Vol.2, No.9 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2013.2.9.369

An Organization Framework for Role-based Adaptive Distributed Systems

Hwang Seong-yun[†] · Jung Jong-yun[††] · Lee Jung-tae[†††] · Ryu Ki-yeol[††††]

ABSTRACT

Recently, role-based distributed system models have been proposed to support adaptive interactions in ubiquitous application environment. A Role-based distributed model regards an application as an organization composed of roles, and separate players running role processes from the roles. When an application is running, it binds a role with a player, and the player runs dynamically assigned role processes provided by an application for supporting adaptability. However, there has not been much attention on researches about development and runtime environment for role-based distributed systems. In this paper we suggest an application framework as an environment for developing and executing role-base distributed systems. The application framework is divided into two parts: an organization framework to manage and construct an organization composed of roles necessary in the application, and a player framework to provide running environment for players. In this paper, we focus on the organization framework which supports the creation and management of organizations, directory service for players and allocation of players to roles, and message brokering between roles and players. The proposed framework makes developers to be able to develop highly adaptive distributed systems in the ubiquitous environment.

Keywords : Role-Based, Adaptive Distributed System, Application Framework

역할기반 적응형 분산 시스템을 위한 조직 프레임워크

황 성 윤[†] · 정 종 윤[††] · 이 정 태[†††] · 류 기 열[††††]

요 약

최근 유비쿼터스 컴퓨팅환경에서 적응적 상호작용을 지원하기 위한 역할기반 분산 시스템 모델이 제안되었다. 역할기반 분산 시스템 모델은 응용을 추상적인 역할들로 이루어진 조직으로 보고, 조직을 구성하는 역할과 실제 행위를 수행하는 행위자를 분리한다. 실행 시 응용은 행위자를 역할에 연결하고, 행위자는 수행할 역할 업무를 동적으로 응용으로부터 제공받아 실행함으로써 적응성을 높여주는 모델이다. 하지만 이런 역할기반의 분산 시스템 모델을 위한 실행환경의 구축에 대한 연구는 미비한 상태이다. 본 논문에서는 역할기반 분산 시스템 모델의 실행환경으로 응용 프레임워크를 제안한다. 응용 프레임워크는 응용에 필요한 역할들로 이루어진 조직을 구성하고 관리하는 조직 프레임워크와 행위자의 역할 수행과 역할 프로세스 관리를 수행하는 행위자 프레임워크로 나누어진다. 본 논문에서는 기존에 제안된 행위자 프레임워크와 상호 결합되어 동작할 수 있는 조직 프레임워크를 제안한다. 제안한 조직 프레임워크는 조직의 생성과 관리, 행위자에 대한 디렉토리 서비스 및 역할의 할당, 역할과 행위자사이의 메시지 브로커의 역할을 수행한다. 본 논문에서 제안한 조직 프레임워크는 개발자로 하여금 적응성이 높은 분산 시스템을 개발할 수 있도록 도와준다.

키워드 : 역할기반, 적응형 분산 시스템, 응용 프레임워크

1. 서 론

최근 활발한 연구가 진행되고 있는 유비쿼터스 시스템이나 모바일 시스템은 다양한 컴퓨팅 개체들의 동적 참여가 이루어지고 상호동작을 수행하는 분산 시스템의 일종이다. 이러한 분산 응용 시스템들은 구성요소들과 그들 간의 관계가 비결정적인 특성을 가진다[1]. 비결정적인 분산 응용 시스템은 실행하는 컴퓨팅 디바이스 변경과 각 컴퓨팅 디바이스의 네트워크 이동이 시스템 실행 중에 빈번히 발생한다. 이를 위해 실행시간에 각 컴퓨팅 개체의 변경, 네트워크 환경의 이동과 같은 다양한 변화를 동적으로 지원하는 특성인 적응성을 향상시키기 위한 연구가 활발히 이루어지고 있다[2, 3].

적응성을 지원하기 위한 다양한 연구들 중에서 역할기반(role-based) 분산 시스템에 대한 연구인 ROAD(Role-Oriented

† 준 회 원 : 아주대학교 컴퓨터공학과 석사과정
†† 준 회 원 : 아주대학교 정보통신전문대학원 박사과정
††† 종신회원 : 아주대학교 소프트웨어융합학과 교수
†††† 종신회원 : 아주대학교 정보컴퓨터공학과 교수
논문접수 : 2013년 5월 14일
수 정 일 : 1차 2013년 7월 25일
심사완료 : 2013년 7월 26일
* Corresponding Author : Ryu Ki-yeol(kryu@ajou.ac.kr)

Adaptive Design)[4]에서는 응용을 특정 기능을 수행하는 역할들로 이루어진 하나의 조직으로 간주한다. 각 컴퓨팅 개체는 각 역할을 담당하는 행위자로 분리하여 실행 시간에 동적으로 역할과 연결함으로써 컴퓨팅 개체와는 독립적인 조직을 생성하거나 변경할 수 있게 하였다. 기존의 역할기반 모델에 대한 연구에서는 시스템의 설계 단계에서만 역할과 행위자를 분리하여 실제 실행시간에서의 적응성 확보에 미흡했다. ROAD는 실행시간의 적응성 확보를 보완하기 위해 실행시간에서도 역할과 행위자를 분리한 모델을 보여줌으로써 이를 보완하고 있다.

하지만 ROAD에서도 설계와 실행의 중간 단계인 구현 단계에서 컴퓨팅 개체들을 실제로 구현하고 배치하기 위한 현실적인 방안에 대해서는 고려하지 않고 있어 실제 모델의 적용에 있어서 설계와 실행 단계에서 동일하게 적용시키기가 어려워 구현 단계에서의 적용 방안에 대한 연구가 진행될 필요가 있다. 그리고 ROAD의 모델에서는 행위자가 수행하는 역할에 대한 업무를 행위자 내부에 정의하여 행위자는 사전에 정의된 역할의 업무만을 수행하기 때문에 이렇게 행위자가 미리 정의된 기능만을 수행하는 방법은 높은 수준의 적응성을 요구하는 분산 응용 시스템에는 적합하지 못하다.

이러한 문제점들을 개선하기 위해 행위자가 수행하는 개별 역할의 업무를 역할 프로세스로 정의하고 역할 외부에 역할 프로세스를 따로 두어 해당 역할의 행위자가 실행시간에 동적으로 역할 프로세스를 할당받아 수행하도록 하는 분산 시스템 모델을 제안되었다[5]. 또한 [5]에서는 행위자의 구현과 실행을 위한 행위자 프레임워크를 제시하고 있다[5]. 하지만 실제 분산 응용 시스템을 구현하기 위해서는 개별 행위자 프레임워크 외에도 조직의 구성과 관리를 수행하는 조직 프레임워크가 필요하나 아직까지 조직 프레임워크에 대한 연구가 없는 실정이다.

본 논문에서는 역할 프로세스를 적용하는 적응형 분산 시스템 모델의 구현에 도움을 주기 위한 응용 프레임워크를 제안한다. 응용 프레임워크는 시스템의 조직을 구성하고 관리하기 위한 조직 프레임워크와 행위자의 역할 수행과 역할 프로세스 관리를 위한 행위자 프레임워크로 분류된다. 응용 프레임워크를 구현하기 위해서는 기존 연구에서 제시된 행위자 프레임워크와 결합하는 조직 프레임워크의 설계가 필요하다. 본 논문에서는 응용의 조직이 행위자 프레임워크와 협업하면서 발생하는 상호작용을 분석하여 조직 프레임워크의 요구사항을 파악하고 필요한 기능을 정의하여 각 기능을 수행하기 위한 조직 프레임워크의 구조를 설계하여 구현 방안을 제시하고자 한다.

