情報保護學會論文誌
第15券 第2號, 2005. 4

유비쿼터스 컴퓨팅 환경의 역할 기반 접근제어에서 발생하는 상황 충돌

남 승 좌,[1†] 박 석[2‡]
[1]LG 전자, [2]서강대학교

Context Conflicts of Role-Based Access Control in Ubiquitous Computing Environment

Seung-Jwa Nam,[1†] Seog Park[2‡]
[1]LG Electronics, [2]Sogang University

요 약

기존의 응용에 따라 보호 객체들에 대한 접근을 역할들로 분류하여 역할을 중심으로 접근제어를 수행하는 역할 기반의 접근제어에 사용자 및 환경 정보를 이용한 정보 접근제어 기법이 연구되고 있다. 중요 정보와 자원에 대하여 상황 정보를 이용하여 사용자와 환경에 유연하면서 강력한 접근제어의 수행에 관하여 연구되고 있다.

본 논문에서는 상황에 대한 정의와 중요한 의미 정보에 대하여 사용자와 자원, 환경을 고려한 유연한 접근제어 방법을 제시하고, 이때 발생할 수 있는 접근에 대한 충돌을 찾아내고 해결 방안을 제안한다. 상황 정보의 분류와 정의를 이용하여 상황 정보를 접근제어에 적용하는 방법과 상황 정보를 접근제어에 이용하였을 경우 발생할 수 있는 충돌을 분류하여 해결 방안을 제시한다. 이 논문은 유비쿼터스 컴퓨팅 환경에서 상황 정의와 분류를 이용하여 적절한 사용자가 적절한 객체와 어플리케이션 서비스의 사용을 보장하는 보안 정책과 모델의 개발을 위한 기초 연구이다. 권한이 있는 사용자에 의한 객체 접근의 단순 접근제어가 아니라 사용자는 자신의 상황과 연관된 객체, 자원, 서비스에 접근할 수 있음을 보장한다.

ABSTRACT

Traditional access control models like role-based access control model are insufficient in security needs in ubiquitous computing environment because they take no thought of access control based on user's context or environment condition. In these days, although researches on context-aware access control using user's context or environment conditions based on role-based access control are emerged, they are on the primary stage.

We present context definitions and an access control model to provide more flexible and dynamic context-aware access control based on role-based access control. Specially, we describe the conflict problems occurred in the middle of making an access decision. After classifying the conflict problems, we show some resolutions to solve them.

접수일 : 2004년 12월 3일 ; 채택일 : 2005년 3월 28일
* 본 연구는 정보통신부 정보통신연구진흥원에서 지원하고 있는 정보통신기초기술연구 지원사업(B1220-0401-0358)의 연구 결과의 일부임.
† 주저자 : chico@lge.co.kr
‡ 교신저자 : spark@sogang.ac.kr

In conclusion, we will lay the foundations of the development of security policy and model assuring right user of right object(or resource) and application service through pre-defined context and context classification in ubiquitous computing environments. Beyond the simplicity of access to objects by authorized users, we assure that user can access to the object, resource, or service anywhere and anytime according to right context.

Keywords : *Ubiquitous computing, role-based access control, context, context conflict*

Ⅰ. 서 론

다종의 다바이스들이 현실 세계에 스며들어 상호 연결되고, 인간-사물-정보 간에 언제, 어디서나 통신이 가능한 유비쿼터스 컴퓨팅(Ubiquitous Computing)은 사용자 및 환경을 인식(context aware)하고, 언제 어디서나 어떤 디바이스를 통해서도 끊이지 않는(seamless) 서비스를 제공한다[1]. 사용자와 사용자의 환경을 인식하여 시공간의 제약을 받지 않고 사용자가 원하는 정보를 자동적으로 제공할 수 있다. 그러나 시공간의 제약 없이 정보에 접근하는 것은 정보에 대한 보안을 위협한다.

사용자의 상황을 인식하여 상황에 적합한 접근제어를 수행하기 위해서 기존의 접근제어 기법에 상황 정보를 추가하여 접근제어를 수행할 수 있다. 예를 들어, 병원의 모든 정보시스템과 기기들이 자동적으로 처리될 수 있는 컴퓨팅 능력과 통신 능력을 갖는 의료 환경에서 사용자에 따라 다른 접근제어 수준을 제공할 수 있다. 주요 자원과 기기에 대하여 일반인의 접근을 막을 수도 있고, 의료진의 접근을 허용할 수 있다. 이러한 접근은 사용자의 역할에 따라 접근제어를 수행할 수 있다. 그러나 환자의 과거병력이나 진료 정보 등과 같이 중요 기밀 정보에 대해서는 인가된 사용자라 할지라도 언제, 어디서나 정보의 접근이 허용되어서는 안 된다. 접근하고자 하는 정보가 필요한 상황에만 인가된 사용자에게 접근이 허용되어야 하고 이를 사용자와 주변의 상황 정보를 이용하여 제어할 수 있다.

그 동안 기존 역할 기반 접근제어에 상황정보를 이용하는 연구가 진행되고 있다. 그러나 특정 어플리케이션을 위한 상황 정보의 정의를 이용하여 일반화된 정의와 구조를 사용하지 않고 있다. 또한, 상황 정보를 이용하였을 경우 발생하는 충돌에 대한 고려가 없고, 해결방안도 제시하지 못하고 있다.

본 논문에서는 상황 정보를 접근제어에 이용하기 위하여 일반화된 상황 정보의 정의와 계층구조를 이용한 구조를 정의하고, 일반화 정의와 구조를 이용하여 접근제어를 수행하기 위하여 기법을 제시한다. 또한, 상황에 따른 유연한 접근제어를 위하여 상황 정보를 이용하여 접근 제어를 수행하였을 경우 발생하는 충돌들을 살펴보고, 해결방안을 제시한다. 충돌의 분류와 해결을 통하여 사용자는 상황에 적합한 권한을 부여 받을 수 있는 유연한 접근제어가 가능해진다.

본 논문의 구성은 다음과 같다. 2장에서는 유비쿼터스 컴퓨팅과 상황에 대한 정의, 역할 기반 접근제어와 상황 정보를 이용한 접근제어 등 기존 관련 연구들을 살펴보고, 3장에서는 연구동기, 상황 정의와 제안하는 접근제어 기법 제시하고, 충돌의 분류와 충돌 해결 기법에 관해 살펴본다. 4장에서는 이러한 제안하는 접근제어 기법과 충돌 해결을 다른 접근제어 기법과 비교하여 평가한다. 마지막으로 5장에서는 결론 및 추후 연구를 기술한다.

Ⅱ. 관련연구

정보에 대한 접근제어를 위한 연구는 오래 전부터 연구되어 왔으며 역할 기반 접근제어를 중심으로 연구되어 왔다. 최근에는 새로운 유비쿼터스 컴퓨팅 환경에서 사용자의 상황을 고려한 접근제어에 대한 연구가 진행되고 있다.

2.1 상황 정보(context information)

Anind K. Dey과 Gregory D. Abowd는 상황 정보를 사용자와 응용 서비스 사이의 상호 작용으로 인해 필요한 사용자, 장소, 대상물 등의 개체 상태를 나타내는 정보로 정의하였다[2,3]. Thomas는 상황 정보를 사용자와 객체의 환경을 제외한 접근 요청이 발생한 위치나 접근될 객체의 위치와 같은 위치 정보와 접근 요청이 발생한 시간이나 시간 간격 등의 시간 정보뿐만 아니라 특정 행동을 위해 필요로 하는 데이터로 정의[4]하고, Masera는 접근제어를 수행하기 위한 사이트, 도메인과 같은 위치 정보를 상황 정

보로 정의[5]하고, 시간 제약사항을 두었다. 즉 위치와 시간 정보를 이용하여 접근제어를 수행 한다.

2.2 역할기반 접근제어(RBAC : Role-Based Access Control)

RBAC 모델은 접근 결정이 개개인의 사용자들이 조직원으로서 요구되는 역할에 근거하여 이루어진다. 접근 권한은 역할 이름에 따라 그룹화 되고, 이러한 역할이 개개인의 업무적 책임과 권한에 따라 사용자에게 부여됨으로써 자원의 이용이 제약된다. 접근을 통제하기 위해 역할을 사용하는 것은 기업 특성에 맞는 보안 정책을 시행하고 개발하며 보안 관리를 능률화 하는데 효과적인 기법이 된다[6,7]. 또한 정보에 대한 통제가 제한된 수의 관리자에 의해 수행됨으로써 사용자가 대단히 많은 실제 기업 환경에 효율적으로 적용할 수 있는 등 RBAC이 가지는 여러 가지 장점으로 인해 활발히 연구되었다.

