협업 기반의 유비쿼터스 컴퓨팅 시스템을 위한 추상화 모델 및 응용 시스템의 개발

정유나[O] 이정태 김민구
아주대학교 정보통신전문대학원
{serazade[O], jungtae, minkoo}@ajou.ac.kr

Abstraction Model and Application Development Process for Cooperative Ubiquitous Computing Systems

Youna Jung[O], Jungtae Lee, Minkoo Kim
Graduate School of Information and Communication, Ajou University

요 약

유비쿼터스 컴퓨팅 시스템을 설계하고 개발하기 위한 방법들 중 하나로서, 최근 다중 에이전트 모델을 이용하는 방식이 연구되었다. 다중 에이전트 모델을 이용함으로써 얻을 수 있는 장점도 있지만, 다중 에이전트 모델로는 유비쿼터스 컴퓨팅 시스템이 가지는 고유한 특성을 완벽히 기술하기는 부족한 면이 있다. 예를 들어, 유비쿼터스 컴퓨팅 시스템에서 어떤 서비스를 제공하기 위하여 여러 컴퓨팅 요소들이 협업을 해야 할 때, 이러한 협업 조직을 효과적으로 표현하기 어렵다. 즉, 다중 에이전트 모델에서는 이러한 협업 조직의 동적인 생성과 소멸, 그리고 동적인 조직의 구성방식과 서비스를 제공하기 위한 조직 내에서의 또한 조직들 간의 협업 방식을 기술하기가 쉽지 않다는 것이다. 게다가 대부분의 다중 에이전트 모델들은 시스템을 설계하는 것만을 고려할 뿐 시스템의 개발은 개발자에게 전가하고 있다. 그러나, 유비쿼터스 컴퓨팅 시스템처럼 다양한 기종이 복잡하게 분산되어 있는 경우에는 설계만으로 실제 시스템을 개발을 하는 것은 쉬운 일이 아니다. 따라서 본 논문에서는 컴퓨팅 요소들의 협업 조직을 커뮤니티라 하고, 유비쿼터스 컴퓨팅 시스템을 그러한 커뮤니티에 기반하여 기술하는 고 수준의 추상화 모델을 제시하였다. 또한 이러한 고수준의 추상화 모델로부터 실제 시스템 개발을 이끌어 낼 수 있도록 하기 위하여 Model driven architecture 방식을 적용하여, 유비쿼터스 컴퓨팅 시스템의 개발 과정을 정의하였다.

1. 서 론

이제까지 에이전트는 소프트웨어 개발자들에게 복잡한 분산 시스템을 좀 더 쉽게 이해하고 모델화하여 개발을 쉽게 하는 도구로서 사용되었다 [1]. 최근 들어, 관심을 받고 있는 유비쿼터스 컴퓨팅 시스템도 매우 복잡하게 분산되어 있는 시스템의 일례로서 [2], 사람의 생활에 필요한 서비스를 지원하기 위한 다중 에이전트 응용 시스템으로 볼 수 있겠다. 다중 에이전트 기반의 유비쿼터스 응용 시스템은 단지 다중 에이전트 시스템의 용용 분야를 유비쿼터스로 확장하는 것으로만은 충분하지 않다. 왜냐하면 많은 유비쿼터스 컴퓨팅 시스템은 서비스를 제공하기 위하여 협업 조직이 동적으로 생성되며, 이러한 조직 내에서 구성 멤버들간에 동적인 상호작용이 이루어지는 등의 고유의 특징을 지니기 때문이다. 그러나 기존의 다중 에이전트 모델들로는 이러한 협업 기반 유비쿼터스 컴퓨팅 시스템의 특징들을 완벽히 표현하기 어려운 면이 있다. 그러므로 협업 중심의 유비쿼터스 컴퓨팅 시스템에 적용 가능한 추상화 모델이 필요하다. 게다가 기존의 모델들은 시스템에 대해 높은 수준에서만 추상적으로 기술하고 있을 뿐이다. 그러나 이러한 고수준의 추상화 모델만으로 매우 복잡하게 분산되어 있는 시스템을 개발하는 것은 쉽지 않은 일이다. 따라서 본 논문에서는 다중 에이전트 기반의 협업 중심 유비쿼터스 컴퓨팅 시스템에 대하여 고수준의 추상화 모델뿐만 아니라 구현을 고려하는 모델까지 정의하여 하나의 유비쿼터스 컴퓨팅 시스템을 개발하는 과정을 제안하여 보고자 한다.

