논문 2005-42CI-5-6

상황정보를 기반으로 한 서비스 관리 시스템 설계

(Design of Service Management System based on Context Information)

이 승 근*, 임 기 욱**, 이 정 현***

(Seungkeun Lee, Kiwook Rim, and Junghyun Lee)

요 약

상황 인식 기반 편재형 컴퓨팅(Pervasive Computing) 환경의 다양한 응용에 관심이 증대되고 있으며, 개발자들이 상황인식 응용을 보다 쉽게 개발할 수 있도록 지원하는 개발환경들에 대한 연구가 활발하게 이루어지고 있다. 서비스 관리 시스템은 상황 인식 응용이 필요로 하는 서비스를 찾아서 제공해 주는 시스템으로 상황 인식 응용 개발환경에 필수적인 부분이다. 그러나, 기존 연구에서는 단순 구문 매칭이나 서비스 타입 등의 제한적인 온톨로지 기반 매칭을 사용하고 있으며 상황 정보에 대한 고려가 없다. 또한, 사용자가 원하는 서비스가 없는 경우 기존 서비스들을 조합해서 이용할 수 있도록 할 수 있어야 한다. 이 논문에서는 상황 정보 기반 서비스 관리 시스템을 제안한다. 제안하는 시스템은 온톨로지를 이용한 의미적 매칭 방법과 상황 정보를 고려함으로써 보다 정확한 검색이 가능하게 하고, 서비스 조합 기능을 제공함으로써 사용자가 원하는 서비스가 서비스 레지스트리에 없는 경우에는 기존 서비스들을 조합하는 서비스 리스트를 제공할 수 있다. 설계한 시스템을 평가하기 위해서 서비스 질의를 위한 프로토타입을 개발하였으며 이를 통해서 상황 정보 기반 검색과 서비스 조합이 적절하게 지원함을 보였다.

Abstract

There has been an increase in the interest of applications that use a combination of both pervasive computing technology and context-aware technology. This application based on the development environment along with the support of developing context-aware applications is now being researched thoroughly and by many. The service management system provides services that are needed for context-aware applications. This system is an integral part of the developmental environment of context-aware applications. But there is a restrictive matching based on ontology that uses simple syntactic matching or a plain type of service used in previous researches. And there is also no consideration for context-aware information. Also, if the user is unable to find a service that is satisfactory, or is a service which a user does not desire, they may use a service which is composed of other existing services. This paper proposes a service management system based on context-aware information. The proposed system enables the accurate finding of services by considering semantic matching methods based on ontology and context-aware information. If the user does not find a service that is helpful in the service registry, it can provide the service list to enable other existing service compositions, by providing the functionality of these service compositions. As a result, the experiment of the system proposed has shown that the system properly supported the service discovery based on context-aware information and service composition.

Keywords: 상황 인식, 서비스 관리, 온톨로지, 편재형 컴퓨팅

I. 서 론

컴퓨팅 기술과 네트워크 기술이 급속도로 발전하면서 유비쿼터스 환경에 대한 관심이 증대되고 있으며, 다양한 응용의 개발을 포함한 관련 연구가 진행되고 있다[1][2]. 유비쿼터스 환경은 컴퓨팅과 네트워크를 기반으로 물리공간을 지능화함과 동시에 물리공간에 펼쳐진 각종 사물들을 네트워크로 연결시키려는 노력으로 정의할 수 있다. 또한, 상황 인식(Context Awareness) 기능

* 학생회원, 인하대하교 컴퓨터정보공학부
(Dept. of Computer Sci. & Information Eng., Inha Univ.)
** 평생회원, 선문대학교 컴퓨터정보학부
(Dept. of Computer & Information Sci., Sunmoon Univ.)
*** 평생회원, 인하대학교 컴퓨터공학부
(School of Computer Sci. & Eng., Inha Univ.)
※ 본 연구는 정보통신부 대학 IT 연구센터 육성 지원 사업의 연구 결과로 수행되었습니다.
접수일자: 2005년8월17일, 수정완료일: 2005년9월6일

을 보유함으로써 사용자 및 사물 등의 객체를 인식하고 이들의 현 상태에 따른 상황 정보를 수집하여 서비스에 적용되고 있다[3]. 유비쿼터스 환경에서 사용자에게 제공되는 서비스는 기존의 전통적인 서비스보다 다양한 환경을 기반으로 하고 있으며, 수많은 서비스들이 혼재되어 있기 때문에 사용자가 필요로 하는 가장 적절한 서비스를 제공해 주기 위한 서비스 검색 방법과 더 나아가서 서비스들을 적절하게 조합해서 사용자가 원하는 기능을 제공해 주는 기술이 필요하다[3][4].

기존의 전통적인 소프트웨어 환경에서 서비스 검색 및 조합에 대한 노력이 진행되어 왔으며, 특히 웹서비스 분야에서는 WSDL (Web Service Desctiption Language), UDDI(Universal Description, Discovery, and Integration)과 같은 표준화된 연구와 구문적 검색 (syntactic matchmaking) 방식의 한계를 극복하고 보다 정확한 검색이 가능한 온톨로지를 이용한 의미적 검색 (semantic matchmaing)을 지원하기 위한 OWL-S등의 연구가 진행되고 있다[5]. 서비스 제공자가 중앙의 레지스트리에 서비스 명세를 등록하고, 서비스 요청자는 질의를 전달하면, 레지스트리는 질의내에 포함되어 있는 특정 키워드를 포함하는 서비스 명세를 찾아서 해당 서비스를 리턴해주는 과정을 거친다. 이 방식은 구문적으로는 다른 용어이나 의미적으로는 유사한 키워드를 질의하는 경우나 구문적으로는 유사하나 의미적으로는 다른 경우가 존재할 수 있기 때문에 검색된 결과의 낮은 정확성을 가져온다. 이러한 키워드의 모호성은 사용자 질의와 서비스 기술간의 불일치성을 야기할 수 있으며, 검색 결과의 품질이 매우 낮을 수 있다. 구문 매칭 방식의 단점을 극복하기 위해서 제시된 의미적 매칭 방법은 온톨로지를 통한 시멘틱 매칭을 사용함으로써 기존 구문적 매칭 방법보다 정확한 서비스 발견을 가능하게는 한다.

