2

커뮤니티 컴퓨팅을 위한 커뮤니티 관리 시스템

아주대학교 ▌조위덕* · 조규찬 · 김양원 · 최재동 · 신진아

1. 서 론

일상생활 환경에 편재되어 있는 다양한 컴퓨팅 자원 간의 협업을 위한 연구는 분산 컴퓨팅을 시작으로 발전되어 왔다. 그러나 유비쿼터스 환경에서는 컴퓨팅 파워, 네트워킹 기능을 가진 디바이스뿐 아니라 서비스를 제공해주는 소프트웨어의 형태도 다양하며 이들의 커뮤니케이션을 지원하는 프로토콜과 네트워크 또한 혼재한(heterogeneity) 특성을 갖게 된다. 또한 사용자와 서비스의 이동성 및 상황 인지 기술이 중요한 특성이 된다[1]. 따라서 사용자, 서비스, 디바이스, 네트워크간의 다양한 범위에서 통합과 협업을 요구하게 되므로 상황 인지 기반의 협업 서비스를 기술하는 방법과 협업 요소의 이질성을 극복하기 위한 새로운 방법이 요구되었다. 이처럼 유비쿼터스 컴퓨팅 환경에서 요구되는 특성을 다음과 같이 정의 하였다.

첫째, 유비쿼터스 컴퓨팅 환경은 매우 다양한 이기종의 요소들로 구성되어 있으므로, 이러한 이기종의 요소들 간의 의사소통이 원활이 진행될 수 있도록 인프라가 갖춰져 있어야 한다.

둘째, 서비스에 대한 요구가 동적이고 동시다발적으로 발생가능 하므로, 미리 구현해 놓은 서비스 외에 미처 예측하지 못한 서비스에 대한 요구도 동적으로 만족시킬 수 있어야 한다.

셋째, 복잡하고 커다란 서비스일수록 단일 컴퓨팅 요소가 제공하기보다는 여러 컴퓨팅 요소들이 서로 협력하여 서비스를 제공해야 하는 경우가 많다. 유비쿼터스 서비스는 서로 다른 이기종의 컴퓨팅 요소들 간의 동적인 협업이 가능해야 한다.

넷째, 유비쿼터스 환경을 구성하고 있는 사람을 포함한 컴퓨팅 요소들의 상태가 끊임없이 변하게 된다. 이러한 환경 변화를 인지하여 변화에 맞는 서비스가 제공될 수 있어야 한다.

다섯째, 이기종의 객체들이 혼재하여 빈번하게 협업을 수행하는 유비쿼터스 시스템을 설계하고 개발하는 것은 매우 복잡한 작업이므로 개발을 최대한 편리하고 쉽게 작업할 수 있어야 한다.

이러한 유비쿼터스 컴퓨팅에 대한 필요 요건들 특히 협업을 통해서 유비쿼터스 서비스를 쉽고 빠르게 개발하고 제공하려는 목적으로 커뮤니티 컴퓨팅이라는 개발 패러다임을 새롭게 제안하였다.

2. 관련 연구

본 장에서는 유비쿼터스 컴퓨팅 환경에서 사용되는 몇 가지 기술을 짚어보고 이와 관련된 멀티에이전트, 서비스 조합 등의 몇 가지 관련연구들을 살펴본다.

2.1 멀티 에이전트 시스템

멀티에이전트 시스템은 하나의 에이전트로 해결하지 못하는 복잡한 문제의 해결을 위하여 여러 에이전트간의 협동을 통해 작업을 수행한다. 최소단위의 멀티에이전트 시스템은 하나의 조정 에이전트(Coordination Agent)와 둘 이상의 응용 에이전트들로 구성된다[2].

기존의 하나의 독립적인 에이전트는 지식표현과 추론 등을 통해 문제를 해결하지만 하나의 에이전트가 모든 기능을 다 갖출 수는 없으므로 자신이 해결하지 못하는 부분은 다른 에이전트와 협동을 해야 한다. 그러므로 응용 에이전트들 간의 메시지를 전달하고 각 에이전트의 제어를 수행하는 조정 에이전트를 필요로 한다. 조정 에이전트의 학습은 다른 에이전트가 보이는 반응과 정보 등을 수집하고 분석하여 전체 시스템의 효율이 최대화되도록 에이전트들의 기능과 역할 분담을 조정한다. 예로는 협상 에이전트, 자원 할당 에이전트 등을 들 수 있다. 시스템의 전체적인 기능을 하나의 에이전트로 구성하지 않고 각 세부 기능마다 별도의 에이전트를 구성하면, 여러 에이전트의 작업을

* 종신회원

† 본 연구는 지식경제 프론티어기술개발사업의 일환으로 추진되고있는 지식경제부의 유비쿼터스컴퓨팅및 네트워크원천기반기술개발사업의 09C1-T3-10M 과제로 지원된 것임

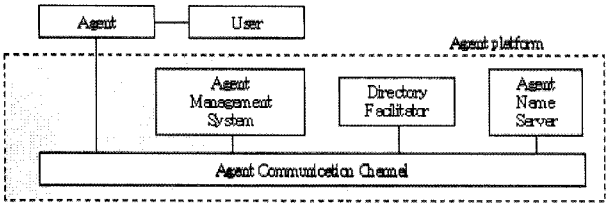

그림 1 FIPA 모델

결합시켜 최종 목표를 달성할 수 있다. 그리고 같은 기능의 에이전트를 중복적으로 개발해야 하는 자원의 낭비도 막을 수 있다. 멀티에이전트 시스템에는 EMAF (an Extensible Multi-Agent Framework), ARCHON, JATLite(Java Agent Template Lite), FIPA(Foundation for Intelligent Physical Agents) 모델 등이 존재한다.

2.2 서비스 지향 아키텍처

최근 들어 서비스 지향 아키텍처(Service Oriented Architecture)가 주목을 받는 큰 이유 중 하나는 추상화로부터 시작되는 서비스간의 느슨한 결합이 개발의 생산성 뿐 아니라 유지보수 관리의 편리성을 향상시키기 때문이다. SOA는 기존의 컴포넌트 소프트웨어 개발 방법과는 달리 통신 프로토콜에 상관없이 서비스 작성(composition) 및 조립(assemble)을 용이하게 하여 기업 내부의 시스템은 물론 B2B 시스템에 이르기까지 전사적인 시스템 통합을 가능하게 한다는 점에서 각광 받고 있다. 특히 웹 서비스와 ESB(Enterprise Service Bus)와 같은 구현 기술에 힘입어 실시간 기업 환경을 실현하기 위한 비즈니스 분야에 급속도로 확산되고 있다.

