RSS기반의 동적 QoS 조정을 지원하는 웹 서비스 매치메이킹

Web Service Matchmaking with RSS-Driven Dynamic QoS Adaptation

강필석(Pilseck Kang)*, 안철범(Chulbum Ahn)**, 서보원(Bowon Suh)*,
나연묵(Yunmook Nah)*

초 록

웹 서비스 기술의 급속한 발전과 더불어 동일한 기능의 서비스를 제공하는 웹 서비스 제공자들 또한 많이 증가 하였다. 따라서 동일한 기능을 제공하는 웹 서비스의 품질 정보는 서비스 매치메이킹에 중요한 고려 요소가 되었고, 웹 서비스의 품질 정보를 고려한 다양한 매치메이킹 시도가 있었다. 그러나 대부분 일반적인 웹 서비스 품질 정보를 반영하는데 그치고 각 서비스별로 다른 특징을 가지는 품질 정보는 고려하지 못하였다. 본 논문에서는 RSS 기법을 이용하여 동적으로 QoS를 반영하는 웹 서비스 매치메이킹 방법을 제안한다. 이를 위해 먼저 다양한 QoS 요소를 분류하고, 확장 가능한 QoS 모델에 대해 기술한다. 또한 제공된 QoS 요소들을 활용한 매치메이킹 기법에 대해 설명하고, RSS 기술을 이용하여 동적으로 사용자 제공 QoS를 추가하는 방법과 그 결과 매치메이킹 처리의 정확도를 높이는 방법을 소개한다. 끝으로 본 논문에서 제안한 방법을 구현하고 실험을 통해 그 유용성을 보였다.

ABSTRACT

With the rapid development of Web Service technology, the number of Web Service providers which provide same Web Services are increasing. As a result, the QoS elements become more and more important factors for Web Service matchmaking. But, most of previous research efforts do not consider various QoS elements including Service-specific QoS elements during the matchmaking process.

In this paper, we propose a Web Service matchmaking scheme which allow dynamic QoS adaptation using the RSS technique. We first classify various QoS elements and describe an extensible QoS model. We then explain a matchmaking scheme, which can utilize the proposed QoS elements. We also show how to dynamically add user-specific QoS elements using the RSS mechanism, thus improving the correctness of web service matchmaking process. Finally, we implemented the proposed schemes and showed the usefulness of our methods through some experiments.

키워드 : 웹 서비스, 매치메이커, QoS, RSS
Web Service, Matchmaker, QoS, RSS

이 연구는 2008년도 단국대학교 대학연구비 지원으로 연구되었음.

* 단국대학교 컴퓨터공학부
** 교신저자, 단국대학교 컴퓨터공학부

2009년 09월 23일 접수, 2009년 10월 07일 심사완료 후 2009년 11월 06일 게재확정.

1. 서 론

웹 서비스(Web Service)의 확산에 따라 동일한 기능을 제공하는 서비스 제공자 (Service Provider)들의 수도 자연히 늘어나게 되었다. 따라서 서비스 사용자(Service Requester)는 서비스의 홍수 속에 자신의 조건에 가장 적합한 서비스가 어떤 것인지, 또한 선택된 서비스가 사용자의 요구조건을 유지 할 수 있는지 알기 어렵다. 최근 들어서는 비 기능적(non-functional) 속성인 서비스 품질(QoS : Quality of Service)이 서비스의 성패를 구별하는 중요한 요소로 인식되고 있다[2].

기존의 웹 서비스 구조에서는 UDDI (Universal Description Discovery and Integration)를 통해 웹 서비스를 검색할 때, UDDI는 단지 서비스 설명 정보인 WSDL(Web Service Description Language)문서에 포함된 서비스의 기능적인(functional) 측면 만을 나열한다. 웹 서비스를 등록하는 UDDI와 웹서비스의 정보를 담고 있는 WSDL에는 서비스의 비 기능적인 측면인 서비스 품질은 전혀 고려되지 않는다. 따라서 서비스 사용자로부터 비 기능적인 측면이 고려되지 않은 서비스의 선택은 서비스 이용 시 처리 속도의 현저한 감소, 서비스 사용 가격의 상승으로 서비스 이용에 이익이 없을 수도 있으며, 최신의 정보를 반영하지 못하는 결과를 얻을 수도 있다.

본 논문에서는 웹 서비스의 QoS 정보를 웹 서비스의 기능에 맞게 보다 효율적으로 분류하고 웹서비스 사용자로부터의 새로운 QoS 요소에 대한 요구를 RSS(Really Simple Syndication) 방식을 통하여 서비스 제공자에게 제공하고, 서비스 제공자로부터 요청 결과를 반환 받아 서비스 매칭에 고려함으로써, 사용자의 요구를 동적으로 반영한 매치메이킹을 구현 하였다. 또한 UDDI에 등록된 서비스의 등록정보를 최신의 정보로 유지하고 기존의 비 기능적인 속성인 QoS를 고려한 매치메이커의 구조적 문제점을 해결하기 위하여 웹 서비스 등록 구조를 개선하고, 사용자의 새로운 QoS요구를 반영하여 동적인 서비스를 제공하였다.

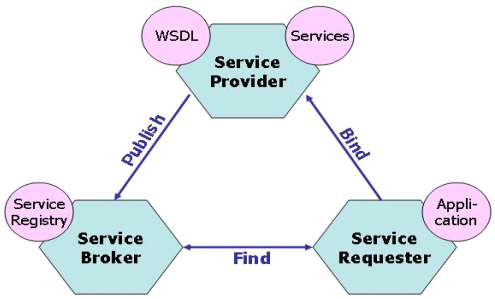
본 논문의 구성은 제 2장에서 웹 서비스와 매치메이커, QoS 웹 서비스 매이메이킹, RSS를 통한 컨텐츠 배포 방법에 대해서 살펴보고, 제 3장에서 효율적인 웹 서비스 QoS의 구성과 동적으로 사용자의 QoS 요구를 반영한 QoS 웹 서비스 매칭 및 랭킹에 대해 기술한다. 제 4장에서는 제안한 동적 QoS 매칭을 재현율과 정확률 측면에서 비교 실험 하여 QoS를 반영한 매치메이킹 결과의 유용성을 보인다. 끝으로 제 5장 에서 결론을 맺는다.

2. 관련 연구

2.1 웹 서비스와 매치메이커

웹 서비스는 분산 웹 환경에서 사용자가 원하는 정보를 동적으로 제공하기 위해 이기종 시스템 간의 상호운용을 가능하게 하는 XML 기반 기술이다. 현재 웹 서비스를 위한 표준으로는 SOAP, WSDL, UDDI가 있다.

SOAP(Simple Object Access Protocol)은 웹 서비스에서 XML 메시지를 주고받기 위한 W3C 표준 프로토콜이며[3] WSDL(Web Service Description Language)은 웹 서비스



〈그림 1〉 웹 서비스 아키텍처

를 기술하기 위한 언어이다. UDDI(Universal Description, Discovery and Integration)는 웹 서비스에 대한 디렉토리 서비스를 지원하는 분산 레지스트리 표준으로 웹 서비스의 등록과 검색을 위한 메커니즘을 제공한다[4].