2장에서는 분산 응용 시스템을 위한 프레임워크와 역할기반 분산시스템에 대한 기존 연구에 대해 분석한다. 3장에서는 이러한 적응형 분산 시스템 모델을 구현하기 위한 응용 프레임워크의 구성을 보여주고, 4장에서 본 논문에서 제안하는 조직 프레임워크를 설계한다. 5장에서는 이 프레임워크를 시나리오를 가지고 적용해본 후 평가하고, 6장에서 결론을 맺는다.

2. 적응성 확보를 역할기반 시스템 모델 연구

분산 응용 시스템을 위한 기존의 프레임워크는 대부분 컴포넌트기반(component-based)의 분산 시스템 구조를 제안하고 있다[6, 7]. 하지만 컴포넌트 기반의 구조는 컴퓨팅 개체가 직접적으로 연결되어 개체간의 변화의 대처하기 어려워 몇몇 연구에서는 컴포넌트간의 인터페이스 역할을 하는 계약이라는 요소를 추가[8, 9]하거나 컴포넌트 개념 대신 역할과 행위자라는 개념의 모델로 시스템을 구성하는 역할기반 연구도 이루어지고 있다[10]. 이 장에서는 역할기반 연구 중 하나인 ROAD 모델에 대해 분석한다.

2.1 ROAD 개요

ROAD는 컴포넌트를 역할과 행위자로 분리하여 각 컴퓨팅 개체들을 역할을 수행하는 행위자 개념으로 보고 역할을 통해 간접적으로 연결하여 일부 컴퓨팅 개체의 문제가 다른 개체에게 직접적으로 영향을 주지 않도록 하고 시스템 기능이나 개체의 변경을 실행시간에 대처할 수 있도록 한다. 그리고 역할들의 네트워크인 조직을 구성하여 조직의 변경 없이 역할과 행위자의 연결을 변경하는 것만으로 장비의 변경을 수행하도록 하고 역할과 역할 간의 연결에는 계약을 두어 각 역할의 변경에 동적으로 대처하기 쉽도록 구성하고 있다.

ROAD에서는 역할기반의 조직을 구성하고 행위자가 이 조직을 통해 역할을 수행하는 모델을 제시함으로써 소프트웨어의 적응성을 높일 수 있는 방안을 제안하고 있다. Fig. 1는 ROAD 모델을 적용한 한 가지 예로서, 카페의 상황을 조직화된 역할과 이를 수행하는 행위자로 나타낸 그림이다. 카페에는 Cafe Manager, Shift Manager 등 5가지 역할이 존재하며 이들은 계약으로 묶여있고, Player1부터 Player4까지 4개의 행위자가 해당 역할을 맡아 수행함으로써 서로의 상호작용을 통해 시스템의 목적을 달성한다.

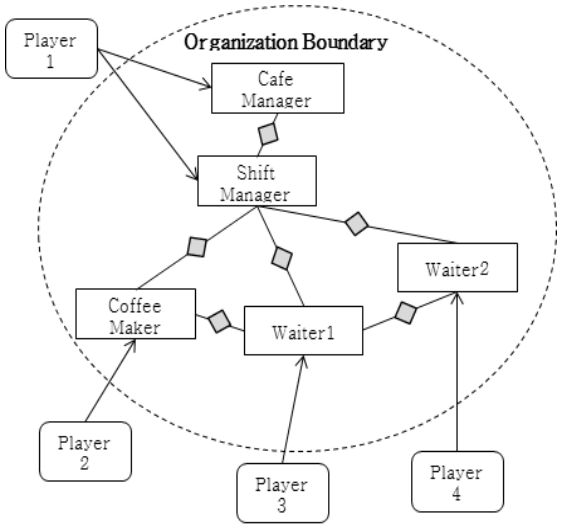


Fig. 1. Organized Roles

ROAD 연구에서는 기존의 컴포넌트기반 적응성 연구와 역할 개념을 기반으로 한 ROAD의 적응성을 항목별로 정리하여 평가하고 있다[4]. 이 평가를 통해 ROAD의 역할기반 연구가 실행시간 적응성 확보에 있어서 컴포넌트 기반의 연구보다 비교적 다양한 장점을 갖는다는 사실을 확인할 수 있다.

2.2 ROAD의 한계점

ROAD 모델에서는 행위자가 역할을 통해 수행하는 작업 명령, 즉 기능을 프로세스로 정의하였으며, 프로세스의 배치 방법을 분석하여 행위자 내부에 배치하는 Fig. 2와 같은 방법을 택하고 있다.

그러나 ROAD 모델이 채택한 프로세스의 배치 방법은 행위자가 정해진 프로세스의 기능만을 수행할 수 있기 때문에, 비록 환경 변화에 대응하여 동적으로 교체될 수는 있으나 해당 역할을 수행하는 프로세스를 사전에 탑재한 행위자를 찾지 못하는 경우 조직 구성이 무너지고 서비스가 중지될 수 있는 위험이 존재한다.

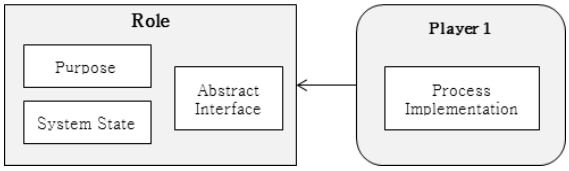


Fig. 2. Process in player

2.3 역할 프로세스의 분리

ROAD의 이 같은 문제점을 해결하기 위한 연구로 역할 외부에 프로세스를 배치하는 방안에 대한 연구가 진행되었다. 이 방안은 Fig. 3과 같이 프로세스를 역할의 외부에 배치하고 이를 역할 프로세스로 정의한다[5].

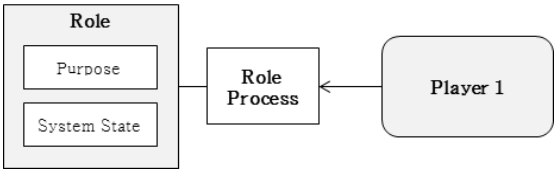


Fig. 3. External process of role

이러한 프로세스 배치 방법을 선택하게 되면 행위자는 조직에 참여하여 역할을 수행할 때, 역할의 외부에 정의된 역할 프로세스만 자신의 환경으로 복사하여 실행하면 된다. 따라서 ROAD에서처럼 행위자 내에 프로세스를 정의하지 않기 때문에 행위자는 고정된 역할이 아닌 자유로운 역할 수행이 가능하며, 프로세스 내에 정의된 기능적 구현이 변경되는 경우에도 프로세스를 사전에 정의해두었던 모든 행위자를 수정해야 하는 문제점 역시 해결 가능하다. 이와 같은 장점은 ROAD 모델보다 실행 시간 적응성을 더 유연하게 확보할 수 있게 해주는 중요한 차이점이라 할 수 있다.

하지만 역할 프로세스에 대한 연구에서는 이를 구현하기 위해서 조직을 구성하고 관리하는 대부분의 작업을 개발자의 몫으로 남겨두고 있다. 개발자는 이러한 모델을 구현하려면 조직 관리와 역할의 구현, 역할과 역할 프로세스의 설치 관리 등 많은 작업을 직접 구현해야 하는 문제점이 있다. 이를 보완하기 위해 본 논문에서는 역할 프로세스 개념을 적용한 적응형 분산 시스템 모델을 토대로 한 응용 프레임워크를 제안한다.

3. 응용 프레임워크

본 장에서는 역할기반 적응형 분산 시스템 모델의 구현방안의 하나로 응용 프레임워크를 구조를 소개하고 이를 구성하는 행위자 프레임워크와 조직 프레임워크의 상호작용을 분석한다.

