상황 인식 역할 기반 접근 제어에 기반한 유비쿼터스 환경 정책 기술 언어*

신재호[1] 강경구[01] 안준선[1] 창병모[2] 도경구[3]
[1]한국항공대학교 {windsjh, chwoddl[0], jsahn}@kau.ac.kr
[2]숙명여자대학교 chang@sookmyung.ac.kr
[3]한양대학교 doh@ces.hanyang.ac.kr

Policy Description Language for Ubiquitous Environment based on Context-aware Role-based Access Control

JaeHo Shin[1], KyoungKoo Kang[01], Joonseon Ahn[1], ByeongMo Chang[2], KyoungGoo Doh[3]
[1]Korea Aerospace University, Koyang
[2]Sookmyung Women's University, Seoul
[3]Hanyang University, Ansan

요 약

본 논문에서는 상황 인식 역할 기반 접근 제어에 기반한 유비쿼터스 환경 정책 기술 언어를 제시한다. 역할 기반 접근 제어는 권한을 사용자에게 부여하지 않고 역할에 부여함으로써 접근 제어를 효율적으로 관리할 수 있다. 유비쿼터스 환경에서는 빈번하게 변화하는 사용자들의 위치 정보와 같은 동적인 상황을 고려해야 할 필요성이 있다. 본 논문에서는 유비쿼터스 환경에서 동적으로 변화하는 상황을 고려하는 역할 기반 접근 제어 모델을 고안하고 이에 기반하여 정책 기술 언어를 설계하였다. 제안된 언어는 역할 기반 접근 제어의 장점을 활용하며, 동적인 상황 조건에 따라 사용자에게 역할을 할당하고 정적인 상황 조건을 이용하여 역할에 권한을 부여하는 방법을 기술함으로써 역할 기반 접근 제어에 기반한 상황 인식 유비쿼터스 환경의 특징을 효율적으로 구현할 수 있다.

1. 서 론

유비쿼터스 환경이란 사용자가 시스템을 의식하지 않고 장소에 상관없이 자유롭게 시스템에 접속하여 서비스를 이용할 수 있는 정보통신 환경을 말한다. 따라서 유비쿼터스 시스템은 빈번하게 변화하는 사용자들의 환경 정보에 따라 서비스할 수 있어야 하며, 이를 위해 상황 정보를 데이터로써 표현할 수 있는 기술 언어가 필요하다. 또한, 사용자들이 유비쿼터스 시스템을 통해 서비스를 이용하여 많은 정보에 접근하게 되는데 이때 정보의 불법적인 사용을 막기 위해 접근 제어가 필요하다.

기존의 접근 제어 모델에는 임의 접근 제어(Discretionary access control)와 강제 접근 제어(Mandatory access control)[1], 역할 기반 접근 제어(Role-based access control)[2]라는 세 가지의 주요한 모델이 있다. 임의 접근 제어 모델[1]은 자원의 소유자가 사용자에 따라 임의로 접근 제어하는 방식이다. 즉, 자원의 소유자는 자원 접근 대상을 시스템과 상관없이 임의로 결정할 수 있다. 그러므로 시스템은 정보 흐름에 대한 실질적인 보장을 제공하지 않으며, 접근 제어 관리가 쉽지 않다. 강제 접근 제어 모델[1]은 시스템이 자원에 대한 접근 허가 여부를 결정한다. 이를 위해 시스템 관리자는 모든 사용자와 자원에 보안 레벨을 부여한다. 이 레벨에 따라 사용자는 자신보다 낮은 레벨이나 같은 레벨의 자원에 대해 권한을 가질 수 있다. 그러나 관리자만이 자원의 접근 권한을 변경할 수 있고, 모든 사용자와 모든 자원에 대해 관리하기 때문에 효율적인 접근 제어 관리가 어렵다. 또한, 임의/강제 접근 제어 모델을 기반으로 제시한 정책 기술 언어[6]는 접근 권한을 각 사용자에게 부여하기 때문에 권한이 변경되면 권한을 부여받은 모든 사용자에게 권한의 변경 사항을 알려야 하기 때문에 비효율적이다.