<그림 1>은 웹 서비스 아키텍처를 보여준다. 서비스 제공자(Service Provider)는 자신이 제공하는 웹서비스를 UDDI 레지스트리에 등록한다. 서비스 요청자(Service Requester)는 자신이 원하는 서비스를 UDDI 레지스트리를 통해 검색한 뒤 적절한 웹 서비스를 발견하면 해당 서비스를 제공하는 서비스 제공자와 연결하여 서비스를 요청하게 된다. 서비스 중재자(Service Broker)는 서비스 제공자로부터 서비스를 등록받고 서비스 사용자에게는 서비스 검색을 위한 웹 서비스 정보 저장소인 UDDI 레지스트리를 운영 관리한다.

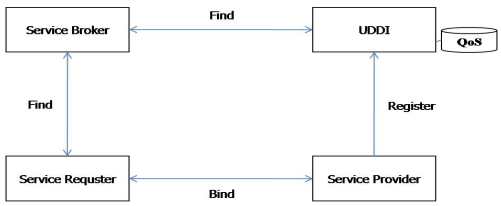
UDDI 레지스트리에 등록된 수많은 웹 서비스들 중에서 사용자가 원하는 최적의 서비스를 찾는 것을 '매치메이킹'이라고 하며 이러한 역할을 수행하는 에이전트를 '매치메이커(Matchmaker)' 또는 '매치메이킹 시스템'이라 한다[1]. 웹 서비스 아키텍처 관점에서 봤을 때 매치메이커는 서비스 중재자의 서브시스템에 속하게 된다. 즉, 서비스 요청자가 서비스에 대한 검색 요청을 하면 서비스 중재자는 매치메이커를 통해 적절한 서비스 목록을 찾게 된다. 최근 웹 서비스 매치메이킹을 위한 연구가 활발히 진행 중이며, 이러한 대표적인 연구로 DAML-S 매치메이커[5]와 OWL-S 매치메이커[6]가 있다.

2.2 QoS 정보를 반영한 웹 서비스 매칭

표준 웹 서비스 레지스트리는 등록된 웹 서비스의 QoS 정보를 저장하고 있지 않다. 따라서 기존 웹 서비스 매치메이킹은 서비스에 대한 QoS 정보를 반영하지 않고 매칭이 이루어지고 있으며 동일한 기능의 웹 서비스에 대해서 사용자의 비 기능적 측면인 QoS에 대한 요구를 반영 할 수 없다. 따라서 사용자의 기능적 요구와 QoS요구에 맞는 최적의 서비스를 제공하기 위해서 QoS를 반영한 매치메이킹이 연구 되었다[7].

동일한 기능을 제공하는 웹 서비스 중에서 웹 서비스의 비 기능적인 측면인 사용자의 QoS 요구를 반영하기 위해서 QoS 브로커를 활용한 매치메이킹이 연구 되었다. <그림 2>는 웹 서비스 브로커 시스템의 구조 이다.

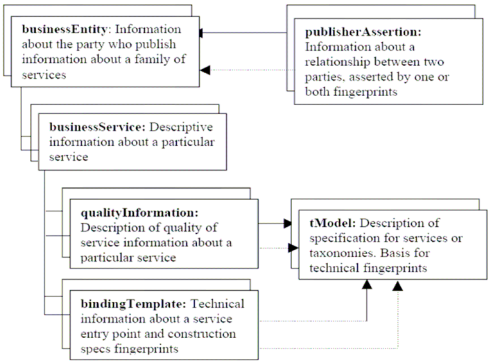
표준 UDDI레지스트리는 QoS 정보를 저장하지 않기 때문에 QoS 정보를 포함 하는 새로운 UDDI레지스트리를 제안 하였다[8]. 제



〈그림 2〉 QoS 브로커 시스템 구조

안된 모델에서는 QoS 정보는 tModel 형태로 저장하게 된다. 웹 서비스 브로커는 UDDI에 포함된 서비스 등록정보와 각 서비스 제공자들의 서비스에 대한 QoS 정보를 포함하고 있으며 서비스 사용자의 서비스 요청이 발생하면, UDDI에 저장 되어 있는 WSDL정보와 브로커의 QoS 정보를 포함하여 서비스 매칭을 실시하고 그 결과를 반환 한다[9, 10]. QoS 웹 서비스 매칭을 위해서 표준 UDDI 레지스트리의 데이터 구조를 확장 하였다. 확장된 데이타 구조에는 quility Information 정보를 포함하게 된다. 제안한 데이터 구조는 <그림 3>과 같다. 현재의 웹 서비스 구조는 QoS 요소에 대한 정의를 가지고 있지 않다.

확장된 UDDI의 데이터 구조에서는 다음과 같은 QoS 정보를 포함 하고 있으며, tModel을 참조 하여 기술 하게 된다. 현재 표준 UDDI레지스트리에서 tModel은 서비스의 기술적 정보(technical Information)을 설명하고 있다. tModel의 구성은 key, name, 추가설명, URL(Uniform Resource Locator)을 담고 있다. 현재 tModel은 2



〈그림 3〉 QoS 정보를 포함하는 확장된 UDDI 데이타 구조

가지의 역할을 수행 하고 있으며, 첫째는 웹 서비스의 기술적 명세를 설명하고, 둘째는 서비스 분류 정보를 등록하는데 사용한다.

2.3 RSS

RSS는 Really Simple Syndication의 약자로 간단한 정보 조각을 전달하기 위해 쓰이는 XML기반의 데이터 표준 포맷이다[11]. RSS로 포멧팅된 정보는 RSS 명세에 따른 구조에 맞추어 정보의 의미를 담게 된다. 표준 포멧인 XML을 사용하기 때문에, RSS로 만들어진 모든 정보는 같은 방법으로 구조화되어 있으며, 따라서 이 구조를 이해하는 모든 컴퓨터가 가져다가 쉽게 처리하여 서비스할 수 있다. 기존의 정보의 배포는 사용자가 직접 관련 페이지에 접속하여 정보를 획득하는 방식이였다. 하지만 RSS를 이용하면 RSS 수집기에 관심이 있는 정보가 있는 페이지의 URL을 등록해 두면 페이지의 내용이 갱신될 때 마다 RSS 수집기가 정보를 수집해 알려 주는 역할을 한다. 메일링 서비스와의 큰 차이점은 사용자의 관심 내용만 수집을 한다는 점이다.

