을 위한 웹 온톨로지(ontology) 언어이다. OWL-S의 목적은 웹 서비스에 관해서 추론과 서비스 조합을 위한 계획과 에이전트들에 의한 서비스의 이용의 자동화가 가능하게 한다. OWL-S를 사용하여 서비스 제공자들은 잠재적인 사용자들에게 자신들의 서비스의 기능을 알릴 수 있다. 시맨틱 웹 서비스는 온톨로지를 활용하여 서비스를 기술하고, 온톨로지의 의미적 상호운용성을 이용해서 서비스 관련 제반 처리를 자동화 한다. 그 중 서비스 발견(discovery)은 요청(requirement)으로부터 만족하는 서비스를 찾는 과정이다. 이를 위한 프레임워크는 매치 메이킹(matchmaking), 입출력 타입 매칭(input/output types matching) 그리고 선조건과 효과분석(precondition/effect analysis)으로 구성되어 자동화된 발견 서비스를 지원 한다[5,6]. 그러나 온톨로지의 개념(concept)에 대한 의미적 매칭의 정도(degree) 와 순위(ranking)를 지원하지 못하는 단점을 가지고 있다.

2.3 동적 재구성

동적 재구성은 애플리케이션을 중지하는 일 없이 애플리케이션을 구성하는 기능 유닛을 교체할 수 있는 메커니즘이다. 동적 재구성 기능은 높은 적응성과 가용성을 위한 시스템에서 매우 유용하다.

동적 소프트웨어의 진화를 위한 연구들은 많이 진행되어왔다. 특히 전통적인 컴포넌트 기반

의 프레임워크를 제시하는 많은 연구들이 제시되었다[13,14]. 이러한 컴포넌트 프레임워크는 컴포넌트들이 서로 통신하기 위한 컴포넌트 인터페이스를 제공한다. 하지만 특정 기술에 종속적이면서 상대적으로 강한 결합(tightly coupled)을 갖는 단점이 있다. C2 스타일은 또 다른 컴포넌트 기반의 동적 아키텍처이다[15,16]. 이것은 실행시간 변화를 지원하기 위해서 연결자 (connector)의 역할(role)을 강조한다. 연결자는 컴포넌트들을 서로 바인드(bind)하고 그들 사이에서 중개자로서 활동 한다. 컴포넌트들은 연결자를 통해 비동기적인 메시지를 전달한다. 이러한 연구들은 주로 비기능적인 요구사항을 고려한 진화 방법들이 제시되어왔고, 기능적인 부분에서는 기능을 명세하기 위한 방법과 대체 자원의 부재 시, 동적 적응을 달성하지 못하는 문제들을 가지고 있다.