이 연구에서 관심을 갖고 있는 상황 인식 환경에서는 상황 인식 응용에서 서비스 호출시 적용되는 상황 정보에 대한 고려가 통해서 보다 정확한 서비스를 발견할 수 있다. 예를 들어서, 사용자가 거실에 들어오면 전등을 자동으로 켜주는 상황 인식 서비스를 생각해 보자. 이때, 사용자의 위치와 전등의 위치의 상황 정보를 고려하면 사용자에게 가장 가까운 전등을 켜줄 수 있다. 사용자가 고속도로를 주행중에 배가 고파서 자동차에 상황 인식 개인 자동차 시스템에 요청한다. 서비스 발견 메커니즘은 사용자로부터의 요구를 받아서 이 요구를 만족시킬 수 있는 서비스를 찾기 시작한다. 이때, 사용자의 위치와 현재 시간과 식당의 운영시간 등의 상황 정보를 고려해서 가장 적합한 서비스를 제공할 수 있다.

서비스 조합은 사용자가 원하는 서비스를 제공하기 위해서 기존 서비스들을 조합해서 제공해 주는 기술이다. 온톨로지 기반 웹 서비스 기술 언어인 OWL-S는 웹 서비스의 조합 및 자동화된 실행을 지원하고 있다[5]. 이를 위해서 하나의 서비스를 서비스 프로파일, 서비스 모델, 서비스 그라운딩으로 구성된 온톨로지로 기술한다. 서비스 조합 방법은 사용자의 질의에서 입력과 출력을 추출한 후 하나의 서비스의 출력이 다른 서비스의 입력으로 연결하는 방식으로 서비스를 조합하게 된다. OWL-S는 온톨로지 기반 서비스 명세와 자동화된 서비스 조합에 적합하나 웹 서비스에 초점이 맞추어져서 정의되어 있다. 상황 인식 응용은 일반적인 웹서비스보다 실시간적인 특징을 갖는다. 따라서, 서비스 조합에 포함되는 서비스의 실행 시간이나 신뢰도등의 정보를 알지 못하면 사용자가 서비스의 결과를 기대하는 시간보다 초과되는 경우나 서비스 실행중의 문제로 인해서 서비스의 롤백(Rollback)이 발생할 수도 있는 등의 문제가 발생할 수 있다. 따라서, 실제 서비스를 조합해서 실행하기 전에 전체 실행 시간이나 신뢰성 등의 QoS에 대한 예측이 가능해야 한다.

상황 인식 응용은 환경내의 다양한 센서, 실행기 (Actuator), 가전 등을 네트워크로 연결하고 센서 등으로부터 얻어진 데이터를 해석하고 추론해서 상황을 유추하고 이에 따른 적절한 서비스를 제공하게 된다. 이러한 상황 인식 응용은 이들 주위 환경과 다양한 프로토콜과 미들웨어를 통해서 연결된다. 1999년 OSGi Alliance에서 발표한 OSGi 프레임워크 기술은 센서, 실행기, 가전등과 연결될 수 있는 다양한 미들웨어를 지원하며 홈 네트워크와 텔리메틱스 환경에서 게이트웨이에 적용될 수 있다[6]. 따라서 OSGi는 상황 인식 응용을 위한 기본 프레임워크로 매우 적합한 기술이며 여러 연구들이 OSGi 기반한 상황 인식 응용에 대해서 이루어지고 있다[8][9].

우리는 현재 OSGi 기반 상황 인식 응용 개발 환경에 대한 연구를 진행 중에 있다. 이 연구는 OSGi 프레임워크상에서 온톨로지 기반 상황 기능과 서비스 검색 및 조합 기능, 이동 서비스 지원 기능을 개발하고 있으며, 이를 통해서 상황 인식 응용을 쉽게 개발하고 운용할 수 있도록 하는데 초점을 맞추고 있다. OSGi 프레임워크에도 서비스 검색 기능이 있으나, 단순한 키워드 매

칭에 의한 구문 검색을 이용하기 때문에 검색의 효과가 떨어진다. 이 논문에서 제시한 서비스 검색 시스템은 전체 연구의 한 부분이며, OSGi 기반 상황 인식 서비스 플랫폼에서 서비스 발견의 효율을 향상시키고, 다양한 서비스들을 조합할 수 있는 기능을 갖는다. 이를 위해서 온톨로지에 기반한 의미적 매칭과 상황 정보를 고려한 서비스 검색 방안, 서비스 조합 방안 및 평가 모델을 제시한다.

이 논문에서 설계하는 서비스 관리 시스템은 상황 정보를 고려한 온톨로지 기반 서비스 검색과 자동화된 서비스 조합을 가능하게 하기 위한 조합 모델 및 평가 모델을 적용한다. 이를 위해서 서비스 검색 및 조합을 위한 온톨로지를 정의하며, 온톨로지 추론을 위한 온톨로지 추론 엔진을 사용한다. 또한, 서비스 조합 계획을 QoS 기반으로 평가할 수 있는 평가 모델을 제시한다. 전체 시스템은 OSGi 프레임워크에서 실행될 수 있게 설계된다. 설계된 시스템은 상황 인식 응용의 프레임워크에 사용될 수 있으며, 기존 서비스 검색보다 정확한 검색을 제공하고 서비스의 조합을 지원할 수 있는 장점을 갖는다.

Ⅱ. 관련연구

1. 상황 인식 기술

상황 인식 시스템은 그들의 상황을 인식하고 변화하는 상황에 이음새 없이 적응할 수 있는 응용 개발을 위한 메커니즘을 제공한다. 상황 인식 응용은 개체의 상황을 사용해서 상황 내에 사용자의 요구에 최대한 적합하게 행위를 변형한다. 이러한 응용은 스마트홈과 같은 다양한 응용 범위에서 사용될 수 있다. 스마트홈은 네트워크로 센서와 디바이스, 가전들을 연결하고 가정 내의 작업을 지원하기 위해서 이러한 디바이스들과 서비스들이 상호작용을 통해서 지능형 환경을 구성한다. 예를 들어서, 스마트홈내 환경에서, 상황 인식 응용은 사용자의 혈압이 특정 값을 넘는 경우 병원에 자동으로 연락을 하거나 집에 있는 가족 구성원에게 의사의 전자 진단서에 따라서 약을 처방한다.

