시맨틱 웹을 이용한 UML 기반의 웹 서비스 애플리케이션의 진화*

이창호°, 김진한, 이재정, 이병정
서울시립대 컴퓨터과학부
{leechangho°, kimjinhan, jaejeong, bjlee}@uos.ac.kr

UML based Evolution of Web Service Applications using Semantic Web

Changho Lee°, Jinhan Kim, Jaejeong Lee and Byungjeong Lee
School of Computer Science, University of Seoul

요 약

시장 적시성과 서비스의 품질을 만족하기 위해 웹 서비스는 서비스를 재사용하여 애플리케이션을 구성한다. 그리고 동적 진화(evolution)는 소프트웨어에 유연성을 제공하고, 계속 변경되어 예측하기 어려운 비즈니스 요구사항에 애플리케이션이 적응할 수 있도록 해준다. OWL 기반의 OWL-S(Web Ontology Language for Services)는 서비스를 기술하기 위한 온톨로지 언어이다. OWL-S의 의미적인 정보는 자동화된 시맨틱 웹 서비스의 발견과 호출, 서비스 조합을 위해 사용될 수 있다. 본 논문에서는 이러한 시맨틱 웹 서비스를 이용한 웹 애플리케이션의 동적 진화를 지원하는 프레임워크를 제안한다. 그리고 프로토타입을 통해 제안한 프레임워크의 타당성을 보인다.

1. 서 론

웹 서비스는 프로그래밍 언어, 프로토콜 또는 플랫폼에 독립적인 장점을 가져 널리 확산되었다. 웹 서비스 애플리케이션의 진화를 위해서는 서비스 요청자가 그들의 요구사항을 만족하는 서비스 제공자들을 자동적으로 찾고, 시스템이 중단되지 않고 새로운 서비스로 적응하는 것이 중요하다. 그렇지만 웹 서비스는 서비스 요청과 발견 시 의미 정보의 부족으로 원하는 서비스를 찾기가 어렵다. 반면에 시맨틱 웹 서비스에서는 의미정보를 사용하여 서비스가 가지는 능력들의 자동화된 매칭(matching), 서비스들의 순서화(rankng), 서비스 발견(discovery)을 도와준다 [1, 2]. 그러나 서비스를 찾고 순서화할 때, 이전 연구들은 서비스 온톨로지(ontology) 개념(concept)의 정확한 매칭을 통한 서비스 검색만을 지원하고, 타입 매칭 정도(matching degree)를 지원하지 않는다.

시스템의 규모가 커지고 환경 변화의 속도가 빨라짐에 따라 동적 진화의 필요성은 점점 커져가고 있다. 하지만 동적 진화에 관한 연구들은 구현 기술에 의존하는 경향을 보인다. 이러한 경향은 구현 기술에 변화가 일어나면 문제가 발생한다. 따라서 동적인 진화 기술은 자주 변하는 사용자의 요구사항뿐만 아니라 환경과 시스템의 변화에 대처하고, 구현기술에 독립적으로 진화할 수 있어야 한다.

본 논문에서는 웹 서비스로 구성된 애플리케이션이 새로운 서비스를 필요로 할 때 시맨틱 웹 서비스 기술을 사용하여 실행시간에 동적으로 진화하는 프레임워크를 제안한다. 또한 이 프레임워크에서는 웹 서비스의 특징들을 고려하여 유즈케이스 모델로부터 서비스 설명(service description)을 유도하기 위한 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 서비스 발견을 위한 요구사항에 대한 연구들과 서비스를 동적으로 재구성하는 연구들을 소개한다. 3장에서는 동적 진화 프레임워크와 새로운 서비스 요구사항 유도 방법을 제안한다. 그리고 4장에서는 동적 진화를 설명하기 위한 프로토타입을 보여주고, 마지막으로 5장에서는 결론과 향후 연구를 기술한다.

2. 관련 연구

OMG에서 제안한 UML의 유즈케이스 모델은 사용자의 요구사항을 나타내기 위해서 자주 사용되어왔다. 또한 이런 유즈케이스 모델을 분석하여 리팩토링하고 재구성하는 연구들도 진행되었다 [3, 4, 5]. 이 연구들은 유즈케이스 모델을 분해하고, 분해된 것들을 조작하기 위한 방법이나 가이드라인을 정의하고 사용한다. 그러나 이러한 방법들은 유즈케이스로부터 분해한 것들을 서비스로 식별하지 않는다. 또한 이 연구들은 요구사항을 분석함에 있어서 서비스라는

*본 연구는 한국과학재단 특정기초연구(R01-2006-000-11150-0)지원으로 수행되었음.