3.1 시스템 동작 환경

본 논문에서 제안하고자 하는 응용 프레임워크의 구체적인 아키텍처를 기술하기에 앞서, 제안하는 프레임워크 동작 환경을 환경을 간단히 살펴본다. 시스템이 동작하기 위해서는 Fig. 4와 같이 크게 세 가지 구성요소가 존재한다. 첫 번째는 조직에 대한 정보를 저장하여 조직을 구성하고 관리하기 위한 서버가 필요하다. 두 번째는 행위자에 대한 정보를 저장하고 각 행위자에 대한 디렉터리 서비스를 하기 위한 잘 알려진 서버가 필요하다. 이 두 가지 서버는 본 논문에서 구현하고자 하는 조직 프레임워크에 의해 동작하며 이들의 세부 기능 및 구성은 이후 자세히 기술한다.

시스템이 동작하기 위한 세 번째 구성요소는 행위자로, 각 역할을 수행하기 위한 단말 장치를 의미하며 행위자 컨테이너 프레임워크를 탑재하고 있어야 한다.

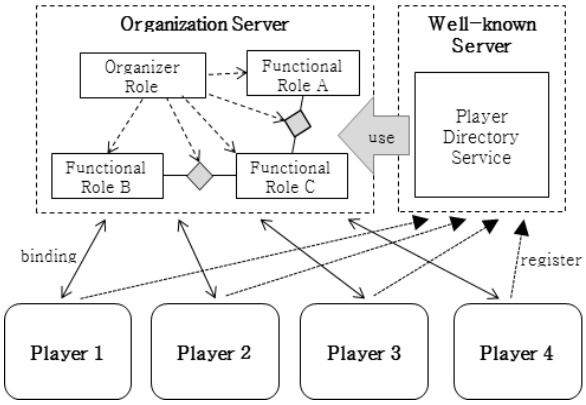


Fig. 4. System running environment

3.2 응용 프레임워크 구조

행위자와 조직에 대한 다양한 요구사항들을 해결하기 위해서 적응형 분산 시스템을 위한 응용 프레임워크의 전체적인 구조는 Fig. 5와 같다. 응용 프레임워크는 조직 프레임워

크와 행위자 프레임워크로 나누어진다. 본 논문에서는 기존 연구[5]에서 제안한 행위자의 실행환경인 행위자 프레임워크와 상호 연동할 수 있는 조직 프레임워크를 제안한다. 행위자 프레임워크는 응용의 시스템 동작 환경의 행위자 개발을 위한 프레임워크를 제공하고 있고 조직 프레임워크에서는 역할들로 이루어진 조직을 구성하고 관리하며 행위자 정보를 제공하여 역할과 행위자를 결합하기 위해 필요한 디렉터리 서비스를 위한 프레임워크를 제공한다.

조직 프레임워크의 설계에 있어서 행위자 프레임워크와 개발 범위의 구분을 위해 행위자 프레임워크와의 상호작용에 대해 알아볼 필요가 있다. 조직 프레임워크에서는 행위자에 대한 디렉터리 서비스를 수행하기 위해서 행위자의 정보 등록을 잘 알려진 서버에서 수행할 수 있도록 해주어야 한다. 그리고 역할을 수행할 행위자 프레임워크에 역할 프로세스를 설치하고 실행시간에 응용의 실제 메시지를 주고받게 된다.

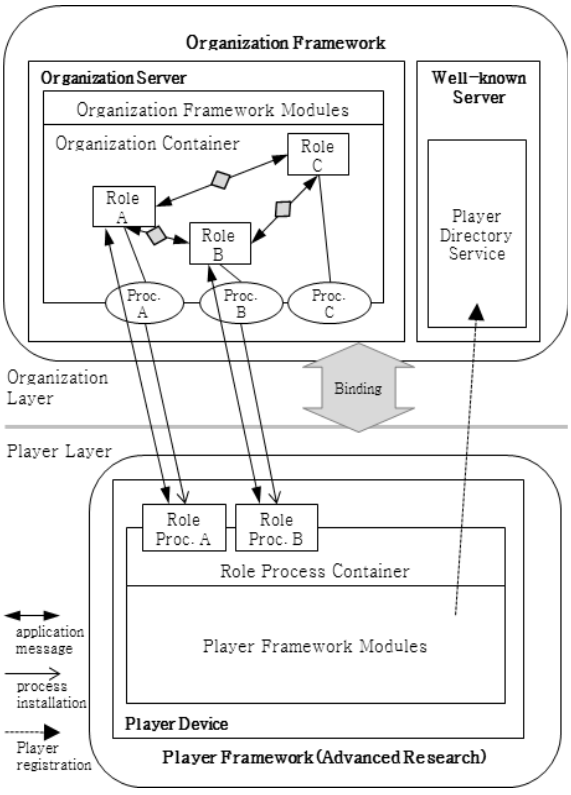


Fig. 5. Player framework and organization framework

4. 조직 프레임워크

이 장에서는 조직 프레임워크의 요구사항을 분석하여 이를 바탕으로 구체적인 조직 프레임워크를 설계한다.

4.1 조직 프레임워크의 요구사항

적응형 분산 시스템을 적용하기 위한 조직 프레임워크는 다음과 같은 요구사항을 만족해야 한다.

- 조직의 생성 및 관리: 조직을 구성하는 역할과 역할간의 관계를 기술하는 계약의 인스턴스를 생성하고 저장, 관리하는 기능이다. 상황에 따라 조직은 실행시간에 변경되기도 하기 때문에 역할과 계약을 실행시간에 변경할 수도 있어야 한다.
- 행위자 정보에 대한 디렉터리 서비스 및 역할과의 연결: 조직 프레임워크에서는 필요한 행위자를 식별하기 위해 행위자가 제공하는 기능과 행위자를 연결시키기 위한 정보를 저장하고 서비스를 수행하여야 한다. 또한 프레임워크는 이 디렉터리 서비스를 통해 역할 수행이 가능한 행위자를 역할에 연결할 수 있어야 한다. 행위자는 행위자 내부에서 수행할 수 있는 역할을 정의하지 않기 때문에 행위자 자신이 할 수 있는 기능적 요소를 서버에 등록하여 그 기능으로 수행할 수 있는 역할과 연결될 수 있도록 해야 한다.
- 역할 프로세스의 할당: 역할 외부에 정의된 역할 프로세스는 역할과 행위자가 연결될 때 행위자에게 할당된다. 그리고 역할 프로세스가 변경될 경우나 행위자의 교체가 이루어질 경우, 행위자에게 이를 재할당하는 과정이 필요하다.
- 실행 시간의 메시지 통신: 응용을 실행하는 동안 역할과 역할, 역할과 행위자 사이의 데이터 전송이나 조직의 구성, 변경과 관련된 메시지를 전달하게 된다. 조직 프레임워크에서는 이를 수행하기 위한 모듈이 필요하다. Fig. 6은 ROAD에서 제시한 역할과 역할 사이의 메시지 전송 방식을 보여준다. 조직 프레임워크에서는 이러한 메시지 전송 방식을 수행할 수 있는 기능을 갖추어야 한다.

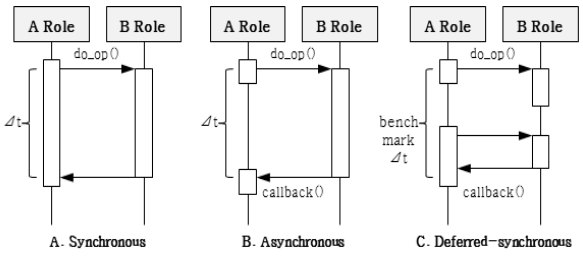


Fig. 6. Message passing type between roles

4.2 조직 프레임워크의 구조

이 절에서는 앞에서 언급한 적응형 분산 시스템 모델을 구현하기 위해 필요한 요구사항을 반영하여 조직 프레임워크의 구조를 설계한다. 조직 프레임워크의 전체적인 구조는 Fig. 7과 같다.

1) 실행환경 모듈

실행환경 모듈(Run Time Module)에서는 조직 프레임워크의 시작 및 초기화부터 종료까지의 전반적인 요소를 관리한다. 이 실행환경 모듈에서 프레임워크 코어 모듈을 불러오는 조직 프레임워크의 시작에서부터, 조직의 초기화, 실행, 종료 단계를 상태별로 관리하고 이를 위한 환경을 구성한다.