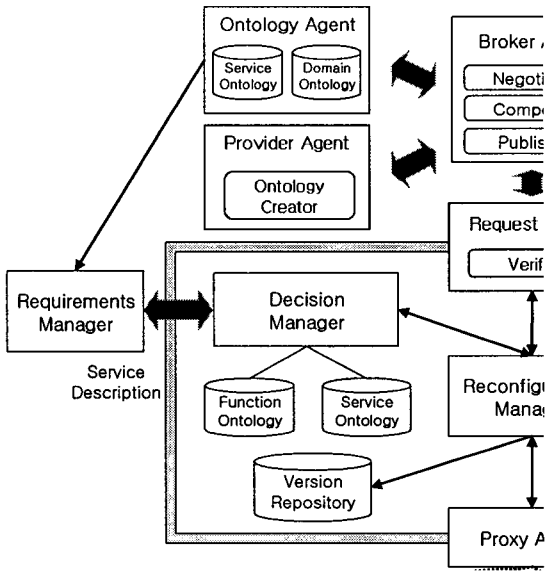
3. 서비스 진화 프레임워크

3.1 아키텍처 개요

전체 아키텍처는 (그림 1)과 같다. 크게 요구관리자(requirements manager), 런타임 재구성 엔진(run-time reconfiguration engine), 프레임워크의 기능을 지원하는 에이전트들로 구성된다. 요구관리자는 사용자로부터 새로운 요구사항이 발생했을 때, 유즈케이스를 바탕으로 작성된 요구사항으로부터 정형적인 방법으로 표시, 최종적으로는 서비스를 발견하기 위한 서비스 설명(service description)을 생성한다. 런타임 재구성 엔진은 결정관리자(decision manager)와 재구성관리자(reconfiguration manager)로 구성된다. 결정관리자는 서비스 설명을 바탕으로 추론을 통해 재구성을 위한 명령을 생성한다. 이 정보를 바탕으로 실질적으로 재구성관리자에서 재구성이 발생한다.

요청 에이전트(request agent)는 새로운 서비스에 대한 요청을 받았을 때 중개 에이전트(broker agent)로 요청을 전달하고 받은 결과를 검증한다. 서비스 설명은 시스템이 요구하는 기능을 기술하는 개념과 IOPE(Input, Output, Pre-condition, Effect)같은 서비스를 찾기 위해 필요한 정보들을 명세 한다.

중개 에이전트는 요청 에이전트와 제공자 에이전트(provider agent)의 요청을 처리한다. 요청 에이전트로부터의 요청은 협상자(negotiator)가 처리한다. 협상자는 사용자의 요구사항으로부터 온톨로지 에이전트(ontology agent)를 검색해서 후보 서비스들을 찾아서 최종 서비스를 선택한다. 이 때 선택된 서비스는 단일 또는 복합 서비스일수 있다.



(그림 1) 서비스 진화 아키텍처

합성자(composer)는 최종적으로 선택된 서비스들의 조합을 기술한다. 이 내용에는 서비스 조합에 대한 순서와 제어, 그리고 서비스 간의 메시지 변환에 대한 내용 등이 표현된다. 표현 방식은 OWL-S의 프로세스 모델을 사용한다.

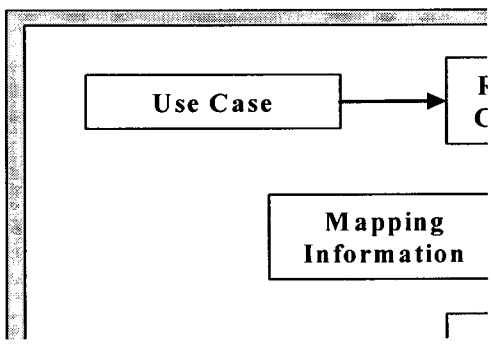
온톨로지 에이전트는 도메인 온톨로지(domain ontology)와 서비스 온톨로지(service ontology)를 저장한다. 도메인 온톨로지는 도메인에 대한 정보들을 포함하고 웹 서비스를 기술하기 위해 사용되는 개념과 용어들을 정의한다. 그리고 지식 표현을 위해 OWL을 사용한다. 서비스 온톨로지는 웹 서비스를 기술하기 위한 정보들을 OWL-S로 표현하여 저장한다. 서비스 온톨로지를 위한 기본정보들은 WSDL(Web Service Description Language) 정보를 통해 생성된다[17]. 온톨로지 에이전트로 검색 요청이 들어왔

을 때, 요청을 만족하는 개념을 찾고, 그 개념에 대해서 참조를 가지는 웹 서비스를 찾는다.

제공자 에이전트는 서비스 제공자를 위한 에이전트이다. 온톨로지 생성자(ontology creator)는 온톨로지 에이전트에서 설명한 도메인 온톨로지에 대한 개념과 서비스 온톨로지를 생성한다. 생성할 때 개념과 서비스에 대한 기본 정보들은 서비스 제공자가 제공해야 한다.

