I

유비쿼터스 환경에서 커뮤니티 컴퓨팅 모델을 위한 멤버 어플리케이션 프레임워크

아주대학교 ▌김한욱* · 이정태** · 김민구**

1. 서 론

유비쿼터스 컴퓨팅은 1988년 마크 와이저에 의해 "일상의 물리적 환경 도처에 컴퓨팅 디바이스가 편재되어 사용자의 상황에 적합한 서비스의 제공을 추구하는 새로운 컴퓨팅 패러다임"으로 소개 되었다[1]. 이후에 고성능의 유무선 네트워킹 기술과 소형화 및 처리 능력이 향상된 컴퓨팅 개체들이 인간생활에 보급됨에 따라 이를 이용한 보다 나은 서비스 제공에 대한 관심이 높아지고 있다.

유비쿼터스 컴퓨팅 환경은 처리 능력, 사용자 상호작용 방법과 이동성 등이 서로 다른 컴퓨팅 개체들이 유무선의 다양한 네트워킹 방법으로 복잡하게 구성되어지고 있으며 서비스의 대상 범위 역시 하나의 마을 내지는 도시 전체 등으로 확대되고 있다[2,3].

유비쿼터스 컴퓨팅 서비스의 대상 범위가 대규모화 되면서 사전에 예측하지 못한 새롭게 발생하는 요구사항에 대한 서비스를 사용자의 기존 서비스 사용을 방해하지 않으면서 배포 가능하게 하는 서비스 진화(Evolution) 기술 및 더욱 복잡해진 이질적 환경의 적응성(Adaptability) 및 확장성(Scalability) 확보 기술이 향후 유비쿼터스 컴퓨팅 서비스의 중요한 기술로 대두되고 있다[4].

그 중 하나의 접근 방법으로, 새로운 커뮤니티 컴퓨팅(Community Computing) 개념이 제시 되고 있다. 커뮤니티 컴퓨팅의 개념은 1990년대부터 이미 사람들이 특정한 관심과 목표를 위해 커뮤니티를 형성하여 협력하는 경우 이를 지원하기 위한 컴퓨팅 개념으로 소개되었고, 현재도 인간 중심의 커뮤니티를 지원하기 컴퓨팅 기술에 대한 연구가 활발히 진행되고 있다. 이에 반하여 본 연구에서의 커뮤니티 컴퓨팅은 유비쿼터스 컴퓨팅 환경에서 존재하는 사람뿐만 아니라 센서, 프린터, 핸드폰, 컴퓨터 등의 이기종의 개체들이 특정 상황에서 특정 목표 달성을 위한 서비스 실현을 위하여 커뮤니티를 형성하여, 이들 간에 협업을 통해 필요한 서비스를 제공함 을 의미한다. 이는 기존 인간 커뮤니티를 위한 커뮤니티 컴퓨팅 개념의 확장된 컴퓨팅 개념으로 볼 수 있다. 이러한 확장된 커뮤니티 컴퓨팅 기술을 활용하는 경우 향후 유비쿼터스 컴퓨팅 서비스에서 요구되는 서비스의 진화 및 이질적 환경에 대한 적응성 측면에서 여러 가지 장점이 있다고 보고 현재 연구를 진행 중에 있다[5]. 향후 본고에서의 커뮤니티 컴퓨팅은 후자를 의미한다.

커뮤니티 컴퓨팅에서는 유비쿼터스 컴퓨팅 서비스 대상 공간에 존재하는 컴퓨팅 개체들과 이들의 그룹으로 필요한 서비스 제공을 위하여 만들어진 커뮤니티들의 집합을 소사이어티라 하며, 이의 관리를 담당하는 소프트웨어를 소사이어티 매니저라 한다. 소사이어티 내에서 서비스를 제공하기 위해 커뮤니티가 생성되는데, 커뮤니티를 운영하고 커뮤니티에 소속된 컴퓨팅 개체와 커뮤니티 상황을 관리하는 소프트웨어를 커뮤니티 매니저라 한다. 커뮤니티 컴퓨팅 시스템에서 유비쿼터스 컴퓨팅 서비스의 실현을 위해서는 소사이어티 매니저와 커뮤니티 매니저가 필요하고 각 컴퓨팅 개체에 배포되는 어플리케이션이 필요하게 된다.

이러한 커뮤니티 컴퓨팅 모델을 이용하여 일부 유비쿼터스 컴퓨팅 서비스에 대한 커뮤니티 컴퓨팅 시스템을 JADE(Java Agent Development Framework)[7] 플랫폼을 기반으로 개발함으로써 이 모델의 구현 가능성을 입증한 연구가 있었으며[6], 현재는 특정 플랫폼이 아닌 이질적인 플랫폼으로 구성된 공간에서도 작동 가능하며 서비스의 진화 까지도 지원하기 위한 방향으

* 학생회원
** 종신회원
† 본 연구는 지식경제 프론티어 기술개발사업의 일환으로 추진되고 있는 지식경제부의 유비쿼터스컴퓨팅 및 네트워크 원천기반 기술개발사업의 지원에 의한 것임.

로 이 기술을 확장하기 위한 연구가 진행되고 있다. 이러한 연구의 한 방향으로 커뮤니티의 컴퓨팅 개체에 배포될 어플리케이션이 플랫폼 간의 이질성에 적응적이며, 새로운 서비스가 요구 되면 이를 위한 커뮤니티를 기존 작동 중인 커뮤니티와 충돌 없이 생성할 수 있는 기능을 지원하는 어플리케이션 프레임워크가 개발되고 있는데, 이를 멤버 어플리케이션 프레임워크라 한다.

이 프레임워크는 유비쿼터스 환경에서 커뮤니티 컴퓨팅 모델을 기반으로 서비스를 구현하는 경우 컴퓨팅 개체의 이질성(Heterogeneity)과 컴퓨팅 개체의 수정 없이 서비스를 추가하거나 변경할 수 있는 확장성(Scalability)을 지원한다.

본고의 구성은 2절에서 커뮤니티 컴퓨팅 모델을 히 설명하고, 3절에서는 멤버 어플리케이션 프레임워크의 요구사항을 분석하고, 4절에서는 멤버 어플리케이션 프레임워크의 구조를 제시하고, 5절에서는 시나리오를 기반으로 구현 사례를 설명하고, 6절에서는 관련 연구에 대해 소개 한 후, 7절에서는 결론으로 마무리 한다.

