상황 인식 기반의 유비쿼터스 컴퓨팅을 위한 접근 제어 시스템

이 지 연[†] · 창 병 모[††] · 안 준 선[†††] · 도 경 구[††††]

요 약

다양한 모바일 기기들이나 무선 네트워크들에 의한 무분별한 자원 접근은 시스템에 문제를 일으킬 수 있으므로 접근 권한 관리는 매우 중요하다. 본 논문에서는 프로그래머가 각 응용 프로그램에 맞는 접근 권한 규칙을 정책 파일로 작성하고 이를 실행시키는 접근 제어 시스템을 구현하였다. 본 논문에서 구현된 접근 제어 시스템인 CACM(Context-awareness Access Control Manager)은 상황 인식 기반의 유비쿼터스 컴퓨팅을 위한 프레임워크인 JCAF을 바탕으로 구현하였다. CACM은 프로그래머가 작성한 정책 파일을 바탕으로 접근을 제어한다. 또한 본 논문에서는 정책 파일을 정적 분석하여 잘못된 정책 파일 규칙 알려주는 지원 시스템을 제공하며 본 시스템을 사용하여 개발된 유비쿼터스 응용 프로그램의 실행을 시뮬레이션 할 수 있는 시뮬레이터와 시뮬레이션 결과를 제공한다.

키워드 : 접근 제어, 정책 기술 언어, 정책 파일, 상황 인식, JCAF, CACM

An Access Control System for Ubiquitous Computing based on Context Awareness

Jiyeon Lee[†] · Byeong-Mo Chang[††] · Joonseon Ahn[†††] · Kyung-Goo Doh[††††]

ABSTRACT

It is important to manage access control for secure ubiquitous applications. In this paper, we present an access-control system for executing policy file which includes access control rules. We implemented Context-aware Access Control Manager(CACM) based on Java Context-Awareness Framework(JCAF) which provides infrastructure and API for creating context-aware applications. CACM controls accesses to method call based on the access control rules in the policy file. We also implemented a support tool to help programmers modify incorrect access control rules using static analysis information, and a simulator for simulating ubiquitous applications. We describe simulation results for several ubiquitous applications.

Keyword : Access Control, Policy Description Language, Policy File, Context Awareness, JCAF, CACM

1. 서 론

무선 네트워크와 모바일 기기들의 발전과 더불어 이를 통한 다양한 자원 접근이 가능해짐에 따라 안전한 유비쿼터스 응용프로그램을 위한 자원 접근 제어는 필수적이라 할 수 있다.

본 연구는 상황 인식 기술을 기반으로, 실생활에서 발생할 수 있는 다양한 상황들을 반영하여 정확하면서도 유연하게 접근 제어함을 목적으로 한다. 이를 위해 본 논문에서는 프로그래머에 의해 작성된 접근 제어 정책 파일에 따라 유비쿼터스 응용 프로그램에 대한 정확하고 유연한 접근 제어가 가능하도록 접근제어 시스템을 개발하였다.

본 논문에서는 프로그래머가 접근 제어를 위한 정책들을 정책 파일에 기술한다. 정책 파일은 정책 기술 언어(Policy Description Language)[1]를 통해 기술하는데 이 언어는 쉽고 명확하게 접근 제어 정책을 정의할 수 있도록 설계된 언어이며 비교적 간단한 문법을 가지고 있으면서도 현실 세계 상황을 자연스럽게 표현할 수 있다.

본 논문에서는 JCAF를 기반으로 접근 제어를 수행하는 접근제어기 CACM (Context-aware Access Control Manager)을 설계 구현하였다. 이 시스템은 상황 인식 기반의 유비쿼터스 컴퓨팅을 위한 프레임워크인 JCAF(Java Context Awareness Framework)[2]를 바탕으로 구현하였다. CACM은

※ 본 연구는 한국과학재단 특정기초연구(R01-2006-000-10926-0) 지원으로 수행되었음.
† 정 회 원 : UC Berkeley, Traffic Safety Center 방문연구원
†† 정 회 원 : 숙명여자대학교 컴퓨터과학과 교수(교신저자)
††† 정 회 원 : 항공대학교 항공전자및정보통신공학부 교수
†††† 정 회 원 : 한양대학교 컴퓨터공학과 교수
논문접수 : 2007년 1월 12일, 심사완료 : 2008년 1월 8일

DOI: 10.3745/KIPSTA.2008.15-A.1.35

프로그래머가 작성한 정책 파일이 실제로 제대로 수행될 수 있도록 해준다. CACM은 정책 파일을 분석하여 해시테이블 형태로 저장한 뒤, 응용프로그램으로부터 권한 검사 요청이 있을 때마다 상황정보를 비교하여 그 결과를 반환한다.

본 연구에서는 정적 분석을 통해 프로그래머가 정책 파일을 보다 효과적으로 작성될 수 있도록 지원하는 분석도구를 제공한다. 이 분석도구는 실행 전에 정적 분석을 통해 누락된 접근 제어 규칙과 잘못된 권한 검사 요청을 찾아줌으로써 프로그래머가 쉽게 정책 파일과 응용프로그램을 수정할 수 있도록 도와줄 수 있다. 또한 본 연구에서는, 본 시스템을 기반으로 개발된 유비쿼터스 응용프로그램을 시뮬레이션할 수 있는 시뮬레이터를 구현하였고, 다양한 유비쿼터스 응용프로그램과 정책 파일을 개발하여 시뮬레이션한 실행 결과를 제공한다.

본 논문의 구성은 다음과 같다. 2절에서는 관련 연구들을 소개한다. 3절에서는 정책 기술 언어를 설명하고 4절에서는 접근 제어 정책들의 실행을 위한 시스템의 설계 및 구현에 대해 기술한다. 5절에서는 정적 분석 도구에 대해서 기술한다. 6절에서는 유비쿼터스 응용프로그램을 시뮬레이터를 통해 실행한 결과에 대해 소개하고 마지막으로 본 논문의 결론을 맺는다.

2. 관련 연구

2.1. 유비쿼터스 컴퓨팅 환경의 접근 제어 연구

지금까지 접근 제어를 위한 유비쿼터스 컴퓨팅 환경에 관한 몇몇 연구들이 진행되어 왔다 [2, 3, 4, 5, 6].