3.2 요구사항 기술

(그림 2)는 (그림 1)의 요구관리자를 상세하게 보여준다. 동적 진화를 위한 요구사항 기술을 위해서 요구사항 개념(Requirements Concept, RC)을 제안한다. RC는 시스템의 전체 흐름을 보여주는 활동도(activity diagram)의 생성을 도와주는 집합이다. RC는 하나의 유즈케이스에 대응하고, 온톨로지를 사용하여 유즈케이스로부터 유도된다. 따라서 RC는 요구사항으로부터 활동도로의 전환을 돕는다. RC의 정의는 다음과 같다(정의 1).



(그림 2) 요구관리자

정의 1. 요구사항 개념 (RC)

RC = <ID, Event, IOPE, QoS>
Event = <Num, Func, IOPE, QoS>

where
- ID: 유즈케이스의 이름.
- Num: 숫자는 순서에 따라 기술된다. 동시다발적인 흐름에서는 '2.a', '2.b'와 같은 표기법을 사용한다. 그리고 단순 분기인 경우에는 '2.1', '2.2'와 같은 방법을 사용한다.
- Func: 하나의 이벤트에 대해서 동사+목적어 형태로 표현한다.
- IOPE: 입력, 출력, 선조건, 결과를 나타낸다.
- QoS: 비기능적인 속성들 중 비용과 응답시간만 고려한다. 이 값들은 서비스 발견 결과를 걸러주는 역할을 한다.

유즈케이스의 자연어와 정형화된 기술 방법의 차이를 줄이기 위해 RC는 자연어보다 좀 더 간단한 표현방법을 사용한다. 그리고 RC에 사용된 단어의 의미를 가진 개념을 온톨로지 저장소를 검색해서 찾는다. 그래서 RC의 단어는 컴퓨터가 이해할 수 있는 의미를 가진 개념으로 대체가 된다. 그리고 그 개념에 관한 추가적인 정보도 포함시킨다. 의미가 더해진 RC로부터 활동도로의 매핑을 다음과 같이 정의 한다(정의 2).

정의 2. 요구사항 개념으로부터 활동도로 매핑

Func(event) ⇒ 활동도의 활동(activity) 이름
Input(event or rc) ⇒ 활동 입력 핀들의 집합
Output(event or rc) ⇒ 활동 출력 핀들의 집합
Precondition(event or rc) ⇒ 활동에 들어오는 감시 조건(guard condition)들의 집합
Effect(event or rc) ⇒ 활동으로부터 나가는 감시 조건(guard condition)들의 집합
QoS(event or rc) ⇒ 활동 제약사항들의 집합
Num(event) ⇒ 활동도에서의 순서

where
- event와 rc는 각각 Event와 RC의 실례들이다.

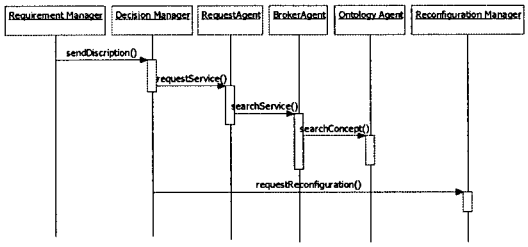
각각의 유즈케이스에 대해 매핑 규칙들을 사용하여 RC로부터 활동도로 변환되면 활동도에서 그들의 기능을 보여준다.

각각의 유즈케이스에 대해 매핑 규칙들을 사용하여 RC로부터 활동도로 변환되면 활동도에서 그들의 기능을 보여준다.

3.3 시멘틱 웹 서비스 발견 및 재구성

(그림 3)은 우리의 프레임워크에서 서비스들의 매칭과 재구성하는 과정의 순서를 보여준다.

요구관리자가 결정관리자(decision manager)에게 서비스 기술을 보내면, 결정관리자는 요청에이전트에게 받은 정보와 함께 서비스를 요청한다. 이 요청의 결과로는 하나의 서비스 혹은 여러 개의 서비스가 결합된 형태의 서비스가 돌아온다. 재구성의 완료되면 서비스 매칭과 재구성 과정은 완료된다.