2. 커뮤니티 컴퓨팅

본 연구에서는 유비쿼터스 공간을 "서로 이질적인 개체들이 동적으로 그룹을 생성하여 상호 협력할 수 있는 통신이 가능한 환경"으로 정의하고 이를 'uT-공간(uT-space)'이라 부르며, uT-공간 내에 존재하는 컴퓨팅 개체를 'uT-개체(uT-entity)'라고 부른다. uT-개체에는 시스템 프로세스나 응용프로그램과 같이 다양한 종류의 소프트웨어가 포함되고, 컴퓨터, 센서, 디지털 가전 등과 같이 다양한 하드웨어들도 포함된다. 사람도 uT-개체들을 사용하고 uT-개체들과 상호 작용하므로 uT-공간의 구성요소라고 할 수 있다[5].

2.1 커뮤니티 컴퓨팅의 개념

커뮤니티 컴퓨팅에서 소사이어티나 커뮤니티를 구성하는 uT-개체를 '멤버(member)'라고 한다. 멤버에 대해서는 2.2절에 자세히 설명한다.

그림 1은 uT-공간을 커뮤니티 컴퓨팅 관점에서 추상화한 그림이다. 존재 가능한 uT-개체와 커뮤니티의 생성이 가능한 공간을 '소사이어티(Society)'라 하며, 커뮤니티 컴퓨팅 시스템이 처음 운용 되는 시점에는 어떠한 커뮤니티도 없이 커뮤니티의 멤버가 될 수 있는 소사이어티 멤버들만 존재 한다. 특정 uT-개체가 소사이어티 멤버로 등록되는 과정은 자신의 식별자 및 능력(Capability)등 관련된 속성 정보를 소사이어티에 알림으로써 이루어진다.

커뮤니티의 생성은 소사이어티를 구성하는 멤버가 소사이어티에 요청하여 이루어지거나, 소사이어티의 자체 판단에 의해 이루어질 수 있다.

커뮤니티는 그림 1과 같이 소사이어티에 참여하고 있는 멤버들 중에서 해당 서비스가 요구하는 역할(Role)을 담당할 수 있는 멤버들을 선택(Cast)하여 커뮤니티를 구성한다.

커뮤니티의 멤버들이 역할을 수행하는 과정에서 수집되는 정보를 멤버의 '컨텍스트(Context)'라 하며, 이를 커뮤니티에 전송하고 커뮤니티는 이를 바탕으로 커뮤니티의 전체적인 '상황(Situation)'을 판단한다.

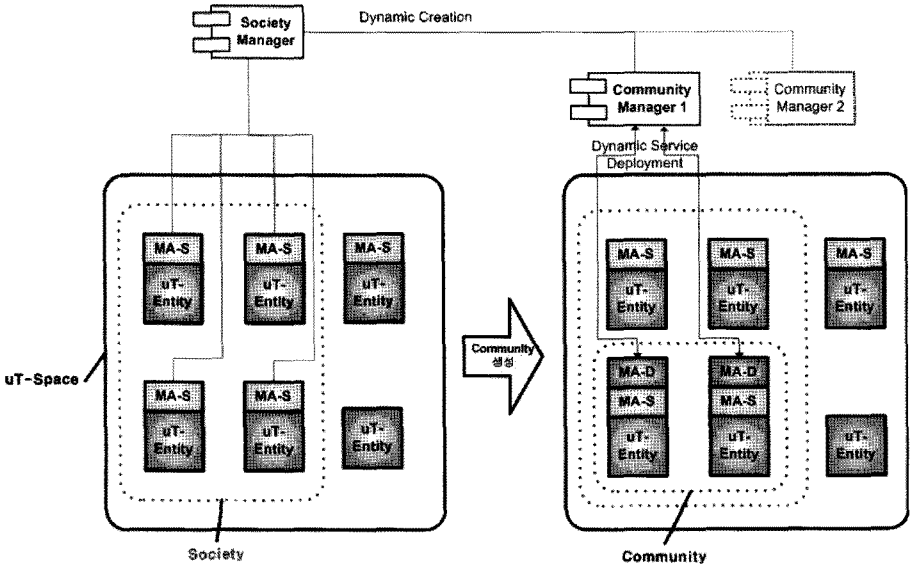

그림 1 uT-공간과 멤버어플리케이션의 개념도

커뮤니티의 상황이 멤버들에게 전달되고, 각 멤버는 상황을 기반으로 자신이 수행하고 있는 역할에 따라 그 상황에서 수행하기로 약속된 행위(Behavior)를 수행하게 된다. 이렇게 수행되는 멤버의 행위는 다시 커뮤니티 상황의 변화를 유도한다.

커뮤니티가 목표(Goal)로 설정된 상황이 발생했음을 감지하면, 필요한 서비스의 수행이 종료된 것으로 간주하고 커뮤니티 매니저는 멤버를 커뮤니티로부터 탈퇴시키고, 해당 커뮤니티는 소사이어티 매니저에 의하여 소멸된다[8].

2.2 멤버와 멤버어플리케이션

멤버는 커뮤니티 컴퓨팅 시스템을 이루는 구성요소로서, uT-개체가 자신의 고유 기능(예를 들어, 에어컨이 설정된 온도에 시원한 바람의 양을 조절하거나 엘리베이터가 지정 층수로 이동하거나 하는 기능들이 여기에 속한다)을 수행하고 있다가 특정 서비스 목표를 위한 커뮤니티가 구성되면, 해당 목표를 달성하기 위해 자신의 역할을 수행한다. 이 uT-개체에 배포되어 커뮤니티 컴퓨팅 서비스를 지원하는 어플리케이션을 '멤버 어플리케이션(Member Application)'이라 한다.

멤버 어플리케이션은 그림 2에서 처럼 구조 상 크게 Static Package와 Dynamic Package의 두 부분으로 나누어진다. 그리고 실행 시 다음과 같은 세 가지 상태 중 어느 하나의 상태에 있게 되며, 각 상태에 따라 멤버 어플리케이션의 구조 및 작동 방식도 달라진다.

① 소사이어티 멤버가 되기 전 상태(소사이어티 멤버 후보 상태)
② 소사이어티 멤버로 등록되었으나 특정 커뮤니티의 역할을 맡고 있지 않은 상태(소사이어티 멤버 상태)
③ 소사이어티 멤버로 등록되었으며 특정 커뮤니티의 역할을 맡고 있는 상태(커뮤니티 멤버 상태)

멤버 어플리케이션은 상태①과 상태②에서는 Static Package 만으로 수행되고, 상태③에서는 Static Package와 Dynamic Package가 연결된 상태에서 수행되게 된다.