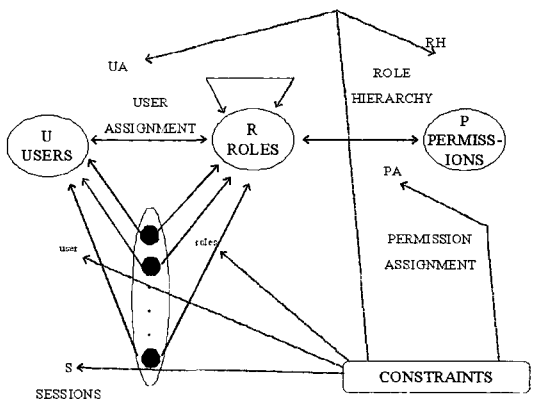

그림 1. RBAC96 모델

그림 1과 같이 역할은 부분 순서(partial order)를 가지고 계층 구조를 형성한다. 이때 계층구조 상에서 상위 역할은 하위 역할의 권한을 모두 상속 받는다. 세션(Session)은 한 사용자와 여러 개의 역할들의 집합으로 표현되는데, 사용자는 세션을 통해 자신에게 지정된 역할들 중 일부 또는 전체를 활성화(Activate)하여 수행할 수 있다. 제약사항은 모델의 각 구성요소 매핑에 적용되는 것으로 의무분리(SOD: Seperation of Duty), 한 역할에 지정될 수 있는 최대/최소 사용자의 수(cardinality), 최소권한의 원리(least privilege principle) 등이 있다.

다양한 상황 정보를 어플리케이션에서 얻을 수 있는 유비쿼터스 컴퓨팅 환경에 RBAC을 적용하기 위한 연구들이 진행되고 있다.

2.3 상황 정보를 이용한 접근제어

□ GRBAC : Generalized Role-Based Access Control

기존의 역할기반 접근제어는 시간에 따른 접근제어 등과 같이 상황에 근거한 접근제어를 수행할 수 없어 상황에 근거한 접근제어를 수행하기 위하여 GRBAC 모델을 제안하였다. GRBAC 모델은 접근제어 결정에 사용자 역할(subject role), 객체 역할(object role), 환경 역할(environment role)을 사용함으로써 기존 역할기반 접근제어를 확장하였다. 사용자, 객체, 환경 요소를 역할로 구조화함으로써 접근제어 정책 기술의 단순함(simplicity)과 융통성(flexibility)을 제공한다[8,9].

이 모델은 상황 정보를 환경 역할로 표현하여 접근제어를 위한 권한 정책에 기술하고 전파 규칙을 적용하여 사용자의 접근 요청을 처리한다. 그러나 접근 권한의 전파로 발생하는 권한들 사이의 충돌에 대하여 해결 방안을 제시하지 않는다. 또한, 사용자가 사용자의 고려사항을 환경 역할을 정의함에 많은 계층 구조가 발생하여 관리의 어려움이 발생한다.

□ xoRBAC : 상황 제약사항을 사용한 역할기반 접근제어

Gustaf Neumann과 Mark Strembeck는 상황 정보를 접근제어 결정에 이용하기 위하여 역할 기반 접근제어의 제약사항을 사용하는 xoRBAC 모델을 제안하였다. 상황 제약사항(context constraint)은 상황 정보 속성의 실제 값을 미리 정의된 조건과 체크하는 역할 기반 접근제어 제약사항이다[10].

상황 제약사항은 특정 연산의 수행을 허용하기 위해 상황 정보 속성이 충족 되어야 할 조건을 기술한다. 상황 제약사항은 속성(context attribute), 함수(context function), 조건(context condition)의 튜플(tuple)을 갖는다. 속성은 시간, 요일과 같은 동적으로 변화하는 속성을 나타내거나 위치, 소유관계, 생일이나 국적과 같은 객체의 인스턴스(instance)에 따라 변화하는 속성을 나타낸다.

권한 결정은 특정 주체 또는 역할이 가지는 권한 및 상황 제약사항에 따라 이루어진다. 권한은 여러 개의 상황 제약과 관련되고 모든 상황 제약이 참 값을 가질 때 접근이 허용된다.

이 모델은 상황 정보를 상황 정보 제약에 기술하고 각 권한에 대하여 상황 정보 제약을 둔다. 사용자의 접근 요청은 해당 객체에 대한 연산의 권한이 갖는 상황 정보 제약이 모두 참 값을 가질 때 허용 또는 거부된다. 따라서 각 객체의 권한에 대하여 상황 정보제약이 기술되므로 사용자의 상황에 따른 접근제어를 수행하고 자 할 경우 최악의 경우 사용자수 $*2^{상황의\ 수}$의 제약사항에 대한 기술이 필요하다. 이것은 사용자의 접근요청이 발생하였을 때 권한을 부여하기 위하여 상황 정보제약의 탐색과 평가 시간을 길게 하는 문제를 발생시킨다. 또한, 각 객체의 권한을 부여 받기 위하여 만족되어야 할 조건을 기술하므로 상황정보의 일반화된 정의와 구조를 사용하지 않으므로 관리의 어려움이 따른다.

Ⅲ.제안하는 기법

3.1 연구동기

유비쿼터스 컴퓨팅 환경에서 사용자는 시간과 공간의 제약을 받지 않고 원하는 정보를 제공 받을 수 있다. 하지만 사용자가 원하는 모든 정보가 제공되는 것은 정보의 기밀성에 문제를 초래한다. 따라서 유연한 정보 접근과 보안 충족이라는 상반되는 요구사항을 만족시킬 수 있는 접근제어가 필요하다. 높은 보안 수준을 요구하는 정보의 접근에 높은 제약이 부여되고, 낮은 보안 수준을 요구하는 정보의 접근에는 낮은 제약이 부여되는 보안 수준에 따른 정보의 접근제어가 필요하다.

정보의 보안 수준에 따른 접근제어를 위해서는 접근권한이 있는 사용자일지라도 사용자의 상황에 따라 접근 가능한 정보가 제한되어야 한다. 사용자의 역할에 따라 정보의 접근을 제어하는 기존의 접근제어 기법으로는 사용자의 상황에 따른 정보의 접근을 제어할 수 없다. 예를 들어, 의사 역할을 할당 받는 사용자는 모든 정보에 접근이 가능하다. 그러나 의사는 진료, 치료, 진단, 처방 등과 같은 상황에 따라 필요로 하는 정보가 다르다. 따라서 필요하지 않은 정보에 접근이 허용되는 것은 정보 기밀성에 문제를 발생시키고 보안을 위협한다.

이러한 필요성에 의해 사용자의 상황을 고려한 역할 기반 접근 제어 연구가 진행되었으나 다양한 상황 정보에 대한 일반화된 정의와 정형화된 구조[7,8]를 정의하지 않아 관리상의 문제와 권한의 전파에 대해 상황 정보를 고려하지 않고, 충돌에 대하여 해결 방안을 제시하지 못하고 있다.

본 논문에서는 정보의 보안 수준에 따른 접근제어를 수행하기 위한 상황정보의 정의와 구조화를 통해 상황 정보를 이용한 접근제어 기법과 상황 정보를 접근제어에 이용하였을 경우 발생하는 충돌문제와 상황 정보를 고려한 해결방안 제안하고자 한다.

3.2 환경 및 가정

본 논문의 가정과 환경은 다음과 같다.

□ 가정
- 객체에 대한 연산은 읽기(read)와 쓰기(write)이 허용된다.
- 상황이 활성화 되었을 때, 참(true)을 리턴(return)한다.
- 사용자의 접근 요청은 〈Subject, Object, operation〉의 트랜잭션과 상황 정보가 함께 전달된다.
- 접근 권한 정책은 〈Subject, Object, operation, Context, sign〉의 순서로 기술된다.
- 권한은 전파 원칙을 따른다.

□ 환경
- 주체(subject)

객체의 접근을 요청하는 사용자는 조직 내에서의 임무나 책임, 조직구조에서의 위치, 보안등급에 따라 계층 구조(subject hierarchy)를 갖는다. 사용자는 그림 2와 같은 계층구조 상의 역할(role)을 할당 받는다. 그림 2는 병원에서의 사용자가 할당 받을 수 있는 역할에 대한 계층구조를 나타낸 것이다.