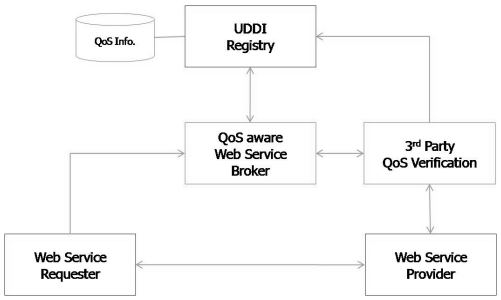
3. 사용자 QoS를 반영한 웹서비스 매치메이킹

3.1 QoS 반영 웹서비스 아키텍처

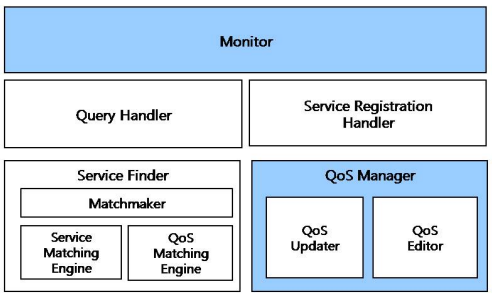
본 논문에서는 기존의 웹 서비스 구성 요

소인 UDDI 레지스트리, 서비스 제공자, 서비스 사용자에 2가지 역할을 추가 한다. 첫 번째는 QoS 브로커로서 사용자 요구인 웹 서비스의 기능적인 측면과 비 기능적인 측면 모두를 반영한 매치메이킹을 수행 하고, 두 번째는 각 디렉토리의 웹 서비스별 QoS 요소의 신뢰성 있는 검증을 위해서 인증된 제 3기관을 두어 서비스 제공자의 QoS 정보에 대한 검증을 수행하여 등록된 서비스정보의 신뢰도를 향상 시키고 구조화된 QoS 정보의 등록으로 서비스 매칭시 효율적이고 정확한 매치메이킹을 가능하게 한다. <그림 4>는 제안 시스템 구조를 보여 준다. 또한 UDDI 레지스트리에는 tModel을 사용하여 서비스 도메인별 QoS 정보를 저장한다.

<그림 5>는 전체 적인 QoS 브로커의 구조를 보여 준다. 이 구조에는 Query Handler가 사용자 질의를 분석하여 요구하는 웹 서비스의 기능적인 측면과 비 기능적인 측면을 분리 하여 Service Finder에게 서비스 검색을 요청한다. 이때 Monitor는 사용자의 서비스 질의 내용을 분석하고 새로운 QoS요소의 질의가 요청 되었을 때 관련 서비스 제공자에게 요청 내용을 RSS형식으로 배포 하고 수집하여 서비스 매치메이킹에 사용한다. 분



〈그림 4〉 Qos 반영 웹서비스 아키텍처



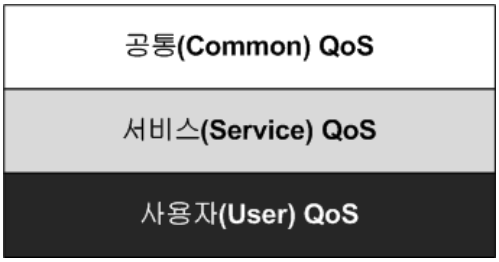
〈그림 5〉 QoS서비스 브로커의 구조

리된 사용자의 질의는 Service Finder로부터 저장되어 있는 서비스 등록 정보와 QoS 정보를 매칭하여 매칭된 결과 리스트를 반환 한다.

QoS Manager는 웹 서비스의 도메인별 QoS 정보를 관리 한다. QoS Updater는 등록된 서비스의 QoS 정보를 관리하고 갱신한다. 서비스 제공자 또한 tModelKey에 의해서 접근하여 갱신 할 수 있다. QoS Editor는 분류된 서비스 도메인별 QoS 및 사용자 QoS의 효율적인 매칭을 위한 조정 역할을 수행 한다. Service Registration handler는 서비스 제공자의 서비스 등록 정보와 QoS 정보가 구조화된 형식을 갖추었는지 검사 하여 제한적인 서비스 등록을 수행하며, QoS 정보의 갱신 시 서비스 제공자로부터의 서비스 정보 갱신을 검사 하게 된다.

3.2 QoS 모델

본 논문에서 제안하는 웹 서비스의 QoS정보는 <그림 6>과 같이 3가지 타입의 정보로 분류 된다. QoS 정보는 공통(Common) QoS, 서비스(Service) QoS, 사용자(User) QoS 정보로 나뉘어 관리 되며, 공통 QoS는 모든 웹 서비스가 갖추어야 할 필수 QoS 요소로 구



〈그림 6〉 QoS 분류 모델

성된다. 서비스 QoS는 서비스 도메인별로 갖추어야 할 QoS 요소로 구성되며, 사용자 QoS는 사용자가 요청하는 다양한 QoS 요소가 정의되어 있다.

각 서비스의 QoS 정보는 표준 UDDI 레지스트리의 tModel을 사용하여 각 웹 서비스의 QoS 정보를 저장 하게 된다. 서비스 제공자가 서비스 등록 시 tModel을 생성하여 등록하려는 웹 서비스의 QoS 정보를 등록하게 된다. 등록이 완료 되면 tModelKey로 등록된 웹 서비스의 QoS 정보 갱신을 수행 할 수 있다. 단 각 QoS 요소들은 본 논문에서 제시하는 qosType정보를 가지고 있다.

신뢰성 있는 웹 서비스의 QoS의 측정을 위해서 본 논문에서는 서비스별로 구조화된 QoS 정보를 브로커 시스템이 제공하고, 인증된 제 3기관이 검증하여 서비스 제공자들은 자신의 웹 서비스를 UDDI 레지스트리에 등록하게 된다. <표 1>은 의료 동영상 정보 제공 서비스의 QoS 요소를 명세화 하였다.

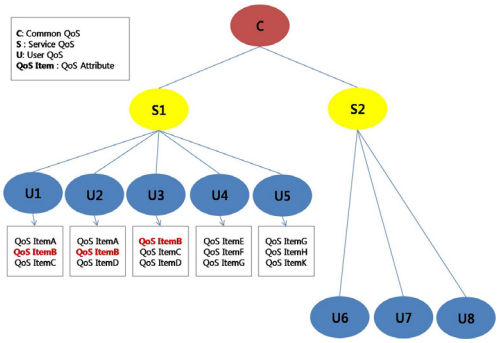
〈표 1〉 의료 동영상 제공 서비스의 QoS 요소 분류

공통 QoS	Availability, cost, Interoperability
서비스 QoS	time, reliability, capacity, robustness, data rate, streaming delay
사용자 QoS	number of media, media resolution

3.3 QoS 모델과 저장

현재 UDDI 레지스트리의 서비스 갱신은 서비스 제공자로부터의 자발적인 갱신이 이루어 지지 않으면, 등록 초기에 저장 된 서비스 정보를 사용하여 매치메이킹이 이루어지기 때문에 정보의 최신성을 유지하기 어렵다. 서비스 정보와 QoS 정보의 갱신 또한 일정 주기를 통한 전체 서비스 갱신 방법과 서비스별 타임 스템프 방식으로 갱신을 수행하고 있어 서비스 요청 시점에서의 정확한 서비스 정보를 바탕으로 한 매치메이킹이 이루어 지지 않는다. 본 논문에서는 이러한 문제점을 개선하고 효율적인 QoS의 갱신 메커니즘을 통해서 사용자의 요구를 모두 반영할 수 있는 갱신 매커니즘을 제안한다. <그림 7>은 서비스 도메인별 QoS 항목들의 저장을 트리 형태로 구성하였다.