멤버 어플리케이션을 Static Package와 Dynamic Package로 나눈 이유는 커뮤니티 생성 전에는 uT-공간의 이질적인 uT-개체들이 서로 다른 고유의 기능을 수행하다가, 커뮤니티 생성 후에는 커뮤니티 서비스의 목표를 수행할 역할을 부여 받아서 협업해야 하기 때문에 두 부분으로 나누게 된다.

Static Package의 구조는 uT-공간 내의 이질적인 uT-개체를 지원하기 위해서 커뮤니티 컴퓨팅을 지원할 공통된 부분과 이질적인 특성에 따라 서로 다르게 선택될 부분으로 나누어 져야 한다. 공통된 부분은 커뮤니티 컴퓨팅에서 소사이어티 및 커뮤니티에 등록하는 기능, 커뮤니티 역할-구성원 바인딩(Role-member Binding)에 대한 응답 기능과 역할 다운로딩 기능이 있고, 선택될 부분은 uT-개체의 네트워킹 방법, 센싱 방법, 컨텍스트 관리 방법과 개체 고유의 기능이 있으며 이들은 개체의 특성에 맞게 선택되어 공통된 부분과 연결 되어야 한다.

Dynamic Package는 커뮤니티 생성 될 때, 커뮤니티 역할-구성원 바인딩 과정에서 커뮤니티로부터 동적으로 다운로딩 되어 배포된다. 커뮤니티 생성 후에는, 커뮤니티의 상황을 기반으로 수행할 역할에 따라 그 상황에서 수행하기로 약속된 행위를 수행하여 커뮤니티 서비스의 목표를 달성하게 된다. 이질적인 uT-개체들이 특정 커뮤니티 서비스에 해당하는 Dynamic Package를 동적 다운로딩과 역할 수행을 하게 됨으로써 uT-개체마다 서로 다른 역할을 수행하여 특정 커뮤니티 서비스 목표를 달성하며, 새로운 Dynamic Package를 배포 받아 해당 역할을 수행하면 새로운 커뮤니티 서비스로의 확장도 가능하다.

멤버 어플리케이션의 구조를 Static Package와 Dynamic Package로 나눔으로써, 상태①과 상태②에서는 uT-공간 내의 이질적인 uT-개체들을 지원하여 서로 다른 자신의 고유의 기능을 수행할 수 있고, 상태③으로 변화되면 특정 커뮤니티 서비스의 역할을 부여 받아 서비스 목표를 달성할 수 있으며 새로운 서비스에 대한 확장성을 지원한다.

3. 요구사항

uT-공간 내에서 이종의 uT-개체에 배포될 멤버 어플리케이션이 플랫폼 간의 이질성을 적응하고 새로운 서비스 요구에 대한 확장성을 지원하기 위해서는 멤버 어플리케이션 프레임워크가 필요하며, 이는 아래 네 가지 요구사항을 만족해야 한다.

첫째, 멤버 어플리케이션이 2.2절의 상태①과 상태②에서는 Static Package만 가지고 실행하고 있다가 상태③에서는 Dynamic Package가 기존의 Static Package에 연결된다. 이를 위해서는 멤버 어플리케이션 프레임워크가 Static Package내부에 존재하여 Dynamic Package를 연결할 수 있는 확장점을 지원해야 한다[11,12].

둘째, uT-공간에서 커뮤니티 컴퓨팅이 제공하는 상황인지(Situation-aware)기반의 멤버간 협업 기능을 지

원해야 한다[2,10].

셋째, uT-공간에서 서로 다른 벤더에서 제작된 이종의 uT-개체들이 다양한 네트워킹 방식으로 연결되어 있는 이질성(heterogeneity)을 지원해야 한다[1]. uT-개체는 센싱 능력, 엑추에이터 능력, 통신 방법, 개체 고유의 기능과 컨텍스트 관리 기능이 다르게 된다.

넷째, uT-공간에서 서비스에 대한 새로운 요구사항이 발생하면, 새로운 서비스가 빠르게 추가되거나 변경할 수 있는 확장성(Scalability)을 지원해야 한다[3]. 멤버 어플리케이션은 커뮤니티 생성 전에 자동적으로 로드 되어 관리 되며, 커뮤니티 생성 후에 새로운 서비스가 요구되면 서비스가 동적으로 배포되어야 한다[2]. 이런 특성을 만족하기 위해서는 커뮤니티 컴퓨팅 시스템을 구성하는 uT-개체가 기존의 커뮤니티 서비스에 영향을 주지 않고, 새로운 서비스로 확장하여 진화할 수 있는 기능을 지원해야 한다.

4. 프레임워크 구조

커뮤니티 컴퓨팅의 멤버 어플리케이션은 Static Package와 Dynamic Package의 두 부분으로 나누어진다. Static Package는 Static plug-ins와 Framework로 Dynamic Package는 Dynamic plug-ins로 구성된다.

첫째, 그림 2와 같이 멤버 어플리케이션 프레임워크는 Static plug-ins와 Dynamic plug-ins가 각각 프레임워크에 연결할 수 있는 확장점(Extension Point)을 제공한다. Static plug-ins는 uT-개체의 통신방법, 센서, 엑추에이터, 컨텍스트 관리와 고유의 기능에 따라 다르게 탑재되는 플러그인으로 Communication Adaptor, Sensor Adaptor, Actuator Adaptor, Context Manager와 Stand Alone Function 플러그인으로 구성된다. Frame-

