유비쿼터스 컴퓨팅 환경에서의 커뮤니티 컴퓨팅을 위한 멤버 프레임워크

Member Framework for Community Computing in Ubiquitous Computing Environments

김한욱, Hanwook Kim*, 송세헌, Seheon Song*, 이정태, Jungtae Lee** 김민구, Minkoo Kim**
*아주대학교 정보통신전문대학원 정보통신공학과,
**아주대학교 정보통신대학 정보및컴퓨터공학부

요약 유비쿼터스 지능 공간(Ubiquitous Smart Space)에서 동적으로 발생하는 다양한 문제를 협업을 통하여 해결할 수 있는 방법론으로 제안된 커뮤니티 컴퓨팅 (Community Computing) 모델을 기반으로 하는 개발 도구(Community Computing Development Tool Kit : CDTK)를 사용하면 특정 문제를 해결할 수 있는 커뮤니티 컴퓨팅 어플리케이션이 생성된다. 이 커뮤니티 컴퓨팅 어플리케이션이 실제로 유비쿼터스 지능 공간에 존재하는 uT-entity에 이식되어 동작하기 위해서 uT-entity의 종류에 상관없이 커뮤니티 컴퓨팅 어플리케이션이 배포될 수 있는 환경을 필요로 한다. 본 연구에서는 CDTK를 이용하여 생성된 커뮤니티 컴퓨팅 어플리케이션이 uT-entity에 배포(Deployment)되어 각 uT-entity가 커뮤니티의 멤버로 참여하여 멤버간의 협업을 통해 목적(Goal)을 달성할 수 있도록 지원하는 어플리케이션 프레임워크인 멤버 프레임워크(Member Framework)를 제안하고, 이를 이클립스 리치클라이언트 플랫폼(Rich Client Platform) 기반의 플러그인(Plug-In)으로 설계하고자 한다.

핵심어: Member Framework, Member Application, Community Computing

1. 서론

유비쿼터스 지능 공간 네트워킹 능력, 초소형 프로세싱 능력을 가지고 있는 사물인 uT-entity들이 다양한 네트워크를 통해 연결되어 있다. uT-entity는 이동이 가능한 것과 고정되어 있는 것으로 분류할 수 있고, 환경 정보를 수집할 수 있는 센싱 능력을 가질 수 있으며, 단독으로 고유한 작업을 수행할 수 있다.

유비쿼터스 지능 공간에서 동적으로 발생하는 다양한 문제를 협업을 통하여 해결할 수 있는 방법론으로 제안된 커뮤니티 컴퓨팅 모델[2]을 기반으로 하는 CDTK를 사용하면 특정 문제를 해결할 수 있는 커뮤니티 컴퓨팅 어플리케이션이 생성된다. 이 커뮤니티 컴퓨팅 어플리케이션이 실제로 유비쿼터스 지능 공간에 존재하는 uT-entity에 배포되어 동작하기 위해서는 uT-entity의 종류에 상관없이 커뮤니티 컴퓨팅 어플리케이션이 배포될 수 있는 환경을 필요로 한다.

본 연구에서는 CDTK를 이용하여 생성된 커뮤니티 컴퓨팅 어플리케이션이 uT-entity에 배포되어 각 uT-entity가 커뮤니티의 멤버로 참여하여(Memberfication), 멤버간의 협업을 통해 목적(Goal)을 달성할 수 있도록 어플리케이션 프레임워크인 멤버 프레임워크를 제안한다.

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

또한, 멤버 프레임워크를 실제 리치클라이언트 플랫폼 기반의 플러그인 형태로 개발할 수 있도록 프레임워크의 기능을 분석하고자 한다.

2. 관련 연구

2.1 VPC (Virtual Private Community)

P2P 환경에서 에이전트를 사용하여 허가된 사용자가 서비스를 제공 가능한 커뮤니티 안으로 들어오면 권한(Role)을 부여하여 서비스를 이용할 수 있도록 하는 가상 개인화 커뮤니티이다.