서비스는 서비스 인터페이스와 그 인터페이스를 구현한 본체로 이루어지며 하나의 서비스는 단일 서비스로 이루어진 서비스일 수도 있고, 여러 개의 서비스들이 조합된 서비스 일 수도 있다.

자동으로 서비스를 조합하는 기술은 간단한 스크립트 또는 워크플로우 기반의 템플릿을 사용하여 실행시간에 서비스들의 조합을 생성하는 접근 방법과, 사전에 정의된 템플릿 없이 실행시간에 지능적 계획(AI planing)을 통해 목적 지향적 서비스 조합을 하는 연구가 존재한다[3].

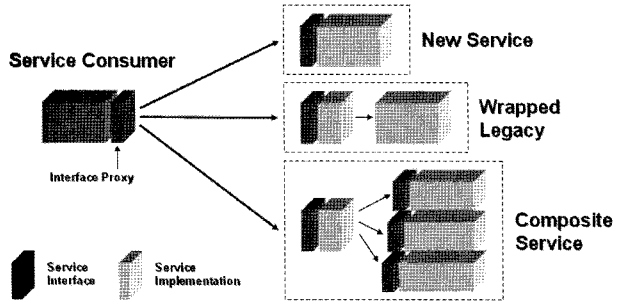

그림 2 서비스 지향 아키텍처

2.3 컨텍스트 어웨어

애플리케이션의 상황인지 서비스 지원을 위해 센서들로부터 컨텍스트 정보를 취합하여 상황을 판단하고 이를 애플리케이션들에 전달 해주는 시스템 아키텍처로는 UMBC의 Cobra[4]가 초기 시스템으로 가장 잘 알려져 있으며 가장 많이 참조되고 있다. 유비쿼터스 컴퓨팅 환경에서 상황인지 시스템의 초기 모델은 거의 대부분 Cobra와 마찬가지로 미들웨어 서비스 또는 서버 애플리케이션으로 개발되었으나, 최근에는 서버에 집중되는 부하를 줄이기 위해 분산 상황 인지 시스템들에 대한 연구도 활발히 진행되고 있다.

2.4 온톨로지

온톨로지(Ontology)는 현실 세계에 존재하는 개념과 존재의 본질에 대한 학문이다. 온톨로지라는 단어의 어원은 존재(being)와 학문(-logy)의 합성어로, 철학에서 온톨로지는 존재의 학문이고 순수 철학의 기본 주제를 형성한다. 온톨로지는 그것에 포함되어 있는 존재하는 것들과 그 유형을 정의하기 위하여, 존재의 기본 분류와 관계를 설명하고 가정한다. 컴퓨터 과학에서 온톨로지에 대한 연구는 인공지능 분야에서 지식을 표현하는 방법에 대한 연구로 진행되어 왔으며, 인지된 사실로부터 새로운 사실을 추론하기 위해 필요한 개념들을 설명하고 이들 사이의 관계를 제공한다.

온톨로지에 대한 주목과 연구는 표준화로 연결되어 W3C(World Wide Web Consortium)와 ISO(International Organization for Standardization)에서 진행되고 있다. W3C는 온톨로지 표현 방법에 대한 표준으로 RDF (Resource Description Framework)와 OWL(Web Ontology Language)을 제안하였다. RDF는 웹에 존재하는 자원들에 대한 정보를 표현하기 위한 언어로 SPO(Subject, Predicate, Object) 모델과 XML을 기반으로 한다. OWL은 XML과 RDF, RDF/S(RDF-Schema)에 정형화된 의미를 추가함으로써 기계의 정보 해석 능력을 향상시킨다.

웹의 발전과 웹-서비스의 보급에 힘입어 시맨틱 웹-서비스를 구현하기위한 OWL-S[10] 등의 서비스 온톨로지에 관한 몇몇 연구가 진행 중에 있다. OWL-S의 주요 기능은 자동 웹-서비스 발견, 자동 웹서비스 호출, 자동 웹 서비스 조합 및 상호 운영 기능이 존재한다.

2.5 유비쿼터스 관련 프로젝트

기존의 컴퓨팅 환경과 다른 유비쿼터스 환경에서 사용자는 장치를 직접 사용하지 않고도 원하는 서비스를 제공 받을 수 있으며, 집과 사무실 그리고 거리의 도처에 존재하는 다양한 자원은 사용자를 거치지 않고서도 서로간의 협업이 가능해야 한다. 즉, 높은 수준의 사용자 요구와 장치들에 의해 제공되는 저 수준의 기능간의 차이를 관리해야 하며 주변에 둘러 쌓인 다양한 장치들 간의 통합 및 협업이 가능해야 한다. 이를 실현하기 위한 여러 프로젝트가 존재하지만 몇 가지만 예로 들기로 한다.

AURA는 미국의 CMU(Carnegie Mellon University)에서 진행되고 있는 프로젝트이며 유비쿼터스 컴퓨팅의 구축을 목표로 하고 있다. AURA는 각각의 사용자의 환경을 AURA라는 추상적인 개체로 모델링하고 있으며, 이는 다시 context observer, task manager 등 여러 개의 지원 컴포넌트로 구성된다. AURA 내부의 context observer는 사용자 및 환경의 변화를 지속적으로 모니터링 함으로써 상황인지적인 서비스를 제공할 수 있게 된다.

GAIA는 UIUC(University of Illinois at Urbana-Champaign)에서 진행하고 있는 유비쿼터스 컴퓨팅 프레임워크를 개발하는 프로젝트이며 환경을 active space로 묘사하여 유비쿼터스 환경을 일련의 active space로 구성한다. GAIA는 사용자를 지원하는 소프트웨어의 형태가 적정수준의 지능을 가지고 있다고 가정함으로써 일종의 에이전트 기반의 구성 방식으로 볼 수 있다.