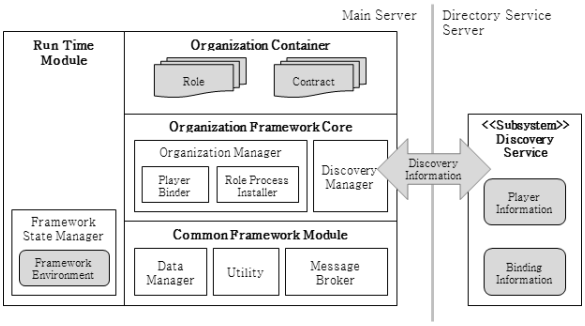


Fig. 7. Organization framework structure

2) 조직 관리자와 조직 컨테이너

조직 관리자(Organization Manager)는 조직의 초기화 단계에서 조직자 역할을 생성하고 이를 해당 조직의 조직자와 바인딩하여 조직자가 조직의 관리를 수행할 수 있도록 한다. 그리고 조직자는 조직 관리자를 통해서 조직 컨테이너(Organization Container)를 구성하고 계약의 인스턴스를 생성한다. 그리고 조직의 실행 단계에서 조직자는 조직 관리자에서 조직 컨테이너의 조직 변경 관리, 역할과 행위자의 연결 관리를 지시한다. 역할과 행위자가 연결되면 조직 관리자는 해당 역할의 역할 프로세스를 불러와 가지고 온 역할 프로세스를 행위자 환경에 설치한다. 역할 프로세스 설치자는 행위자에게 이미 설치된 동일한 역할 프로세스가 있을 경우와 조직 내의 역할 프로세스가 변경되었을 경우를 고려하여 역할 프로세스를 설치할 것인지를 결정한다. 조직 컨테이너는 이러한 조직 관리 작업에서 역할과 계약을 관리하기 위한 인터페이스 기능을 한다.

3) 메시지 브로커

조직의 생성과 실행, 종료 단계에서는 조직의 구성 변경이나 행위자의 데이터 전송 등 다양한 메시지가 전송된다. 이 메시지들 중에서 행위자와 조직, 조직과 조직 프로세스 간의 메시지 전송과 같은, 다른 컴퓨팅 개체와의 통신과 관련된 메시지 전송을 메시지 브로커(Message Broker)에서 관리한다. 이에 대해서는 이후 자세히 설명한다.

4) 데이터 관리자와 유틸리티

조직 프레임워크에서는 서버에서 사용하는 역할의 생성에 필요한 데이터나 역할의 수행 정지 상태에서 저장되는 데이터들의 저장과 관리를 할 데이터 관리자(Data Manager)와 메시지 통신, 메모리 관리 등 다양한 자원에 대한 유틸리티(Utility)를 제공한다.

5) 발견 서비스와 발견 관리자

조직 관리자가 실행 단계에서 역할과 행위자의 연결을 관리할 때 잘 알려진 서버 내에 행위자가 등록한 기능 정보를 이용하게 된다. 이 기능 정보를 통해 각 행위자에 대한 역할의 연결 가능성을 확인하고 역할과 행위자를 연결한다. 발견 서비스(Discovery Service)에서는 이렇게 연결된 연결

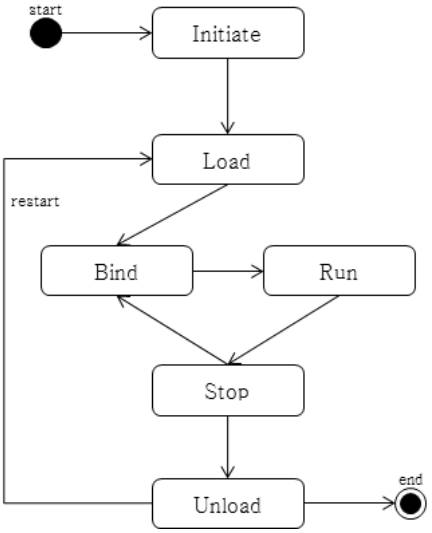


Fig. 8. Life cycle of organization

정보를 저장하여 역할과 행위자의 메시지 통신에 사용할 수 있도록 제공하고 이는 조직 프레임워크의 발견 관리자(Discovery Manager)에서 관리한다.

4.3 조직 프레임워크의 생명주기

조직 프레임워크는 해당 응용에서 보유하고 있는 조직의 생명주기에 따라 상태가 결정된다. 조직 프레임워크를 실행할 때는 크게 프레임워크의 초기화, 실행, 종료 과정을 거치고 프레임워크의 실행 과정에 Fig. 8에서와 같이 조직의 생명주기에 따라 응용에 포함된 각 조직을 관리하게 된다. 응용 내에는 다수의 조직이 포함될 수 있고 이는 개별적인 생명주기를 가지고 관리된다. 조직 프레임워크의 실행 과정에서 각각의 조직을 상태별로 관리함으로써 하나의 프레임워크에서 다양한 조직에 대한 관리를 수행할 수 있게 한다.

1) 프레임워크의 초기화

프레임워크가 시작되면 Fig. 9에서처럼 실행환경 모듈에서 프레임워크의 모듈들을 생성함으로써 프레임워크를 초기화하는 작업을 수행하게 된다. 실행환경 모듈은 먼저 공통 프레임워크 모듈(Common Framework Module)을 생성하고 메시지 브로커의 통신을 초기화하여 행위자와의 통신을 시작할 수 있도록 한다. 그리고 각 조직에 대한 조직 프레임워크 코어(Organization Framework Core)를 생성하고 조직 프레임워크 코어 내의 조직 관리자에서 조직 컨테이너를 생성한다. 조직과 관련된 조직 프레임워크 코어와 조직 컨테이너는 한 응용에서 여러 조직을 관리할 때는 Fig. 10과 같이 조직별로 다중 쓰레드나 다중 프로세스 방식을 통해 개별 조직을 상태에 따라 관리하도록 한다.