이에 비해, 역할 기반 접근 제어 모델[2]은 자원에 대한 접근 권한을 각 사용자가 아닌 역할에 부여하고 시스템은 사용자들에게 역할을 할당한다. 따라서 역할 기반 접근 제어 모델을 기반으로 정책 기술을 언어를 설계하면, 권한이 변경되면 권한의 변경사항을 역할에만 알리면 되므로 임의/강제 접근 제어모델보다 관리가 효율적이다. 그러나 이 모델은 유비쿼터스 환경과 같이 동적으로 변화하는 상황 정보가 존재하는 환경에서는 효과적인 접근 제어를 할 수가 없다. 상황 정보를 이용한 역할 기반 접근 제어 모델에 대한 기존 연구들도 있지만 위치 정보만을 이용하여 제한된 접근 제어를 하거나[3], 사용자, 역

* 본 연구는 한국과학재단 특정기초연구(R01-2006-000-10926-0) 지원으로 수행되었음.

할, 객체의 모든 상황 정보를 이용[4, 5]하거나 역할과 권한 사이에 상황 정보들을 적용함으로 자원에 대한 접근 제어의 복잡성이 증가한다.

상황 정보를 이용하여 권한을 특정 사용자에게 부여하는 접근 제어 모델[6]은 모든 상황 조건을 검사해야 하며 권한 기술시 관련 조건을 모두 명시해야 하는 단점이 있다. 따라서 상황 정보를 동적인 상황과 정적인 상황으로 구분하여 활용하는 것이 필요하며 이러한 상황 정보를 이용하여 역할 기반의 접근 제어가 필요하다.

본 논문에서는 우선 동적인 상황 정보와 정적인 상황 정보를 구분하여 동적 상황 조건을 통해 사용자에게 역할을 할당하고 정적 상황 조건을 이용하여 역할에 권한을 부여하는 상황 인식 역할 기반 접근 제어 모델을 고안하고 이에 기반하여 유비쿼터스 환경에서의 정책 기술 언어를 설계하였다. 제안된 언어는 역할 기반 접근 제어의 장점을 활용하며, 동적인 상황 조건에 따라 사용자에게 역할을 할당하고 정적인 상황 조건을 이용하여 역할에 권한을 부여하는 방법을 통하여 상황 인식에 기반한 역할 기반 접근 제어를 효율적으로 구현할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 상황 인식 역할 기반 접근 제어 모델을 설명하고 3장에서는 유비쿼터스 환경에서의 정책 기술 언어를 설명한다. 4장에서는 역할 기반 정책 기술 언어를 설명하고, 5장에서는 본 논문의 결론과 향후 연구에 대해 설명한다.

2. 상황 인식 역할 기반 접근 제어 모델

그림 1은 상황 인식 역할 기반 접근 제어 모델의 구조를 나타낸다. 표 1은 상황 인식 역할 기반 접근 제어 모델을 대수적으로 표현한 것이다.

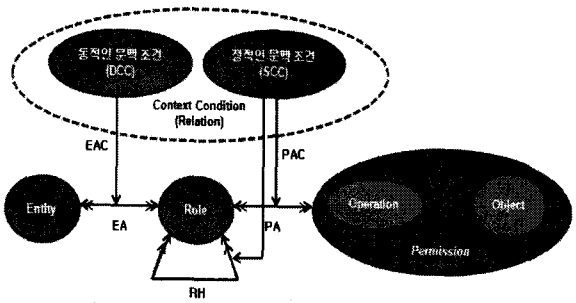

그림 1. 상황 인식 역할 기반 접근 제어 모델 구조

엔티티(E=Entity)는 사람이나 Pda, 프린터, 빌딩 등을 나타내며, 역할(R=Role)은 특별한 엔티티로써 교수, 학생과 같은 임무를 나타낸다. 권한(PRMS=Permission)은 대상(Obs=Object)과 작업(Ops=Operation)의 관계(Relation)로 이루어진다. 상황 조건(CC=Context Condition)은 동적인 상황 조건(DCC=Dynamic Context Condition)과 정적인 상황 조건(SCC=Static Context Condition)으로 구분된다. EA(Entity Assignment)는 엔티티가 가질 수 있는 역할을 나타내고, PA(Permission Assignment)는 역할과 관련된 권한을 나타내며, RH(Role Hierarchy)는 역할 간의 관련성에 의한 계층을 나타낸다. EAC(Entity Assignment Context Condition)는 동적인 상황 조건(DCC)과 EA의 관계로 이루어진다. PAC(Permission Assignment Context Condition)는 정적인 상황 조건(SCC)과 PA의 관계로 이루어진다. 이때 PAC의 SCC는 시스템 관리자 단계에서 결정되기 때문에 PA가 고정된다.

