韓國IT서비스學會誌
第11卷 第3號
2012年 9月



# 유비쿼터스 환경에서 자발적 상호연동을 지원하는 역할 기반 응용 모델

정종윤* · 류기열** · 이정태***

# Role-Based Application Model for Supporting Spontaneous Interoperation in Ubiquitous Environments

Jong-Yun Jung* · Ki-Yeol Ryu** · Jung-tae Lee***

■ Abstract ■

The spontaneous interoperation is an important characteristic of ubiquitous applications and is closely related with mobility. The mobile components in ubiquitous environments are liable to appear in and disappear from one physical space to another. Because this characteristic certainly influences the structure and behavior of applications, they should adapt themselves to the changed environments by configuring their structure and behaviors. Consequently, developers are faced with the above challenging issue when they design and implement ubiquitous applications. The role concept is an efficient tool to model participant entities, their relationship, and collaboration, so role-based model are appropriate to describe a structure and behavior of software system. But, little attention has been given to reflect characteristics of ubiquitous applications. To tackle the problem, this study considers a ubiquitous application as a software organization which consists of software components and proposes an enhanced role-based application model for supporting spontaneous interoperation.

Keyword : Spontaneous Interoperation, Role-based Model, Software Organization, Ubiquitous Applications

논문투고일 : 2012년 01월 31일 논문수정완료일 : 2012년 08월 06일 논문게재확정일 : 2012년 08월 10일

* 아주대학교 정보통신 전문대학원
** 아주대학교 정보 및 컴퓨터공학부, 교수
*** 아주대학교 소프트웨어융합학과, 교수

# 1. 서 론

유비쿼터스 응용(ubiquitous application)의 핵심은 환경에 내재된 컴포넌트-컴퓨팅 장치-들의 자발적인 상호연동(spontaneous interoperation)을 통한 협업이다. 자발적 상호연동은 유비쿼터스 시스템의 주요한 특성이자 요구사항이다[8, 11]. 유비쿼터스 환경에서 컴포넌트들은 물리적으로 분산되어 있으며 자유롭게 이동이 가능하다. 컴포넌트들의 이동은 응용의 상황 변화를 초래하고 이는 응용의 구조와 행위에 영향을 준다. 유비쿼터스 응용은 변화된 환경에 적응하기 위해 기존 컴포넌트를 새로운 컴포넌트로 교체하고 컴포넌트들 사이의 새로운 관계를 설정하고 이들 사이의 상호동작(interaction)을 지원해야 한다. 이를 위해 응용은 필요한 컴포넌트들을 동적으로 발견하고 이들을 응용에 참여하게 한다. 자발적 상호연동은 컴포넌트의 이동성(mobility)과 응용을 구성하는 실제 컴포넌트들이 실행시간에 참여하는 비결정적 특성(non-deterministic)과 밀접한 관련을 가진다[3, 4].

유비쿼터스 환경에 내재된 컴포넌트들은 특정 응용과는 무관하게 개발되기 때문에 특정 응용의 요구사항을 완전히 만족시키기 어렵다. 따라서 응용 개발자는 비즈니스 로직 뿐만 아니라 컴포넌트들 사이의 자발적 상호연동에 필요한 분분을 작성해야 한다. 자발적 상호연동을 비롯해서 유비쿼터스 환경의 주요한 특성에 대한 연구는 미들웨어와 같은 하부 구조(software infrastructure)에 초점을 두고 있다[5, 8]. 응용의 개발에 필요한 개발 방안이나 모델에 대한 연구가 부족한 현실이다. 유비쿼터스 응용은 소프트웨어 하부 구조에서 제공하는 기능을 통해서 자발적 상호연동을 지원할 수 있지만 여전히 유비쿼터스 응용을 개발하는 것은 어렵고 힘든 작업이다.

먼저, 컴포넌트의 비결정적 특성을 지원하기 위해서 개발단계에서 응용을 기술할 수 있는 추상화 모델이 필요하다. 역할은 구성요소에 대한 요구사항과 이들 사이의 협업, 즉 상호동작을 기술하기에 적합하다. 역할의 요구사항을 만족시키는 컴포넌트는 그 역할을 수행하는 행위자(player)가 될 수 있다. 역할은 MAS(Multi Agent System), CSCW (Computer supported Cooperative Works), RBAC (Role-based Access Control) 등과 같은 다양한 소프트웨어 분야에 광범위하게 적용되고 있다[14]. 본 논문은 유비쿼터스 응용에 참여하는 컴포넌트와 이들 사이의 상호동작을 역할 을 이용해 추상화한다.

그러나 유비쿼터스 응용에 기존의 역할 모델들을 적용하기에는 부적합하다. 이는 기존 모델들이 자발적 상호연동과 관련한 중요한 특성인 이동성을 고려하지 않기 때문이다. 컴포넌트의 이동은 상황 변화를 초래하고 참여하고 있는 응용의 상황에 영향을 준다. 따라서 응용은 변화된 상황에 적응하기 위해 자신의 구조와 행위를 제어해야 한다. 또한, BRAIN[2]이나 E-CARGO[13]와 같은 기존의 역할 모델들은 에이전트, 사용자, 그리고 객체와 같이 동질적인 개체들을 컴포넌트로 보고 역할로 추상화한다. 그러나 유비쿼터스 환경에 내재된 컴포넌트들은 센서나 임베디드 장치에서 고성능 서버까지 다양하다. 컴포넌트들이 할 수 있는 작업의 수준이 다르다는 것을 의미한다. 이런 컴포넌트들의 이질성을 고려하는 것이 유비쿼터스 응용에 적합하다.

본 논문은 유비쿼터스 환경에 내재된 컴포넌트들, 즉 역할의 행위자에 대한 고찰을 통해 선행 연구에서 제안했던 능동 역할과 수동역할 개념을 도입하였다[7]. 이를 통해 자발적 상호 연동을 지원하는 유비쿼터스 응용을 추상화하기에 적합한 역할 기반의 응용 모델을 제안한다. 본 논문은 ROAD (Role-Oriented Adaptive Design)[3]를 비롯한 최근의 역할 연구가 지향하고 있는 조직적 관점의 역할을 적용하였다. 유비쿼터스 응용은 자발적 상호동작이 필요한 상황을 인지하고 필요한 경우에 응용의 구조와 행위를 조정해야 한다. 이를 위해 응용에 참여하는 컴포넌트들은 자발적 상호작용에 필요한 작업을 수행해야 한다. 그러나 유비쿼터스 환

경의 컴포넌트들은 응용과는 무관하게 개발되기 때문에 특정 응용의 요구사항을 만족시키기 어렵다. 따라서 응용에 참여하는 컴포넌트들의 자발적 상호연동을 지원하기 위한 추가적인 기능이 필요하다. 이런 기능은 컴포넌트들의 일정한 패턴을 가지는 상호동작 모델로 정의할 수 있다. 본 논문은 자발적 상호연동에 필요한 기능을 능동역할과 수동역할들 사이의 상호동작 모델로 정의하였다. 이를 통해 자발적 상호연동과 관련한 기능은 공통의 소프트웨어 프레임워크(Common Software Framework)로 개발 단계나 실행 단계에서 사용될 수 있다. 응용 개발자는 역할 기반 조직 모델을 사용해서 자발적 상호연동에 필요한 최소한의 정보만을 기술하고 비즈니스 로직에 집중할 수 있다. 자발적 상호연동과 관련된 부분은 공통 프레임워크를 사용함으로써 견고한 응용 소프트웨어의 개발을 돕고 개발자의 부담과 개발 비용을 줄여준다.

본 논문의 내용은 다음과 같이 구성된다. 다음 제 2장에서는 기존 연구와 유비쿼터스 환경에서의 자발적 상호연동에 대해 논의한다. 또한, ROAD를 중심으로 역할 모델에 대해 소개하고 유비쿼터스 응용과의 연관성을 살펴본다. 제 3장에서는 자발적인 상호연동을 지원하기 위해서 능동 및 수동 역할 개념을 포함하는 확장된 역할 기반 조직 모델에 대해 기술한다. 제 4장에서는 구현 방안 및 사례 연구에 대해서 기술한다. 제 5장에서는 기존 연구들과 비교하고 논의 한다. 제 6장에서는 결론 및 향후 연구에 대해 기술한다.