웹 서비스의 QoS 정보는 3가지 타입으로 분류 되어 있으며, <그림 7>과 같은 트리



〈그림 7〉 QoS 정보 트리

형태의 계층구조를 가진다. S1을 동영상 정보 제공 도메인을 가진 서비스 특유의 QoS라고 가정하자. S1에 대한 하위 서비스 도메인으로 U1는 의료 정보 동영상 서비스, U2는 여행정보 동영상 서비스, U3는 교육자료 동영상 제공 서비스, U4는 영화자료 동영상 제공 서비스, U5는 뮤직비디오자료 제공 서비스라고 하고 다음과 같은 QoS 정보를 각각 포함 하게 된다. 동영상 제공 서비스의 공통 QoS요소를 Availability, Cost, Interoperability라고 하고 서비스 특유의 QoS 요소를 Time, Capacity, Data Rate, Streaming Delay라고 하면 <표 2>과 같이 서비스의 QoS 요소를 정리 할 수 있다.

S1은 5개의 User specific QoS 노드(node)를 가지고 있다. 각 노드는 사용자로부터 요청 받은 QoS요소를 포함 하고 있으며, 그 중 QoSItemB는 S1하위 노드인 U1, U2, U3에 포함 되어 있어, S1하위노드의 과반수이상을 차지하고 있어 상위 노드인 S1에 포함하게 된다.

서비스 도메인별 QoS의 갱신은 QoS 에디터에 의해서 수행이 되며, 향후 동영상정보를 제공하는 서비스 제공자들은 Common QoS와 QoSItemB가 포함된 Service specific QoS에 대한 정보를 포함하여 서비스를 등록하게 된다.

본 논문에서는 서비스의 QoS를 매칭하기 위해서 서비스의 QoS 값을 조정하여 저장하게 된다. 점수의 범위는 0과 100사이의 값을 가지며, QoS의 최대값은 100의 값을 가지게 된다. <표 4>은 QoS값 조정 내역이다.

<표 5>는 서비스 도메인 QoS측정값을 표시하였고 <표 6>는 기준에 따라 조정된 QoS의 값을 나타낸다. QoSItem1의 경우는 값은 범위가 양수의 범위이고 최대값이 1500이므로

〈표 2〉 동영상정보 제공 서비스의 QoS

Common QoS : C = {Availability, Cost, Interoperability}
Service specific QoS : S1= {Time, Capacity, Data Rate, Streaming Delay}
User specific QoS1의료정보 서비스 = U1{QosItemA, QosItemB, QoSItemC}
User specific QoS2여행정보 서비스 = U2{QosItemA, QosItemB, QoSItemD}
User specific QoS3교육자료 서비스 = U3{QosItemB, QosItemC, QoSItemD}
User specific QoS4영화자료 서비스 = U4{QosItemB, QosItemF, QoSItemG}
User specific QoS5뮤직비디오 서비스 = U5{QosItemG, QosItemH, QoSItemK}

〈표 3〉 갱신된 동영상정보 제공 서비스의 QoS

Common QoS : C = {Availability, Cost, Interoperability}
Service specific QoS : S1= {Time, Capacity, Data Rate, Streaming Delay, QoSItemB}
의료정보 서비스 = {D, S1, U1{QosItemA, QoSItemC}}
여행정보 서비스 = {D, S1, U2{QosItemA, QoSItemD}}
교육자료 서비스 = {D, S1, U3{ QosItemC, QoSItemD}}
영화자료 서비스 = {D, S1, U4{QosItemB, QosItemF, QoSItemG}}
뮤직비디오 서비스 = {D, S1, U5{QosItemG, QosItemH, QoSItemK}}

〈표 4〉 QoS 값의 범위 조정

1. True/False는 각각100과 0의 값을 가진다.
2. QoS요소의 값이 낮을수록 좋은
   QoS요소= 100-QoSscore
3. QoS요소의 값이 높을수록 좋은
   QoS요소= QoSscore

〈표 5〉 측정 QoS 값

	QoSItem 1	QoSItem 2	QoSItem 3	QoSItem 4
Service 1	1500	10	True	0.1
Servcie 2	1000	-5	True	0.5
Service 3	500	-10	False	1.0
Service 4	1200	5	False	5
Service 5	100	20	True	0.5

〈표 6〉 조정된 QoS 값

	QoSItem 1	QoSItem 2	QoSItem 3	QoSItem 4
Service 1	100	68	100	2(98)
Servcie 2	67	15	100	10(90)
Service 3	33	0	0	20(80)
Service 4	80	50	0	84(16)
Service 5	7	100	100	9(91)

변경된 기준값 1은 15를 나타낸다. QoSItem2의 경우는 값의 범위가 음수를 포함하고 있고 최대값이 20이고 최소값이 -10이므로 변경된 기준값 1은 0.3을 나타낸다.

QoSItem3의 경우는 True/False로 표현이 되므로 True는 100 False는 0을 나타낸다. QoSItem4 의 기준값 1은 0.06 이지만 낮은 값이 좋은 요소로 분류되어 있다고 가정하여 괄호안의 값으로 조정된다.

〈표 7〉 서비스 도메인의 QoS점수의 합

	Sum of QoSscore	TotalQoSscore
Service1	366	91.5
Service2	272	68
Service3	53	13.3
Servcie4	214	53.5
Servcie5	298	74.5

<표 7>에서 총 QoS 점수(TotalQoSscore)는 전체QoS점수의 합을 QoSItem의 수로 나눈값을 사용한다.

3.4 사용자 QoS를 반영한 동적 서비스 발견

사용자로부터의 웹 서비스 요청은 기능적인 측면과 비 기능적인 측면의 서비스 요청으로 분리하여, QoS 서비스 브로커가 UDDI에 등록된 WSDL의 서비스 정보와 QoS정보를 반영하여 매칭을 수행한다. 본 논문에서는 다양한 사용자의 QoS 요구를 반영하기 위해서 효율적인 QoS 저장 구조를 제시하였다. 하지만 사용자로부터의 다양한 QoS 요구는 UDDI 레지스트리에 저장되어 있지 않을 수도 있다. 따라서 동적으로 사용자의 QoS 요구를 반영하기 위해서 서비스 브로커의 모니터시스템은 사용자 질의를 분석하여 새로운 QoS 요구가 발생될 경우 RSS를 통하여 서비스

제공자에게 사용자의 요구사항을 효율적으로 전달하고 서비스 제공자로부터의 갱신된 서비스 정보를 반영한 매치메이킹을 수행함으로써 사용자의 요구를 모두 반영할 수 있으며, 요청된 내역에 대한 서비스 정보를 갱신함으로써 UDDI에 등록된 서비스 정보를 최신의 정보로 유지 할 수 있다. <표 8>은 의료정보 서비스의 사용자 QoS로 데이터암호화(data encryption)를 반영하기 위한 시나리오이다.

