韓國 컴퓨터情報學會 2008 제38차 하계학술발표논문집
第16卷 第1號, 2008. 06.

QoS 기반 웹서비스 동적 교환 지원 프레임워크

진 상 찬*, 송 유 진*, 이 은 주*

Dynamic Reconfiguration of Web Services Based on QoS Properties

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

요 약

웹서비스는 XML기반의 표준인 SOAP, WSDL, UDDI를 이용하여 인터넷을 통해 기능을 제공해 주는 서비스 기반 아키텍처의 대표적 기술이다. 웹서비스의 개수가 많아지고 종류가 다양해 짐에 따라, 요청자의 요구사항에 맞는 웹서비스를 선택하는 것이 중요하게 되었으며, 여기서 웹서비스의 선택 시 QoS(Quality of Service)는 중요한 기준이 된다. 하지만 웹서비스의 QoS는 동적으로 변하고, 이로 인해 요청자의 QoS요구사항에 만족하지 않는 웹서비스가 되기도 한다. 본 논문에서는 요청자의 QoS요구사항에 적합한 웹서비스를 찾고, 웹서비스의 QoS변화로 인하여 웹서비스를 동적으로 변경하는 프레임워크를 제안한다. 그리고 웹서비스를 동적으로 변경하는 경우에 필요한, 각 웹서비스의 인터페이스 적응 및 기존 작업의 보장에 대한 해결방법도 함께 제시한다.

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

• 제1저자 : 진상찬
* 경북대학교 전기전자컴퓨터공학부

Ⅰ. 서 론

웹서비스는 서비스들이 플랫폼에 종속 받지 않고 네트워크를 통한 상호 작용을 하도록 설계된 서비스기반 아키텍처(SOA; service oriented architecture)의 대표적 기술이다. XML기반의 표준인 WSDL(Web Service Description Language), UDDI(Universal Description Discovery and Integration), SOAP(Simple Object Access Protocol)를 사용하여 동적인 연동을 통한 분산 서비스를 지원한다[1][2][3]. 웹서비스 제공자(Provider)들은 서비스를 빠르고 효율적으로 제공하기 위하여, 기존의 웹서비스를 조합하여 프로세스를 구성하기도 한다.

이러한 환경에서 웹서비스 요청자(Requester)의 QoS(Quality of Service)요구사항은 웹서비스 선택 시 중요한 기준이 될 수 있다 [3][4]. 웹서비스의 QoS는 동적으로 변경될 수 있으므로, 해당 웹서비스의 사용 중에 서비스 요청자의 QoS요구사항에 만족하지 않는 웹서비스가 될 수도 있다. 웹서비스의 선택 시 QoS를 중요기준으로 하는 연구는 진행되었으나, 웹서비스를 선택한 후 QoS의 동적 변화로 인한 요청자의 QoS요구사항에 만족하지 못하는 서비스를 동적으로 변경하는 연구는 많지 않다 [5].

본 논문에서 요청자의 QoS요구사항에 적합한 웹서비스를 찾고, 웹서비스의 QoS변경에 대하여, 웹서비스를 동적으로 변경하는 프레임워크를 제안한다. 또한, 웹서비스를 동적으로 변경하는 경우 각 웹서비스의 인터페이스 적응 및 사용 중인 웹서비스의 변경으로 인한 작업의 보장에 대한 해결방법도 함께 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 동적 QoS 지원 프레임워크를 알아보고, 3장에서는 동적 웹서비스 변경에 대한 예를 보이고 4장에서 결론 및 향후 과제를 맺는다.

Ⅱ. 동적 QoS 지원 프레임워크

2.1. QoS 지원 프레임워크

본 논문이 제안하는 프레임워크는 크게 UDDI QoS Broker, QoS Server, QoS Agent로 구성된다 (그림1)(그림2).