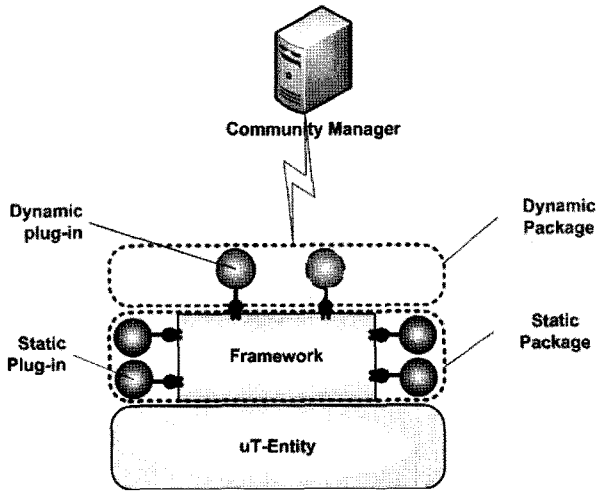

그림 2 Static Package와 Dynamic Package

표 1 Plug-In의 기능

플러그인		설명
Static Package	Main	Member Application의 시작점, Community 생성 전에 Stand Alone Function의 본래의 기능을 수행하고, Community 생성 될 때 커뮤니티의 Role을 수행하는 두 가지 상태를 Switching하는 기능
	Society Registration Manager	uT-entity가 Society Member로 등록 및 탈퇴 하는 기능
	Role Acceptor	Community Manager와 Society/Community Member 간에 Role-member Binding 작업을 지원하는 기능
	Download Manager	Community Manager로부터 Dynamic Package를 다운로드 및 설치하는 기능
	Function Mapper	Cooperation Plan의 Behavior와 Stand Alone Function의 Action을 매핑하는 기능
	Configuration Manager	Member Application의 전체 Plug-In들의 상태와 Member Application의 IP, name, Type, port와 Society/Community Manager의 IP, port 등을 관리하는 기능
	Message Broker	내부 플러그인들 간의 메시지 교환 통로 기능
	Policy Manager	Community 간의 충돌과 우선순위를 정하는 Policy 를 실현시켜주는 기능
	Context Manager	Sensor로부터 Raw Data와 Stand Alone Function의 결과로 Member의 Context를 생성하고 관리
	Stand Alone Function	uT-entity가 고유의 기능을 수행하는 액션들이 모인 모듈
	Sensor Adaptor	uT-entity에 장착된 sensor device를 작동할 수 있는 Adaptor
	Actuator Adaptor	uT-entity에 장착된 actuator device를 작동할 수 있는 Adaptor
	Communica-tion Adaptor	Network 접속장치와의 창구 역할을 하는 Adaptor
Dynamic Package	Cooperation Plan	동적으로 다운로드 되며, Situation 기반으로 Role를 수행하기 위한 프로그램을 실행하기 위한 모듈

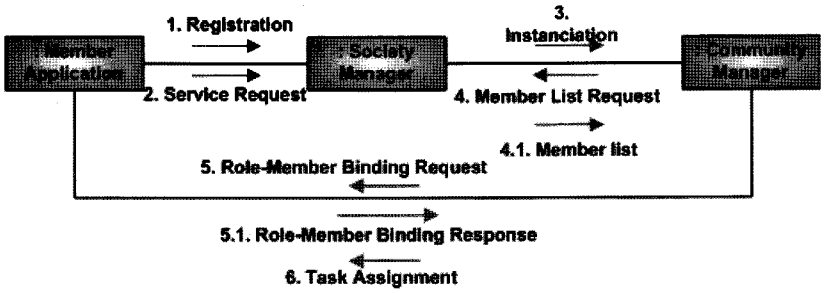

그림 3 Role-member Binding 과정

work는 uT-개체마다 공통적으로 필요한 컴포넌트의 집합으로 Main, Configuration Manager, Society Application Manager, Role Acceptor, Download Manager, Message Broker, Function Mapper와 Policy Manager로 구성된다.

Dynamic plug-ins는 커뮤니티의 상황에 따라 역할을 수행하기 위한 Cooperation plan 플러그인이 있다.

표 1은 각 플러그인에 대한 기능을 요약한 것이다.

둘째, 멤버 어플리케이션 프레임워크는 uT-공간에서 커뮤니티 컴퓨팅이 제공하는 상황인지(Situation-aware) 기반의 멤버간의 협업 기능을 지원한다.

그림 3은 멤버어플리케이션이 소사이어티 매니저에 등록하고 커뮤니티 매니저에 의해 역할-구성원 바인딩과정을 보여주고 있다.

2.2절에서 설명한 소사이어티 멤버 후보 상태의 멤버 어플리케이션은 그림 2에서 프레임워크 내부의 Society Registration Manger 모듈에 의해 멤버의 식별자 및 능력 정보를 소사이어티 매니저에 전송하여 소사이어티 멤버로 등록한다. 그 이후에 멤버 어플리케이션은 특정 서비스를 소사이어티 매니저에게 요청하고, 소사이어티 매니저는 커뮤니티 서비스를 관리하는 커뮤니티 매니저를 생성한다. 커뮤니티 매니저는 소사이어티 매니저에게 소사이어티 맴버 리스트를 요청하고 이를 받게 된다. 소사이어티 매니저는 멤버 리스트를 바탕으로 역할-구성원 바인딩 요청을 멤버 어플리케이션에 요구하게 되고, 멤버 어플리케이션 프레임워크 내부의 Role Acceptor 모듈이 바인딩 요구에 대해 프레임워크 내부의 Policy Manager모듈의 커뮤니티 우선순위 정책을 참조하여 수락(Accept) 또는 거절(Reject)을 소사이어티 매니저에 응답한다. 멤버 어플리케이션이 역할-구성원 바인딩 요청에 수락하였다면, 소사이어티 매니저는 Dynamic Package의 URL를 멤버 어플리케이션에게 전달하고, 멤버 어플리케이션 프레임워크의 Download Manager 모듈은 해당 URL에서 동적으로 Dynamic Package를 다운로드 받는다. 이 Download Manager 모듈은 Dynamic Package 내부의 Cooperation Plan 플러그인을 '설치하기(install)', '분석하기(resolve)', '시작하기(start)' 과정을 거쳐 플러그인이 실행된 활성(Activate) 상태로 만든다. 이후에 멤버 어플리케이션은 Cooperation Plan 플러그인에 의해 커뮤니티 전체 상황을 인지하여 Cooperation plan 내부의 수행하기로 약속된 행위를 기반으로 커뮤니티 멤버간의 협업을 하게 된다.