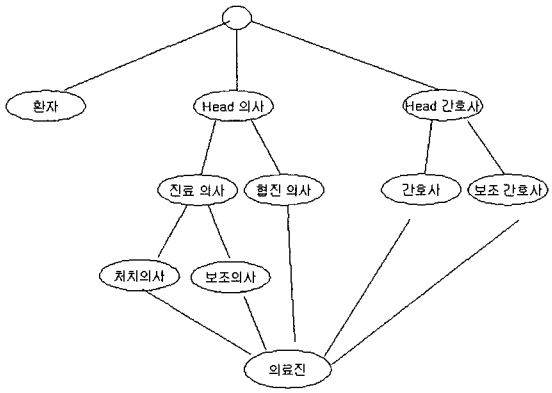

그림 2. 병원에서의 주체 역할의 계층구조

- 객체(object)

사용자가 접근을 요청하는 객체들 사이에는 포함관계가 존재한다. 예를 들면, 그림 3과 같이 전자 의료 기록에는 환자의 진료와 처치에 대한 전반적인 내용을 기록하는 의무기록이 있고, 의무기록에는 진료기록과 검사기록, 투약기록이 포함된다.

포함관계를 갖는 객체들 사이에는 서로 다른 보안등급을 갖는다. 모든 사용자의 접근을 허용하는 정보가 있는가 하면, 특정 상황에서 권한을 갖는 사용자의 접근을 허용하는 정보도 존재한다.

그림 3. 병원 객체간의 포함관계

이러한 포함관계가 존재하는 객체들 사이에서 객체의 보안등급에 따른 계층구조를 구성할 수 있다. 예를 들면, 진료기록, 검사기록, 투약기록을 포함하는 의무기록에서 객체들의 보안등급에 따라 계층구조를 구성할 수 있다. 진료기록은 진료 일시, 진료의사, 진료과, 주 호소증상, 진단병명 등의 내용과 협진의, 의뢰내용과 회신 내용 등을 기록하는 협진기록, 임상소견, 신체검진을 기록하는 소견기록과 처치 일시와 처치의, 처치 내용 등을 기록하는 처치 기록을 포함한다. 검사기록에는 검사명, 의뢰의, 검사 결과, 결과보고일, 임상병리사 등이 기록되고, 투약기록에는 투약 일시, 투약인, 투약 약명 리스트 등과 처방 일시, 처방의, 처방 약명, 주의사항 등이 기록된 처방기록이 포함된다. 진료기록에서 소견기록과 처치기록은 다른 정보에 비해 높은 보안 수준을 요구하고, 소견기록에서 협진기록은 더 높은 보안 수준을 요구한다. 검사기록 중 검사 결과, 투약기록 중 처방기록이 다른 정보에 비해 높은 보안 수준을 요구한다. 의무기록에 포함된 기록들을 포함관계와 보안 등급에 따라 계층 구조를 형성하면 그림 4와 같다.

위와 같이 환자의 개인정보와 병력기록, 활력기록을 기록하는 환자기록에 대하여 객체간의 포함관계와 보안등급에 따라 그림 5와 같이 계층구조를 구성할 수 있다.

서로 다른 보안등급을 갖는 객체들의 포함관계에 의해 구성되는 객체 계층구조에 따라 객체에 대한 연산 권한이 전파(propagation)된다. 허용권한은 더

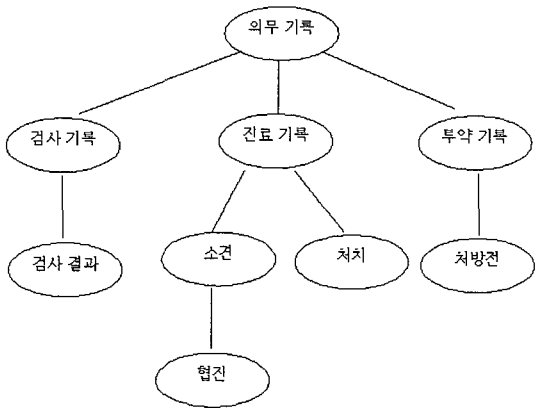

그림 4. 의무기록의 계층구조

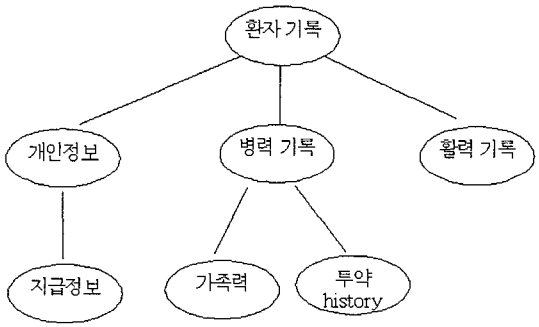

그림 5. 환자기록의 계층구조

높은 보안을 요구하는 객체에 대한 연산이 허용되면 낮은 보안등급을 갖는 객체에 대하여 읽기와 쓰기가 허용됨을 의미하고, 거부권한은 낮은 보안등급의 객체에 읽기와 쓰기가 거부되면 더 높은 보안등급을 요구하는 객체에 대해서도 읽기와 쓰기 연산이 거부됨을 의미한다.

표 1. 객체에 대한 권한 전파 규칙[1)]

	허용 권한	거부 권한
읽기 연산	↑	↓
쓰기 연산	↑	↓

예를 들면, 의사가 소견기록에 쓰기 권한을 가지고 있다면 진료 기록에 대해서도 쓰기 권한이 허용된다. 또한, 간호사가 검사기록의 읽기가 허용되지 않는다면 더 높은 보안등급을 갖는 검사 결과에 대해서

1) ↑은 객체 계층에 대하여 권한 전파의 방향이 아래(높은 보안 등급)에서 위(낮은 보안등급)로 향함을 의미하고, ↓은 그 반대를 의미한다.

도 읽기가 허용되지 않는다.

3.3 상황 (context)의 정의와 계층 구조

□ 기본 상황 정의

정의 1. 차원 (dimension) : 시간, 위치와 같이 상황을 구분하는 기준으로 상황 정보들이 갖는 속성의 분류이다.

각 차원에 대하여 기본 상황이 정의된다.

정의 2. 기본 상황(basic context) : 기본 상황은 차원을 구성하는 상황 정보로 원자 값을 갖는 상황 정보와 원자 값을 갖는 상황 정보를 그룹으로 묶는 상황 정보이다.

예를 들어, 시간 차원에 속하는 시간, 날짜, 요일 등의 상황정보는 시간 기본 상황에 포함될 수 있고, 위치 차원에는 센서가 포함된 사용자가 위치할 수 있는 위치 정보가 위치 기본 상황으로 포함될 수 있다.

□ 상황 계층 구조

각 차원에 속하는 기본 상황은 상황 정보의 포함 관계에 따라 상황 계층구조를 구성할 수 있다. 상황 계층구조의 하위에 존재하는 상황은 상위에 존재하는 상황에 비해 상세적(detailed)이다. 예를 들면, 그림 6과 같은 구조를 갖는 병원의 일부는 그림 7과 같은 계층구조를 갖는다. 병실은 사용자의 위치가 병원에 있다는 정보보다 더 상세적이고, 의국(환자 zone)은 병실이나 병원보다 더 상세적이다. 따라서 상황의 발생의 전파 방향은 하위에서 상위로 향하고 특정 하위 상황의 발생은 상황 계층구조 상에서 그 상황의 상위 상황의 발생을 일으킨다.

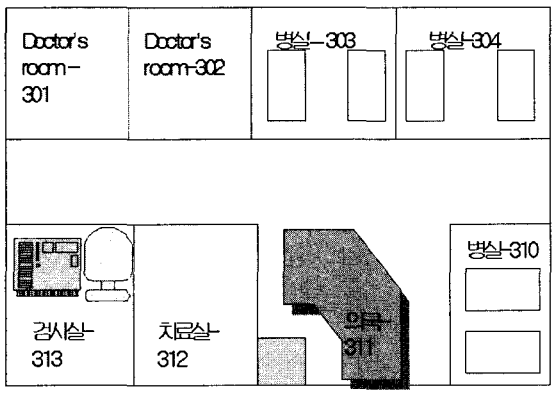

그림 6. 병원의 구조

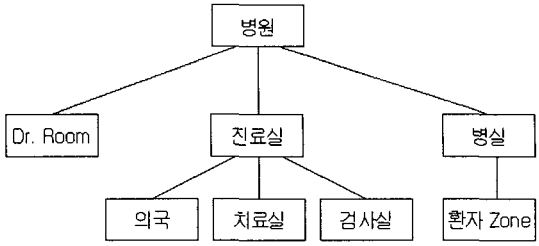

그림 7. 병원 위치 상황의 계층구조

□ 사용자 정의 상황 정의

정의 3. 사용자 정의 상황(user defined context) : 사용자 정의 상황은 시스템 관리자나 사용자가 원하는 상황에 정보의 접근을 허용하거나 거부하는 것을 기술하는데 사용되는 상황 정보이다.

사용자 정의 상황은 상황 정보의 구성요소의 수에 따라 크게 단일 상황과 복합 상황으로 구분된다.