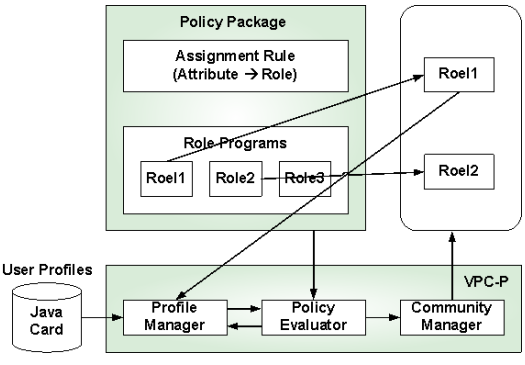

그림 1 VPC Platform

VPC는 그림 1과 같이 권한과 권한을 할당받는 조건으로 구성된 정책패키지(Policy Package), 변수와 값으로 구성된 유저 프로파일(User Profiels), 에이전트간의 협업을 수행할 수 있는 기반인 VPC-Platform으로 구성된다.

에이전트는 포탈 에이전트(portal agent)와 일반 에이전트(ordinary agent)로 구성된다. 일반 에이전트는 포탈 에이전트에게 자신의 공개 프로파일을 전송하여 포탈 에이전트로부터 정책 패키지를 전송받게 된다. 일반에이전트는 자신의 비공개 프로파일로 정책패키지를 평가하여 권한을 부여받게 된다. [2]

2.3 OSGi (Open Service Gateway Initiative)

OSGi 원래의 목적은 가정용 게이트웨이, 셋톱 박스, 자동차 계기판 컴퓨터 등 임베디드 기기를 개발하기 위한 자바 컴포넌트 및 서비스 모델을 제공하는 것이었다. OSGi 프레임워크 R.4.0은 컴포넌트 또는 번들(bundles)을 정의하고, 조합하고, 실행하기 위한 프레임워크 개발을 명세화하고 있다. [3]

2.3 이클립스 리치클라이언트 플랫폼

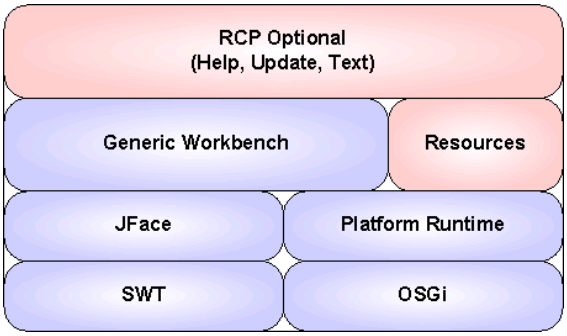

그림 2 리치클라이언트 플랫폼

이클립스는 구조화된 아키텍처로 개발자가 지원하고자 하는 툴 관련 어플리케이션의 종류에 따라 재사용될 수 있는 프레임워크를 지원한다. 그림 2와 같이 리치클라이언트 어플리케이션을 개발할 때 필요한 플러그인들을 모아놓은 이클립스 플랫폼의 한 부분으로 존재한다. 즉, 이클립스 플랫폼에서 지원하는 기능들을 사용하는 것이 가능하고, 플러그인들을 추가하여 확장하는 것이 가능하다.[4] [5]

리치클라이언트 플랫폼으로 개발된 어플리케이션은 OSGi 런타임에 기반하기 때문에 플러그인들을 결합하여 확장이 가능하고, 배포한 어플리케이션은 플러그인의 동적인 업데이트가 가능하기 때문에 추후에 유지 및 보수가 용이하다.[6]

3. 멤버 어플리케이션

3.1 정의

그림 3과 같이 uT-entity는 외부의 개입 없이 혼자서 본래의 기능(Own Action)을 수행한다. 만약 커뮤니티가 생성되면 특정 커뮤니티가 가지고 있는 목표(Goal)를 해결하기 위해 유비쿼터스 지능 공간에 존재하는 uT-entity를 멤버화 과정을 통해 커뮤니티의 멤버로 임명한다. 이때 선택된 uT-entity에 멤버화를 위해 배포되는 어플리케이션을 멤버 어플리케이션이라고 한다.