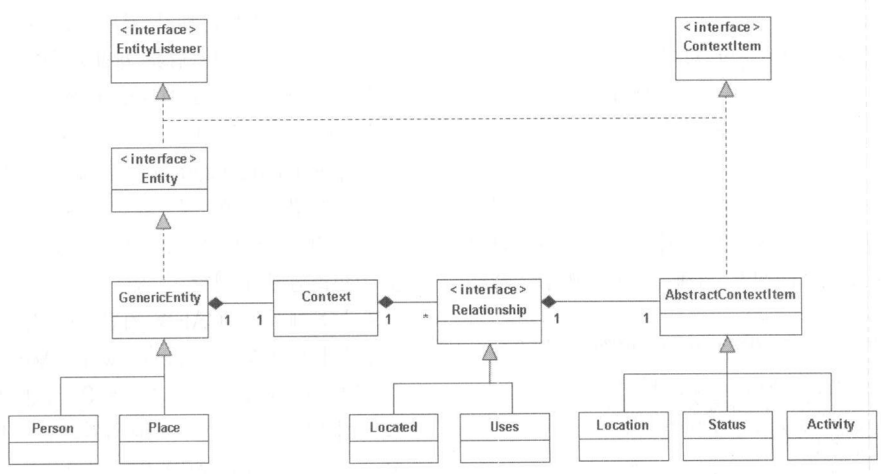
CARMAN[4]은 상황 인식 자원 관리를 위한 미들웨어이다. CARMAN을 기반으로 UbiCOSM[5]이라는 접근 제어 시스템으로 확장하였다. 이 시스템은 UbiCOSM 접근제어정책을 기술하여 UbiCOSM 미들웨어를 통해 실행되도록 구현되었다. 이는 사용자의 현재 상황 정보를 고려하여 다양한 접근 제어를 할 수 있도록 설계되었다. 하지만 정책을 기술함에 있어, 제어하고자 하는 자원의 인스턴스 하나하나마다 지정해야 하는 어려움이 있다. 이에 비해 본 논문에서는 클래스 타입이나 계층적 공간 관계를 표현할 수 있는 정책 기술 언어[1]를 기반으로 정책 정의의 편의성을 제공한다. 따라서 접근 제어를 필요로 하는 주체(subject)를 인스턴스, 클래스 등으로 다양하게 표현할 수 있고, 다양한 상황 정보를 고려함에 있어 물리적 상황 뿐 아니라 공간적인 상황 정보도 유연하게 표현할 수 있다.

[6]에서는 PLUE라는 유비쿼터스 언어를 기반으로 한 보안 정책 정의 모델을 제안하였다. 이를 이용하여 보안 관련 규칙을 정의하고 이에 따라 보안 검사가 이루어진다. 이는 전통적인 주체(subject) 기반의 접근 제어 시스템으로서, 상황 인식을 기반으로 한 본 시스템에 비해 상황 정보를 활용한 유연한 제어가 어렵다. 현 상황에 대한 고려 없이 각 주체가 어떤 객체(object)에 대해 갖는 접근 권한 정의를 통해서만 제어가 이루어지므로 상황 변화에 유동적으로 대처할 수 없다는 단점이 있다. 이해 비해 본 논문은 상황에 따라 유연하게 실행되는 상황인식 기술을 바탕으로 하였으며, 실세계와 근접한 형태의 공간적 계층관계를 표현할 수 있는 정책 기술언어를 사용함으로서 공간 상황 인식에 대한 처리도 유연하게 실행할 수 있다.

2.2. JCAF

JCAF는 상황 인식 프로그램 개발을 위한 하부 구조와 API를 제공하는 Java로 구현된 간단하면서도 견고한 프레임워크로 상황 인식을 효과적으로 지원할 수 있다. JCAF는 ContextService를 통해 여러 Entity들에 대한 정보 수집 및 관리를 지원한다. ContextService 인터페이스는 엔티티들의 추가 삭제 수집 등을 위한 관련 메소드를 제공한다.

JCAF의 핵심요소는 Entity, Context, Relationship, ContextItem이다. (그림 1)은 이러한 요소들의 관계를 보여준다. 각 Entity는 하나의 Context를 가지게 되고, 각 Entity가 다른 ContextItem들과 맺고 있는 Relationship들의 집합이 Context를 구성하게 된다. Entity의 예로는 사람, 장소, 사물 등이 될 수 있고, ContextItem의 예로는 위치, 상태 등이 될



(그림 1) JCAF의 핵심 구조

수 있다. 예를 들어 PDA가 로비에 있다면 PDA는 Entity, located는 Relationship, lobby는 ContextItem이 된다. 그리고 'PDA가 로비에 있다', 'PDA의 주인은 Bob이다' 등의 PDA와 관계를 맺고 있는 릴레이션들의 집합이 PDA의 Context가 된다.

Entity의 상황 변화를 처리하는 주요 부분은 EntityListener 인터페이스에 있는 contextChanged() 메소드이다. 이 메소드는 Entity의 Context가 바뀔 때마다 Entity Container에 의해 호출되어 Context 변화에 효과적으로 대응한다. 유비쿼터스 프로그램 개발자는 상황 변화에 따라 발생 가능한 이벤트들을 처리하기 위한 행동들을 이 메소드 내에 기술하면 된다.

```
public void contextChanged(ContextEvent event) {
  // 이벤트 발생 시에 적용시키고 싶은 액션을 기술한다.
  // 예) 누군가가 방에 들어왔다면 램프 스위치를 켜라.
}
```

3. 접근 제어 정책

(그림 2)처럼 프로그래머는 각 엔티티와 릴레이션들이 정의된 유비쿼터스 응용프로그램과 접근 제어를 위한 정책 파일을 작성한다. 그리고 엔티티의 메소드 호출시 접근 제어가 필요하다면 CACM에게 접근 권한 검사를 요청하는 코드를 프로그램에 삽입하고 접근 제어 규칙을 정책 파일에 정의한다.

3.1. 정책 기술 언어의 문법 소개

프로그래머는 정책 파일에 다음 문법에 따라 접근 제어 규칙을 기술한다. 이 규칙은 어떠한 조건을 만족할 때, 엔티티 집합들이 메소드에 접근 권한이 있는지에 대한 내용을 기술한다. 접근 제어 정책들의 기술에 필요한 엔티티와 릴레이션의 문법은 다음과 같다.

```
p ∈ Entity-Expression := id1:id2 | $id | $id_n | *
                                 | p1/p2 | .../p
r ∈ Relation-Expression := id1(p1,id2,p2) | ~r | r1 ^ r2
n ∈ Number
```

Entity-Expression은 어떤 한 개의 엔티티나 엔티티의 집합을 표현한다. id_1:id_2는 Pda:BobPda와 같이 id_1이라는 이름을 갖는 클래스의 인스턴스 id_2를 표현한다. $id는 id라는 이름을 갖는 클래스의 일반적인 인스턴스 중의 하나를 나타내는 변수를 의미한다. $Room은 Room이라는 클래스의 어떤 인스턴스를 담는 변수가 된다. 그리고 $id_n는 $id클래스의 인스턴스 중 서로 다른 두 개 이상의 인스턴스를 n이라는 숫자로서 구분한다.

