정책 기반 적응형 어플리케이션을 위한 프레임워크 분석

김소영°문미경 염근혁
부산대학교 컴퓨터 공학과
소프트웨어 공학 연구실
{yangunim°,mkmoon,yeom}@pusan.ac.kr

Analysis of the Framework for the Policy-Based Adaptive Applications

Kim, So Young° Mikyeong Moon Keunhyuk Yeom
Computer Engineering Department, Pusan National University

요 약

유비쿼터스 소프트웨어 개발에 대한 연구가 활발히 진행되고 있는 가운데 여러 개발 프레임워크가 제안되고 있다. 유비쿼터스 환경에서는 문맥 정보가 중요하게 인식되고 있고 문맥정보를 제공하는 자원은 개발 도메인에 따라 다양하다. 또한 유비쿼터스내의 여러 요소들은 동적인 특성을 가지고 있다. 이러한 유비쿼터스 환경에 맞는 소프트웨어를 프레임워크의 도움 없이 개발하는 데는 많은 시간과 비용이 든다. 따라서 본 논문에서는 일반적인 유비쿼터스 환경에서 동적으로 적응 가능하도록 하는 정책 기반 적응형 어플리케이션을 위한 프레임워크를 분석,제시한다.

1. 서론

유비쿼터스 컴퓨팅 소프트웨어에 대한 관심이 많아 지면서 문맥(context)에 기반한 적응형 소프트웨어를 만들기 위한 연구가 많아지고 있다[1]. 문맥은 사용자가 어디에 있는가(where), 사용자의 주위에 무엇이 있는가(what), 사용자의 주위에 어떤 자원들이 있는가(resources)등의 정보를 표현한 것이다[2]. 이러한 정보는 외부 상황을 표현하는 기본적인 것으로 적응형 소프트웨어는 이러한 문맥에 따라 동적으로 자신의 내부 컴퍼넌트 구성을 변화 시켜 원하는 서비스를 제공한다. 그러나 아직까지는 기능적인 동적 변화를 구현하기에는 많은 어려움이 따르고 있다[7]. 따라서, 비기능적인 측면으로 서비스를 정의하는 방식의 접근이 적절하고 보인다. 문맥을 표현하는 방식은 매우 다양하며 다양한 자원이 문맥정보로 이용된다. 이러한 문맥 정보를 나타내는 자원을 이용하는 주체에 대하여 제약을 둘 수가 있는데 이러한 방법의 하나로 정책이 있다. 정책은 시스템 동작의 선택을 제어하는 규칙으로 일회성이 아닌 영구적 속성을 가지는 규칙이며 사용자에 의해 동적으로 갱신되며 미리 정의된 기능이나 동작을 실행하기 위한 제약이다[4]. 이러한 본 프레임워크에서는 정책이 영향을 주게 되는 영역을 데이터 스페이스(data space)이라 정의하고, 이러한 데이터 스페이스 내부에 사용자, 어플리케이션, 센서 등 어떠한 요소들이 존재하게 된다. 이러한 요소들의 속성과 기능을 이용하여 응용어플리케이션은 자신이 필요로 하는 서비스를 정의하게 되고, 이러한 서비스들은 결과적으로 환경정보 즉, 정책에 영향을 받게 된다. SOA(Service-Oriented Architecture) 단위로 구성되는 서비스들은 정책에 따라 내부적으로 서비스의 품질이 바뀌게 된다. 이러한 접근 방식은 효과적으로 외부 환경에 적응하는 소프트웨어를 만드는 프레임워크를 가능하게 한다.

2. 관련 연구

적응형 소프트웨어는 자신의 기능을 평가하고 추론하고 환경 정보를 가지고 실행시간에 상황에 적응하는 소프트웨어이다[7]. 적응 프로세스는 주위 환경이나 사용자의 필요에 따라 보다 나은 서비스를 제공해 준다. 이러한 종류의 적응형 소프트웨어를 만들 수 있는 기능을 제공하는 미들웨어에 대한 프로젝트는 GARF, CodA, mChaRM, Open ORB 등이 있다[7].

