韓國컴퓨터情報學會 論文誌
第14卷 第4號, 2009. 4.

2009-14-4-4-1

동적 QoS 속성을 고려한 단일 웹서비스의 교환을 지원하는 프레임워크

진 상 찬*, 이 은 주**

A Framework to Support Reconfiguration of Single Web Service Based on Dynamic Qos Properties

Sang-Chan Jin *, Eun-Joo Lee **

요 약

현재 웹 어플리케이션을 구축하는데 있어 웹서비스는 중요한 역할을 담당하게 되었다. 웹서비스의 수와 종류가 많아짐에 따라, QoS는 웹서비스의 선택과 조합에서 중요한 기준이 되었다. 하지만 웹서비스의 QoS요소는 실행환경에 따라 동적으로 변하고, 이로 인해 선택된 웹서비스가 요청자의 QoS요구사항에 부합하지 않는 웹서비스가 되기도 한다. 본 논문에서는 요청자의 QoS요구사항에 적합한 웹서비스를 찾고 웹서비스의 QoS에 기반하여 웹서비스를 동적으로 변경하는 프레임워크를 제안하였다. 그리고 웹서비스 변경 시 발생하는 웹서비스의 인터페이스 적응문제와 수행중인 작업의 보장문제에 대한 해결방법도 함께 제시한다.

Abstract

At present, Web services become main means to construct web applications. As the number of web services grows and various kind of web services have been produced, QoS properties of web services are one of important criteria to select more appropriate web services. However, the selected web services may become inappropriate for the Requester's requirements, because QoS is kind of dynamic properties. this paper proposes a framework that proposed that finds an appropriate web service for Requester's QoS requirements, and dynamically replaces an inappropriate web service with right one. The problems which happen in substitution step are also identified and the solutions are described with an illustrative example.

▸ Keyword : web service, dynamic QoS, framework, interface adaptation

• 제1저자 : 진상찬 교신저자 : 이은주
• 투고일 : 2008. 11. 10, 심사일 : 2009. 01. 06, 게재확정일 : 2009. 02. 12.
* 경북대학교 전자전기컴퓨터학부 석사과정 ** 경북대학교 컴퓨터공학과 조교수

Ⅰ. 서 론

최근 분산된 웹 어플리케이션을 구축할 수 있는 새로운 패러다임으로 웹서비스(Web Services)가 출현하였고, UDDI, WSDL, SOAP 표준을 바탕으로 하는 웹서비스는 서비스 지향 아키텍처를 구현하는 대표적인 형태로 인정받고 있다. 다양한 웹서비스중 적절한 것을 선택하는 것이 중요한 문제로 대두되었는데, 웹서비스의 활용이 증가됨에 따라, QoS(Quality of Service)는 웹서비스 선택과 조합에 필수적인 요소가 되었다. QoS는 요청자의 필요와 사용하는 네트워크 리소스에 기반을 둔 서비스 제공자의 필요를 부합 시키는 기술로써 웹서비스의 QoS는 제공되는 서비스의 가용성, 보안, 응답시간, 신뢰성 등의 여러 품질 요소들의 조합으로 표현된다[1][2][3].

웹서비스의 QoS요소들은 실행 환경에 따라 동적으로 변화한다. 따라서 사용 중인 웹서비스 요청자의 QoS요구사항을 만족하지 않는 웹서비스가 될 수도 있다. 웹서비스의 선택에서 QoS를 중요기준으로 하는 연구는 진행되었으나, QoS의 동적인 변화로 인해 사용 중인 웹서비스가 요청자의 QoS요구사항에 만족하지 못하는 서비스를 동적으로 대치하는 연구는 많지 않다[4].

본 논문에서 요청자의 QoS요구사항에 적합한 웹서비스를 찾고, 웹서비스의 QoS변경에 대하여, 웹서비스를 동적으로 변경하는 프레임워크를 제안한다. 제안된 프레임워크는 동적으로 웹서비스를 교환하는 프레임워크[5]를 확장, 보완한 것으로 요청자, 제공자, UDDI로 이루어진 웹서비스의 아키텍처에 UDDI QoS Broker와 QoS Agent, QoS Server등을 추가하여 웹서비스의 선택과 동적인 QoS의 변화에 반응하여 웹서비스를 다시 선택할 수 있게 하였다. 또한, 웹서비스를 동적으로 변경하는 경우 웹서비스를 사용 중인 요청자의 인터페이스와 교체하게 될 후보 웹서비스의 인터페이스와 서로 상이 할 수 있는데, 이 경우 중개자를 통한 각 인터페이스를 적응하는 방법을 제시한다. 그리고 웹서비스 교체 시 사용 중인 웹서비스로 인해 수행중인 작업이 있다면, 그 작업에 대한 대처가 필요하며, 이를 위해 웹서비스를 통해 수행중인 작업에 대한 구분과 보장 가능여부 및 보장방법도 함께 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구에 대하여 알아보고, 3장에서는 동적 QoS 지원 프레임워크에 대해 알아본다. 4장에서 제안된 프레임워크의 구현 및 테스트를 보이고, 5장에서 결론 및 향후 과제를 맺는다.

Ⅱ. 관련 연구

QoS를 이용한 웹서비스의 선택은 주어진 기능적 요구사항을 만족하는 웹서비스가 여러 개 있을 경우, 응답시간, 신뢰성, 가용성 등과 같은 비 기능적인 요소인 QoS를 고려하여 가장 적합한 웹서비스를 선택하는 것을 말하며, 웹서비스를 효율적으로 선택하기 위해 QoS를 이용한 많은 연구들이 있었다.