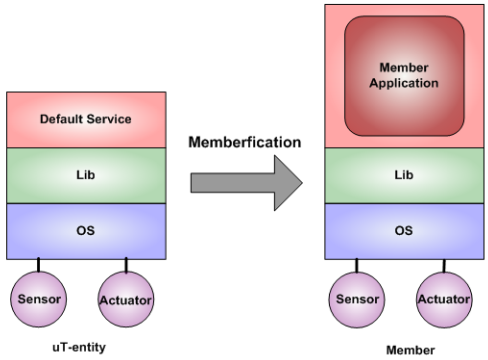

그림 3 멤버화 (memberfication) 과정

3.2 소사이어티 멤버와 커뮤니티 멤버

위에서 말하고 있는 멤버화 과정은 소사이어티(Society) 멤버화와 커뮤니티 멤버화로 나누어진다. 즉, 유비쿼터스 지능 공간을 구성하는 다양한 uT-entity는 특정 소사이어티에 포함되면 소사이어티(Society Member)가 된다[2].

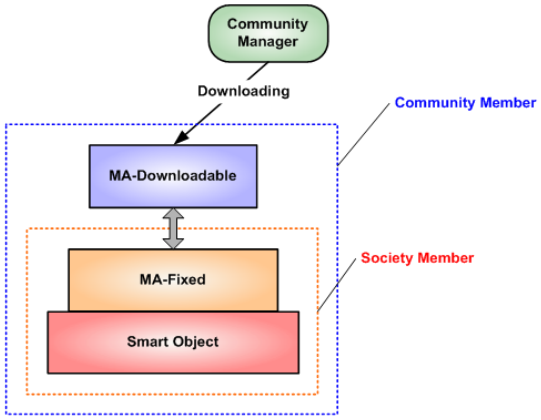

그림 4 Society member와 Community member

멤버화된 uT-entity에 멤버 프레임워크의 고정된 부분(Fixed Part)가 배포되어 그림 4와 같이 소사이어티 멤버가 된다. 이후에 소사이어티의 멤버가 커뮤니티 매니저(Community Manager)에 의해 특정 커뮤니티의 멤버로 캐스팅되면 멤버 프레임워크의 다운로드 가능한 부분(Downloadable Part)가 커뮤니티 매니저로부터 다운로드되

어 배포되면 그림4와 같이 커뮤니티 멤버가 된다.

4. 멤버 프레임워크

4.1 정의

다양하고 이질적인(heterogeneous) 환경을 지원하는 멤버 어플리케이션을 보다 쉽게 개발하기 위한 소프트웨어 프레임워크를 멤버 프레임워크라 한다.

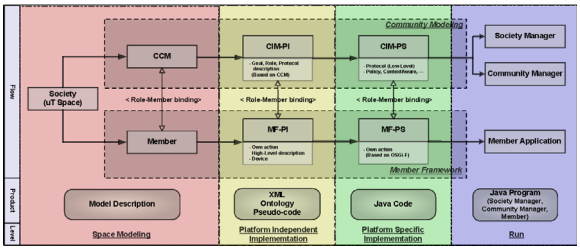

그림 5 CDTK의 흐름도

멤버 어플리케이션에 미리 이식되는 고정된 부분과 커뮤니티 매니저로부터 다운로드 가능한 부분으로 나누어진다. 고정된 부분은 모든 멤버에 공통적인 프레임워크와 멤버에 따라 선택 가능한 플러그인들로 구성된다.

그림 5와 같이 CDTK에 의해 CCM, CIM-PI, CIM-PS 단계를 거쳐 프레임워크와 플러그인들이 결합된 멤버 어플리케이션이 생성된다.[2]

그림 6과 같이 플러그인 방식으로 구성함으로써 유비쿼터스 환경의 다양한 이질적인 특성을 모두 지원할 수 있다.