(그림 3) 서비스 발견 및 재구성 프로세스

```
<profile:Profile rdf:ID="example">
<profile:serviceProduct></profile:serviceProduct>
<profile:serviceClassification></profile:serviceClassif
ication>
<profile:serviceCategory></profile:serviceCategory>
<profile:hasInput></profile:hasInput>
<profile:hasOutput></profile:hasOutput>
<profile:hasPrecondition></profile:hasPrecondition>
<profile:hasResult></profile:hasResult>
<profile:textDescription></profile: textDescription >
</profile:Profile>
```

(그림 4) 서비스 설명

(그림 4)는 서비스 설명을 보여준다. ServiceProduct는 UNSPSC(The United Nations Standard Products and Services Code)의 형식을 따른다. 그리고 serviceClassification는 NAICS(North American Industry Classification System)의 자세한 명세를 따른다. 이 두 원소들은 프로파일로부터 서비스의 OWL 온톨로지로 매핑하는 정보들을 제공한다. 또한 serviceCategory는 서비스 매칭에 있어서 좀 더 제약이 가해진 검색을 도와준다. HasInput, hasOutput, hasPrecondition과 HasResult는 찾기 원하는 서비스의 IOPE를 나타낸다. textDescription는 IOPE 각 부분에 가중치를 명세한다. 이 가중치들은 고정된 형태가 아니며 가중치가 요청하는 쪽의 주어진 환경에 대한 정확한 제약사항들을 나타내는 것이기 때문에 도메인과 목적에 따라 달라진다. 서비스는 IOPE 각각의 정보들이 온톨로지의 URI 정보를 포함하기 때문에 의미적으로 해석된다.

```
Match(request){
 matchlist = empty
 ontologyMatch(matchlist, request)
 categoryFilter(matchlist, request)
 iopeMatch(matchlist, request, request.weight)
 sortMatch(matchlist)
}
iopeMatch(matchlist, request, w) {
 for sc in matchlist.iopes {
  sc.score =  wI*degreeMatch(request.i, sc.i)
  sc.score += wO*degreeMatch(request.o, sc.o)
  sc.score += wP*degreeMatch(request.p, sc.p)
  sc.score += wE*degreeMatch(request.e, sc.e)
 }
}
degreeMatch(typeR, typeL) {
 for itemR in typeR {
  for itemL in typeL {
   if itemR equivalence itemL then
    degree += Exact
   if itemR subClassOf itemL then
    degree += Subsume
   if itemR superClassOf itemL then
    degree += Relaxed
   }
  }
}
```

(그림 5) 서비스 매칭 알고리즘

본 논문에서는 서비스와 서비스 설명사이의 매칭의 단계를 제안한 [18]의 방법을 확장한 방법을 제안한다. 매칭의 온톨로지 단계에 관하여 이 방법은 내림차순 순서로 Exact, Subsume, Relaxed and Fail의 네 가지로 표현한다.

- Exact: 서비스의 온톨로지 개념과 서비스 설명 사이의 정확한 매칭
- Subsume: 요청이 서비스의 하위 타입인 경우
- Relaxed: 요청이 서비스의 상위 타입인 경우

- Fail: 서비스의 온톨로지 개념과 서비스 설명 사이의 매칭 실패를 의미

(그림 5)는 서비스들의 순서화된 리스트를 반환하는 서비스 매칭과 랭킹에 대한 방법이 포함된 매칭 알고리즘을 보여준다. 매칭 함수는 타입들 사이에서 매칭의 단계를 반환하는 degreeMatch를 호출하는 iopeMaching를 호출한다. iopeMatch함수는 IOPE 각각의 원소들의 점수를 계산하기 위해 wI, wO, wP, wE 변수들을 사용한다.

