모바일 단말기의 가용성을 높이기 위한 자율 관리 시스템의 설계

강준명, 최미정, 박창근, 홍원기
포항공과대학교
{eliot, mjchoi, pck1982, jwkhong}@postech.ac.kr

Design of an Autonomic Management System for Increasing Availability of Mobile Devices

Joon-Myung Kang, Mi-Jung Choi, Chang-Keun Park, and James Won-Ki Hong
POSTECH

요 약

모바일 네트워크와 서비스의 발전으로 인해서 사용자들이 사용하는 모바일 단말기의 기능도 다양해지고 시스템도 복잡해지고 있다. 그러나 사용자는 항상 좀 더 편하고 신뢰성 있는 서비스를 받기를 원한다. 이러한 요구사항을 충족시키기 위해서 나온 것이 모바일 단말 관리 기술이다. OMA (Open Mobile Alliance) DM (Device Management) 작업 그룹 (Working Group)에서는 이러한 요구사항을 해결하기 위해서 단말 관리 기술에 대한 표준을 제정하고 있다. 이를 통해서 단말기의 원격 관리 기술에 대한 것은 많이 연구되고 있지만, 이 방법을 통해서 관리자가 수많은 단말기들을 관리하기에는 비효율적이다. 본 연구에서는 자율 컴퓨팅(Autonomic Computing) 개념을 적용하여 재사용 가능한 단말 관리 컴포넌트를 정의하고 활용한 단말기를 위한 자율 관리 시스템 (Autonomic Management System)인 AutoMO (Autonomic Mobile Device Management System)를 소개한다. 본 논문에서는 AutoMO 의 구조에 대해 설계한 것과 사례 연구로서 소프트웨어 리셋 진단 방법을 적용한 단말기 관리 방법의 예를 통해 설계를 검증한 결과를 제시한다.

I. 서론

차세대 모바일 정보 사회는 언제 어디서나 어떤 단말기로도 네트워크에 접속하여 어떤 서비스라도 받을 수 있는 유비쿼터스 네트워크가 될 것이다. 그리고 현재 우리가 사용하고 있는 IEEE 802.x 계열의 무선 네트워크, 2G/2.5G/3G 의 셀 기반의 네트워크, ZigBee, Bluetooth 등의 근거리 네트워크, 위성을 기반으로 한 네트워크 및 차세대 4G 네트워크까지 다양한 네트워크가 혼재하게 될 것이다. 게다가 네트워크 운영자, 서비스 제공자, 모바일 단말기 제조사 등도 다양해 질 것이다. 단말기도 이미 카메라, MP3 플레이어, 캠코더, 모바일 뱅킹, 네비게이션, DMB (Digital Multimedia Broadcasting) 수신기 등 다양한 기능을 가지고 있고, 차세대 단말기는 부가적인 서비스가 더 추가되고 다양한 네트워크를 액세스할 수 있을 뿐만 아니라 사용자의 요구에 맞게 네트워크와 서비스를 연결시켜주는 기능도 가지게 될 것이다. 이렇게 네트워크, 시스템, 서비스들이 복잡하게 되더라도 사용자의 입장에서는 좀 더 편하고 신뢰성 있는 서비스를 받기를 원한다. 이런 문제를 다루는 기술이 단말 관리 기술이고 이 기술의 표준화를 위해 OMA (Open Mobile Alliance)에서는 DM (Device Management) [1] 작업 그룹을 만들어서 단말 관리를 위한 관리 정보, 관리 프로토콜 및 부트스트랩을 정의하고 있다. 그리고 이를 기반으로 모바일 단말기의 원격 진단을 통한 문제점 해결을 위한 연구도 진행이 되었다 [2, 3]. 그러나 이러한 방법을 통해서는 관리자가 원격에서 단말을 관리하는 것은 가능하지만, 수많은 단말기들을 일괄적으로 관리하기에는 비효율적이다. 관리자 입장에서는 business goal 만을 정해 놓으면 그에 따라 단말 관리가 자동적으로 이루어질 수 있는 방법이 필요하다. 또한 단말기 사용자들도 자신의 단말기에 맞는 policy 만 설정해 놓으면 해당 목적에 맞게 자율적으로 관리가 되는 단말기를 필요로 한다.