개념을 고려하지 않기 때문에 웹 서비스 특징들을 고려하여 기술하기에는 적당하지 않다.

서비스 발견은 사용자의 요구사항을 만족하는 서비스들을 찾는 과정이다. 시맨틱 웹에서 서비스를 찾기 위한 과정은 사용자의 요구사항을 만족하는 서비스를 찾기 위한 입/출력 타입에 대한 매칭, 그리고 선조건과 결과에 대한 분석의 단계 등을 포함한다 [6, 7]. 이러한 과정에 대해서 여러 연구들이 진행이 되었지만 온톨로지를 사용하여 매칭의 정도를 지원하지 못한다는 단점이 있다. [8]에서는 온톨로지 개념을 사용하여 매칭의 정도를 지원하는 방법을 제안하고 있지만, 그 방법에는 비기능적인 요소를 고려하지 않는다.

동적 재구성은 이용가치가 높고 규모가 큰 시스템에 있어서 필요한 기능이다. 이러한 시스템에 동적 재구성을 제공하기 위한 많은 연구들이 제안되었다. 그 중에는 기존의 컴포넌트 기반의 개발 방법을 기반으로 하는 프레임워크를 제안하는 연구들도 있었다 [9, 10]. 이 컴포넌트 기반의 프레임워크들은 컴포넌트들이 서로서로 통신을 위한 인터페이스를 제공한다. 그러나 그 프레임워크들의 단점은 구현 기술에 강하게 종속되어 있다는 것이다. 또한 그들은 기능적인 요구사항들 보다는 비 기능적인 요구사항들에 중점을 두는 경향을 보인다.

3. 서비스 진화 프레임워크

동적으로 이루어지는 진화는 시스템에 의해 자동적으로 진행이 되는 진화와 개발자에 의한 진화가 이루어질 수 있다. 두 가지 웹 서비스를 이용하기 때문에 실행 시간에 시스템의 중단 없이 진화를 달성할 수 있다. 그 중 본 논문에서는 개발자에 의해 이루어지는 진화에 중점을 두어 기술한다. 그림 1은 서비스 진화 프레임워크를 보여준다. 이 프레임워크는 서비스 식별 계층(Service Identification Layer, SI), 서비스 진화 계층(Service Evolution Layer, SE) 그리고 서비스 발견 계층(Service Discovery Layer, SD)로 구성된다.

- SI 계층은 유즈케이스 모델로부터 서비스 유도 과정을 통해 서비스 설명을 생성하고 이 정보를 결정 관리자로 보낸다.
- SE 계층은 서비스 설명을 사용하여 서비스 요청자를 통해 웹 서비스들을 요청하고, 재구성 관리자에서 발견한 서비스들을 이용하여 애플리케이션을 동적으로 재구성한다.
- SD 계층에서는 웹 서비스를 등록하고 검색하는 작업이 일어난다. 중개자는 서비스 요청자와 서비스 제공자로부터의 요청들을 조정하는 역할을 한다. 온톨로지 저장소에 저장되는 서비스들을 기술하기 위해 OWL-S 를 사용한다.

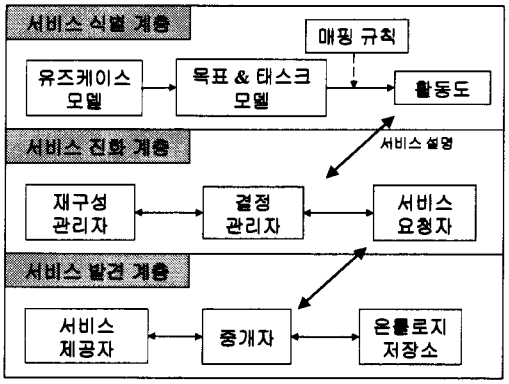

그림 1. 서비스 진화 프레임워크

3.1 서비스 식별과 활동도로의 매핑

본 논문에서는 사용자의 요구사항을 나타내는 유즈케이스 모델과 활동들로 서비스를 표현하는 활동도 사이의 간격을 줄이기 위한 새로운 서비스 설명 방법을 제안한다. 이 방법은 온톨로지 저장소에 저장된 웹 서비스들의 검색을 가능하게 하기 위해 IOPE (input, output, precondition and effect) 정보를 사용한다. 그리고 비 기능적인 요구사항을 위해서는 QoS 정보를 포함한다. QoS 정보는 발견된 서비스를 걸러내기 위해 중개자에 의해 사용된다. 사용자의 의도를 고려하기 위해 유즈케이스 모델을 사용하며, 요구사항을 분해하여 목표와 태스크 모델로 재구성한다. 목표와 태스크 모델에 대한 정의는 다음과 같다.