이를 위해, 본 논문에서는 목적 지향적 협업 조직의 동적 생성과 해체 및 조직 내에서 또는 조직들 간의 동적인 협업 등의 유비쿼터스 컴퓨팅 시스템의 특징을 표현하기 위해 협업 조직을 커뮤니티로 정의하고 유비쿼터스 컴퓨팅 시스템을 커뮤니티를 기반으로 기술할 수 있는 추상화 모델을 제안하였다. 또한, Model Driven Architecture 개발 방식을 적용한 유비쿼터스 컴퓨팅 시스템의 개발 과정을 제안하여 보았다. 제안된 개발 과정으로 다중 에이전트 기반의 유비쿼터스 컴퓨팅 시스템을 좀 더 일관되고 쉽게 개발할 수 있을 것으로 기대한다.

[1] "본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅및네트워크원천기반기술개발사업의 지원에 의한 것임"

본 논문의 구성은 아래와 같다. 2장에서는 관련연구와 함께 연구의 필요성을 제시하고, 3장에서는 제안하고자 하는 협업 중심 유비쿼터스 컴퓨팅 시스템에 대한 추상화 모델과 시스템 개발 과정에 대해 간략히 소개한다. 이어서 4장에서는 개발 과정에 사용되는 모델들을 세부적으로 소개하고, 그들 간의 변환관계를 제시한다. 5장에서는 4장에서 제안한 모델들을 이용하여 소규모의 유비쿼터스 컴퓨팅 시스템을 구현한 결과를 소개하고, 마지막으로 6장에서는 결론과 함께 향후 연구를 논하도록 한다.

2. 관련 연구 및 연구의 필요성

지금까지 유비쿼터스 서비스를 제공하는 시스템을 개발하기 위해 다양한 방식이 시도되었다. 그 중에서도Gaia, AALADIN, BRAIN과 같은 다중 에이전트 기반의 방식들과 PICO나 Active Space와 같은 미들웨어 기반 방식들이 주목을 받고 있다. 본 장에서는 이러한 관련 연구들에 대하여 간략히 살펴보도록 한다.

2.1 다중 에이전트 기반의 유비쿼터스 서비스를 제공하는 시스템 개발 방식

에이전트의 유동적이고 자율적인 문제 해결 방식과 활발한 상호작용이 유비쿼터스 서비스를 제공하는 시스템의 특성을 만족시켜 많은 유비쿼터스 서비스를 제공하는 시스템이 다중 에이전트 방식을 적용하여 개발되었다. 다중 에이전트 방식은 시스템의 요구사항을 만족시키기 위해 필요한 유비쿼터스 개체(에이전트나 분산 시스템 등)를 알아내는 것에 목적이 있다. 따라서 결과물로 유비쿼터스 개체들이 정의된다. 그러나 앞서 말했다시피, 유비쿼터스 서비스를 제공하는 시스템을 기존에 존재하는 개체들을 기반으로 개발하는 경우에는 유비쿼터스 서비스를 제공하는 시스템에 존재하는 개체들의 특성과 기능들은 이미 정의되어 있다고 볼 수 있다. 따라서 이러한 상황에서는 필요한 개체들을 정의하는 것이 아니라 주어진 개체들을 이용하여 시스템의 목적에 충족하는 방법을 알아내는 것이 중요하다.

Gaia. Gaia는 다중 에이전트 시스템의 개발을 위한 에이전트 기반의 분석 및 설계를 위한 방법을 제안하였다[4][5]. Gaia에서는 다중 에이전트 시스템을 서로 상호작용을 하는 여러 종류의 역할(role)들로 이루어진 조직들의 집합으로 정의하였다. Gaia가 제시한 방법론을 이용하여 시스템에 대한 요구사항을 모델로 표현하고 이를 더 구체적인 모델로 발전시키는 방식을 통하여 시스템에 대한 설계를 할 수 있다.