본 연구에서는 이러한 단말기 자율 관리 시스템을 위한 프레임워크로서 IBM 에서 제안한 자율 컴퓨팅 개념을 적용하고 재사용 가능한 관리 컴포넌트 기반의 AutoMO (Autonomic Mobile Device Management System)를 제안한다. 관리자는 AutoMO 를 활용해서 자신의 비즈니스 목적만을 정의하여 수많은 모바일 단말기를 효율적으로 관리할 수 있고, 시스템 개발자는 쉽게 단말 관리 시스템을 개발하고 배포할 수 있다. 또한 단말기 사용자도 간단한 사용자 policy 를 단말기에 적용해 사용할 수 있게 된다. 본 논문에서는 AutoMO 의 요구사항 및 전체적인 구성요소를 설명하고, 특히 구조 설계와 이를 토대로 소프트웨어 리셋 진단 사례 연구에 적용하여 설계한 내용을 검증한 것을 제시한다.

본 논문의 구성은 다음과 같다. 2 장에서는 단말 관리 기술 및 시스템에 대한 기존 연구와 자율 네트워크 및 시스템 관리를 관련 연구로서 간략하게 설명한다. 3 장에서는 AutoMO 시스템이 가져야 될 요구사항을 토대로 해서 설계한 내용에 대해서 설명을 한다. 4 장에서는 간단한 단말 관리 예제를 통하여 AutoMO 시스템의 설계를 검증한다. 끝으로 5 장에서는 결론과 향후 연구에 대해서 기술한다.

II. 관련 연구

이번 장에서는 관련 연구로서 자율 네트워크 및 시스템 관리에 대한 기존 연구를 설명한다. 2001 년 IBM 은 인체의 자율신경계(Autonomic Nervous System)를 기반으로 해서 시스템의 복잡성(complexity), 이종성(heterogeneity)

본 연구는 2007 년도 두뇌한국 21 사업과 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음(IITA-2007-(C1090-0701-0045))

및 불확실성(uncertainty)을 해결할 수 있는 방법으로 자율 컴퓨팅 방법을 제안하였다 [4]. 이것은 시스템이 인간의 간섭 없이 스스로 자기를 관리할 수 있다는 자가 관리(self-management) 기능을 가지는 것을 말한다. IBM 은 자가 구성 (self-configuration), 자가 치유(self-healing), 자가 최적화(self-optimization) 및 자가 방어(self-protecting)을 통해서 자가 관리를 할 수 있다고 말한다. 이 개념이 나온 뒤 많은 사람들이 이 방법을 다양한 분야에 적용을 해 오고 있다. [5]에서는 자율 컴퓨팅 환경으로서 Autonomia 를 제시하고 있다. 이것은 소프트웨어 모듈이나 네트워크 자원을 CMI (Component Management Interface)와 CRM (Component Runtime Manager)를 통해서 자율적으로 관리할 수 있게 만들어준다. 여기서 CMI 는 configuration port, control port, operation port 를 통해서 CRM 과 application 자원간의 인터페이스 역할을 담당한다. CRM 은 IBM 에서 제안한 Autonomic Manager (AM) 같이 monitoring, analysis, plan, execution 과정을 통해서 시스템을 자율적으로 동작하게 해 주는 역할을 한다. 이 시스템은 자율 컴퓨팅 환경을 제공하고 있지만, 구체적인 사례에 대한 연구가 부족하다. [6]에서는 분산환경에서 자율 그리드 어플리케이션(autonomic grid application)을 개발하기 위한 방법으로 Automate 를 제안하고 있다. 이것은 프로그래밍 모델, 프레임워크 및 미들웨어 서비스 를 제공하고 있고, 자율 관리 그리드 어플리케이션 의 배포, 개발 및 실행을 가능케 한다. 여기서 주로 한 연구는 자율 요소(Autonomic Element)의 정의, 자율 application 의 동적인 구성, 자율 미들웨어 서비스 에 대한 것이다. [7]에서는 자율 네트워킹 구조 로서 FOCALE (Foundation - Observation - Compare - Act - Learning - Environment)을 제안하고 있다. 이 시스템은 context 정보 및 policy 정보를 이용하여 관리자의 business goal 을 받아들여서 이를 각 vendor 에 맞는 목적까지 연결시키고, 어떻게 knowledge 를 활용할 수 있는지에 대해서 언급하고 있다. 그리고 이를 B3G(Beyond 3G) 네트워크와 Motorola 의 Seamless Mobility 에 적용을 한 사례를 제시하고 있다.