3. 커뮤니티 컴퓨팅 모델

커뮤니티 컴퓨팅은 동일한 목적을 달성하기 위해 모인 집단을 일컫는 사회과학 분야의 커뮤니티 개념을 컴퓨팅 환경에 적용한 것으로서 특정 목적을 달성하기 위해 결성된 컴퓨팅 요소들의 협업 조직을 커뮤니티라고 정의하고 이러한 커뮤니티를 이용하여 유비쿼터스 서비스를 제공하는 시스템을 위한 새로운 개발 패러다임이다. 커뮤니티 컴퓨팅의 궁극적인 목적은 유비쿼터스 컴퓨팅 환경 내에 존재하는 컴퓨팅 요소들 간의 협업으로 문제를 해결하여 필요한 서비스를 제공하는 유비쿼터스 컴퓨팅 시스템을 쉽고 빠르게 개발하는 것이다.

커뮤니티는 특정 목적을 달성하기 위해 구성되는 멤버들의 협업 조직으로서, 각 멤버들이 특정 역할(Role)을 맡아 협업을 진행하여 커뮤니티의 목적을 달성한다. 이때, 어떤 멤버가 커뮤니티에서의 특정 역할을 맡기 위한 조건을 기술하여 이에 맞추어 멤버를 선택하도록 한다. 커뮤니티는 달성할 목표가 발생한 시점에서 동적으로 생성되어 멤버들 간의 협업을 진행시키며, 목적이 달성되면 커뮤니티가 해체된다[5].

커뮤니티 컴퓨팅은 유비쿼터스 환경에 존재하는 센서, 디바이스, 사람 등의 지능화 수준이 다양한 이기종의 객체들을 하나의 커뮤니티의 멤버로 정의하여 서비스를 제공하여 혼재한 장치들 간의 통합을 가능하게 한다.

또한, 협업 서비스가 요구되는 시점에 실제 커뮤니티를 조직하여 동적으로 대처할 수 있는 장점을 가진다.

서비스 개발자들은 멤버를 개발할 때는 협업에 대한 고려를 하지 않고 개발하더라도 커뮤니티를 통한 협업이 가능하므로 개발의 효율성이 증대된다. 커뮤니티 컴퓨팅의 주요 정의 및 개념은 다음과 같다.

멤버(Member) - 커뮤니티 컴퓨팅 시스템을 이루는 기본 구성 요소로서 하나의 에이전트로 구현된다. 멤버는 컴퓨팅 디바이스, 하드웨어, 사람, 또는 소프트웨어를 대표할 수 있으며 커뮤니티를 구성하지 않고 있을 때에는 자신의 일을 수행하고 있다가 특정 목표(Goal)가 발생하여 커뮤니티가 구성되면 커뮤니티에 속해 목적을 달성시키기 위해 자신의 역할을 수행한다.

커뮤니티(Community) - 여러 에이전트 멤버들이 어떠한 목적을 달성하기 위하여 형성하는 협업 조직으로서 하나의 커뮤니티는 자신의 목적들과 이를 달성하기 위한 방법을 기술한 프로토콜들에 대한 정보를 갖는다. 커뮤니티를 조직할 때, 커뮤니티에서 역할을 수행하는 멤버가 될 수 있는 조건을 제시하여 각 역할에 맞는 멤버들을 선택하여 커뮤니티를 생성한다.

목표(Goal) - 하나의 커뮤니티가 만족시켜야 할 목표로서 특정 상황(Situation)으로 볼 수 있으며 해당 커뮤니티를 구성하는 이유라고 할 수 있다.

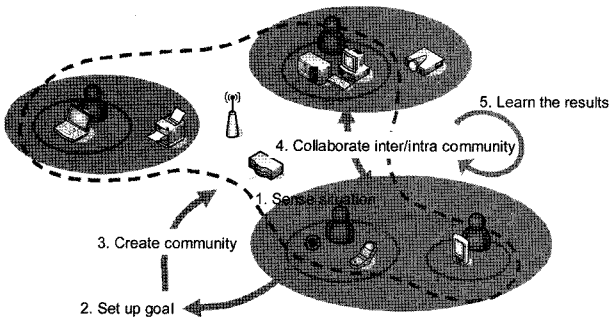

그림 3 커뮤니티 컴퓨팅 시스템의 예

협업(Cooperation) - 커뮤니티 내의 멤버들이 특정 목적을 달성하기 위해 협업하는 방식을 기술한 것으로서 여러 종류의 통신 메시지와 각 멤버의 자체 작업 수행들의 나열로 정의된다.

4. SOA 기반 커뮤니티 컴퓨팅 시스템

커뮤니티 컴퓨팅을 구현하기 위해 다양한 접근방법이 존재할 수 있다. 앞서 언급한 멀티에이전트를 기반으로 커뮤니티를 구현할 수 있으며 서비스 아키텍처를 이용하여 구현할 수도 있다. 본 연구에서는 SOA의 장점인 재사용성, 느슨한 결합을 적용하여 SOA 기반의 커뮤니티 컴퓨팅을 구현하였다. 본 장에서는 SOA 기반으로 동작하는 커뮤니티 관리 시스템이 서비스를 제공하기 위한 절차를 설명 한다.

유비쿼터스 도메인에서 사용되는 정보들은 실로 다양하게 존재한다. 해당 도메인의 물리적 공간 정보 뿐 아니라 서비스에 대한 정보, 상황 인지에 사용되는 컨텍스트들을 지식 공학에서 사용되는 온톨로지[10,11]로 정의하고, 도메인에서 발생할 수 있는 서비스 시나리오를 기반으로 추상적인 수준에서 서비스들을 추출한다. 그 서비스 실행이 도메인에 미치는 영향을 'effect'로 기술하고 개개 추상서비스들의 상관관계를 정의한다. 이렇게 정의된 추상 서비스들의 집합을 메타서비스 온톨로지라고 하며 이를 이용하여 커뮤니티가 구성되고 동작되어야 하는 스크립트를 작성할 수 있게 된다. 커뮤니티의 구성 및 동작 정보를 담은 스크립트를 커뮤니티 템플릿이라고 정의하였다. 커뮤니티 템플릿의 커뮤니티 역할(role)들을 메타서비스로 정의하여 런타임 시스템에 반영하게 되면 메타서비스와 실행 가능한 서비스를 바인딩 하여 커뮤니티 컴퓨팅 모델의 동적 role-member 바인딩이 이루어지고 이러한 실행 가능한 서비스의 조합으로 커뮤니티가 생성된다. 생성된 커뮤니티 인스턴스는 커뮤니티 목표에 따라 사용자 서비스를 제공하게 된다[6].