4. 사례연구

본 논문에서는 실행시간에 동적으로 서비스 발견과 진화를 보여주는 프로토타입을 구현했다. 프로토타입은 자바(Java) 언어를 사용하여 구현하였다. 그리고 표현 계층(presentation layer)은 GlassFish 애플리케이션 서비스의 JSF(Java Server Faces)를 사용하여 구현하였다. 온톨로지를 구축하기 위해 Jena를 기반으로 하는 SOFA (Simple Ontology Framework API)를 사용하였다. 서비스들에 관련된 온톨로지 개념들은 서비스 온톨로지에 저장하였다. 이 온톨로지는 물리적으로 파일 시스템에 저장되고 매칭과 랭킹에 사용된다.

구현한 프로토타입은 온실가스 배출량을 보여주는 애플리케이션이다. 이 애플리케이션을 위한 간단한 시나리오는 다음과 같다:

- 첫째, 이 애플리케이션의 사용자는 연도를 입력하고 결과를 알고 싶은 대륙과 온실가스를 선택한다.
- 둘째, 애플리케이션은 테이블 형식으로 온실효과를 일으키는 가스의 배출량을 보여준다.

<표 1>과 <표 2>는 새로운 요구사항이 더해진 유즈케이스와 그 유즈케이스로부터 생성된 RC를 보여준다. 이 간단한 예제에서는 유즈케이스에 중점을 두었기 때문에, <표 2>에서 각각의 event에 대해서 IOPE와 QoS를 명시하지 않았다.

<표 1> 새로운 유즈케이스

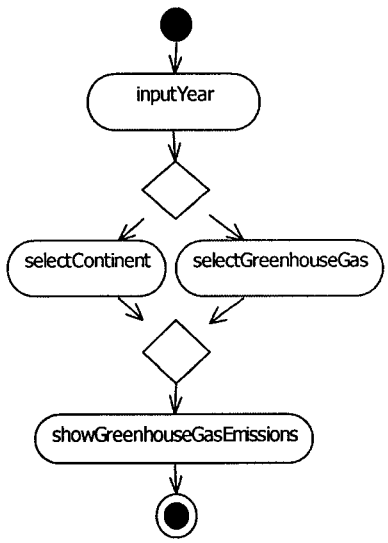
항목	내용
이름	온실가스배출
주액터	사용자
사전조건	연도, 대륙, 온실가스 배출 자료가 유효하다.
사후조건	유효한 HTML
기본흐름	1. 연도를 입력하고, 대륙과 온실가스를 선택한다. 2. 시스템은 자료를 그래프 형태로 보여준다.

<표 2> <표 1>로부터 유도된 요구사항 개념

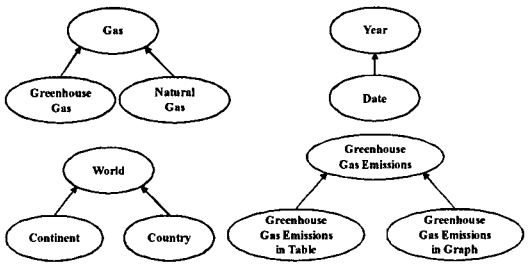
이름	온실가스배출		
사건	1	V: 입력	O: 연도
	2.1	V: 선택	O: 대륙
	2.2	V: 선택	O: 온실가스종류
	3	V: 표시	O: 온실가스배출량
IOPE	I: 연도, 세계, 온실가스종류 O: 온실가스배출량그래프 P: 유효한 연도, 대륙, 온실가스종류 E: 유효한 HTML		
QoS	비용 <=100 응답시간 <=3		

RC로부터 매핑 규칙을 가지고 (그림 6)의 활동도를 생성한다. 아직 자동화 도구가 구현이 되지 않아서 매핑 규칙에 따라 수동적으로 매핑하였다. 활동도로부터 생성된 XMI 문서를 파싱해서 (그림 4)에서 설명한 서비스 설명의 각 항목을 채운다.