III. AutoMO 의 설계

3.1 요구사항

AutoMO 를 크게 세 부분으로 나누어서 요구사항을 정리할 수 있다. Policy 와 context 정보를 관리하는 지식 관리, 모바일 단말기를 관리하기 위한 모바일 단말 관리 서비스, 그리고 이러한 서비스와 지식을 기반으로 해서 실제 단말 관리를 자율적으로 수행하도록 해 주는 자율 관리 미들웨어이다.

3.1.1 지식 관리

- 관리자나 사용자로부터 business goal 을 입력 받고, 이를 XML 같은 특정한 형태로 언어로 기술하여 특정 repository 에 저장할 수 있어야 한다.
- Business goal 을 만족하기 위한 하위 수준의 policy 를 선택해서 이를 연결시킬 수 있어야 한다.
- 네트워크, 시스템, 사용자로부터 얻을 수 있는 다양한 context 정보를 XML 같은 특정한 형태로 언어로 기술하여 특정 repository 에 저장할 수 있어야 한다.
- 정적인 policy 적용뿐만 아니라 context 에 맞게 동적으로 변화되는 policy 를 적용할 수 있어야 한다.
- Knowledge 정보를 기술하는 언어는 모바일 환경을 고려해서 경량화되어야 한다.

3.1.2 모바일 단말 관리 서비스

- 모바일 단말기에서 관리하고자 하는 다양한 기능들을 네트워크 관리의 기능적 요소인 FCAPS(Fault, Configuration, Accounting, Performance, Security Management) 기반으로 각 서비스들을 분류할 수 있어야 한다.
- 각 서비스나 컴포넌트는 XML 같은 특정한 형태의 텍스트 기반의 언어로 기술되어 쉽게 수정하거나 업데이트가 가능해야 한다.
- 각 서비스나 컴포넌트를 정의하는 방법은 재사용성을 고려하여야 한다.
- 모바일 단말 관리 서비스를 기술하는 언어는 모바일 환경을 고려하여 경량화되어야 한다.

3.1.3 자율 관리 미들웨어

- AM 의 기본적인 동작인 monitoring, analysis, plan, execution 을 통하여 자율 관리가 가능해야 한다.
- AM 은 모바일 단말 전체를 관리하는 최상위층으부터 단말 내부의 컴포넌트 (CPU, 메모리)를 관리하는 최하위까지의 계층적 구조를 가지고 있어야 한다.
- 각 AM 은 관리해야 할 리소스와 관련된 knowledge repository 를 가지고 관리가 이루어져야 한다.
- 미들웨어 시스템은 vendor 독립적으로 구성이 되어야 하고, 독립적인 layer 정의해야 한다.
- 미들웨어 시스템은 모바일 단말기에 탑재되어야 하기에 경량화되어야 한다.

3.2 구조 설계

3.2.1 전체 구조

그림 1 에서 보는 것처럼 AutoMO 는 관리자(administrator)와 시스템 개발자(system developer) 및 사용자(user)에게 다양한 네트워크상에 존재하는 모바일 단말기를 자가 관리할 수 있는 프레임워크를 제공한다. AutoMO 는 Manager-Agent 구조를 따르고, AutoMO Manager 는 다양한 모바일 단말기의 전체적인 관리를 하는 것이고, AutoMO Agent 는 단말기에 탑재되어 단말기 자체를 자가 관리할 수 있는 기능을 한다. AutoMO Manager 와 AutoMO Agent 는 단말 관리 표준 프로토콜인 OMA DM 프로토콜을 이용하여 관리 정보를 주고 받는다.