상황 인식 컴퓨팅은 최근 몇 년 동안 많은 연구가 진행되고 있으며 기술의 유용성을 입증할 상황 인식 시스템들이 개발되고 있다. 그러나, 상황 인식 응용의 개발은 적절한 인프라스트럭처의 부재로 인해서 아직도 복잡하고 시간 소모적인 작업이다. 이를 위해서 상황 인식 응용을 위한 프레임워크에 대한 연구가 진행되고 있으며, 그 연구들은 나름대로의 서비스 발견 모델을 제안하고 있다. 조지아 공대에서 연구한 Context Toolkit은 상황 인식 응용의 개발과 배치를 가능하게 하는데 목적을 두고 있다[10]. Context Toolkit에서 사용되고 있는 발견 메커니즘은 중앙집중식 모델이다. 단일 발견자 또는 중앙 레지스트리를 사용하며, 모든 컴포넌트는 네트워크 주소와 포드 정보를 레지스트리에 등록한다. 발견자는 각 컴포넌트들이 제대로 동작하는지 확인하기 위해서 등록된 각 컴포넌트들을 사전에 확인하며, 응용은 레지스트리가 제공하는 화이트 페이지(white page)와 엘로우 페이지 (yellow page) 서비스를 이용해서 서비스를 발견할 수 있다. 메릴랜드 대학에서 개발한 상황인식 브로커 구조 (COBRA: Context Broker Architecture)는 상황 인식 에이전트를 개발하는 비용과 노력을 감소시키는 아키텍처 개발을 목적으로 하고 있다[11]. COBRA 구조는 지능형 회의실 환경등에서 상황 인식 시스템을 런타임시 지원하기 위한 브로커 중심의 에이전트 구조를 제시하고 있다. COBRA는 세계를 몇 개의 응용 도메인으로 구분하고 있으며, 각 도메인은 사람, 디바이스, 물리적 객체들로 구성된 지식 모델을 나타낸다. 브로커의 지식 베이스는 이 도메인의 상황 모델과 도메인의 시스템 지식을 위한 데이터 저장소로써, W3C에서 발표한 웹 온톨로지 언어인 OWL (Ontology Web Language)를 이용해서 도메인을 기술하는데 필요한 도메인 온톨로지를 사용한다. 이것은 잘 정의된 어휘, 개념, 기능, 관계의 집합으로 브로커가 추론하고 행동하는데 사용된다. COBRA 시스템은 디렉토리 서비스 에이전트를 제공한다. 이 에이전트는 일정 반경 내의 특정 도메인의 디렉토리를 발견한다. 예를 들어서, 예를 들어서, 두 사용자가 회의실에 만났을 때, 에이전트는 두 사용자가 공유할 수 있는 디렉토리를 발견한다. 온톨로지를 이용한 COBRA 시스템은 단순한 구문적 매칭에 의한 서비스 발견이 아니라 의미적 매칭을 이용함으로써 보다 정확한 검색을 가능하게 한다. 온톨로지의 주요한 목적은 특정 도메인내에서 공통의 이해를 가능하게 하는 것이다. 온톨로지를 이용함으로써 상황 정보가 모호함없이 명확히 정의될 수 있기 때문에 보다 향상된 매칭을 가능하게 한다. 그러나, 기존 연구들은 서비스 매칭시 상황 정보를 고려하고 있지 않으며, 서비스 조합 기능이 제공되지 않는다.

2. OSGi

OSGi는 1999년에 많은 주요 기업들이 참여해서 설립

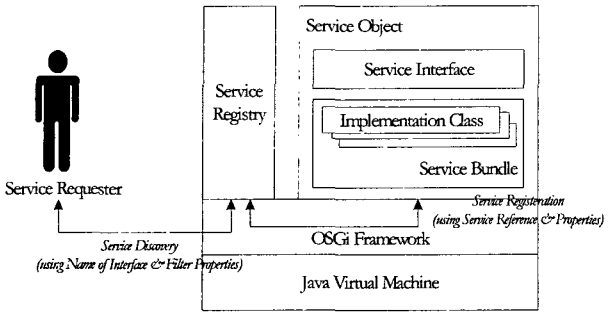

그림 1. OSGi 서비스 발견 모델
Fig. 1. Service Discovery Model in OSGi.

한 개방형 포럼으로 홈, 자동차, 모바일 환경 등에서 모든 종류의 네트워크로 연결된 디바이스들을 위한 응용과 서비스를 관리하기 위한 개방형 서비스 플랫폼을 개발을 목적으로 하며 현재 스펙 버전 3을 발표하고 있다 [6][7].

OSGi 스펙은 서비스 제공을 위해 자바 플랫폼을 정의하고 있으며 서비스의 기능은 번들(bundle)로 구현된다. 번들은 사용자나 다른 서비스에게 기능을 제공할 수 있는 자바 클래스들의 집합이며, 서비스는 자바 서비스 인터페이스로 기술된다. 인터페이스와 구현 클래스는 서비스 참조 (service reference)로 함께 사용된다. 또한, 서비스는 서비스의 특정 속성을 규정하기 위해서 키와 값의 쌍을 사용할 수 있다. 서비스는 서비스 제공자에 의해서 중앙의 서비스 레지스트리 (service registry)에 등록되며, 따라서 그림 1과 같이 중앙집중화된 서비스 발견 모델을 사용한다.

서비스가 서비스 레지스트리에 등록된 이후 서비스 요청자는 서비스 호출을 위해 필요한 서비스 참조를 얻기 위해서 서비스를 발견할 수 있다. 요청은 인터페이스 이름의 구문적 매칭(syntactical matching)이나 필터를 이용해서 정의된 프로퍼티에 기반한다. 서비스 참조를 얻은 후 서비스 요청자는 서비스의 프로퍼티를 이용해서 어떤 서비스를 이용할 것인지를 결정할 수 있다. 이와 같이 OSGi 서비스 발견 모델은 서비스 인터페이스나 프로퍼티의 구문적인 매칭에 의존하기 때문에 서비스 요청자가 서비스에 대한 정확한 검색이 어렵다.

Ⅲ. 서비스 발견 및 조합 모델

1. 서비스 온톨로지 정의

이 절에서는 이 논문에서 제안하는 상황정보 기반 서비스 검색 및 조합에 사용될 온톨로지를 정의한다. 서비스는 서비스 그라운딩과 서비스 QoS, 서비스 타입 세

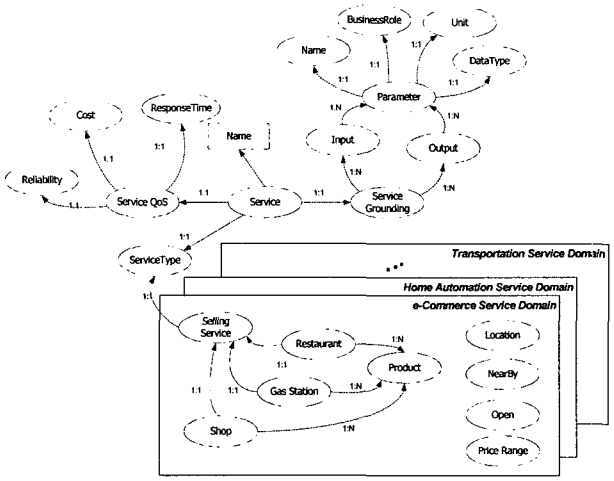

그림 2. 서비스 기술을 위한 온톨로지
Fig. 2. Ontology for Service Description.