정의 1. 목표 & 태스크 모델 (Goal & Task Model, GTM)

Goal=<ID, Task, IOPE, QoS>
Task = <Func, IOPE, S-name, S-num, QoS>

- ID: 유즈케이스의 이름
- Func: 태스크를 식별해주며 동사와 목적어 형태를 가진다.
- IOPE: 각가 목표과 태스크의 입/출력, 선조건 그리고 결과를 나타낸다.
- QoS: 비 기능적인 특성들 중 비용(cost)과 응답시간(response time)만 고려한다.
- S-name: 서비스 식별을 위해 필요한 서비스 이름.
- S-num: 순서에 따라 할당한다. 동시다발적인 경우에는 '2.a', '2.b', '2.c' 그리고 조건 분기인 경우에는 '2.1', '2.2', '2.3' 와 같은 표기를 사용한다.

태스크는 유즈케이스 모델의 흐름에서 더 이상 분해될 수 없는 최소한의 단위를 말한다. 목표는 유즈케이스로 식별될 수 있는 전체 태스크들의 집합이다. 서비스는 하나 이상의 태스크들로 구성된다. 목표와 태스크의 IOPE는 전체와 부분 집합의 관계를 가진다. QoS는 웹 서비스의 비 기능적인 속성이다. 본 논문에서 비용과 응답시간을 다루는 이유는 웹 서비스 사용에 대한 비용을 지불해야 하며, 인터넷의 특성상 서비스의 응답 시간은 중요한 요소이기 때문이다. 태스크 각각의 QoS의 값의 합은 목표의 QoS의 값을 초과해서는 안 된다. S-name은 서비스를 식별하기 위해 사용되지만, 개발자의 경험과 지식에 많이 의존한다. 목표와 태스크의 각각의 항목들을 유즈케이스의 정보들로 채운 후에, 태스크들로부터 서비스를 유도하는 작업을 해야 한다. 이 작업은 유즈케이스 리팩토링과는 다른 점을 가진다. 서비스 유도를 위한 기본 가이드라인은 다음과 같다.

- 전체 대응: 전체 태스크들이 하나의 서비스로 식별되는 경우이다. 전체 대응은 전체로써 하나의 특별한 기능을 가지며, 부분적으로는 의미가 없는 경우에 사용된다.
- 부분 대응: 목표가 여러 태스크의 집합들로 이루어져있고, 각각의 집합은 특별한 기능을 가지며 서비스로 식별된다. 이 경우 서비스들간의 순서와 관계는 기술되어야만 한다.
- 재사용: 이전에 서비스로 식별되었던 태스크들이 있을 경우 사용한다.

세가지 경우로 서비스를 식별하고 남은 태스크들이 서비스로 식별되지 못하는 경우에는 개발자의 판단에 의해 태스크의 누락이 없게 한다. GTM 의 S-name 과 S-num 은 서비스의 식별이 이루어진 후에 채워진다. 서비스 식별 과정은 서비스 검색 결과에 의해 피드백을 받아 반복적으로 이뤄져야 한다. GTM 을 활동도로 변환하기 위해 매핑 규칙은 다음과 같다.

정의 2. 매핑 규칙

- S-name(태스크) → 활동의 이름
- Input(태스크 or 목표) → 활동의 입력 핀들의 집합
- Output(태스크 or 목표) → 활동의 출력 핀들의 집합
- Precondition(태스크 or 목표) → 활동으로 들어오는 감시조건(guard condition)들의 집합
- Effect(태스크 or 목표) → 활동으로부터 나가는 감시조건들의 집합
- S-num(태스크) → 활동도에서 순서
- QoS(태스크 or 목표) → 태스크의 QoS 는 활동으로부터 나가는 감시조건의 집합에 기술한다. 그리고 목표의 QoS 는 마지막 노드로 들어가는 감시조건에 기술한다.