그리고 공간 엔티티들간의 포함 관계 표현을 위해, '/'를 사용하게 된다. 예를 들어,

Hospital:ubihosp/Floor:fl1/$SickRoom

은 ubihosp라는 병원에 fl1이라는 층에 있는 어떤 병실을 의미한다. 또한 엔티티의 표현을 효율적으로 하기 위한 여러 가지 문법들을 제공하고 있는데, 예를 들어,

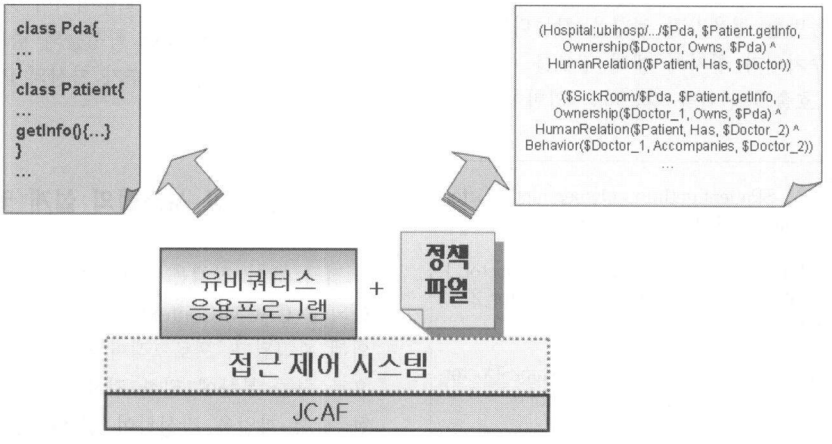
Hospital:ubihosp/.../$Pda

은 ubihosp에 있는 모든 Pda들의 인스턴스를 의미한다. 이를 통해 $Pda의 공간적인 포함관계를 모두 표현하지 않고 쉽게 설명될 수 있도록 정의할 수 있다. 또한,

Hospital:ubihosp/Floor:fl1/*

은 ubihosp 병원의 fl1층에 있는 모든 인스턴스를 표현한다. '*'는 하나의 클래스 타입으로 한정하지 않고 모든 클래스 타입의 인스턴스들에 허용될 규칙을 정의할 수 있도록 돕는다.

다음으로, Relation-Expression은 엔티티들 간의 관계에 대해서 표현하며, 이것은 참, 거짓으로 해석될 수 있다.



(그림 2) 응용프로그램 및 정책 파일 작성

$id_1(p_1,id_2,p_2)$는 p1이라는 Entity가 p2라는 ContextItem과 id2라는 Relationship을 맺음을 의미한다. 여기서 id1은 id2의 릴레이션 타입을 말한다. 예를 들어, HumanRelation($Patient, Has, $Doctor)은 Patient라는 클래스의 한 인스턴스가 어떤 Doctor를 주치의로 Has하는 관계를 나타낸다. '~'은 'not'연산자를 의미하며, '^'은 'and'연산자로서 여러 개의 관계를 대해 동시에 참, 거짓을 판단하도록 유도한다.

접근 제어 정책 기술의 문법은 다음과 같다.

```
x ∈ Access-Rule ::= (p,o,r) | x1 x2
o ∈ object ::= p.id
```

Access-Rule은 주체(p)가 조건(r)을 만족할 때 대상(o)를 호출할 수 있다는 정책을 정의한다. 메소드를 호출하는 호출자가 주체를 의미하며 이는 Entity-Expression으로 표현된다. 또한 대상은 엔티티와 엔티티의 메소드 이름으로 표현되며, 권한을 갖기 위해 필요한 조건은 Relation-Expression으로 표현된다. 따라서 이는 어떠한 엔티티가 다른 엔티티의 메소드에 접근하고자 할 때 갖추어져야 하는 조건이 릴레이션으로 표현되어야함을 의미한다. 간단한 예를 보면,

($Pda, $Printer.print, Location($Printer, IsIn, $Lobby))

는 어떤 PDA가 프린터에 print 메소드를 호출하고자 한다면 그 프린터가 로비에 있는 것이어야 한다는 의미로 로비에 있는 프린터는 개방하되, 회의실이나 다른 공간에 있는 프린터 사용은 제한하고자 할 때 사용될 수 있는 규칙이다.

3.2. 정책 파일의 예

(그림 3)은 유비쿼터스 병원에 대한 정책 파일의 예로 유비쿼터스 병원 프로그램에 적용될 엔티티들의 메소드 접근 권한에 대한 규칙이 포함되어 있다. 첫 번째 규칙은 주치의인 의사가 자신의 PDA를 통해 환자의 정보를 볼 권한을 갖는다는 규칙이다. 이 규칙에는 PDA가 ubihosp 병원 안에 있어야 한다는 의미가 내포되어 있다. 두 번째 규칙은 어떤 의사가 PDA를 가지고 병실에 들어와서 환자의 정보를 보려고 할 때, 주치의인 의사를 동반한 경우라면 권한을 갖는다는 규칙이다. 마지막 규칙은 주치의가 환자에 대해 정보를 수정하기 위해 setInfo 메소드를 호출할 권한을 갖는다는 의미이다.

```
(Hospital:ubihosp/.../$Pda, $Patient.getInfo, Ownership($Doctor, Owns, $Pda) ^ HumanRelation($Patient, Has, $Doctor))

($SickRoom/$Pda, $Patient.getInfo, Ownership($Doctor_1, Owns, $Pda) ^ HumanRelation($Patient, Has, $Doctor_2) ^ Behavior($Doctor_1, Accompanies, $Doctor_2))

(Hospital:ubihosp/.../$Pda, $Patient.setInfo, Ownership($Doctor, Owns, $Pda) ^ HumanRelation($Patient, Has, $Doctor))
```

(그림 3) 유비쿼터스 병원의 정책 파일

```
(Building:ubisoft/$Lobby/$Pda, $Lobby/$Printer.print, true)

($Pda,$Room/$Printer.print,Ownership($Pda,Owns, $Room))

($Pda, $Lounge/$Printer.print, Relation($Pda, Employed, Building:ubisoft))

($Pda_1, $Room/$Printer.print, Relation($Pda_2, Hosts, $Pda_1) ^ Ownership($Pda_2, Owns, $Room))
```

(그림 4) 유비소프트 빌딩의 정책 파일

```
($Pda, $Lecture.getData, Ownership($Student,Owns,$Pda) ^ Relation($Student,Register,$Lecture))

($Pda, $Lecture.getStudentInfo, Ownership($Professor,Owns, $Pda) ^ Relation($Lecture, Has, $Professor))

($Pda, $Lecture.AttendingStdents, Ownership($Professor, Owns, $Pda) ^ Relation($Lecture, Has, $Professor))
```