정의 4. 단일 상황(single context) : 하나의 기본 상황과 조건을 속성으로 갖는 상황 정보를 단일 상황(single context)라 정의하고, 다음과 같이 타입, 상황, [조건]의 구성요소를 갖는 형태로 표현된다.

Context(name) = type: basic context [condition]

여기에서 type은 기본 상황이 포함되는 차원을 나타낸다.

예를 들면, context(근무요일)= T: weekdays, context(근무시간)= T : 9:00 ≤ 시간 ≤ 18:00으로 근무요일과 근무시간의 사용자 정의 상황을 정의할 수 있다. 근무요일은 시간(T) 차원의 weekdays 기본 상황으로 표현되고, 근무시간은 시간 차원의 9:00~18:00사이의 시간으로 표현된다.

정의 5. 복합 상황(composite context) : 두 개 이상의 단일 상황으로 구성된 상황 정보를 복합 상황(composite context)라고 정의한다.

복합 상황은 구성 요소의 결합에 따라 AND 상황과 OR 상황으로 구분된다.

정의 6. OR 상황(OR context) : 상황 정보들이 or-관계로 결합된 복합 상황이며, 다음과 같은 형태를 갖는다.

Context(name) = Context(name)∨ Context(name)

OR 상황은 구성 상황 정보들 중 적어도 하나가 참(ture)일 때, 참 값을 갖는다. 즉, 상황 구성 상황 정보들 중에서 적어도 하나만 활성화 되면 OR 상황도 활성화됨을 의미한다.

예를 들면, 의사가 위치할 수 있는 위치는 진료실, 병실 중 하나에만 존재할 수 있다.

Context(의사위치) = Context(진료실) ∨ Context (병실)

의사가 진료실이나 병실 중 한 곳에 존재하면 의사 위치 상황은 참값을 갖는다.

정의 7. AND 상황(AND context) : 상황 정보들이 and-관계로 결합된 복합 상황이며, 다음과 같은 형태를 갖는다.

Context(name) =Context(Name)∧ Context(Name)

AND 상황은 구성 상황 정보들이 모두 참일 때 참 값을 갖는다. 즉, 상황 구성 상황 정보들이 모두 활성화 되었을 때 AND 상황도 활성화됨을 의미한다.

예를 들면, 의사와 관련된 상황은 근무 시간과 위치 정보를 이용하여 정의할 수 있다.

Context(doctor) = Context(근무시간) ∧ Context (의사 위치)

근무시간과 의사위치의 값이 모두 참이면 의사 상황은 참값을 갖는다.

3.4 정책 기술

기본 상황 및 사용자 정의 상황을 고려한 권한부여 규칙은 다음과 같은 형식으로 표현된다.

〈Subject, Object, operation, [Context1 [con Context2 [con ...]]], sign〉

단, con ={& or |}, sign={+(허용) or -(거부)}

[Context1 [con Context2 [con]]]은 상황이 없거나 하나 이상의 상황을 나타낸다. con은 연결 타입(conjunction type)으로 and-결합을 의미하는 & 타입과 or-결합을 의미하는 | 타입이 있다. and-결합으로 연결된 상황은 연결된 모든 상황이 참 값을 가질 때, 즉 모두 활성화 되었을 때 권한부여 규칙이 적용됨을 의미한다. 반대로, or-결합으로 연결된 상황은 연결된 상황 중 적어도 하나의 상황이 활성화 되었을 때 권한부여 규칙이 적용된다.

[예 1]

〈간호사, 처방기록, 읽기, L:병원&T:근무시간, +〉

〈환자, 병력기록, 쓰기, -〉

〈치료의사, 처치기록, 쓰기, L:치료실 | L:환자 zone, +〉

첫 번째 권한부여 규칙은 간호사는 근무시간과 병원 상황에서 처방 기록의 읽기에 대하여 허용권한을 갖는다는 것을 나타낸다. 두 번째 권한부여 규칙은 환자는 어떠한 상황 하에서도 병력기록에 대하여 쓰기 권한이 없음을 나타낸다. 세 번째는 치료의사는 치료실이나 환자zone에서만 처치기록에 대하여 쓰기 권한이 허용됨을 나타내는 권한부여 규칙이다.

3.5 접근제어 수행 과정

사용자의 접근 요청에 대하여 권한을 부여하기 위하여 상황 정보를 이용하여 권한평가를 수행한다. 사용자가 특정 객체에 연산을 수행하고자 요청하면 시스템에는 사용자의 상황도 함께 전달된다. 사용자의 접근 요청이 발생하였을 때 사용자의 현재 상황 값이 시스템에 전달되어 현재 상황 값과 계층구조, 미리 기술된 정의 등을 이용하여 기본 상황, 유도 상황 및 사용자 정의 상황을 활성화시킨다.

권한 평가 프로세서는 사용자의 접근 요청 트랙잭션과 사용자의 상황을 포함하는 권한부여 규칙을 찾고 규칙에 따라 권한을 부여한다. 그림 8은 사용자의 접근 요청에 대한 접근제어 수행 과정을 나타낸다.

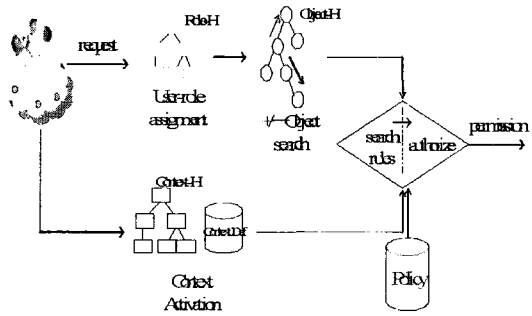

그림 8. 접근제어 수행 과정

□ 권한부여 규칙을 찾는 알고리즘

먼저, 권한 정책에서 사용자 역할, 객체, 연산, 상황, 사인의 다섯 개의 튜플로 구성된 권한부여 규칙을 찾기 위해서는 다음 과정을 수행하여 사용자 요청에 대한 규칙을 찾는다.

사용자-역할 할당표를 참조하여 사용자에게 할당된 역할을 찾는다. 객체에 대한 권한은 허용권한과 거부권한이 있다. 허용권한은 계층구조의 하위에서 상위로 권한이 전파되므로 객체보다 하위에 존재하는 객체들이 가지는 권한이 같이 평가되어야 한다. 거부권한은 허용권한과 반대 방향으로 권한이 전파되므로 객체보다 상위에 존재하는 객체들이 가지는 권한도 평가되어야 한다. 따라서 객체 계층구조 상에서 특정

연산에 대하여 거부권한을 평가하기 위하여 해당 객체와 객체의 상위에 존재하는 객체들을 찾고, 허용권한을 평가하기 위하여 해당 객체와 객체의 상위에 존재하는 객체들을 찾는다. 또한, 활성화된 각 상황에 대하여 각 차원의 상황 계층구조 상에서 상위에 존재하는 상황을 찾는다.

각 사용자 역할, 허용권한에 대한 객체와 연산의 집합, 거부권한에 대한 객체와 연산의 집합, 상황 집합들의 각각의 원소에 대하여 권한이 기술되어 있는 권한 부여 규칙을 찾는다.

□ 권한부여 알고리즘

찾은 권한 규칙들에서 권한을 부여하기 위하여 허

<U, O, op> : 사용자의 접근 요청 트랜잭션
<<U, O, op>, [Context,[Context,[...]]]> :: 시스템에 요청되는 사용자 접근 요청 트랜잭션
Subject_Set : 전체 주체 역할 집합
Object_Set : 전체 객체의 집합
t:Context_Set : t 타입의 상황 집합
n : 상황들이 포함된 차원의 수 (t의 종류의 수)
authorized_Subject(U) : 사용자 U에게 할당된 역할 집합
authorized_Subject(U) ⊆ Subject_Set
negative_Object(O, op) : 객체 O에 대한 연산 op의 접근 거부권한을 평가할 객체 집합
negative_Object(O,op) = {O}∪ {O'| O'는 객체 계층구조에서 O의 상위 객체들}
negative_Object(O,op) ⊆ Object_Set
positive_Object(O,op) : 객체 O에 대한 연산 op의 접근 허용권한을 평가할 객체 집합
positive_Object(O,op)={O} ∪ {O'| O'는 객체 계층구조에서 O의 하위 객체들}
negative_Object(O,op) ⊆ Object_Set
active_ t:Context(Ci) : 각 상황 타입의 활성화된 상황
Ci : 타입별 접근 요청 시 발생한 상황, i=0,1,2,3,
active_ t:Context(Ci) = {Ci} ∪ {C'| C'은 상황 계층구조에서 Ci의 상위 상황}
active_t:Context([Ci,[...]])⊆t:Context_Set