2) 조직의 초기화 상태

조직을 생성하기 위해서는 조직자가 그 역할을 수행할 수 있어야 한다. 이를 위해 조직자 역할을 생성하고 조직자 역

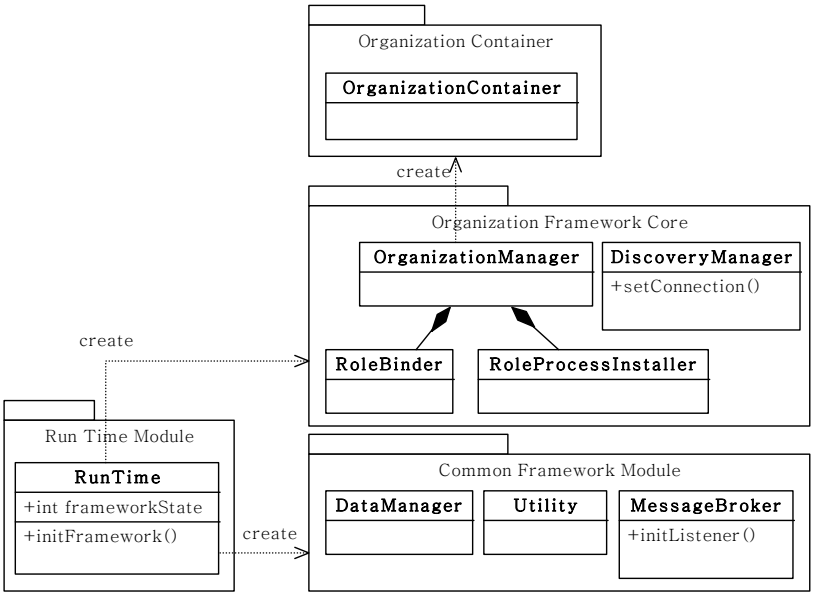


Fig. 9. Process in framework initiation

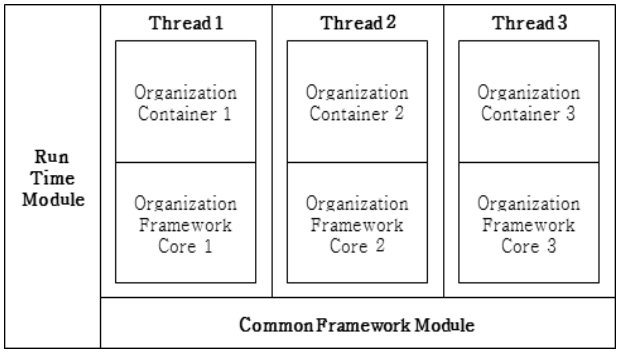


Fig. 10. Multi organization management structure by multi-thread methodology

할을 수행할 행위자에게 연결하는 과정이 필요하다. 이러한 과정을 수행하는 상태가 조직의 초기화 상태이다. 조직의 초기화 상태에서는 조직 관리자가 조직 컨테이너 내에 조직자 역할의 인스턴스를 생성하고 조직자 역할을 수행할 행위자에게 이를 연결한다. 조직자 역할을 수행할 행위자는 조직마다 고유하게 지정되어 미리 행위자 정보에 저장되어 있도록 한다. 이러한 행위자 정보에 대해서는 5장에서 자세히 다루도록 한다.

3) 조직의 로드 상태

조직자 역할이 연결되고 나면 조직자 역할을 맡은 행위자가 나머지 기능적 역할들을 생성할 수 있다. 조직의 로드 상태에서는 조직자 역할이 기능적 역할들의 인스턴스를 생성하고 그 사이의 계약들의 인스턴스를 생성하여 조직 전체를 구성하도록 한다. 이 상태가 종료되면 조직 컨테이너는 Fig. 11와 같은 구조를 갖추게 된다.

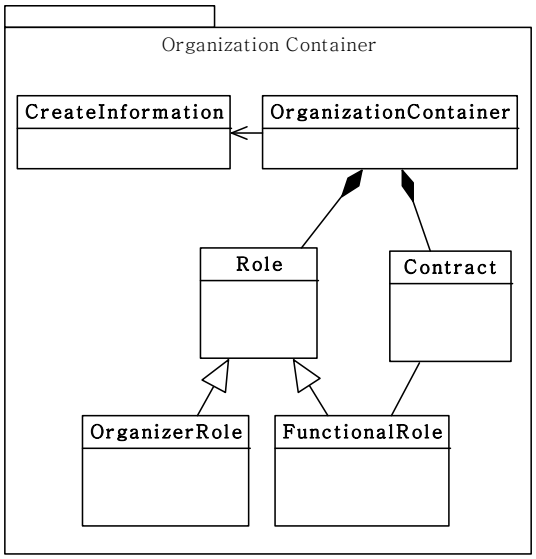


Fig. 11. Organization container structure

4) 조직의 연결 상태

역할이 수행되기 위해서는 각 역할이 모두 행위자에게 연결되어야 한다. 역할의 연결은 조직자가 직접 역할의 연결을 요청하거나 발견 관리자에서 불러온 행위자의 정보를 통해 조직 관리자가 연결 조건을 분석하고 이를 수행할 수도 있다. 모든 역할에 한 행위자씩 연결이 수행되면 조직은 연결 상태를 종료하고 실행 상태로 변경된다.

5) 조직의 실행 상태

조직을 구성하는 각 역할들이 실제 응용을 실행하는 상태가 조직의 실행 상태이다. 조직자 역할에서 다른 역할에게

조직의 실행을 알리면서 조직의 실행 상태는 시작되고 각 역할들에 연결된 행위자들은 각자가 맡은 역할을 수행하게 된다. 조직의 실행 상태에서는 응용의 다양한 메시지를 역할과 역할, 역할과 역할 프로세스 사이에서 전달한다. 이 과정에서 하나의 역할이 수행될 때 다른 역할에 메시지를 전송할 경우 단순한 클래스의 메소드 호출 방법으로는 프레임워크의 실행시간 메시지 통신에 대한 요구사항을 만족시키기 어렵다. 이를 위해 조직 프레임워크에서는 다른 역할의 메소드를 호출하는 공통 메소드를 정의하여 계약을 통해 메소드 호출을 수행하고 메소드 호출 방식에 따라 동기, 비동기 등의 메시지 통신을 구현한다. 조직이 실행되는 도중에 통신 장애나 기타 이유 등으로 역할의 연결이 취소되어 조직을 실행할 수 없는 상태가 되거나 조직의 변경이나 조직의 종료가 필요할 경우 조직은 실행 중인 작업을 중단하기 위해 정지 상태로 변경된다.

6) 조직의 정지 상태

조직이 실행 중이던 정보는 정지 상태에서 각 역할에 저장되어 이후 다시 조직이 실행되었을 때 이후의 작업을 수행할 수 있도록 한다. 조직의 정지 상태를 위한 데이터 저장이 끝나면 역할의 연결을 수행하기 위하여 연결 상태로 변경하거나 조직의 실행을 종료하기 위하여 언로드 상태로 변경한다.

7) 조직의 언로드 상태

조직의 언로드 상태는 조직의 실행을 종료하기 위해 조직 컨테이너 내의 역할과 계약 인스턴스들을 모두 제거하는 상태이다. 조직의 모든 인스턴스를 제거하고 나면 조직은 언로드 상태를 끝내면서 필요할 경우 재시작을 수행하여 다시 로드 상태로 변경되거나 조직의 생명주기를 종료하게 된다. 프레임워크 내의 모든 조직이 종료되고 나면 프레임워크는 종료 작업을 수행하게 된다.

8) 프레임워크의 종료

프레임워크의 모든 조직이 언로드되고 나서 프레임워크의 모든 모듈을 제거하고 프레임워크를 종료하기 위한 작업들을 수행한다.

4.4 조직 프레임워크의 설계

앞에서 조직 프레임워크의 생명주기에 따라 조직 프레임워크의 실행에 필요한 모듈에 대해 알아보았다. 이 절에서는 이들을 실제 구현하기 위해 조직 프레임워크의 클래스와 데이터 구조를 설계하도록 한다.

1) 역할

역할은 조직 컨테이너 내에서 Fig. 12와 같은 구조의 클래스 상속 관계를 가진다. 역할은 프레임워크 단계에서 구성된 조직자 역할과 기능적 역할을 개발 단계에서 각각 상속받아 구현하게 된다. 개발자가 새롭게 적용한 클래스들은 프레임워크 내에 포함된 클래스들이 아니기 때문에 이를 생성할 생성 조건이 없다. 그래서 적용된 클래스들을 생성하기 위해서는 클래스의 생성 정보를 따로 저장하여 프레임워크가 실행되면 이 정보들을 바탕으로 동적인 역할 클래스의 생성을 수행해야 한다.

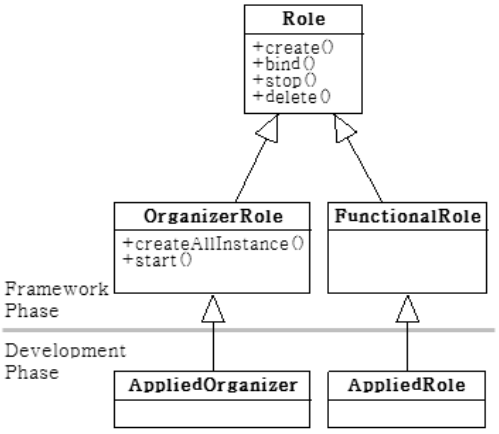


Fig. 12. Inheritance diagram of Roles

역할 클래스는 Fig. 12와 같이 클래스의 생성 및 제거, 행위자와의 연결과 정지 상태에 수행할 메소드들을 포함한다. 이는 역할 클래스를 상속하는 다양한 적용 클래스들에서 추가적인 작업을 수행할 수도 있다.