S-name이 활동의 이름으로 매핑이 될 때, 같은 S-name을 가진 태스크들은 하나의 활동으로 매핑이 된다. 제어 흐름(Control flow)은 활동들이 생성된 후에 S-num에 따라 더해진다. QoS는 OCL (object constraint language) 형태를 사용하여 기술한다. 태스크 각각의 QoS 정보는 활동으로부터 나가는 감시조건에 기술한다. 목표의 QoS 정보는 마지막 노드의 들어오는 감시조건에 기술한다.

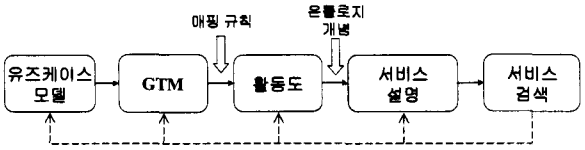

그림 2. 서비스 식별 프로세스

그림 2는 서비스를 식별하는 전체 과정을 보여주는 그림이다. 활동도는 GTM이 만들어진 후에 매핑 규칙을 사용하여 목적 시스템의 구성과 흐름을 보여주기 위해 생성한다. 이 활동도의 정보는 서비스 설명의 정보로 추출이 되는데, 이 과정에서 시맨틱 웹 서비스의 발견을 위해 온톨로지의 개념으로 대체하여 서비스 설명을 채우게 된다. 그러므로 서비스 설명은 모든 속성들에 들어가는 내용이 활동도로부터 추출하여 온톨로지 개념으로 대체하여 유도한 후에 완성된다. 또한 서비스를 식별하는 프로세스는 서비스의 검색을 통해 피드백을 받아 서비스 식별의 정제화를 달성한다.

3.2 활동도의 구성

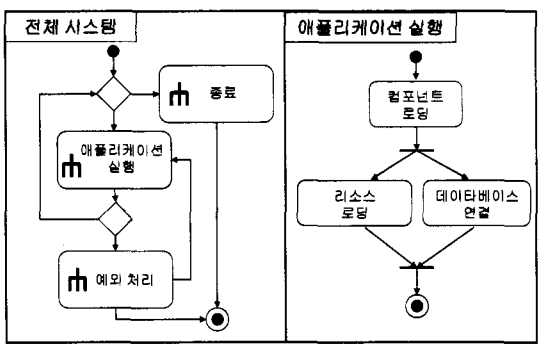

그림 3. 활동도의 예

활동도는 전체 시스템의 흐름을 기술하기 위한 널리 쓰이는 방법이다. 이 활동도는 유즈케이스 모델로 표현

할 수 없는 컨트롤과 데이터의 흐름을 기술할 수 있다. 이런 정보의 일부는 GTM으로도 표현할 수 있지만, GTM은 전체 시스템을 직관적으로 표현하고 이해할 수 있는 표현 방법은 아니다. 따라서 활동도가 시스템을 정확하게 표현하고 직관적인 이해를 도와주기 때문에 시스템이 가지는 기능들 사이의 관계를 보여주기 위해 사용한다. 그림 3의 왼쪽 편은 중첩된 활동으로 시스템의 구성을 보여준다. GTM으로부터 생성된 활동도는 그림 3의 오른쪽과 같이 시스템의 한 부분이 가지는 기능을 보여준다. 만약 활동도가 새로운 요구사항으로부터 유도되었다면, 새로운 활동이 전체 시스템의 활동도에 추가된다. 전체와 부분으로 활동도를 구성함으로써, 활동도의 복잡도를 조절하고, 새로운 요구사항으로부터 더해지거나 수정으로부터 오는 변화를 줄일 수 있다.