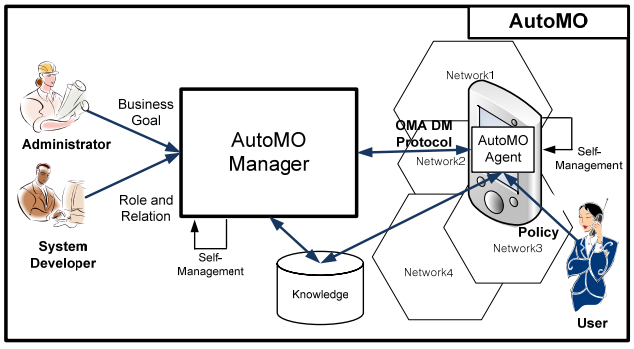

그림 1. AutoMO의 High Level View

3.2.2 AutoMO Manager 의 구조

그림 2 는 앞에서 설명한 요구사항을 토대로 설계한 AutoMO Manager 의 전체 구조를 보여준다. 지식 관리에 대한 요구사항을 만족시키기 위해서 policy 및 context 정보를 저장하는 각각의 Information Base 를 만들었다. 모바일 단말 관리 서비스에 대한 요구사항을 만족시키기 위해서 Role Information Base (ROLIB)을 정의하고 이와 policy 의 연결관계인 relation 도 정의를 하였다. 그리고 끝으로 자율 관리 미들웨어에 대한 요구사항을 만족시키기 위해서 Autonomic Management Middleware (AMM)를 만들고, role 과 policy 들의 relation 으로 이루어진 Autonomic Service (AS)를 관리하는 Autonomic Service Manager (ASM)와 role 을 관리하기 위한 Autonomic Role

Manager (ARM)를 만들었다.

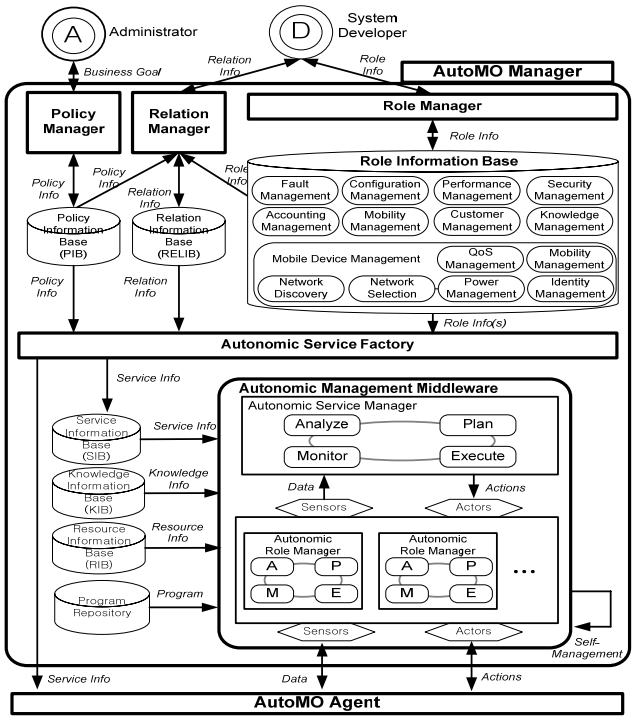

그림 2. AutoMO Manager의 구조

모바일 단말 관리 시스템 개발자는 단말 관리를 위한 role 을 만들고, 필요한 세부 policy 를 정의하고 각 policy 와 role 과의 관계를 정의한 다음에 이 관리 시스템을 배포한다. 단말 관리 서비스를 위한 role 을 정의할 때는 재사용성을 고려한 방법론인 FORM (Feature-Oriented Reuse Method) [8]를 이용하여 재사용가능한 Feature 를 선택하여 role 을 정의하여 재사용성을 높인다. 그 후 시스템 관리자는 관리하고자 하는 business goal 을 Policy Manager (PM)을 통하여 설정하면 Autonomic Service Factory (ASF)에서는 policy, role 및 relation 을 토대로 해서 business goal 을 만족시키기 위한 Autonomic Service (AS)를 만들어서 Autonomic Management Middleware (AMM)에 주어 실행을 하게 한다. 또한 ASF 에서는 단말기에서 실행되어야 할 서비스의 경우에는 AutoMO Agent 에 AS 를 전달해서 실행하게 한다. AMM 에서는 이러한 서비스가 자가 관리를 할 수 있도록 해 주는 ASM 이 있어, 해당 business goal 에 맞게 관리가 진행이 된다. 그리고 이 ASM 은 각 AS 를 구성하는 role 들을 관리하는 ARM 들을 관리하며, 이 ARM 이 모바일 단말기에 있는 AutoMO Agent 를 관리한다. 그리고 ASM 과 ARM 은 자가 관리 정보를 위해서 Knowledge Information Base (KIB), Resource Information Base (RIB) 및 Program Repository (PR)을 이용하게 된다. KIB 에서는 기존에 알려진 다양한 관리 지식에 대한 것 및 네트워크에서 이용할 수 있는 상황 정보들을 받아오고, RIB 에서는 관리해야 할 대상인 모바일 단말기에 대한 관리 요소에 대한 정보를 이용한다. PR 에서는 단말기에서 각 role 을 위해서 필요로 하는 기본적인 program 들이 저장되어 있어 이를 활용할 수 있다. 이런 과정을 토대로 해서 AutoMO Manager 에서는 AutoMO Agent 가 탑재된 모바일 단말 관리에 대한 자가 관리를 제공한다.

