# 유비쿼터스 시스템에서의 상황인식 응용을 위한 기반구조의 설계 및 구현

강호석° 노세훈 심영철
홍익대학교 컴퓨터공학과
{hskang°, sehun, shim}@cs.hongik.ac.kr

# Design and Implementation of a ubiquitous System Infrastructure for Context-Aware Applications

Ho-Seok Kang° Sehun Noh Young-Chul Shim
Dept. of Computer Engineering, Hongik University

요 약

유비쿼터스 컴퓨팅 시스템은 무선 네트워크와 모바일 네트워크 환경과 함께 발전하고 있으며, 일부에서 실제 유비쿼터스 컴퓨팅 시스템이 우리 생활에 사용되고 있고, 앞으로는 더 많이 이용 될 것이다. 이러한 유비쿼터스 시스템을 여러 분야에 적용시키기 위해서 실제 유비쿼터스 시스템을 설계, 구축하고 적절한 시나리오에 맞추어서 동작시켜봄으로써 유비쿼터스 시스템의 발전에 도움을 줄 수 있을 것이다. 또한 이러한 시스템을 설계하기 위하여 센서, DB, PDA등의 여러 장비를 실제 다뤄 볼 수 있고, 이러한 시스템을 보다 안전하게 제어하는 기술을 습득하고 이러한 시스템을 실제 생활에 적용시키는 시나리오에 대하여 분석한다.

## 1. 서 론

유비쿼터스 컴퓨팅 시스템 환경은 무선 네트워크와 모바일 네트워크 환경과 함께 발전하고 있으며, 일부에서 실제 유비쿼터스 컴퓨팅 시스템이 우리 생활에 사용되고 있고, 앞으로는 더 많이 이용 될 것이다. 이러한 유비쿼터스 컴퓨팅 시스템을 여러 분야에 적용시키기 위해서 실제 유비쿼터스 컴퓨팅 시스템을 설계, 구축하고 적절한 시나리오에 맞추어서 동작시켜봄으로써 유비쿼터스 시스템의 발전에 도움을 줄 수 있을 것이다.

이 논문에서는 실제 센서장비와 블루투스를 이용한 PDA장비를 이용하여 분리된 두 공간(방)에 센서를 설치하여 센서와 PDA를 이용한 정보를 수집, 분석하여 적절한 상황에 대한 동작을 시험하는 시나리오를 구축하고 테스트 해 보는데 있다.

2장에서는 유비쿼터스 상황 인식 시스템의 구조를 살펴고고, 3장에서는 현재 이 논문에서 설계한 유비쿼터스 시스템의 구성에 대하여 살펴본다. 그리고 4장에서는 우리가 구축한 시스템을 바탕으로 제약사항과 이벤트에 대한 명세와 설계한 시스템에 맞는 다양한 시나리오를 적용하여 어떻게 활용할 수 있는지에 대하여 살펴본다. 그리고 마지막으로 5장에서 결론을 내린다.

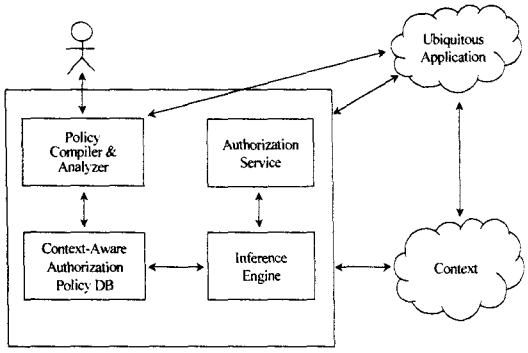
## 2. 관련 연구

유비쿼터스 상황 인식 시스템의 연구는 U.C Berkeley와 Carnegie Mellon같은 유명한 대학들에서 이미 제시되어 있다. 그리고 일리노이(Illinois) 주립대학에서는 Gaia Project에서 Cerberus라 불리는 상황 인식 보안 서비스가 제안되어 있다. 이러한 상황 인식 응용의 기본 구조와 보안 서비스는 여러 가지가 있지만 구조간의 상호 연관성이 명확하게 정의 되어 있지 않다. 그래서 상황인식 응용과 유비쿼터스 시스템 구조와의 연관성을 정의한 시스템들이 개발되었다. 그 중에 하나로 대표적인 상황인식과 유비쿼터스 컴퓨팅 응용 환경에서의 접근제어 시스템의 구조를 살펴본다.