3.3 시맨틱 웹 서비스의 발견과 재구성

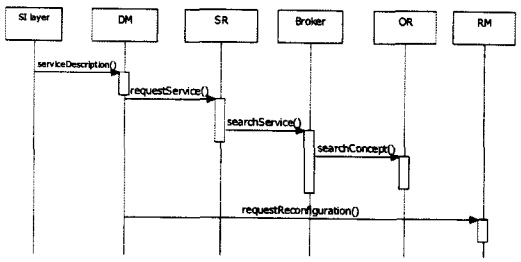

그림 4. 서비스 발견과 재구성

그림 4 는 이 프레임워크에서 서비스 발견과 재구성 과정의 순서를 보여준다. 사용자의 요구사항으로부터 생성된 서비스 설명을 SI 계층으로부터 결정 관리자에 전달할 때, 결정 관리자는 그 정보를 가지고 서비스 요청자에 서비스를 요청한다. 서비스 요청자는 요청을 중개자로 넘겨주고, 중개자는 온톨로지 저장소로 보내 서비스를 검색하게 한다. 온톨로지 저장소는 후보 서비스들을 찾아 결과를 중개자로 보낸다. 중개자는 후보 서비스들을 가지고 적합한 서비스들을 찾아 다시 서비스 요청자를 통해 결정 관리자로 전달한다. 받은 결과를 가지고 그 중 가장 적합한 서비스를 가지고 결정 관리자는 재구성 관리자에게 애플리케이션의 재구성을 요청한다. 재구성 후에 사용자의 요구사항을 만족하게 되면 이 과정은 완료되고 그렇지 않다면 다른 서비스를 가지고 재구성 작업이 재차 이뤄지게 된다.

그림 5 는 서비스 설명을 보여준다. 서비스 설명의 기본 형태는 OWL-S 로 기술된 서비스를 검색하기 위해 서비스의 프로파일이 포함하는 내용으로 구성된다. 또한 서비스 설명에 들어가는 내용은 GTM 으로부터 매핑된 활동도로부터 추출된다. 서비스 설명의 ID 는 활동의 이름으로 채워지고 찾고자 하는 서비스나 서비스가 가지는 기능을 나타낸다. serviceCategory 는 서비스가 속하는 도메인의 정보를 가지며 제한된 서비스 검색을 도와준다. hasInput, hasOutput, hasPrecondition, hasResult 는 활동의 입/출력 핀들과 감시조건들로부터 채워지며, 각각 검색하고자 하는 서비스의 IOPE 를 나타낸다. QoS 는 마지막 노드의 감시조건으로부터 채워진다. textDescription 은 IOPE 의 가중치를 정보를 포함한다. 가중치 값은 고정되지 않고 도메인과 목적에 따라 다른 값을 가진다. 그 이유는 가중치는 사용자 환경의 제약사항들을 나타낼 수 있기 때문이다. 서비스 설명은 활동도의 활동 수만큼 만들어지고, 서비스 요청은 서비스 설명의 수만큼 발생한다.

```
<serviceDescription ID="service_name">
    <serviceCategory></serviceCategory>
    <hasInput></hasInput>
    <hasOutput></hasOutput>
    <hasPrecondition></hasPrecondition>
    <hasResult></hasResult>
    <QoS></QoS>
    <textDescription></textDescription >
</serviceDescription>
```

그림 5. 서비스 설명

```
Match(request){
    matchlist = empty
    categoryFilter(matchlist, request)
    iopeMatch(matchlist, request)
    QoSfilter(matchlist, request)
}
```

그림 6. 서비스 매칭 알고리즘

그림 6은 매칭의 정도를 지원하는 알고리즘을 보여준다. Match 함수는 iopeMatch 함수를 호출하는데, 이 함수는 내부의 degreeMatch 함수를 사용하여 서비스의 IOPE의 타입들간의 매칭 정도를 계산한다. 매칭의 정도는 Exact, Subsume, Relaxed, Fail 네 가지로 분류한다. 이 때 서비스들의 순서화를 위해 요청 정보로부터 넘어온 가중치 값을 사용한다 [11]. QoSfilter 함수는 iopeMatch 함수로부터 반환된 서비스들의 결과를 가지고 요청 정보의 QoS를 만족하지 않는 서비스들을 걸러낸다. 이 알고리즘은 보통 서비스 사이에서 가장 높은 점수로 계산된 하나의 서비스를 선택한다. 그렇지만 서비스 발견 과정 중, 중개자에 의해 넘어온 결과가 적절한 서비스들이 아닐 수도 있기 때문에 소프트웨어 개발자가 개입할 수 있다.