## 2. 관련 연구

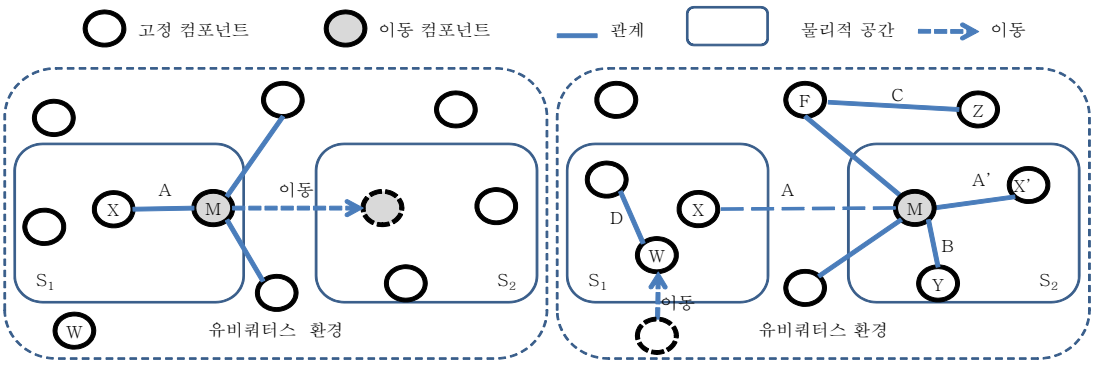
### 2.1 자발적 상호 연동

유비쿼터스 환경의 컴포넌트들은 자유롭게 이동하면서 물리적 공간에 포함되거나 벗어난다. 이런 컴포넌트의 이동성은 상황 변화를 초래하고 응용은 변화된 상황에 적응하기 위해 자신의 구조와 행위를 조정해야 한다. 응용의 구조를 조정하는 것은 참여중인 컴포넌트를 새로운 컴포넌트로 교체하고 이들 사이의 관계를 재설정하는 것을 의미한다. 관계는 컴포넌트들 사이의 논리적 연결을 의미하며 컴포넌트들 사이에 관계가 설정되면 서로 메시지를 교환할 수 있다. 자발적 상호연동은 이런 관계가 동적으로 설정되고 필요한 상호동작(interactions)이 인간의 개입이나 프로그램의 중단 없이 수행될 수 있는 것을 의미한다. 컴포넌트가 새로운 환경에 포함되어 다른 컴포넌트와 자발적인 상호연동을 하려면 다음 세 가지 작업들이 가능해야 한다[4]. 첫째, 컴포넌트는 자신이 속한 환경에서의 네트워크 주소가 필요하기 때문에 자율적으로 네트워크 환경에 참여해야 한다. IEEE 802.11과 같은 프로토콜을 사용해서 네트워크에 참여하고 네트워크 주소를 획득한다. 둘째, 서비스 발견 시스템을 이용해서 컴포넌트의 요구사항에 부합하는 컴포넌트를 발견해야 한다. 컴포넌트를 찾는 과정에서 물리적 공간(위치)이 결정 요인이 될 수 있다. 셋째, 상호 연동하는 컴포넌트들은 공통의 상호동작 모델(예 : event system, tuple-based system)을 따라야 한다.

컴포넌트의 이동성과 자발적 상호연동은 밀접한 관련을 가진다. 이미 이동성에 기인한 자발적 상호연동에 대한 연구가 네트워크 수준(network-level)에서 수행되었다. 자발적 상호연동에 영향을 주는 이동은 터미널 이동(terminal mobility), 세션 이동(session mobility), 퍼스널 이동(personal mobility), 그리고 서비스 이동(service mobility)으로 구분된다[12]. 기존 연구들은 사용자나 사용자가 소유하거나 사용 중인 장치의 이동에 초점을 두고 있다. 장치의 이동은 IP 서브넷(subnet)이 변경되는 것을 의미한다. 사용자나 단말기의 이동과는 무관하게 일관되고 지속적인 서비스를 사용자에게 제공하는 것이다[10].

### 2.2 유비쿼터스 응용에서 자발적 상호연동

유비쿼터스 응용에서 컴포넌트의 이동은 상황 변




(a) 컴포넌트들이 이동하기 전의 관계 (b) 컴포넌트들의 이동에 의한 관계 변화

[그림 1] 유비쿼터스 환경에서의 자발적 상호 연동

화를 초래하고, 변화된 상황에 적응하기 위해 응용의 구조와 행위가 변경되어야 한다. [그림 2]에서 컴포넌트 M이 물리적 공간 $S_1$에서 $S_2$로 이동함으로써 컴포넌트 X가 새로운 컴포넌트 X'로 교체되는 것을 볼 수 있다. 컴포넌트 X는 물리적 공간 $S_2$에서 필요하지 않다. 또한, 컴포넌트 M의 이동에 의한 상황 변화로 새로운 컴포넌트 Y가 응용에 참여할 수 있다. 이와 같이 상황에 따라 응용을 구성하는 컴포넌트들과 이들 사이의 관계가 동적으로 변화한다. 이는 컴포넌트들 사이의 연결성의 문제보다는 상황 변화에 의해 필요한 컴포넌트에 대한 요구사항의 문제와 밀접한 관련을 가진다. 다음 응용 사례들은 유비쿼터스 환경에서 자발적 상호연동이 필요한 상황을 묘사한다.

사례 1) 컴포넌트 이동에 의한 협업 파트너의 변경

"김대리는 스마트폰에 저장된 음악 파일(mp3 또는 wav 파일)을 사무실에 있는 오디오 시스템의 스피커를 통해 감상한다. 김대리는 회의 약속이 있어서 사무실을 나와 주차장으로 이동한다. 이동하는 동안에 불루투스 이어폰으로 음악을 감상한다. 주차장에 도착하고 자신의 차에 탑승한다. 차를 타고 약속장소로 이동하는 동안 차량 스피커를 통해 계속 음악을 감상한다."

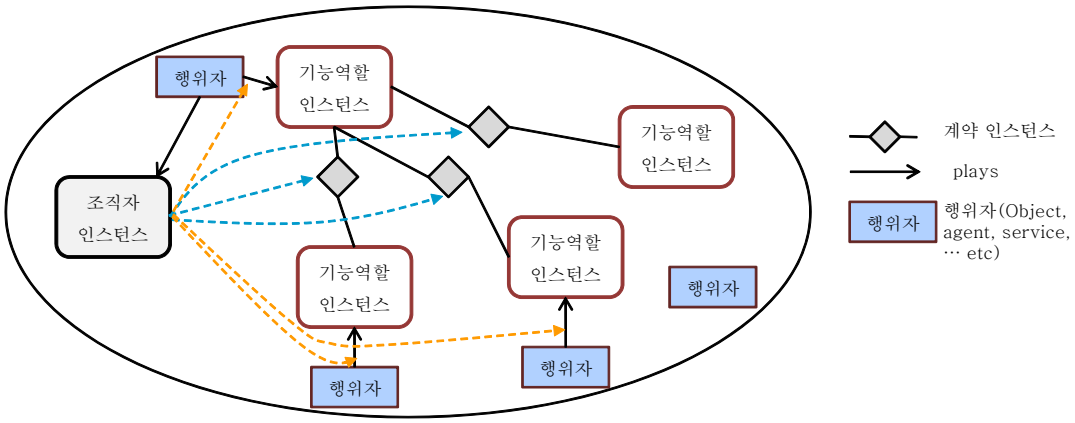
김대리가 사무실에서 나오면 스마트폰과 사무실의 스피커와의 협업 관계는 더 이상 유효하지 않다.

[그림 2]의 A 관계가 A' 관계로 변경되면서 X 컴포넌트가 X' 컴포넌트로 교체되는 경우이다. 이와 같이 컴포넌트의 이동에 의해 컴포넌트들 사이의 협업 관계가 유효하지 않게 되므로 새로운 협업 관계로 교체해야 한다. 이는 협업에 참여하는 컴포넌트의 교체를 의미한다. 스피커의 역할을 수행할 적합한 컴포넌트를 발견해서 필요한 작업을 수행한다.