정책은 자원에 대한 접근과 이벤트에 대한 동작을 기술하기 위해 쓰인다[4]. [4]에서 정의한 정책은 크게 두 가지로 나뉘는데, Authorization 정책은 어떠한 주체에 대한 행위에 대한 권한을 모델링 한 것이고, obligation 정책은 여러 목적물에 대해 어떠한 행위가 가능한 것인지에 대한 것을 모델링한다. 이러한 모델은 쉽게 자원에 대한 권한 부여, 보안 설정 등에 대한 표현과 룰(rule)에 대한 표현을 가능하게 한다.

기존 네트워크상의 자원을 동적으로 관리하는 방식은 각 어플리케이션이 관심을 가지는 자원, 즉 대역폭 같은 요소에 따라 자신을 적응시켜왔다. 그러나 이러한 방식은 하나의 자원에 대해 여러 어플리케이션이 동시에 자신을 적응시키게 되어 결과적으로 자원을 효과적으로 이용할 수 없게 된다[5]. 또한 각 어플리케이션마다 환경에 따라 적응하는 메커니즘이 다른 어플리케이션과의 충돌성,의존성등을 따지지 못하는 문제점이 있다.

본 논문에서 제시하는 정책기반 적응형 어플리케이션을 위한 프레임워크는 엔티티가 자신의 데이터 스페이스 공간의 정책에 따라 자원을 접근하게 만들며, 자원의 공유를 가능하게 하고 어플리케이션의 적응 메커니즘을 제어하여 유비쿼터스 환경의 동적 환경에 적응하여 동작하는 서비스 지향 프레임워크에 대한 분석과 그 결과를 제시한다.

3. 프레임워크 분석

적응형 소프트웨어 아키텍쳐의 요구사항을 바탕으로 프레임워크를 정적인 관점과 동적인 관점에서 분석하도록 하겠다.

3.1 PBA(Policy-Based Adaptation Application) 프레임워크의 정적 특성 분석

적응형 소프트웨어에서 필요로 하는 기능으로는 다음과 같다[1].

- 소프트웨어 내부 구조 재구성
- 물리적 환경으로부터 독립적인 요소들의 논리적 추상화
- 문맥 표현
- 실시간 문맥 정보 분석 및 평가
- 시스템 동작 방향 결정 메커니즘

<그림 1>은 이러한 요구사항을 반영하여 정책 기반 적응형 소프트웨어를 위한 메타모델을 분석했다. 각 구성 요소에 대해서 아래에서 설명한다

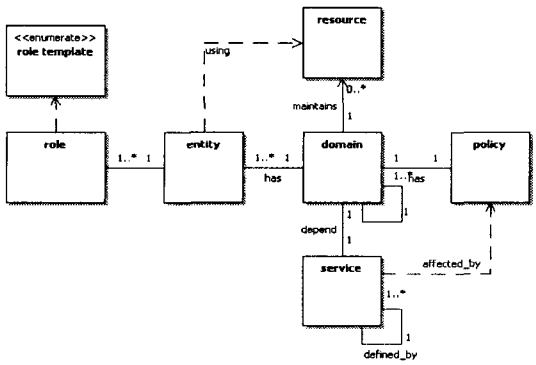

그림 1. PBA 의 메타모델

3.1.1 PBA 프레임워크 메타 모델

<그림 1>은 기능적 요구사항으로부터 도출된 각 요소들과 그 요소들간의 관계를 나타낸다. 각 요소의 내용은 아래와 같다.

3.1.2 프레임워크 요소들

- 엔티티 (entity)

엔티티는 물리적인 센서노드나 컴퓨팅 장치, 인간, 외부 시스템등 "*어떠한 기능을 제공해주는 능력을 가진 것들*" 을 통칭한다. 이러한 요소들은 기능에 따라 수동적, 능동적 요소로 나뉘게 된다. 수동적 요소는 주로 능동적 요소에 의해 쓰이는 자원의 일종으로 센서나 데이터베이스등이 있다. 능동적 요소에는 수동적 요소를 이용하는 사람이나 외부 컴퓨팅 장치, 데이터 수집기등이 있다.