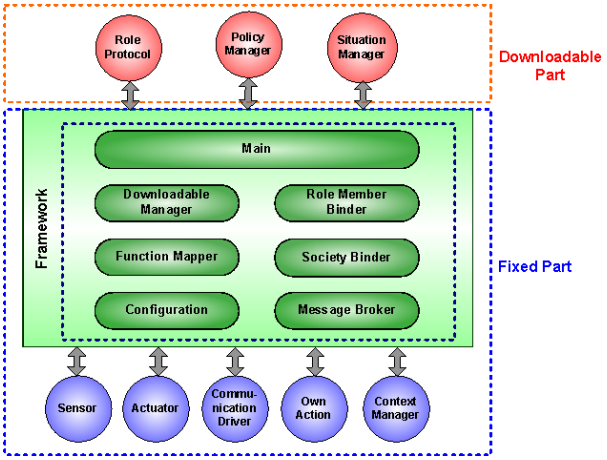

그림 6 프레임워크의 개념도

4.2 플러그인의 기능

CDTK로 플러그인이 생성되기 위해서는 플러그인의 공통적인 형식을 가지고 있어야 한다. 이는 플러그인의 교체가 필요할 시에 플러그인의 규격에 맡게 플러그인이 생성되어야 한다.

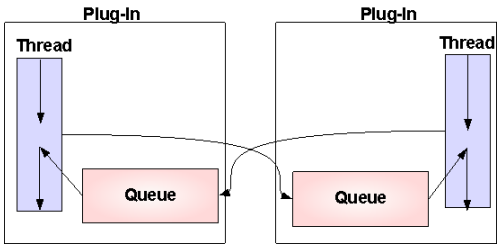

그림 7 Plug-In간의 메시지 전송방법

그림 7과 같이 모든 플러그인은 쓰레드(Thread)로 동작하고 큐(Queue)를 가지고 있다. 쓰레드가 동작하다가 일정한 시간 간격으로 큐를 검사하여 메시지의 처리를 수행한 후에 쓰레드를 계속 수행하게 된다. 표 1은 플러그인을 요약하여 설명하고 있다.

표 1 Plug-In의 기능

플러그인		설명
Downlo-adable Part	Role Protocol	Community Manager로부터 넘어온 role protocol 정보
	Policy Manager	멤버간의 층돌과 우선순위를 정하는 policy 정보
	Situation Manager	Member의 Context 정보를 Community Manager에게 송신, Situation 정보 수신
Fixed Part	Main	Member Application의 시작점으로 Plug-In들의 초기화, 실행, 중지, 재실행의 역할
	Role Member Binder	Community Manager와 Society/Community member 간에 role을 통한 바인딩 작업을 지원하는 모듈
	Download-able Manager	Community Manager로부터 Downloadable Part를 다운로드 및 설치
	Society Binder	uT-entity가 Society의 member화 되기 위한 바인딩 역할
	Function Mapper	Role Protocol의 서비스와 Own Action의 서비스를 매핑
	Configura-tion Manager	Plug-In의 상태 정보를 저장
	Message Broker	외부의 Community Manager와 다른 멤버와 메시지 교환을 위한 버퍼의 역할
	Communi-cation Manager	외부(Community Manager, community member 등)와의 통신을 관리하는 모듈
	Own Actions	uT-entity가 제공하는 기본 단위 동작
	Sensor Driver	uT-entity에 장착된 sensor device를 작동할 수 있는 driver
	Actuator Driver	uT-entity에 장착된 actuator device를 작동할 수 있는 driver
	Context Manager	Sensor로부터 Raw Data와 Own Action의 정보로 Member의 Context를 생성하고 관리

4.2.1 Main 플러그인

그림 8과 같이 멤버 어플리케이션에 설치된 플러그인들의 초기화를 담당하고 Configuration Manager에서 플러그인 설치를 확인하여 각각의 플러그인의 실행, 중지, 재실행을 역할을 담당한다.