그림 4는 멤버 어플리케이션이 특정 커뮤니티의 상황을 인지하여 Cooperation Plan 플러그인의 행위를 어떻게 수행하는지를 보여주고 있다. 우선, 멤버 어플리케이션 Static Package의 Context Manager 플러그인은 Sensor Adaptor로부터 가공되지 않은 데이터(Raw Data)와 Stand Alone Function 플러그인의 액션 수행 결과를 취합하여 멤버의 컨텍스트를 생성하여 이를 커뮤니티 매니저에 전송한다. 커뮤니티 매니저는 각 멤버로부터 전송받은 멤버들의 컨텍스트를 바탕으로 커뮤니티의 전체의 상황을 결정하고, 이를 각 멤버 어플리케이션에 전송한다. 커뮤니티 상황을 전달 받은 멤

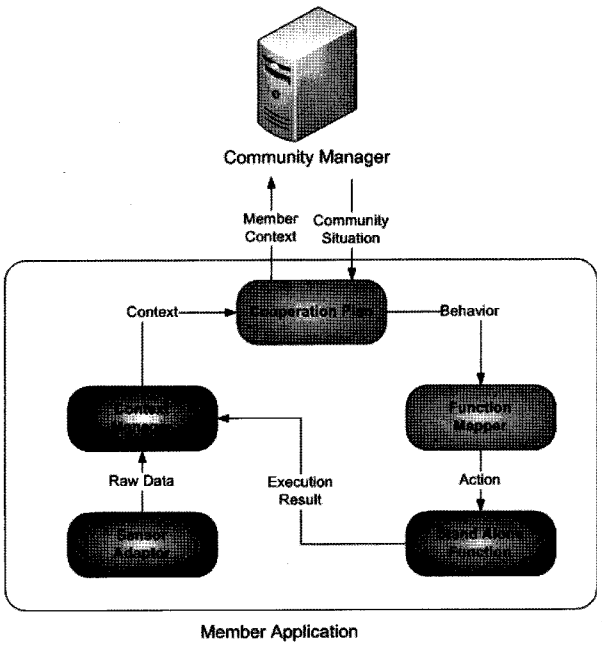

그림 4 Situation기반의 Role 수행 과정

버 어플리케이션의 Cooperation Plan 플러그인은 상황을 인지하여 수행하기로 약속된 행위로 정의 되어 있으며, 이 행위는 멤버 어플리케이션 프레임워크의 Function Mapper모듈에 의해 Stand Alone Function을 구성하는 액션의 조합으로 매핑되게 된다. 멤버 어플리케이션은 액션을 수행한 후, 자신의 컨텍스트 변화를 다시 커뮤니티 매니저에 전송하게 되고, 커뮤니티 매니저는 새로운 상황을 각 멤버에게 전송하게 된다.

셋째, 멤버 어플리케이션 프레임워크는 uT-공간에서 개체의 이질성(Heterogeneity)에 대해 적응해야 한다. uT-공간에 존재하는 uT-entity는 통신 방법, 센서(Sensor), 엑추에이터(Actuator), 컨텍스트 매니저와 고유의 기능이 서로 다르다. 이에 적응하기 위해서 멤버 어플리케이션 프레임워크는 Static Package의 Static Plug-in들을 각 uT-개체에 적합한 것을 선택하여 연결할 수 있는 확장점(Extension Point)을 제공한다.

우선 통신 방법, 센서 타입과 엑추에이터 타입 별로 플러그인이 개발되어야 하며 Context Manager와 Stand Alone Function은 개발자에 의해 uT-개체의 특징에 따라 프로그래밍 작업이 추가될 수 있도록 템플릿을 제공한다.

예를 들어, uT-entity가 무선랜과 유선랜의 두 가지 통신 방법을 지원하면, 멤버 어플리케이션 프레임워크에도 두 가지 통신 방법을 지원하는 각각의 Communication Adaptor플러그인을 선택하여 연결할 수 있고, 마찬가지로 센서나 엑추에이터도 uT-개체에 따라 해당 Adaptor 플러그인을 선택하여 연결할 수 있다. 이 때, 멤버 어플리케이션 프레임워크의 Configuration Manager 모듈이 각 플러그인 상태와 리소스를 관리하게 된다.

넷째, 멤버 어플리케이션 프레임워크는 uT-공간에서 커뮤니티 컴퓨팅 서비스의 확장성(Scalability)을 지원한다. 이는 멤버 어플리케이션이 2.2절에서 설명한 소사이어티 멤버 후보 상태와 소사이어티 멤버 상태로 Static Package를 미리 배포 받아 고유의 기능을 수행하다가, 커뮤니티 멤버 상태로 변화하면 멤버 어플리케이션 프레임워크의 Download Manager 모듈이 Dynamic Package를 동적으로 다운로딩하고 설치하여 Cooperation Plan 플러그인을 활성화 시킨다.

멤버 어플리케이션 프레임워크는 Dynamic Package의 Cooperation Plan 플러그인 연결할 수 있는 확장점을 제공한다.

커뮤니티 멤버 상태의 멤버 어플리케이션이 기존 커뮤니티의 실행 프로그램에 영향을 주지 않고, 다른 새로운 커뮤니티의 Cooperation Plan 플러그인을 다운로드 받아 설치하여 자신의 멤버 어플리케이션 프레임워크의 확장점에 연결할 수도 있다. 이 때, 멤버 어플리케이션 프레임워크의 Download Manager 모듈에 의해 플러그인 충돌해결 정책을 관리하는 Policy Manager 모듈의 우선순위 정책을 참고하여 결정한다.

따라서 멤버 어플리케이션 프레임워크는 기존 커뮤니티의 프로그램을 정지하지 않고, 새로운 커뮤니티의 플러그인만 실행 시키게 됨으로써 서비스의 점진적인 확장을 지원하게 된다.

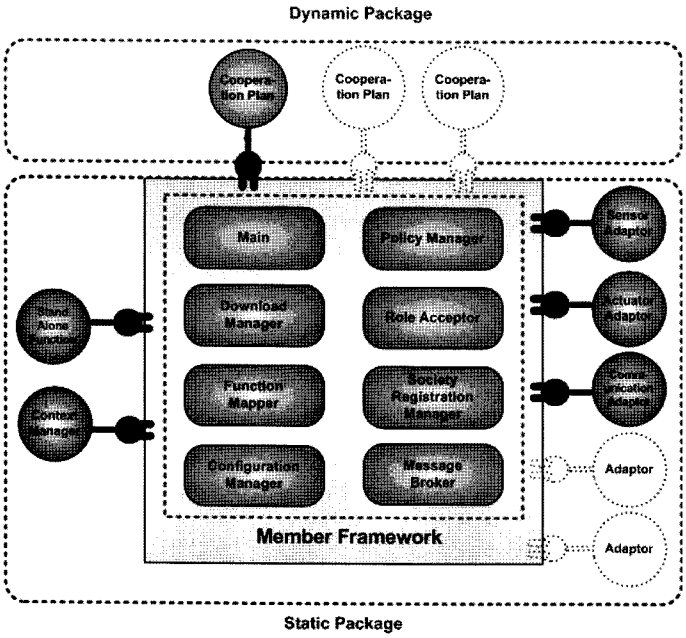

그림 5 멤버 어플리케이션 프레임워크 구조

그림 5는 앞에서 설명한 멤버 어플리케이션 프레임워크의 전체적인 구조를 보여 준다.