사례 2) 컴포넌트의 이동에 의해 새로운 협업 관계의 생성

"U 대학의 김교수는 자신의 사무실에서 업무를 하다가 강의를 위해서 101 강의실로 이동한다. 김교수가 교수가 101 강의실에 입장하게 되면 스마트폰에 동작하는 교수 에이전트는 시간정보와 위치정보를 이용해서 강의 상황을 인지한다. 교수 에이전트는 강의실에 위치한 빔프로젝터를 통해 강의자료를 스크린에 보여준다."

이는 컴포넌트의 이동에 의해 새로운 협업 관계가 요구되는 상황이다. [그림 1]에서 M이 이동함으로써 새로운 협업 관계인 B와 C가 생성되고 새로운 컴포넌트 Y와 Z가 참여하는 경우이다. 협업 관계 C는 M의 이동에 의해 컴포넌트 F가 영향을 받은 경우이다. 컴포넌트의 이동에 따른 상황 변화




[그림 2] ROAD의 조직 구조

는 전체 응용의 구조와 행위에 영향을 줄 수 있다.

사례 3) 물리적 공간에 컴포넌트가 사라지거나 나타나는 경우

"A 회사의 직원인 김대리는 업무 회의를 위해서 B회사를 방문하였다. 회의실에는 랩탑 컴퓨터와 빔 프로젝터가 비치되어 있다. 회의에서 김대리는 회의 안건에 대한 발표를 한다. 회의실에 B회사의 직원들이 참석하고 서로 전자 명함을 교환한다. 김대리는 스마트폰을 이용해서 회의실에 비치된 빔 프로젝터를 이용해서 프리젠테이션을 수행한다. 회의 도중에 B회사의 직원들이 추가적으로 참석하게 되고 서로 전자 명함을 교환한다."

회의 도중에 새로운 인원이 추가적으로 참여하게 되면 발표자와 참석자사이의 새로운 관계가 설정될 수 있다.

유비쿼터스 응용에서 장치, 즉 컴포넌트의 이동은 IP 서브넷(subnet)의 이동이 아니라 물리적 공간의 이동을 의미한다. 이런 물리적 공간의 이동에 의해 IP 서브넷이 변경 될 수도 있지만 본 논문에서는 네트워크 수준이 아니라 응용 수준에서 컴포넌트의 이동성을 고려한다. [그림 1]에서 보듯이 물리적 공간의 변화는 기존 컴포넌트의 교체나 새로운 컴포넌트와의 협업을 요구할 수 있다. 본 논문은 장치가 이동하는 물리적 이동(physical mobility)에 초점을 둔다. 코드 혹은 프로그램의 이동과 같은 논리적 이동은 고려하지 않는다.

## 2.3 역할 모델

역할 개념은 시스템의 구성요소, 관계, 그리고 이들 사이의 협업을 표현하기에 유용한 도구이다. 유비쿼터스 소프트웨어 시스템은 환경에 내재되어 있는 컴포넌트들의 참여와 협업이 필요하다. 컴포넌트들은 항상 네트워크에 연결되어 있으며 자유롭게 이동하면서 다른 컴포넌트들과 통신이 가능하다. 유비쿼터스 응용은 컴포넌트들의 동적인 참여를 요구하기 때문에 응용을 구성하는 컴포넌트들의 관계는 비결정적(non-deterministic)이다[3, 4]. 역할 모델은 유비쿼터스 응용에 참여하는 컴포넌트들, 이들 사이의 관계와 상호 연동을 추상화하기에 적합한 도구이다.

특히, ROAD는 동적으로 변하는 환경에 대한 시스템의 적응성(adaptability)을 지원하기 위해 역할 기반의 조직 모델을 제안하고 있다[1, 8]. 역할 기반 조직은 에이전트 시스템을 추상화하기 위한 도구로서 연구되어 왔다[6]. [그림 2]는 ROAD의 역할 모델에 기반해서 구현된 시스템의 구조를 보여준다. ROAD에서 조직은 역할들의 네트워크(Role-

network)이고 역할과 역할은 계약(contract)에 의해 묶여진다. 계약(contract)은 참여자들의 의무(obligations)를 기술한 것이다. 역할-네트워크, 조직은 행위자들 사이를 조율 하는 작업을 수행할 수 있다. 즉, 역할-네트워크는 변화된 환경에 적응하는 조율자(coordinator) 역할을 수행한다. ROAD에서 소프트웨어 시스템은 제어 프로세스와 기능 프로세스로 구분되고 각각 역할-네트워크와 행위자에 대응된다.

일반적으로 역할의 개념은 설계 단계에서 응용에 참여하는 행위자가 특정 역할을 수행하기 위해 구현되어야 할 요구사항을 정의하기 위해 사용되었다. [그림 2]에서 보듯이 ROAD에서 역할은 개발단계에서 하나의 구현 단위로 간주되고, 실행시간에 역할 인스턴스로 존재한다. ROAD는 역할 모델의 구현 방법으로 소프트웨어 프레임워크를 적용하고 있다. ROAD에서 역할은 조직자(organizer)와 기능역할(functional role)이 있으며, 하나의 조직에는 하나의 조직자(organizer)가 존재한다. 조직이 생성되기 위해서는 먼저 조직자가 생성되고 생성된 조직자는 조직을 구성하는 역할과 계약을 생성하고 이들 사이의 관계를 설정하는 작업을 수행한다. 역할을 수행하는 행위자들이 참여하고 나면 이들 사이의 상호동작이 진행된다. 이때, 상호연동을 위해 요구되는 메시지들은 역할 인스턴스들의 중계를 통해서 전달된다. ROAD에서 역할 인스턴스의 주요 작업은 메시지 라우팅(중계)이고 계약 인스턴스는 역할들 사이에 교환되는 메시지의 순서나 조건을 검사한다.

그러나 기존의 역할 기반의 모델들은 자발적 상호동작을 지원하는 유비쿼터스 시스템의 설계와 개발에 적용하기에는 부적합하다. 주요한 이유는 다음과 같다.

자발적 상호연동과 밀접한 관련이 있는 컴포넌트들의 이동성을 고려하지 않기 때문에 자발적 상호연동을 지원하는 것은 여전히 개발자의 몫이다.

기존 모델은 컴포넌트들의 이질성을 고려하지 않고 있다. 대부분의 모델들은 컴포넌트들이 인간, 에이전트, 혹은 객체와 같은 동질적인 개체로 간주한다. 이질성은 곧 이동성을 가지는 컴포넌트, 제한된 공간에서 동작하는 컴포넌트, 그리고 컴퓨팅 파워가 매우 낮은 컴포넌트들에 대한 고려가 필요하다[7].

기존의 역할 모델은 컴포넌트의 기능적 요구사항에 초점을 두고 있다. 비기능적 요구사항에 대한 고려가 부족하다. 유비쿼터스 환경에서는 컴포넌트의 비기능적 요구사항이 설계단계에서 실행단계에 걸쳐 중요한 요소로 다루어져야 한다.

ROAD 모델은 유비쿼터스 응용을 정의하기에 좋은 도구지만, 자발적 상호연동과 같은 유비쿼터스 응용의 특성에 대한 고려가 미흡하다. 따라서 언급한 특성들을 지원하는 것은 전부 개발자의 몫이 되어 부담을 가중시킨다.

## 3. 자발적 상호연동의 지원

유비쿼터스 응용의 핵심은 구성요소들의 협업이며 응용의 모든 행위는 컴포넌트들과 이들 사이의 상호동작으로 기술된다. 응용을 구성하는 컴포넌트들은 역할에 의해 추상화되고 실행시간에 역할의 요구사항을 만족하는 실제 컴포넌트와 바인딩된다. 환경에 내재된 컴포넌트들은 응용에 정의된 역할을 수행할 수 있는 잠재적인 행위자(player)들이다. 역할 모델의 관점에서 역할에 바인딩되는 컴포넌트를 행위자라 칭한다. 본 논문은 조직의 관점(organizational viewpoint)에서 유비쿼터스 응용을 바라보고 역할을 통해 추상화한다. 여기서는 자발적 상호연동을 지원하기 위한 확장된 역할 기반 조직 모델에 대해서 기술한다.