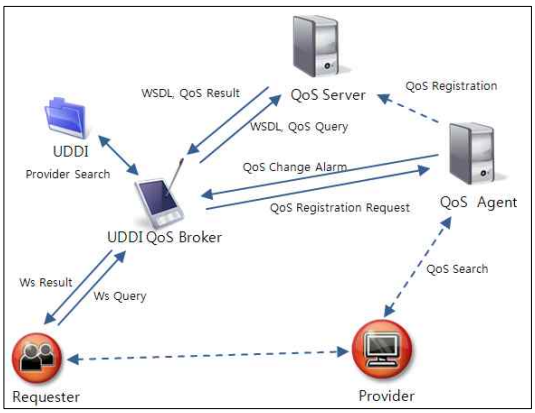

그림 1. 동적 QoS 지원 프레임워크

제안된 QoS 지원 프레임워크는 다음 두 가지 사항에 초점을 맞추었다.

- 웹서비스의 동적 QoS변경으로 사용 중인 웹서비스가 요청자의 QoS요구사항을 만족하지 못하면 웹서비스를 동적으로 변경한다.
- 웹서비스의 동적 변경 시 수행 중이던 작업을 보장한다.

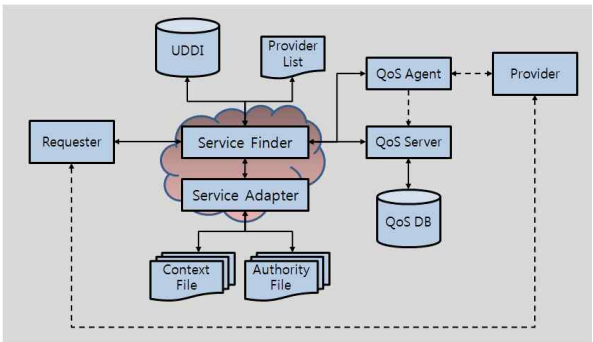

그림 2. QoS 지원 프레임워크 구성

2.1.1. UDDI QoS Broker

UDDI QoS Broker는 Service Finder와 Service Adapter로 구성된다 (그림2). 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 Adapter는 웹서비스의 동적 QoS변경으로 인해 사용 중인 웹서비스가 요청자의 QoS요구

조건에 만족하지 못하면, 웹서비스를 변경한다. 웹서비스를 동적으로 변경하기 위해, 후보 웹 서비스의 인터페이스는 현재 사용 중인 웹서비스의 인터페이스에 적응시킨다.

2.1.2. 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.

2.1.3. QoS Agent

QoS Agent는 UDDI QoS Broker로부터 QoS등록 요청을 받거나 자체적으로 서비스제공자들로부터 받은 웹서비스의 QoS를 QoS Server에 등록한다.

그리고 웹서비스의 QoS변경이 발생할 경우에는 UDDI QoS Broker에게 알려준다.

아래 그림3은 프레임워크내 구성요소들 간의 상호작용(interaction)을 UML 시퀀스 다이어그램으로 나타낸 것으로, 요청자가 UDDI QoS Broker를 통하여 원하는 웹서비스를 찾고, QoS의 변화로 인하여 웹서비스가 변경되는 과정을 보여준다.

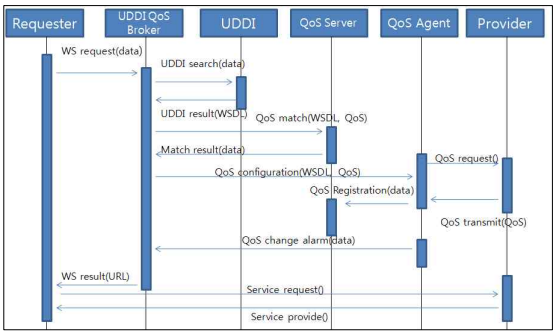

그림 3. 프레임워크의 실행 순서

2.2. 동적 웹서비스 변경 시 고려사항

본 논문에서 제안하는 QoS 지원 프레임워크에서 웹서비스를 동적으로 변경하는 경우 다음과 같은 사항을 고려해야 한다.