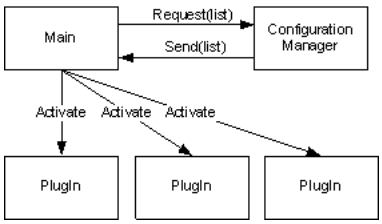

그림 8 Main과 다른 플러그인의 관계

4.2.2 Role Member Binder 플러그인

그림 9와 같이 소사이어티 멤버가 멤버 캐스팅(Casting)과 Role Protocol 다운로드를 통해 커뮤니티 멤버가 되기 위한 연결 작업을 담당한다.

커뮤니티 매니저로부터 멤버 캐스팅 요청을 받으면 멤버의 디바이스 명세(device Description)과 서비스 명세(Service Description)을 확인하여 커뮤니티 참여 여부를 결정하여 회답한다.

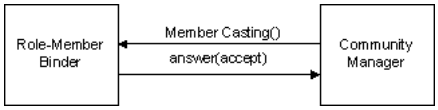

그림 9 Role Member Binder와 다른 플러그인의 관계

4.2.3 Downloadable Manager 플러그인

그림 10과 같이 커뮤니티 메니저로부터 다운로드 가능한 플러그인들의 URL를 받고, 해당 플러그인들을 다운로드 받아서 설치하는 역할을 한다.

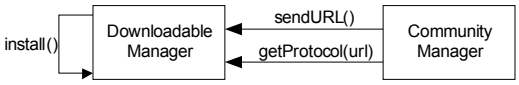

그림 10 Downloadable Manager와 다른 플러그인의 관계

4.2.4 Society Binder 플러그인

그림 11과 같이 소사이어티 매니저에 멤버 등록 요청하고 소사이어티 매니저로부터 멤버 ID를 수신하여 멤버화를 수행한다. 또한 커뮤니티 매니저에게 커뮤니티 생성을 요구할 수 있다.

그림 11 Society Binder와 다른 플러그인의 관계

4.2.5 Function Mapper 플러그인

그림 12와 같이 Role Protocol과 Own Action의 서비스를 매핑(Mappping)하고, 서비스 명세 기능과 서비스 명세를 관리하는 기능을 담당한다.

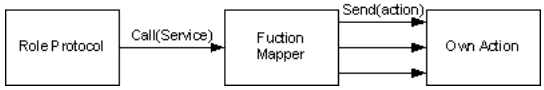

그림 12 Function Mapper와 다른 플러그인의 관계

4.2.6 Message Broker 플러그인

그림 13과 같이 커뮤니티 매니저와 다른 멤버 어플리케이션과의 메시지 교환을 위한 버퍼(Buffer)의 역할을 한다. 송신할 메시지에 대해 XML 형식으로 메시지를 작성하여 Communication Manager에게 전달하고, 수신된 메시지에 대해서 파싱(Parsing)을 하여 해당 플러그인에 전달하는 역할을 한다.

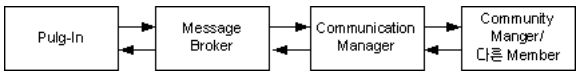

그림 13 Message Broker와 다른 플러그인의 관계

4.2.7 Communication Manager 플러그인

그림 13과같이 커뮤니티 매니저와 다른 멤버 어플리케이션과의 통신을 수행할 수 있는 Communication Driver 관리하게 된다.

4.2.8 Configuration Manager 플러그인

그림 8과 같이 멤버 어플리케이션이 실행되기 위해 필요한 Sensor, Actuator, Communication Driver, OS와 같은 Resource, 구성요소 관리를 하고, 플러그인들의 상태관리를 담당한다.

4.2.9 Own Action 플러그인

커뮤니티 멤버가 되기 전에 uT-entity가 가지는 고유의 기능이 기술된 플러그인이다.

4.2.10 Sensor Driver 플러그인

그림 14와 같이 uT-entity에 장착된 Sensor Device를 작

동하는 Driver를 관리 하는 기능을 가지며, 센서로부터 센싱된 Raw Data를 Context Manager에게 전달하는 역할을 담당한다.