(그림 5) 유비쿼터스 캠퍼스의 정책 파일

(그림 4)의 예제는 ubisoft 빌딩 안에 있는 여러 개의 프린터에 대해서 프린터 사용 권한을 제어하는 예제이다. 첫 번째 규칙은 ubisoft 빌딩 안의 로비에 있는 PDA는 로비에 있는 프린터로 출력할 수 있다는 의미이다. 두 번째는 PDA가 방안에 있는 프린터에 대해 출력하려면 그 PDA를 가진 사람이 그 방의 주인이어야 한다는 의미이다. 세 번째는 라운지에 있는 프린터로 출력하려면 그 PDA를 가진 사람은 ubisoft 빌딩에 고용된 직원이어야 한다는 의미이다. 네 번째는 방에 있는 프린터로 출력하고자 할 때 그 방의 주인인 다른 PDA와 동반(Hosts)한다면 print 메소드 호출 권한을 갖는다는 의미이다.

(그림 5)의 예제는 유비쿼터스 캠퍼스에 대한 접근 제어 정책 파일이다. 첫 번째 규칙은 학생이 자신의 PDA로 강의 자료를 다운로드 받고자 할 때 해당 강의에 등록되어있어야 한다는 규칙이다. 두 번째는 어떤 강의를 수강하는 학생들의 이름, 성적 등 정보를 PDA를 통해 보려면 PDA의 주인인 교수가 해당 강의의 담당교수이어야 한다는 의미이고 세 번째 규칙은 한 강의에 대한 학생들의 출결사항을 PDA로 확인하려면 그 강의의 담당교수가 자신의 PDA를 통해서 확인할 수 있다는 의미이다.

4. 접근 제어 시스템의 설계 및 구현

접근 제어 정책은 실행 시간에 ContextService와 함께 구동되는 CACM(Context-aware Access Control Manager)에 의해 관리된다. 프로그래머가 응용프로그램의 접근 제어를 필요로 하는 부분에 권한 검사 요청 코드를 삽입하고 그에 대한 정책 파일을 작성하여 실행하면 CACM은 그 접근 제어 규칙에 따라 권한 검사를 시행한다.

4.1. 실행 방법

프로그래머가 작성한 정책 파일은 CACM에 의해 접근 제어가 이루어진다. CACM은 권한 검사를 위한 다음의 메소드를 제공하고 있다. 이 메소드를 통해 메소드 접근에 대한 권한 검사를 요청할 수 있다. 프로그래머는 메소드 실행 전에 접근 제어 권한 검사를 요청하고자 한다면 아래의 메소드를 호출하여 <entity1>이 <entity2>의 메소드 <method name>을 접근할 권한을 갖는지 검사한다.

```
checkMethodAccess(<entity1>,<entity2>,<method name>)
```

권한 검사가 필요한 부분에 checkMethodAccess 메소드를 호출하면 CACM은 현재의 상황(context)과 정책 파일이 저장된 해시테이블을 비교하여 결과를 '참', '거짓'으로 반환한다. '참'이 반환되면 아무런 제한 없이 메소드가 실행되고 '거짓'이 반환되면 AccessControlException()이 발생한다.

```
if ( cacm.checkMethodAccess(Pda1, Printer01, print) )
{
        Printer01.print();
}
```

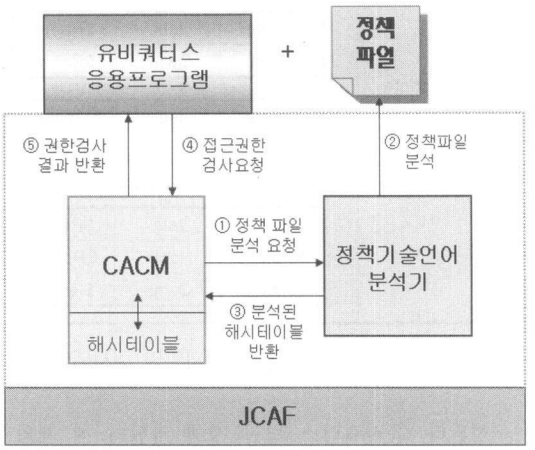
PDA가 프린터를 사용하기 위한 권한에 대해 정책 파일을 작성하고 유비쿼터스 프로그램에 위와 같은 코드를 작성했다면 checkMethodAccess 메소드에 의해 권한 검사 요청을 하게 된다. CACM의 검사 결과에 따라 출력명령이 수행될 수도, 거부될 수도 있다. 정책 파일 상의 규칙 중 현재의 상황이 수용할 수 있는 규칙이 존재하지 않을 경우 메소드 접근이 거부된다.

CACM의 checkMethodAccess 메소드는 매개변수로 들어온 entity1이 entity2의 어떠한 메소드를 실행할 권한이 있는지 검사하여 결과를 반환하는 메소드이다. 메소드 실행에 관한 권한 규칙은 해시 테이블로 저장하고 있는 상태이고, 현 상황 정보(context)가 이 규칙을 만족하는지 entity 인스턴스들의 릴레이션 정보들을 ContextService를 통해 수집한다.

4.2. 실행 구조

CACM은 접근 권한 검사 요청이 들어오면, 정책파일을 분석한 해시테이블 정보를 이용하여 현재 엔티티의 상황들이 조건을 만족하는지 검사하여 결과를 반환한다.

(그림 6)은 실행 시간에 접근 제어가 이루어지는 과정을 보인 그림이다. 우선 프로그래머는 앞서 설명된 정책 기술 언어를 통해 엔티티들이 메소드 접근에 대해 권한을 갖기 위한 조건을 정책 파일에 기술한다. 그리고 그와 더불어 각 클래스들을 정의한 Java프로그램을 작성하게 된다. 그 다음 CACM은 정책 기술 언어 분석기에게 정책 파일을 분석해 줄 것을 요청한다. 정책 기술 언어 분석기는 정책 파일을 분석하여 조건들을 추출해내고 해시테이블 형태로 CACM에게 반환한다. 프로그래머가 작성한 Java 프로그램은 실행 중 메소드 호출이 발생할 때 checkMethodAccess 메소드를 호출함으로써 엔티티 객체가 호출에 대한 권한을 갖는지 권한 검사를 요청하게 된다. 마지막으로 CACM은 현재의 상황(context)과 해시테이블의 조건이 일치하는지 분석하여 결과를 반환한다.