- 조직자 역할

조직자 역할 클래스는 Fig. 12와 같은 구조로 역할 클래스를 상속받으며 조직을 구성하기 위한 메소드들을 추가적으로 가진다. 조직을 구성하기 위해 조직자 역할 클래스는 기능적 역할의 인스턴스 생성과 제거, 변경과 관련된 메소드들을 가진다. 조직자 역할 클래스는 조직을 구성하는 조직자 역할 외에도 start() 메소드를 포함하면서 조직의 실행을 시작하는 시작자 역할도 수행한다. 조직자 역할은 조직의 로드 상태에서 다른 역할들의 인스턴스를 모두 생성한 후 조직의 실행 상태가 되면 start() 메소드를 실행하여 다른 역할들에게 응용의 시작을 알리게 된다.

- 기능적 역할

개발자의 대부분의 개발 작업은 기능적 역할을 구현하면서 이루어진다. 기능적 역할은 기능적 역할 클래스를 상속하도록 하는 적용 클래스를 통해 구현된다. 모든 기능적 역할은 적용 클래스를 생성하거나 변경하면서 관리되고 적용 클래스의 구현은 계약과 역할 프로세스에 대한 정보를 포함하게 된다.

2) 계약

계약은 역할간의 접근 권한, 메소드의 실행 순서화, 메시지의 전송을 관리한다. 이러한 정보는 역할의 적용 클래스 메소드를 구현하면서 메소드의 접근권한 설정, 메소드의 실

Process –> Role Message Protocol

Player ID	Role ID	Collee ID	Method Name	Parameter List	Synchronous Type

Role –> Process Message Protocol

Role ID	Player ID	Method Name	Parameter List

Organization –> Player Message Protocol

Binding Message	Player ID	Role ID	Role Name
Role Process Installation Message	Player ID	Role ID	Process Data

Player –> Discovery Server Message Protocol

Player ID	Register Type	Register Data

Fig. 13. Message Protocol

행 순서 지정을 하여 이들 데이터를 계약 클래스에 적용할 수 있도록 한다.

3) 역할 프로세스

역할 프로세스는 개발자가 역할을 개발하면서 생성한 메소드를 실제로 보유하게 된다. 그리고 해당 역할은 동일한 메소드의 내용을 역할 프로세스와의 통신을 수행하는 내용으로 변경하여 역할의 메소드 실행을 역할 프로세스에서 수행할 수 있도록 만들 수 있다.

4) 메시지 통신

- 역할과 역할 간의 메시지 통신

역할과 역할 간에는 계약이 존재하고 이는 역할 사이의 메시지 전송 방식을 다양하게 결정할 수 있도록 해준다. 계약은 메시지의 동기 전송, 비동기 전송, 연기된 동기 전송, 그리고 반환값이 없는 동기화가 없는 전송까지 네 가지의 전송 방식을 택한다. 동기 전송은 계약에서 호출되는 역할의 메소드를 직접 실행하여 반환값을 받아 호출한 역할에게 되돌려준다. 비동기 전송의 경우 호출되는 역할의 메소드를 실행하는 작업은 동일하게 하지만 계약에서 새로운 쓰레드를 생성하여 호출한 역할로 미리 돌아가고 호출된 메소드가 수행되고 나면 실제 반환값을 반환하게 된다. 연기된 동기 전송의 경우는 비동기 전송 방식과 동일하게 계약에서 쓰레드를 생성하여 메소드를 실행시키고 하나의 쓰레드는 호출한 역할로 돌아간다. 하지만 연기된 동기 전송의 경우는 호출된 메소드를 실행하고 나서 이 반환 정보를 계약 내에 저장해두고 호출한 역할에서 다시 반환값을 받고자 하는 메시지를 전송하면 이 반환값을 되돌려준다.

- 메시지 프로토콜

조직 프레임워크에서 행위자와의 메시지 통신이나 역할 프로세스의 설치와 같은 파일의 전송은 모두 메시지 브로커를 통해서 이루어진다. 메시지 브로커는 행위자 프레임워크와 통신하기 위해 간단한 메시지 프로토콜을 가지며 이는 Fig. 13과 같은 구조를 가진다.

5. 응용 프레임워크의 적용

제안한 조직 프레임워크와 행위자 프레임워크를 사용한 응용 프레임워크를 분산 시스템 환경인 모바일 유비쿼터스 시나리오에 적용시켜보고 ROAD와의 차이점을 확인하고 평가한다. 그리고 조직 프레임워크를 적용하면서 응용을 개발, 설치 및 배포, 실행의 세 가지 단계로 구분하여 프레임워크에 필요한 적용 사항을 살펴본다.

5.1 시나리오

모바일 단말기 및 다양한 장비들이 서로 협업하여 목적을 이루는 모바일 유비쿼터스 분산 환경을 응용 프레임워크를 적용하기 위한 시나리오로 구성한다. 이를 도식화하면 Fig. 14와 같다. 본 절에서는 이와 같은 환경에서 서비스될 수 있는 간단한 시나리오를 통해 ROAD의 분산 시스템 조직 구조와 본 논문에서 제안하는 분산 시스템 구조를 비교, 분석하여 살펴본다.

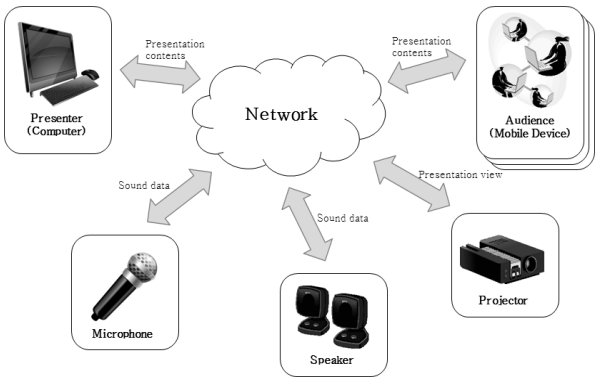


Fig. 14. Mobile ubiquitous seminar room scenario

다양한 장비들로 꾸며진 모바일 유비쿼터스 세미나실 시나리오는 Fig. 14와 같은 구조로 프레젠테이션 서비스를 제공한다. 발표자는 실내에 설치된 다양한 기기들을 프레젠테이션에 활용할 수 있으며, 이러한 기기들은 발표자가 가진 모바일 기기로 컨트롤이 가능하다. 발표자는 자신이 가진 발표 자료를 빔 프로젝터를 이용하여 보여주거나, 프린터로 인쇄를 하여 나누어 줄 수도 있으며, 설치된 서버를 통해 자료를 네트워크로 배포할 수도 있고, 또한 실내의 조명을 조정하는 것도 가능하다. 청중들은 자신의 모바일 기기를 활용하여 해당 세미나실에서 진행 중인 프레젠테이션의 발표 자료를 다운로드 받거나, 카메라를 통해 녹화되는 발표 영상을 제공받을 수 있다. 이와 유사한 환경의 세미나실은 다수가 존재할 수 있다고 가정한다.

ROAD의 모델로 분산 시스템 환경을 구현하는 경우 기존의 컴포넌트기반 연구와는 달리 환경의 변화들에 비교적 유연하게 대처할 수 있는 적응성을 확보할 수 있다. 시스템 전체의 서비스 제공을 위해 각 행위자가 수행해야 할 역할들을 정의하며, 이러한 역할들은 서로 연결되어 조직을 이루어 관리되게 된다.