[6]와 [7]에서는 동일한 기능을 제공하는 웹서비스들의 품질을 구분하기 위해 비 기능적 요소인 성능, 응답시간, 신뢰성, 가용성 등의 QoS요소들을 파라미터로 정의하여 웹서비스들 간의 QoS를 평가하였다.

[8]에서는 UDDI가 QoS에 관한 정보를 제공하여 중개자나 측정요소를 따로 두지 않아도 고품질의 웹서비스의 선택이 가능하도록 제안하였다. 일반적으로 UDDI는 QoS에 관한 정보를 제공하지 않기 때문에 요청자는 고품질의 웹서비스를 선택하는데 어려움이 있었고, 따라서 요청자는 고품질의 웹서비스를 선택하기 위해 QoS에 대한 정보를 제공해주는 중개자(Broker)와 QoS를 모니터링 하기위한 측정요소(Metrics)가 필요하였다. [8]의 QoS에 관한 정보를 제공하는 UDDI의 경우는 따로 웹서비스의 QoS에 대한 정보를 제공하는 중개자를 둘 필요가 없어 다른 연구들 보다 좋은 성능을 보이지만, QoS에 관한 정보를 제공하는 중개자가 없기에 UDDI는 필수적으로 QoS에 관한 정보를 제공해야 한다는 한계가 있으며, QoS의 동적인 부분을 반영하지 못한다.

[9]과 [10]에서는 QoS요소들에 대하여 요청자와 제공자 간의 서비스가 제공이 될 것인지를 측정이 가능한 조건으로 명시한 Service Level Agreements(SLAs)를 생성하여 웹서비스 QoS의 측정기준을 만들고, 모니터링을 통하여 웹서비스의 QoS를 평가하여 웹서비스를 품질을 구분하도록 하였다.

이러한 연구들은 웹 서비스 선택을 위한 QoS 평가에 중점을 두고 있으며 동적으로 변하는 QoS로 인하여 웹 서비스를 재선택 해야 하는 부분에 대한 고려가 부족하다. 또한 일부 연구[8]에서는 웹 서비스가 제공될 때 QoS 정보를 포함한다는 제한이 존재하기도 한다. 본 논문에서는 기존 연구들과는 달리 QoS의 동적 교환에 초점을 맞추었다. 이를 위하여, 제안된 프레임워크에서는 주기적으로 QoS 데이터를 수집하여 활용하고, 수행중인 웹 서비스의 동적 교환이 이루어질 때 발생하는 문제점들을 파악하고 해결한다.

III. 동적 QoS 지원 프레임워크

1. QoS 지원 프레임워크

본 논문이 제안하는 프레임워크는 크게 UDDI QoS Broker (Service Finder, Service Adapter), QoS Server, QoS Agent의 핵심요소로 구성된다(그림1).

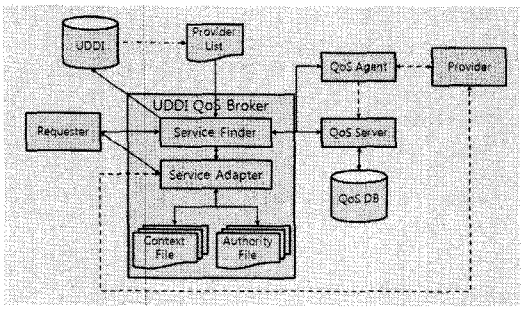

그림 1. QoS 지원 프레임워크
Fig. 1. QoS support framework

- Service Finder

Service Finder는 UDDI를 탐색하여 웹서비스의 리스트를 찾고, QoS Server를 통하여 리스트내의 웹서비스들 중 요청자의 QoS요구사항에 가장 근접한 웹서비스를 찾는다. 또한, 리스트내의 웹서비스 중 QoS Server에 등록되어 있지 않은 웹서비스가 있으면, QoS Agent에게 QoS Servier로 등록되지 않은 웹서비스들의 QoS등록을 요청한다.

- Service Adapter

Service Adapter는 웹서비스의 동적 QoS변경으로 인해 사용 중인 웹서비스가 요청자의 QoS요구조건에 만족하지 못하면, 웹서비스를 변경한다. 웹서비스를 동적으로 변경하기 위해, 후보 웹서비스의 인터페이스는 현재 사용 중인 웹서비스의 인터페이스에 적응시킨다.

- QoS Server

제공자가 제공하는 웹서비스의 QoS를 QoS DB에 저장한다. UDDI QoS Broker로부터 웹서비스의 리스트를 받고, 리스트 내에 있는 웹서비스의 QoS 항목들과 요청자의 QoS 요구사항을 비교하여 랭킹리스트를 만들어 UDDI QoS Broker에게 전달한다.

- QoS Agent

QoS Agent는 UDDI QoS Broker로부터 QoS등록 요청을 받거나 자체적으로 서비스제공자들로부터 받은 웹서비스의 QoS를 QoS Server에 등록한다. 그리고 웹서비스의 QoS 변경이 발생할 경우에는 UDDI QoS Broker에게 알려준다.

1.1. 사용 시나리오

QoS 지원 프레임워크는 동적으로 변화하는 웹서비스의 QoS를 모니터링하여 요청자에게 가장 적합한 웹서비스를 제공하며, 그 작동 순서는 다음과 같다.

① 요청자는 Service Finder에게 사용하고자하는 웹서비스의 키워드와 QoS를 보내고, Service Adapter로부터 웹서비스 사용하기 위해 대기 한다.