(그림 7)은 온톨로지의 개념들과 그 개념들의 관계들을 보여준다. 예를 들어 'World' 개념은 'Continent'와 'Country' 개념들의 상위 타입인 관계를 보여준다. 이것은 'World' 개념이 'Continent'와 'Country' 개념들과 'Subsume' 관계에 있다는 것을 의미한다.

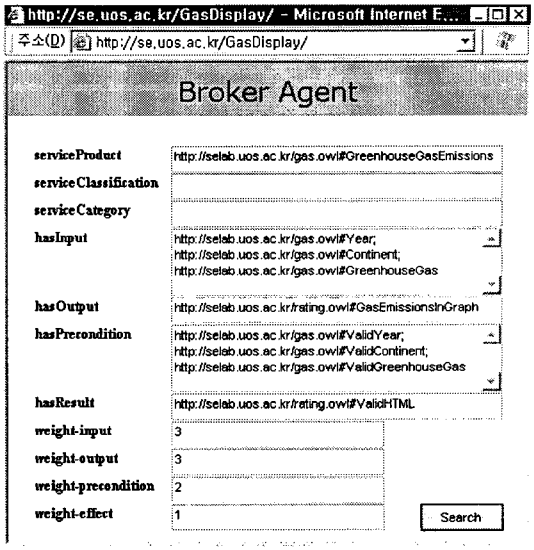


(그림 6) <표 2>로부터 유도된 활동도

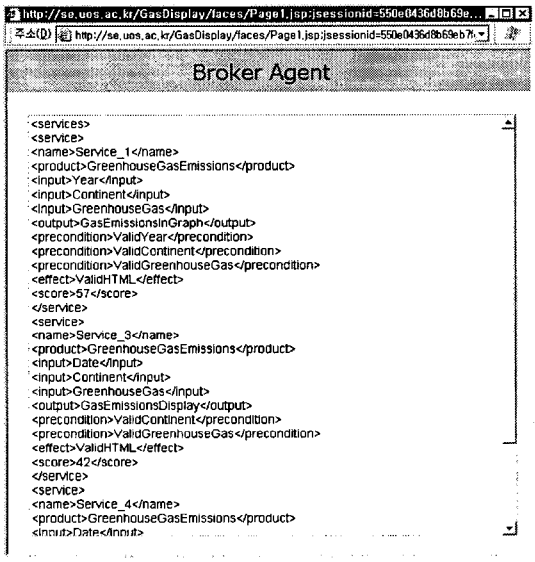


(그림 7) 온톨로지

(그림 8)은 애플리케이션의 진화를 위해 시나리오에서 중개 에이전트를 통한 서비스 요청을 보여준다. 이 서비스 요청 명세는 서비스 타입을 serviceProduct 부분에 gas 온톨로지의 GreenhouseGasEmissions 개념으로, 서비스 입력을 hasInput 부분에 Year, Continent, GreenhouseGas 개념으로, 서비스 출력을 hasOutput에 GasEmissionsInGraph 개념으로 제약한다. ValidYear, ValidContinent, ValidGreenhouseGas는 선조건을 제약한다. 그리고 효과를 위한 validHTML 제약이 명세 한다. 이 연구는 (그림 8)의 각각의 부분들에 대해서 가중치를 가지는 사용자의 요구사항에 맞는 웹 서비스 검색을 지원한다.