AALADIN. AALADIN[6]은 조직 개념을 기반으로 다중 에이전트 시스템을 기술하는 메타 모델로서 어떠한 종류의 조직도 그룹(group), 에이전트, 그리고 역할(role)의 개념으로 기술이 가능하도록 하였다. 다중 에이전트 시스템의 조직적 구조는 그룹 구조들의 집합으로 정의되며, 하나의 그룹 구조는 그룹이 포함할 수 있는 모든 역할과 그들간의 상호작용(interaction)으로 정의한다. 하나의 에이전트는 여러 그룹에서 여러 종류의 역할을 수행할 수 있으며, 역할은 한 그룹 내에서 특정 에이전트의 기능, 서비스, 식별자를 추상화 표현이다. AALADIN의 확장버전에서는 그룹의 생성과 그룹에 대한 에이전트의 가입 및 탈퇴, 그리고 그룹의 역할 획득에 대한 방법도 고려하고 있다.

BRAIN. BRAIN[7]은 다중 에이전트 시스템에서 에이전트들 간의 상호작용을 역할을 기반으로 기술하고 이를 개발하기 위한 프레임워크이다. BRAIN에서는 역할을 XML 방식의 XRole을 이용하여 기술하는데, 하나의 역할은 에이전트가 수행하는 행동(action)들과 에이전트가 다른 곳에서 받거나 센서를 통하여 인지하는 이벤트(event)들로 표현한다. 이러한 정의를 기반으로 BRAIN에서는 역할간의 상호작용을 위한 인프라스트럭쳐(interaction infrastructure)인 Rolesystem을 구현하여 소개하였다.

2.2 미들웨어 방식

미들웨어 방식의 목적은 자원을 관리하고 상황 정보를 인지하며 유비쿼터스 서비스를 제공하는 시스템의 개발과 실행을 지원하는 미들웨어 인프라스트럭처를 제공하는 것에 있다.

Active Space의 Gaia. Active Space 프로젝트[3]에서는 Gaia라는 미들웨어를 소개하였다. Gaia는 소프트웨어와 이기종의 네트워크 디바이스들을 조정하는 분산 미들웨어 인프라 스트럭쳐이다. 프로젝트에서는 Gaia를 이용하면 유비쿼터스 환경을 서로 분리된 이 기종들 간의 집합이 아니라 프로그램이 가능한 Active Space로 볼 수 있다고 주장한다. 그러나 그러한 Active Space를 기술할 수 있는 추상화 모델은 제시하고 있지 않다. 하지만 Active Space를 구성하는 유비쿼터스 개체들과 이들 간의 상호 작용을 기술할 수 있는 추상화 모델은 반드시 필요하다.

PICO(Pervasive Information Community Organization). PICO[10] 는 유비쿼터스 환경에서 끊임 없이 실시간으로 사용자가 원하는 서비스를 제공하기 위하여 이를 처리할 수 있는 커뮤니티를 동적으로 생성하여 운용하기 위한 미들웨어 프레임워크이다. 커뮤니티는 목적을 가지는 협업 조직의 구조를 추상화하기 위하여 도입한 것으로, 이러한 커뮤니티 개념은 이미 여러 에이전트 협업 모델에서 소개된 바 있다[8][9]. PICO 프로젝트의 주목할 만한 성과는 에이전트간의 협업을 위한 프레임워크로서 커뮤니티 컴퓨팅 개념을 소개한 것이라고 할 수 있다. 커뮤니티 컴퓨팅 개념은 이질적인 하드웨어와 에이전트 간의 협업과 의사소통을 가능하게 하는 플랫폼을 제공한다. PICO의 커뮤니티는 공통의 목적을 달성하기 위해 협력하는 하나 이상의 에이전트들로 정의되는데, 이는 에이전트들 간의 협업을 위한 프레임워크를 제공한다. 커뮤니티 컴퓨팅 개념은 이질적인 유비쿼터스 환경에서 자동적이고 연속적인 서비스를 제공하기 위하여 컴퓨팅 요소간의 실시간 협업이라는 요구사항을 충족시켜 주고 있다. 따라서 우리의 협업 기반 유비쿼터스 서비스를 제공하는 시스템을 기술하는 추상화 모델에서 컴퓨팅 요소들이 구성하는 목적지향적인 조직 구조를 이러한 커뮤니티 개념을 응용하여 기술하였다. 본 논문에서는