4.1 커뮤니티 기술 언어

커뮤니티 기술 언어는 커뮤니티의 정보를 기술하여 서비스 가능한 커뮤니티 템플릿을 만들기 위한 XML 기반의 언어이며 CDL(Community Description Language)라고 정의하였다. 그림 4는 CDL의 스키마를 도식화 한 그림이다.

커뮤니티 정의 시 커뮤니티의 목표와 커뮤니티 구성에 필요한 역할(role), 그리고 각 role들이 특정 상황에서 어떻게 협업해야 하는지를 기술하게 된다. 이렇게 기술된 커뮤니티 템플릿은 실행 타임에 커뮤니

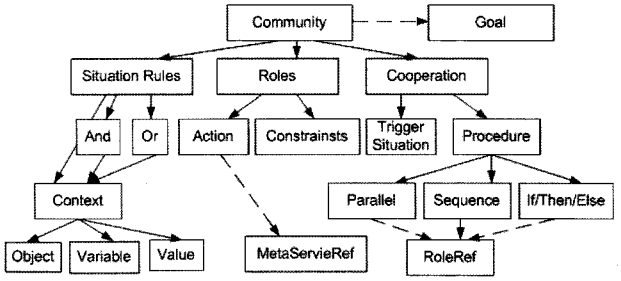

그림 4 CDL 스키마

티 인스턴스로 생성되어 커뮤니티 서비스를 제공하게 된다.

CDL의 루트 엘리먼트인 Community는 총 10개의 구성 요소로 이루어져 있으며, 각 요소들도 계층 구조를 가지고 여러 개의 하위 구성 요소로 이루어져 있다.

Community Name - 커뮤니티의 이름으로 일반 문자열 형식으로 필수 구성요소이다.

Situations - 커뮤니티의 라이프사이클을 컨트롤하는 이벤트들로 커뮤니티의 구성과 실행, 종료 등에 관련되는 상황 정보를 표현하는 룰이다. Situation은 Situation Name과 RDF 트리플[9] 형태로 표현된 Context들로 구성되며, Context들 간의 Not, And, Or 논리 표현을 부여할 수 있도록 되어있다.

Activation Situation - 커뮤니티를 활성화 시키는 이벤트 조건으로 특정 상황에 대응하는 Situation을 기술한다.

Terminate Situation - 커뮤니티를 종료시키는 이벤트 조건으로 특정 상황에 대응하는 Situation을 기술한다.

Community Goal - 커뮤니티가 달성해야할 목표로 일반 스트링 타입으로 기술한다.

Roles - 커뮤니티가 구성되기 위해 필요한 역할(Role)들에 대한 것으로 Role Name, Role Category, Role Constraint, Role Action 으로 구성된다. Constraint는 커뮤니티의 특정 역할로 바인딩 될 때 체크해야 할 제약 조건들로 type과 value로 이루어져 있다. Role간 Relation 및 Role과 사용자간 물리적 위치에 대한 조건 등이 제약조건으로 정의될 수 있다. Role Action은 Parameter들로 이루어져 있으며 Parameter는Parameter Name, Parameter Category, Parameter Type 및 Parameter Value로 이루어져 있다.

Procedures - 커뮤니티의 실행 프로세스에 관한 것으로 Procedure를 트리거 시키는 Situation Name과 Parallel, Sequence, If/Then/Else 중 하나의 프로세스 타입으로 정의된다.

Member - 커뮤니티의 부가 구성 요소로 커뮤니티

```
<?xml version="1.0" encoding="EUC-KR"?>
<communityTemplate name="S2-UCS-수면환경조성">
    ..........
    ..........

    <activateSituation>perceiveSleepMode</activateSituation>
    <goal>To make Sleep-Mode for the user by using unspecified services, control the
    illuminance sound temperature level</goal>
    <terminateSituation>SleepModeDone</terminateSituation>
    <role name="manageIlluminance" roleID="r1">
        <constraint type="IsLocatedIn" value="$location" />
        <constraint type="COUNT" value="*" />
        <constraint type="Effect" value="E-Illuminance" />
        <roleAction roleID="r1" />
    </role>
    <role name="manageSound" roleID="r2">
        <constraint type="IsLocatedIn" value="$location" />
        <constraint type="COUNT" value="*" />
        <constraint type="Effect" value="E-Sound" />
        <roleAction roleID="r2" />
    </role>
    <role name="manageTemperature" roleID="r3">
        <constraint type="IsLocatedIn" value="$location" />
        <constraint type="COUNT" value="1" />
        <constraint type="Effect" value="E-Temperature" />
        <roleAction roleID="r3" />
    </role>

    <procedure>
        <triggerSituation>perceiveSleepMode</triggerSituation>
          <parallel>
              <roleAction roleID="r1" />
              <roleAction roleID="r2" />
              <roleAction roleID="r3" />
          </parallel>
    </procedure>

    </communityTemplate>
```

그림 5 커뮤니티 템플릿 예제

를 구성하는 서비스들에 대한 정보를 표현한다. Member 요소가 정의되어 있으면, 해당 커뮤니티는 동적으로 서비스를 검색하여 커뮤니티를 구성하는 것이 아니라, Member에 해당하는 서비스들로 커뮤니티를 구성한다. Member는 커뮤니티를 실행하기 위해 필요한 정보들로 Service Id, Service Name, Port Type, Parameter들을 Member Action의 하위요소로 정의하며, Member State, Vendor 등에 대한 부가 정보를 함께 가진다.

그림 5는 CDL로 기술된 커뮤니티 템플릿의 예제를 보여준다.