시스템은 동적으로 변화하는 상황을 인식하여 모든 EAC중 DCC가 참인 EAC를 결정하고 해당 엔티티에게 역할을 할당한다. 역할을 할당받은 엔티티는 할당된 역할의 권한과 하위 역할의 권한을 모두 갖게 된다. 예를 들어 RH의 원소로써 (유비쿼터스 강의교수, 유비쿼터스 강의조교, True)가 있고 Tom이 유비쿼터스 강의교수라는 역할을 할당받았다면 Tom은 유비쿼터스 교수의 권한뿐만 아니라 유비쿼터스 강의조교의 권한까지 갖게 된다.

표 1. 상황 인식 역할 기반 접근 제어 모델의 대수적 표현

$$
\begin{aligned}
&\text{Dynamic\_Attribute} = \{\text{Date, Time, Location, ...}\}\\
&\text{Static\_Attribute} = \{\text{ClassID, Name, Age, ...}\}\\
&\\
&DCC \subseteq 2^{\text{Dynamic\_Attribute}} \quad /*\ \text{DCC:Dynamic Context Condition}\ */\\
&SCC \subseteq 2^{\text{Static\_Attribute}} \quad /*\ \text{SCC:Static Context Condition}\ */\\
&(R_1, R_2, SCC) = R_2 \subseteq R_1 \text{ if } SCC = \text{True}\\
&\qquad /*\ R_1\text{:상위 역할, } R_2\text{:하위 역할}\ */\\
&RH \subseteq 2^{(R1,\ R2,\ SCC)} \quad /*\ \text{RH:Role Hierarchy}\ */\\
&EA \subseteq E \times R \quad /*\ \text{EA:Entity Assignment. E:Entity, R:Role}\ */\\
&PRMS \subseteq 2^{Ops*Obs} \quad /*\ \text{PRMS:Permission, Obs:Object, Ops:Operation}\ */\\
&PA \subseteq R \times PRMS \quad /*\ \text{PA:Permission Assignment}\ */\\
&EAC \subseteq EA \times DCC \quad /*\ \text{EAC:Entity Assignment Context Condition}\ */\\
&PAC \subseteq PA \times SCC\\
&\qquad /*\ \text{PAC:Permission Assignment Context Condition}\ */\\
&\text{Check\_DCC} : DCC \rightarrow \text{True/False}\\
&\text{Find\_Entities} : R \rightarrow \text{Entities}\\
&\text{Find\_Entities}(R) = \{E \mid (E, R) \in EA,\\
&\qquad ((E, R), DCC) \in EAC\\
&\qquad \text{where Check\_DCC}(DCC) = \text{True}\}\\
&\qquad \subseteq 2^{E}\\
&\text{Find\_Roles} : E \rightarrow \text{Roles}\\
&\text{Find\_Roles}(E) = \{R \mid (E, R) \in EA,\\
&\qquad ((E, R), DCC) \in EAC\\
&\qquad \text{where Check\_DCC}(DCC) = \text{True}\}\\
&\qquad \subseteq 2^{R}\\
&\text{Find\_Permissions} : R \rightarrow \text{Permissions}\\
&\text{Find\_Permissions}(R) = \cup\{PRMS \mid (R'', PRMS) \in PA,\\
&\qquad \forall R'' \in \{R\} \cup \{R' \mid (R, R', SCC) \in RH\}\}
\end{aligned}
$$

3. 정책 기술 언어

유비쿼터스 환경에서 정책 기술 언어는 상황으로써 데이터 표현과 상황 적응 규칙, 접근 제어 규칙으로 구성된다. 따라서 각 엔티티와 그들의 관계 등을 상황으로

기술할 수 있고, 상황으로 기술한 조건이 만족하는 경우 해당 액션의 수행이나 환경의 변화를 기술할 수 있으며, 상황으로 기술한 조건이 만족하는 경우 엔티티에 대한 접근 권한을 기술할 수 있다.

3.1 엔티티

유비쿼터스 환경에서 상황 엔티티란 물리적 또는 논리적인 공간이나 고정된 객체 또는 이동성이 있는 객체 등을 모두 포함한다[6]. 예를 들어, 한국항공대학교의 상황 엔티티는 표 2와 같은 것들로 생각할 수 있다.

표 2. 한국항공대학교의 상황 엔티티 예

물리적 또는 논리적 공간 : 도서관, 전자관, 2층, 205호...
고정된 객체 : 프린터, 복사기, 빔 프로젝터, 에어컨...
이동성이 있는 객체 : PDA...