② Service Finder는 키워드를 이용하여 UDDI를 검색하여 제공자리스트를 받고, 리스트를 QoS Server로 보내어 리스트내의 각 웹서비스들의 QoS를 받는다. 그리고 QoS Server에 QoS가 등록되지 않은 웹서비스는 QoS Agent에게 QoS Server에 QoS등록을 요청하고, QoS Server에 QoS등록이 완료되면 다시 QoS를 받는다. QoS Server로부터 받은 웹서비스들의 QoS를 가지고 웹서비스의 랭킹리스트를 만들고, 가장 높은 순위의 웹서비스를 Service Adapter에게 알리고, QoS Agent에게 리스트의 웹서비스들의 QoS모니터링을 요청한다.

③ Service Adapter는 선택된 웹서비스의 인터페이스와 요청자의 인터페이스를 연결한다.

④ QoS Agent는 지속적인 모니터링을 통하여 QoS의 변경이 발생한 웹서비스를 QoS Server에 등록하고, 웹서비스를 Service Finder에게 알려준다.

⑤ Service Finder는 랭킹리스트의 최상위 웹서비스가 변하면 최상위순위가 된 웹서비스를 Service Adapter에게 알려준다.

웹서비스는 실행 환경에 따라, 웹서비스의 가용성, 신뢰도, 성능이 저하되는 등, QoS가 동적으로 변한다. 이 경우, 요청자는 같은 서비스를 제공하는 새로운 웹서비스로의 교체가 필요하다.

사용 중인 웹서비스를 다른 후보 웹서비스로 교체 시 몇 가지 사항을 고려해야 하며 이를 다음 절에서 기술한다.

2. 웹서비스 교환 시 고려사항

2.1. 인터페이스 적응

기존 웹서비스와 교체할 후보 웹서비스는 서로 상이한 인터페이스를 가지고 있고, 요청자의 인터페이스 또한 사용 중인 웹서비스와 다를 수 있다. 웹서비스가 교체될 때마다 새로운 웹서비스를 사용하기 위해서는 웹서비스의 인터페이스와 요청자의 인터페이스가 서로 통신할 수 있도록 변환하는 과정이 필요하며, 변환과정의 주요 역할을 Service Adapter가 수행한다. 그리고 변환하는 과정에는 어댑터 패턴(Adapter Pattern)[11][12]을 적용하였다.

제안한 프레임워크에서 요청자와 제공자간의 통신은 Service Adapter를 통해 이루어진다(그림2).

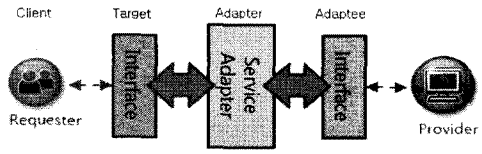

그림 2. 요청자와 제공자간의 통신
Fig. 2. Communication between the requester and the provider

어댑터 패턴을 적용하기 위해 요청자와 제공자 사이에 서비스를 제공하기 위한 통신과정에서 요청자는 클라이언트(Client)로, 요청자의 인터페이스는 타깃(Target)으로, Service Adapter는 어댑터(Adapter)로, 웹서비스의 인터페이스는 어댑티(Adaptee)로 각각 역할을 구분을 하였다.

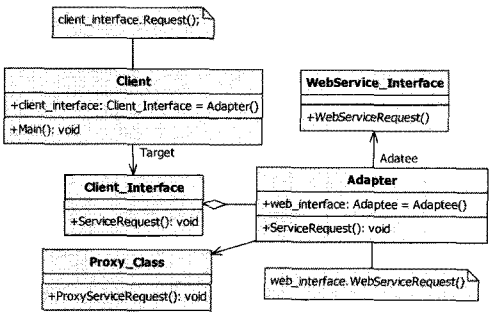

그림 3. 인터페이스 매칭에 적용된 어댑터 패턴
Fig. 3. Adapter pattern applied to interface matching

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:tns="http://tempuri.org/"
  xmlns:s="http://www.w3.org/2001/XMLSchema"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  targetNamespace="http://tempuri.org/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
 <wsdl:types>
  - <s:schema elementFormDefault="qualified"
      targetNamespace="http://tempuri.org/">
    - <s:element name="Yeyak">
      - <s:complexType>
        - <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="sinNm"
              type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="sinPhon"
              type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="sinMail"
              type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="reNm"
              type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="rePhon"
              type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="reJuso"
              type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="reMemo"
              type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
    - <s:element name="YeyakResponse">
```

그림 4. WSDL 예
Fig. 4 An example of WSDL

제안한 프레임워크에서 QoS의 동적 변화에 따라 웹서비스가 동적 교체된다. 이로 인해, 웹서비스의 인터페이스가 변하게 되고, 이것은 어댑티가 변하는 것을 의미한다. 본 논문에서는 이러한 상황에 유연하게 대처하기 위하여 위임(delegation)을 이용한 어댑터 패턴[13]에 프록시 클래스(Proxy Class)를 추가하여 구현하였다(그림3). 프록시 클래스는 Service Adapter로부터 만들어지며, 요청자와 제공자 사이에서 실질적인 메시지전달 역할을 한다.

웹서비스의 사용은 메서드를 이용하며 이 메서드는 서비스 제공자가 제공하는 WSDL(Web Service Description Language)에 기술되어 있으며, 이 WSDL을 통해 웹서비스의 인터페이스를 알 수 있다[14][15](그림4).

Service Adapter는 WSDL을 통해 확인한 웹서비스의 인터페이스와 요청자의 인터페이스가 서로 호환되도록 변환하는 역할을 수행하는 것으로, Service Adapter는 요청자가 웹서비스를 사용하기위해 해당 웹서비스의 WSDL을 이용하여 프록시 클래스를 만든다. 프록시 클래스는 요청자와 웹서비스 사이에서 대리로 통신을 수행하는 기능을 가지는 클래스로서 Service Adapter가 웹서비스에 실제로 접근이 가능하게 하는 기능을 한다(그림5). 이렇게 만들어진 프록시 클래스를 통해 Service Adapter는 요청자의 인터페이스로부터 받은 서비스에 관한 요청의 결과를 웹서비스로부터 받는다.