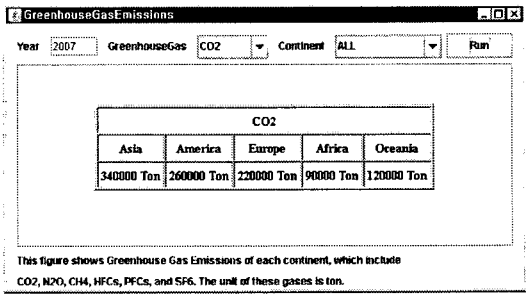
(그림 8) 서비스 발견 요청



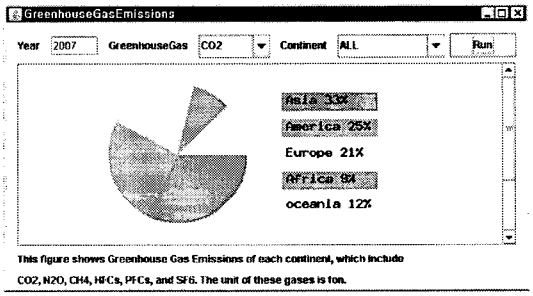
(그림 9) 매칭된 서비스 목록

(그림 9)는 (그림 8)의 명세로부터 얻어진 검색 결과를 보여준다. 이 프레임워크에서 가정하고 있는 것은 온톨로지 개념들 사이의 Exact, Subsume, Relaxed, Fail의 매칭 단계가 각각 3, 2, 1, 0으로 매핑된다는 것이다. 검색으로부터 세 가지 서비스들이 발견되었다. 첫 번째 서비스는 모든 IOPE 속성들에 대해서 가장 정확하게 매칭되는 서비스이다. 두 번째 서비스는 세 번째 서비스보다는 높은 점수를 얻은 서비스이다. 이

것은 서비스 설명의 hasOutput 원소가 두 번째 서비스에서는 Relaxed 관계를 가지고 세 번째 서비스에서는 Fail 관계를 가지기 때문이다. (그림 9)에 보이는 리스트는 각각의 원소들에 대해서 매칭 단계와 가중치의 곱셈들을 합한 총 점수를 계산하여 얻어진 것이다. 재구성 관리자(reconfiguration manager)는 실행시간에 이 서비스들 중에서 첫 번째 서비스를 선택하여 동적으로 재구성을 수행한다.



(그림 10) 재구성전: 테이블 기반 온실가스배출량



(그림 11) 재구성후: 그래프 기반 온실가스배출량

(그림 10)은 본 논문에서 제시한 실행시간에 시스템의 진화를 지원하는 프레임워크를 기반으로 구현된 온실가스 배출량을 표시하는 애플리케이션을 보여준다. (그림 10)과 (그림 11)은 각각 (그림 9)의 리스트에 있는 첫 번째 서비스를 선택함으로써 진화 전의 테이블 기반의 온실가스 배출량 애플리케이션과 진화 후의 그래프 기반의 온실가스 배출량 애플리케이션을 보여준다

5. 결 론

본 논문에서 우리는 시맨틱 웹 서비스를 사용하여 실행시간에 서비스를 발견하고 동적으로 진화하는 프레임워크를 제안하였다. 이 프레임워크는 크게 요구사항 부분과 진화 부분을 포함한다. 요구 관리자를 통해 유즈케이스로부터 서비스 설명을 생성했다. 그리고 동적 진화 엔진을 통해 웹 서비스를 찾았다. 찾은 후 프레임워크는 찾은 서비스에 바인딩하여 동적으로 서비스에 의해 애플리케이션을 재구성하였다.

이 프레임워크는 총 점수로 서비스들을 정렬했지만 점수 그 자체에는 충분한 조건을 설명하지 않고 있다. 그리고 소프트웨어 개발자가 같은 점수들을 가진 서비스들 중에 하나를 고르기 위해서는 더 많은 정보를 줘야한다는 것이다. 앞으로 이러한 부분을 보완해야 할 것이다. 그리고 요구사항 부분에 있어서도 RC를 기술하는 부분에 있어서 QoS 부분을 구체화하고 정형화된 형식의 RC를 가지고 재사용 패턴을 찾는 연구가 진행되어야 한다.

참 고 문 헌

[1] M. P. Singh and M. N. Huhns, Service-Oriented Computing, John Wiley & Sons, Ltd., 2005.

[2] OASIS, Reference Model for Service Oriented Architecture 1.0, Feb. 2006.

[3] K. Dirk, B. Karl and S. Dirk, Enterprise SOA, Prentice Hall, 2005.