4. 사례 연구

ID	GreenhouseGasEmissions						
	Func		S-name	S-num	IOPE	Cost	R-time
	Verb	Object					
Task	LogIn		LogIn	1	I: ID, password O: GreenhouseGasEmissionsPage P: validID, validPassword E: :Logged	30	5
	Select	Year,Continent,GreenhouseGas	showGreenhouseGasEmissions	2	I: year, continent, greenhousegas O: year, continent, greenhousegas P: validYear, validContinent, validGreenhouseGas E:	20	5
	Process	GreenhouseGasEmissions	showGreenhouseGasEmissions	2	I: year, continent, greenhousegas O: GreenhouseGasEmissionsInGraphPage P: E: validHTML	100	10
IOPE	I: ID, password, year, continent, greenhousegas O: GreenhouseGasEmissionsPage, year, continent, greenhousegas, GreenhouseGasEmissionsInGraphPage P: validID, validPassword, validYear, validContinent, validGreenhouseGas E: :Logged, validHTML						
QoS	Cost <= 150 Response time <= 20						

그림 8. 그림 7로부터 유도된 GTM

본 논문에서 제안하는 동적으로 서비스 발견과 진화를 보이기 위해 자바 언어를 사용하여 프로토타입을 구현하였다. 프로토타입의 프레젠테이션 계층은 JSF (Java Server Faces)를 사용하여 구현하였다. 온톨로지를 구성하기 위해 Jena를 기반으로 하는 SOFA (Simple Ontology Framework API)를 사용하였다. 프로토타입은 온실효과 가스의 배출량을 보여주는 애플리케이션이고, 이 애플리케이션에서 나타내고자 하는 시나리오는 다음과 같다.

- 연도를 입력하고 대륙과 온실효과 가스에 대한 정보를 선택한다. 이 애플리케이션은 테이블 형식으로 온실효과 가스의 배출량을 보여준다.

Name	GreenhouseGasEmissions
Main Actor	Users
Pre-condition	A year, continent and greenhouse gas must valid
Post-condition	The output must be a valid HTML
Successful scenario	1. Log-in. 2. Input a year and select a continent or greenhouse gas. 3. The application processes greenhouse gas emission and displays the result in graph format.

그림 7. 변경된 유즈케이스

그림 7과 8은 변경된 유즈케이스와 그 유즈케이스로부터 생성된 GTM을 보여준다. 기본 시나리오는 온실가스 배출량을 테이블 형태로 보이도록 기술하고 있으나, 그림 7의 변경된 유즈케이스는 그래프 형태로 보이도록 기술하고 있다. 그림 8에서는 태스크들을 두 개의 서비스로 식별하였다. LogIn 태스크는 서비스 식별 가이드라인 중 재사용을 적용하여 LogIn 서비스로 식별하였다. 그리고 SelectYearContinentGreenhouseGas와 ProcessGreenhouseGasEmissions 태스크는 부분 대응에 의해 showGreenhouseGasEmissions 서비스로 식별하였다.

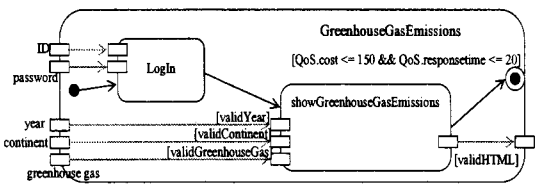

그림 9. GTM으로부터 생성된 활동도

그림 9는 GTM으로부터 생성된 활동도의 모습을 보여준다. 활동도는 매핑 규칙을 통해 생성이 되며, 이 활동도로부터 서비스 설명의 정보를 추출한다. 서비스 설명들은 GTM에서 식별한 서비스들의 수만큼 생성된다.