4.2 SOA 기반 커뮤니티 컴퓨팅 시스템

본 연구에서 개발한 SOA 기반 커뮤니티 컴퓨팅 아키텍처의 런타임 시스템으로는 커뮤니티 매니저와 서비스 디스커버러, 컨텍스트 브로커와 같은 시스템이 있다. 개발타임에 커뮤니티 템플릿 디자인을 보조해 주는 도구로는 커뮤니티 에디터와 시뮬레이터가 있다. 커뮤니티 에디터는 커뮤니티 설계자들이 커뮤니티 템플릿을 쉽게 작성할 수 있도록 데이터의 연동 자동화, 문법 오류 자동 진단 등의 기능을 제공하는 편집 도구이며 커뮤니티 시뮬레이터는 커뮤니티 템플릿을 사용자 실행 환경과 유사한 가상공간을 직접 설계하고 그 공간에서 커뮤니티 서비스를 시뮬레이션하여 그 동작을 확인할 수 있게 하는 도구이다. 커뮤

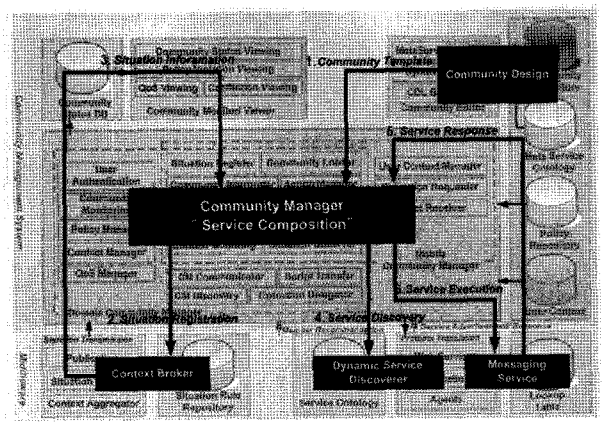

그림 6 SOA 기반 커뮤니티 컴퓨팅 시스템 아키텍처

니티 매니저는 추상화 되어 있는 커뮤니티 템플릿을 로딩 하여 서비스 디스커버러를 통해 실행 가능한 서비스들의 조합인 커뮤니티 인스턴스를 구성하고 컨텍스트 브로커를 통해 전달되는 상황 정보를 인식하여 커뮤니티 서비스의 실행을 트리거하고 그 생명주기를 관리하는 시스템이다. 컨텍스트 브로커는 센서로부터 환경 컨텍스트를 수집하여 특정 상황을 인지한 후 커뮤니티 매니저에게 상황 정보를 전달해주는 시스템이며, 서비스 디스커버러는 지능공간에 존재하는 모든 서비스를 등록하여 요청 시 효과(effect)와 공간정보 등을 기반으로 서비스를 검색하여 결과를 반환해주는 서비스 검색 시스템이다.

커뮤니티 매니저의 주요 기능으로는 의미기반 커뮤니티 구성, 상황 기반의 커뮤니티 실행, 라이프 사이클 모델에 따른 커뮤니티 관리가 있다.

커뮤니티 매니저는 커뮤니티 에디터를 이용하여 추상화된 레벨로 작성된 커뮤니티 템플릿을 읽어 들여 해석하고 컨텍스트 브로커에게 커뮤니티가 전달받기를 원하는 상황을 등록한다. 컨텍스트 브로커로부터 상황정보를 전달받으면 커뮤니티 매니저는 해당 상황하에서 구동되어야 하는 커뮤니티를 검색하고 적합한 커뮤니티가 존재하면 커뮤니티 role과 멤버 서비스를 바인딩 하여 커뮤니티를 구성하는 작업을 시작한다. 커뮤니티의 멤버 서비스는 서비스 디스커버러를 통해 검색하게 된다. 구성 과정에서 커뮤니티 템플릿에 기술된 role과 적합하게 대응되는 멤버 서비스를 찾지 못한 경우 메타서비스의 의미적 관계 정보를 통해 대체 가능한 추상 서비스들로 재검색하여 커뮤니티를 구성한다.

또한 커뮤니티는 생성, 구성, 활동, 수면, 소멸이라는 라이프 모델을 가지는데 이 모델에 따라 커뮤니티의 라이프 사이클 단계를 관리하고 제어하는 기능을 제공한다. 이 정보들은 커뮤니티의 상태를 모니터하는 자료로도 사용된다[6,7].

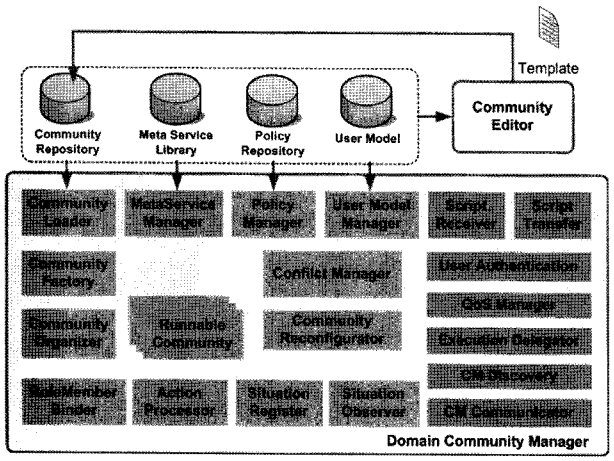

그림 7 커뮤니티 매니저 아키텍처

커뮤니티 매니저의 아키텍처는 그림 7과 같이 구성되었으며 주요 모듈의 기능은 다음과 같다.