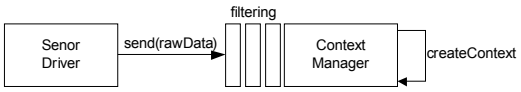

그림 14 Sensor Driver와 다른 플러그인의 관계

4.2.11 Actuator Driver 플러그인

그림 15와 같이 uT-entity에 장착된 Actuator Device를 작동할 수 있는 드라이버(Driver) 관리 기능을 가지며, Own Action으로부터 받은 액션(Action)을 실질적으로 실행하는 역할을 담당한다.

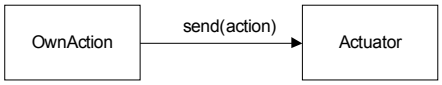

그림 15 Actuator와 다른 컴포넌트의 관계

4.2.12 Context Manager 플러그인

그림 16과 같이 센서로부터 받은 Raw Data와 Own Action의 결과 정보를 가지고 멤버의 컨텍스트(Context)를 생성한다. Role Protocol이나 Situation Manager에게 현재의 컨텍스트를 제공한다.

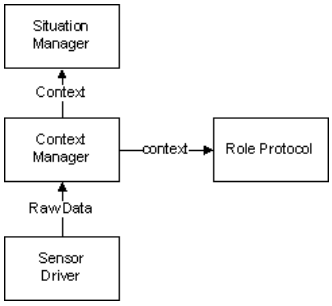

그림 16 Context Manager와 다른 플러그인의 관계

4.2.13 Role Protocol 플러그인

그림 10과 같이 Community Manager로부터 Downloadable Manager에 의해 Role Protocol이 다운로드 된 후 설치된다. 생성된 Protocol을 해석되어 실행하게 하는 역할을 한다. Situation Manager로부터 Situation을 받으면 Situation에 따라서 다른 Member와 협업을 하게 된다. 그림 12와 같이 Service를 Function Mapper에 의해 Own Action을 수행한다.

4.2.14 Policy Manager 플러그인

그림 17과 같이 Community Manager로부터 Policy 정보를 받아서 Role Protocol를 실행하는데 영향을 미치고 그림 15와 같이 Role Protocol을 Policy Manager가 실행시킨다.

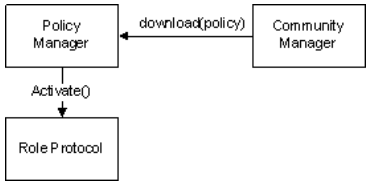

그림 17 Policy Manager와 다른 플러그인의 관계

4.2.15 Situation Manager Plug-In

그림 18과 같이 멤버의 Context Manager로부터 생성된 컨텍스트를 받아서 커뮤니티 매니저에게 전송하고, 커뮤니티 매니저로부터 Situation을 받아서 Role Protocol이 다른 멤버와 협업할 수 있도록 한다.

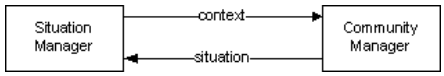

그림 18 Situation Manager와 Community Manager와의 관계

4.3 Device/Service Description

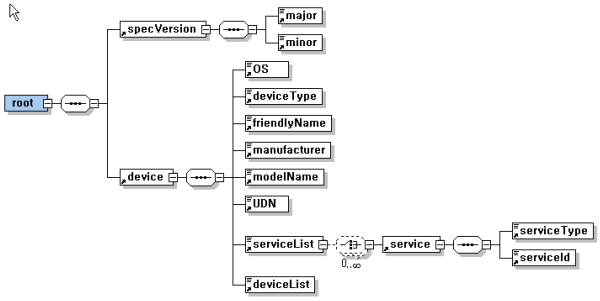

그림 19 Device Description Schema

표 4 Device Description Elements

엘리먼트	설명
root	Device의 Root Element
specVersion	Description의 버전
device	device에 element
OS	운영체제
deviceType	Device의 Type
friendName	Device의 설명
manufacture	Device의 제조회사
modelName	Device의 모델 이름
UDN	Unique Device Name
serviceList	Service의 List
service	Service
serviceType	Service의 Type
serviceID	Device Description에서 유일한 Service의 ID

명세(Description)은 UPnP의 디바이스 명세와 서비스 명세를 참고하여 작성되었다.[7] 디바이스 명세의 스키마(Schema)는 그림 19과 같고 자세한 엘리먼트(Element)의 설명은 표4와 같다. 디바이스 명세는 Configuration Manager에 의해서 참고되어 디바이스의 리소스나 구성요소 정보를 제공한다.