개의 요소를 갖는다. 서비스 그라운딩은 입력과 출력으로 구성되며 각각 이름, 역할, 단위, 타입 속성을 갖는 파라미터로 구성된다.

서비스 QoS는 서비스 실행 시간, 신뢰도, 실행 비용으로 구성된다. 마지막으로 서비스 타입은 도메인별로 정의될 수 있다. 그림 2는 전체 서비스 온톨로지의 개요를 설명하고 있다. 그림 2에서는 e-Commerse 서비스 도메인을 예로 구성한 것이다. 예에서 판매 서비스는 레스토랑, 주유소, 샵 서비스가 있으며 위치, 오픈, 가격대 등의 상황 정보를 정의한다. 정의된 온톨로지는 다음에 설명하는 서비스 발견 및 조합을 위해서 사용된다.

2. 서비스 발견

서비스 제공자는 서비스 레지스트리에 그림 3의 온톨로지를 이용해서 서비스를 등록한다. 이 논문에서 설계하는 서비스 관리자는 서비스 요청자의 질의를 분석해서 서비스 타입과 서비스 그라운딩 요소의 매치를 통해서 등록되어 있는 서비스 중 가장 적합한 서비스를 발견한 후 리턴한다. 만약, 적합한 서비스가 발견되지 않으면 서비스 조합을 실행한다. 그림 3은 서비스 발견 과정을 대략적으로 표현한 것이다.

서비스 매칭 방법은 그림 3에서 나타낸 것처럼 서비

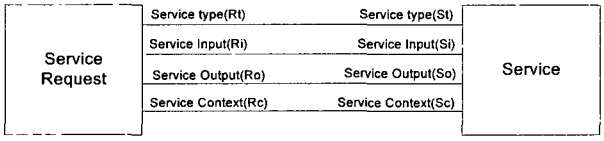

그림 3. 서비스 매칭 방법
Fig. 3. Method os Service Matching.

스 타입, 입력, 출력, 그리고 상황 정보를 이용한다. 서비스 타입 온톨로지 모델은 서비스가 응용에서 가질 수 있는 서비스 타입을 나타낸다. 서비스 도메인마다 다를 수 있으며, 그림 3에서 "Selling Service"에 3가지 서비스로 표현하였다. 서비스의 입력과 출력은 서비스 그라운드에 속한 온톨로지로 서비스는 호출되기 위해서 WSDL 인터페이스로 매핑된다. 실제적인 호출은 인터페이스에 있는 연산의 호출을 위해서 서비스의 연산에 전달하는 파라미터에 의해서 이루어진다. 서비스의 입력과 출력은 WSDL의 메시지로 매핑되는데 이때, 입력과 출력은 파라미터를 포함하며, 파라미터는 이름, 역할, 단위, 데이터 타입을 갖는다. 서비스 상황은 사용자가 원하는 서비스를 찾기 위해서 정의될 수 있다. 이때, 서비스의 위치, 근접성, 서비스 오픈 여부, 제품의 판매 가격 등이 포함된다.

서비스 매칭은 서비스 요청과 등록된 서비스 기술의 온톨로지 매핑에 의해서 이루어진다. 이 논문에서 제안하는 매칭 알고리즘은 서비스 요청자가 원하는 서비스 타입, 서비스 입력과 출력, 상황 정보를 서비스 제공자의 기술과 순서대로 매칭함으로써 정확한 서비스 제공자를 찾아낸다. 이때, 사용자 요청자와 서비스 기술 간에 온톨로지가 얼마나 매칭 되는지를 나타내기 위해서 매칭 정도를 표 1과 같이 정의한다. 표 1에서 Req는 사용자 요구를 나타내며, Svr는 제공되는 서비스를 의미한다. "Exact"는 서비스 요청자와 서비스 제공자의 온톨로지가 정확히 일치하는 경우를 말한다. "Plug-In"은 서비스 요청자가 원하는 것보다 더 큰 범위의 온톨로지를 서비스 제공자가 사용하는 경우를 말한다.

이 두 경우 매칭 모듈은 두 서비스의 온톨로지가 일치한다고 간주한다. 따라서, 서비스 제공자는 서비스 요청자가 원하는 서비스를 제공할 수 있다. "Subsume"은 서비스 요청자가 원하는 내용이 서비스 제공자보다 큰 경우이며, "Intersection"은 양측이 일부분의 온톨로지만을 공유하는 경우이다. 이 두 경우는 모두 서비스 제공자의 일부분만이 서비스 요청자에게 제공될 수 있는 경우이며 따라서 서비스 제공자의 일부분의 기능이 제공

표 1. 온톨로지 매칭 등급
Table 1. Degree of Ontology Matching.

Category	Case	Matching
"Exact"	Req = Svr	Ontology Matching
"PlugIn"	Req ⊂ Svr	
"Subsume"	Req ⊃ Svr	Approximate Matching
Intersection	Req ∩ Svr ≠ ∅	
Dispoint	Req ∩ Svr = ∅	Mismatch

될 수 있음을 나타낸다. 마지막으로 "Dispoint"는 양측이 공유하는 온톨로지가 존재하지 않기 때문에 불일치로 간주한다. 이유는 서비스 타입과 사용자가 요구하는 출력값이 상황 정보나 사용자의 입력값 보다 중요한 기준이기 때문이다. 전체 매칭 정도는 불일치되는 속성의 숫자에 종속적이다. 따라서, 매칭 순서는 서비스 타입, 사용자가 요구하는 출력값, 사용자의 입력값, 상황 정보 순을 따른다.

3. 서비스 조합

사용자가 원하는 서비스가 검색되지 않으면 서비스들을 조합해서 원하는 서비스를 제공해 줄 수 있다. 이때, 시스템은 두 서비스가 조합이 가능한지 여부를 판단 할 수 있어야 한다. 두개의 서비스가 조합 가능한지를 알기 위해서 먼저 하나의 서비스가 다른 서비스의 연산을 호출 할 수 있는지 판단한다. 연산의 호출은 메시지 전달에 의해서 이루어지며, 이를 위해서 서비스를 구성하는 서비스와 메시지의 상호운영성을 판단해야 한다.