5. 시나리오 및 구현

커뮤니티 컴퓨팅 시스템에서 멤버 어플리케이션 프레임워크를 검증하기 위하여 아파트 단지의 엘리베이터에서 치한퇴치 시나리오를 이클립스 리치 클라언트 플랫폼(Eclipse Rich Client Platform)[12]을 기반으로 구현하였다.

5.1 시나리오

'치한 퇴치 커뮤니티'의 시나리오 배경은 다음과 같다. 여인 A씨가 입주한 아파트 단지는 유비쿼터스 환경을 갖추고 있으며, 이곳에는 모든 거주자의 안전을 위해서 단지 내 통합관제센터와 연결된 CCTV가 설치되어 있고, 통합관제센터에서 근무하는 경비원은 CCTV가 촬영한 영상을 통해 주민의 안전을 관찰한다. 또한 아파트 현관에는 RFID태그 이용하여 아파트 주민이 출입하는 시스템을 갖추고 있으며, 이를 통해 출입한 주민의 인적사항을 확인할 수 있다. 엘리베이터에는 CCTV가 설치되어 있으며 비상시 안전한 층수에 이동하는 기능을 갖추고 있다.

A씨는 평소에 안전한 귀가를 위해 팬던트의 버튼에 치한퇴치 커뮤니티 서비스를 요청할 수 있는 기능을 설정하였다.

엘리베이터의 같은 라인의 주민 소사이어티는 담당 경비원, 엘리베이터, 각 세대의 출동 가능한 주민을 멤버로 구성한다. 현재 5층·9층·15층 주민이 출동 가능하다.

아래는 본 시나리오의 자세한 과정을 나열하였다.

① 여인 A씨는 학교에서 근무를 마치고 귀가하여, 아파트 현관 앞에 서게 되며 A씨 펜던트의 RFID를 인식하여 문이 열리게 된다. 이때, 수상한 남자가 RFID인식 없이 A씨와 함께 현관문을 통과하게 된다.

② A씨와 남자는 엘리베이터를 동승하고 A씨는 10층을 누르고, 치한은 15층을 누르게 된다.

③ 갑자기 남자가 치한으로 돌변하여 A씨 뒤에서 접근하여 추행하려 한다. 이때, A씨는 소리를 지르고 자신의 팬던트 비상 버튼을 누름과 동시에 소사이어티 매니저에게 '치한퇴치 커뮤니티' 서비스 생성을 요구한다.

④ 커뮤니티 매니저는 소사이어티 맴버 중에 '치한퇴치 커뮤니티'를 구성하는 역할을 담당할 수 있는 멤버를 캐스팅하여 해당 역할을 맡긴다. 엘리베이터, 경비원, 5층 주민과 9층 주민이 '치한퇴치 커뮤니티' 멤버로 캐스팅 된다.

⑤ 커뮤니티 매니저는 'Start Community' 상황을 각 커뮤니티 멤버들에게 전달하며, 엘리베이터는 비상 정지할 층수를 커뮤니티 매니저에게 전송하고, 5층, 9층 주민과 경비원의 스마트 단말기는 자신의 위치를 커뮤니티 매니저에게 전송한다.

⑥ 커뮤니티 매니저는 'StopElevator&MovePlace' 상황을 각 커뮤니티 멤버들에게 전달하며, 주민들과 경비원의 단말기는 엘리베이터에게 비상정치 층수(7층)와 CCTV 영상을 요청하여 이를 응답받고 7층으로 출동한다. 그리고, 엘리베이터는 7층으로 비상정지를 한다.

⑦ 커뮤니티 매니저는 'ChaseCriminal' 상황을 각 커뮤니티 멤버들에게 전달하며, 엘리베이터가 7층에 정지하여 문이 열리고 5층 주민과 9층 주민은 7층으로 이동하여 치한과 격투를 한다. 경비원은 아직 도착하지 않았으며 격투도중 범인은 비상계단을 통해 아래로 도주한다. 5층 주민은 이 사실을 각 멤버에게 전송하고, A씨를 보호한다. 9층 주민은 치한을 추격하여 위층으로 출동하는 경비원과 협력하여 치한을 검거한다. 이후 경비원은 치한을 경찰서에 인도한다.

⑧ 커뮤니티 매니저는 'End Community' 상황을 각 커뮤니티 멤버들에게 전달하고 소사이어티 매니저에 의해 커뮤니티는 소멸한다.

5.2 구현

5.1절의 시나리오를 바탕으로 본 연구에서는 치한퇴치 커뮤니티 컴퓨팅 시스템을 이클립스 리치클라언

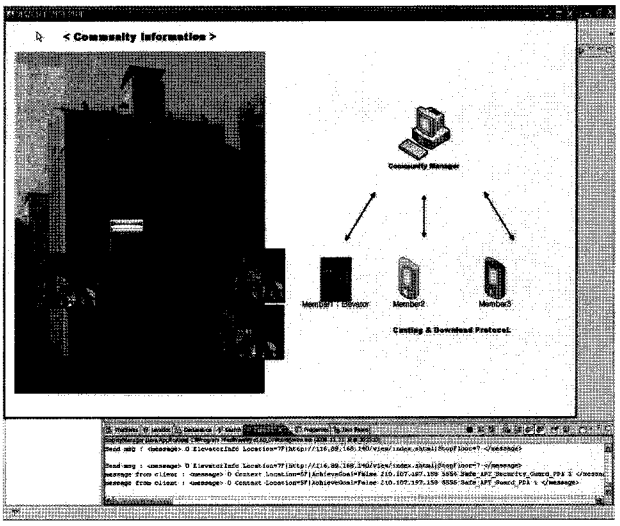

그림 6 커뮤니티 매니저의 모니터링 화면

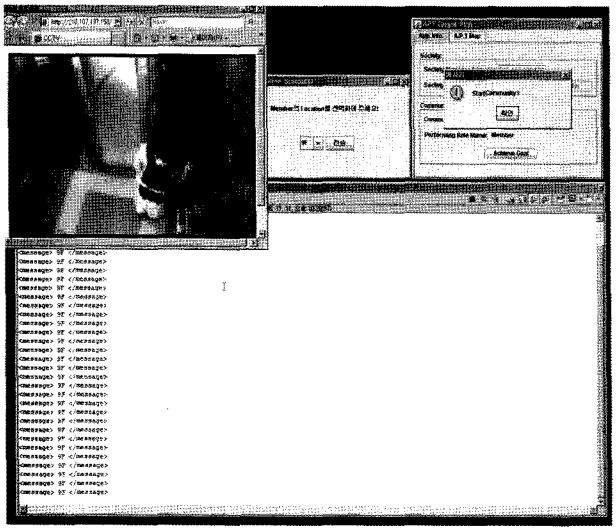

그림 7 경비원 멤버의 단말기 화면

트 플랫폼을 기반으로 멤버 어플리케이션을 구현하였으며, 이를 PC에 배포하여 시뮬레이션 하였다.

