지능형 협업환경에서 서비스 발견을 위한 다중 에이전트 시스템 적용 방법

Multi-agent-based approach for service discovery in smart meeting spaces

배창혁, Changhyeok Bae*, 한상우, Sangwoo Han**, 김종원, JongWon Kim***

요약 지능형 협업환경에서 사용자가 원하는 서비스를 제공받기 위해 선행되어야 할 기술은 사용자가 원하는 협업 기능 요구사항에 따라 협업환경 내 서비스들을 적절히 발견하는 서비스 발견(service discovery) 기술이다. 일반적인 서비스 발견과 관련하여, UPnP(universal plug & play)의 SSDP(simple service discovery protocol)을 비롯한 여러 종류의 서비스 발견 기술이 존재하나, 지능형 협업환경이 요구하는 서비스 발견 특징을 만족시키기에는 일부 제약사항이 존재한다. 이에 본 논문에서는 지능형 협업환경에서 제공되어야 하는 서비스 발견 기술의 요구사항을 살펴보고, 이를 만족하기 위한 방법으로써 다중 에이전트 시스템 (multi-agent system, MAS)을 적용한 실용적인 서비스 발견 방법을 제시한다. 또한 지능형 협업환경의 일부 서비스에 제안된 방법을 적용하여, 본 접근방법의 타당성을 구현 결과를 통해 확인하고자 한다.

Abstract The service discovery method is an important technology finding and offering users' desirable services in smart meeting spaces. Extensive researches of the service discovery methods are achieved: SSDP(simple service discovery protocol) of UPnP(universal plug & play) and so on. However, there are several limitations to satisfy the requirements of service discovery in smart meeting spaces. In this paper, the requirements of service discovery in smart meeting spaces are investigated and the service discovery method based on multi-agent system is proposed in the practical aspect. Additionally, we explore the possibilities of the proposed approach by implementing a couple of services belonging to the smart meeting spaces.

핵심어: 지능형 협업환경, 서비스 발견, 다중 에이전트 시스템

본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅및네트워크원천기반기술개발사업의 지원에 의한 것임.

*주저자 : 광주과학기술원 정보기전공학부; e-mail: chbae@nm.gist.ac.kr

**공동저자 : 광주과학기술원 정보기전공학부; e-mail: swhan@nm.gist.ac.kr

***교신저자 : 광주과학기술원 정보기전공학부; e-mail: jongwon@nm.gist.ac.kr

1. 서론

컴퓨터의 성능 향상과 고성능의 인터넷 망이 빠르게 보급됨에 따라 지능형 협업환경 구축이 용이해지고, 이를 위한 구성 장치 및 서비스도 다양하게 개발되고 있다. 지능형 협업환경(Smart Meeting Space: SMeet) [1,2]은 지리적으로 멀리 떨어져 있는 사용자들이 마치 한 공간에서 협력하는 것과 동일한 수준의 공동 작업을 지원하기 위한 공간이다. 일례로 그림 1에서 보는바와 같이 현지/원격 여부에 무관하게 다중 지점의 다수 참가자들이 동일한 공간에서 협력하는 것과 같은 자연스러운 인터랙션을 네트워크로 공유되는 대형 디스플레이 장치를 중심으로 진행하면서, 이를 활용하여 공동 작업을 효과적으로 수행할 수 있는 공간을 생각해 볼 수 있다.

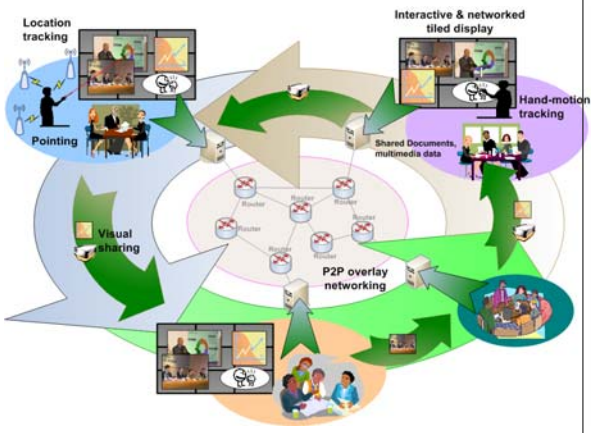

그림 1. 지능형 협업환경 개념도 [1].