메시지 상호운영성 : 서비스간의 상호 운용성은 먼저 메시지 교환에 의해서 이루어진다. 메시지는 파라미터들로 구성되며 특정 데이터 타입을 갖는다. 전송하는 파라미터는 이를 수신하는 서비스의 파라미터와 호환되어야 한다. 모든 파라미터는 잘 정의된 시멘틱을 가진다. 메시지 매칭 방법은 직접인 매칭(direct mathching)과 간접적인 매칭(indirect matching)이 있다. 두개의 데이터 타입이 같으면 직접적 매칭이 되고, 하나의 데이터 타입이 다른 데이터 타입으로부터 상속되었으면 간적접인 매칭이 된다. 다음 알고리즘은 두 메시지가 상호 운용가능한지를 판단하는 알고리즘이다. 알고리즘 1에서 Pi는 메시지에 포함되는 파라미터의 이름이며, T는 데이터 타입, U는 단위, R은 역할을 나타낸다.

알고리즘 1. 메시지 상호운용성
Algorithm 1. Message Compatibility.

```
Input : Mi = {Pi, Ti, Ui, Ri}, Mj = {Pj, Tj, Uj, Rj}
Output : True / False
Begin message_compoable
  matchedList = false;
  for eache param pik ? Pi do
    found = false
    for each param pjl ? Pj | pjl is not matchedList do
      if (T(pik) = T(pjl) or T(pjl) is derived from
        T(pik)) and (U(pik) = U(pjl)) and
      R(pik) = R(pjl)) then
        found = true;
        matched = matched ? pjl
      end if
      if found is not true then return false;
    end for
```

```
    return true;
  end for
End.
```

연산 상호운영성 : 연산과의 상호 운영성은 두 연산 간의 모드에 따라서 결정된다. 각 연산은 표 2에 기술되어 있는 모드를 갖는다. 예를 들어서, 'one-way" 모드를 갖는 연산은 입력 메시지를 가지나 출력 메시지를 갖지 않는다 따라서, notification 모드를 갖는 연산과 쌍을 이룰 수 있다. 반대로, "notification" 모드를 갖는 연산은 출력만 있고 입력을 갖지 않으므로, "one-way" 연산자와 쌍을 이룰 수 있다.

"Solicit-response"는 입력을 받은 뒤에 출력을 하기 때문에 "request-response"와 쌍을 이룰 수 있으며, "request-response"는 메시지를 출력한 후에 입력을 받기 때문에 "solicit-response" 연산과 쌍을 이룰 수 있다.

조합된 서비스의 QoS 평가 : 조합된 서비스들은 트랜잭션의 시간이 길 수 있다. 따라서, 가장 최적화된 실행 시간을 갖는 조합 계획을 갖는 것은 매우 중요하다. 서비스의 조합 계획이 복수개가 발견되는 경우, 각각의 서비스 조합 계획 중 최적의 계획을 선택해야 한다. 이를 위해서 적절한 QoS 측정 방법이 필요하다. 이 논문에서는 서비스의 신뢰성, 실행 시간과 실행 비용의 측면으로 평가하며 평가 방법은 다음과 같다. (1)은 전체 서비스 조합의 QoS값으로 이 값을 비교해서 최적의 서비스 조합을 찾는다. QtValue값은 서비스의 신뢰성에 비례하고 실행 시간과 비용에 반비례한다.

$$QtValue(S) = \frac{m3 * \dim(S, \mathrm{Re}liabilty)}{m1 * \dim(S, Time) * m2\dim(S, Cost)} \quad (1)$$

$$\dim(S, \dim) = \sqrt[3]{Min(S, \dim) * Avr(S, \dim) * Max(S, \dim)} \quad (2)$$

$$\mathrm{Re}liabilty = (1 - \frac{failure}{totalExecution}) * 100(\%) \quad (3)$$

// sequence execution plan

$$\dim(CompositeS, Time) = \sum_{i=1} \dim(Si, Time) \quad (4)$$

$$\dim(CompositeS, Cost) = \sum_{i=1} \dim(Si, Cost) \quad (5)$$

// parallel execution plan

표 2. 연산 조합 모드
Table 2. Composition Mode of Operation.

Operation mode	Description
One-way	Input message, No output message A pair with notification
Notification	No Input message, Output message A pair with one-way
Solicit-response	Input Output. A pair with request-response
request-response	Output Input. A pair with solicit-response

$$\dim(CompositeS, Time) = Max[\dim(Si, Time)] \quad (6)$$

$$\dim(CompositeS, Cost) = \prod_{i=1} \dim(Si, \mathrm{Re}liability) \quad (7)$$

이때, 서비스의 조합이 순차적인 경우와 병렬적인 경우에는 실행 시간과 비용을 측정하는 방법이 차이가 있다. 순차적인 경우의 실행 시간은 각 서비스의 실행 시간의 합으로 표현되지만 병렬적인 경우에는 서비스의 실행 시간 중 최대값으로 결정된다. 또, 순차적인 경우의 실행 비용은 서비스의 비용의 합으로 표현된다. 조합된 서비스의 QoS는 QtValue(S)로 나타나며 신뢰도에 비례하고 실행 시간과 비용에 반비례한다.

Ⅳ. 시스템 설계 및 구현

이 장에서는 Ⅲ장에서 제시한 모델을 바탕으로 상황 인식 응용을 위한 서비스 발견 및 조합을 가능하게 하는 서비스 관리 시스템을 설계하고 구현한다. 전체 시스템은 자바 기반으로 개발되며 OSGi 프레임워크상에서 동작될 수 있도록 설계한다. 전체 시스템의 구성도는 그림 4와 같다. 이 논문에서 설계한 서비스 관리 시스템은 상황 인식 응용으로부터 전달받은 서비스 검색 요청을 해석해서 적절한 서비스를 리턴하거나 조합된 서비스 리스트를 리턴하는 역할을 수행한다. 이를 위해서 서비스 관리 시스템은 서비스 요청 파서(Request Parser), 서비스 조합기(Service Composer), QoS 평가기(QoS Evaluator), 상황관리자(Context Manager), 매치메이커(Matchmaker), 서비스 레지스트리(Service Registry), 온톨로지 추론기(Ontology Reasoner), 통지관리자(Notification Manager)로 구성된다.