3.2 엔티티 관계 정의

엔티티 관계 정의의 문법은 표 3과 같다. id1!id2(id3)에서 id2는 관계의 이름이고 id1과 id3은 엔티티 클래스의 이름이다[6]. 예를 들어, Pda!IsIn(Room)은 Pda 타입의 엔티티와 Room 타입의 엔티티 사이에 IsIn 관계가 있을 수 있다는 의미이다.

표 3. 엔티티 관계 정의 문법

id ∈ Identifier
c ∈ Context-Relation ::= id1!id2(id3) | c1; c2
s ∈ Space-Relation ::= id | id1 : id2 | id[s] | id1 : id2[s] | s1 + s2 | ε

특히 공간과 고정된 객체는 "[]"를 사용하여 계층적인 구조를 표현할 수 있다.

```
Building:elec!Contains(Floor:f2)
Floor:f2!Contains(Lab)
Floor:f2!Contains(Lectureroom)
Lab!Contains(Printer)
Lectureroom!Contains(BeamProjecter)
```

위의 5개 Contains관계로 표현된 것을 다음과 같이 간략하게 표현할 수 있다.

```
Building:elec[Floor:f2[Lab:205[Printer]+
Lectureroom[BeamProjecter]]]
```

3.3 엔티티 표현

하나의 엔티티를 표현하는 문법은 표 4와 같다[6].

id_1:id_2에서id_1은 엔티티 클래스의 이름이고 id_2는 엔티티 클래스의 객체를 나타낸다. $id는 id클래스의 한 객체를 나타내는 일종의 변수이다. *는 임의의 엔티티를 나타내며 /은 공간의 포함 관계를 나타낸다.

```
Building:elec/.../$Lab
```

이것은 전자관(elec)내의 한 연구실 객체를 의미한다.

표 4. 엔티티 표현 문법

n ∈ Number
p ∈ Entity-Expression ::= id_1:id_2 | $id | $id_n | * | id_1:id_2/p | $id/p | $id_n/p | .../p

3.4 상황 관계 표현

상황 관계 표현은 엔티티들 사이의 관계 조건을 표현한다[6]. 관계 조건 표현은 해당 상황에서 참/거짓으로 해석될 수 있다. 상황 관계 표현의 문법은 표 5와 같다.

p_1!id(p_2)에서 p_1과 p_2는 엔티티 표현이고 id는 관계 이름이다. 예를 들어, Pda가 연구실 205호 안에 있는지에 대한 관계 표현은 $PDA!IsIn(Lab:205)로 표현된다.

표 5. 상황 관계 표현 문법

re ∈ Relation-Expression ::= p_1!id(p_2) | ~ re | re_1 ^ re_2

3.5 적응 규칙

적응 규칙은 동적으로 상황 정보의 변화가 일어날 때 이에 대한 적응 동작을 명세 한다[6]. 적응 규칙의 문법은 표 6과 같다[6].

표 6. 적응 규칙 문법

d ∈ Adaptation-Rule ::= re => a if b | d1 d2
a ∈ Action ::= p1:id(p2) | p1!id(p2) | a1 a2

문법에 있는 b는 자바의 조건적인 표현이거나 상황 관계 표현이다. 다음은 적응 규칙의 예이다.

```
Location($Pda_1,IsIn,$PCRoom)^Location($Pda_2,IsIn,$
PCRoom)=>
    Relationship($Pda_2,Host,$Pda_1)

Location($Pda,IsIn,$PCRoom)=>
    $Pda.registerPrinter($PCRoom/$Printer)
```

첫 번째는 $Pda_1이 $PCRoom안에 있고 $Pda_2가 같은 $PCRoom안에 있으면 $Pda_1과 $Pda_2사이에

Host관계가 설정된다. 두 번째는 $Pda가 $PCRoom안에 있으면 $Pda는 registerPrinter 메소드를 호출하여 $PCRoom안에 있는 프린터를 등록한다.

4. 역할 기반 정책 기술 언어

4.1 역할 엔티티

역할은 특별한 엔티티로써 표현 문법은 표 7과 같다.

표 7. 역할 엔티티 표현 문법

```
rt ∈ Role_Type ::= Role | id
rl ∈ Role ::= rt:id | $rt | $rt_n
```

예를 들어, rt는 Role이나, Lecturer, Professor, TA 등과 같은 것이 되고 rl은 $Professor, $Lecturer, $Student, Lecturer:CS101, Professor:ProfIT 등으로 특정 역할을 표현할 수 있다. rt의 Role은 슈퍼 엔티티로써 누구나 갖는 역할을 나타낸다.