```
[System.Web.Services.Protocols.SoapDocumentMethodAttribute
    ("http://tempuri.org/Reservation", RequestNamespace="http://tempuri.org/",
    ResponseNamespace="http://tempuri.org/",
    Use=System.Web.Services.Description.SoapBindingUse.Literal,
    ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped)]
public string Reservation(string send_name, string send_address,
    string send_post, string send_phonNumber, string send_mail,
    string receive_name, string receive_address, string receive_post,
    string receive_phonNumber) {
    object[] results = this.Invoke("Reservation", new object[] {
                send_name,
                send_address,
                send_post,
                send_phonNumber,
                send_mail,
                receive_name,
                receive_address,
                receive_post,
                receive_phonNumber});
    return ((string)(results[0]));
}

/// <remarks/>
public System.IAsyncResult BeginReservation(string send_name,
    string send_address, string send_post, string send_phonNumber,
    string send_mail, string receive_name, string receive_address,
    string receive_post, string receive_phonNumber,
    System.AsyncCallback callback, object asyncState) {
    return this.BeginInvoke("Reservation", new object[] {
                send_name,
                send_address,
                send_post,
                send_phonNumber,
                send_mail,
                receive_name,
                receive_address,
                receive_post,
                receive_phonNumber}, callback, asyncState);
}
```

그림 5. WSDL의 정보로 생성된 프록시 클래스
Fig. 5 A proxy class generated using WSDL information

요청자의 웹서비스에 관한 요청과 제공자의 웹서비스 제공은 메서드를 통해 이루어지며, 변환과정에서 Service Adapter는 요청자의 메서드 파라미터와 웹서비스의 메서드 파라미터를 매칭한다. 이 파라미터들 간의 매칭을 통해 웹서비스 사용에 필요한 정보들을 찾는다. 파라미터들 간의 매칭을 하려면 각 파라미터가 어떠한 정보를 담고 있는지를 파악해야 하며, 이를 위해 파라미터의 이름을 이용한다. 파라미터의 이름은 제공자만 알 수 있는 암호형식의 단어나 숫자 등의 경우를 제외하면, 일반적으로 파라미터가 담고 있는 정보를 나타낸다. 이러한 파라미터의 이름을 대표 단어와 파생단어들의 매핑으로 이루어진 온톨로지 기반의 전거사전(Authority File)을 통하여 매칭 한다[16]. 그리고 파라미터들 간의 데이터 타입은 일반적으로 사용하는 프로그램언어의 데이터타입 변환방식을 사용하여 매칭 한다. 예를 들어 사용 중인 메서드의 파라미터 타입이 integer이고, 후보 메서드의 파라미터 타입이 float일 경우 integer타입은 float타입으로 변환한다.

2.2. 수행중인 작업의 보장

웹서비스를 교체할 경우 기존 웹서비스를 통해서 어떠한 작업을 수행중이라면 요청자가 요청한 작업의 신뢰성, 정확성 등을 보장하기 위해 작업의 보장이 필요하다. 웹서비스를 통해 수행되는 작업은 여러 가지 유형이 있으며, 작업의 유형에 따라 보장이 가능한 작업과, 불가능한 작업을 구분하게 된다.

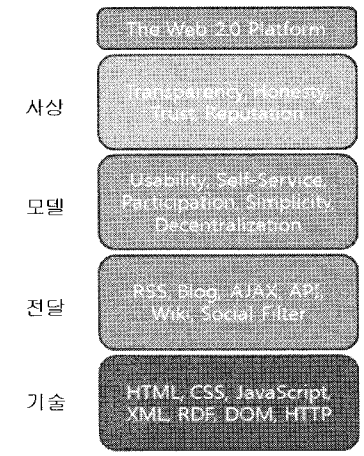

그림 6. 웹2.0 의 플랫폼[17]
Fig. 6 Web 2.0 Platform[17]