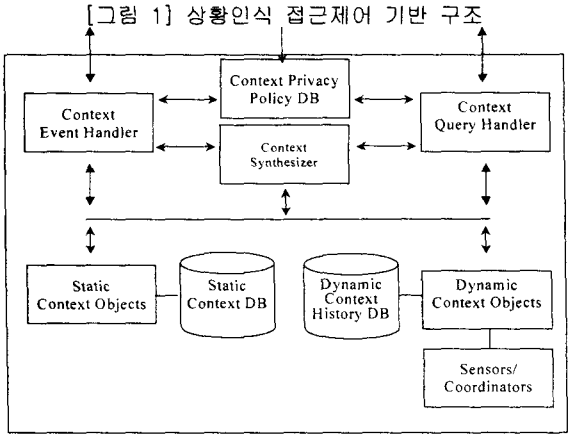
유비쿼터스 환경에서 안전한 상황인식 응용 시스템을 구성하기 위하여, 상황 인식 시스템의 구조는 접근제어 구조(Authorization Infrastructure)와 상황 기본 구조의 (Context-Aware Infrastructure) 두 가지 모듈로 구성 되어 있다.

접근제어 기반 구조는 [그림 1]과 같은 구조이다. 유비쿼터스 응용시스템이 접근을 하기 위해서는 접근제어 서비스에게 접근요구를 요구한다. 접근제어 서비스는 추론(Inference) 엔진 에게 접근 제어 여부를 묻게 되고 그 결과를 유비쿼터스 응용프로그램에게 보낸다. 추론 엔진은 접근 제어에 대한 결정을 상황 인식 접근제어 정책 DB에 저장 된 정책 DB를 기반으로 결정한다.

본 연구는 정보통신부 및 정보통신진흥원의 대학 IT연구센터 육성 지원사업과 2006년도 2단계 두뇌 한국(BK) 21 사업에 의하여 지원되었음




[그림 1] 상황인식 접근제어 기반 구조

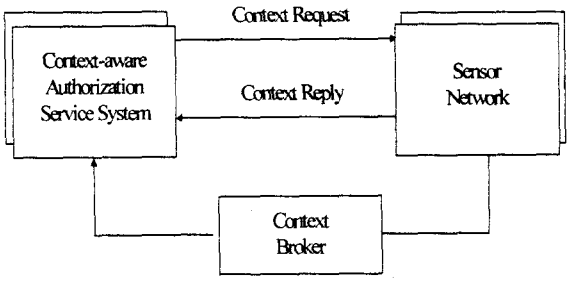



[그림 3] 상황 기반 구조

[그림 2]는 상황 기반 구조의 구조를 나타낸다. 상황 기반 구조를 보면, 두 가지 종류의 상황을 제공하는데, 동적 상황(Dynamic Context) 와 정적 상황(Static Context) 이다. 유저의 위치와 온도 등과 같이 시간이 지나면 변하거나 센서에 의해 수집될 때마다 바뀌는 상황 인식을 동적 상황이라 하고, 건물의 구조나, 집 전화번호 같이 DB에 단순이 저장되어 있는 상황을 정적 상황이라 한다. 상황 소비자(Context Consumer)들은 센서들로부터 상황을 주기적으로 받거나, 혹은 특정 질의를 주었을 경우 해당하는 값을 받을 수 있다. 또는 특정 이벤트가 발생하였을 경우 상황 소비자에게 알리는 기능을 할 수 있게 된다.