- 기존 웹서비스와 후보 웹서비스간의 인터페이스 적응
  - 웹서비스는 서비스 제공자가 정의한 메서드를 통하여 사용 가능하다. 요청자는 기존 웹서비스의 인터페이스 정보를 가지고 있고, 이 정보들을 후보 웹서비스의 인터페이스에 사용할 수 있어야 한다. 이로 인해, 요청자는 최소한의 입력으로 후보 웹서비스의 인터페이스를 요청자의 인터페이스에 적응할 수 있다.
- 사용 중인 웹서비스의 작업 보장
  - 새로운 웹서비스로 변경 시 기존 웹서비스를 통해 어떤 작업이 수행 중 일 경우, 새로운 웹서비스로 변경되어도 수행하던 작업을 연결시켜 진행 할 수 있어야 한다.

2.2.1. 웹서비스간의 인터페이스 적응

요청자와 제공자들 간의 서비스의 요청과 제공은 WSDL을 이용한다. 웹서비스의 메서드에 대한 정보는 요청자와 제공자가 이용하는 WSDL을 통하여 알 수 있다 [6][7] (그림4).

```
<?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="Reservation">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="send_name" type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="send_address"
                type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="send_post" type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="send_phonNumber"
                type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="recive_name"
                type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="recive_address"
                type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="recive_post" type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="recive_phonNumber"
                type="s:string" />
            </s:sequence>
          </s:complexType>
        </s:element>
      + <s:element name="ReservationResponse">
      </s:schema>
```

그림 4. 웹서비스의 WSDL문서

요청자가 가지고 있는 인터페이스 정보와 후보 웹서비스로부터 추출한 인터페이스를 대조하여 사용 중인 웹서비스의 인터페이스 정보들 중에서 후보 웹서비스의 인터페이스에 사용 가능한 파라미터들을 찾는다. 파라미터간의 매칭은 다음과 같은 순서로 한다. 단, 메서드에서 사용되는 파라미터의 이름이 파라미터가 받는 정보와 관계없는 숫자나 단어로 이루어진 경우를 제외한다.

① 후보 웹서비스의 WSDL내의 메서드와 파라미터 추출
② 후보 웹서비스의 메서드 파라미터 순서를 기준으로 요청자가 소유한 파라미터들과 이름의 유사성 매칭
③ 후보 웹서비스의 메서드 파라미터 순서를 기준으로 요청자가 소유한 파라미터들과 데이터타입 매칭
④ 후보 웹서비스의 메서드에 매칭된 파리미터 적용

메서드 정보의 이름에 대한 유사성 매칭은 다음의 예를

들어 설명한다. 도서 검색 서비스를 제공하는 두 웹서비스가 있으면, 이 두 웹서비스는 각각의 메서드를 가지고 있다. 이 메서드들은 모두 도서검색을 위한 메서드이므로 각 메서드의 정보는 책제목, 저자, 출판사, 출판년도, ISBN등의 정보를 요구할 것이다. 그렇다면 저자라는 정보를 받는 파라미터의 이름으로 쓰인 단어는 '이름', 'name'같은 단어를 쓰거나 같은 의미지만 다른 형태를 띠는 단어 혹은 유사단어를 포함한다. 이와 같은 단어를 전거사전(Authority File)을 통하여 단어의 유사성을 체크한다. 전거사전은 대표 단어와 전거 단어가 매핑 되어 있는 형태로 구성된다 [8]. 예를 들면 '차'라는 단어가 있다면 '승용차', '자가용'등의 단어들과 매핑 시켜서 모두 차라는 동일한 의미를 부여한다.

단어들의 유사성 체크가 끝나면 매칭된 단어들이 사용된 파라미터들의 데이터 타입을 매칭한다. 데이터 타입에 대한 매칭은 일반적으로 사용하는 프로그램언어의 데이터타입 변환방식을 사용한다. 데이터타입의 변환은 후보 웹서비스의 메서드 정보를 기준으로 변환한다. 예를 들어 사용 중인 메서드의 파라미터 타입이 integer이고, 후보 메서드의 파라미터 타입이 float라면 두 파라미터를 매칭을 하면 후보 파라미터를 기준으로 integer타입은 float타입으로 변환한다.