수행중인 작업의 보장 가능여부를 구분하기 위해서는 웹서비스를 통해 수행되는 작업에 대한 구분이 필요하였고, 단순한 요청자와 제공자의 관계로 구분되는 웹1.0보다 동적이며 블로그의 트랙백이나 위키와 같이 각 주체가 요청자이면서 동시에 제공자가 되는 상호작용이 가능한 웹2.0을 기준으로 구분 하였다. 웹2.0 플랫폼(그림6)은 사상, 모델, 전달, 기술 4단계 구분된다. 그 중 전달계층은 요청자가 웹서비스를 사용할 경우 실제 웹서비스의 제공부분을 관여하는 요소들을 포함하며, 이 요소에 의해서 웹서비스를 통해 수행되는 작업들의 위치여부가 결정된다.

본 논문에서는, 전달계층을 기준으로 하여 웹서비스를 통해 수행되는 작업을 다음과 같이 나누었다.

- 정보 조회 작업
- 데이터 수정 작업
- 데이터 기록 작업
- 데이터 전송 작업

정보 조회 작업은 날씨정보, 영화 및 드라마 정보, 금융정보 등과 같이 요청자가 필요한 정보를 조회하여 제공자로부터 정보를 제공 받는 작업을 말하며, 데이터 수정 작업은 번역, 바이러스검사 등과 같이 요청자의 데이터나 제공자가 소유한 데이터를 수정하는 작업을 말한다. 그리고 데이터 기록 작업은 예약, 인터넷 뱅킹 등과 같이 요청자나 제공자의 시스템에 데이터를 기록하는 작업을 말하며, 데이터 전송 작업은 p2p 기반의 파일전송, ftp 등과 같이 요청자와 제공자 사이의 파일을 전송하는 작업을 말한다.

정보 조회 작업에서 요청자는 필요한 정보를 제공자로부터

제공받는 역학을 한며, 데이터는 항상 요청자에게 위치하게 된다. 하지만, 데이터의 전송, 수정, 기록 작업은 데이터의 위치가 각 작업의 유형에 따라 다르며, 데이터의 위치에 따라 작업보장의 여부를 결정한다. 데이터가 요청자에게 전송, 기록, 수정되는 경우는 새로운 제공자로 변경이 되어도 Context File을 통한 작업의 보장이 가능하지만, 제공자에 위치하는 경우는 새로운 제공자로 변경이 되면, 데이터는 기존에 사용하던 제공자에게 종속되는 것이므로 작업의 보장이 불가능 하다. 예를 들어 p2p 기반의 파일 전송 작업의 경우, 요청자는 동일한 파일을 제공하는 제공자로부터 파일을 연속하여 전송 받을 수 있지만, 데이터를 제공자에게 보내는 ftp의 경우는 새로운 제공자로 바뀌면, 데이터는 기존의 제공자에게 전송을 하던 것이었고, 새로운 제공자에게 데이터를 제공한다는 것은 다른 ftp서버에 데이터를 저장 하는 것이 된다.

하지만, 이 방법이 모든 유형의 작업에 적용이 되는 것은 아니다. 본 논문에서는, 앞에서 구분한 웹서비스를 통해 수행되는 작업들은 작업의 정보나 데이터의 위치에 따라서 작업의 보장을 결정한다. 하나의 작업에 필요한 정보나 데이터는 요청자와 제공자가 서로 주고받는다. 작업의 정보나 데이터가 요청자에게 저장되게 된다면 작업의 보장이 가능하지만, 제공자에게 저장되게 된다면 작업의 보장이 불가능 하게 된다. 간단하게 문서번역과 인터넷결제를 예로 들면, 번역서비스의 경우는 제공자가 요청자로부터 번역할 문서를 받아서 번역을 하고, 요청자에게 다시 전달한다. 이 경우는 데이터가 요청자에게 위치하는 경우에 속한다. 이때, 동적 변경이 일어나게 되면, 번역할 문서는 Service Adapter를 통하여 진행 위치와 가공 중인 문서를 Context File에 저장하고 웹서비스 교체 후 문서를 마지막으로 번역했던 위치에서 다시 번역하게 된다.

표 1. Context File의 형식
Table 1. Format of a context file

작업 상태 (Work State)	생성(New)	작업이 생성 되는 상태
	준비(Ready)	작업생성 후 실행되기 위한 설정상태
	실행(Run)	작업이 실행 중인 상태
	대기(Wait)	실행 작업 완료 후 다음 작업을 설정하는 상태
	완료(Complete)	작업의 종료
작업 명령 (Work Command)	실행중인 작업과 실행후의 작업 상태	
입출력 상태 정보 (IO Information)	작업에 할당된 입출력 정보와 사용 중인 파일목록	

예약 서비스의 경우는 제공자는 요청자로부터 예약에 관한 데이터를 제공받고, 그 데이터를 제공자의 시스템에 저장한다. 이는 데이터가 제공자에게 위치하는 경우에 속한다. 이 경우에서 웹서비스 교체가 일어나게 되면, 예약에 관한 데이터를 저장해야할 제공자의 시스템이 바뀌게 되는 상황이 발생할 수 있게 되며, 이와 같은 경우는 작업의 보장으로 인해 요청자의 의도와는 다른 결과가 나온다.

작업의 보장은 웹서비스 교체 시 Context File에 수행중인 작업을 저장한다. Context File에 저장되는 작업은 작업 상태(Work State), 작업 명령(Work Command), 입출력 상태 정보(IO Information)로 저장 된다[18]〈표1〉.

웹서비스를 통해 수행되는 작업은 생성, 준비, 실행, 대기, 완료의 상태를 가지고 있고, 실행 상태에서 수행중인 작업이 완료된 후 다음 수행할 작업의 유무에 따라 대기상태와 완료상태로 분기한다(그림7).

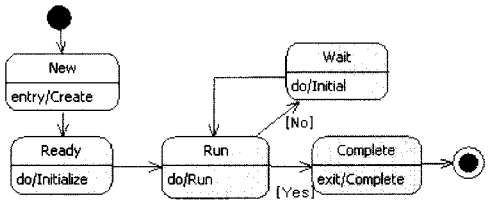

그림 7. 작업 상태 다이어그램
Fig. 7 State diagram of a work