그림 9. 기본 집합의 정의

```
input : << U, O, op>,
        [Context,[Context,[...]]]>
output : authorization rules for <<U, O,
         op>, [Context,[Context,[...]]]>
1. input에 대하여
   authorized_Subject(U),
   negative_Object(O, op),
   positive_Object(O, op),
   active_ t:Context(Ci)를 찾는다.
2. authorized_Subject(U)의 임의의 원소
   SR, SR ∈ authorized_Subject(U),
   negative_Object(O, op)의 임의의 원소
   NO, NO ∈ negative_Object(O, op),
   positive_Object(O, op)의 임의의 원소
   PO, PO ∈ positive_Object(O, op),
   active_ t:Context(Ci)의 임의의 원소
   AC, AC ∈ active_ t:Context(Ci)에
   대하여
   <SR, NO, op, [AC,[...]], ->와
   <SR, PO, op, [AC, [...]], +>을 찾는다.
```

그림 10. 권한부여 규칙 탐색 알고리즘

용권한과 거부권한이 기술된 권한부여 규칙에 대하여 비교를 수행한다. 먼저, 거부 권한이 기술된 권한부여 규칙이 존재한다면 허용권한이 기술된 권한부여 규칙이 존재하는지 비교한다. 만약, 허용권한이 기술된 권한규칙이 존재한다면 권한 충돌이 발생하므로 충돌 해결이 필요하다. 허용권한이 기술된 권한규칙이 존재하지 않는다면 사용자의 요청 트랜잭션은 거부권한이 기술된 규칙이 적용되어 거부권한이 부여된다. 거부권한이 기술된 권한부여 규칙이 존재하지 않는다면 다음으로 허용권한이 기술된 권한부여 규칙이 존재하는지 찾아보고 있다면 허용권한 규칙이 적용되어 사용자의 요청 트랜잭션은 허용권한이 부여된다. 만약, 허용권한과 거부권한이 기술된 규칙이 존재하지 않는다면 사용자의 요청에 대하여 적용될 권한부여 규칙이 존재하지 않으므로 디폴트 규칙(허용 또는 거부)에 따른다.

3.6 충돌 분류와 해결방안

상황을 접근제어에 이용하였을 경우 충돌이 발생할 수 있고 충돌에 대한 해결이 필요하다. 충돌은 권한 충돌과 의미적 충돌로 구분된다.

```
input : authorization rules for << U,
  O, op>, [Context,[Context,[...]]]>
output : authorization for user request.
 입력에서
1. if (negative permission rule <SR,
   NO, op, [AC,[ ... ]], -> )
 1.1 if (positive permission rule <SR,
     PO, op, [AC,[ ... ]], +>)
       권한 충돌
 1.2 else if
       거부권한 부여
2. else if (positive permission rule)
       허용권한 부여
3. else default permission 부여
```

그림 11. 권한부여 알고리즘

정의 8. 권한 충돌 : 상황의 계층구조에 따라 객체에 대한 권한이 전파되면서 발생하는 권한들 사이의 충돌로 같은 연산에 대하여 서로 다른 권한, 즉, 허용권한과 거부권한이 부여되었을 경우 권한 충돌이 발생한다.

정의 9. 의미 충돌 : 상황들의 의미에서 발생하는 충돌로 상충적, 양립할 수 없는(incompatible) 의미를 가지는 상황이 함께 기술되었을 경우 의미 충돌이 발생한다.

3.6.1 권한 충돌과 해결방안

권한 충돌이 발생하는 원인은 상황에 따라 객체에 대한 접근 권한이 기술되기 때문이다. 상황의 계층구조 상에서 하위에서 상위에 존재하는 상황을 고려하여 접근 권한을 평가하기 때문에 같은 연산에 대하여 서로 다른 권한을 갖는 경우 권한 충돌이 발생한다.

단일 상황과 복합 상황에서 발생하는 권한 충돌의 원인과 해결방안을 제안한다.

□ 단일 상황에서 발생하는 권한 충돌과 해결방안

단일 상황에서 권한 충돌이 발생하는 원인은 상황 계층구조상 하위 상황에서 객체에 대한 권한과 상위 상황에서 객체에 대한 권한이 다른 경우 발생한다.

그림 12에서 상황 c1과 c2에서 각각 객체 o2에 대하여 허용권한과 거부권한을 기술하는 권한부여 규칙이 있는 경우, c2보다 하위에 존재하는 상황에 있는 사용자가 객체 o2에 접근을 요청하는 경우 권한 충돌이 발생한다.

그림 13에서 상황 c1에서는 객체 o2에 대한 거부권한을 기술하는 권한부여 규칙과 상황 c3에서는 객체 o6에 대한 허용권한을 기술하는 정책부여 규칙이 있다고 가정하자.

상황 c2에 있는 사용자가 객체 o5에 접근을 요청하였다면 c2에 있는 사용자는 c1의 o2에 대한 거부권한이 전파되어 o5에 대한 거부권한을 갖게 된다. 하지만, c2에 있는 사용자는 c3보다 상위에 존재하므로 c3의 권한부여 규칙에 영향을 받지 않으므로 권한 충돌이 발생하지 않는다. 그러나 상황 c4에 있는 사용자가 객체 o5에 접근을 요청하였다면 c4에 있는 사용자는 c1의 o2에 대한 거부권한이 전파되어 o5에 대한 거부권한과 c3의 o6에 대한 허용권한이 전파되어 o5에 대하여 허용권한과 거부권한을 갖게 되어 권한 충돌이 발생한다.(마찬가지로, 상황 c4 또는 c3에 있는 사용자가 객체 o6 또는 o2에 접근을 요청한다면 권한 충돌이 발생한다.) 이와 같이, 권한 충돌은 사용자가 속해 있는 상황보다 상위 상황이 차례로 객체에 대해 거부권한과 허용권한을 기술하는 권한부여 규칙이 존재하는 경우에 발생한다.

```
< S, Oi, op, Ci, Pi>,
< S, Oj, op, Cj, Pj>에 대하여
Loi = Lo= Loj이고 Poi ≠ Poj이면
권한 충돌2)
```

식 1. 단일 상황의 권한 충돌

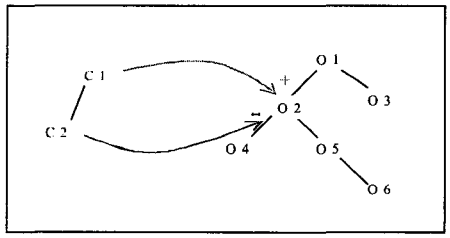

그림 12. 단일 상황의 충돌 1

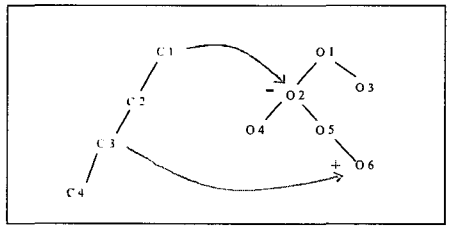

그림 13. 단일 상황의 충돌 2

2) Loi는 Oi의 객체 계층구조 상에서의 레벨을 나타내고, Poi 는 Oi의 권한을 나타낸다.

기존의 방법들에서는 권한 사이에 충돌이 발생하였을 경우 우선 규칙(precedence rule)을 적용하여 처리한다. 우선 규칙이 허용권한이 높은 우선순위를 갖는 경우 허용권한을 적용되고, 거부권한이 높은 우선순위를 갖는 경우 거부권한을 적용한다. 대부분 디폴트 값을 따라 거부권한이 적용된다.

그러나 사용자의 상황을 고려하여 접근제어를 수행하므로 사용자의 상황에 밀접한 접근 권한을 부여하는 것이 적합하다. 단일 상황에서 발생하는 충돌은 사용자의 상황에 가장 구체적 상황(3장에서 언급한 상황 계층구조를 기반으로)의 객체가 갖는 권한을 따르도록 함으로써 해결 가능하다.

```
< S, Oi, op, Ci, Pi>,
< S, Oj, op, Cj, Pj>에 대하여,
if Lci[3] > Lcj , Po = Poi
else if Lci < Lcj , Po = Poj
```

식 2. 단일 상황의 권한 충돌 해결

따라서 사용자가 각각 c2와 c4에 존재하는 경우 c2와 c3의 권한을 따라 각각 거부권한과 허용권한을 갖는다.

〔예 2〕 다음과 같이 두 개의 권한부여 규칙이 있다고 하자.