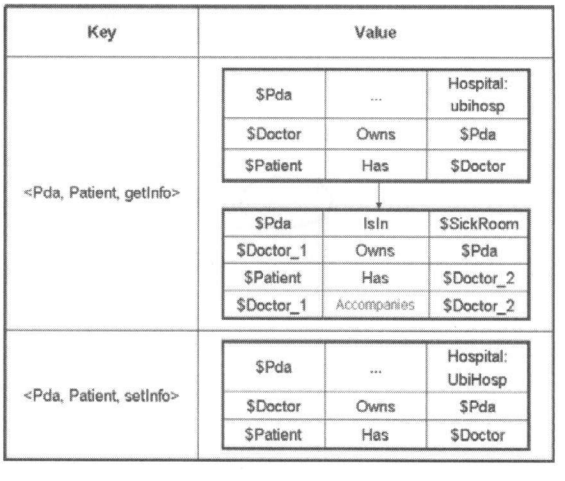


(그림 6) 접근 제어 시스템의 실행 구조

4.3. 해시테이블의 구성

정책 파일을 저장하고 있는 해시 테이블은 메소드의 호출자인 엔티티 객체와 메소드를 가지고 있는 피호출자, 그리고 메소드 이름을 키로 갖는다. 예를 들어 <Pda, Printer, print>를 키로 하면 어떤 Pda가 프린터의 print 메소드를 호출할 때 필요한 권한과 조건들이 이 키의 값으로 저장되어 있다. 하나의 키로 접근할 수 있는 조건 집합들은 여러 개일 수 있다. 조건 집합(condition set)은 하나의 접근 제어 규칙에 의해 얻어지는 조건들의 집합을 말한다. 키를 통해 얻어진 여러 조건 집합들 중 하나의 조건 집합이라도 만족하게 되면 CACM은 참을 반환하고 검사를 마치게 된다.

(그림 7)은 3.2.1절에서 소개된 유비쿼터스 병원의 정책 파일을 분석하여 구성된 해시테이블이다. 우선 하나의 규칙을 통해서 키가 추출되고 권한을 갖기 위한 규칙들이 조건 집합



(그림 7) 유비쿼터스 병원 정책 파일의 해시테이블

Key	Value
<Pda,Printer,print>	$Lobby | IsIn | Building: ubisoft $Pda | IsIn | $Lobby $Printer | IsIn | $Lobby $Printer | IsIn | $Room $Pda | Owns | $Room $Printer | IsIn | $Lounge $Pda | Employed | Building: ubisoft $Printer | IsIn | $Room $Pda_2 | Hosts | $Pda_1 $Pda_2 | Owns | $Room

(그림 8) 유비소프트 빌딩 정책 파일의 해시테이블

으로 구성된다. 유비쿼터스 병원 정책 파일의 세 개의 규칙 중 환자의 getInfo 메소드에 대한 권한 제어와 관련된 두 개의 규칙이 같은 키를 갖게 되고 각각의 조건 집합을 구성하게 된다. 또한 환자의 setInfo 메소드 접근 제어에 관한 하나의 규칙이 키를 구성하고 하나의 조건 집합을 구성하게 된다.

유비쿼터스 응용프로그램에서 checkMethodAccess(Pda01, Patient_Tom, getInfo)를 호출함으로써 Pda01을 가진 의사가 Tom의 정보를 불러오기 전에 권한 검사를 요청했다고 가정하자. CACM은 메소드 호출의 인자들을 분석하여 <Pda, Patient, getInfo>라는 키를 생성할 것이다. 따라서 이와 관련된 조건들이 포함된 두 개의 조건 집합이 고려 대상이 된다. 먼저 첫 번째를 고려해보면 Pda01의 주인인 의사가 Tom의 주치의가 아니라면 해당 조건 집합은 만족하지 못하므로 두 번째 조건 집합검사로 넘어간다. 만일 권한검사가 이루어지는 시점의 상황(context)을 분석한 결과 Pda01의 주인인 의사가 Tom의 주치의와 동반하여 병실에 들어온 것이라면 권한 검사 결과를 '참'으로 반환할 것이다. 하지만 두 번째도 만족하지 못한다면 최종 권한 검사 결과는 '거짓'으로 반환된다.

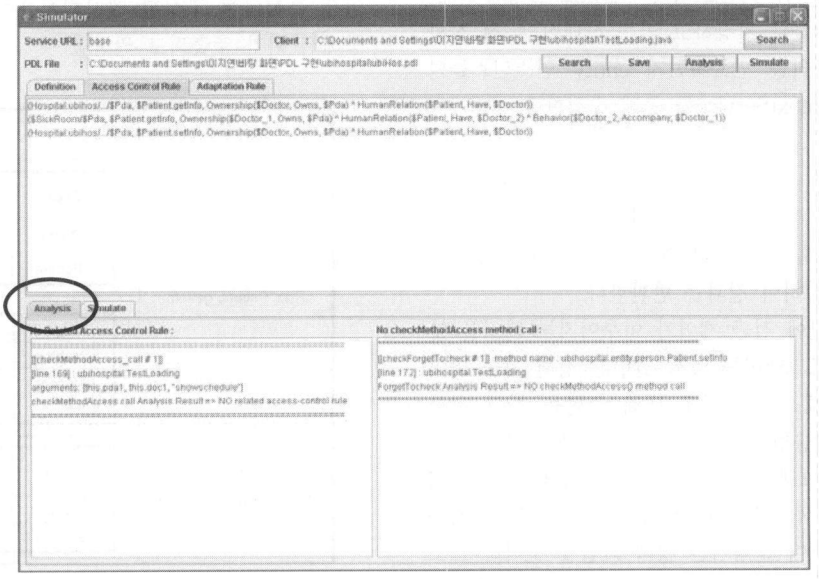
검사 시간의 효율성을 위하여 하나의 조건 집합이라도 '참'의 결과를 얻게 되면 바로 검사를 중지하고 최종 결과를 '참'으로 반환한다. 또한 하나의 조건 집합검사 중 하나의 조건도 만족하지 않으면 해당 조건 집합의 검사는 '거짓'으로 판정된다.

5. 접근 제어 분석도구

정책 파일의 접근 제어 규칙과 응용프로그램의 권한 검사 요청이 명확하게 정의되어야 효과적인 접근 제어가 이루어질 수 있다. 본 연구에서는 접근 제어 분석도구를 제공함으로써 프로그래머가 효과적으로 정책 파일을 작성하고 프로그래머의 의도가 최대한 프로그램에 반영될 수 있도록 도와준다.