상황 기반 구조의 구성요소들을 다음과 같은 역할을 하게 된다.

- 동적 상황 객체(Dynamic Context Object): 센서로부터 수집된 정보를 모아 요약하여 정보를 제공한다. 타입, ID, 위치, 설치시간, 밀도, 샘플링 간격 그리고 센서정보 등과 같은 인자와 탐지 시작시간, 정지시간, 시험, 리포트, 밀도 변화, 샘플링 간격 변화 와 같은 명령어를 함께 담고 있다.
- 정적 상황 객체(Static Context Object): 정적 상황 정보를 제공한다. 실제 정적인 데이터의 경우 DB에 저장된다.
- 상황 합성기(Context Synthesizer): 위의 상황 개체의 경위 낮은 레벨의 정보만을 제공한다. 그러나 이 합성기는 낮은 레벨의 상황 정보를 사용하여 높은 레벨의 상황 정보를 생성하여 제공한다.
- 상황 질의 핸들러(Context Query Handler): 상황 소비자가 원하는 상황을 얻기 위한 질의를 보내는 것을 관리한다. 동적/정적 상황 개체와 상황 합성기에 저장된 상황을 사용하여 질의에 대답하게 된다.
- 상황 이벤트 핸들러(Context Event Handler): 상황 소비자는 이 이벤트 핸들러에 상황 명세를 등록하게 되고 이 상황 소비자가 설정해 놓은 이벤트를 센서가 감지해 내게 되면 이 정보를 상황 소비자에게 알려주게 한다.
- 상황 프라이버시 정책 DB(Context Privacy Policy DB): 상황 정보에 프라이버시를 적용하여 관리한다. 센서로부터 수집된 상황 정보는 적당한 레벨을 부여하여 제공하게 된다.




[그림 2] 상황 인식 접근제어 서비스와 센서 네트워크와의 관계

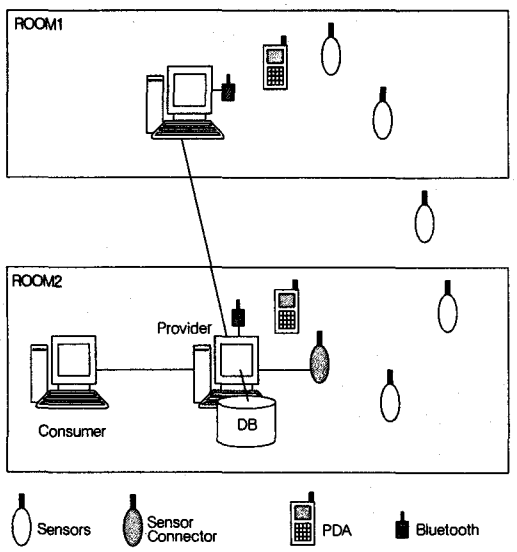
[그림 3]은 많은 수의 센서들로 구성된 네트워크와 상황 인식 접근제어 서비스 시스템과의 관계를 나타내고 있다. 지역적으로나 기능적으로 넓게 분포된 센서들을 서로 협력하여 연결되고 이러한 센서 군들과 인증 서비스 시스템과의 사이에 상황 브로커(Context Broker)가 해당 센서 네트워크 코디네이터에게 쉽게 접근할 수 있게 도와주고 있다.

## 3. 시스템 구성

이 논문에서는 실제 센서장비와 블루투스를 이용한 PDA장비를 이용하여 분리된 두 공간(방)에 센서를 설치하여 센서와 PDA를 이용한 정보를 수집, 분석하여 적절한 상황에 대한 동작을 시험하는 시나리오를 구축하고 테스트 해 보는데 있다.