그림 6은 커뮤니티 매니저의 커뮤니티 전체 모니터링 화면을 나타내며 멤버에 의해 전송된 컨텍스트 값을 바탕으로 표시되었다.

그림 7은 경비원이나 주민의 단말기에 나타나는 화면을 보여 주는 것으로 유저 인터페이스는 두 개의 탭으로 구성된다. 이 화면의 첫 번째 탭은 소사이어티 매니저, 커뮤니티 매니저, 다른 멤버와의 상호작용을 하기 위한 탭이고, 두 번째 탭은 엘리베이터의 위치와 멤버 자신의 위치를 표시한다.

표 2는 각 멤버가 소사이어티에 등록할 때 소사이어티 매니저에 전송하는 속성과 액션의 리스트를 요약한 것이다.

표 3은 멤버의 컨텍스트에 의해 커뮤니티의 상황이 판단되는 것을 간단하게 정리한 것이다. G는 경비원, R은 주민, E는 엘리베이터라 가정한다.

표 2 커뮤니티 멤버의 속성리스트, 액션리스트

Member	attributes	actions
엘리베이터	location, CCTVURL, stopFloor	move stop
경비원	location	showCCTV responseStopFloor move, chase
주민	location	showCCT5V, requestStopFloor, move, chase

표 3 멤버의 컨텍스트, 상황, 행동 정의

	conditions	situation	behavior
G	url_success	StartCommunity	sendInfo
R	url_success		sendInfo
E	url_success		sendInfo
G	location	StopElevator& MovePlace	requestStopFloor requestCCTVURL move
R	location		requestStopFloor requestCCTVURL move
E	location stopFloor		stopElevator
G		ChaseCriminal	chase
R			chase
E	location=7F		stopElevator
G	achiveGoal=T	EndCommunity	
R	achiveGoal=T		
E			

시나리오를 바탕으로 구현한 결과를 살펴보면, 멤버 어플리케이션 프레임워크는 엘리베이터와 사람의 단말기를 대신하여 PC에 각각 미리 배포 되었으며, 각 uT-개체의 이질성을 지원하기 위해 Static Plug-in 템플릿을 이용하여 Function 부분을 직접 코딩하여 Stand Alone Function 플러그인을 완성하였으며, Sensor Adaptor, Communication Adaptor와 Context Manager 플러그인은 각각 uT-개체의 특징에 맞게 선택하여 멤버 어플리케이션 프레임워크의 확장점에 연결하였다. 또한 엘리베이터와 사람의 단말기는 커뮤니티 생성 후에 커뮤니티 매니저로부터 역할-구성원 바인딩을 통하여 커뮤니티 역할에 해당하는 Dynamic Package를 각각 동적으로 다운로딩 하여 설치하였다.

본 구현을 바탕으로 uT-개체의 이질성을 만족하고 서비스에 대한 확장성을 보여 주고 있다.

6. 관련연구

그간 일반 응용 어플리케이션 프로그램의 개발을 지원하기 위한 프레임워크에 대한 연구는 많이 되어 있으나 유비쿼터스 컴퓨팅 환경을 대상으로 하는 어플리케이션 프레임워크에 대해 연구는 시작 된지 얼마 되지 않으며, 특히 커뮤니티 컴퓨팅을 지원하는 어플리케이션 프레임워크에 대한 연구는 아직 발표된바 없다.

Aura 프로젝트[14]는 사용자 집중도(User Attention)

를 떨어뜨리지 않고, 작업할 수 있는 컴퓨팅 환경의 구축을 목표로 이의 실현을 위한 소프트웨어 시스템의 각 계층에 대한 요구사항을 정립하고 이를 실험적으로 구현한 프로젝트이다. Aura에서는 어플리케이션 프레임워크로 프리즘(Prism) 이라는 프레임워크를 제안하였는데, 이는 상황인지에 기반한 태스크의 재구성 및 이동(Task reconfiguration and migration) 기능을 위주로 개발되었다. 그러나 프리즘은 Aura 환경에서만 작동 가능한 프레임워크로서 다양한 플랫폼들이 이질적으로 존재하는 대규모 유비쿼터스 환경에 대한 고려는 되어 있지 않을 뿐만 아니라 서비스의 동적 배포 등의 기능도 제공되고 있지 않아 본 연구에서 추구하는 커뮤니티 컴퓨팅 모델에 적용하기는 어렵다.

GAIA[15] 프로젝트는 실생활의 물리적 환경을 프로그래밍이 가능한 액티브 공간(Active Space)으로 변환하기위한 기존 OS를 확장한 개념의 미들웨어를 디자인하고 구현하였다. GAIA는 액티브 공간 내 디바이스들의 이질성을 극복하기 위해서 상황인지 기반으로 태스크를 재구성할 수 있는 MVC 모델을 확장한 어플리케이션 프레임워크를 제시하였다[15]. 하지만 이 역시 프리즘과 비슷한 이유로 본 연구에서 추구하는 커뮤니티 컴퓨팅 모델에 적용하기는 어렵다.

한편, 서비스의 동적로딩 및 원격 제어를 지원하기 위한 프레임워크로는 OSGi의 프레임워크[16]가 있는데, 이질적인 디바이스들이 모여 있는 환경에서 원격으로 서비스의 동적 배포 관리가 가능할 뿐 아니라, Bundle로 불려지는 컴포넌트 조합에 의한 어플리케이션 구성의 간편함 등이 인정되어 현재 다양한 형태로의 발전이 이루어지고 있다. 본 연구에서 제시된 프레임워크도 이러한 OSGi의 프레임워크의 기본 모델을 수용하고 이에 커뮤니티 컴퓨팅 모델, 유비쿼터스 환경의 이질성에 대한 적응성, 서비스의 진화 등을 지원할 수 있는 형태로 확장 시킨 것이 라고도 볼 수 있다.