〈진료의사, 협진기록, 쓰기, L:진료실, + 〉

〈진료의사, 진료기록, 쓰기, L:병원, -〉

사용자가 치료실에서 〈남소연, 소견기록, 쓰기〉를 요청하였을 경우 접근제어는 다음과 같이 처리된다. 사용자는 사용자-역할 할당표를 참조하여 진료의사 역할을 할당 받는다. 진료의사 역할, 소견기록, 쓰기와 관련된 권한을 갖는 위의 두 개의 권한부여 규칙의 적용을 받는다. 진료의사는 협진기록에 대하여 진료실에서 쓰기가 허용되고, 진료기록에 대하여 병원에서 쓰기 기록이 거부되므로 사용자는 소견기록에 대한 쓰기에 대하여 허용권한과 거부권한을 가지므로 권한 충돌이 발생한다. 계층구조 상에서 치료실은 병원보다 진료실이 더 구체적인 상황이므로 구체적 상황(more specific context) 우선 규칙에 따라 사용자는 소견기록에 대한 쓰기에 대하여 허용권한을 갖게 된다.

3) Lci는 상황 계층구조에서의 상황 Ci의 레벨을 의미한다.

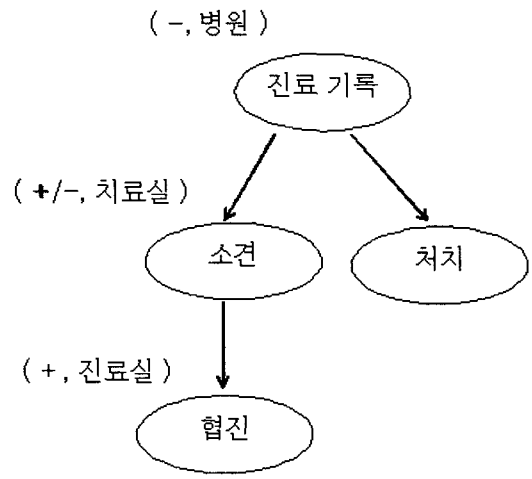

그림 14. 권한충돌 발생과 해결의 예

□ 복합 상황에서 발생하는 권한 충돌과 해결방안

두 개 이상의 상황이 and-결합으로 결합된 복합 상황들의 구성 상황 중에서 부모-자식 관계를 갖는 상황들이 객체에 대하여 각각 서로 다른 사인을 갖는 경우에 발생한다.

그림 15에서 T2와 L2를 구성요소로 하는 AND 상황에서 객체 o2에 대하여 거부권한을 기술하는 권한부여 규칙과 T2, T3과 L3을 구성요소로 하는 AND 상황에서 객체 o6에 대하여 허용권한을 기술하는 권한부여 규칙이 있다고 가정하자. 상황 L3과 하위 상황에 있는 사용자가 객체 o5에 대하여 접근을 요청한다면 o2의 거부권한과 o6의 허용권한이 o5에 전파되어 충돌이 발생한다.

```
< S, Oi, op, Ci1, Ci2,, Pi>,
< S, Oj, op, Cj1, Cj2,, Pj>에 대하여
Loi = Lo = Loj and Poi ≠ Poj이면
권한 충돌
```

식 3. 복합 상황의 권한 충돌

단일 상황에서 발생하는 충돌과 마찬가지로 기존 논문[6,9]에서는 권한 충돌에 대하여 우선 규칙을 적용하여 처리한다. 상황을 고려하여 접근제어를 수행하므로 상황에 따라 권한을 결정해 줄 수 있다. 활성화 되어 있는 복합 상황의 상황 중에서 사용자의 상황과 가장 구체적인 상황의 객체가 갖는 권한을 따르도록 함으로써 권한을 결정할 수 있다. 같은 타입의 상황에 대하여 사용자와 가장 구체적 상황이 각각의 권한 부여 규칙에 존재하는 경우 권한충돌은 해결되

지 않는다. 이러한 경우 우선 규칙을 적용하여 처리한다. 사용자가 L3과 하위 상황에 있다면 AND 상황들의 구성요소 중 충돌을 발생시키는 상황에 대하여 사용자와 더 구체적 상황의 권한을 부여 받는다. 따라서 사용자는 객체 o5에 대하여 허용권한을 갖는다.(단, T2와 T3이 동시에 활성화 된 상황 속에 사용자가 있다.)

〔예 3〕 다음과 같이 두 개의 권한부여 규칙이 있다고 하자.

〈의료진, 환자기록, 쓰기, L:병원&T:근무요일,-〉
〈치료의사, 투약history, 쓰기, L:진료실&T:근무요일&T:근무시간, +〉

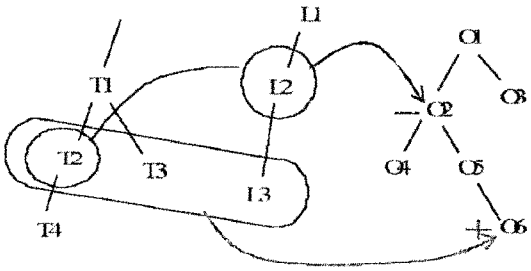

그림 15. 복합 상황의 충돌

사용자가 의국에서〈박철수, 병력기록, 쓰기〉를 요청하였을 경우 접근제어는 다음과 같이 처리된다. 사용자는 사용자-역할 할당표를 참조하여 치료의사와 의료진 역할을 할당 받는다. 의료진과 치료의사 역할, 투약 히스토리, 쓰기와 관련된 권한을 갖는 위의 두 개의 권한부여 규칙의 적용을 받는다. 치료의사는 투약 히스토리에 대하여 근무요일과 근무시간, 진료실에서 쓰기가 허용되고, 환자기록에 대하 ㅓ 근무요일, 병원에서 쓰기 기록이 거부되므로 사 ㅏ는 병력기록에 대한 쓰기에 대하여 허용권한과 권한을 가지므로 권한 충돌이 발생한다. 각 상 대하여 사용자의 상황과 계층구조 상에서 가장 ㅣ 상황을 찾는다. 의국은 병원보다 진료실이 적인 상황이므로 구체적 상황 우선 규칙에 ㅏ는 소견기록에 대한 쓰기에 대하여 허용된다.

과 해결방안
복합 상황 중 AND 상황에서 발생한 구성하는 상황들이 서로 상충적인 ' 함께 기술되었을 때 발생한다.

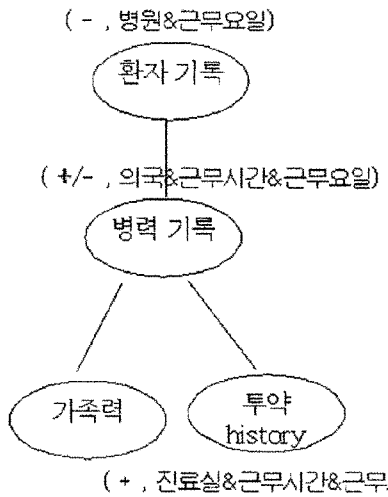

그림 16. 복합 상황에서 발생하는 권한 충돌의 예

예를 들면, 사용자의 위치와 관련한 상황은 하나만 포함될 수 있다. 사용자의 위치 값은 꼭 하나의 값만 참이 되기 때문이다. 따라서 둘 이상의 상황이 복합 상황 정의에 이용될 경우 그 복합 상황은 항상 거짓 값을 갖게 된다. 상충적인 의미를 갖는 상황은 AND 상황에 함께 기술되지 않도록 하는 제약이 필요하다. 상황 계층구조 상에서 같은 레벨에 존재하는 상황을 함께 기술될 수 있는 양립 가능한 상황(compatible context)과 함께 기술될 수 없는 상황(incompatible context)으로 구분할 수 있다. 계층구조 상 같은 레벨에 위치한 상황들 중에서 동시에 발생할 수 있는 상황을 양립 가능한 상황이라 하고, 동시에 발생할 수 없는 상황을 함께 기술될 수 없는 상황이라고 한다.

그림 17에서 사용자의 위치는 의국, 치료실, 검사실 중 하나의 값만 참 값을 가질 수 있으므로 위 위치 상황은 함께 기술될 수 없는 상황이다. 그림 18에서 주말과 주중 상황은 함께 기술될 수 없는 상황으로 함께 기술될 경우 의미충돌이 발생한다. 시간과 관련한 상황 중 시각과 요일은 동시에 발생하므로 양립 가능한 상황으로 함께 기술될 수 있다. 요일과 날짜는 시간 관련 상황으로 동시에 발생한다. 따라서 요일과 날짜를 동시에 접근제어 정책에 기술할 수 있다.