커뮤니티 개념으로 협업 구조를 모델링 한 유비쿼터스 서비스를 제공하는 시스템을 커뮤니티 컴퓨팅 시스템이라고 정의하여 사용하겠다. 커뮤니티 컴퓨팅 시스템을 표현하기 위해서는 추상화 모델이 필요하다. PICO에서도 모델을 제시하였으나 아직 설명되지 않은 부분이 있다. 예를 들어, 모델에서는 커뮤니티와 멤버의 생성과 소멸, 실제 객체에 커뮤니티의 멤버를 할당하는 방법, 그리고 커뮤니티의 목적을 달성하기 위해 수행하는 멤버들의 협업 과정 등이 표현되지 않는다. 따라서 본 논문에서는 그러한 부분까지 모두 고려한 추상화 모델을 제시하고자 한다.

3. 커뮤니티 컴퓨팅 모델 및 커뮤니티 컴퓨팅 시스템 개발과정

3.1 커뮤니티 컴퓨팅 모델 (CCM)

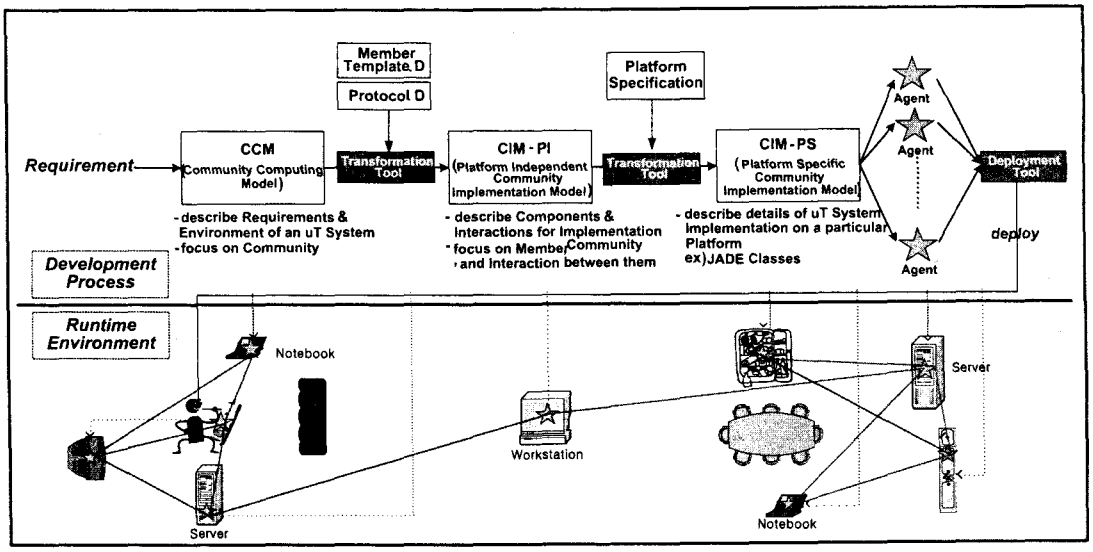
CCM 은 커뮤니티 컴퓨팅 시스템에 대한 가장 높은 수준의 추상화 모델로서, 시스템의 환경과 요구사항을 기술하는 것을 목적으로 한다. 하나의 커뮤니티 컴퓨팅 시스템은 컴퓨팅 요소들과 커뮤니티들의 집합으로 기술할 수 있다. 따라서 CCM 에서는 전체 시스템을 사회(Society)라는 개념으로 추상화하고 사회 기술 부분에 커뮤니티 컴퓨팅 시스템에 존재할 수 있는 컴퓨팅 요소를 표현하고, 요소들간의 협업을 통하여 제공될 수 있는 서비스 (해결될 수 있는 요구사항)를 커뮤니티로 표현한다. CCM 을 좀 더 자세히 살펴보면, 어떤 에이전트가 커뮤니티의 특정 역할(Role)을 맡기 위한 조건을 'Cast'에서 기술하여 커뮤니티에 대한 에이전트의 소속을 관리한다. 커뮤니티 자체도 자신이 달성할 목적(Goal)을 명시하여, 어떤 에이전트에 의해 그러한 목적이 인지되면 커뮤니티가 생성되고, 이후 목적이 달성되면 커뮤니티가 해체되는 것으로 커뮤니티의 생성과 해체를 관리한다. 또한 시스템 내의 모든 커뮤니티는 해당 사회에 속해있는 요소들로 구성된다고 보고, CCM 의 사회기술 부분에서는 사회에 대한 에이전트의 소속을 관리하기 위해 특정 사회 내의 모든 에이전트 멤버가 가져야 하는 조건을 제시하도록 하였다.