서비스 사용자로부터의 새로운 서비스 QoS에 대한 요구 사항을 보다 동적으로 반영하기 위해서 제안한 구조의 모니터 시스템은 사용자의 요구가 있을 경우 매칭된 서비스 제공자에게 RSS 형식으로 서비스 사용자의 요구사항을 전달하고 새로운 QoS요소에 대한 서비스 제공자로부터의 WSDL 문서 갱신 및 QoS 정보 갱신에 의한 매치메이킹 결과를 반환하여 서비스 사용자의 요구를 반영한 결과를 통보한다. <그림 8>은 사용자 QoS 반영을 위한 전체 시퀀스 다이어그램을 나타낸다.

3.5 서비스 매치메이킹 알고리즘

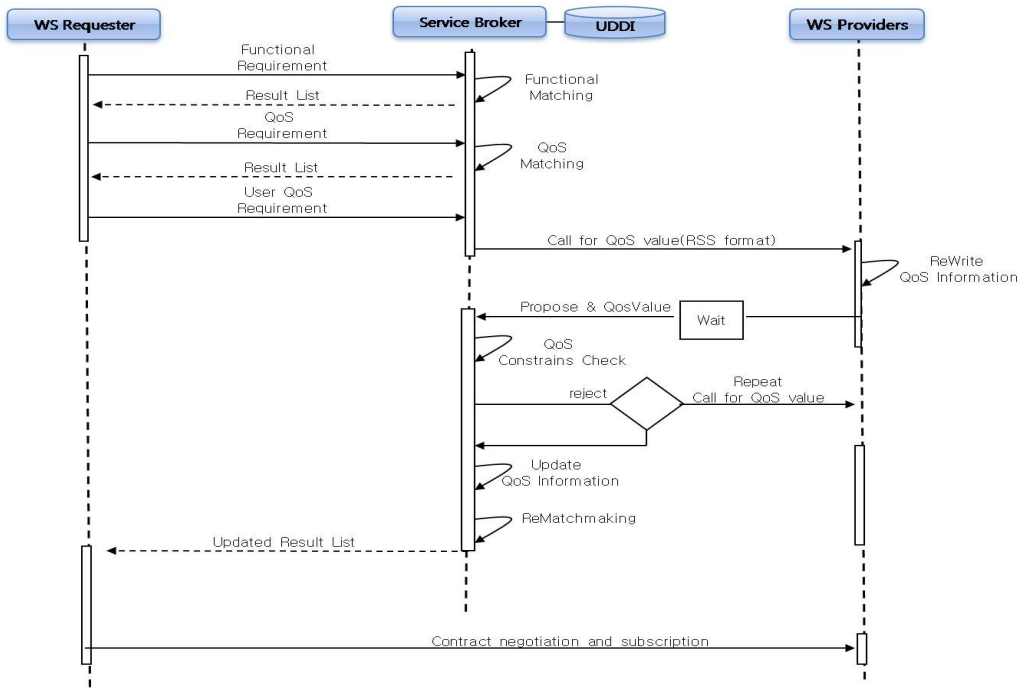
사용자의 요구를 반영한 매치메이킹을 수행 하기 위해서 QoS 브로커 시스템은 서비스의 기능적 측면과 비 기능적 측면을 고려한 매치메이킹을 수행 한다. <그림 9>는 서비스 발견 프로세서를 보여 준다.

그림에서 보듯이 매치메이킹은 3가지 과정으로 이루어진다. 첫번째는 기능적(functional) 매칭으로 UDDI 레지스트리로부터 서비스 도메인에 해당하는 모든 서비스에 대한 검색을 수행하여 이를 목록으로 작성하는 과정이며, 두 번째는 작성된 목록을 바탕으로 각 서비스들의 QoS를 평가하여 그 평가 점수가 좋은 순서대로 매칭하는 비 기능적(non-functional) 매칭으로 이루어지며, 마지막으로 사용자로부터의 QoS 요소인 사용자 QoS에 대한 매칭을 수행 하여 매칭 결과를 정렬하는 서비스 랭킹으로 매치메이킹을 완료한다.

<표 9>는 전체 매치메이킹 알고리즘을 보여

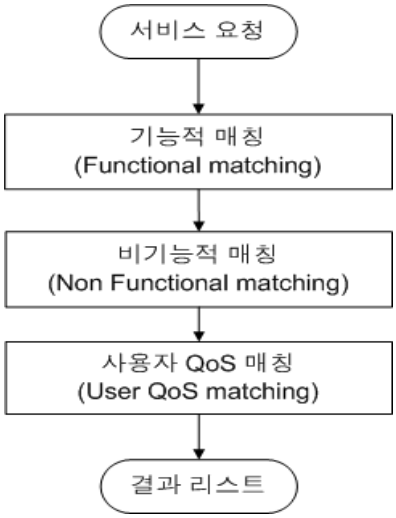
〈표 8〉 의료정보 동영상 서비스의 사용자 QoS 반영 시나리오

1	서비스 요청자는 의료 정보 동영상 서비스검색을 QoS 서비스 브로커에게 요청 한다.
2	서비스 브로커는 의료 정보 서비스를 제공하는 서비스 제공자의 QoS 측면을 고려한 매치메이킹을 수행하고 그 결과를 반환한다.
3	서비스 요청자는 새로운 QoS 요소인 데이터 암호화(data encryption)에 대한 요청을 한다.
4	서비스 브로커의 모니터는 data encryption에 대한 User specific QoS를 QoS정보에 포함하고 의료정보 서비스를 제공하는 서비스 제공자에게 RSS형식으로 관련 내용을 배포하게 된다.
5	RSS Reader를 사용하고 있는 서비스 제공자는 서비스 브로커로부터 내역을 통지 받고 자신의 QoS 정보를 Data Encryption를 반영하여 재 작성하여 QoS 정보를 갱신한다.
6	서비스 브로커는 서비스 제공자들로부터 갱신된 QoS 정보를 바탕으로 서비스 매치메이킹을 수행하여 결과를 통지한다.
7	동영상 제공 서비스를 제공하는 서비스 제공자들은 새롭게 등록된 data encryption에 대한 User specific QoS가 생성된다.



〈그림 8〉 동적 웹 서비스 요청 시퀀스 다이어그램

준다. fmatch 는 서비스의 기능적인 측면을 고려한 매칭을 수행 한다. qosmatch는 서비스의 비 기능적인 측면을 고려한 매칭을 수행 한다. qosRank는 매칭된 서비스의QoS점수를 계산하고 순위에 따라서 결과를 반환한다.



〈그림 9〉 서비스 발견 프로세서

userqosRank는 User specific QoS를 반영하여 QoS점수를 계산하고 순위에 따라서 결과를 반환한다. selectService는 사용자가 요청한 서비스의 수만큼 매칭 결과리스트를 반환하여 준다. userqosRank는 User specific QoS를 포함한 매치메이킹을 수행 한다.