### 3.1 요구 사항과 가정

먼저, 자발적 상호 연동을 지원하는 유비쿼터스 응용의 요구사항을 식별하고 이를 모델에 반영해야 한다. 다음은 자발적 상호연동에 대한 요구사항이다.

- 유비쿼터스 응용과 여기에 참여하는 실제 행위

자는 서로 무관하게 개발되기 때문에 응용의 구성요소를 정의하기 위한 추상 컴포넌트 모델이 필요하다.

- 응용을 구성하는 추상 컴포넌트, 즉 역할에 적합한 실제 행위자들을 동적으로 발견할 수 있다.
- 행위자들은 동적으로 응용에 참여 및 탈퇴가 가능하므로 응용은 실행 시간에 자신의 구조를 조정할 수 있다.
- 응용의 모든 행위는 구성 요소들의 상호동작으로 기술되고 응용은 자신의 행위를 상황에 맞게 제어할 수 있다.

본 논문은 자발적 상호 연동에 초점을 두기 위해서 다음 사항을 가정한다.

- 행위자들은 자신이 속한 환경에 적합한 통신 프로토콜을 사용해서 항상 네트워크에 연결할 수 있고 주소(network address)를 획득할 수 있다고 가정한다.
- 물리적 공간에는 공간 관리자(space manager)가 존재하고 행위자들은 공간 관리자에게 자신의 출입을 알린다. 행위자들은 공간 관리자를 통해 다른 행위자들의 정보나 출입 정보를 획득할 수 있다. 공간 관리자는 well-known 서비스로 제공된다.

## 3.2 역할 모델의 확장

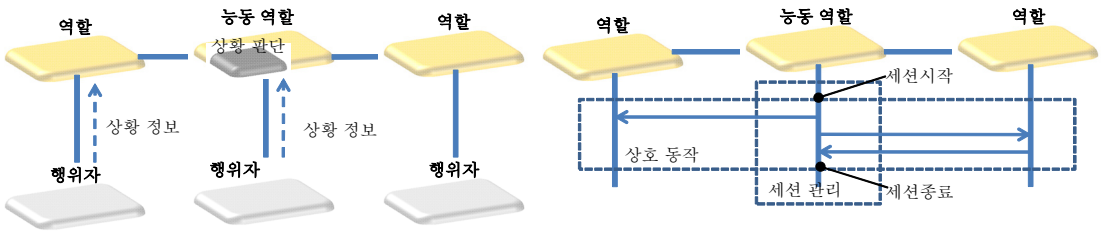
역할은 기본적으로 조직에 참여하게 되는 잠재적인 행위자에 대한 요구사항을 정의한다. 이런 요구사항은 기능과 비기능 요구 조건으로 구분된다. 기능 요구 조건은 메시지 인터페이스(message interface)로 표현된다. 비기능적 요구 조건은 행위자의 상태나 주변 환경 상태를 표현한다. 요구사항은 주로 상호동작에 필요한 추가 모듈의 구현과 실행시간 역할-행위자의 바인딩에 사용된다. 역할의 요구사항을 기술하는 것은 조직에 참여하게 되는 잠재적 행위자들의 특성을 반영해야 한다.

### 3.2.1 능동 및 수동 역할

본 논문은 유비쿼터스 환경에 내재된 잠재적인 행위자들의 특성에 주목하고 이를 모델에 반영하였다. 우선적으로 고려해야 하는 특성은 행위자의 이동성이다. 스마트 폰, 랩톱 컴퓨터, 그리고 자동차와 같은 행위자들은 자유롭게 이동가능하며 사용자와 밀접한 관련을 가진다. 이런 이동 행위자들은 자신이 속한 물리적 공간에 존재하는 응용의 다른 행위자들과 자유롭게 상호 연동할 수 있다. 이동 행위자들은 자신이 속한 공간에 존재하는 고정 행위자(TV, 음향 장치, 프린터, 조명 장치 등)들의 서비스를 요청할 수 있다. 이와 같이 이동 행위자들은 자신이 속한 공간에 내재된 서비스를 제어하는 능동적인 작업을 수행할 수 있다.

두 번째는 행위자들이 서로 다른 수준의 컴퓨팅 파워와 기능을 가지는 것이다. 이는 행위자들이 할 수 있는 일들이 차별화될 수 있음을 의미한다. 웹 서비스나 응용 서버와 같은 컴퓨팅 파워가 높은 행위자들과 조명이나 음향 기기와 같이 낮은 컴퓨팅 파워를 가지고 제한적인 기능을 수행하는 행위자들이 존재한다. 특히, 스마트폰-단말기-은 사용자의 대리자로 간주되며 소유자에 대한 식별 정보를 제공할 수 있다. 단말기를 소유한 사용자는 하나의 컴포넌트로 응용에 참여해서 역할을 수행하는 행위자가 될 수 있다. 하드웨어 기술의 발달과 함께 이동 단말기나 스마트폰의 성능 향상을 통해 사용자가 할 수 있는 작업이 다양해졌다. 또한, 사용자 단말기에 장착된 위치 인식 시스템(GPS)이나 각종 센서를 통해 다양한 상황 정보를 획득하고 이를 바탕으로 상황을 판단할 수 있다.

위와 같은 사항을 반영하기 위해 능동 및 수동 역할 개념을 도입하였다. 유비쿼터스 응용은 자발적인 상호연동이 필요한 상황을 인지하고 응용의 구조와 행위를 조정해야 한다. [그림 3]에서 보듯이 이런 기능은 역할 인스턴스에 추가되어 행위자들 사이의 자발적 상호연동을 가능하게 한다. 자발적 상호연동을 위해서는 상황 인지, 조직 구조의 조정, 그리고 상호동작 제어가 필요하다. 상황 인지는 자발적 상호연동이 필요한 상황을 인지하기 위해 필요한 상황 정보를 수집하고 판단하는

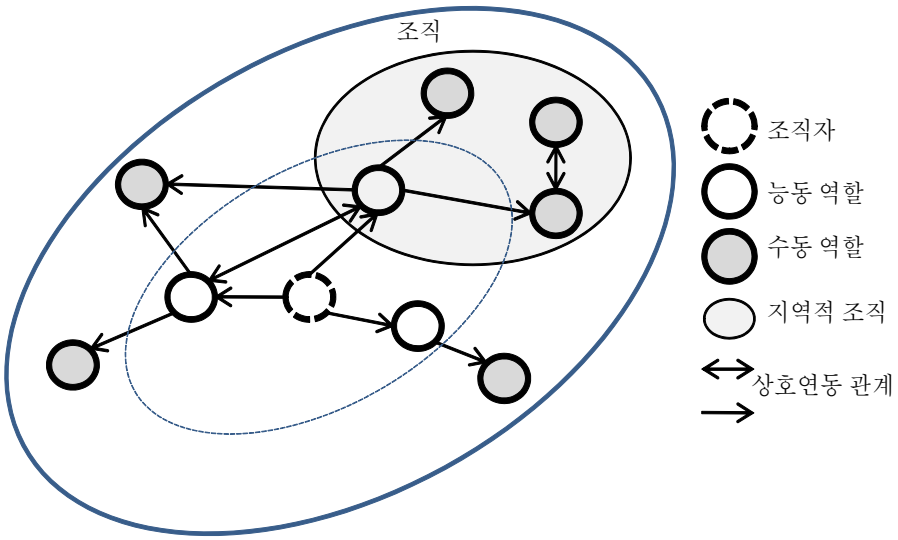



[그림 3] 능동 역할에 의한 상황 인지와 상호동작의 세션 관리

작업이다. 자발적 상호연동과 관련한 상황을 인지하면 조직 구조를 조정하기 위해 역할 관계를 생성하거나 파괴하고 행위자의 교체나 제거 작업이 필요하다. 조직 구조가 조정되면 상황에 맞는 상호동작이 수행되어야 한다. 이런 일련의 과정들이 행위자들 사이에서 자발적으로 수행되어야 한다.

그러나 응용에 참여하는 행위자들은 특정 응용과 무관하게 개발되기 때문에 이런 요구사항을 만족시켜주기 어렵기 때문에 행위자들을 도와주는 추가적인 기능을 역할의 인스턴스가 제공한다. 개발 단계에서 정의된 역할에 대응하는 실행시간의 단위를 역할 인스턴스라고 부른다. 따라서 수동/능동 역할은 각각 수동/능동 역할 인스턴스로 구체화된다.