5.2 응용 프레임워크의 시나리오 적용

여기서 발표자의 발표내용을 카메라로 찍어 저장할 수 있도록 세미나실 시스템을 변경하고자 한다. 세미나실 환경이 역할의 조직 구조로 구현이 되면 전체 시스템을 수정할 필요 없이 기존의 조직 구조에 카메라 역할을 추가하여 해당 역할을 수행하기 위한 행위자로 스마트폰이나 카메라 등 카메라 기능이 가능한 기기를 연결하여 변경된 시스템을 수행할 수 있다. 이것은 실행시간에 동적으로 행위자를 교체하거나 추가할 수 있음을 의미한다. 그러나 ROAD는 프로세스가 행위자 내에 정의되어 있기 때문에, 새롭게 카메라 역할을 수행할 행위자는 그 역할의 프로세스를 사전에 탑재하고 있어야만 역할의 수행이 가능하다. 따라서 카메라 역할의 프로세스를 탑재한 행위자를 찾지 못하는 경우 카메라 기능이 가능한 스마트폰 행위자가 존재해도 그 역할을 수행할 수 없다는 단점이 발생한다. 그리고 ROAD의 모델에서는 이를 해결하기 위해서 행위자들은 자신이 조직에 참여하지 않는 경우, 수행하지 않을 수도 있는 모든 역할의 프로세스를 모두 갖고 있어야 하는 비효율적인 구조를 가져야 하며, 발표자 역할의 프로세스를 변경할 경우에도 발표자 역할의 프로세스를 가지고 있는 모든 행위자를 변경해야 하는 번거로움도 존재한다.

이러한 단점들을 해결하기 위해 본 논문에서는 Fig. 15와 같이 역할 프로세스를 설치하는 방식의 모델을 채택하여 세미나실 환경에 적용한다. 기존의 ROAD의 조직구조와 달리 역할을 수행하는 프로세스를 역할과 분리하여 조직구조에 포함시키며, 행위자는 조직의 역할을 수행하기 위해 필요한 경우에만 동적으로 프로세스를 가져올 수 있다. 따라서 발표자는 발표자 역할만을 수행하는 것이 아니라, 필요에 따라서 다른 사람의 발표를 듣는 청중 역할을 수행할 수도 있다. 또한 발표자 프로세스의 변경이 발생하더라도 역할에 분리된 발표자 역할 프로세스만을 수정하고 발표자 역할을 할 행위자는 다시 발표자 프로세스를 가져오기만 하면 되기 때문에, ROAD 모델의 모든 행위자의 프로세스를 변경 및 배포해야 하는 단점도 해결 가능하다.

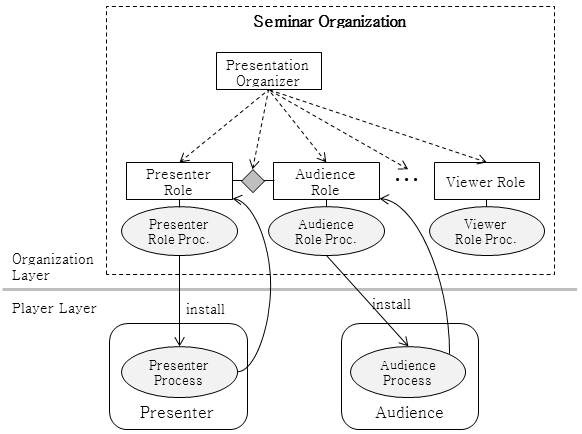


Fig. 15. Example to deploy role process

5.3 조직 프레임워크의 적용

1) 개발자의 응용 개발 단계

개발자는 응용을 개발하기 위해서 행위자 프레임워크와 조직 프레임워크를 사용하여 행위자 환경과 조직 서버 환경을 구성할 필요가 있다. 행위자 환경은 이미 설계된 행위자 컨테이너 프레임워크[5]를 바탕으로 각 행위자의 기능을 수행하기 위한 모듈들을 구현하고 이 모듈들을 각기 다른 디바이스에서 적용할 수 있도록 하드웨어 추상화 계층을 구현하게 된다.

그리고 본 논문에서 중점적으로 설계한 조직 프레임워크를 바탕으로 조직 서버 환경을 구성하기 위해서는 응용 내에 필요한 각 조직들을 구성할 필요가 있다. 조직을 구성하는 요소로는 조직자 역할과 기능적 역할, 각각의 기능적 역할들을 이어주는 계약의 세 가지 요소가 있다. 개발자는 이 중에서 조직의 초기화를 위한 조직자 역할의 모듈들을 실행시키고 조직의 실행을 위해 조직자 역할 내부의 시작 모듈을 구현하기 위해 조직자 역할 클래스를 상속받는 클래스를 생성해야 한다. 그리고 기능적 역할들의 모듈들을 구현하여 실제 응용의 실행을 수행할 수 있도록 기능적 역할 클래스를 상속받는 실제 역할들을 구현해야 한다. 조직의 나머지 요소인 계약의 경우는 구현한 각 역할 내의 모듈들에서 계약에 대한 속성을 정의하고 구현할 수 있도록 한다.

2) 응용의 설치 및 배포 단계

응용의 설치를 위해서는 조직 프레임워크와 행위자 프레임워크를 이용하여 구현된 조직 서버 프로그램과 행위자 프로그램을 설치해야 한다. 조직 서버 프로그램을 설치하고 실행하면 조직 프레임워크가 실행되고 조직자 역할을

```
<?xml version="1.0" encoding="utf-8" ?>
<SeminarOrganization>
   <OrganizerRole>
      <Name> PresentationOrganizer </Name>
      <Path> programmer.code.PresentationOrganizer </Path>
   </OrganizerRole>
   <Role>
      <Name> Presenter </Name>
      <Path> programmer.code.PresenterRole </Path>
   </Role>
   <Role>
      <Name> Audience </Name>
      <Path> programmer.code.AudienceRole </Path>
   </Role>
   ...
</SeminarOrganization>
```

Fig. 16. Information to create role instance

```
<?xml version="1.0" encoding="utf-8" ?>
<playerlist>
   <player name="player1" id="P1">
      <organization access="Organizer"> Organization1 </organization>
      <capability>
         <process clock="1GHz" />
         <display resolution="1024x768">
            <method name="showPicture" param="4" />
            <method name="showText" param="1" />
         </display>
         ...
      </capability>
   </player>
   ...
</playerList>
```

Fig. 17. Information of player capability and state

생성하여 조직에 포함된 기능적 역할들을 생성하면서 서버 프로그램은 시작된다. 조직자 역할과 기능적 역할들은 개발자가 프레임워크의 클래스를 상속받아 구현한 것으로 프레임워크에서 기존에 모르고 있던 이 클래스들의 인스턴스를 생성하기 위해서는 이들에 대한 정보를 저장해 둘 필요가 있다. 이 정보는 Fig. 16에서와 같이 xml의 형태로 저장할 수 있다.

행위자 프로그램이 설치된 디바이스들은 각 행위자 디바이스들의 기능과 상태 정보를 가지고 있다. 이러한 행위자들의 정보는 발견 서비스에 등록되어 조직 프레임워크에서 사용한다. 등록된 정보는 xml 형식으로 Fig. 17과 같이 저장, 관리될 수 있다.

3) 응용의 실행 단계

응용이 실행되면 조직 컨테이너에서 구성된 조직의 역할들이 각각 행위자와 연결된다. 그리고 연결된 행위자에 해당 역할 프로세스를 설치하고 조직자 역할에서 다른 역할들에게 응용의 시작을 알리게 된다.

5.4 응용 프레임워크의 평가

응용 프레임워크의 적용에 대한 평가에는 다양한 평가 지표가 있을 수 있다. 이들은 응용의 개발 단계, 설치 및 배포 단계, 서비스 실행 단계의 세 단계에서 확인해볼 수 있다. 그리고 각 단계별 평가지표에 따라 정적인 컴포넌트기반의 시스템과 ROAD 시스템, 제안한 시스템을 비교 평가한다. 제안한 시스템은 개발 단계에서 디바이스에 독립적으로 개발할 수 있고 ROAD가 조직과 행위자의 통신 과정과 같은 개발 단계의 요소들을 확실히 정립하지 않은 것에 비해 개발 단계의 서버, 행위자 배치 등을 결정한 본 시스템이 개발 시 관심의 분리를 더욱 분명하게 수행한다.