3.2.3 AutoMO Agent 의 구조

그림 3 은 앞에서 설명한 요구사항을 토대로 해서 설계한 AutoMO Agent 의 전체 구조를 보여준다. AutoMO Agent 는 사용자로부터 기본적인 user policy 를 PM 을 통해서 받을 수 있고, 이를 User Policy Information Base (UPIB)에 저장한다. 그리고 AutoMO Manager 에서 관리를 위해 제공된 AS 를 SIB 에 저장하고 AMM 에서는 이를 ASM 과 ARM 을 통해서 실행을 하게 된다. AutoMO Manager 와 마찬가지로 AutoMO Agent 에서도 ASM 이 전체 service 에 대한 관리를 담당하고, 각 service 를 위한 role 들에 맞게 ARM 이 관리를 수행한다. AutoMO Agent 는 단말기에 들어가는 모듈이기 때문에 실제 관리 대상이 단말기의 각 관리 요소들이 된다. ARM 은 관리정보를 모니터링하고 이를 분석하고 해결하기 위한 action 을 만들어서 수행한다. 그리고 각 ARM 은 자신이 관리해야 할 것을 직접 관리하되 해결하지 못하는 것은 상위의 ARM 이나 ASM 에게 결정을 맡긴다. 궁극적으로는 단말기 자체에 있는 AutoMO Agent 가 관리를 수행하여 단말기가 자가관리하는 것이 목적이지만, AutoMO Agent 의 ASM 이 해결하지 못하는 문제에 대해서는 AutoMO Manager 에게 정보를 전달해서 결정을 맡긴다. 그리고 최하단의 ARM 은 단말기의 관리요소를 접근할 때 Resource Manager 를 통해서 접근하고, 특정 system API 에 의존적이지 않은 일반적인 API 를 Porting Layer (PL)에 정의를 하여 사용한다.

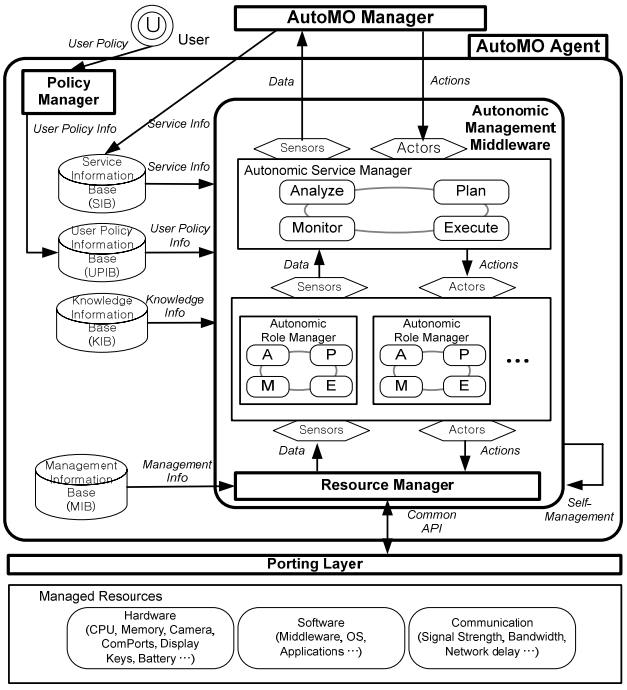

그림 3. AutoMO Agent의 구조

IV. 사례 연구 및 검증

3 장에서 설계한 AutoMO 를 적용하여 기존의 모바일 단말 관리 기능 중 소프트웨어 리셋 진단을 통해서 설계의 실효성을 검증하고자 한다. [2]에서 모바일 단말기가 갑작스럽게 재부팅하거나 갑자기 멈추는 등의 문제가 발생했을 때 이를 소프트웨어 리셋 문제라고 정의하였다. 그리고 이 문제를 해결하기 위해서 필요한 관리 정보를 정의하고 세가지 단계의 동작을 정의하여 원격에서 문제점을 진단할 수 있는 방법을 제시하였다. 여기서 사용된 예를 AutoMO 를 이용하여 개선할 수 있는 방법을 설명한다.