지능형 협업환경에서 현지/원격지간 공동 작업을 효율적으로 하기 위해서는 다양한 기술들이 지원되어야 한다. 먼저 지능형 원격협업을 지원할 수 있는 시스템 및 네트워킹 구조를 설계하는 기술, 다양한 종류의 입력 채널의 내용을 동시에 보여줄 수 있는 디스플레이를 제공하는 기술, 협업 구성원 간의 자연스러운 데이터 및 태스크 마이그레이션 기술, 다양한 사용자 인터랙션을 제공하는 기술, 그리고 이 모든 기술의 장치와 서비스를 발견하고 합성하여 사용자의 요구사항에 맞게 서비스를 제공하는 기술 등이 지원되어야 한다.

지능형 협업환경의 서비스들은 서로 자유로운 정보전달이 가능해야하고 사용자의 지시 없이 서비스 스스로 또는 다른 서비스와 협상 후 판단하여 행동할 수 있어야 한다. 또한 어떤 입력에 대해 서비스 자체적인 처리과정을 통해 자신에게 유리한 방향으로 행동할 수 있어야 한다. 다중 에이전트 시스템 (multi-agent system: MAS) 기술을 사용하면 지능형 협업환경에서 서비스들이 요구하는 요구 조건을 만족할 수 있다고 알려져 있다 [3]. 지능형 협업환경에서 다중 에이전트 시스템 기술을 이용하여 협업 도구와 결합하면 사용자가 서비스를 요청할 때 이를 이행하기 위해서 관련된 서비스들끼리 자율적으로 통신하여 사용자에게 서비스를 제공하도록 할 수 있다. 또한 제공하는 서비스가 문제가 생겼을 때 에이전트 스스로 또는 다른 에이전트와의 협력을 통해 문제점을 판단하여 문제를 해결하거나 다른 서비스로 대체한다.

또한 지능형 협업환경에서 사용자가 원하는 서비스를 구성하고 공간 내에 어떠한 서비스들이 있는지 알려면 서비스 발견이 선행되어야 한다. 일반적 서비스 발견과 관련하여 Universal Plug and Play(UPnP)의 SSDP [4], Jini [5]의 발견 프로토콜을 비롯한 여러 종류의 서비스 발견 기술이 존재한다. 하지만 UPnP, Jini 등 홈 네트워크 표준 등에서 제안하는 서비스 발견 규약은 서비스를 발견하고 서비스의 상태와 정보를 알려주는데 그쳐 지능형 협업환경에서 요구하는 서비스들 간의 자유로운 의사교환이 가능한 상호작용성(interactivity), 사용자나 다른 장치의 직접적인 지시 없이도 스스로 행동하는 자율성(autonomy)과 입력에 대해서 자체적인 처리과정을 통해 자신에게 이로운 방향으로 행동하는 반응성(reactivity) 등의 요구조건을 만족하지 못하고 있다 [6]. 또한 홈 네트워크에 중점을 두어 제안된 기술들이기 때문에 외부 네트워크의 장치와 서비스를 발견하고 서로 통신하는데 어려움이 있다. 따라서 향후 서비스 지향 구조(service-oriented architecture)를 반영할 수 있는 지능형 협업환경의 특성을 고려할 때, 서비스 간 상호작용성, 자율성, 반응성을 지원하기 위한 구조가 필수적이며, 서비스 발견 기술도 이러한 요구사항을 대비하여 준비되어야 한다.

따라서 본 논문에서는 GIST에서 개발중인 SMeet [1,2] 지능형 협업환경에 대하여 서비스들 간에 자율성, 반응성, 협력을 포함한 서비스 간 상호작용성을 지원하기 위해, 다중 에이전트 시스템(multi-agent system, MAS) [3]의 적용 방안을 논의한다. 제안 방식에서는 각 협업 도구(collaboration tools)들은 에이전트와 결합하여 서비스로 만들어지며, 서비스 발견 에이전트에 의해서 발견된 각 서비스는 GUI상에서 다양한 형태의 뷰(view)로 표현된다. 제안된 방식을 경험적으로 검증하기 위하여, 에이전트를 구현하여 지능형 협업환경의 일부 요소 서비스에 적용하여 실현 가능성을 검토해 보고자 한다.

본 논문의 구성은 다음과 같다. 2절에서는 서비스 발견 방법에 관한 관련연구를 소개한다. 3절에서는 서비스 발견을 위한 다중 에이전트 시스템 적용방법에 대해서 설명하고, 4절에서 실제 구현 방법과 실제 적용을 통한 검증 결과에 대해 소개한다. 마지막으로 5절에서 결론과 향후 계획에 대해서 논의하면서 본 논문을 마무리한다.