CCM 의 구조를 자세히 살펴보기 전에, 먼저 커뮤니티 컴퓨팅에서 사용되는 개념들을 소개한다.

- 멤버(Member): 커뮤니티를 이루는 구성 요소로서, 하나의 에이전트로 구현된다. 컴퓨팅 디바이스, 하드웨어, 사람, 또는 소프트웨어 모두가 멤버가 될 수 있다. 각 멤버는 커뮤니티에서 역할을 맡고 있지 않을 때에는 자신의 작업(예를 들어, 가로등이 빛을 밝힌다던가 시계가 시간을 표시하며 특정 시간에는 알람을 울리는 일등이 이에 속한다)을 수행하고 있다가 특정 목적(Goal)이 생겨 커뮤니티가 구성되면 커뮤니티에 속해 목적을 달성시키기 위해 자신의 역할을 수행한다. 커뮤니티에 속하여 역할을 수행하는 것도 결국은 목적을 달성하기 위한 프로토콜에 맞추어 자신의 고유 작업을 수행하는 것이다. 본 논문에서는 시스템에 존재할 수 있는 멤버의 종류와 이들이 가지는 속성(속성값은 멤버 개체에 따라 다르다)과 고유 작업들이 미리 정해져 있다고 가정하고 이를 모델에서 기술한다.
- 커뮤니티(Community): 여러 에이전트 멤버들이 특정 목적을 달성하기 위하여 형성하는 협업 조직으로서, 하나의 커뮤니티는 구성 멤버들이 맡을 역할(Role), 목적(Goal), 그리고 이를 달성하기 위한 방법을 기술한 프로토콜(Protocol)들로 기술한다. 이 때, 커뮤니티에서 해당 역할을 수행하는 멤버로 선택(cast)될 수 있는 조건을 제시하여 커뮤니티에 대한 멤버 개체(instance)의 소속을 관리한다. 특정 멤버에 의해 목적(goal)이 감지되면 해당 목적을 성취할 수 있는 커뮤니티를 알아내고, 커뮤니티를 구성하는 각 역할에 맞는 멤버들을 선택(Casting)하여 커뮤니티를 생성한다. 그리고, 이들 간의 협업으로 목적이 달성되면 해당 커뮤니티는 해체된다. 좀 더 자세한 내용은 5 장에서 실제 시나리오를 바탕으로 기술하도록 하겠다. 본 논문에서는 시스템에 존재할 수 있는 커뮤니티의 종류와 그 커뮤니티의 구성이나 멤버들 간의 협업 방식이 미리 정해져 있다고 가정하고 모델을 작성하였다.
- 목적(Goal): 하나의 커뮤니티가 만족시켜야 할 특정 상황(Situation)으로서, 해당 커뮤니티를 구성하는 이유라고 할 수 있다.
- 프로토콜(Protocol): 커뮤니티 내의 멤버들이 특정 목적(goal)을 달성하기 위해 협업하는 방식을 기술한 것으로서, 여러 종류의 통신 메시지와 각 멤버의 자체 작업 수행들의 나열로 정의된다. 하나의 프로토콜을 수행하는 동안, 또 다른 목적(goal)이 발생하여 그를 달성하기 위한 또 하나의 프로토콜이 수행될 수도 있다.
- 사회 (Society): 전체 커뮤니티 컴퓨팅 시스템을 추상화한 개념이다. 커뮤니티 컴퓨팅 시스템이 처음 운용된 시점에서는 어떠한 커뮤니티도 없이 사회(Society)만 존재하는 형태가 될 것이다. 그 후에 여러 멤버 에이전트들이 그 사회(Society)에 소속됨으로써, 사회 내에 여러 멤버들이 존재하는 형태로 변할 것이다. 이윽고, 커뮤니티를 구성해야 하는 목적(Goal)이 생기면 비로소 커뮤니티가 구성되고 멤버들간의 협업에 의한 서비스가 제공된다. 사회가 소멸되기 전까지 이러한 커뮤니티들은 필요에 따라 동적으로 생성되고 소멸된다.