웹서비스의 변경은 생성, 준비, 실행, 대기, 완료상태중 어느 상태에서도 일어날 수 있다. 작업상태가 생성, 준비, 완료상태일 경우는 작업이 수행되기 전이거나 완료된 상태이므로 수행중인 작업은 없다. 그러므로 웹서비스의 변경이 있어도 작업의 보장과는 관계가 없다. 하지만, 작업상태가 실행, 대기상태일 경우는 작업이 수행중이거나 다음 작업의 수행을 위한 대기 상태이므로 웹서비스 변경 시 작업의 보장을 위한 과정이 필요하다.

저장된 작업은 웹서비스 교체 후 변경된 웹서비스에 적용한다. 웹서비스의 적용은 기존의 수행하던 작업의 상태와 실행중인 작업, 그리고 작업에 할당된 파일등을 저장하고, 이 정보들이 변경된 후보 웹서비스에 사용된다.

Context File의 저장형식은 DTD(Document Type Definition)[19]를 이용한 연구[20]의 일부를 적용 및 변형하였다. 웹서비스 교체 시 수행중인 작업의 보장 순서는 다음과 같다.

① 수행되던 작업상태의 확인 및 기록

- 웹서비스를 통해 수행되던 작업은 웹서비스의 변경 시 Service Adapter는 작업의 상태를 Context파일에 XML스키마를 이용하여 기록한다(그림8). workState는 작업 상태를 내타내고, workCommand는 작업 명령을, 그리고 IOInfo는 입출력 상태 정보를 나타낸다.

작업상태가 실행, 대기상태일 경우는 작업이 수행중이거나 다음 작업의 수행을 위한 대기 상태이므로 작업의 보장을 위해 ②의 과정이 필요하다. 그 외의 생성, 준비, 완료 상태는 작업을 수행하기 전과 작업 수행의 완료를 뜻하는 것이므로 작업의 보장이 필요하지 않다.

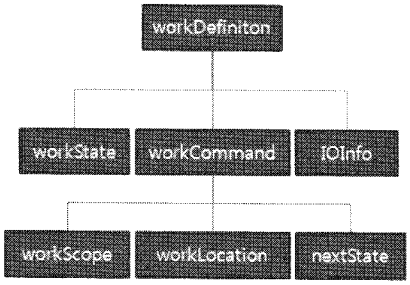

그림 8. Context파일의 구조
Fig. 8 Structure of context file

② 작업의 상태에 따라 실행 중인 작업 기록

- 작업의 상태가 실행이나 대기상태일 경우는 현재 수행중인 작업이 존재하는 것이고, 이 작업의 보장을 위해 현재 실행중인 명령과 다음에 실행할 명령이 필요하다. workCommand는 작업범위를 나타내는 workScope, 진행위치를 나타내는 workLocation, 다음 작업 상태를 나타내는 nextState로 구성되어 있다(그림8). 예를 들어 특정 지역의 날씨 조회를 요청할 경우, 범위는 전체 요청 쿼리의 크기, 진행위치는 제공자에게 보낸 쿼리의 양, 작업 상태는 쿼리를 보낸 후의 상태를 기록한다. 날씨 조회의 결과를 받을 경우, 범위는 전체 받을 정보의 양, 진행위치는 현재까지 받은 양, 날씨 정보를 모두 받은 후의 상태를 기록한다. 또 다른 예로 파일을 다운로드 하고 있을 경우, 범위는 다운받은 받는 파일의 전체크기, 진행위치는 현재까지 받은 양, 작업 상태는 파일을 모두 다운로드 한 다음의 상태를 기록하게 된다.

③ 작업에 할당된 파일이나 입출력 정보의 목록 기록

- ②의 단계에 따라서 작업수행에 포함된 데이터의 목록을 기록한다. 정보에 대한 조회일 경우, 데이터들의 목록은 그 정보를 표현하기 위한 데이터들의 목록을 말하는 것이고, 파일을 다운로드 하는 중이라면, 다운로드 하는 파일을 말하는 것이다.

IV. 예제 구현

본 장에서는 도서 검색서비스와 Filedown예제를 통해 인터페이스 적응과 웹서비스 교체 시 발생하는 수행중인 작업의 보장의 예를 보인다.

1. 인터페이스 적응 - 도서 검색서비스

그림 9는 도서 검색서비스의 예로 QoS Server로부터 전달받은 웹서비스의 랭킹리스트이다.

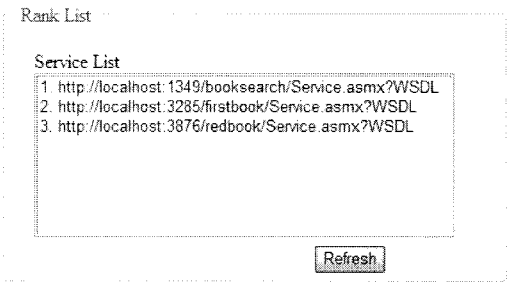

그림 9. 웹서비스 랭킹 리스트
Fig. 9 Web services ranking list

Service Finder는 웹서비스의 QoS가 변경되면, QoS Server로부터 랭킹 리스트를 다시 받아서 연동된 웹서비스의 순위를 확인하고 순위변화를 확인하고, 순위변화가 없으면 연동된 웹서비스를 그대로 사용하지만, 순위변화가 있으면 후보 웹서비스로 교체한다. 이때, 후보 웹서비스로 교체 시 Service Adapter는 요청자가 사용 중인 웹서비스와 후보 웹서비스의 WSDL로부터 메서드의 정보를 추출하고(그림10), 두 웹서비스의 메서드를 적응 한다.

적응순서는 각 파라미터의 이름에 대한 유사성을 매칭하고, 매칭 한 파라미터의 데이터 타입을 매칭 한다. 앞의 과정에서 후보 웹서비스의 서비스를 이용하기 위한 메서드의 파라미터가 충분하지 않다면, 요청자에게 입력을 요청한다.

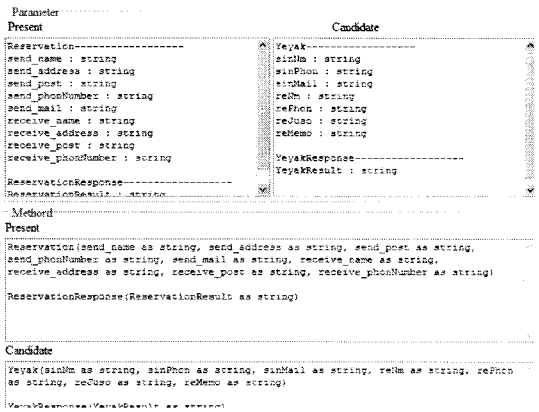

그림 10. 각 WSDL로부터 추출된 메서드 정보
Fig. 10. Method information extracted from each WSDL

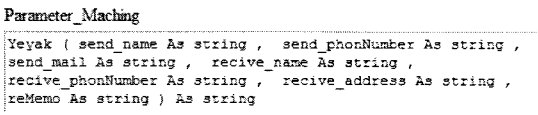

그림 11. 후보 웹서비스의 메서드 (메서드 적응 후)
Fig. 11. A method of a candidate web service (after method interface adaptation)

그림11은 메서드의 매칭결과로, 요청자가 사용 중인 웹서비스 메서드의 파라미터들이 후보 웹서비스 메서드의 파라미터에서 사용되는 것을 확인 할 수 있다.