이 분석 도구는 정적 분석을 통하여 두 가지의 정보를 제공한다. 첫 번째는, 접근 제어 규칙의 누락에 대한 분석으로 이런 경우에는 접근 제어 권한 검사 요청 시, 항상 '거짓'이 되어 효율성을 떨어뜨리는 원인이 된다. 이는 해당 권한 검사에 관한 규칙이 정책 파일에 전혀 포함되어 있지 않은 경우에 발생한다. 두 번째는, 권한 검사 요청이 정확히 이루어지지 않은 경우에 대한 분석으로 정책 파일에 접근 제어의 필요성으로 인하여 기술된 규칙이 존재함에도 불구하고, 실제 프로그램 상에서는 검사를 요청하지 않고 있는 경우에 대한 분석을 말한다. (그림 9)는 정책 파일 분석 지원 시스템의 실행 화면이다.

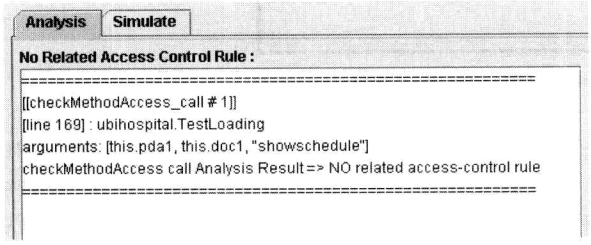
예를 들어, 프로그래머가 작성한 프로그램 중에 다음과 같은 코드가 삽입되어 있었다고 가정해 보자.



(그림 9) 정적 분석을 통한 정책 파일 분석 시스템 실행화면

```
if ( cacm.checkMethodAccess(Pda1, Printer01, print) )
{
        Printer01.print();
}
```

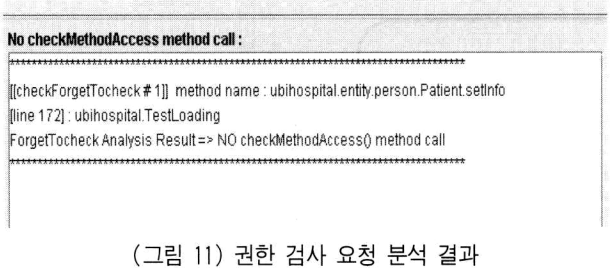
그런데, 만약 정책 파일 상에 PDA가 프린터에 출력하기 위한 권한 규칙에 대한 정보가 존재하지 않는다면 이 권한 검사는 언제나 '거짓'을 반환하는 불필요한 검사가 될 것이다. 따라서 권한 검사가 꼭 필요한 메소드 접근이라면 관련 규칙을 정책 파일에 기술해야 할 것이고, 아니라면 checkMethodAccess 메소드 호출을 하지 말고 직접 메소드에 접근해야 할 것이다. 프로그래머는 이러한 정보를 통해 정책 파일과 응용프로그램을 수정해야 한다.



(그림 10) 접근 제어 규칙 부재 분석 결과

(그림 10)은 유비쿼터스 병원 응용 프로그램의 분석 결과이다. ubihospital패키지의 TestLoading파일의 169번째 줄에서 checkMethodAccess 메소드 호출이 발생하였는데 이 권한 검사가 의미 있는 권한 검사인지 분석해주는 부분이다. Pda1이 의사 doc1의 showschedule 메소드를 호출하기 전에 접근 권한이 있는지 검사하겠다는 요청이 있었다. 하지만 정책 파일에 showschedule 메소드 호출에 관한 접근 제어 규칙이 존재하지 않으므로 이 검사는 잘못된 검사 요청이라는 것을 프로그래머에게 알려주는 분석 결과이다.

또한 프로그래머는 정책 파일과 응용 프로그램을 따로 작성하게 되므로 그 파일이 유기적으로 해석될 수 있어야 한다. 정적 분석을 통해 정책파일에 접근 제어 규칙이 선언되어 있으나 실제 프로그램 상에서 메소드 접근 전 검사 요청을 하지 않는 부분에 대한 정보를 제공함으로써 프로그래머가 접근 제어 요청을 보정할 수 있도록 한다. 즉 접근 제어를 하고자 정책 파일에 기술하였으나 프로그래밍 중에 미처 checkMethodAccess 메소드 호출을 통한 검사 요청을 하지 못한 경우를 찾아내서 알려줌으로써 해당 메소드 호출을 추가할 수 있도록 돕는다.



(그림 11) 권한 검사 요청 분석 결과

(그림 11)은 유비쿼터스 병원 응용 프로그램의 분석 결과이다. ubihospital 패키지의 TestLoading 파일의 172번째 줄에서 정책 파일에 포함되어 있는 setInfo 메소드가 호출된 경우에 대한 분석이다. 유비쿼터스 병원 정책 파일에 환자의 setInfo 메소드 호출에 대한 권한 규칙이 기술되어 있는데 실제 프로그램 상에서 setInfo 메소드 호출 시 checkMethodAccess 메소드 호출을 통한 권한 검사요청이 누락되어 있어 제대로 접근 제어가 이루어지지 않음을 프로그래머에게 알려주는 결과화면이다.

6. 시뮬레이터 및 실행 결과

이 장에서는 본 시스템을 기반으로 한 유비쿼터스 응용 프로그램 실행을 위해 개발된 시뮬레이터[7]를 소개하고, 이를 통해 시뮬레이션한 결과를 보인다.

6.1. 시뮬레이터

본 시스템을 이용하여 구현한 유비쿼터스 프로그램의 실행을 자유롭게 시뮬레이션 할 수 있는 환경을 위하여 시뮬레이터를 개발하였다. 이 시뮬레이터는 또한 접근 제어에 의하여 메소드 접근이 거부되는 경우 이에 대한 정보를 제공해준다. 또한 시뮬레이션 과정에서 정책 파일 상의 오류가 파악되면 수정하여 다시 재적용 후 시뮬레이션 할 수 있다.

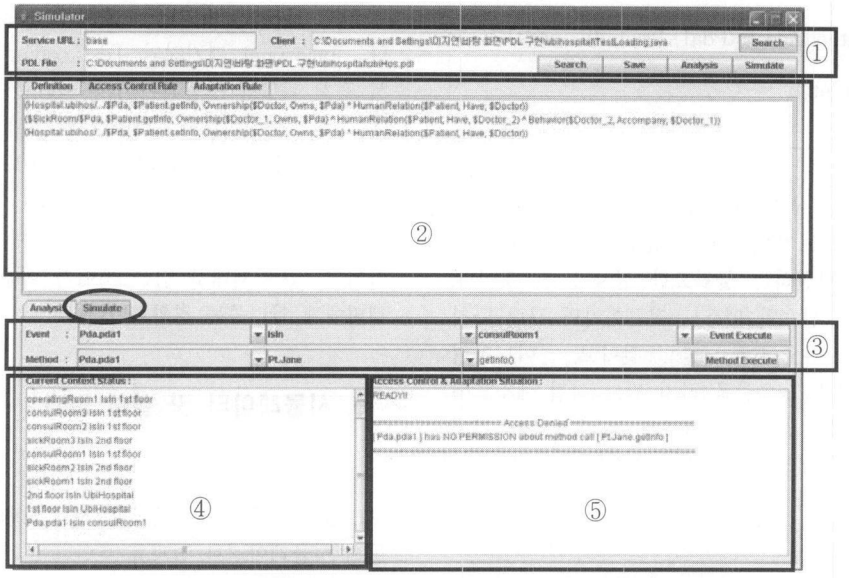
(그림 12)는 유비쿼터스 병원 프로그램을 실행한 결과로 시뮬레이터는 크게 다섯 영역으로 구성된다.