CCM 은 커뮤니티를 기술하는 부분과 사회를 기술하는 부분으로 구성된다. 커뮤니티를 기술하는 부분에서는 시스템에 존재 가능한 모든 커뮤니티에 대한 구조와 목적(Goal)들, 그리고 그에 대한 프로토콜들을 정의한다. 특정 커뮤니티의 종류에 대해 필요한 역할들, 목적들, 그리고 사용하는 온톨로지를 명시한다. 역할은 그것의 속성들과 그 역할을 맡을 수 있는 멤버의 조건으로 정의하는데, 속성은 속성의 이름과 가능한 값들의 집합으로 표현하는데, 만약 가능한 값의 범위를 알 수 없을 경우에는 생략할 수 있다. 속성들 중에서 그 역할로 필요한 멤버의 수를 말해주는 개수 속성은 가장 필수적인 속성으로, 역할 이름 바로 옆에 기술한다. 목적(Goal)은 목적의 이름과 이를 달성하게 해주는 프로토콜로 정의되는데, 이 때 프로토콜은 참여 역할들과 프로토콜 수행 도중에 수행될 수도 있는 또 다른 프로토콜을 명시하는 것으로 표현된다.

그리고 만약 커뮤니티가 특정 온톨로지를 사용한다면 그 이름을 기술하여 시스템 운용 중에 사용할 수 있도록 하였다. 사회(Society) 기술 부분에서는 그 사회의 이름과 멤버 개체가 그 사회에 속할 수 있는 조건을 기술한다.