이와 같이 능동 역할은 상황 정보를 수집해서 판단하고 역할들 사이의 상호 동작을 관리하는 주도적인 작업을 포함하고 수동역할은 능동 역할의 지시에 따르는 수동적인 작업을 포함하게 된다. 이동 행위자들은 물리적 공간을 이동하면서 공간에 내재된 서비스를 이용하거나 제공받는다. 이동 행위자는 자신이 속한 공간에 내재된 서비스, 즉 고정 행위자들을 발견하고 응용에 참여시키는 작업을 수행할 수 있다. 따라서 이동 행위자는 능동 역할을 수행하고 환경에 내재된 고정 행위자들은 서비스를 제공하는 수동 역할을 수행한다. 이동하지는 않지만 컴퓨팅 파워가 높은 행위자들은 능동 역할이 부여된다. 웹 서비스나 검색 서비스와 같은 컴퓨팅 파워가 높은 장치에서 동작하는 컴포넌




[그림 4] 역할 기반의 조직 구조

트들은 능동 역할을 수행할 수 있다. 본 논문에서는 능동 또는 수동 역할을 수행하는 행위자는 능동 또는 수동 행위자라고 칭한다.

### 3.2.2 역할 관계와 조직 구조

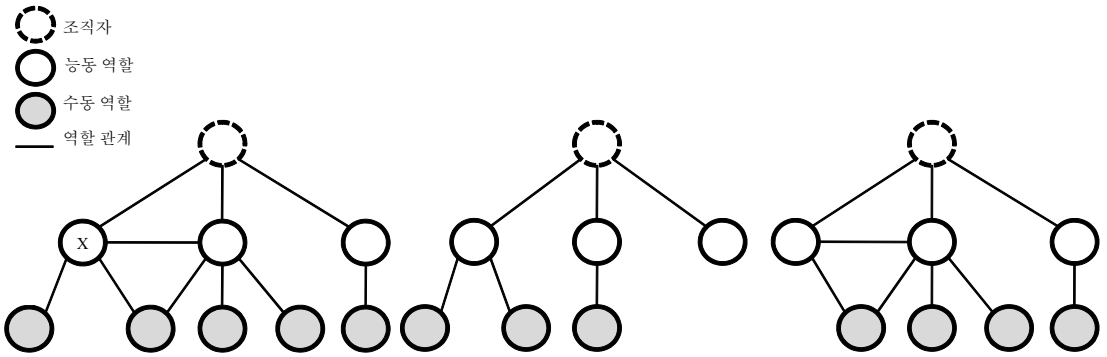
유비쿼터스 응용은 소프트웨어 조직(software organization)으로 간주되고 조직의 구조는 역할-네트워크로 추상화된다. [그림 4]에서 보듯이 하나의 응용은 조직자, 능동 역할, 그리고 수동 역할들에 의한 계층적인 구조를 가진다. 조직 구조는 역할들 사이의 관계로 구성되며 이런 역할 관계는 조직자-능동역할, 능동역할-수동역할 사이의 수직적 관계와 능동역할-능동역할, 수동역할-수동역할 사이의 수평적 관계로 구분된다. 하나의 조직에는 하나의 조직자가 존재하며 조직자는 능동역할의 특별한 형태로 볼 수 있다. 조직자는 조직의 조직의 생성 및 소멸을 관리하고 상황 정보를 수집하고 사전에 정의된 상황 조건을 검사한다. 조직자 상황에 따른 조직의 구조를 조정하는 작업을 수행해야 하고 계층 구조에서 자신의 하위 역할과의 관계를 관리한다. 실행 시간에 조직자는 최상위에 위치하고 조직의 모든 능동 역할 인스턴스들과 관계를 맺고 이들을 제어한다. [그림 4]의 전체 조직 구조는 [그림 5](a)와 같이 역할들 사이에 가능한 모든 관계를 표현한 계층 구조로 표현된다.

응용 개발자는 개별 상황 조건에 대해 역할들과 이들 사이의 관계를 정의할 수 있다. 예를 들어 응용이 직면하는 두 개의 상황(A, B)가 있다고 가정한다. [그림 5](b)와 [그림 5](c)는 이런 상황 A와 B에 대한 조직 구조를 보여준다. 따라서, 조직자는 현재의 상황에 적합한 조직 구조를 유지해야 할 책임이 있다. [그림 6]에서 보듯이 X 역할은 상황에 따라 인접한 다른 역할과의 관계가 변화된다. 실행 시간에 능동 역할 인스턴스들은 자신과 인접한 다른 능동 또는 수동 역할 인스턴스들과의 관계를 조정하게 된다. 수동 역할 인스턴스들은 상위의 능동 역할 인스턴스와의 지시에 따라 필요한 작업을 수행하게 된다.

능동 역할 인스턴스는 자신의 능동 행위자와 상호동작하는 수동 행위자를 자신이 속한 물리적 공간에서 찾아서 조직에 참여시킬 수 있다. 즉, 능동 역할 인스턴스는 특정 물리적 공간에서의 역할들 사이의 관계를 조정하는 조직자 역할을 수행할 수 있다. 능동 행위자가 자유롭게 이동하면서 특정 물리적 공간에 포함되어 해당 공간에 내재된 다른 행위자들과 상호 동작하는 경우에 이를 지역적 협업이라 한다.

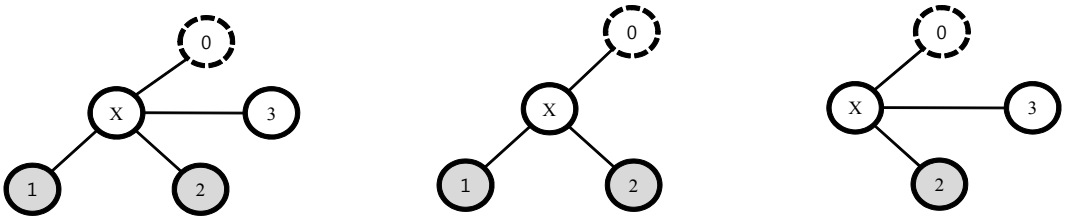
### 3.2.3 역할과 역할 인스턴스

실행 시간에 역할 인스턴스는 두 가지 유형의 작업을 수행한다. 하나는 조직 구조와 행위를 제어하기 위해 필요한 관리 기능과 응용의 목표를




(a) 조직 전체 구조 (b) 상황 A (c) 상황 B

[그림 5] 상황에 따른 조직 구조의 변화




(a) X의 전체 역할 관계들 (b) 상황 A에서 X의 역할 관계 (c) 상황 B에서 X의 역할 관계

[그림 6] 상황에 따른 개별 역할과 주변 역할들의 관계

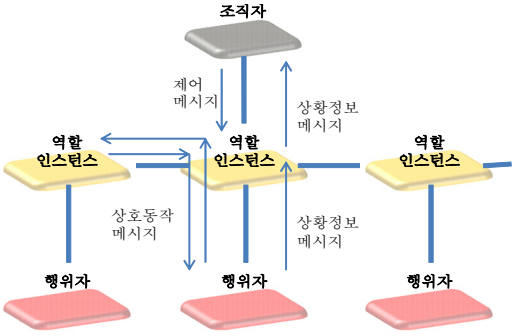
달성하기 위한 행위자들 사이의 상호동작을 지원하는 협업 기능이다. 개발자는 설계단계에서 응용의 상황에 따라 필요한 컴포넌트들을 식별하고 역할들과 이들 사이의 관계를 결정함으로써 조직 구조를 정의한다. 응용 개발자가 자발적 상호 연동을 위해 필요한 작업은 상황에 따른 조직 구조를 정의하는 것뿐이다. 조직 구조가 결정되고 나면 응용의 목표를 달성하기 위해 필요한 구성요소들 사이의 협업을 역할들 사이의 상호동작으로 기술한다. 역할과 조직 구조 정보는 역할 인스턴스의 관리 기능으로 구현되고 협업을 위한 상호동작은 역할 인스턴스의 협업 기능으로 구현된다. ROAD는 역할 인스턴스 사이의 상호동작을 제어하기 위해 역할들 사이에 계약 인스턴스를 두고 있다[4]. 본 논문은 상호동작에 필요한 부분을 각각의 역할에 분산하고 계약 인스턴스를 제거하였다. 이는 개별 역할 인스턴스들을 물리적으로 분산 배치할 수 있는 가능성을 제공한다.