(1) 유비쿼터스 응용 프로그램과 정책 파일 선택 영역 : ContextService 서버의 이름을 적고, 응용 프로그램과 정책 파일을 선택한다.
(2) 정책 파일 편집 영역 : 1번 영역에서 선택된 정책 파일이 보여주고 필요시 편집이 가능하며 편집된 새로운 정책파일을 재적용시킬 수 있다.
(3) 이벤트 발생 및 메소드 실행 영역 : 프로그래머가 자유롭게 이벤트를 발생시키거나 메소드를 실행시킴으로써 시뮬레이션 할 수 있다.
(4) Context 상황 정보 표시 영역 : 3번 영역의 시뮬레이션을 통해 context 정보의 변화되는 상황을 표시한다.
(5) 실행 결과 영역 : 접근 제어를 통한 메소드 접근 거부사항이나, 메소드 접근 허용으로 메소드가 실행될 경우의 결과 화면을 보여준다.

6.2. 실행 결과

본 시스템을 이용하여 세 가지 응용 프로그램을 시험하였다. 각 프로그램의 시나리오와 시뮬레이션 결과를 설명한다.

유비쿼터스 병원은 의사가 자신의 PDA를 통해 언제 어디서나 쉽게 환자의 정보를 확인할 수 있도록 개발된 프로그램으로 환자가 진료실에 들어오면 자동으로 환자의 정보를 보여주거나, 병실에서 환자에 다가서면 해당 환자의 정보를 자동으로 보여줄 수 있다. 이 때 환자의 정보를 보거나 수정하



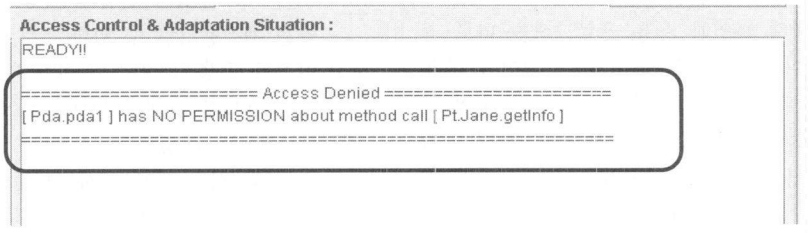
(그림 12) 시뮬레이터를 이용한 유비쿼터스 병원 실행 결과

는 권한을 제어하기 위해 정책 파일을 작성하고 CACM을 이용해 접근 제어할 수 있도록 프로그래밍하였다.

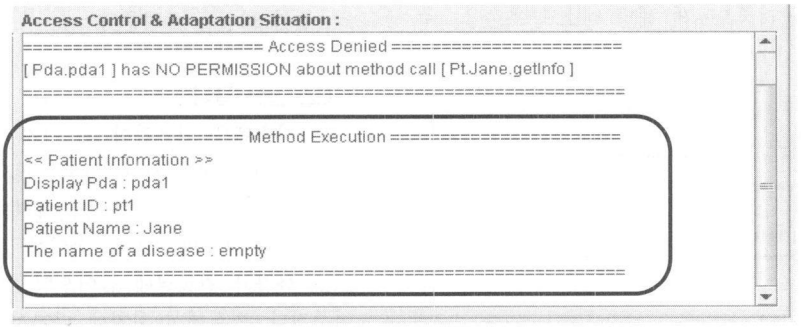
(그림 13)은 유비쿼터스 병원을 시뮬레이션했을 때 접근 제어에 의해 메소드 호출이 거부된 결과를 보여준다. 시뮬레이터를 통해 Pda1이 환자 Jane의 정보를 보고자 메소드 호출을 요청했는데 Pda1은 정책 파일 상에 Jane의 정보를 볼 수 있는 권한이 없기 때문에 접근이 거부되었다. 이 때 Pda1의 주인을 의사 Dr. Park으로 Ownership 릴레이션을 설정하고, Dr. Park을 환자 Jane의 주치의로 'Has' 릴레이션을 설정하면 결과는 (그림 14)와 같이 달라진다. 정책 파일 상에 환자의 정보를 보려면 그 환자의 주치의여야 한다는 규칙이 존재하므로, Jane의 주치의인 Dr. Park은 권한을 갖게 된 것이다.

유비소프트 빌딩은 하나의 빌딩 안에 설치된 여러 프린터들에 대해 접근 제어를 하고자 개발된 프로그램으로 빌딩 안에 있는 각각의 프린터들마다 그 위치에 따라 사용할 수 있는 권한을 가진 그룹이 분리된다. (그림 4)의 정책 파일에 따라 이 프로그램을 시뮬레이션하면 로비, 방, 라운지 등의 위치에 따라 프린터 사용 권한이 다르게 제한된다.

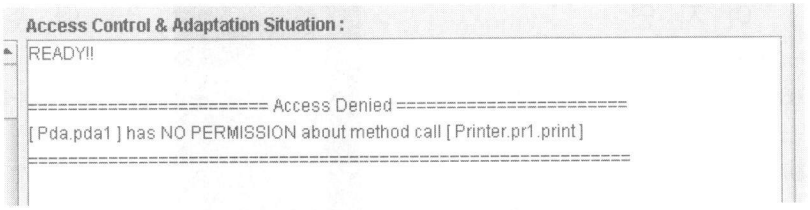
(그림 15)는 Pda1이 Lounge1에 있는 Printer1에 프린트를 시도한 결과이다. 정책 파일에 의하면 라운지에 있는 프린터를 사용할 수 있는 사람은 그 빌딩에 고용(Employed)되어 있는 사람이어야 하는데 Pda1은 그렇게 않기 때문에 접근이 거부된 것이다. 따라서 (Pda1, Employed, Building:ubisoft)라는 릴레이션을 발생시키면 접근이 허락될 것이다.