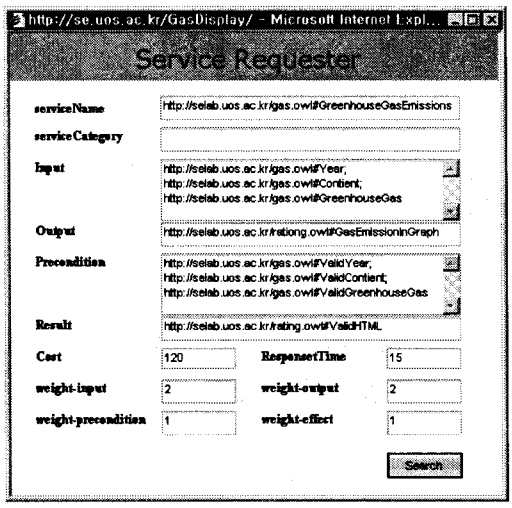

그림 10. 서비스 검색을 위한 요청

그림 10은 서비스 설명의 정보를 가지고 서비스 요청자를 통해 웹서비스를 요청하는 화면이다. 서비스 요청자에서 중개자로 서비스를 요청을 하면, 중개자는 온톨로지 저장소로부터 타입 매칭을 통해 후보 서비스들을 검색한다. 결과를 순서화 하기 위해 프로토타입에서는 온톨로지 개념들 사이에서의 매칭의 정도를 각각 3, 2, 1, 0으로 매핑하는 것을 가정한다. 이 숫자들과 가중치의 곱으로 계산된 점수에 의해 중개자는 높은 점수 순서로 정렬하여 서비스 요청자로 결과를 보내게 된다. 이 리스트에 있는 서비스들 중 최상위에 있는 서비스가 일반적으로 사용자의 요청 정보에 적합한 서비스가 된다.

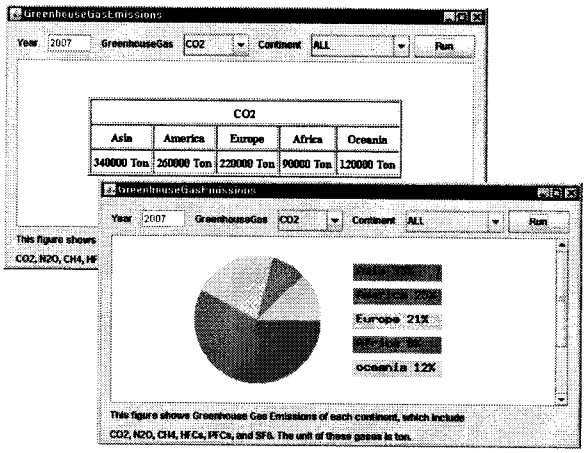

그림 11. 진화 전/후의 애플리케이션

그림 11은 반환된 리스트에서 가장 적합한 서비스를 선택하여 온실효과 가스의 배출량을 보여주는 애플리케이션의 진화 전/후의 모습을 보여준다. 그림 11의 위 그림은 진화 전 테이블 형태의 가스 배출량을 보여주고 있고, 그림 11의 아래 그림은 진화 후의 그래프 기반의 애플리케이션을 보여준다.

5. 결론 및 향후 연구

본 논문에서는 GTM을 사용하여 요구사항을 기술하며 OWL-S로 웹 서비스를 기술하여 실행시간에 동적으로 서비스들을 찾아 진화하는 프레임워크를 제안하고, 그에 따르는 프로토타입 애플리케이션을 구현하였다. 이 프레임워크는 SI, SE와 SD의 세 계층으로 구성된다. SI 계층은 유즈케이스 모델로부터 목표와 태스크들을 유도하여 서비스 설명들을 생성한다. 중개자는 온톨로지의 개념들을 사용하여 요청 정보와 서비스들을 타입 매칭하고 결과를 정렬한다. SE 계층은 SD 계층으로부터 결과를 받아서 동적으로 애플리케이션을 재구성한다.

향후 연구로는 부분적으로 매칭되는 서비스 검색에 대한 연구가 진행되어야 할 것이다. 또한 검색된 서비스들을 조합하여 실행할 방법에 대해서도 연구가 이뤄져야 한다. 마지막으로 신뢰할 수 있고 효과적인 진화를 위해 재사용 가능한 서비스 패턴에 대한 연구가 이뤄져야 할 것이다.

6. 참고 문헌

[1] R. Groenmo and M. Jaeger, "Model-Driven Semantic Web Service Composition," *Proc. of the 12th Asia-Pacific Software Engineering Conference*, 2005.

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

[3] W. Yu, J. Li, G. Butler, "Refactoring Use Case Models on Episodes," *Proc of 19th IEEE International Conference on Automated Software Engineering*, 2004.

[4] K. Rui and G. Butler, "Refactoring use case models: the metamodel," *Proc. of the 26th Australasian computer science conference*, Vol. 16, pp 301-308, 2003.

[5] M. Moon, K. Yeom and H. Chae, "An Approach to Developing Domain Requirements as a Core Asset Based on Commonality and Variability Analysis in a Product Line," *IEEE Transactions on Software Engineering*, Vol. 31, Iss. 7, pp. 551-569, July, 2005.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] J. Kim, J. Lee and B. Lee, "Runtime Service Discovery and Reconfiguration using OWL-S based Semantic Web Service," *Proc. of IEEE 7th International Conference on Computer and Information Technology, to be published.* 2007.