### 3.3 자발적인 상호연동을 위한 공통 프로세스

자발적 상호연동을 위해 역할 인스턴스들과 행위자들 사이에서 수행되는 상호동작은 정형화될 수 있다. 정형화된 부분은 개발을 위한 소프트웨어 프레임워크나 응용의 실행 환경을 위해 사용될 수 있다.

#### 3.3.1 자발적 상호 연동의 공통 프로세스

우선적으로 필요한 작업은 자발적 상호연동이 필요한 상황을 감지하는 것이다. [그림 7]에서 보듯이 역할 인스턴스는 행위자의 상태를 감시하고 필요한 경우 상태 정보를 조직자에게 전달한다. 특히, 능동역할 인스턴스는 행위자의 이동을 감지하고 자발적 상호동작이 필요한지 판단해야 한다. 역할 인스턴스는 역할 속성에 정의된 행위자의 상태를 감시한다.




[그림 7] 구성요소간 메시지 교환

다음은 상황 정보를 수집하고 판단하는 과정을 기술한 것이다.

1) 모든 역할 인스턴스들은 바인딩 된 행위자의 상태가 변경되면 상위의 역할 인스턴스에(수동 → 능동, 능동 → 조직자) 이를 전파한다.
2) 조직자는 하위에서 수집된 상황 정보를 이용해서 상황을 판단한다.
3) 조직자는 새로운 협업 상황을 위해 조직 구조를 조정한다.
   a) 필요한 경우 조직자는 새로운 역할 인스턴스를 생성한다.
   b) 새로운 역할 인스턴스에 대응하는 행위자를

찾아서 바인딩한다.

4) 조직의 구조 조정이 끝나면 새로운 협업 상황을 하위의 능동 역할 인스턴스에 전파한다.
5) 능동 역할 인스턴스는 새로운 상황에 필요한 상호동작을 시작한다.

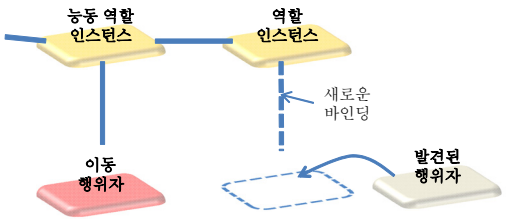
위에서 언급한 행위자를 발견하는 작업은 조직자나 능동 행위자에 의해 수행될 수 있다.

1) 조직자는 발견 서비스를 이용해서 행위자를 찾는다.
2) 능동 역할 인스턴스는 자신이 속한 공간에서 수동 행위자를 찾는다.
3) 새롭게 생성된 역할 인스턴스는 발견된 행위자의 네트워크 주소를 조직자나 능동 역할 인스턴스로 부터 획득한다.
4) 역할 인스턴스는 자신의 행위자와 연결하고 이를 자신의 상위 역할 인스턴스에게 알린다.

위의 내용은 모든 유형의 자발적 상호연동에 공통적으로 수행된다.

### 3.3.2 자발적 상호연동의 유형

자발적 상호연동은 [그림 8]과 같이 기존의 관계는 변경하지 않고 행위자만 교체하는 경우와 새로운 관계를 생성하고 행위자를 연결하는 두 가지 경우로 구분할 수 있다.




[그림 8] 행위자 교체

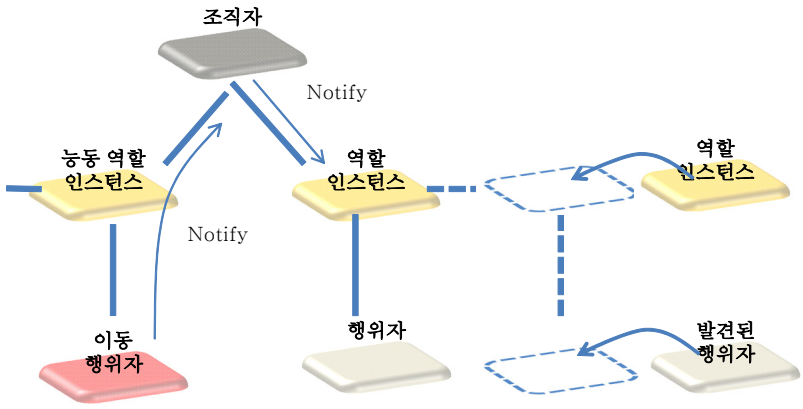
행위자의 교체는 능동 행위자의 이동에 의해 새로운 공간에 존재하는 수동 행위자가 필요한 경우이다.

1) 먼저, 능동 역할 인스턴스는 행위자의 공간 이동을 인지한다.
2) 기존 행위자를 교체해야 하는 상황인진 판단한다.
   a) 기존의 상호 동작의 세션을 종료하고 필요한 작업을 수행한다.
3) 교체 가능한 행위자를 해당 공간에서 찾는다.
   a) 교체 가능한 행위자가 없는 경우는 조직자에게 알린다.
4) 역할 인스턴스에게 발견된 행위자에 대한 정보를 전송한다.
5) 생성된 역할 인스턴스와 발견된 행위자가 바인딩된다.
6) 능동 역할 인스턴스는 상호동작을 시작한다.

새로운 관계의 설정은 응용의 관심과 관련된 상황에 직면한 경우에 발생한다. [그림 9]에서와 같이 조직자는 역할 인스턴스로부터 개별 행위자들의 상황 정보를 수집해서 응용이 직면한 상황을 판단한다. 새로운 관계를 설정하는 작업은 주체가 능동 역할 인스턴스인 경우와 조직자인 경우로 구분할 수 있다. 필요한 역할과 행위자가 이미 참여하고 있는 기존의 행위자의 물리적 공간과 관련 있는 경우에는 해당 능동 역할 인스턴스에게 이를 위임한다. 행위자가 물리적 공간과 관련 없는 경우에는 조직자가 새로운 관계를 설정한다.

다음은 이 과정을 기술한 것이다.

1) 새로운 역할 인스턴스를 생성한다.
2) 기존의 역할 인스턴스와 생성된 역할 인스턴스와의 관계를 설정한다.
3) 역할을 수행할 행위자를 찾는다.
   a) 조직자가 찾거나 기존의 역할 인스턴스에게 위임할 수 있다.
4) 발견된 행위자의 정보를 역할 인스턴스에게 전송한다.
5) 생성된 역할 인스턴스와 발견된 행위자가 바인딩된다.
   a) 능동 인스턴스는 작업이 완료되었음을 조직자에게 알린다.



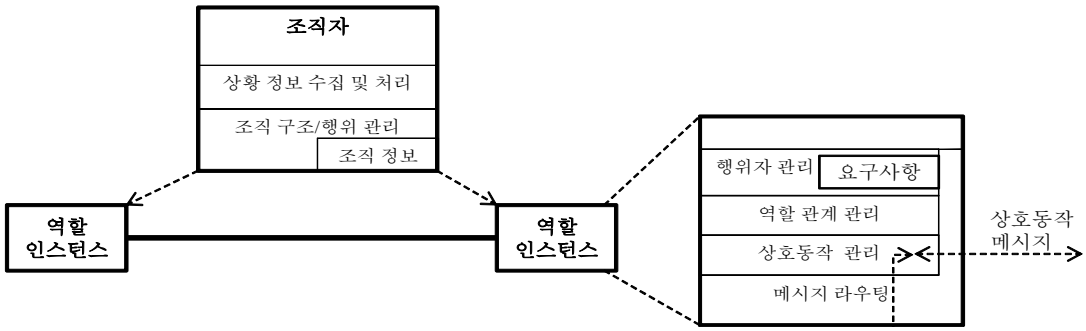

[그림 9] 상황 변화에 따른 새로운 관계 설정

6) 조직자는 새로운 상황과 협업에 필요한 상호동작을 시작할 것을 능동 역할 인스턴스들에게 알린다.
7) 역할 인스턴스들은 상황에 필요한 상호동작을 시작한다.