- Community Loader: 커뮤니티 템플릿 저장소로부터 커뮤니티 템플릿을 로딩 하여 커뮤니티 템플릿의 인스턴스인 커뮤니티 오브젝트를 생성한다.
- Community Factory: 커뮤니티 템플릿을 커뮤니티 오브젝트화 하여 저장하고 있으며 사용자 컨텍스트와 선호도를 반영한다.
- Community Organizer: 커뮤니티 인스턴스의 멤버 서비스를 구성하고 서비스들의 action을 동작 환경 및 사용자 환경에 맞게 조직한다.
- Community Reconfigurator: 커뮤니티 재구성 요청이 들어오면 커뮤니티의 멤버 후보를 재검색하고 인터페이스를 조정하는 작업을 거쳐 커뮤니티를 재구성 한다.
- Action Processor: 커뮤니티 멤버들에게 보내는 서비스 실행에 대한 정보를 액션 큐에 우선순위를 부여하여 관리하고 서비스의 충돌이 발생하지 않게끔 조정한다.
- Situation Register: 템플릿에 기술된 상황을 컨텍스트 브로커에 등록한다.
- Situation Observer: 외부에서 메시지 브로커를 통해 입력 받은 상황정보를 감시하고 있다가 관련 커뮤니티 인스턴스의 Situation Listener에게 전달한다.
- User Model Manager: 사용자의 프로퍼티, 선호도 등을 사용자 컨텍스트로 관리한다.
- Script Receiver: 커뮤니티 템플릿을 전송 받아 처리한다.
- CM Communicator: 커뮤니티 매니저간의 통신을 담당한다.
- Script Transfer: 커뮤니티 템플릿에 대한 요청을 받으면 커뮤니티 템플릿 저장소에 있는 전체 템플릿 또는 요청 타입에 맞는 템플릿을 검색하여 요청자에게 전송한다.
- Execution Delegator: 커뮤니티 실행의 대행에 대한 요청을 접수 받아 처리한다. 커뮤니티 템플릿을 전송 받을 수도 있고 커뮤니티 인스턴스를 전송 받을 수도 있으며 이에 대한 구분은 커뮤니티의 상태 값으로 구분한다. 이 상태 값에 따라 요청 정보를Community Organizer 또는 Community Executor에 전달한다.
- User Authentication: 커뮤니티 실행 요청자에 대한 인증 절차를 처리한다.
- Policy Manager: 도메인 정책을 관리한다.

4.3 커뮤니티 에디터

유비쿼터스 컴퓨팅 환경에서 사용자에게 서비스를 제공하기 위해선 물리적 인프라 및 서비스 인프라뿐 아니라, 다양한 환경에서 동적으로 서비스들이 적응하는 것을 돕기 위한 서비스의 추상화 기술이 요구된다. 이를 위하여 커뮤니티 컴퓨팅은 커뮤니티 메타포로 서비스를 추상화하여 협업 서비스로 모델링하고 이를 커뮤니티 템플릿으로 기술하는 것을 지원하고 있다. 기존에는 개발자가 CDL 언어로 커뮤니티 템플릿을 기술하고 템플릿 파일을 커뮤니티 매니저의 로딩 폴더에 복사하는 과정을 수동으로 하였으나 개발자가 직접 CDL을 작성하는 과정에서 철자가 틀리거나 반복된 구문을 계속 작성해야 하는 등 불편함이 야기되었다. 따라서 다이내믹 커뮤니티 에디터를 개발하여 UML 다이어그램으로 클래스를 설계하는 것처럼 CDL 다이어그램으로 커뮤니티 템플릿을 디자인할 수 있도록 하고 이에 따라 자동으로 커뮤니티 템플릿이 기술되도록 하였다.

이러한 커뮤니티 에디터는 메타 서비스 저장소를 참조하여 커뮤니티 템플릿을 생성할 수 있는 GUI 환경을 제공한다. 에디터를 통해 공간에 사용되는 컨텍스트, 서비스, 메타 커뮤니티를 기반으로 하여 사용자는 쉽게 지능 공간 내의 복합 서비스를 커뮤니티 모델 언어로 제작할 수 있다. 또한 커뮤니티 에디터는 통합 개발환경인 이클립스의 플러그인 형태로 개발되어 쉬운 커뮤니티 개발환경을 제공한다.

커뮤니티 에디터는 다음과 같은 기능을 지원한다.

- 도메인 모델 정의

커뮤니티를 모델링 할 수 있는 스키마 파일과 실행 타임에서 사용 가능한 메타 서비스를 입력 모델로 받아들인다. 또한 입력 모델을 기반으로 그래픽 편집 기

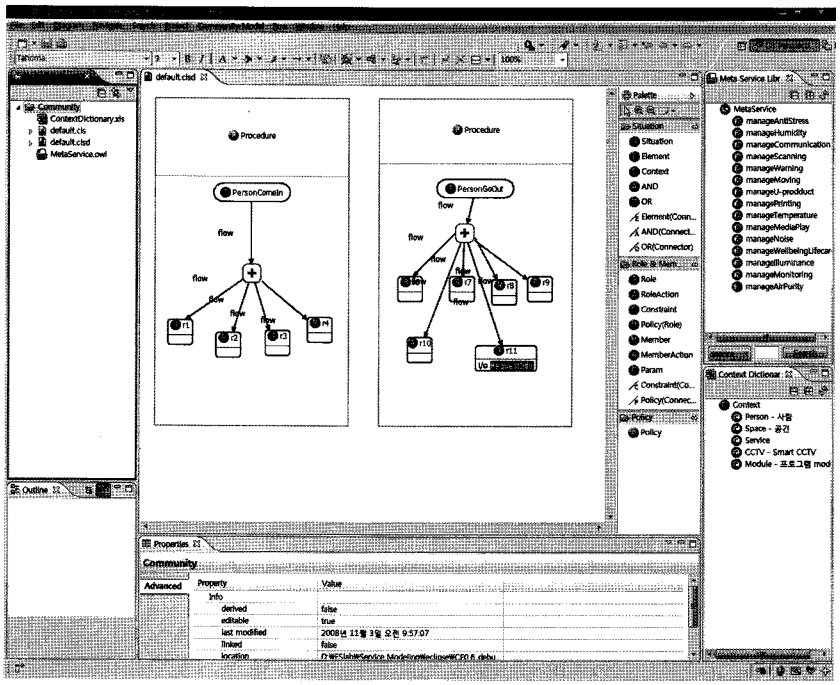

그림 8 커뮤니티 에디터

능을 제공하기 위한 기타 모델들을 정의한다.

- 데이터 가공 및 처리

입력 모델로 들어온 스키마 파일과 메타 서비스를 해석할 수 있어야 한다. 또한 이 해석된 내용을 바탕으로 커뮤니티를 디자인하고 이에 따른 커뮤니티 템플릿이 자동으로 생성될 수 있어야 한다.

- 사용자 뷰 처리

Model Data나 Controller에서 넘겨받은 Data를 화면에 보여주는 기능과 사용자의 입력을 받을 수 있는 뷰를 제공한다.

- CDL다이어그램 처리

커뮤니티 에디터가 그래픽 에디터로서 커뮤니티 모델에 따라 CDL다이어그램이 생성되고 처리된다.