(Business Goal) 소프트웨어 리셋 문제를 해결해야 한다.

- (P1) 소프트웨어 리셋을 감지
  - (P1-1) 정상적인 재부팅이 아닌 경우 감지
  - (P1-2) 시스템이 아무런 동작을 하지 않고 멈춘 경우 (Halting Problem, Freezing Problem) 감지
- (P2) 문제점 해결

- (P2-1) 문제점을 분석한 뒤 관련 패치가 있으면 이를 비교하여 적용하고 그렇지 않으면 이 정보를 단말 관리 서버로 전송

(Autonomic Service) 소프트웨어 리셋 문제 해결 service

- (R1) 소프트웨어 리셋 감지 role
  - (R1-1) 정상적인 재부팅이 아닌 상황 감지 role
  - (R1-2) 시스템이 아무런 동작을 하지 않고 멈춘 경우 감지 role
- (R2) 관리 정보 관련 role
  - (R2-1) 관리 정보를 모바일 단말기에 추가하는 role
  - (R2-2) 관리 정보를 모바일 단말기에 실행하는 role
  - (R2-3) 관리 정보(Register, Stack, UIPrimitive, KeyEvent, ScenCb, ExitCode)를 모니터링 하는 Role
  - (R2-4) 관리 정보를 덤프하여 저장하는 role
  - (R2-5) 패치를 적용하는 role
  - (R2-6) 관리 정보를 서버로 전송하는 role

Policy P1 에 의해서 role R1 이 선택되고, policy P2 에 의해서 role R2 가 선택된다.

이렇게 policy, relation 및 role 들이 정해지고 나면 ASF 에서는 이를 이용하여 하나의 AS 를 만든다. AS 에서는 policy 정보, role 정보 및 relation 정보들을 가지고 있고, relation 정보를 토대로 해서 각 role 과 policy 가 적용되는 순서가 명세된다. 이 경우에는 그림 4 와 같이 각각의 role 과 policy 의 순서가 결정되고, 이 순서에 따라서 실행하면서 자가 관리를 하게 된다. 이렇게 만들어진 AS 는 단말기 내에서 모든 관련 role 들이 실행되어야 하기에 AutoMO Agent 에 있는 AMM 의 SIB 에 저장이 되고 AMM 은 새로운 AS 에 대해서 ASM 에게 알려주고 ASM 은 하나의 서비스에 대해 관리를 할 수 있게 로딩된다. 앞에서 설명한 것처럼 각 role 들마다 이를 관리하기 위한 ARM 이 만들어지고, 이 ARM 은 자신에 맞는 관리 정보를 4 단계의 life cycle 을 통해서 관리하게 된다.

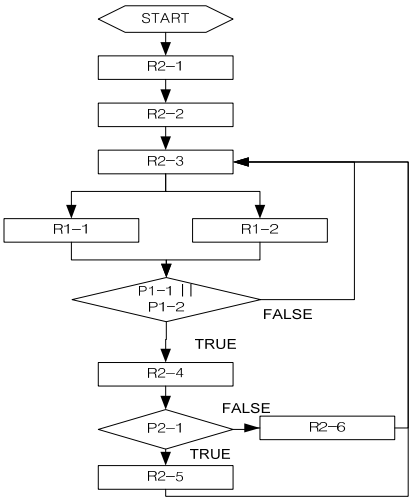

그림 4. 소프트웨어 리셋 진단의 순서도

예를 들어, 소프트웨어 리셋 중에서 비정상적인 재부팅을 감지하는 것 같은 경우인 R1-1 에는 다음과 같이 총 4 가지의 life cycle 을 가진다.