## 4. 구현 방안 및 사례 연구

응용 개발자는 설계단계에서 목표 응용에 참여하게 되는 컴포넌트들을 식별하고 이에 대응하는 역할을 결정한다. 그리고 협업이 요구되는 상황에 대한 조건들과 역할들의 관계에 대한 정보를 정의한다. 이 정보를 이용해서 자발적 상호연동에 필요한 기능을 역할 모듈에 포함시킬 수 있다. 조직 구조 정보나 역할들 사이의 상호동작을 명세화 할 수 있으면 개발 과정을 자동화 시킬 수 있다. 소프트웨어 프레임워크에서 제공되는 템플릿 클래스들을 결합해서 역할을 구현할 수 있다. 역할과 조직이 명세화 되면 이 과정을 자동화 할 수 있다. 따라서, 응용 개발자는 역할과 역할 관계에 대한 최소한의 정보를 기술함으로써 자발적 상호연동이 가능한 유비쿼터스 응용을 효과적으로 작성할 수 있으며 개발자는 비즈니스 로직에 집중할 수 있다.

[그림 10]은 조직자와 역할 인스턴스의 구조를 보여준다. 조직자는 응용의 시작과 함께 생성되어 응용이 종료할 때 소멸된다. 조직자의 상황 정보 수집 및 처리는 사전에 정의된 상황 조건에 필요한 정보를 하위의 역할 인스턴스로부터 수집해서 검사하는 작업을 수행한다. 조직 구조/행위 관리는 상황에 따른 조직 구조의 변경 및 유지를 실행




[그림 10] 조직자 및 역할 인스턴스의 구조

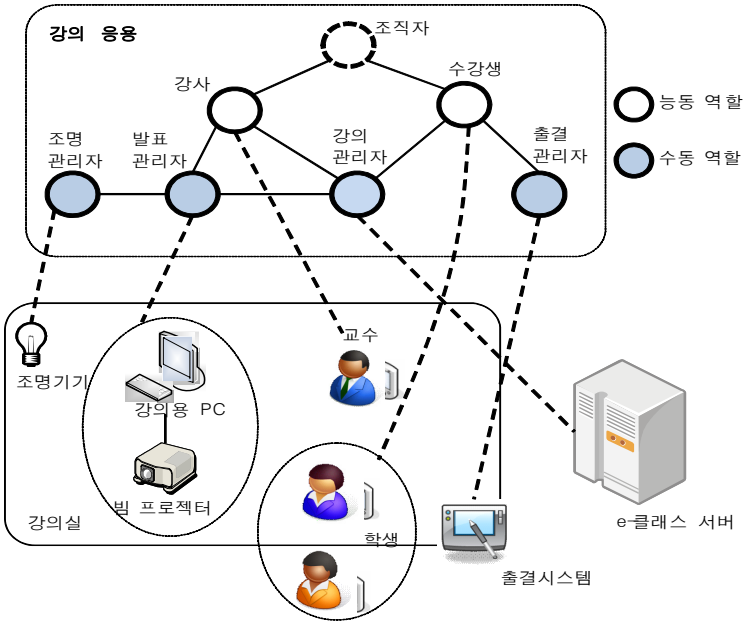
하고 역할 인스턴스들의 동작을 제어하는 작업을 수행한다. 역할 인스턴스는 행위자 관리, 역할 관계 관리, 그리고 메시지 라우팅 기능을 포함한다. 행위자들 사이의 메시지 전달은 행위자와 연결되어 있는 역할 인스턴스들이 중계한다. 개발 단계에서 실제 행위자의 네트워크 주소를 알 수 없기 때문에 역할들 사이의 상호동작 메시지는 역할의 이름을 사용해서 기술한다. 메시지 라우팅 모듈은 다른 역할 인스턴스로부터 전달된 메시지를 행위자에게 전달하고 행위자가 생성한 메시지를 수신해서 관계를 맺고 있는 역할 인스턴스에 전달하는 작업을 수행한다. 역할 관계 관리는 상황에 따라 인접한 역할들과의 관계 정보, 협업에 필요한 상호동작하는 메시지 정보, 상호동작에 필요한 세션을 관리하는 작업을 수행 한다. 행위자 상태 관리는 행위자의 상태 변경을 감시하고 변경된 상태 정보를 조직자에게 전파한다.

조직을 생성하고 실행하는 가장 단순한 방법은 조직자와 역할 인스턴스들을 동일한 머신에서 생성되어 동작하는 중앙 집중형이다. 그러나 유비쿼터스 환경에서 행위자들은 광범위한 지역에 분산되어 있기 때문에 역할 인스턴스들을 하나의 머신에서 실행시키는 것은 비효율적이므로 역할 인스턴스들이 분산되어 실행되는 것이 효율적일 수 있다. 실행시간에 역할 인스턴스들이 배포되고 생성 및 실행되는 구체적인 방안은 자세히 논의하지 않는다.

## 4.1 강의 응용 사례

이 절에서는 제안된 모델을 강의 응용에 적용한 사례를 기술한다. 다음은 강의 상황에서의 시나리오를 기술 한 것이다.

“교수는 컴퓨터 프로그래밍 과목을 강의하기 위해 강의실 101호에 오전 10시 57분에 입장한다. 교수의 일정에는 강의실 101호에서 오전 11시부터 컴퓨터 프로그래밍 강의에 대한 정보가 포함되어 있다. 교수는 스마트폰을 소유하고 있으며 강의 응용을 위한 강사 역할의 인스턴스가 동작하고 있다. 강사 인스턴스는 강의 상황을 인지하고 해당 강의실의




[그림 11] 강의 용용의 조직 구조

프리젠테이션 관리자를 발견하고 관계를 설정한다. 강사 역할은 프리젠테이션 관리자에게 프리젠테이션 서비스를 요청한다. 프리젠테이션 관리자는 e-class 서버에 접속해서 파일을 다운로드 받는다. 프리젠테이션 관리자는 빔프로젝트를 이용해서 파일의 내용을 출력하고 강의실 조명의 밝기를 조절한다. 학생들의 스마트폰에는 수강생 역할이 동작하면서 수업 시간 시작하기 전/후에 강의실에 입장하게 되면 출결시스템에 출석을 등록하게 된다. 수업이 종료되면 출석 시스템은 e-class 서버에 출결 정보를 전송한다."

[그림 11]은 강의 시나리오에서 행위자로써 응용에 참여하는 구성요소들과 추상화된 역할 조직을 보여준다. 개발자는 [그림 11]에서 보듯이 강의 응용에 필요한 구성요소들을 역할로 추상화하고 이들 사이의 관계를 정의한다. 강의 응용은 기본 상황과 강의상황으로 구분할 수 있다. 개발자는 기본 상황과 강의 상황에 대한 조건을 추가적으로 기술한다. 강의 상황이 되면 [그림 11]와 같이 조명 관리자, 발표 관리자, 출결 관리자 역할들과 이에 대응하는 행위자들이 바인딩 되고 이들 사이의 관계가 설정된다. 이동성을 가지는 강사와 수강생은 능동 역할이고 나머지 구성요소들은 수동 역할이 된다. 여기서, 강의 관리자는 이동성을 가지지는 않지만 위치에 의존적이지 않기 때문에 능동 역할로 추상화 될 수도 있다.

## 5. 논의

유비쿼터스 응용에 참여하는 컴포넌트들은 특정 응용과는 무관하게 개발되기 때문에 특정 응용의 요구사항을 완전히 만족시키기 어렵다. 따라서 응용 개발자는 비즈니스 로직 뿐만 아니라 컴포넌트들 사이의 자발적 상호연동에 필요한 부분을 개발해야 한다. 그러나, 유비쿼터스 용용의 개발에 필요한 추상화 모델이나 개발 방안에 대한 연구가 부족한 현실이다. 본 논문은 유비쿼터스 응용의 추상화에 적합한 역할 기반 조직 모델을 제안한다. 역할은 구성요소에 대한 요구사항과 이들 사이의 협업, 즉 상호동작을 기술하기에 적합하다. 유비쿼터스 응용은 전통적인 응용 소프트웨어의 개발 및 실행 환경과는 다르다.