2. 관련연구

먼저 서비스 검색에 관련한 기술을 잠시 살펴보면, UPnP는 1999년 Microsoft사에 의해 제안된 기존의 PC와 주변기기를 연결하던 Plug-and-Play의 확장 기술로 HTTP 프로토콜을 사용하여 홈 네트워크 기기간의 상호 운용을 목표로 하고 있다. 또한 Jini는 1998년 SUN사에 의해 제안된 기술로 UPnP와 마찬가지로 홈 네트워크에서 미들웨어로 동작하며 다양한 장치들 사이의 연결성과 서비스의 제어를 목적으로 개발된 것이다. 추가적으로 Service Location Protocol (SLP) [7]는 IETF에서 제정한 분산, 확장이 가능한 서비스 발견 프로토콜이다. 이 프로토콜은 서비스 발견을 요청하는 User 에이전트, 서비스들의 광고를 담당하는 Service 에이전트 그리고 각 에이전트들이 보고한 내용을 저장하는 Directory 에이전트로 구성된다. 하지만 위의 서비스 발견 프로토콜은 홈 네트워크에 제한적이기 때문에 원격지간 서비스 발견과 상호작용성이 제한되어 지능형 협업환경에는 적합하지 않다.

Ronin Agent Framework (Ronin) [8]는 Jini 기반의 동적인 분산 에이전트 시스템 환경을 위해 개발된 프레임워크이다. Ronin에서 에이전트는 Common-Agent 속성과 Domain-Agent 속성 두가지 유형으로 표현한다. Common-Agent는 에이전트의 일반적인 기능과 능력을 정의한다. Domain-Agent는 한 에이전트의 특징적인 기능을 정의한다. Ronin 에이전트는 Common-Agent 속성을 공유하고 이를 바탕으로 서비스를 발견한다. 그리고 발견된 서비스의 에이전트들이 에이전트간 통신규약을 가지고 통신을 하면서 협상을 한다. 하지만 이 프레임워크는 레퍼런스가 많이 없어 아직 신뢰성 있는 검증을 할 수 있는 단계는 아니다.

웹 온톨로지 언어인 OWL-S와 에이전트를 사용하여 적응형 소프트웨어에 대한 웹 서비스 발견 방법 [9]에서는 크게 세 단계로 나누어 서비스 발견이 이루어 진다. 첫 번째 단계는 사용자의 목적을 달성하기 위해서 후보 서비스들을 발견한다. 그 다음 두 번째 단계에서는 후보 서비스들 중 협상을 통해 적절한 서비스를 선택하게 된다. 하지만 목적에 만족하지 못하면 다른 여러 서비스들의 조합을 통해 사용자가 요구하는 서비스를 찾게 된다. 이 때 시스템적인 오버헤드가 발생하고 이를 줄이기 위해 하나의 레지스터 아래 여러 개의 하위 레지스터를 두어 최소한의 검색으로 찾게 한다. 마지막 단계에서 적절한 서비스를 찾으면 서비스를 사용자에게 제공을 해준다.

University Rey Juan Carlos에서 헬스케어를 위한 에이전트 기반의 서비스 발견방법 [10]을 제안하였다. 각 인터랙션 유형과 역할은 온톨로지로 서비스를 정의한다. 서비스는 하나의 디렉토리에 서비스의 역할 기반의 정보를 저장되고 사용자가 서비스를 요청을 하면 디렉토리에 저장된 역할 기반의 정보를 이용해 가장 적합한 서비스를 발견해서 사용자에게 알려준다. 다른 서비스 발견 기술은 웹서비스를 이용하였거나 서비스의 입출력 결과를 이용하였지만 여기서 제안한 에이전트 기반의 서비스 발견 기술에는 인터랙션 문맥 정보와 서비스의 역할 정보를 이용하였다.

3. 서비스 발견을 위한 다중 에이전트 시스템 적용

그림 2는 협업 도구와 에이전트의 관계, 각 에이전트와 서비스 발견자와의 관계를 나타내고 있다. 협업 도구는 지능형 협업환경에서 이용하고 있는 미디어/네트워킹/디스플레이/인터랙션에 이르는 협업 작업을 지원하기 위한 소프트웨어 및 하드웨어이며, 에이전트는 협업 도구를 사용자가 발견하고 협업 도구의 정보를 획득하기 위해 사용자와 협업 도구를 잇는 연결 소프트웨어이다. 협업 도구는 에이전트와 결합되어 하나의 SMeet Service로 정의된다[1]. 디바이스는 디바이스를 관리해주는 소프트웨어와 에이전트가 결합되어 하나의 서비스 개념으로 만들어진다. 서비스 발견자의 에이전트는 각 서비스와 디바이스 관리 소프트웨어의 에이전트와 통신을 통해 서비스를 발견하여 Device&Service GUI에 표현한다.