앞의 과정을 통해 요청자의 인터페이스를 후보 웹서비스의 인터페이스와 적응하기위한 메서드의 적응 과정을 보였다.

2. 수행중인 작업 보장 - Filedown

Filedown예제는 수행중인 작업의 보장에 관한 예로, 웹서비스 교체 시 수행중인 작업을 후보 웹서비스에서 연속적으로 수행되도록 Service Adapter의 Context File을 이용하여 작업을 보장하는 것을 보인다.

Filedown은 filePage와 contextPage로 구성되어 있으며, 파일다운로드 작업 중 웹서비스교체로 인해 새로운 제공자로 변경되면, Context File의 내용을 이용하여 동일한 파일을 제공하는 다른 제공자로부터 기존에 받던 파일을 연속하여 다운로드받게 된다.

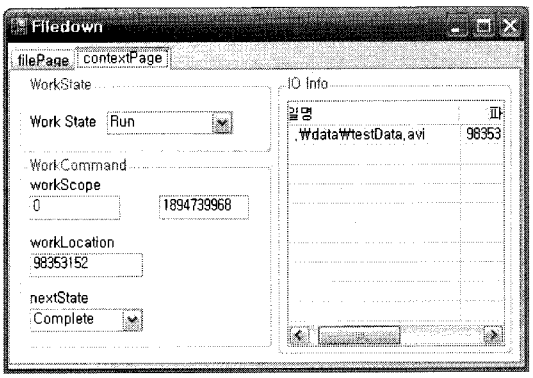

그림 12. Context File에 저장되는 작업
Fig. 12. Work stored in the Context File

contextPage는 웹서비스교체 시 기존에 수행 중인 작업이 저장되는 Service Adapter내의 Context File의 역할을 하는 것으로, 저장되는 형식은 작업상태와 작업명령, 그리고 입출력 상태정보로 3.2.2에서 설명한 형식으로 저장된다. 그림12는 contextPage에서 수행중인 작업이 저장되는 모습을 실시간으로 보여준다. 작업명령의 workScope는 작업의 범위, workLocation은 작업의 진행위치, nextState는 다음 작업의 상태를 나타내며, IO info는 현재 다운로드 중인 파일을 나타낸다.

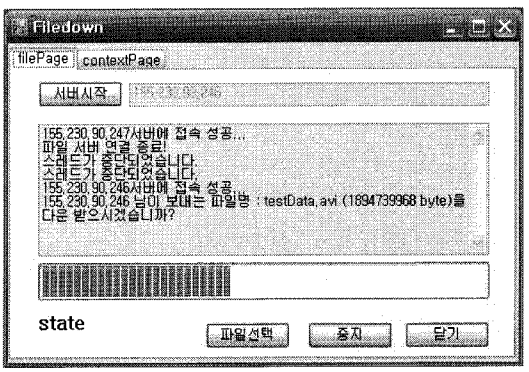

그림 13. 수행 중인 작업 상황(웹서비스 변경 전)
Fig. 13. Web Services change the reconfiguration before the work state Work state (before web service exchange)

현재 수행중인 작업이 파일다운로드 이므로 workScope는 파일의 크기를 나타내고, WorkLocation은 현재까지 파일을 다운받은 양을 나타낸다.

filePage는 현재 다운중인 파일의 진행 상태를 나타내는 것으로 현재 다운중인 파일의 URL과 진행 정보를 보여준다.

그림13과 그림14에서는 파일을 전송하는 제공자의 주소와 작업의 진행 상태를 확인 할 수 있고, 현재 수행중인 작업이 웹서비스 교체로 인해 다른 제공자로부터 파일을 연속적으로 받는 것을 보여준다.

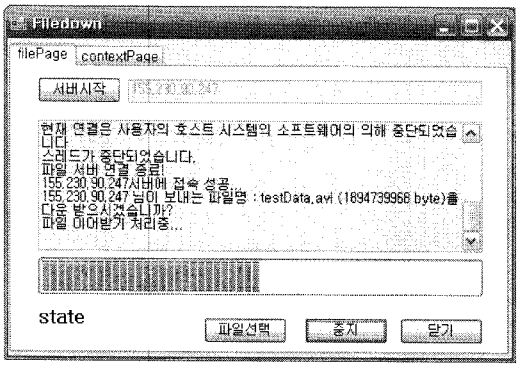

그림 14. 수행 중인 작업 상황(웹서비스 변경 후)
Fig. 14. Work state after Web Services change the reconfiguration Work state (after web service exchange)

그림14의 상태에서 기존 제공자와 연결이 종료되고 후보 제공자로 연결이 될 때, Filedown은 저장된 작업의 진행위치와 파일이름, 파일의 크기를 후보 제공자에게 보내고, 후보 제공자는 파일의 이름과 크기를 이용하여 요청하는 파일을 찾고, 요청자로부터 받은 진행위치다음부터 파일을 전송한다.

위의 Filedown예제를 이용한 테스트를 통해, 웹서비스의 동적 교체 시 기존 웹서비스를 통해 수행중인 작업을 Context File을 이용하여 보장하는 과정을 보였다. 그리고 Context File을 이용한 작업보장은 테스트로 보인 파일전송 이외의 다른 작업에도 적용이 가능할 것이다.

V. 결론 및 향후 과제

본 논문에서는 UDDI와 웹서비스의 QoS를 적용하여 요청자의 QoS요구사항에 부합하는 웹서비스를 찾고, 웹서비스 QoS의 동적 변경으로 인해 요청자의 QoS요구사항에 만족하지 못하게 된 웹서비스를 새로운 웹서비스로 변경하는 프레임워크를 제안하였다. 그리고 동적으로 웹서비스를 변경하기위한 웹서비스간의 인터페이스 적응문제와, 사용 중인 웹서비스의 작업 보장 문제에 대한 해결 방안도 제시하였다. 제안된 프레임워크는 웹서비스 사용 시 요청자의 요구사항에 맞지 않는 웹서비스를 동적으로 교체하여 요청자의 만족도를 높일 수 있을 것이다.

향후 연구는 웹 환경에서 실제 사용되는 웹서비스들을 대상으로 실험을 수행하여 제안된 프레임워크의 실용성을 검증하는 실험을 수행할 계획이다.

감사의 글

이 논문은 2007년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었습니다 (KRF-2007-331-D00407).

참고문헌

[1] A. Mani, N. Arun, "Understanding Quaility of Service for Web Services," *http://www.ibm.com/developerworks/library/ws-quality.html*, 2002.

[2] D.A Menasce, "QoS issues in Web Services," IEEE Internet Computing, Vol. 6, Issue 6, pp. 72-75, 2002.