인터페이스 적응에 대한 구체적인 예를 3장에서 보인다.

2.2.2. 기존 작업의 보장

수행중인 작업의 보장을 위해 작업의 진행정도나 위치, 작업이 수행중인 데이터를 기록한다. 작업의 기록은 웹서비스의 동적 변경 시 Service Adapter의 Context File에 <범위, 진행위치, 데이터>형식으로 저장된다. 작업 중 웹서비스의 변경이 일어나면, Service Adapter는 Context File에 작업의 진행위치를 기록하고, 데이터는 필요한 경우 저장한다. 웹서비스 변경 후 Context File을 확인하여 변경된 웹서비스에 적용한다. 웹서비스의 적용 방법은 기존 작업을 수행하던 데이터의 진행위치를 웹서비스 변경 후 수행하는 작업범위의 시작으로 설정한다. 그러나 이것이 모든 유형의 작업에 다 적용되는 것은 아니다.

[9]에서는 웹서비스의 적용분야로 데이터 통합, 종합, 분석 서비스 제공 분야와 정보 제공 서비스 분야, 집중화 서비스 제공 분야, 어플리케이션의 통합과 확장, 피어(Peer) 및 푸시(Push) 기반 시스템의 예를 들고 있다.

본 논문에서는, 이들을 웹서비스의 서비스 제공형태를 기준으로 하여 다음과 같이 나누었다.

- 정보를 제공받는 작업
- 데이터 통합, 종합, 분석 등의 데이터 가공 작업
- 피어 및 푸시기반 작업

정보제공 작업의 일반적인 형태는 환율정보, 교통정보, 날씨정보 등과 같이, 제공자로부터 단순히 정보를 제공 받는 것이다. 이러한 정보는 동적 변경으로 인해 제공자가 바뀌어도 정보의 내용은 동일하므로 Context File을 통한 작업 보장이 가능하다.

데이터 가공 작업의 일반적인 형태는 문서번역, 호텔이나 극장 등의 예약, 인터넷결제 등과 같이, 제공자가 요청자로부터 데이터를 받아서 가공하여 요청자에게 다시 전달하거나 제공자에게 저장하는 형태의 작업을 말한다. 이 경우 데이터의 위치가 작업의 보장을 결정한다. 즉, 데이터가 요청자에게 위치할 경우는 작업의 보장을 받지만, 제공자에게 위치하는 경우는 작업의 보장을 받지 못한다. 이러한 두 가지 경우의 예로 번역서비스와 예약서비스가 있다.

번역서비스의 경우는 제공자가 요청자로부터 번역할 문서를 받아서 번역을 하고, 요청자에게 다시 전달한다. 이 번역서비스는 앞에서 말한 데이터가 요청자에게 위치하는 경우에 속한다. 이때, 동적 변경이 일어나게 되면, 번역할 문서는 Service Adapter를 통하여 진행 위치와 가공 중인 문서를 Context File에 저장하고 동적 변경 후 문서를 마지막으로 번역했던 위치에서 다시 번역하게 된다.

예약서비스의 경우는 제공자는 요청자로부터 예약에 관한 데이터를 제공받고, 그 데이터를 제공자의 시스템에 저장한다. 이것은 데이터가 제공자에게 위치하는 경우에 속한다. 이때, 동적 변경이 일어나게 되면, 예약에 관한 데이터를 저장해야할 제공자의 시스템이 바뀌게 된다. 호텔예약을 예로 들면, 요청자는 A호텔을 예약하였지만 동적 변경이 일어나 B호텔에 예약된 것이다. 이와 같은 경우는 작업의 보장으로 인해 요청자의 의도와는 다른 결과가 나온다.