요청 파서는 상황 인식 응용이 전달한 서비스 요청을 분석해서 이를 서비스 조합기로 전달한다. 서비스 조합기는 서비스 매치메이커를 통해서 서비스를 검색한다. 매치메이커는 온톨로지 추론기와 상황 정보를 관리하는

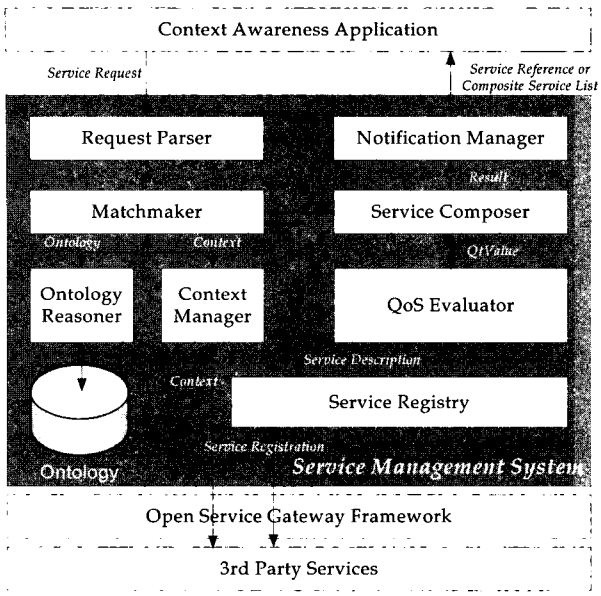

그림 4. 서비스 관리 시스템
Fig. 4. A Overview of Service Management System.

상황 관리자를 통해서 두 서비스를 비교하고 적합한지를 결정한다. 만약, 적합한 서비스가 없으면 서비스 조합기는 기존 서비스를 조합해서 사용자가 원하는 서비스 결과를 얻기 위한 조합 계획을 생성하고 이를 QoS 평가기를 통해서 가장 적합한 조합 계획을 통지 관리자를 통해서 상황 인식 응용에 리턴한다. 온톨로지 추론기는 이 연구의 개발 범위에 포함되지 않기 때문에 Jena2.0를 활용하였으며[14], OSGi 게이트웨이는 Knopflerfish 1.3.3을 이용하여 개발하였다[15]. 알고리즘 2는 서비스 매칭메이커가 두 서비스를 비교해서 적합한지를 판단하기 위한 방법을 기술하고 있다.

알고리즘 2. 서비스 매칭 방법
Algorithm 2. A Method of Service Matching.

```
Input : Req, Svr_List
Output : Svr_List
Begin
  Svr' = selectService( Req.type, Svr_List );
  Svr'' = selectService( Req.output, Svr' );
  ForAll s in Svr'' Do
  Begin
    si = getInput(s);
    IF isExits( si, Req.input ) then
      ExactCandidate.append( s );
      ForAll  svr in ExactCandidate Do
      Begin
        ForAll attr in Ra Do
        Begin
          If evalContextRule( s, attr ) Then
          ExactResult.addList( s );
        End ForAll
      End ForAll
    Else
      ApproixiateCandidate.append( s );
      ForAll attr in Ra Do
      Begin
        If evalContextRule( s, attr ) Then
          ApproxiateResult.addList( s );
        Return Result;
    End IF
  End ForAll
  Return( ExactResult, ApproxiateResult );
End.
```

서비스 레지스트리에 저장되어 있는 서비스 리스트에서 사용자 요구의 타입이 일치하는 서비스를 일차로 추출한다. 이렇게 추출된 서비스 리스트에서 서비스 요청의 출력값과 일치하는 서비스 리스트를 추출한 후 입력을 비교한다. 이때, 레지스트리에 저장되어 있는 입력 타입이 요청된 입력 타입과 정확히 일치하는 경우와 상위 개념일 경우에는 입력 타입이 일치하는 것으로 간주한다. 입력 타입까지 일치한 경우에는 상황 정보의 attr 값을 이용해서 가장 적절한 서비스를 추출한다.

이때, 적절한 서비스를 찾지 못한 경우에는 서비스를 조합한다. 먼저, 사용자의 요구로부터 입력값을 추출하고, 원하는 출력값의 리스트를 작성한다. 그리고 사용자가 원하는 출력을 갖는 서비스를 먼저 찾아서 서비스 체인의 시작으로 잡는다. 서비스 체인의 시작 서비스의 입력을 출력으로 하는 서비스를 다시 찾은 후, 찾아진 서비스의 입력이 사용자의 입력과 일치할 때까지 반복해서 서비스를 체인에 추가한 후, 서비스의 리스트를 리턴한다. 만약, 더 이상 서비스가 서비스 레지스트리에 존재하지 않으면 서비스의 조합이 불가능한 것으로 판단하고 Null을 리턴한다. 알고리즘 은 서비스의 조합 과정을 나타낸 것이다.

알고리즘 3. 서비스 조합 방법
Algorithm 3. A Method of Service Composition.

```
Input: weHave = {inputs provided by User Requirement};
Output : weWant = {outputs desired by
User Requirement};
Begin
  chainStartList = findServiceChainStart(weWant);
  while nextChainStart(chainStartList) do
    chainList.add(new chain(chainStart));
    MakeServiceChain(chain, chainStart.input,
    chainStart.output);
  End While
  Function MakeServiceChain(chain, input, output)
  Begin
    svcs = getServicesHavingOutput(chainLast.input);
    if svcs.count = 1 then
      chain.add(svcs)
    else if
      foreach service in svcs
        chain.add(new chain(service));
        MakeServiceChain(chain, service.input,
        service.output);
      end foreach
    if input in weHave then
      return chain;
    end if
    return null; // no chain found
  End function
End
```

Ⅲ. 실 험

이 장에서는 Ⅳ장에서 설계한 상황 인식 응용을 위한 서비스 관리 시스템이 서비스 조합 및 검색 기능이 제대로 동작하는지를 보이기 위해서 서비스 질의를 위한 프로토타입을 개발하였다. 개발한 프로토타입은 JSP로 개발되었으며 사용자에게 서비스를 질의할 수 있는 웹 기반 GUI를 제공한다. 사용자는 GUI를 통해서 원하는 서비스를 기술하면, 프로토타입은 웹서비스 클라이언트를 이용해서 SOAP으로 변환 후 OSGi 프레임워크상의 서비스 관리 시스템에 전달한다. 먼저, 시스템의 동작 과정이 적합한지 보이기 위해서 다음 표와 같은 샘플 서비스를 입력한 후 서비스의 조합 과정을 보인다.

서비스 검색 : 서비스 검색은 사용자의 질의 내에 있는 서비스 타입과 입력, 출력, 상황 정보를 이용해서 정확히 일치하는 서비스와 정확히 일치하지는 않으나 이용 가능한 서비스의 목록을 리턴한다. 실험에서는

표 3. 서비스 레지스트리에 등록된 서비스 정보
Table 3. Service Description in Service Registry.