<표 10>과 <표 11>은 QoS 매칭과 랭킹 알고리즘을 보여준다. getServiceQoS는 UDDI 레지스트리에 저장 되어 있는 QoS 정보를 가지고 온다. qosMatchAdvert는 서비스의 등록된 QoS 값을 찾아낸다. calculateQoSScore는 QoS 점수를 계산한다. sortByQoSScore는 서비스를 QoS점수에 따라서 정열 한다.

〈표 9〉 매치메이킹 알고리즘

```
/*Web Service matching, ranking and selection algoritm*/
findService(FunctionalRequirements, QoSRequirements, UserQoSRequirement, maxNum){
//1. Find Service that meet the FunctionalRequirements
FMATCH = fmatch(FunctionalRequirements)
//2. Match service with QoS information
QoSMATCH = qosmatch(fmatch,QoSRequirements)
// 3. User specific QoS match
If User QoS requirements specified
MATCHS= userqosRank(qosmatch,QoSRequirements,UserQoSRequirement)
Reture selectServices(MATCHS,maxNum);
Else
//rank MATCHS with QoS information
MATCHS = qosRank(QoSMATCH,QoSRequriements);
// Select Service according the maxNum of Service to be returned
Return selectService(MATCHS,maxNum);
}
```

〈표 10〉 QoS 매칭 알고리즘

```
// find services that match QoS requirements
qosmatch (services, qosReqt) {
MATCHS = Service [];
for each s in services
// get QoS info from UDDI
qosAds = getServiceQoS(s);
// if QoS information available and satisfies QoS requirements
if qosAds != null && qosMatchAdvert(qosAds,qosReqt)
MATCHS.add(s);
end for
return MATCHS;
}
```

〈표 11〉 QoS 랭킹 알고리즘

```
// rank matches with QoS information
qosRank(service,qosReqt)
//calculate QoS Score
services = calculateQoSScore(service,qosRequt);
// sort the result by QoS score in descending order
 services = sortByQoSScore(services);
return service;
}
```

4. 구현 환경 및 성능 평가

4.1 구현 환경

본 논문에서 제안한 QoS브로커 시스템은 <표 12>와 같은 개발 환경에서 구현되었다. 웹 서비스의 기능적 정보와 QoS 정보를 저장하고 있는UDDI 레지스트리는 마이크로 소프트의 UDDI 레지스트리를 이용하여 구현하였고 QoS 브로커 시스템은 ASP.NET을 사용하였다. 또한 서비스 제공자의 RSS를 수집하기 위해 아웃룩 2003을 사용하였다.

〈표 12〉 동적 QoS 브로커의 구성

CPU	Intel Pentium 4/2.8Ghz
RAM	1 Gbytes
운영 체제	Microsoft Windows Server 2003 Enterprise Edition
UDDI 종류	Microsoft UDDI version 5.2
구현 언어	Microsoft ASP.NET

4.2 성능 평가

본 논문에서는 제 3장에서의 “의료정보 동영상 제공 서비스(medical information video contents)”를 검색하는 시나리오를 바탕으로 질의를 수행한다. 첫 번째 단계는 웹 서비스의 기능적 측면을 반영한 질의로 “의료정보 동영상제공 서비스 검색” 과정이다. 두 번째 단계는 검색된 서비스 제공자의 QoS정보 중에서

〈표 13〉 매치메이킹 실험을 위해 설정된 서비스 제공자의 QoS정보

	QoS	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Common QoS	Availability	100	100	100	100	100	100	100	100	0	0
	Cost	90	90	50	50	30	50	10	90	60	60
	Interoperation	90	90	80	80	85	85	85	85	70	50
	Capacity	90	90	90	50	80	50	80	80	30	30
Service specific QoS	Accuracy	100	100	70	70	90	50	70	80	50	50
	Robustness	90	90	80	90	60	80	70	50	80	40
	Reliability	50	80	70	70	60	80	70	50	85	85
	Response Time	90	90	100	70	40	70	85	30	85	85
User specific QoS	Resolution	80	60	80	80	80	100	80	30	10	10
	Compression Rate	30	50	50	60	100	40	50	90	90	30
	Reputation	100	100	100	100	100	0	0	0	50	80
	Num of Media	90	80	90	80	30	80	80	0	50	90
new QoS	Streaming Delay	90	70	90	30	50	90	0	0	30	80
	Information Loss	90	70	50	90	80	0	0	0	0	0
	Exception handling	80	80	30	30	0	0	0	90	40	20
	data encryption	60	100	10	100	70	30	90	100	0	100

“해상도가 좋은 서비스 검색”을 수행 하였다. 마지막으로 “데이터암호화 지원여부”에 대한 새로운 User QoS를 반영한 매치메이킹 결과를 반환하게 된다. 웹 서비스의 검색에 대한 평가 척도로서 재현율(recall)과 정확율(precision)을 측정하여 비교하였다.

구현된 매치메이커가 정확히 동작하는지 실험하기 위하여 <표 13>과 같은 의료 정보 동영상 제공 서비스의 QoS요소 값들을 가상으로 설정하여 서비스 레지스트리에 저장한 후 서비스의 매치메이킹 실험을 수행하였다.

실험에서 정한 의료정보 동영상 제공 서비스 제공자는 P1부터 P10까지 총 10개로 지정하였으며, 표에서 보이는 것과 같이 P1부터 P4는 일반적인 서비스의 모든 QoS 요소 값을 우수하게 설정하였고, P5부터 P8은 User QoS 값은 조정하였으며, 시각 미디어 서비스의 공통적인 QoS 요소 값을 우수하게 설정하였으며 P9와 P10은 Common QoS인 Availability를 값을 “false”으로 설정 하였다.

<표 14>는 <표 13>에 설정된 정보들을 바탕으로 수행하는 5 가지 실험에 대한 설정으로 실험 1은 의료 정보 동영상 제공 서비스의 기능적 측면만을 고려한 매치메이킹을 수행하였고, 실험2는 공통 QoS와 서비스 QoS를 반영한 일반적인 QoS 매치메이킹을 수행하였고, 실험 3는 실험 2의 QoS 요소 중 사용자로부터의 “해상도” 요소에 가중치를 변경하여 실험 하였다. 실험 4와 5는 사용자 QoS를 반영한 매치메이킹을 수행하였고, 실험4는 저장된 사용자 QoS 중 “미디어 수”를 고려한 매치메이킹을 수행하였다. 마지막으로 실험 5는 새로운 사용자 QoS인 “데이타암호화”를 반영한 매치메이킹을 실시 하였다.

이렇게 설정된 질의 정보를 바탕으로 5가지 매치메이킹 실험을 수행하였다. <표 15>는 각각의 실험 결과를 보여 주고 있다.