유비쿼터스 환경에서 행위자-컴퓨팅 장치-들은 자유롭게 이동할 수 있다. 특히, 응용에 참여하는 컴포넌트들의 이동은 상황 변화를 초래하고 이는 응용의 구조와 행위의 변경을 필요로 한다. 따라서, 응용 개발자는 개발단계에서 이런 특징들을 고려해야 한다. 특히, 유비쿼터스 응용과 컴포넌트들은 서로 무관하게 개발되기 때문에 응용의 추상화 모델이 필요하다. 구성요소들과 이들 사이의 협업을 추상화하기 위해 역할 개념이 널리 사용되고 있다. BRAIN[2] 이나 E-CARGO[13]와 같은 기존의 역할 모델들은 행위자 중심의 관점에서 역할을 보고 있다. 이는 유비쿼터스 응용과 같은 오픈 시스템에는 적합하지 않다. 유비쿼터스 응용은 변화하는 환경에 적응하기 위해 자신의 구조와 행위를 조정해야 한다[4].

ROAD를 비롯한 최근의 역할 모델들은 조직적 관점에서 역할을 바라본다. 조직의 구조와 행위는 역할 인스턴스들에 의해 조율된다. 행위자가 존재하지 않더라도 역할 인스턴스는 존재할 수 있다[3]. 조직에 포함된 하나의 역할을 수행하는 행위자가 없거나 불능이더라도 조직 전체에 미치는 영향을 최소화 할 수 있다. 이는 기존의 설계단계에서만 사용되던 기존의 역할 개념을 구현의 단위로 간주하고 있다. 또한 역할 인스턴스는 컴포넌트들 사이의 상호동작을 돕기 위한 추가적인 부분을 제공할 수 있다. 역할은 행위자의 기능적 요구사항을 주로 정의한다. 자발적 상호연동을 지원하기 위해서 행위자들의 비기능적 요구사항이 설계 단계에서 실행 단계에 걸쳐 적용되어야 한다. 자발적 상호연동은 응용에 참여하는 행위자들의 이동성(mobility)과 밀접한 관련을 가진다. ROAD를 비롯한 기존의 역할 모델은 행위자의 이동성을 고려하고 있지 않다. 유비쿼터스 응용의 잠재적 행

위자들에 대한 고찰을 통해서 능동 및 수동 역할 개념을 도입하였다. 응용의 구조와 행위를 조정하기 위해서는 구성 요소들 사이의 상호동작을 정형화 시킬 수 있다. 상호동작에 필요한 공통 프로세스를 분석하고 정의함으로써 응용의 개발을 용이하게 하고 자발적 상호연동을 지원하는 실행 환경을 구성할 수 있다.

## 6. 결 론

본 논문은 자발적 상호연동과 밀접한 관련을 가지는 컴포넌트의 이동성과 비결정적 특성을 반영한 확장된 역할 모델을 제안하고 있다. 이를 위해 기존의 역할 개념을 확장하여 능동 역할과 수동역할 개념을 도입하였다. 응용 개발자는 역할 모델을 사용해서 자발적 상호연동에 필요한 최소한의 정보만을 기술하고 비즈니스 로직에 집중할 수 있다.

자발적 상호연동에 필요한 상호동작을 분석하고 공통의 상호동작 프로세스를 정의해서 공통의 소프트웨어 프레임워크로 제공할 수 있다. 이는 개발자의 부담을 줄이고 견고한 소프트웨어의 개발을 돕는다.

향후 연구로는 조직과 역할에 대한 명세 모델을 정의하는 것이다. 명세 모델은 MDA(Model-driven Architecture)[9]와 같이 개발 단계에서 응용의 개발을 자동화하거나 실행 단계에서 사용될 수 있는 가능성을 제공한다. 응용의 실행에서 중요한 요소중에 하나는 유비쿼터스 응용, 즉 조직의 생성에서부터 소멸까지의 생명주기를 관리하는 것이다. 즉, 조직에 기반한 응용의 관리를 의미한다. 이를 위해서 역할 조직 모델을 확장하고 이를 구현하고 실행하는 방안에 대한 연구가 필요하다.

## 참 고 문 헌

[1] Banavar, G., J. Beck, E. Gluzberg, J. Munson, J. Sussman, and D. Zukowski, “Challenges : an application model for pervasive computing”, *Proceedings of the 6th annual international conference on Mobile computing and networking*, (2000), pp.266-274.

[2] Cabri, G., L. Leonardi, and F. Zambonelli, “BRAIN : A Framework for Flexible Role-Based Interactions in Multiagent Systems”, *On The Move to Meaningful Internet Systems 2003 : CoopIS,* DOA, and ODBASE, (2003), pp.145-161.

[3] Colman, A. and J. Han, “Roles, players and adaptable organizations”, *Appl. Ontol.,* Vol. 2, No.2(2007), pp.105-126.

[4] Colman, A. and J. Han, “Using role-based coordination to achieve software adaptability”, *Science of Computer Programming*, Vol.64, No.2(2007), pp.223-245.

[5] Costa, C. A. D., A. C. Yamin, and C. F. R. Geyer, “Toward a General Software Infrastructure for Ubiquitous Computing”, *IEEE Pervasive Computing*, Vol.7, No.1(2008), pp. 64-73.

[6] Ferber, J., O. Gutknecht, and F. Michel, “From Agents to Organizations : An Organizational View of Multi-agent Systems”, *Agent-Oriented Software Engineering IV*, (2004), pp.443-459.

[7] Jung, J. K. Ryu, and J. Lee, “Role-based application model for supporting spontaneous interaction”, *Proceedings of Networked Computing (INC), 6th International Conference*, (2010), pp.1-6.

[8] Kindberg, T. and A. Fox, “System Software for Ubiquitous Computing”, *IEEE Pervasive Computing*, Vol.1, No.1(2002), pp. 70-81.

[9] Kleppe, A., J. Warmer, and W. Bast, *MDA Explained : The Model Driven Architecture(TM) : Practice and Promise.* Addison-

Wesley Professional, 2003.

[10] Latvakoski, J., D. Pakkala, and P. Paakkonen, “A communication architecture for spontaneous systems”, *IEEE Wireless Communications,* Vol.11, No.3(2004), pp.36-42.

[11] Niemel, E. and J. Latvakoski, “Survey of requirements and solutions for ubiquitous software”, *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, (2004), pp.71-78.

[12] Schulzrinne, H. and E. Wedlund, “Application-layer mobility using SIP”, *IEEE Service Portability and Virtual Customer Environments*, (2000), pp.29-36.

[13] Zhu, H. and M. C. Zhou, “Role-based collaboration and its kernel mechanisms”, *Systems, Man, and Cybernetics, Part C : Applications and Reviews, IEEE Transactions on*, Vol.36, No.4(2006), pp.578-589.

[14] Zhu, H. and M. C. Zhou, “Roles in Information Systems : A Survey, Systems”, *Man, and Cybernetics, Part C : Applications and Reviews*, *IEEE Transactions on*, Vol.38, No.3(2008), pp.377-396.

## ◈ 저 자 소 개 ◈


**정 종 윤 (jongyun@ajou.ac.kr)**

아주대학교 컴퓨터공학과 석사학위를 취득하고 동 대학에서 정보통신공학 박사과정에 재학 중이다. 주요 관심분야는 컴포넌트 결합, 웹 서비스 결합, 모바일 시스템, 유비쿼터스 시스템, 미들웨어 관련 기술 등이다.

**류 기 열 (kryu@ajou.ac.kr)**

현재 아주대학교 정보컴퓨터공학과에 교수로 재직 중이다. KAIST 전산학 전공으로 공학석사 및 공학박사 학위를 취득하였으며, 주요 관심분야는 Large Scale Software Systems, 유비쿼터스 시스템, 프로그래밍 언어 등이다.

**이 정 태 (jungtae@ajou.ac.kr)**

현재 아주대학교 소프트웨어융합학과에 교수로 재직 중이다. 서울대학교 계산학 이학석사 및 계산학 공학박사 학위를 취득하였으며, 주요 관심분야는 컴포넌트 베이스 시스템, 미들웨어, 객체지향 응용 프레임워크 등이다.