3.7 충돌 해결이 적용된 권한 탐색 및 평가

충돌을 고려한 제안 기법의 권한부여 알고리즘 ' 서 활성화된 상황을 이용하여 권한결정을 '

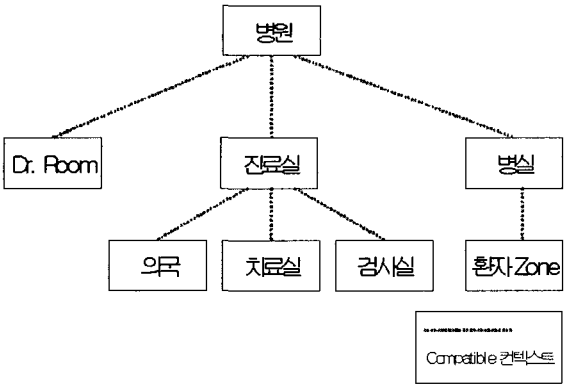

그림 17. 제약이 표현된 위치 상황

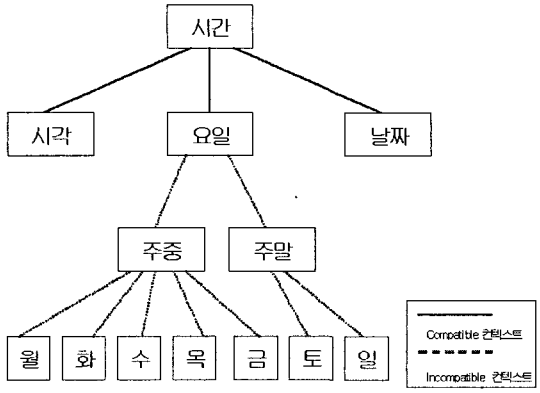

그림 18. 제약이 표현된 시간 상황

한 정책에서 권한부여 규칙을 찾는 과정은 식 1의 루프만큼 비교를 수행해야 한다. 그림 19의 알고리즘은 각 타입에 대하여 활성화된 상황 집합의 전체 원소에 대하여 비교를 수행하는 것이 아니라 가장 구체적 상황부터 탐색을 수행하고 찾으면 탐색은 종료된다. 따라서 최악의 경우 각 타입에 대하여 상황 집

```
[Algorithm]
input: <<U, O, op>,[Context,[Context],[ ... ]]]>
output: authorization rule for user request
current_ t:Context : 사용자의 접근 요청이 발생한 t 타입의 상황
current_t:Context ∈active_ t:Context
t:Context_in_rule : rule 규칙에 포함된 t 타입의 상황
auth_rule : 사용자 U에게 부여될 수 있는 권한의 집합
all_auth(most_specific_rule_t:Context) : t 타입의 구체적 상황들과 관련된 모든 권한
n= 상황 타입의 수 m= most_specific_rule의 총수
1. 각 상황 타입 t 에 대하여
 most_specific_ t:Context = current_t:Context
2. For (n){
 2.1 rule_set = all authorization rules , auth_rule={}
 2.2 for (t∈상황 타입의 집합){
 2.2.1 LOOP : for(rule ∈ rule_set){
 2.2.1.1 if(most_specific_t:Context== t:Context_in_rule) {
 auth_rule=auth_rule + rule } }
 2.2.2 if (auth_rule==NULL) {
 most_specific_t:Context= next_most_specific_t:Context(active_t:Context)
 goto LOOP} }
 2.3 most_specific_t:rule=auth_rule
}
3. rule=most_specific_ t:rule의 합집합
4. for(t ∈ 상황 타입의 집합){
 4.1 for(m){
 most_specific_rule_t:Context= get_rule(most_specific_t:Context) }
 4.2 각 상황 타입 t에 대하여
 4.2.1 if (all auth(most_specific_rule_t:Context)=='+')
 return positive permission
 4.2.2 if (all auth(most_specific_rule_t:Context)=='-')
 return negative permission
 4.2.3 else return default permission
}
```

그림 19. 구체적 상황 우선 규칙이 적용된 규칙 탐색 및 권한부여 알고리즘

합의 전체 원소에 대하여 비교를 수행하고, 그렇지 않은 경우 탐색은 중간에 종료된다. 따라서 최악의 경우를 제외하면 탐색의 비교 횟수가 줄어들어 응답 시간이 줄어든다. 예를 들면, 예 3의 접근 권한 요청에 대하여 구체적 상황 우선 규칙이 적용된 권한부여 규칙 탐색과 권한 결정 알고리즘을 이용하여 다음과 같이 처리될 수 있다.

사용자의 접근 요청이 발생한 current_L:의국, current_T:근무요일, current_T:근무시간 상황에 대하여 Location과 Time 각 상황에 대하여 현재 사용자의 접근 요청이 발생한 current_L:의국, current_T:근무요일, current_T:근무시간 상황과 같은 상황이 권한 부여 규칙에 존재하면 권한부여 규칙을 탐색 리스트에 추가한다. current_L:의국 상황에 대하여 L:옥구, L:진료실과 L:병원 상황이 포함된 권한부여 규칙을 auth_rule에 추가한다. 마찬가지로, current_T:근무요일과 current_T:근무시간에 대하여 권한부여 규칙을 탐색하여 auth_rule에 추가한다.

위에서 탐색 결과 추가된 auth_rule 리스트에 대하여 각 타입의 상황에 대하여 사용자의 요청이 발생한 상황과 가장 구체적인 상황에 대한 권한부여 규칙을 most_specific_t:Context으로 선택한다. 각 Location과 Time에 대하여 사용자의 요청이 발생한 상황과 가장 구체적인 상황인 most_specific_L:진료실과 most_specific_T:근무시간에 대하여 auth_rule을 비교하여 auth_rule 에 각각의 구체적인 상황이 기술된 권한부여 규칙의 권한을 가져온다. most_specific_L:진료실에 대하여 접근 허용권한이, most_specific_T:근무시간에 대하여 접근 허용권한이 기술되어 있으므로 사용자에게는 접근 허용권한이 주어진다.

Ⅳ.비교 및 평가

4.1 상황 정보를 이용한 접근제어 비교 및 평가

상황의 적용 대상, 적용방법, 종류 및 분류, 구조화 방법, 수행 방법에 있어서 기존 모델들과 제안한 기법 비교하면 다음 표 2와 같다.

제안한 기법은 차원에 따른 기본 상황을 정의하여 사용자 정의 상황을 기술하고, 상황들의 의미적 관계에 따라 OR 관계를 갖는 상황을 정의하여 사용함으로써 상황의 추가, 삭제, 수정 등의 관리의 용이함을 제공한다. 또한 현실적 의미를 반영한 정책을 기술할 수 있도록 하였고, 충돌을 해결하는 방안인 가장 구

표 2. 평가 기준

	항목	xoRBAC 모델	GRBAC 모델	제안하는 모델
모델 평가	상황 적용 대상	객체	주체, 객체	주체, 객체
	권한 전파	권한 전파 불가능	권한의 전파 가능	권한의 전파 가능
	권한 할당의 유연성	명시된 상황만 고려	명시된 상황, 전파에 의한 상황을 고려	명시된 상황, 전파에 의한 상황 고려
관리	상황 관리의 용이성	정형화된 구조가 없어 관리가 어려움	계층구조만 이용	계층구조, 기본 상황, 사용자 정의 상황을 정의하여 관리가 용이
	정책 기술의 표현성	만족되어야 할 조건만 나열	만족되어야 할 환경 역할만 나열	사용자 정의 상황, OR 상황의 사용. 풍부한 정책 기술 가능
수행	상황 활성화	만족해야 할 조건만 검사	현재 활성화된 환경 역할과 정책에 기술된 상황과 비교	센싱된 실제값을 기본 상황과 사용자 정의 상황에 매핑시킴
	권한 탐색	규칙 탐색 및 객체에 대한 환경 조건을 기술한 제약사항 탐색	주체, 객체, 환경 역할에 대한 규칙 탐색	주체, 객체, 상황에 대한 규칙 탐색. 구체적 상황 우선 고려시 탐색의 횟수 감소
	권한 평가	제약 사항의 조건 검사만으로 권한 부여.	탐색한 규칙들의 권한을 비교 및 충돌을 고려한 후 권한 부여	상황 타입의 수에 대한 규칙의 권한 비교 후 권한 부여
	구현의 복잡성	제약사항의 추가만 필요	환경 역할의 정의와 계층구조, 환경 역할의 활성화 필요	상황의 정의와 계층구조, 활성화 프로세서, 탐색과 평가 프로세서 필요

체적 상황 우선 원칙을 사용하여 사용자의 접근 요청에 대하여 충돌을 해결하고 상황에 따른 권한을 부여할 수 있었다. 그러나 제안한 기법은 사용자 정의 상황을 정의하고 있어 다단계의 상황 평가에 따른 오버헤드가 발생하였고, 구현이 복잡한 단점이 있다.

4.2 충돌의 분류와 해결방안 비교 및 평가

상황을 접근제어에 적용하는 방법에 따라 발생하는 충돌의 원인과 분류, 해결방법에 차이점을 갖는다.