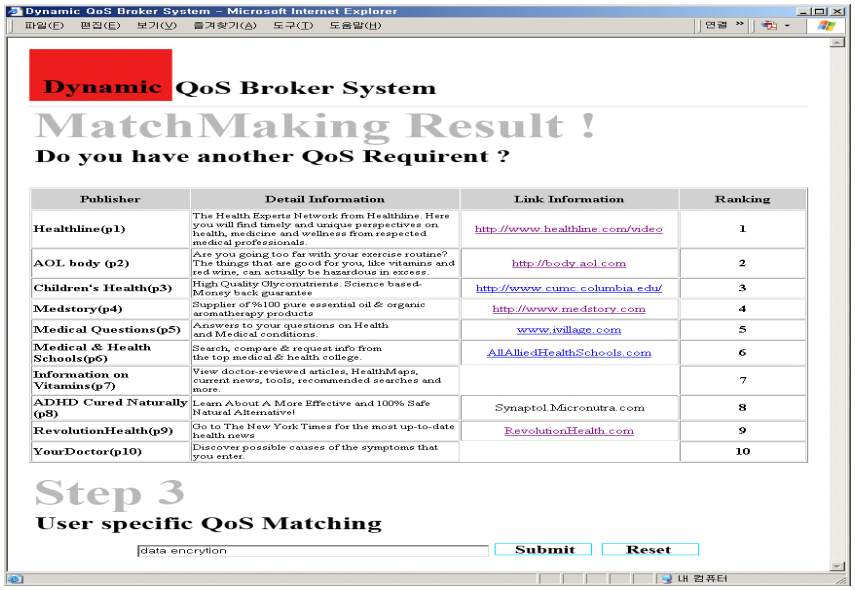
실험 결과 5가지 실험 설정에 대하여 서비스 제공자의 순위가 다르게 나타나는 것을 볼 수 있다. 실험 1의 경우는 일반적인 웹 서비스의 매치메이킹으로 기능적 측면의 매칭을 수행한 결과로, 웹 서비스 정보를 담고 있는 WSDL의 문서를 검색하여 얻은 결과를 나타낸다. 실험 2의 경우는 서비스의 QoS 정보를 반영한 매치메이킹으로 P9와 P10은 Common QoS의 값이 없어 매칭 결과에서 제외 되었다. 실험 3의 경우는 사용자로부터 “해상도” 값에 가중치를 부여 받아 매치메이킹 결과를 보여 준다. 사용자로부터의 중요한 QoS 요소에 가중치를 부여 함으로서 실험 2에서의 결

〈표 14〉 매치메이킹 실험을 위한 5가지 설정

실험	실험 내용
실험 1	서비스 기능만을 반영한 매치메이킹
실험 2	공통 QoS와 서비스 QoS를 고려한 매치메이킹
실험 3	실험 1에 QoS 요소 가중치를 고려한 매치메이킹
실험 4	사용자 QoS를 고려한 매치메이킹
실험 5	새로운 사용자 QoS를 고려한 매치메이킹

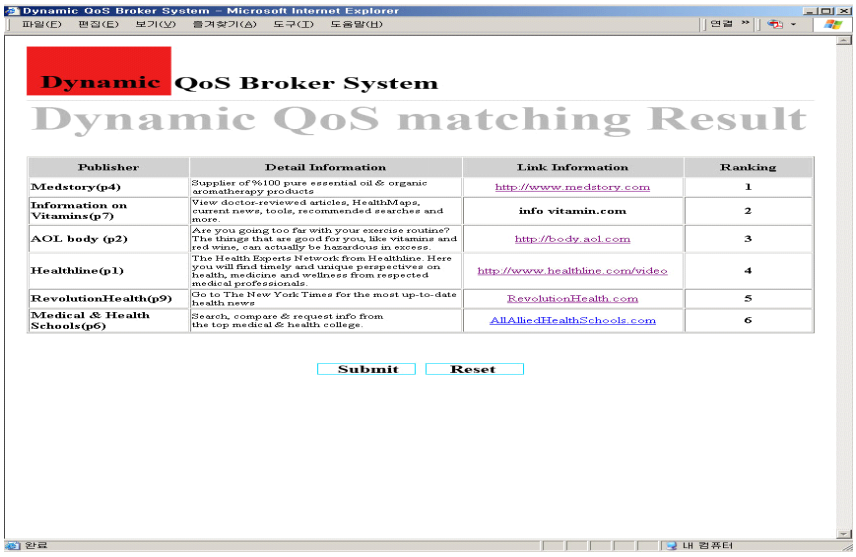
〈표 15〉 매치메이킹 실험 결과

순위	실험 1		실험 2		실험 3		실험 4		실험 5	
	서비스 제공자	랭킹 점수	서비스 제공자	랭킹 점수	서비스 제공자	랭킹 점수	서비스 제공자	랭킹 점수	서비스 제공자	랭킹 점수
1	P1	100	P2	840	P1	420	P1	1260	P2	1320
2	P2	100	P1	810	P2	405	P2	1240	P1	1340
3	P3	100	P3	770	P3	385	P3	1130	P4	1140
4	P4	100	P5	725	P5	360	P4	1050	P3	1150
5	P5	100	P4	720	P4	363	P5	985	P5	1055
6	P6	100	P6	705	P5	353	P6	875	P6	905
7	P7	100	P7	700	P6	350	P7	780	P8	870
8	P8	100	P8	685	P7	342.5	P8	775	P7	875
9	P9	100								
10	P10	100								



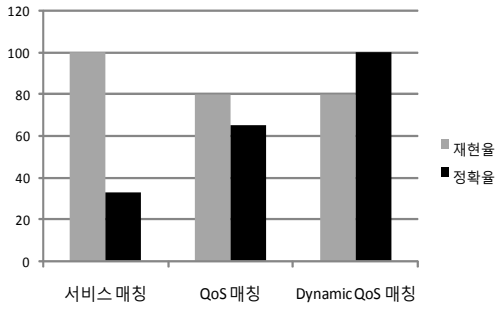
〈그림 10〉 QoS 매칭 결과

과와 다른 순위를 보여준다. QoS 매치메이킹의 장점은 사용자가 중요하게 생각하는 QoS 요소의 가중치를 높게 설정하여 그 결과를 반환 받는다. 실험 4의 경우는 서비스의 필수 QoS는 아니지만 서비스에 저장된 User QoS의 요청 내역을 반영한 결과이다. 실험 5의 경우는 본 논문에서 제안한 새로운 QoS 요소에 대한 매치메이킹 결과를 보여 주고 있다.



〈그림 11〉 암호화에 대한 QoS 결과

실험 결과를 통해 본 논문에서 제안하고 구축한 사용자의 QoS 요구를 동적으로 반영한 브로커 시스템이 사용자의 요구에 대한 매치메이킹을 올바르게 수행하고 있음을 볼 수 있다. <그림 10>은 QoS요소인 “해상도”의 가중치 값을 조정하여 매치메이킹을 수행한 결과를 나타내고 있다. 또한 사용자의 QoS요구를 반영하기 위해서 사용자 QoS 를 위한 사용자 입력을 받아 새로운 매치메이킹 결과를 반환한다.



〈그림 12〉 재현율과 정확률 비교

<그림 11>은 사용자로부터 “데이타암호화”에 대한 사용자 QoS 요구를 반영한 매치메이킹 결과를 보여준다.