[3] K.C. Lee, J.H. Jeon, W.S. Lee, S.H. Jeong, S.W. Park, "QoS for Web Services:Requirements and Possible Approaches," *http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/*, 2003

[4] 김원상, 장희정, 이강선, "QoS 기반의 웹 프로세스 조합 방법론 및 지원도구의 개발," 한국시뮬레이션학회 2005년 춘계학술대회논문집, 134-138쪽, 2005년.

[5] 진상찬, 송유진, 이은주, "QoS 기반 웹서비스 동적 교환 지원 프레임워크," 한국컴퓨터정보학회 하계학술발표논문집, 105-110쪽, 2008년.

[6] A. Eyhab, H. M. Qusay, "QoS-based Discovery and Ranking of Web Services," ICCCN 2007. Proceedings of 16th International Conference, pp529-534, 2007.

[7] M.Tian, A. Gramm, T. Naumowicz, H. Ritter, J. Schiller, "A Concept for QoS Integration in Web Services," Web Information Systems Engineering Workshops, 2003.

[8] OASIS. "Technical Note, Repre-senting Web Services Quality of Service Information in UDDI", 2004.

[9] A. Dan, A. R. Franck, A. Keller, R. King, H. Ludwing, "Web Service Level Agreement(WSLS)

LanguageSpecification," *http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf*, 2003.

[10] A. Sahai, V. Machiraju, M. Sayal, L. Jie Jin, F. Casati, "Automated SLA Monitoring for Web Services, *http://www.hpl.hp.com/techreports/2002/HPL-2002-191.pdf*, 2002.

[11] J.W. Cooper, *The Design Patterns Java Companion*. addison wesley, first edition, pp. 81-89, 1998.

[12] E.Gamma, R.Helm, R.Johnson, J.M. Vlissides, *Design Pattern: Elements of Reusable Object-Oriented Software*. addison wesley, 1995.

[13] 이호성, "컴포넌트 기반의 레거시그로그램 통합을 위한 Adapter와 Facade 패턴의 적용기법," 한국정보과학회 학술발표논문집B, .322-3324쪽, 2005년.

[14] M. Ivan, Y. Boris and S. Zoran, "Towards Dynamic Web Service Generation on Demand, Software in Telecommunications and Computer Networks," SoftCOM 2006, pp. 276-280, 2006.

[15] P. W. Chan, L. R. Michael, "Dynamic Web Service Composition: A New Approach in Building Reliable Web Service," AINA 22nd International Conference, pp. 20-25, 2008.

[16] 김영수, 남택용, 원동호, "등급에 따른 웹 유해 문서 분류 기술", 정보처리학회 논문집C, .859-864쪽, 2006년

[17] D. Zarnbonini, posted on oreilly.com, *http://www.oreillynet,com/xml/blog/2006/08/why_you_should_let_web_20_into.html*, 2006.

[18] Deitel, Harvey M., *"An introduction to operating systems,"* pp. 57-58, Addison-Wesley, Boston, MA, USA, 1990 .

[19] B.Tim, P.Jean, C.M.Sperberg, M.Eve, F.Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)," *http://www.w3.org/TR/REC-xml/#dt-doctype*, 2008.

[20] 허율, 홍의경, "EJB를 이용한 XML 문서 저장," 한국정보과학회 학술발표논문집 제28권 제2호(I), .157-159쪽, 2001년.

저 자 소 개

진 상 찬

2006년 2월 : 경일대학교 컴퓨터공학과 (학사)

2008년 3월 ~ 현재 : 경북대학교 전자전기컴퓨터학부 (석사과정)

관심분야 : 웹공학, 웹서비스

이 은 주

2005년 2월: 서울대학교 전기컴퓨터공학부 (공학박사)

2006년 3월 ~ 현재: 경북대학교 컴퓨터공학과 조교수

관심분야 : 웹공학, 메트릭, 웹서비스