4.2 적응 규칙을 이용한 역할 할당

동적인 상황에 적합한 역할 할당을 위해 앞서 설명한 적응 규칙을 이용하여 기술할 수 있다. 다음은 역할 할당 기술의 예이다.

```
Pda:Tom!hasRole(Lecturer:cs101lec)

$Pda_1!IsIn($Room)^$Pda_2!IsIn($Room)^$Pda_2!Owns(
$Room)^$RoomGuest!For($Room)=>
    $Pda_1!hasRole($RoomGuest)

Env:cenv.getDate()>20070801=>
    Pda:Tom!hasRole(Lecturer:cs101lec)
```

첫 번째는 항상 Tom이 Lecturer:cs101lec의 역할을 할당 받는다는 것이고, 두 번째는 $Pda_1이 방안에 있고 $Pda_2가 같은 방안에 있으며, $Pda_2가 그 방의 주인이고 $RoomGuest역할이 그 방의 대한 역할일 때 $Pda_1은 $RoomGuest역할을 할당받는다. 세 번째는 현재 날짜가 2007년 8월 1일 이후인 경우 Tom은 Lecturer:cs101lec역할을 할당받는다.

4.3 접근 제어 규칙

접근 제어 규칙은 역할에 대한 권한 부여를 기술하기 위한 규칙이다. 그 문법은 표 8과 같다. 다음은 접근 제어 규칙을 기술한 예이다.
첫 번째는 $RoomGuest역할이 그 방의 프린터의 프린트 메소드를 호출할 수 있는 권한을 기술한 것이며, SCC로써 $RoomGuest가 그 방에 대한 역할인 경우 활성화된다. 앞서 설명한 것처럼 SCC는 시스템 관리자 단계에서 판별되므로 역할과 권한 사이의 반복적인 조건 검사가 필요 없다. 두 번째는 $RoomGuest역할이 전원 스위치를 켜고 끌 수 있는 권한을 명시한 것이다. 세 번째는 $Lecturer역할이 수강생의 학점을 입력할 수 있는 권한을 기술한 것이며, $Lecturer가 강의에 대한 역할이고 $Listener도 같은 강의에 대한 역할이며 $Pda가 $Listener의 역할을 가지고 있는 경우 활성화된다. 이 경우도 SCC는 시스템 관리자 단계에서 판별됨으로 시스템의 효율을 향상시킨다.

```
($RoomGuest,$Room/$Printer.print,$RoomGuest!For($R
oom),CALL)

($RoomGuest,$Room/$Switch.turn,$RoomGuest!For($Ro
om),CALL)

($Lecturer,$Listener.putGrade(),$Lecturer!For($Lect)^$
Pda!hasRole($Listener)^$Listener!For($Lect),CALL)
```

표 8. 접근 제어 규칙

```
x ∈ Access Rule ::= (rl, Obs, SCC, Ops) | x1 x2
Obs ::= p.id | p1!id(p2)
Ops ::= READ | WRITE | CALL
```

4.4 역할 계층 규칙

역할 계층 규칙은 역할과 권한의 관계를 간략화 하는데 유용하게 사용되어 진다. 주로 정적인 상황에 의존하여 기술되며 그 문법은 표 9와 같다.

표 9, 역할 계층 규칙

```
rh ∈ Role Hierarchy ::= (R1, R2, SCC) | rh1 rh2
```

다음은 역할 계층 규칙을 기술한 예이다.

```
($Lecturer,$TA,$Lecturer!For($Lect)^$TA!For($Lect))

(Lecturer:cs101lec,TA:cs101ta,true)
```

첫 번째는 $Lecturer역할은 $TA와 같은 $Lect에 대한 역할일 때 $TA보다 상위 역할임을 명시한 것이다. 두 번째는 역할 클래스의 객체로 기술한 것이다. 이 의미는 cs101lec의 역할은 cs101ta의 모든 권한을 갖는다는 의미이다.