[그림 1] 커뮤니티 컴퓨팅 시스템의 개발 과정과 운용 환경

3.2 커뮤니티 컴퓨팅 시스템 개발 과정

본 논문에서는 다중 에이전트 기반의 커뮤니티 컴퓨팅 시스템을 위한 고 수준의 추상화 모델과 이로부터 시스템을 구현하는 체계적인 개발 과정을 제안하고자 한다. 제안한 추상화 모델에서는 커뮤니티 개념을 이용하여 공통된 목적 아래 실시간으로 결성되는 협업 조직들과 그들 내부에서의 상호작용을 명시적으로 기술할 수 있다. 이를 기반하여 시스템을 구현하기 위해 MDA 개발 방식을 적용한 개발 과정을 통해 제안된 추상화 모델로부터 좀 더 구체적인 모델로의 변환과정을 통해 시스템 개발에 이르도록 한다. MDA 개발 방식을 따르기 위해서 본 논문에서는 서로 다른 추상화 수준을 표현하는 모델들과 각 모델을 기술하기 위한 언어를 정의하였다. 커뮤니티 컴퓨팅 시스템을 개발하기 위해서는 먼저 개발하고자 하는 시스템을 커뮤니티 개념에 입각하여 가장 고수준의 추상화 모델인 CCM(Community Computing Model)으로 기술한다. CCM은 시스템에 대한 요구사항과 운영 환경을 모델링 하는 것을 목적으로 한다. 이러한 CCM을 기술하기 위하여 CML(Community computing Modeling Language)를 정의하였다. CCM보다 좀 더 구체적으로 시스템을 기술하는 모델로서 CIM-PI (Platform Independent Community Implementation Model)을 정의하는데, 이는 CML에서 기술한 요구사항의 해결 방안이 실제 시스템에서 어떤 종류의 컴퓨팅 요소들과 그들간의 어떠한 상호작용에 의하여 해결되는지를 플랫폼에 독립적인 수준으로 기술한다. CIM-PI의 구조적인 틀과 커뮤니티 구성 정보들은 CCM으로부터 생성하고, 그 외에 나머지는 추가정보를 이용하여 개발자가 채워 넣는다. 이러한 CIM-PI의 모델링 언어로서 CIL-PI를 정의하여 사용한다. 그 다음 단계로, 실제 시스템이 구현될 특정 플랫폼을 고려하여 시스템을 기술한 CIM-PS(Platform Specific Community Implementation Model)으로 모델을 구체화시킨다. CIM-PS는 CIM-PI에 시스템이 어떻게 특정 에이전트 플랫폼에서 운용되는지에 대한 내용을 더하여 기술한 것으로, 이를 위한 모델링 언어로서 CIL-PS를 정의하여 사용한다. 이러한 CIM-PS의 틀도 CIM-PI로부터 추출되며, 나머지 부분은 개발자가 채운다. 위에서 제시한 CCM에서부터 CIM-PS까지의 모델 변환 과정을 통해서 시스템 구현을 위한 소스 코드의 틀을 생성해낼 수 있어서 실제로 이는 개발자의 프로그래밍 작업량을 현저하게 줄여준다. 즉, 제안된 개발 방식을 이용하면 기존의 거대한 유비쿼터스 서비스를 제공하는 시스템을 설계서만 가지고 프로그래밍하는 방식에 비해 좀 더 쉽고 체계적으로 시스템을 개발할 수 있다고 하겠다. 게다가 시스템의 전체 개발 과정을 커뮤니티라는 일관된 개념으로 체계적으로 관리할 수 있다는 것도 큰 장점이라고 할 수 있겠다. 개발자는 초기에 시스템을 커뮤니티의 관점에서 기술하는 것으로 유비쿼터스 서비스를 제공하는 시스템에서의 협업에만 초점을 맞추어 작업을 진행할 수 있고, 후에 컴퓨팅 요소들의 개별적인 행동에 대하여 초점을 맞추어 작업할 수 있다. 각 요소와 요소들간의 협업을 나누어 고려하면서 개발 작업을 진행함으로써 협업 중심의 유비쿼터스 서비스를 제공하는 시스템을 효율적으로 구현할 수 있도록 한다.

4. 결론

본 논문의 기여 사항은 다음과 같다. 협업 조직 기반의 유비쿼터스 서비스를 제공하는 시스템을 위한 추상화 모델로서 커뮤니티 컴퓨팅 모델을 제안하였다. 제안된 모델을 통해 기존의 에이전트 모델로는 완벽하게 표현되지 못했던 목적지향적인 협업 조직의 구조와 그들 간의 동적인 상호작용, 그리고 협업조직의 동적인 생성과 소멸을 커뮤니티 개념을 이용하여 모델로서 기술할 수 있다. 위와 같은 시스템을 다중

에이전트 환경에서 개발하기 위한 개발 과정을 제안하였다. 본 논문에서는 추상화 모델과 실제 개발 시스템간의 격차를 줄이기 위해 MDA 방식을 적용한 커뮤니티 컴퓨팅 개발 과정을 제안하였으며, 이를 위하여 서로 다른 추상화 수준을 가지는 모델들과 기술 언어들을 정의하였다. 그러나, 커뮤니티 컴퓨팅에 대하여 아직 완벽히 해결하지 못한 문제들이 많이 남아있으며, 이는 앞으로도 지속적으로 연구를 진행해야 할 것이다.