피어 및 푸시기반 작업은 P2P기반의 파일 공유나 인터넷 방송 등과 같이 제공자를 통하여 데이터를 가지고 있는 시스템으로부터 데이터를 제공받는 피어기반 작업과, 소프트웨어의 업데이트나 실시간 뉴스 등과 같이 제공자가 요청자에게 단순히 정보를 보내거나 데이터를 보내어 요청자 쪽에서 프로세싱 하는 푸시기반 작업을 말한다.

피어기반 작업을 파일공유의 예로 들면, 요청자는 제공자를 통하여 찾고자 하는 파일을 검색하고, 검색한 파일을 요청하면 제공자는 동일한 데이터를 가지고 있는 시스템들을 요청자에게 연결시켜 준다. 이러한 경우, 동적 변경이 일어나도 시스템을 연결시켜준 제공자는 변경되지만, 데이터를 제공하는 시스템은 변하지 않는다. 따라서 데이터를 제공하는 시스

템과의 연결이 끊어지지 않는다면, 연속적인 데이터 제공을 받는다. 데이터를 제공하는 시스템들과의 연결이 모두 끊어진다면, Context File에 받고 있던 파일의 받은 부분을 기록하고, 파일을 저장한다. 시스템과 다시 연결이 되면 받았던 부분부터 다시 받는다.

푸시기반 작업을 소프트웨어 업데이트의 예로 들면, 요청자는 데이터를 제공자로부터 받아서 요청자 쪽에서 프로세싱한다. 제공자로부터 데이터를 모두 받았다면, 이 경우는 동적 변경이 일어나 제공자가 변경되어도 이미 받은 데이터를 요청자의 시스템에 프로세싱 하면 되므로 별다른 문제가 없다. 하지만, 제공자로부터 데이터를 받고 있던 경우는 Context File에 변경 전까지 받은 파일의 부분과 프로세싱 한 데이터를 저장한다. 동일한 데이터를 제공하는 제공자와 연결되면, 마지막 받은 부분부터 다시 받는다.

III. 동적 웹서비스 변경에 대한 사용 시나리오

본 장에서는 인터넷 도서검색 서비스의 예제를 통해 제안된 프레임워크의 실행 과정 중 동적 웹서비스 변경에 대한 메서드 적응부분의 예를 나타낸다. 인터넷 도서검색 서비스는 검색 키워드와 저자이름, 출판년도, ISBN을 사용자로부터 받아서 연동된 웹서비스로부터 도서를 검색하여 결과를 사용자에게 알려준다. 여기서 인터넷 도서검색 서비스는 요청자가 된다.

다음 시나리오는 UDDI QoS Broker가 요청자의 요구를 받아서 UDDI및 QoS Server를 통하여 웹서비스를 선택 후 사용 중이고, QoS의 동적 변경으로 인해 웹서비스를 변경하게 된 경우, 웹서비스의 인터페이스를 적응하는 과정이다.

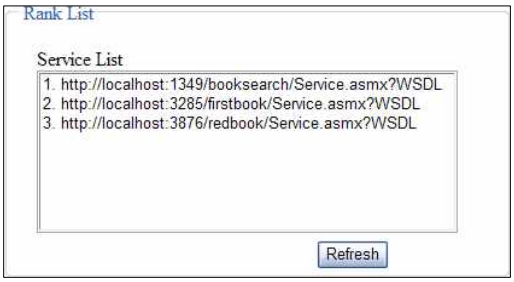

그림 5. QoS Server로부터 받은 랭킹 리스트

① 요청자의 UDDI QoS Broker는 도서검색 중에 연동된 웹서비스의 QoS가 변경되면, QoS Server로부터 랭킹 리스트를 다시 받아서 연동된 웹서비스의 순위를 확인하고 순위변화를 확인한다 (그림5).