<그림 12>는 위 실험을 바탕으로 사용자의 서비스 요청 질의인 “높은 해상도를 보이고 데이터암호화를 지원하는 의료정보 동영상 제공자를 검색하라”하는 질의를 수행한 재현율과 정확율의 비교 그래프이다.

비교 그래프에서 나타난 것처럼 본 논문에서 제안한 시스템이 재현율 측면은 조금 내려갔지만 사용사의 요구를 모두 반영한 매치메이킹을 수행하여 정확율의 향상을 가지고 온 것을 볼 수 있다.

5. 결 론

웹 서비스의 사용이 증가함에 따라서 동일한 기능을 제공하는 서비스들 중에서 사용자가

원하는 서비스를 검색하기 위해서 웹 서비스 비 기능적 측면인 서비스 QoS 정보는 서비스 매칭에 중요한 고려 요소가 되었다.

본 논문에서는 서비스 제공자의 QoS 정보를 저장하기 위해서 서비스 도메인별로 QoS 정보를 분류 하여 서비스 제공자가 자신의 서비스가 지원해야 하는 QoS 정보를 등록 전에 열람하고, 이를 반영한 서비스 정보를 등록함으로써 서비스 제공자의 QoS 요청을 반영한 매치메이킹이 가능하게 하였다. 또한 분류된 QoS 정보는 계층구조를 이루고 있어 사용자의 잦은 요구가 있는 QoS 요소에 대해서 서비스 제공자가 QoS 정보를 갱신 하고 향후 동일한 기능을 제공하는 서비스 제공자가 갱신된 QoS 정보를 반영 할 수 있는 구조를 제안하였다. 또한 저장되어 있지 않은 QoS 요청에 대해서도 RSS 매커니즘을 활용하여 동적으로 반영하게 함으로써 서비스 사용자의 QoS 요구를 모두 반영할 수 있도록 하였다. 이는 서비스 제공자의 서비스 등록 정보의 갱신 효과를 가지고 올 수 있어, 최신의 정보를 반영한 매치메이킹이 가능하도록 하였다. 기존의 서비스 정보의 갱신은 서비스 정보에 대한 변경이 있어도 이를 서비스 제공자가 갱신하지 않으면 등록 시점의 과거 서비스 정보를 반영한 매치메이킹이 이루어지기 때문에 서비스정보의 정확성을 유지하기 어려웠다.

본 논문의 향후 연구에는 서비스 사용자의 프로파일 기반의 서비스 매치메이킹을 수행하여 보다 지능적인 서비스 검색을 수행하고자 한다. 또한 서비스 QoS 요소 중 사용자로부터의 서비스 평가점수인 Reputation(평판) 항목 점수를 보다 의미 있게 고려한 매치메이킹에 대한 연구가 진행 되어야 할 것이다.

참 고 문 헌

[1] 이성우, “분산 시각 미디어 검색 프레임워크를 위한 웹 서비스 매치메이커”, 석사학위논문, 단국대학교 대학원 전자컴퓨터공학과, 2006.

[2] IBM, Microsoft, “Web Service Framework,” http://www.w3.org/2001/03/WSWS-popa/paper51, W3C Workshop on Web Services 11-12, San Jose, CA USA, April 2001.

[3] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, “SOAP Version 1.2 Part 1 : Messaging Framework,” http://www.w3.org/TR/soap12/, W3C Recommendation 24 June 2003.

[4] UDDI.org, “UDDI Technical White Paper,” http://uddi.org/pubs/uddi-tech-wp.pdf, 2006.

[5] T. Kawamura, J. A. De Blasio, T. Hasegawa, M. Paolucci, and K. Sycara, “A Preliminary Report of a Public Experiment of a Semantic Service Matchmaker combined with a UDDI Business Registry,” Proc. 1st International Conference on Service Oriented Computing (ICSOC 2003), Trento, Italy, December 2003.

[6] M. Klusch, B. Fries, K. Sycara, “Automated Semantic Web Service Discovery with OWLS-MX,” Proceedings of 5th International Conference on Autonomous

Agents and Multi-Agent Systems (AAMAS) 2006, ACM Press, Hakodate, Japan, 2006.

[7] Shuping Ran, "A Framework for discovering web services with Desired Quality of Services Attributes," IEEE International Conference on Web Services, Las Vegas, Nevada, USA, June 2003.

[8] A.Shaikali, Q. F. Rana, R. Al-Ali and D. W. walker, "UDDIe : An entended registry for Web Service," Symposium on applications and the Internet Workshops, IEEE CS pp.85-89. 2003.

[9] Daniel A. Menasce, H. Ruan, and H. gomaa, "A Framework for QoS-Aware Software Components," Proc. 2004 ACM Workshop on Software and Performance, San Francisco, CA, January 14, 2004.

[10] Diego Zuquim Guimarães Garcia, Maria Beatriz Felgar de Toledo, "A Web Service Architecture Providing QoS Management," Fourth Latin American Web Congress (LA-WEB'06), 2006, pp. 189-198.

[11] RSS 2.0 Specification "http://cyber.law.harvard.edu/rss/rss.html."

[12] Xu. T., "Reputation-Enhanced Web service discovery with QoS," Ph. D. Dissertation, School of Computing, Queen's University, Canada. 2006.

저 자 소 개

강필석 (E-mail : pskang@dblab.dankook.ac.kr)
2003년 단국대학교 컴퓨터공학과 (공학사)
2007년 단국대학교 대학원 전자컴퓨터공학과 (공학석사)
관심분야 데이타베이스, 유비쿼터스 웹서비스

안철범 (E-mail : ahn555@dankook.ac.kr)
1993년 성균관대학교 생물학과 (이학사)
2001년 단국대학교 대학원 컴퓨터공학과 (공학석사)
2001년~현재 단국대학교 대학원 전자컴퓨터공학과 박사과정 (수료)
관심 분야 데이타베이스, 멀티미디어 정보검색, 위치기반서비스, 유비쿼터스 웹서비스, 서비스지향 아키텍처

서보원 (E-mail : freeaha@naver.com)
1997년 아주대학교 컴퓨터공학과 (공학사)
1999년 아주대학교 컴퓨터교육학과 (공학석사)
2001년~현재 단국대학교 전자컴퓨터공학과 박사과정 (수료)
관심 분야 데이터베이스, 유비쿼터스 데이터베이스, 멀티미디어 데이터베이스, 멀티미디어 정보검색

나연묵 (E-mail : ymnah@dku.edu)
1986년 서울대학교 컴퓨터공학과 (공학사)
1988년 서울대학교 대학원 컴퓨터공학과 (공학석사)
1993년 서울대학교 대학원 컴퓨터공학과 (공학박사)
1993년~현재 단국대학교 전자컴퓨터공학부 컴퓨터공학전공 교수
1991년 IBM T. J. Watson 연구소 객원연구원
2001년~2002년 University of California, Irvine 객원교수
관심 분야 데이타베이스, 데이타 모델링, 데이타베이스 설계, 객체지향 데이타베이스, 멀티미디어 데이타베이스, 멀티미디어 정보검색, 시맨틱 웹, 이동객체 데이타베이스, 대용량 분산 데이타베이스