[4] M. Uschold and M. Gruninger, "Ontologies: Principles, Methods and Applications," Knowledge Engineering Review, Vol. 11, No. 2, pp. 93-136, 1996.

[5] D. Fensel and C. Bussler, "The Web Service Modeling Framework," Proc. of International Semantic Web Conference, 2002.

[6] S. Chaiyakul, K. Limapichat, A. Dixit and E. Nantajeewarawat, "A Framework for Semantic Web Service Discovery and Planning," Proc. of IEEE Conference on Cybernetics and Intelligent Systems, pp. 1-5, June 2006.

[7] IEEE Recommended Practice for Software Requirements Specifications, IEEE Standard 830-1998, 25 June 1998.

[8] M. Lin, H. Guo and J. Yin, "Goal Description Language for Semantic Web Service Automatic Composition," Proc. of the Symposium on Applications and

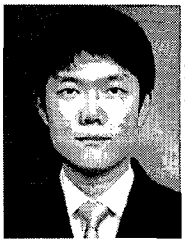
the Internet, 2005.
[9] T. Osaki, A. Kobayashi and T. Kato, "Writing Use-Case with a Minimal Set of Words," Proc. of the 5th IEEE/ACIS International Conference on Computer and Information Science, 2006.
[10] W3C, OWL-S: Semantic Markup for Web Services, Nov. 2004.
[11] J. Timm and G. Gannod "A Model-Driven Approach for Specifying Semantic Web Services," Proc. of the IEEE International Conference on Web Services, 2005.
[12] Groenmo, R. and Jaeger, M.C., "Model-Driven Semantic Web Service Composition," Proc. of the 12th Asia-Pacific Software Engineering Conference, 2005.
[13] Y. Qun, Y. Xan-Chun and X. Man-Wu, "A Framework for Dynamic Software Architecture based Self-healing," ACM SIGSOFT Software Engineering Notes, Vol. 30, July 2005.
[14] J. Malek, M. Laroussi and A. Derycke, "A Middleware for Adapting Context to Mobile and Collaborative Learning," Proc. of Annual IEEE International Conference on Pervasive Computing and Communications Workshops, pp. 221-225, Mar. 2006.
[15] P. Oreizy and Richard N. Taylor, "On the Role of Software Architectures in Runtime System Reconfiguration," Proc. of the International Conference on Configurable Distributed Systems (ICCDS4), May 1998.
[16] P. Oreizy, N. Medvidovic, and Richard N. Taylor, "Architecture-Based Runtime Software Evolution," IEEE/ACM International Conference on Software Engineering (ICSE '98), pp. 177-186, Apr. 1998.
[17] S. Naveen, P. Massimo and S. Katia, "Semantic Web Service Discovery in the OWL-S IDE," Proc. of International Conference on System Sciences, 2006.
[18] R. Gue, J. Le and X. Xia, "Capability Matching of Web Services Based on OWL-S", Proc. Of the 16th International Workshop on Database and Expert Systems Applications, 2005.

김 진 한

2006년 : 서울시립대학교 졸업(학사)
2006년~현 재 : 서울시립대학교 재학(석사과정)
관심분야 : 소프트웨어진화, 시맨틱 웹 서비스

이 창 호

2006년 : 서울시립대학교 졸업(학사)
2006년~현 재 : 서울시립대학교 재학(석사과정)
관심분야 : 소프트웨어진화, 시맨틱 웹 서비스



이 재 정

2007년 : 동국대학교 졸업(학사)
2007년~현 재 : 서울시립대학교 재학(석사과정)
관심분야 : 소프트웨어진화, 시맨틱 웹 서비스

이 병 정

1990년 : 서울대학교 계산통계학과(이학사)
1998년 : 서울대학교 전산과학과(이학석사)
2002년 : 서울대학교 컴퓨터공학부(공학박사)
2002년~현 재 : 서울시립대학교 컴퓨터과학부 교수
관심분야 : 소프트웨어 방법론, 품질 평가, 소프트웨어공학