- Monitor: R2-3 을 관리대상으로 보고 monitoring 을 수행한다. R2-3 ARM 에서는 필요한 관리 정보를 읽어서 R1-1 의 ARM 의 Sensor 를 통해서 보고하게 된다.
- Analyze: 읽어들인 정보를 통해서 P1-1 을 만족하는지를 판단하기 위하여 KIB 에 저장된 context 정보를 이용하게 된다. 예를 들어서 정상적인 재부팅이 아닌 경우에는 시스템의 부팅 플래그를 이용한다. 이것이 1 이면 정상적인 부팅이고, 0 이면 비정상적인 부팅이다라는 정보를 통해서 비정상적인 재부팅 상황을 판단할 수 있다.
- Plan: 정상적인 부팅이라고 판단이 되면 R2-3 에게 이전 정보는 정상적인 정보이니 계속 더 모니터링을 수행하라고 판단을 내리게 되고, 비정상적인 재부팅이라면 이 사실을 R2-4 의 ARM 에게 알리게 된다.
- Execute: Plan 에서 결정된 사항에 대해서 해당 Action 을 R2-3 의 ARM 에게 Actor 를 통해서 수행을 하게 된다.

위와 같은 과정을 거쳐서 각 ARM 은 스스로 맞는 role 을 관리하면서 전체 business goal 을 만족하기 위한 AS 를 수행하게 된다. 앞으로 이 부분을 더 구현하여 이전의 수동적으로 진단하는 것과의 성능 비교를 해서 효율성을 검증할 것이다.

V. 결론

네트워크와 서비스의 발전으로 점점 더 모바일 단말기가 복잡해지고 있으며, 모바일 단말기를 통해 제공되는 서비스도 다양해지고 있다. 사용자는 또한 항상 편리하게 서비스를 받기를 원한다. 그리고 관리자도 다양하고 많은 모바일 단말기를 좀 더 효율적으로 관리하기를 원한다. 이를 만족시키기 위해서는 최상위의 business goal 만 설정하고 나머지 관리는 자율적으로 이루어지는 방법을 필요로 한다. 본 논문에서는 이러한 자율적인 모바일 단말 관리 시스템을 쉽게 개발하고 관리자에게 편리함을 제공해줄 뿐만 아니라 사용자에게도 쉽게 모바일 단말기를 관리할 수 있게 해 주는 프레임워크로 AutoMO 를 제안하고 그 설계 내용에 대해서 설명을 하였다. 그리고 단말기의 소프트웨어 리셋 진단 방법을 사례 연구로 제시하여 그 실효성을 설명하였다.

향후 연구에서는 policy, relation, context 등의 Information Base 에 들어갈 형식을 XML 로 정의하고 단말 관리 서비스의 형식 정의 및 필요한 단말 관리 서비스를 추출하는 작업을 할 것이다. 이를 토대로 해서 제시한 프레임워크에 맞는 시스템을 구현할 것이다.

참고문헌

[1] OMA DM (Device Management) Working Group, http://www.openmobilealliance.org/tech/wg_committees/dm.html.

[2] Joon-Myung Kang *et al*., "OMA DM Based Remote Software Debugging of Mobile Devices," Accepted to appear APNOMS 2007, LNCS 4773, Sapporo, Hokkaido, Japan, Oct. 2007, pp. 51-61.

[3] Joon-Myung Kang *et al*., "OMA DM Based Remote RF Signal Monitoring of Mobile Devices for QoS Improvement," Accepted to appear MMNS 2007, LNCS 4787, San Jose, CA, USA, Oct. 2007, to pp. 76 – 87.

[4] IBM Corporation: An architectural blueprint for autonomic computing. White Paper, (2003).

[5] Salim Hariri, S. *et al.* "AUTONOMIA: an autonomic computing environment," Proceedings of the 2003 IEEE International Conference on Performance, Computing, and Communications (IPCCC 2003), Phoenix, Arizona, USA, April 2003, pp. 61-68.

[6] M. Agrawal *et al*., "Automate: Enabling Autonomic Applications On The Grid", Proceedings of Active Middleware Services (AMS) 2003, Seattle, WA, June 2003, pp. 48-57.

[7] Strassner J., Agoulmine N., Lehtihet E.: "FOCALE: A Novel Autonomic Networking Architecture," in Latin American Autonomic Computing Symposium (LAACS), July 18-19, 2006, Campo Grande, MS, Brazil.

[8] Kyo C, Kang *et al*., "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architecture," Annals of Software Engineering, May 1998, pp. 143-168.