7. 결론 및 향후 연구 방안

본 연구는 uT-공간에서 커뮤니티 컴퓨팅을 기반으로 이질적인 uT-개체를 지원하고 서비스의 점진적인 확장이 가능한 어플리케이션 프레임워크인 멤버 어플리케이션 프레임워크를 설계하여 구현하였으며 이의 검증을 위하여 치한 퇴치 시나리오를 구현해 보았다.

커뮤니티 컴퓨팅은 차세대 유비쿼터스 컴퓨팅 서비스의 구현 방안으로 새롭게 연구되고 있는 분야로서 현재 기본 모델이 정립되고 이의 구현 가능성이 검증되어 있을 뿐 실용화 단계 까지 발전을 위해서는 많은 연구가 필요하다.

본 연구에서는 이러한 커뮤니티 컴퓨팅 개념을 상용화 단계까지 발전시키는데 있어서 멤버 어플리게이션 프레임워크의 개발이 매우 중요한 요소라 보고 연구를 진행 중에 있다. 본 연구에서 제시된 멤버 어플리케이션 프레임워크는 향후 차세대 대규모 유비쿼터스 환경의 중요한 요구 사항인 서비스의 지능화, 확장성 및 동적 진화성 과 환경의 이질성을 효과적으로 해결하기위한 목적 아래 개발이 진행되고 있다.

향후 연구로써, 멤버 어플리케이션 프레임워크를 다양한 시나리오에 적용하여 성능을 검증할 예정이며, 현재 연구가 진행 중인 MDA 기반의 응용 어플리케이션 개발 도구와 연계하여 기능을 보완함으로써, 프레임워크 뿐만 아니라 이를 지원하는 개발 도구도 개발할 예정이다. 멤버 어플리케이션 프레임워크 자체에 대해서는 정책(Policy) 관리 부분 개선 및 성능 개선을 통하여 지능화 및 경량화를 위한 연구를 진행할 예정이다.

참고문헌

[1] M. Weiser, "The Computer for the Twenty-First Century", Scientific Am., vol. 265, no. 3, pp.94-101, 1991.

[2] Eila Niemela, Juhani Latvakoski, "Survey of Requirements and Solutions for Ubiquitous Software", Proc. Mobile Ubiquitous Computing Conference, ACM Press, pp. 71-78, 2004.

[3] C. A. Costa, A. C. Yamin, C. F. R. Geyer, "Toward a general software infrastructure for ubiquitous computing", IEEE Pervasive Computing, Vol 7, Issue 1, pp. 64-73, 2008.

[4] Carnegie Mellon Software Engineering Institute, "Ultra-Large-Scale Systems: The Software Challenge of the Future", 2006.

[5] 강경란, 김민구, 이정태, 정유나, 조위덕, 김현숙, "커뮤니티 컴퓨팅: 협업 기반 환경 자동 적응의 컴퓨팅 모델", 정보과학회지 제24권 제12호, pp 84-91, 2006.

[6] 정유나, 이정태, 김민구, "다중 에이전트 환경에서의 커뮤니티 기반 유비쿼터스 시스템을 위한 모델과 개발도구", 정보과학회논문지: 소프트웨어 및 응용 제33권 제12호, 2006.

[7] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE A FIFA-compliant Agent Framework." In Proc. of Practical Application of Intelligent Agents and Multi-Agents (PAAM'99), London, UK, April, pp. 97-10, 1999.

[8] 김희수, 이정태, 김민구, "커뮤니티 컴퓨팅의 역할

할당 방법", 정보과학회 2008년 가을 학술발표논문집 제35권 제2호, pp 27-32, 2008.

[9] Y. Jung, J. Lee, M. Kim. "Multi-agent based community computing system development with the model driven architecture", Proc. Conference on Autonomous agents and multi-agent systems, pp. 1329-1331, 2006.

[10] van Kranenburg, H. Bargh, M.S. Iacob, S. Peddemors, A., "A context management framework for supporting context-aware distributed applications", Communications Magazine IEEE, Volume: 44, Issue: 8, pp 67-74, 2006.

[11] M. E. Markiewicz and C. J. P. de Lucena, "Object oriented framework development," Crossroads, vol. 7, no. 4, pp. 3-9, 2001.

[12] T. Hansen, "Development of successful object-oriented frameworks," in OOPSLA '97: Addendum to the 1997 ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, pp. 115-119, 1997.

[13] Eclipse Rich Client Platform, http://www.eclipse.org/rcp

[14] Joao Pedro Sousa and David Garlan, "Aura: An Architectural Framework for User Mobility in Ubiquitous Computing Environments", Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture, Kluwer Academic Publishers, pp. 29-43, 2002.

[15] Manuel Roman and Roy H. Campbell, "A User-Centric, Resource-Aware, Context-Sensitive, Multi-Device Application Framework for Ubiquitous Computing Environments", Report No. UIUCDCS-R-2002-2284 UILU-ENG-2002-1728, 2002.

[16] OSGi Framework, http://www.osgi.org

김 한 욱

1995 청주대학교 전자계산학과(공학사)
1999 한국교원대학교 컴퓨터교육학과(교육학석사)
2006 아주대학교 정보통신전문대학원 공학박사 수료
관심분야 : 유비쿼터스 컴퓨팅, 커뮤니티 컴퓨팅, 응용 프레임워크
E-mail : hanoogi@ajou.ac.kr

이 정 태

1979 서울대학교 농과대학 원예과(농학사)
1981 서울대학교 자연과학대학 계산학(이학석사)
1988 서울대학교 자연과학대학 계산학(공학박사)
1983~1988 울산대학교 전산학과 전임강사
1988~현재 아주대학교 정보 및 컴퓨터공학부 교수
관심분야 : 컴포넌트 베이스 시스템, 미들웨어, 객체지향 응용 프레임워크
E-mail : jungtae@ajou.ac.kr

김 민 구

1977 서울대학교 계산통계학과 전산학(학사)
1979 KAIST 전산학과 전산학(석사)
1989 펜실베니아 주립대학교 전산학(박사)
1981~현재 아주대학교 정보 및 컴퓨터공학부 교수
1997~1998 한국정보과학회 인공지능 연구회 운영위원장
1999~2000 University of Louisiana, CACS 연구과학자
관심분야 : 인공지능, 데이터 마이닝, 정보검색, 커뮤니티 컴퓨팅
E-mail : minkoo@ajou.ac.kr