4.5 예제

그림 2는 제안한 정책 명세 언어를 이용하여 기술한 간단한 예제를 나타낸다. 두 명의 사람과 강의교수와 조

교, 수강생의 역할이 있는 시나리오에서 엔티티를 기술하고 상황 관계와 역할 간의 계층을 기술하여 어떻게 역할이 할당되는지 접근 제어 규칙은 어떻게 되는지 기술한 예이다.

```
- 엔티티 표현 -
일반 엔티티
Pda:Ahn, Pda:Shin
Building:elec[Floor:f2[Lab[Printer]+
Lectureroom[BeamProjecter]]]

역할 엔티티
Listener, Lect
Lecturer:CS218Lec, TA:CS218Ta

- 상황 관계 표현 -
$PDA!IsIn($Lab), $PDA!IsIn($Lectureroom)
$PDA!IsIn($Building), $Pda!hasRole($Listener)
$Lecturer!For($Lect), $Listener!For($Lect)
$TA!For($Lect)

- 역할 계층 규칙 -
($Lecturer,$TA,$Lecturer!For($Lect)^$TA!For($Lect))

- 적응 규칙을 이용한 역할 할당 -
Pda:Ahn!IsIn($Building)=>Pda:Ahn!hasRole(Lecturer:C
S218Lec) if Env:cenv.getDate()>20070901

Pda:Shin!IsIn($Lectureroom)^Pda:Ahn!IsIn($Lecturero
om)=>Pda:Shin!hasRole(TA:CS218Ta)

- 접근 제어 규칙 -
($Lecturer,$Listener.putGrade(),$Lecturer!For($Lect)
^$Pda!hasRole($Listener)^$Listener!For($Lect),CALL)
($TA,$Lectureroom/$BeamProjecter.turn,true,CALL)
($TA,$Lab/$Printer.print,true,CALL)
```

그림 2. 예제

5. 결론

본 논문에서 제시한 상황 인식 역할 기반 접근 제어 모델은 상황 정보를 동적인 상황과 정적인 상황을 구분하여 활용함으로써 동적인 상황 조건에 따라 사용자에게 역할을 할당하고 정적인 상황 조건을 이용하여 역할에 권한을 부여하기 때문에 상황 정보를 보다 쉽고 효율적으로 처리할 수 있다. 제안된 정책 기술 언어는 상황 인식 역할 기반 접근 제어의 장점을 활용하여 이 모델에 기반한 유비쿼터스 환경을 기술할 수 있다.

향후 연구로써 제안한 언어를 사용하여 상황 인식 역할 기반 접근 제어 시스템을 구현한다. 제시한 모델은 엔티티가 자신의 프라이버시를 따로 관리할 수 없기 때문에 이를 서비스할 수 있는 방법[8, 9, 10]을 연구한다. 그리고 이 모델을 기반으로 유용성 검사[11]와 정적 분석 방법[12]을 연구한다.

6. 참고 문헌

[1] R. Sandu and P. Samarati, Access Control: Principles and Practice, IEEE Communication Magazine, pages40~48, 1994.

[2] R. S. Sandhu, Role-Based Access Control, In Advances in Computers, volume 46, Academic Press, 1998.

[3] Muhammad Nabeel Tahir, C-RBAC: Contextual Role-Based Access Control Model, UBICC Journal, Volume 2 No. 3, 2007.

[4] Arun Kumar and Neeran Karnik and Girish Chafle, Context sensitivity in role-based access control, ACM SIGOPS Operating Systems Review archive, Volume 36, Issue 3, Pages53-66. 2002.

[5] SHEN Haibo and HONG Fan, A Context-Aware Role-Based Access Control Model for Web Services, Proceedings of the IEEE International Conference on e-Business Engineering, Pages220-223, 2005.

[6] Joonseon Ahn and Byeong-Mo Chang and Kyung-Goo Doh, A Policy Description Language for Context-based Access Control and Adaptation in Ubiquitous Environment, 2006.

[7] J. E. Bardram, The Java Context Awareness Framework-A Service Infrastructure and Programming Framework for Context-Aware Applications, Third Inter national Conference, Pervasive 2005, Munich, Germany, May, 2005.

[8] J. Hong and J. Landay, An architecture for privacy-sensitive ubiquitous computing, In Proceedings of International Conference on Mobile Systems, Applications, and Services, 2004.

[9] Urs Hengartner and Peter Steenkiste, Avoiding Privacy Violations Caused by Context-Sensitive Services, PERCOM'06, Volume 00, Pages222-233, 2006.

[10] Gautham Pallapa and Nirmalya Roy and Sajal Das, Precision: Privacy Enhanced Context-Aware Information Fusion in Ubiquitous Healthcare, SEPCASE'07, 2007.

[11] Paolina Centonze and Gleb Naumovich and Stephen J. Fink and Marco Pistoia, Role-Based access control consistency validation, ISSTA, 2006.

[12] E. Cho and K. Lee, Security Checks in Programming Languages for Ubiquitous Environments, Proceedings of 2004 Workshop on Pervasive, Security, Privacy and Trust, 2004.