- 커뮤니티 모델의 기술 범위와 효용성 검증: 제안된 커뮤니티 모델이 유비쿼터스 서비스를 제공하는 시스템에서 발생할 수 있는 문제 패턴들 중에서 어떠한 것을 해결할 수 있으며, 그 효율성은 어떠한지를 좀 더 구체적으로 실험하여 검증해야 하겠다.
- 커뮤니티 컴퓨팅 시스템에서의 Policy: 제안된 모델이 실제적으로 적용될 수 있으려면 존재 가능한 충돌(Conflict)를 해결할 수 있어야 한다. 이를 커뮤니티 모델에서 Policy로 기술하여 시스템에 적용하고자 한다.
- 동적인 커뮤니티 구조 형성: 본 논문에서는 커뮤니티를 구성하는 역할들이 미리 정해져 있다고 가정하였다. 따라서 커뮤니티를 구성하는 멤버 에이전트의 종류와 해결할 수 있는 문제가 미리 정해져 있는 것이다. 그러나, 문제가 생기면 시스템에서 해결할 수 있는 문제인지를 판단하여 스스로 커뮤니티를 구성할 수 있으면 좀 더 다양한 문제를 해결할 수 있을 것으로 기대한다.
- 멤버들 간의 동적 협업: 현재 커뮤니티의 목적(Goal)을 달성하기 위한 협업 방법은 미리 프로토콜에 의해 정의되어 있다. 즉, 프로토콜로 정의되지 않은 방법으로는 문제를 해결할 수가 없는 것이다. 설사 사회(Society)에 좀 더 효율이 좋은 멤버 타입이 새로 등록되었다고 해도 커뮤니티 프로토콜을 수정하지 않는 한 이를 이용할 수 없다. 따라서 앞으로는 커뮤니티에 필요한 능력을 가지고 동적으로 멤버를 찾아서 그들간의 협업 패턴을 동적으로 만들 수 있으면 변화되는 상황에 맞춘 효율적인 방식으로 목적을 달성할 수 있을 것이다.

참고 문헌

[1]. Nicholas R. J. "On agent-based software engineering," Elsevier, Artificial Intelligence 117, 2000, 277-296

[2].Weiser M. "Ubiquitous Computing," Nikkei Electronics, December 6, 1993, 137-143

[3].Manuel Román, Christopher K. Hess. Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, and Klara Nahrstedt, "Gaia: A Middleware Infrastructure to Enable Active Spaces," In IEEE Pervasive Computing, Oct-Dec, 2002, 74-83

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

[5].R. Jennings, et. al. "Developing Multiagent Systems: The Gaia Methodology, ACM Transactions on Software Engineering and Methodology," Vol.12, No.3, July, 2003, 317-370

[6].J. Ferber and O. Gutknecht, "A meta-model for the analysis and design of organization in multi-agent systems," In Proceedings of 3rd International Conference on Multi-agent Systems (ICMAS'98), 1998

[7].G. Cabri, L. Leonardi, F. Zambonelli, "A Framework for Flexible Role-based Interactions in Multi-agent System," In Proceedings of the 2003 Conference on Cooperative Information Systems (CoopIS), Italy, 2003

[8].Jennings, N. R., et. al. "Transforming Standalone Expert Systems into a Community of Cooperating Agents," Int. Journal of Engineering Applications of Artificial Intelligence 6 (4), 2003, 317-331

[9].Wooldridge M. "An Introduction to Multiagent Systems," John Wiley & Sons, Reading, 2002

[10].Mohan Kumar, et. al. "PICO: A Middleware Framework for Pervasive Computing, Pervasive Computing," 1536-1268, 2003, 72-79

[11].Object Management Group. "Model Driven Architecture Guide," 2003

[12].INMOS Ltd, "OCCAM: Programming Manual," Prentice-Hall, Englewood Cliffs, NJ, 1984

[13].FIPA Standard, SC00037J, "FIPA Communicative Act Library Specification," 2002

[14].Huber M. j. "JAM Agent in a Nutshell version 0.65+0.76i," November 1, 2001

[15].F. Bellifemine, A. Poggi, and G. Rimassa. "JADE - A FIFA-compliant Agent Framework", In Proc. of Practical Application of Intelligent Agents and MultiAgents (PAAM' 99), London, UK, April 1999, 97-100