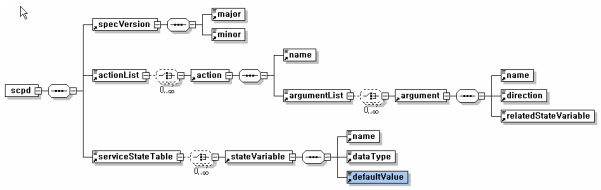

그림 20 Service Description Schema

표 5 Device Description Elements

엘리먼트	설명
scpd	Service Control Protocol Definition의 약어로 Device Description의 Service ID에 해당한다.
specVersion	Description의 버전
actionList	Action의 List
name	Action의 name
argumentList	Argument의 List
argument	Argument의 명세
name	argument의 이름
direction	input, out를 지정
relatedState-Variable	argument와 관련 된 변수를 지정
serviceState-Table	Service의 상태를 나타내는 List
stateVariable	action과 관련된 변수 명세
name	변수의 이름
dataType	변수의 타입
defatultValue	변수의 디폴트 값

그림 20은 서비스 명세의 스키마를 보여주고, 표 5와같이 엘리먼트에 대한 간단한 설명을 하고 있다.

서비스 명세는 Function Mapper에 의해 참조되어 Role Protocol의 서비스와 Own Action의 액션과 매핑하는 역할을 담당한다.

서비스 명세는 액션들의 리스트와 액션에 대한 매개변수(Argument)들로 구성되어 있다. 또한 서비스를 Discovery의 하는 과정은 먼저 Root Device를 찾아서 Service List에서 서비스를 찾고 서비스는 명세에서 각각의 액션을 검색하게 된다.

5. 결론 및 향후 연구 방향

본 연구는 유비쿼터스 지능 공간에서 발생하는 다양한 문제를 협업을 통해 해결하려는 커뮤니티 모델을 실제화하는 연구로써 유비쿼터스 지능 공간에 존재하는 다양한 형태의 uT-entity에 이식 가능하도록 지원하는 프레임워크를 제안한다. 이 프레임워크는 CDTK를 이용하여 생성되고 이를 사용하여 다양한 uT-entity에 커뮤니티 컴퓨팅 어플리케이션에 배치되어 실제로 커뮤니티 컴퓨팅 구현에 도움을 줄 것으로 판단된다. 향후 연구는 실제 시나리오를 바탕으로 멤버 프레임워크를 구현하여 CDTK로부터 다양한 이질적인 환경을 지원할 수 있는 멤버 어플리케이션이 생성되는지를 연구해야 할 것이다.

참고문헌

[1] K.Takahashi, S.Amamiya, T.Iwao, "An Agent-based Framework for Ubiquitous Systems", AAMAS 2003,

[2] Y. Jung, J. Lee, M. Kim. "Multi-agent based community computing system development with the model driven architecture", AAMAS 2006, pp.1329-1331, 2006.

[3] OSGi, http://www.osgi.org

[4] Eclipse.org, "Notes on the Eclipse Plug-in Architecture", http://www.eclipse.org/articles/Article-Plug-in-architecture/plugin_architecture.html, 2003.

[5] Eclipse.org, "Eclipse Forms: Rich UI for the Rich Client", http://www.eclipse.org/articles/Article-Forms/article.html, 2005.

[6] Eclipse.org, "Dev guide: Updating a product or extension", http://help.eclipse.org/help31/index.jsp?topic=/org.eclipse.platform.doc.isv/guide/product_update.htm, 2006.

[7] Upnp.org, "Universal Plug and Play Device Architecture", http://www.upnp.org/download/UPnPDA10_20000613.htm, 2000.