- 역할 (role)

여기서 역할은 어떠한 에이전트(agent)가 가지게 되는 표준적인 동작 레퍼토리(repertory)를 말한다[6]. 유비쿼터스 환경 내에서의 장치들인 엔티티들은 각자 속성과 기능을 가지고 있고 이러한 성질에 따라 각 엔티티들은 특정 역할을 가지게 된다. 역할에 따라 엔티티들은 데이터 스페이스 내의 다른 엔티티-자원,DB 등에 대한 접근권한과 제약 조건을 가지게 된다. 역할은 또한 엔티티들을 계층화 시키고 ID 를 부여하는 등의 기능을 하게된다.

- 데이터 스페이스(data space)

엔티티들은 역할을 가지고 어떠한 영역에서 자신의 기능을 취하게 된다. 엔티티들이 위치하고 자신의 기능을 수행하며 다른 엔티티와 상호작용하는 영역을 데이터 스페이스라고 정의한다. 데이터 스페이스는 엔티티의 멤버쉽이나 유입/유출, 엔티티의 계층화, 엔티티 리스트등을 관리한다. 데이터 스페이스 공간은 크게 Default 스페이스, sub 스페이스 그리고 primitive 스페이스로 나뉜다. Default 스페이스는 모든 요소를 포함하고 있는 영역으로 기본 정책을 제공하며 엔티티의 이동성을 추적한다. 또한 하위 스페이스들 간의 정책 충돌 제어하며 하위 스페이스들을 관리한다. Sub 스페이스은 default 스페이스 내부에 있는 스페이스로 또 다른 sub 스페이스를 포함할 수 있다. Primitive 스페이스는 내부적으로 더 이상 스페이스를 중첩 시킬 수 없는 스페이스이다.

- 자원(resource)

자원은 엔티티들 중에 하나로 Sensor 나 DadaSource 와 같은 역할을 가지고 있으며 주로 수동적인 속성을 가지고 있다. 능동적인 엔티티들은 주로 RFID 센서같이 유동적이며 정보를 수집하는 장치나 사람과 같이 자신의 상태정보를 갱신하여 다른 자원에 의해 쓰이는 것들을 말한다. 자원에 대한 정보는 계속적으로 갱신되는 센서정보들 같은 경우가 있으며 한번 지정되면 바뀌지 않는 데이터베이스나 환경변수등과 같은 정적인 정보도 있다.

- 정책(policy)

데이터 스페이스내의 엔티티들을 관리하는 기법에는 여러 가지가 있다. 그 중 하나인 정책 기반 관리기법은 시스템관리를 위한 접근법으로 최근에 많은 관심을 받고 있다. 왜냐하면 이 기법은 시스템 동작 선택을 제어하는 룰(rule)과 그 시스템이 제공하는 기능을 분리하기 때문이다[4]. 데이터 스페이스 내에서는 엔티티들을 관리하고 자원에 대한 접근성을 다루기 위한 방법으로 정책을 이용한다. 이러한 정책은 각 데이터 스페이스에 대해 하나씩 가지게 되며 해당 데이터 스페이스에 속한 엔티티들은 그 데이터 스페이스의 정책에 영향을 받게 된다. 정책은 데이터 스페이스가 관리하는 자원에 대한 제약사항을 정의하고, 역할 할당에 대한 제약 사항을 기술하며 primitive 서비스에 대한 정의를 제약한다.

- 서비스(service)

서비스는 다양한 엔티티들의 역할과 데이터 스페이스의 정책, 자원 그리고 데이터 스페이스의 기능과 정보를 이용하여 정의된다. 서비스는 SOA 형식으로 어플리케이션에 의해 정의되며 프레임워크 내에 저장된다. 서비스의 종류는 Primitive 서비스와 Derivative 서비스가 있다. Primitive 서비스는 Derivative 서비스의 하부 구조로 Derivative 서비스를 정의하기 위한 문법과 용어를 정의하며 최소단위의 서비스를 제공한다. 이러한 primitive 서비스는 요소의 역할과 자원을 이용하여 정의된다. 그리고 Derivative 서비스는 Primitive 서비스의 기능을 이용하여 정의되는 서비스로 주로 어플리케이션에 의해서 정의된다. 이러한 서비스는 주로 외부 사용자나 어플리케이션,엔티티들에 의해서 쓰이게 된다. 또한 다른 서비스를 이용하기도 하며 다른 서비스에 대한 결과를 이용하기도 한다. 이러한 서비스들은 최종적으로 외부 어플리케이션에 SOA 형태로 제공되며 플랫폼에 독립적인이게 여러 어플리케이션에게 서비스를 제공할 수 있다.