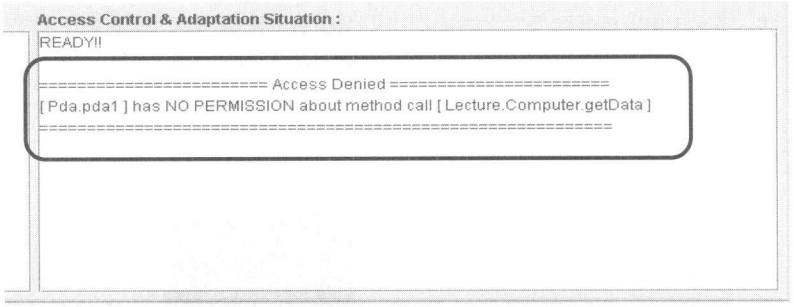
(그림 13) 유비쿼터스 병원의 getInfo 메소드 실행거부 결과



(그림 14) 유비쿼터스 병원의 getInfo 메소드 실행 결과



(그림 15) 유비소프트 빌딩 print 메소드 실행 거부 결과



(그림 16) 유비쿼터스 캠퍼스의 getData 메소드 실행 거부 결과

유비쿼터스 캠퍼스 응용프로그램은 학생과 교수의 상황을 인식하여 자동으로 서비스해주는 프로그램으로 교수가 강의실에 들어오면 자동으로 강의자료를 디스플레이 해주고 학생의 Pda를 통해 강의 자료를 다운로드 받을 수 있다. 이 때 그 강의를 수강하고 있는 학생만 다운로드 받을 수 있도록 하고, 강의의 담당교수만 성적 관리 등의 학생 정보 접근을 할 수 있도록 정책 파일을 작성하였다. (그림 16)은 권한이 없는 학생이 수업자료를 다운로드하려 했을 때 거부되는 실행 화면이다.

7. 결 론

본 논문에서는 안전한 유비쿼터스 프로그래밍을 위한 접근 제어를 지원하고자 프로그래머가 작성한 정책 파일을 바탕으로 실행 시간 중 접근 제어를 수행하는 접근제어 시스템을 구현하였다. 이 시스템은 유비쿼터스 응용 프로그램이 효과적으로 접근 제어를 수행할 수 있도록 한다. 또한 정적 분석을 통해 프로그래머에게 누락된 접근 제어 규칙이나 잘못된 권한 검사를 보여줄 수 있는 정적 분석 도구을 구현하였다. 또한 본 시스템을 이용하여 구현된 프로그램을 다양한 시나리오로 실행해볼 수 있는 시뮬레이터를 제공하였다. 본 연구는 역할에 따른 보다 체계적인 접근제어를 지원하기 위해 역할 기반 접근제어 시스템으로 확장할 것이다.

참 고 문 헌

[1] J. Ahn, B.-M. Chang, and K.-G. Doh, A Policy Description Language for Context-based Access Control and Adaptation in Ubiquitous Environment, TRUST06, August, 2006.

[2] J. E. Bardram, The Java Context Awareness Framework-A Service Infrastructure and Programming Framework for Context-Aware Applications, Third International Conference, Pervasive 2005, May, 2005.

[3] A. Ranganathan, R. H. Campbell, An infrastructure for context-awareness based on first order logic, Personal and Ubiquitous Computing 7(6), pp.353-364, 2003.

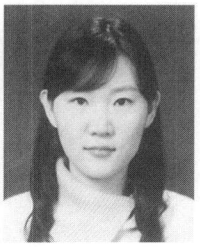
[4] P. Bellavista, A. Corra야, R. Montanari, Context-Aware Middleware for Resource Management in the Wireless Internet, IEEE Trans. on Soft. Eng. 29(12), Aug., 2003.

[5] Antonio Corradi, Rebecca Montanari, Daniela Tibaldi, Context-based Access Control for Ubiquitous Service Provisioning, Proceedings of COMPSAC'04, 2004.

[6] E. Cho and K. Lee, Security Checks in Proframming Language for Ubiquitous Environment, Pervasive computing, pp.52-61, 2003.

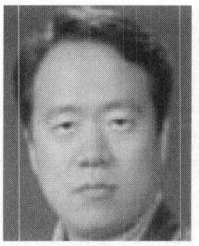
[7] 이지연, 오민경, 창병모, 안준선, 도경구, 유비쿼터스 컴퓨팅을 위한 접근제어와 상황적응 시스템, 한국정보과학회 가을학술발표논문집(B) 제33권 제2호, pp.590-594, 2006년10월.

[8] T. Gu, H. Pung and D. Zhang. A Service-Oriented Middleware for Building Context-Aware Services. Journal of Network and Computer Applications, 28(1), pp.1-18, Jan., 2005.



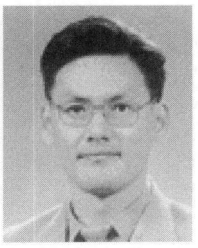
이 지 연

e-mail : jiyeon@sookmyung.ac.kr
2003년 숙명여자대학교 컴퓨터과학과 (이학사)
2007년 숙명여자대학교 컴퓨터과학과 (이학석사)
2007년~현 재 UC Berkeley-Traffic Safety Center 방문연구원
관심분야 : 유비쿼터스 컴퓨팅, 프로그래밍 언어론 등



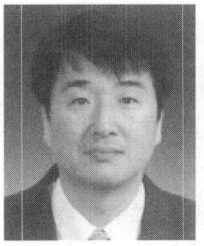
안 준 선

e-mail : jsahn@kau.ac.kr
1992년 서울대학교 계산통계학과 (이학사)
1994년 KAIST 전산학과 (공학석사)
2000년 KAIST 전자전산학과 (공학박사 (전산학))
2000년~2001년 KAIST 프로그램분석 시스템연구단 연구원
2001년~현 재 한국항공대학교 항공전자및정보통신공학부 교수
관심분야 : 프로그래밍언어, 프로그램분석, 웹보안, 프로그램 보안, 유비쿼터스컴퓨팅, 정보검색



창 병 모

e-mail : chang@sookmyung.ac.kr
1988년 서울대학교 컴퓨터공학과(공학사)
1990년 KAIST 전산학과(공학석사)
1994년 KAIST 전산학과(공학박사)
1995년~현 재 숙명여자대학교 컴퓨터과학과 교수
관심분야 : 프로그래밍 언어 및 시스템, 유비쿼터스 소프트웨어



도 경 구

e-mail : doh@hanyang.ac.kr
1980년 한양대학교 산업공학과(학사)
1987년 미국 아이오와주립대학교 전산학과 (석사)
1992년 미국 캔사스주립대학교 전산학과 (박사)
1993년~1995년 일본 會津대학 교수
1995년~현 재 한양대학교 컴퓨터공학과 교수
관심분야 : 프로그래밍언어, 프로그램분석, 소프트웨어보안, 유비쿼터스 프로그래밍 환경, 소프트웨어공학