설치 및 배포 단계에서 제안한 시스템은 다른 시스템이 수행할 역할에 대한 프로세스를 모두 설치하여야 하는 점과는 다르게 행위자 프레임워크 모듈만을 설치하여 자신의 기능으로 수행가능한 모든 역할을 맡을 수 있고, 디바이스 정보를 등록하는 서비스를 분리함으로써 조직 서버와 잘 알려진 서버로서 수행되어야 하는 디렉터리 서비스 서버의 배치에 대해 고려하고 있다.

실행 단계에서 제안한 시스템은 조직 구조를 변경하기 위해 조직의 실행을 중단하고 역할을 다시 로드하면서 실행 중 조직 구조 변경을 중단 상태를 거치는 방식으로 지원한다. 그리고 ROAD와 마찬가지로 실행 중 디바이스를 변경

Table 1. Comparison of frameworks

Level	Estimation	Static System	ROAD System	Proposed System
Development	Independence of Device	X	X	O
	Separation of Concerns	X	△	O
Installation&Deployment	Preparing Player Software	O	O	△
	Separation of Device Registration Service	X	X	O
Service Execution	Application(Organization) Structure change in Runtime	X	X	△
	Process Change in Runtime	X	X	O
	Device Change in Runtime	X	O	O
	Spontaneous Decision of Runnable Device	X	X	X

할 수 있고 제안한 시스템은 실행 중 역할의 프로세스를 변경하는 것 또한 가능하다. 하지만 이들 시스템은 모두 자발적인 행위자 연결은 지원하지 않고 있어 이를 통한 적응성 향상은 기대하기 힘들다. Table 1에서는 이러한 평가 결과를 도표로 보여주고 있다.

6. 결론 및 향후 연구

분산 컴퓨팅은 현재 유비쿼터스 컴퓨팅, 자율 컴퓨팅, 클라우드 컴퓨팅 등의 다양한 패러다임으로 연구되고 있다. 이러한 패러다임들은 분산 시스템의 적응성을 중요한 요소로서 사용하고 있다. 이처럼 적응성을 지원하는 다양한 프레임워크와 플랫폼들의 연구가 확산되고 있다[11, 12]. 본 논문에서는 분산 시스템에서의 적응성 확보를 위해 역할 기반의 ROAD 모델에서 역할 프로세스를 정의하고 이를 적용한 적응형 분산 시스템 모델을 구현하기 위한 응용 프레임워크를 제안한다.

본 연구에서 설계한 조직 프레임워크는 이전 연구에서 설계된 행위자 프레임워크와 결합되어 분산 응용 시스템에서 적응형 분산 시스템 모델을 구현하기 위한 응용 프레임워크를 완성한다. 적응형 분산 시스템 모델은 ROAD의 적응성 문제를 최소화하여 ROAD보다 높은 적응성을 보여주고 있다. 그리고 구현 단계를 위한 프레임워크를 개발함으로써 시스템의 실제 구현과 배치 방안에 대해 보여주고 모델의 적용 가능성을 입증한다.

이후 본 연구에서는 조직 프레임워크의 프로토타입을 구현하고 행위자 프레임워크와 결합하여 응용 프레임워크를 완성하고 이를 활용한 실제 분산 응용 시스템을 개발하고자 한다. 향후에는 프레임워크를 다양한 도메인에 적용하여 프레임워크의 유용성을 검증하고 다양한 도메인에 대해 구체적으로 평가하는 것이 필요하다.

참 고 문 헌

[1] Minh H. Tran, Jun Han, "Social Context, Supporting Interaction Awareness in Ubiquitous Environments," *MobiQuitous 6th Annual International*, 2009, pp.1-10.

[2] Xingjun Lin, Yanli Zhu, Anrong Luo, "Towards Developing a Meta-model for Comprehending Software Adaptability," *Mechanic Automation and Control Engineering(MACE)*, October, 2011, pp.2783-2786.

[3] Lawrence Chung, Nary Subramanian, "Process-Oriented Metrics for Software Architecture Adaptability" *Requirements Engineering 2001, Fifth IEEE international Symposium*, October, 2001, pp.310-311.

[4] Alan Colman, "Role-Oriented Adaptive Design," *Swinburne University of Technology*, PhD Thesis, 2006.

[5] Sang-Jun Hong, Ki-Yeol Ryu, Jung-Tae Lee, "Container Framework for Implementation of Role-based Distributed System," *2011 Spring Conference, The Korea Society of Information Technology Applications*, June, 2011, pp.231-236.

[6] Thais Batista, Ackbar Joolia, Geoff Coulson, "Managing Dynamic Reconfiguration in Component-based Systems," *Lecture Notes in Computer Science*, 2005, Vol.3527/2005, pp. 439-480.

[7] Joao Pedro Sousa, David Garlan, "The Aura Software Architecture: an Infrastructure for Ubiquitous Computing," *Lecture Notes in Computer Science, in Carnegie Mellon University*, August, 2003.

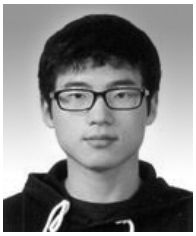
[8] Philippe Collet, Roger Rousseau, "Contracting Hierarchical Components," *I3S Research Report*, Mars. 2004.

[9] Arun Mukhija, Martin Glinz, "The CASA Approach to Autonomic Applications," in *Proceedings of the 5th IEEE Workshop on Applications and Services in Wireless Networks(ASWN 2005)*, Paris, France, June-July. 2005, pp. 173-182.

[10] Tetsuo Tamai, Naoyasu Ubayashi, Ryoichi Ichiyama, "An adaptive object model with dynamic role binding," in *Proceedings of the 27th international conference on Software engineering*, May 15-21, 2005, pp.166-175.

[11] Andreas Frei, Gustavo Alonso, "A Dynamic Lightweight Platform for Ad-hoc Infrastructures," in *Proceedings of the 3rd IEEE Int'l Conference on Pervasive Computing and Communications*, 2005, pp.373-382.

[12] Vaidy Sunderam, Dawid Kurzyniec, "Lightweight Self-organizing Frameworks for Metacomputing," in *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, 2002, pp.113-122.



황 성 윤

e-mail : ctrss@ajou.ac.kr

2011년 아주대학교 정보컴퓨터공학부 (학사)

2011년~현 재 아주대학교 컴퓨터공학과 석사과정

관심분야 : 역할기반 모델, 분산 시스템

정 종 윤

e-mail : jongyun@ajou.ac.kr

2002년 아주대학교 컴퓨터공학과(석사)

2002년~현 재 아주대학교 정보통신전문대학원 박사과정

관심분야 : 유비쿼터스 컴퓨팅, 분산 컴퓨팅, 객체지향 및 관점지향 프로그래밍

이 정 태

e-mail : jungtae@ajou.ac.kr
1979년 서울대학교 농과대학(학사)
1981년 서울대학교 계산학과(이학석사)
1988년 서울대학교 계산학과(공학박사)
1983년～1988년 울산대학교 전산학과 전임강사
1988년～현 재 아주대학교 소프트웨어융합학과 교수
관심분야 : 분산 컴포넌트 시스템, 분산 미들웨어, 객체지향 응용 프레임워크

류 기 열

e-mail : kryu@ajou.ac.kr
1985년 서울대학교 컴퓨터공학과(공학사)
1987년 한국과학기술원 전산학과(공학석사)
1992년 한국과학기술원 전산학과(공학박사)
1993년～1994년 동경대 정산학과 연구원
1994년～현 재 아주대학교 정보컴퓨터 공학과 교수
관심분야 : 분산 시스템, 유비쿼터스 컴퓨팅, 서비스지향 컴퓨팅, 미들웨어, 프로그래밍 언어