3.2 PBA 프레임워크의 동적 특성 분석

- 엔티티의 유동성

데이터 스페이스 내부에 존재하게 되는 엔티티들은 상황에 따라 다른 데이터 스페이스로 이동을 하게 된다. <그림 2>와 같이 역할을 부여 받은 엔티티가 다른 데이터 스페이스로 유입,유출 될 수 있다. Default 데이터 스페이스는 엔티티가 기본적으로

존재하는 공간인데 다른 데이터 스페이스로 유입됨으로써 엔티티는 다른 역할을 가질 수 있다. 엔티티가 사람인 경우 회사에서 엔티티의 역할은 메니저가 되지만 집이라는 데이터 스페이스에 오게 되면 가장이라는 역할로 바뀔 수 있다. 이러한 역할의 변화는 데이터 스페이스의 정보와 자원에 대한 접근성을 변화 시켜 자신의 기능을 한정시키게 된다.

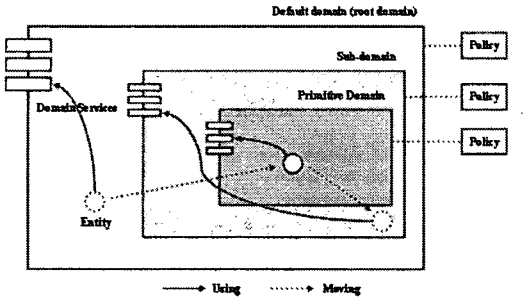

그림 2. 디폴트 데이터 스페이스와 하위 데이터 스페이스

데이터 스페이스의 엔티티 추적(traceability)

유비쿼터스에서 엔티티들은 유동성을 가지고 있다. 특히 능동적인 엔티티들이 이러한 속성을 많이 띠고 있다. 추적성은 이러한 엔티티들이 이동하는 경로인 데이터 스페이스 히스토리(domain history)를 관리하는 것을 말한다. 데이터 스페이스 추적성을 지원하는 것은 Default 데이터 스페이스가 맡게 된다. 엔티티들은 항상 일정한 시간 간격으로 자신의 데이터 스페이스 정보를 요청하게 된다. 요청은 자신의 절대적 위치 정보와 시간정보를 함께 보낸다. 이러한 요청은 Default 데이터 스페이스로 가게 되고 Default 데이터 스페이스는 자신이 관리하고 있는 데이터 스페이스들에게 요청을 전달하게 된다. 요청을 받은 데이터 스페이스들은 요청을 한 엔티티의 위치가 자신이 관리하는 위치내에 있는지 검사한다. 데이터 스페이스는 자신의 영역내에 해당 엔티티가 있으면 그 정보를 default 데이터 스페이스로 전달하게 되며 default 데이터 스페이스는 이러한 정보를 바탕으로 엔티티들의 데이터 스페이스 히스토리를 가지게 된다.

정책의 가변성(variation of policy)

유동성을 가진 엔티티는 여러 데이터 스페이스에 유입,유출되므로 엔티티에 영향을 주는 정책이 가변적이다. 따라서 해당 엔티티에 영향을 줄 수가 있다. 그와 반해 유동성이 없는 엔티티들은 정책이 가변적이지 않아 그 엔티티가 제공하는 기능에 크게 영향을 주지 않는다. 또한 각 엔티티들의 역할은 변동성(transient)을 가지게 되는데, 유동성을 지닌 엔티티는 역할의 변화가 많지만 비유동성을 가진 엔티티는 역할의 변화가 거의 없다. 하지만 비유동성을 지닌 엔티티의 기능의 많고 적음에 따라 비록 유동성이 없다 하더라도 엔티티의 역할에 대한 가변성은 많다고 할 수 있다. 따라서 유동적인 엔티티와 비유동적인 엔티티의 역할의 가변성은 정책의 변화를 겪게 된다.