Service Type	Service Name	Input	Output	QoS (Max/Avg/Min)		
				Time	Cost	Reliabilty
Shop	Srv1	a, b	d	10/12/13	8/10/11	71
Shop	Srv2	a, b	e	11/13/14	12/13/14	83
Shop	Srv3	a	e, f	12/14/15	13/14/15	62
GasStation	Srv4	b	i	8/9/10	13/14/17	94
GasStation	Srv5	f	c, e	5/7/8	20/24/28	83
GasStation	Srv6	e, g, h	j, h	10/11/12	6/8/10	72
Shop	Srv7	a	d, e	13/14/15	11/13/14	81
Shop	Srv8	a	e	20/21/24	15/16/18	57
AirplainRsv	Srv9	c, e	q	15/17/19	12/16/17	84
AirplainRsv	Srv10	s	q	10/11/16	10/10/11	95
AirplainRsv	Srv11	x	p	13/11/12	8/12/11	80
CarRsv	Srv12	p	r	12/8/9	8/10/13	75
CarRsv	Srv13	q	r	10/12/8	6/4/11	95
HotelRsv	Srv14	r	x	7/6/11	3/8/6	93

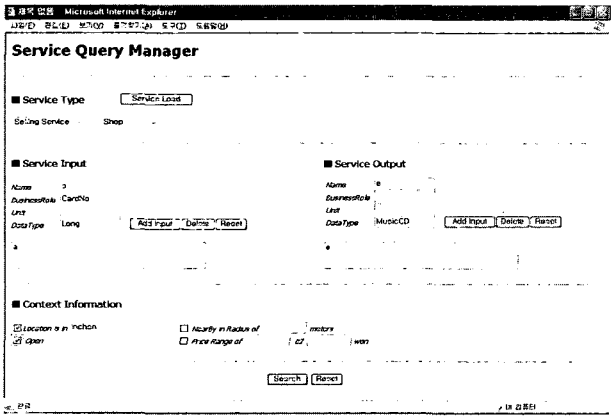

그림 5. 서비스 질의 화면
Fig. 5. Service Query.

ServiceType이 'Shop'이고, 입력이 'a', 출력이 'e'이고, 서비스 제공장소가 'Inchon"인 서비스를 검색한다. 그림 5는 프로토타입에서 제공하는 GUI를 이용해서 질의하는 화면이다.

질의된 내용은 SOAP 객체로 변환 되서 서비스 관리 시스템에 전달되고 서비스 관리 시스템은 이를 OWL로 변환한 후 온톨로지 매칭과 상황 정보를 통해서 서비스를 리턴한다. 그림 은 정확히 일치하는 서비스와 대략적으로 일치하는 서비스의 목록을 보여준다. Svr2는 사용자가 원하는 정확한 출력을 제공하나 입력 파라미터 중 하나가 불일치함을 보여준다.

서비스 조합 : 사용자의 질의에 일치하는 서비스가 없는 경우에는 등록되어 있는 서비스의 조합을 통해서 사용자에게 원하는 결과를 얻을 수 있도록 할 수 있다. 실험에서는 서비스 타입이 "HotelReservation' 서비스 중 입력이 e이고 출력이 x인 서비스를 원하는 경우 서비스 레지스트리에는 정확한 서비스를 검색할 수 없음으로 서비스의 조합을 통해서 사용자에게 x값을 전달한다. 그림 7은 서비스이 조합 결과인 서비스의 리스트를

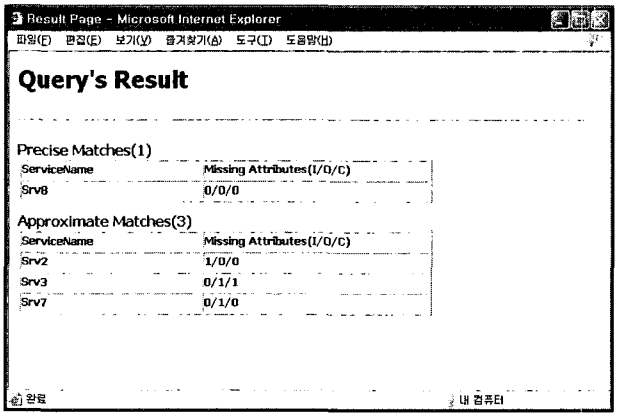

그림 6. 서비스 검색 결과 화면
Fig. 6. A Result of Service Matching.

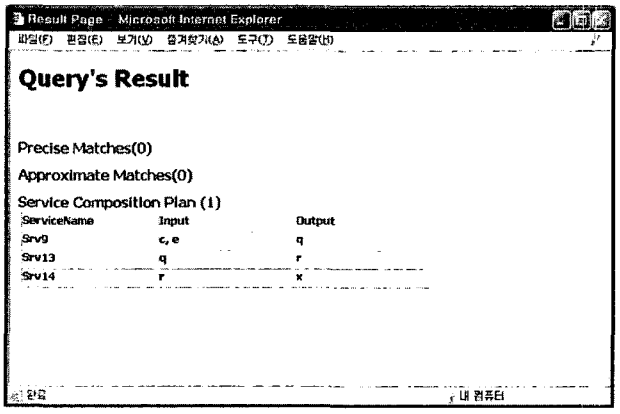

그림 7. 서비스 조합 결과
Fig. 7. A Result of Service Composition.

보여주는 화면이다.

이 논문에서 제안한 상황 정보를 고려한 서비스 검색 방법의 검색 정확성을 확인하기 위해서 서비스 레지스트리에 등록된 서비스의 수를 50개 100개, 200개로 증가하면서 실험을 실시하였다. 실험 결과 구문 검색 방식과 온톨로지 검색 방식과 비교해서 검색 정확도가 증가하였으며, 검색율이 증가였다. 그 이유는 서비스가 많을수록 사용자의 질의에 따라서 조합할 수 있는 서비스의 개수가 많아지기 때문인 것으로 판단된다. 검색 시간은 단일 서비스 검색인 경우에는 1.7초정도 소요되었다. 따라서, 하드 실시간 시스템의 경우가 아닌 일반적인 실시간 시스템에서는 사용하는데 별 문제가 없을 것으로 보인다.

실험을 통해서 이 논문에서 설계한 서비스의 검색 및 조합을 지원하는 서비스 관리 시스템이 올바르게 동작함을 알 수 있었으며, 서비스의 정확도 및 검색율이 기존의 구문 검색과 온톨로지 기반 검색보다 향상됨을 알 수 있었다.