- 편집도구에서의 자동완성 기능

다이내믹 커뮤니티 에디터에서 커뮤니티 템플릿 디자인할 때 개발자의 타이핑에 의존하지 않도록 편집도구에서의 자동 완성기능을 제공한다.

- 이클립스 플러그인

다이내믹 커뮤니티 에디터는 이클립스 환경에서 통합적으로 사용될 수 있는 이클립스 플러그인 형태의 에디터로 구현, 배포된다.

- 파일 시스템

커뮤니티를 모델링 하여 기술하는 시점에 참조할 수 있는 메타 서비스들의 저장소가 존재한다. 또한 다이내믹 커뮤니티 에디터로 자동 생성된 CDL코드를 저장하는 저장소도 존재한다. 즉, 커뮤니티 디자인 1개는 템플릿 파일1개 생성을 의미한다.

- CDL 스키마, 메타서비스 온톨로지 변경에 따른 처리

커뮤니티 모델 및 메타 서비스 온톨로지에 대한 연구가 발전함에 따라 보완되는 CDL 스키마와 메타서비스 온톨로지 스키마를 유연성 있게 흡수하고 확장하는 것이 용이하도록 컴포넌트 단위로 모듈화 하여 개발한다.

4. 커뮤니티 컴퓨팅 적용 사례

커뮤니티 컴퓨팅 시스템이 적용 가능한 분야는 가정에서 사용자의 편의를 돕는 리빙-라이프케어 분야뿐만 아니라 사람들이 붐비는 거리나 백화점, 쇼핑센터와 같은 공공장소에서 발생하는 강도나 미아 같은 치안 문제에도 적용될 수 있다. 또한 개인의 건강이 우선되는 병원이나 노인 복지 센터에서 헬스 케어를 위하여 적용될 수 있고 지진과 같은 자연 재해에 대한 대처를 위해 사용될 수도 있을 것이다. 커뮤니티 컴퓨팅이 적용될 수 있는 도메인은 다양하다고 할 수 있으나 실제 커뮤니티 컴퓨팅 기술을 이용하여 시스템을 구축하는 경우가 기존 기술을 이용하는 것에 비해 효율적인지는 실제 구현하고자 하는 응용 시스템에 따라 달라질 수 있을 것이다.

본 연구에서는 커뮤니티 매니저와 상황인지 시스템을 사용하여 홈 도메인에서 사용자의 쾌적한 실내 환경을 위한 커뮤니티 서비스를 구축하고 시연하였다.

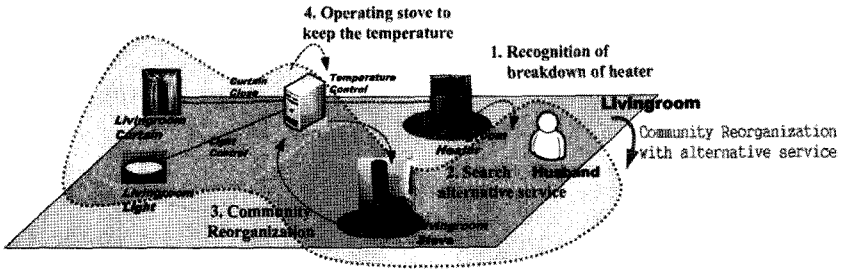

그림 9 지능적 실내 환경제어 커뮤니티 서비스

서비스 시나리오는 다음과 같다. 사용자가 집에 들어오면 상황인지 시스템은 이를 인지하여 컨텍스트를 커뮤니티 매니저에게 전달한다. 커뮤니티 매니저는 사용자의 선호도에 따라 실내 조도, 온도, 습도 등을 유지하기 위한 실내 환경 커뮤니티를 생성하고 서비스 검색자를 통해 실내에 존재하는 기기제어 서비스를 멤버로 구성하여 장치를 제어하게 된다. 만약 커뮤니티를 구성하고 있는 멤버가 기능을 못하게 되었을 경우 대체 가능한 멤버로 재구성하여 커뮤니티 서비스를 지속할 수 있다. 사용자가 잠자리에 들면 숙면을 위한 커뮤니티가 구성되어 최적의 수면 환경을 제공해 준다. 만약 사용자가 수면을 취하는 중 다른 사용자가 TV소음을 내고 있을 경우 서비스 충돌을 감지하여 소음을 내는 장치를 적절히 제어하여 숙면을 돕게 된다. 위의 시나리오를 커뮤니티 컴퓨팅 시스템을 이용하여 구축하고 시연하여 커뮤니티 컴퓨팅의 가능성과 효율성을 실험해 보았다.

5. 결 론

커뮤니티 컴퓨팅은 유비쿼터스 공간을 구성하는 요소들이 사용자에게 서비스를 제공하기 위하여 자유롭게 커뮤니티를 생성하고 그 안에서 요소들의 협업을 통해 서비스가 제공되는 개념을 기반으로 한다. 이것은 기존의 유비쿼터스 컴퓨팅 연구들을 바탕으로 설명된 유비쿼터스 컴퓨팅이 가지고 있는 특징과 그것을 만족시키기 위한 유비쿼터스 컴퓨팅 시스템의 요구사항을 상위 레벨에서 최대한 고려하여 점차 구체화시키는 개발 방법론이다. 커뮤니티 컴퓨팅의 이론적 기반이 되는 기술로는 멀티에이전트 시스템과 서비스 컴포지션, 상황인지, 서비스 및 지식을 표현하기 위한 온톨로지, 학습과 추론 등의 지능화 기술 등이 있다. 또한, 현재 그룹 컴퓨팅과 퍼베이시브 컴퓨팅, 유사 커뮤니티 컴퓨팅 등 많은 나라에서 유비쿼터스 컴퓨팅과 관련한 다양한 프로젝트를 진행하고 있다. 하지만, 우리가 제시하는 커뮤니티 컴퓨팅은 이러한 프로젝트들과 확연히 차별화되는 독자적인 개발 방법론이다.

SOA 기반의 커뮤니티 컴퓨팅 구현에서 커뮤니티 추상화를 위한 메타 서비스 정의하였고 메타 서비스를 사용하여 커뮤니티를 기술하기 위해 커뮤니티 기술언어를 정의하였다. SOA 기반의 커뮤니티 컴퓨팅을 위한 도구로 커뮤니티 에디터와 커뮤니티 매니저가 개발되었다. 마지막으로 커뮤니티 컴퓨팅을 검증하기 위해서, 앞서 설명한 도구와 기술언어를 사용하여 스마트 홈을 구축해 보았다.