SMeet Service로 협업 도구가 포장되므로써 지능형 협업환경에서 요구하는 기존의 협업 도구들끼리의 통신, 서비스가 문제가 생겼을 때 에이전트 스스로 판단하여 문제를 해결하거나 에이전트 간의 협상을 통해 다른 서비스로 대체할 수 있는 기반이 마련된다.

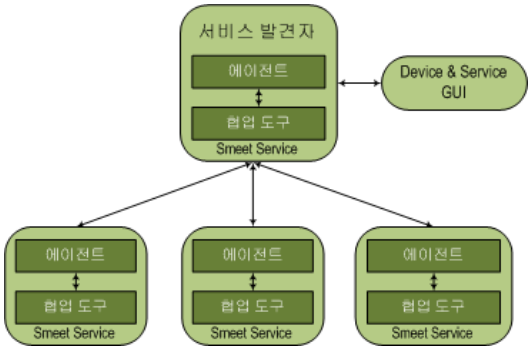

그림 2. 에이전트 기반의 서비스 발견 구조.

- **에이전트와 협업 도구의 결합에 의한 SMeet Service 생성:** 그림 3는 SMeet Service의 구조를 나타내고 있다. 에이전트는 협업 도구를 발견하는 Discovery 모듈, 협업 도구를 실행 및 중단 시킬 수 있는 Start and Stop 모듈, 협업 도구의 상태정보를 주기적으로 체크하여 서비스 발견자에게 보고하는 HeartBeat, Service states 모

1) SMeet에서 서비스는 요소 (component), 합성 (composite) 서비스 등으로 나눠진다.

듈을 가지고 있다. 이 모듈을 이용해 각 서비스 에이전트는 협업 도구의 정보를 제공, 서비스를 실행 및 중단할 수 있고 동작여부를 확인하는 기능을 제공한다. 모든 에이전트와 협업 도구 사이에 정보 표현방식을 정의하고 소켓통신, 웹서비스(예를 들면 SOAP/XMLRPC 등)를 비롯한 다양한 통신방법으로 정보를 교환한다. 협업 도구를 실행, 중단시킬 때는 Device&Service GUI를 통해서 사용자가 특정 에이전트를 통제하여 협업 도구에 명령을 내릴 수 있다. 또한 Device&Service GUI가 협업 도구의 상태와 성능을 표현할 수 있도록 협업 도구는 주기적으로 자신의 상태와 성능을 에이전트를 통해 서비스 발견자에게 전달한다.

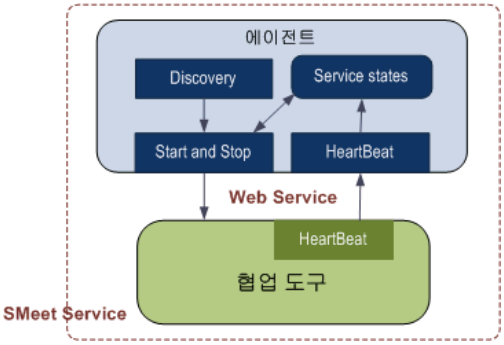

그림 3. SMeet Service와 에이전트의 연계 구조.

- **서비스 발견 및 정보제공:** 서비스 발견자는 서비스를 발견하고 Device&Service GUI에 발견된 디바이스와 서비스를 표시한다. 사용자의 정보제공 요청이 있으면 서비스 발견자 에이전트는 협업 도구의 정보를 서비스 에이전트에 요청하여 협업 도구에게 받아 사용자에게 전달한다. 이때, 서비스 에이전트와 서비스 발견자 에이전트 간에 에이전트 간 통신 규약(ACL communication)을 통해 서비스 발견을 위한 메시지를 교환한다.
- **Service&Device GUI의 정보 표현:** 서비스 발견자에 의해서 발견된 SMeet Service들은 Device&Service GUI에 아이콘 뷰, 트리 뷰 등 다양한 형태로 표시된다. 이 다양한 뷰를 통해서 사용자들은 한 눈에 협업 도구의 정보, 동작여부를 확인할 수도 있고 실행, 중단시킬 수 있다.