Ⅳ. 결 론

이 논문에서는 서비스 등록 및 검색을 위해 사용되는 온톨로지를 제안하였으며, 상황 인식 응용 개발을 위한 상황 정보에 기반한 서비스 관리 시스템을 설계하였다. 제안한 온톨로지는 서비스 타입과 입력과 출력에 관련된 서비스 그라운딩, 조합된 서비스의 QoS 평가를 위한 클래스들로 구성하였다. 또한, 서비스 검색 및 서비스 조합에 관련된 모델 및 알고리즘을 제안하였고 이를 바탕으로 서비스 관리 시스템을 설계하였다. 설계한 시스템은 등록된 서비스를 관리하는 서비스 레지스트리, 사용자 요구를 분석하기 위한 파서, 서비스를 조합하기 위한 서비스 조합기, 두 서비스간의 매칭 여부를 판단하는 매치메이커, 온톨로지 추론을 위한 온톨로지 추론기와 상황 정보를 관리하는 상황 관리기, 조합된 서비스의 성능을 평가하기 위한 QoS 평가기와 검색된 결과를 사용자에게 전달하기 위한 통지 관리자로 구성된다. 전체 시스템은 OSGi 프레임워크에서 동작할 수 있게 설계되었기 때문에 다양한 미들웨어와 프로토콜 등을 통한 시스템 연결이 용이하다.

또한, 실험에서는 서비스 질의 관리기 프로토타입을 구현하고 이를 이용한 실험을 통해서 설계한 시스템의 타당성과 성능을 평가하였다. 실험 결과 서비스의 검색 및 조합을 정확히 수행하였으며, 기존 구문 검색 방법과 온톨로지 기반 검색에 비해서 상황 정보와 서비스 조합을 이용함으로써 검색 정확도와 검색율을 향상되었음을 알 수 있었다. 그러나, 온톨로지 추론으로 인한 서비스 검색 시간이 기존 검색 방법에 비해서 증가하였으나 이에 대한 해결방안을 제시하지 못하였다.

향후 연구과제는 동적인 온톨로지의 확장과 서비스 등록을 위한 시스템의 추가 연구와 온톨로지 추론을 보다 효과적으로 수행할 수 있는 추론 엔진에 대한 연구가 필요하다. 또한, 이 논문에서 제시한 서비스 관리 시스템을 우리의 전체 연구의 과제인 상황 인식 응용 프레임워크에 통합하는 작업을 수행할 예정이다.

참 고 문 헌

[1] K. Romer, T. Schoch, F. Mattern and T. Dubendorfer, "Smart Identification Frameworks for Ubiquitous Computing Application," IEEE International Conference on Pervasive Computing and Communication, 2003.

[2] P. De, K. Basu and S. K. Das, "An Ubiquitous Architectural Framework and Protocol for Object Tracking using RFID Tags," Proceedings of the First Annual International Conference on Mobile and Ubiquitous System:Networking and Services, 2004.

[3] B. Wang, J. Bodily and S. K. S. Gupta, "Supporting Persistent Social Groups in Ubiquitous Computing Environments Using Context-Aware Ephemeral Group Service," IEEE Annual Conference on Pervaive Computing and Communications, 2004.

[4] T. Gu, H. K. Pung and D. Q, Zhang, "Toward an OSGi-Based Infrastructure for Context-Aware Applications," IEEE Pervasive Computing, 2004.

[5] D. Martin et al. OWL-S: Semantic Markup for Web Service Technical report, DAML Consortium, http://www.daml.org/services/owl-s/1.0/owl-s.pdf, 2004.

[6] D. Marples and P. Kriens, "The Open Service Gateway Initiative: An Introductory Review," IEEE Communications Magazine, vol 39, no. 12, Dec. 2001.

[7] Li Gong, "A Software Architecture for Open Service Gateways," IEEE Internet Computing, Jan. 2001.

[8] T. Gu, H. K. Pung and D. Q. Zhang, "Toward an OSGi-Based Infrastructure for Context-

Aware Applications," IEEE Pervasive Computing, 2004.
[9] S. Helai et al. "The Gator Tech Smart House: A Programmable Pervasive Space," IEEE Computer, 2005.
[10] A. K. Dey, D. Salber and G. D. Abowd, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications," HCI Journal, 2001.
[11] H. Chen, An Intelligent Broker Architecture for Context-Aware Systems, PhD thesis, University of Maryland Baltimore County, 2003.
[12] D. Fensel, C. Bussler, "The Web Service Modeling Framwork WSMF," White Paper and International Report, www.cs.vu.nl/swws/download/ wsmf.paper.pdf, 2002.
[13] M. Paolucci, K. Sycara, "Autonomous Semantic Web Services," IEEE Computer Society, 2003.
[14] Jena: A Semantic Web Framwork for Java, http://jena.sourceforge.net/.
[15] Knopflerfish: A OSGi Framework, http://www.knopflerfish.org/.

저 자 소 개

이 승 근(학생회원)
1996년 인하대학교 전자계산공학과
1998년 인하대학교 전자계산공학과(공학석사)
2000년 인하대학교 컴퓨터정보공학과 박사수료
2000년~2005년 (주)하이켐텍 기술연구소 책임연구원
<주관심분야 : 홈네트워크, 상황인식, 웹서비스, 시멘틱웹>

이 정 현(평생회원)
1977년 인하대학교 전자공학과
1980년 인하대학교 대학원 전자공학과(공학석사)
1988년 인하대학교 대학원 전자공학과(공학박사)
1979년~1981년 한국전자기술연구소 시스템연구원
1984년~1989년 경기대학교 교수
1989년~현재 인하대학교 컴퓨터공학부 교수
<주관심분야 : 자연어처리, HCI, 정보검색, 음성인식, 음성합성, 컴퓨터구조, 홈네트워크>

임 기 욱(평생회원)
1977년 인하대학교 전자공학과
1987년 한양대학교 전자계산학 석사
1994년 인하대학교 전자계산학 박사
1977년~1983년 한국전자기술연구소 선임연구원
1983년~1988년 한국전자통신연구소 시스템소프트웨어 연구실장
1988년~1989년 미 캘리포니아 주립대학(Irvine) 방문연구원
1989년~1996년 한국전자통신연구원 시스템연구부장, 주전산기(타이컴)Ⅲ,Ⅳ개발사업 책임자
1997년~1999년 정보통신연구진흥원 정보기술전문위원
2001년~2003년 한국전자통신연구원 컴퓨터소프트웨어 연구소장
2000년~현재 선문대학교 컴퓨터정보학부 교수
<주관심분야: 실시간데이터베이스시스템, 운영체제, 시스템구조>