3.3 PBA 프레임워크를 이용한 가변적 서비스 정의

서비스 품질 레벨(quality level of service)

엔티티들의 기능과 속성으로 Primitive 서비스가 만들어지고 어플리케이션에 의해서 derivative 서비스가 정의된다. Primitive 서비스의 이용은 정책에 의해 결정되며 이러한 Primitive 서비스의 선택에 의해 어플리케이션이 사용하는 Derivative 서비스의 내부 품질이 가변적이게 된다. 엔티티인 품질은 서비스의 응답속도, 결과의 정확성, 서비스의 실행 실패율등이 있는데 이러한 요소들에 대한 제약을 정책이 하게되는 것이다. 따라서 Primitive 서비스의 재구성이 일어나게 되고 이로 인해 서비스의 가변성이 생기게 된다. 예를 들어 사용자가 A 데이터 스페이스에서 주로 쓰는 서비스 " *방을 밝게 하라* " 는 " *전기절약* " 을 위한 정책이 해당 데이터 스페이스에 반영되어 있을 수 있다. 이런 경우에는 비록 원하고자 하는 서비스의 품질을 받을 수 없게 되지만 사용자는 낮은 품질의 서비스를 얻게 되어 일반적으로 원하는 기능을 수행할 수 있게 된다.

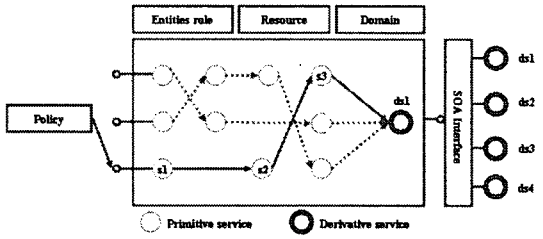

그림 3.정책에 따른 서비스 품질의 가변성

4. 결론 및 향후 연구

본 논문에서는 유비쿼터스 환경에서 동적으로 외부 환경에 적응하는 어플리케이션을 위한 정책 기반 적응형 어플리케이션의 프레임워크를 제시하였다. 프레임워크내에 정의한 서비스는 웹서비스 기술로 정의되어 제공되어 외부 어플리케이션에서 쉽게 이용될 수 있다. 향후 연구에서는 보다 향상된 서비스 조합과 서비스 평가에 대한 연구가 필요하다고 보인다.

5. 참고문헌

[1] Keith Mitchell, *"A Survey of Context-aware computing"*, Internal Technical Report, Dr. Keith Mitchell, Research Associate Distributed Multimedia Research Group, March 2002, U.K.

[2] Dey, A., Abowd, G, *"Towards a better understanding of context and context-awareness"*, GVU Technical report GIT-GVU-00-18, Graphics visualization and usability center, Georgia Institute of Technology, 1999.

[3] L. Lymberopoulos, E. Lupu and M. Sloman, *"An adaptive Policy Based Management Framework for Differentiated Services Networks"*, IEEE Third International Workshop on Policies for Distributed Systems and Networks, Monterey, California June 5--7 2002.

[4] Emil Lupu,Morris Sloman, *"A policy based role object model"*, E DOC'97, October 24,1997, Australlia

[5] C. Efstratiou, K. Cheverst, N. Davies and A. Friday, *"An Architecture for the Support of Adaptive Context-Aware Applications"*, Proceedings of Mobile Data Management (MDM`01), Hong Kong, January, 2001.

[6] James Odell, H. Van Dyke Parunak, and Mitch Fleischer, *"The Role of Roles in Designing Effective Agent Organizations"*, Software Engineering for Large-Scale MultiAgent Systems, Alessandro Garcia et al, LNCS, Springer, 2003.

[7] Qusay H. Mahmoud, "Middleware for Communications",Wiley, 2005