4. 구현방법 및 검증 결과

본 논문에서는 협업 도구를 다중 에이전트 시스템 라이브러리의 한 종류인 JADE(Java Agent DEvelopment Framework) [11]를 이용해 에이전트와 결합하여 SMeet Service로 구성하고 그 서비스들을 보여줄 수 있는 Device&Service GUI를 구현했다. C++로 구현되어 있는 협업 도구와 Java로 구현된 에이전트 간의 통신 채널을 구성하기 위해 웹 서비스의 일종인 XMLRPC [12]를 이용하여 구현했다. 기존에 있는 협업 도구에 최소한의 수정으로 에이전트와 통신이 가능하게 하기 위해 XMLRPC 모듈과 상태, 성능을 보고하는 모듈을 만들고 메인 함수에서 추가적인 스레드를 만들어 두 모듈을 스레드 내부에서 실행시키게 구현하였다. 서비스 에이전트와 서비스 발견자 에이전트간의 통신 채널을 구성하기 위해 에이전트 간 통신 규약(ACL communication)을 이용하였고 온톨로지를 이용하여 에이전트 간 통신하는 정보를 표현했다. Device&Service GUI는 Java 그래픽 라이브러리를 이용해 트리 뷰와 이이콘 뷰를 나타낼 수 있게 구현했다.

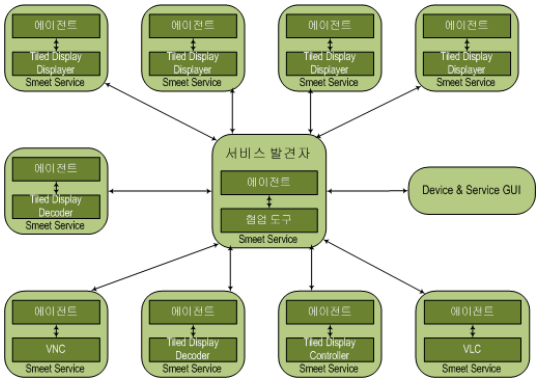

그림 4. Networked Tiled Display 협업 도구 배치도.

지능형 협업환경은 미디어 통신, 네트워킹, 디스플레이, 인터랙션 등 다양한 서비스들로 구성되어 있다. 그 중 지능형 협업환경에서 서비스 중 대표적인 것이 네트워크 기반 타일드 디스플레이 (Networked Tiled Display)를 이용한 서비스가 있다 [13]. 이는 여러 대의 모니터를 격자 형태로 배치하여 마치 하나의 모니터처럼 사용하는 초고해상도 디스플레이 서비스이다. 그림 4에서 보는 바와 같이 Networked Tiled Display 서비스를 구성하는 협업 도구에는 실제 화면에 보여주는 Tiled Display Displayer, 외부에서 데이터(비디오 영상, 컴퓨터 공유화면)를 받아 처리하는 Tiled Display Application, Tiled Display를 제어하는 Tiled Display Controller, Tiled Display의 Display 장치들의 동기화를 맞춰주는 Tiled Display SyncMaster 등이 연관된다. 그리고 비디오 영상을 전송하는 VLC 서비스와 컴퓨터의 화면을 공유하는 VNC 서비스가 있다. Networked Tiled Display에 사용자가 요구하는 서비스를 보여주기 위해서는 사용자가 요구하는 서비스(e.g., VLC, VNC, Image등)와 Networked Tiled Display를 구동하기 위한 Display Application, Displayer, Controller, SyncMaster 등 다양한 서비스가 필요하다. 이러한 서비스들이 서로 유기적으로 협력해 Networked Tiled Display가 완성된다.

그림 5. Networked Tiled Display 서비스 배치.