커뮤니티 컴퓨팅과 관련한 연구는 궁극적으로 자율적인 커뮤니티를 구성하고 유비쿼터스 컴퓨팅 공간에서 효과적으로 서비스를 제공함으로써 인간의 삶을 보다 풍요롭게 하는데 목적이 있다. 현재 커뮤니티 컴퓨팅이 직면하고 있는 문제는 개발 사례들이 아직까지는 다소 정적인 컴퓨팅 단계에 머물러 있다는 점이다. 이러한 아직 미숙한 커뮤니티 컴퓨팅은 퍼베이시브 컴퓨팅과 비교하여 차별화된 특징을 보여주지 못하고 있지만 우리가 제시하고 있는 최종 목표인 자율적 커뮤니티 실현 단계에 도달한다면, 커뮤니티 컴퓨팅은 현재 진행되고 있는 유비쿼터스 컴퓨팅 분야의 모든 개발자들이 추구하는 하나의 개발 방법론으로 사용될 수 있을 것이다.

참고문헌

[1] Puneet Gupta, "Evolving a pervasive IT infrastructure: a technology intefration approach", Pers Ubiquit Comput 2004, 8:31-41

[2] M. Wooldridge, Nicholas R. J. The Gaia Methodology for Agent-oriented Analysis and Design, Autonomous Agents and Multi-Agent Systems, 3, 285-312, 2000

[3] Jinghai Rao and Xiaomeng Su, A Survey of Automated Web Service Composition Methods, SWSWPC 2004

[4] Harry Chen, "An Intelligent Broker Architecture for

Context-Aware Systems", TechReport, PhD Dissertation Proposal, December 2002
[5] 강경란, 김민구, "커뮤니티 컴퓨팅 : 협업 기반의 환경 자동 적응의 컴퓨팅 모델", 한국 정보과학회지 12월호, 2006
[6] 김현숙, 조규찬, 이창열, 조위덕, "SOA 기반의 커뮤니티 컴퓨팅 시스템", 정보과학회 종합학술대회 커뮤니티 컴퓨팅 워크샵 발표집, 2007
[7] Hyeonsook Kim. Community Manager : Dynamic Collaboration Solution on Heterogeneous Environment, Proceedings of IEE ICPS 2006, 39-46, 2006
[8] 이창열, 조규찬, 김현숙, "자가 성장하는 상황 기반 사용자 모델을 이용한 개인화 커뮤니티 서비스 자동 제공 방법", 정보과학회논문지 컴퓨팅의실제 및 레터 제14권, 2008.10
[9] O. Lassila and R. R. Swick, "Resource Description Framework (RDF) model and syntax specification," W3C Working Draft WD-rdf-syntax-19981008, See http://www.w3.org/TR/WD-rdf-syntax. http://citeseer.ist.psu.edu/article/lassila98resource.html
[10] David Martin, et al, "OWL-S: Semantic Markup for Web Services", W3C Member Submission, 2004
[11] Jos de Bruijn, et al, "Web Service Modeling Ontology(WSMO)", W3C Member Submission, 2005
[12] 조위덕, 김민구, 이정원, "u-서비스 융합 커뮤니티 컴퓨팅", 진한M&B, 2008
[13] 조위덕, 이경전, 이호근, 권오병, 김경규, 이은종, "유비쿼터스 패러다임과 u-소사이어티", 진한M&B, 2006
[14] 조위덕, "u-지능공간 시스템 솔루션 - 기술총괄", 진한M&B, 2009
[15] 조위덕, "다이나믹 커뮤니티 컴퓨팅 시스템 플랫폼", 진한M&B, 2009
[16] 조위덕, "u-지능공간 모델링 및 테스트베드", 진한M&B, 2009
[17] 조위덕, "u-지능공간 시스템 기술 규격서", 진한M&B, 2009

조 위 덕

현 아주대학교 전자공학부 교수
현 아주대학교 유비쿼터스시스템연구센터장
현 정보통신부 21세기프론티어사업 (재)유비쿼터스컴퓨팅사업단장
전자부품연구원 시스템연구본부 본부장
영국 TTP/Cambridge GSM Division 공동개발연구원
미국 TCSI/Berkeley PCG Group 공동개발연구원
금성전기(현 LG전자) 기술연구소 DSP 연구실장
관심분야 : u-지능공간시스템개발, 스마트임베디드오브젝트개발(스마트베드, 스마트테이블, 스마트미러, 스마트 카메라 등)
E-mail : wdukecho@gmail.com

조 규 찬

아주대학교 정보통신전문공학 석사
아주대학교 정보통신전문공학 박사수료
현 아주대학교 유비쿼터스시스템연구센터 선임연구원
관심분야 : 유비쿼터스 플랫폼, 커뮤니티 컴퓨팅
E-mail : netopia@ajou.ac.kr

김 양 원

LG정보통신 통신망연구소 선임연구원
LG전자 디지털S/W연구소 연구기획팀장
㈜위드웨이브 이동통신시스템연구소 연구소장
현 아주대학교 유비쿼터스시스템연구센터 연구개발그룹장
관심분야 : Ubiquitous Smart Space Design, u-Service Platform Architecture, Community Computing Model
E-mail : kimyw@ajou.ac.kr

최 재 동

1992~1998 인천대학교 영어영문학과 학사
2005~2007 아주대학교 정보통신공학과 석사
2000~2003 플럼라인, IT 컨설팅팀장
2003 미디어포드, 기획팀장
2003~현 아주대학교 유비쿼터스시스템연구센터
E-mail : gentle@ajou.ac.kr

신 진 아

1996~1999 한동대학교 전산전자공학부 학사
2000~2001 한국정보통신대학교 공학부 석사
2002~2006 한국전자통신연구원 무선보안응용연구팀 연구원
2007~현재 유비쿼터스시스템연구센터 선임연구원
관심분야 : 유비쿼터스 플랫폼, 상황인지 미들웨어
E-mail : jashin@ajou.ac.kr