표 3에서 알 수 있듯이 xoRBAC은 상황에서 발생할 수 있는 충돌에 대한 고려를 하지 않고 있다. 반면, GRBAC과 제안한 기법은 상황의 자체 의미가 갖는 의미적 충돌과 상황을 접근제어에 적용함에 따라 발생하는 권한들 사이의 충돌에 대하여 고려하고 있다. 그러나 GRBAC은 권한 충돌에 대하여 우선 규칙에 따라 권한을 부여하는 비교적 간단한 해결방법으로 권한 충돌을 해결하고 있고, 의미 충돌에 대해서도 제약사항에 기술하는 방법으로 해결하고 있다. 제안한 기법은 권한 충돌이 발생할 경우 사용자의 접근 요청이 발생한 현재 상황을 고려하여 근접한 상황에서의 권한을 따르도록 함으로써 접근제어 수행의 유연성을 제공하고 있다. 의미 충돌에 대하여 함께 기술될 수 없는 타입과 양립할 수 있는 타입으로 구분하여 제약사항에 기술함으로써 사용자 정의 상황의 평가 때에 충돌을 발견하도록 하고 있다. 또한, OR 상황을 정의하여 기술될 수 없는 타입의 상황들이 함께 기술되어도 충돌이 발생하지 않는다. 기술된 상황들 중 적어도 하나만 만족되면 활성화 될 수 있는 의미를 갖는 OR 상황의 정의는 사용자 정의 상황의 기술을 편리하게 한다.

표 3. 충돌의 분류 및 해결방안 비교

항목		xoRBAC	GRBAC	제안한 기법
발생 원인		양립할 수 없는 상황 기술	권한 전파 및 양립할 수 없는 상황의 기술	권한 전파 및 양립할 수 없는 상황의 기술
분류		충돌에 대한 고려가 없음	권한 충돌과 의미충돌	권한 충돌과 의미 충돌
해결 방법	권한 충돌	없음	Precedence policy	More specific context
	의미 충돌	없음	error로 기술	비호환 상황 기술, OR 상황 이용

V. 결 론

기존의 역할 기반 접근제어에 사용자 상황을 이용하여 사용자의 상황에 따른 접근제어를 수행하기 위한 기법을 제시하였다. 차원에 따른 기본 상황과 사용자 정의 상황을 정의하여 시스템 관리자나 사용자가 풍부한 의미를 포함하는 접근제어 정책을 기술할 수 있도록 하였다. 또한, 각 차원의 상황에 대하여 계층구조 상의 레벨을 상속 받도록 하여 권한부여 규칙을 찾기 위한 탐색에 이용되도록 하고 있어 탐색시간을 줄일 수 있도록 하였다.

한편, 상황을 기존의 역할 기반 접근제어에 이용하였을 경우 권한의 전파로 인해 발생할 수 있는 권한 충돌과 상황 자체가 가지는 의미 충돌을 고려하였고, 권한 충돌을 해결할 수 있는 알고리즘을 제시하여 권한부여 규칙을 찾기 위한 탐색 시간을 최악의 경우를 제외하고 줄일 수 있었다.

그러나 상황에서 발생하는 충돌 중 권한 발생 충돌에 대하여 사용자의 상황에서 가장 가까운 상황의 권한을 따르도록 함으로써 해결 방안을 제시하고 있으나, 계층구조 상 두 상황의 거리가 같은 경우 충돌을 해결하지 못하고 있다. 또한, 순간적으로 변화하는 사용자 환경에 적절한 접근제어를 수행하기 위해서는 응답시간에 대한 제약을 필요로 한다. 본 논문은 권한 결정 과정에서 권한부여 규칙을 찾기 위한 탐색시간을 단축할 수 있지만, 빠른 응답시간을 보장할 수는 없다. 따라서, 탐색시간의 단축과 빠른 응답시간 보장을 위한 추가 연구가 필요하다.

한편, 유비쿼터스 컴퓨팅 환경에서는 인증이 없는 기밀성 제공은 기밀성 여부의 정당성을 확보할 수 없어 인증이 기밀성보다 중요한 문제가 된다[11,12]. 즉, 인증은 기밀성이나 무결성, 가용성의 선행조건으로 유비쿼터스 컴퓨팅 환경에서 고려되어야 한다. 이와 같은 인증 기술을 기반으로 서비스 제공자의 신뢰정도를 식별하고 검증할 수 있다면, 서비스 제공자의 신뢰수준에 따라 사용자 정보의 접근 정도를 다르게 할 수 있는 사용자 정보 기반의 접근제어[13]도 가능할 것이라 본다.

참 고 문 헌

[1] Mark Weiser, "Some Computer Science Issues in Ubiquitous Computing", In Communications of the ACM ,

36(7), pp.4-7, July 1993.

[2] Anind K. Dey and Gregory D. Abowd, "Understanding and Using Context", In Personal and Ubiquitous Computing Journal, Springer Verlag, 5(1), pp.4-7, 2001.

[3] Anind K. Dey and Gregory D. Abowd, "Towards a Better Understanding of Context and Context-Awareness", In Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000), The Hague, The Netherlands, April 3, 2000.

[4] C.K. Georgiadis, I. Mavridis, G. Pangalos and R. K. Thomas, "Flexible Team-Based Access Control Using Contexts", In ACM Symposium on Access Control Models and Technologies (SACMAT2001), pp.21-30, May 2001.

[5] M. Wilikens, S. Feriti, A. Sanna and M. Masera, "A Context-Related Authorization and Access Control Method Based on RBAC : A case study from the health care domain", In 7th ACM Symposium on Access Control Models and Technologies(SACMAT2002), pp. 117-124, 2002.

[6] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman, "Role Based Access Control Model.", In IEEE Computer, 20(2), pp.38-47, February 1996.

[7] Ravi Sandhu, David Ferraiolo, and Richard kuhn, "The NIST Model for Role-Based Access Control: Towards A Unified Approach", In ACM Workshop on Role-Based Access Control. pp. 47-63, 2000

[8] M. J. Covington, M.J. Moyer and M. Ahamad, "Generalized Role-based Access Control for Securing Future Applications", In 23rd National Information Systems Security Conference (NISSC), pp.115-125, Baltimore, Maryland, USA, October 2000.

[9] Matthew J. Moyer and Mustaque Ahamad, "Generalized Role-based Access Control", In IEEE International Conference on Distributed Computing Systems(ICDCS2001), pp.391-398, Mesa, Arizona, USA, April 2001.

[10] Gustaf Neumann and Mark Strembeck, "An Approach to Engineer and Enforce Context Constraints in an RBAC Environment", In 8th ACM Symposium on Access Control Models and Technologies (SACMAT2003), pp. 65-79, Como, Italy, June 2003.

[11] 김완석, 김정국, "유비쿼터스 컴퓨팅과 정보보호: 유비쿼터스 컴퓨팅의 발전 전망과 보안에 대한 이슈", 정보보호학회지, 제14권 제1호, 2004.

[12] 박춘식, "유비쿼터스 네트워크와 시큐리티 고찰", 정보보호학회지, 제14권 제1호, 2004.

[13] 조영섭, 조상래, 유인태, 진승헌, 정교일, "유비쿼터스 컴퓨팅과 정보보호: 유비쿼터스 컴퓨팅과 보안요구사항 분석", 정보보호학회지, 제14권 제1호, 2004.

〈著者紹介〉

박 석 (Seog Park)
1978년 2월 : 서울대학교 계산통계학과(이학사)
1980년 2월 : 한국과학기술원 전산학과(공학석사)
1983년 8월 한국과학기술원 전산학과(공학박사).
1983년 9월~현재 : 서강대학교 컴퓨터학과 교수
2002년~2004년 : University of Virginia 방문교수
1998년~현재 : 한국정보과학회 데이터베이스 연구회 운영자문위원
2004년 1월~현재 : 한국정보과학회 상임이사
2004년 1월~현재 : 한국정보과학회지 편집위원장
1999년~ 현재 : DASFAA Steering Committee
2004년 : DASFAA 2004 Organization Chair
1997년 1월~ 현재: 한국정보보호학회 이사
〈관심분야〉 데이터베이스 보안, 트랜잭션 관리, 센서네트워크 데이터 관리, XML, 스트리밍 데이터 처리, 유비쿼터스 컴퓨팅, 역할기반 접근제어, 상황-인식 접근제어

남 승 좌 (Seung-Jwa Nam)
2000년 2월 : 서강대학교 컴퓨터학과(학사)
2002년 2월 : 서강대학교 컴퓨터학과(석사)
2002년 2월~현재 : LG전자 정보통신사업본부 단말연구소.
〈관심분야〉 역할기반 접근제어, 상황-인식 접근제어, 유비쿼터스 컴퓨팅