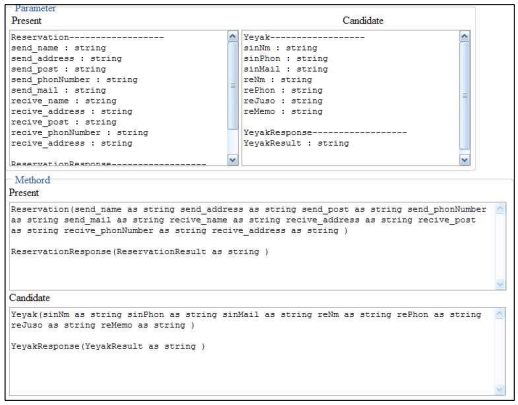

그림 6. WSDL의 파라미터 추출

② 순위변화가 없으면 연동된 웹서비스를 그대로 사용하고, 순위변화가 있으면 후보 웹서비스로 교체한다. 후보 웹서비스로 교체 시 UDDI QoS Broker는 연동된 웹서비스와 후보 웹서비스의 WSDL로부터 메서드의 정보를 추출한다 (그림6).

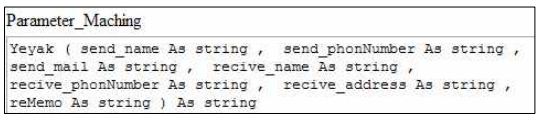

그림 7. 메서드 매칭 후의 후보 웹서비스의 메서드

③ 두 웹서비스의 메서드를 적응 한다. 적응순서는 각 파라미터의 이름에 대한 유사성을 매칭하고, 매칭 한 파라미터의 데이터 타입을 매칭 한다 (그림7). 앞의 과정에서 후보 웹서비스의 서비스를 이용하기 위한 메서드의 파라미터가 충분하지 않다면, 요청자에게 입력을 요청한다.

위의 적용 시나리오를 통하여, 후보 웹서비스의 인터페이스를 요청자의 인터페이스와 적응하는 과정을 보였다.

IV. 결론 및 향후 과제

본 논문에서는 UDDI와 웹서비스의 QoS를 적용하여 요청자가 요구하는 웹서비스를 찾고, 동적으로 변하는 웹서비스의 QoS로 인한 요청자의 QoS요구사항에 만족하지 못하게 된 웹서비스를 새로운 웹서비스로 변경하는 프레임워크를 제안

하였다. 그리고 동적인 웹서비스 변경으로 인해 발생하는 웹서비스간의 인터페이스 적응과, 사용 중인 웹서비스의 작업보장 문제에 대한 해결 방안도 제시하였다.

향후 과제로서, 동적 웹서비스 변경으로 인한 메서드의 파라미터 적응 시 파라미터의 이름을 나타내는 단어의 유사성 매칭을 보다 정확하게 하기 위해 온톨로지 기반의 의미적 분석을 통한 전거사전의 구현이 필요하다. 그리고 웹서비스 변경 시 작업보장에 대하여, 작업을 구체적으로 변경하고 Context File을 세부적으로 정의하며 구현할 것이다.

참고문헌

[1] Web Services Glossary, W3C Working draft, 2002, 6. http://www.w3.org/TR/ws-gloss/

[2] 박병호, 서형준, 임재혁, 유호동, “지능형 UDDI를 사용한 고가용성의 동적바인딩 기법”, 한국정보과학회, 학술발표논문집 제30권 제1호(B), pp. 16-18, 2003.

[3] M.A. Nbazhagan, N. Arun, “Understanding Quaility of Service for Web Services”, www.ibm.com/developerworks /library/ ws-quality.html, 2002.

[4] D.A Menasce, "QoS issues in Web Services", IEEE Internet Computing, Volume6, Issue6, pp.72-75, 2002.

[5] 김원상, 장희정, 이강선, “QoS 기반의 웹 프로세스 조합 방법론 및 지원도구의 개발” 한국시뮬레이션학회 2005년 춘계학술대회논문집, pp. 134-138, 2005.

[6] 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.

[7] 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.

[8] 김영수, 남택용, 원동호, “등급에 따른 웹 유해 문서 분류 기술”, 정보처리학회 논문집C, pp.859-864, 2006.

[9] 송은지, 한수영, “웹서비스 컴퓨팅”, pp.14-17, 인터비전, 2005.