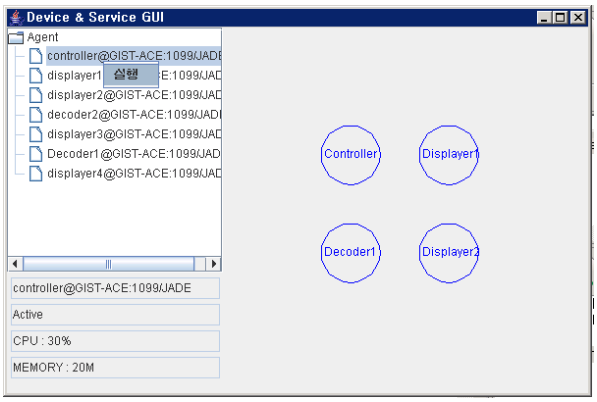

그림 6 Device&Service GUI

그림 5는 GIST에 구축되어 있는 이동형 Networked Tiled Display 시스템이다. 그림 6에서는 그림 5에 있는 장치와 서비스를 발견하여 Device&Service GUI에 발견된 서비스들과 그 서비스들의 상태 정보를 보여주고 있다. Device&Service GUI에서 발견된 SMeet Service를 트리 뷰와 아이콘 뷰를 통해 나타낸다. 사용자는 Device&Service GUI에서 트리 뷰에 나타나는 서비스들의 이름에서 마우스 오른쪽 버튼을 클릭해서 나오는 팝업창을 이용해 SMeet Service를 실행하고 중지시킬 수 있다. 트리 뷰에서 서비스 이름을 더블 클릭하면 해당 서비스는 아이콘 뷰에 나타난다. 그리고 왼쪽 아래 상태 창에는 트리 뷰에서 선택한 SMeet Service의 상태 정보(Active 또는 Inactive)와 서비스 디바이스의 성능 정보(CPU, Memory)를 볼 수 있다.

5. 결론 및 향후 계획

본 논문에서는 SMeet 지능형 협업환경의 효율적인 구성을 위해 협업 도구와 에이전트를 결합한 검색 방법을 제안하고 있다. 서비스 발견자를 이용하여 서비스를 발견하고 Device&Service GUI를 통해 서비스의 실행 및 중지, 동작 여부를 알고 정보를 제공받도록 구성하였다. 그리고 지능형 협업환경에서 여러 협업 도구 중 Networked Tiled Display에 한정해 시험적으로 구현하여 가능성을 알아보았다. 현재 구현에서는 에이전트에 의해 발견된 서비스들은 주기적으로 정보를 Device&Service GUI에 제공한다. 앞으로는 이를 이용해 각 서비스의 자원을 효율적으로 할당하고 활용하도록 하는 기능도 필요하다. 또한 디바이스 장애시 다른 디바이스를 통해 서비스를 대체하는 회복 기능도 검토되어야 한다.

참고문헌

[1] 한상우, 김남곤, 최기호, 고수진, 이현룡, 김종원, "다자간 협업을 위한 컴포넌트 서비스 기반 소프트웨어 구조 설계," *HCI 2007*, Feb. 2007.

[1] Sangwoo Han and JongWon Kim, "SMeet: a smart meeting space with interactive and networked display," in *Proc. of UbiComp Workshop (UbiMEET)*, pp. 115-116, Sep. 2007.

[2] 송종철, 정현수, 홍기채, "멀티 에이전트 시스템의 연구 동향," *주간기술동향 970호*, pp. 13-26, Jan. 2000.

[3] "Understand UPnP: A White," http://www.upnp.org/download/UPNP_UnderstandingUPNP.doc.

[4] "Jini Architecture Overview: Technical White Paper," http://www.sun.com/software/jini/whitepapers/architecture.html.

[5] J. I. Vazquez, D. L. Ipina, and I. Sedano, "SoaM: A web-powered architecture for designing and deploying pervasive semantic devices," *International Journal of Web Information Systems (IJWIS)*, Mar, 2005.

[6] SLP, http://www.ietf.org/html.charters/OLD/svrloc-charter.html.

[7] Ronin Agent Framework, http://ebiquity.umbc.edu/project/html/id/25/Ronin-Agent-Framework.

[8] 이창호, 김진한, 박동민, 이병정, "OWL-S와 문맥 정보를 가진 에이전트를 사용한 웹 서비스 발견 기법," *한국정보과학회 가을 학술발표논문집*, 2006.

[9] C. Caceres, A. Fernandez, S. Ossowski, and M. Vasirani, "Agent-Based Semantic Service Discovery for Healthcare: An Organizational Approach," *IEEE Intelligent Systems*, Nov/Dec, 2006.

[10] JADE, http://jade.tilab.com/.

[11] XMLRPC, http://www.xmlrpc.com/.

[12] Kiho Choi, Sangwoo Han, and JongWon Kim, "Supporting High-Performance Networked-based Clustered Displays for Avanced Collaboration Environments," in *Proc. of SPIE ITCOM*, pp. 677707-1-10, Sep. 2007.