### 3.1 시스템 구조

기본적인 구조는 [그림 4]와 같은 형태이다. 실험 환경은 우선 방1과 방2의 서로 다른 두 공간에서 이루어지며, 센서는 빛 감지 센서와, 모션 감지 센서의 두 종류로 구성되고, 제공자(Provider) 컴퓨터 시스템과 소비자(Consumer) 컴퓨터 시스템으로 구성되어 있다. 또 사용자 ID를 식별하기 위하여 블루투스통신이 가능한 PDA와 PDA와 컴퓨터 시스템의 블루투스 통신을 연결한 블루투




[그림 4] 시스템 배치 구조

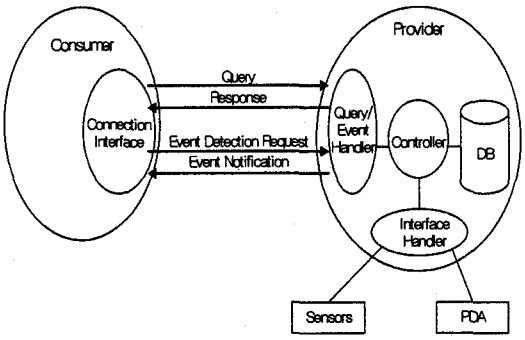
스 장비를 장착한 시스템을 양쪽 방에 두었다. 블루투스 통신 시스템의 경우 제공자 시스템이 있는 경우 제공자 시스템이 블루투스 통신 제어를 같이 수행하게 하였다. 제공자 시스템의 경우 센서와 PDA로부터 수집되는 정보를 저장 분석하기 위하여 DB를 두었다. 우리가 설계한 시스템의 센서네트워크의 경우 센서 커넥터(Connector)를 루트로 하여 모든 센서들의 통신은 애드혹(Ad-hoc) 네트워크로 통신을 한다. 그래서 [그림 4]와 같이 방1과 방2 사이에 센서의 연결을 하기위한 중계센서를 두어 방1의 센서에서 수집된 정보를 제공자에게 전달하게 된다.

제공자 시스템과 연결된 DB는 센서에서 수집된 정보를 담을 수 있고, PDA에서 인식된 정보를 비교할 수 있으며 특정 이벤트나 제약사항에 맞는 정보를 선택적으로 담을 수도 있다. 이러한 제공자는 원하는 정보를 제공할 뿐만 아니라 특정 조건의 질의 대한 응답을 제공 할 수도 있어야한다. 이러한 질의를 주고, 특정 이벤트를 받을 수 있는 사용자 혹은 관리자와의 인터페이스 역할을 하는 것이 소비자 시스템이다.

소비자의 경우 질의를 주고받을 수 있는 기능, 그리고 이벤트를 등록하고 해당 이벤트가 발생 하였을 경우 이벤트 통지를 제공자(Provider)로부터 제공 받아 표시할 수 있는 인터페이스 역할을 하게 된다. 제공자의 경우 센서장비와 내부 DB를 가지고 있으며 다양한 핸들러를 포함 하고 있다.

### 3.2 시스템 핸들러

[그림 5]는 실제 구현된 유비쿼터스 시스템의 구조이다. 이 시스템은 크게 소비자와 제공자로 구분되어 있다. 소비자는 센서 정보를 우리 사람이 보고 분석할 수 있는 최종 응용 시스템이다. 연결 인터페이스(Connection




[그림 5] 유비쿼터스 시스템 구조

Interface)를 통하여 센서에서 수집된 정보를 받아보거나, 원하는 센서의 상태를 알아내고, 특정 이벤트 상황이 발생할 경우 알려주는 역할을 하게 된다. 제공자 시스템은 센서나 PDA로부터 정보를 수집하고 분배, 그리고 저장, 분석을 하는 시스템이다. 센서와 PDA에서 들어오는 정보를 수집하는 역할을 하는 것이 인터페이스 핸들러이다. 그리고 질의/이벤트 핸들러는 소비자 시스템에서 들어오는 질의나 이벤트 알림을 받고 해당 응답을 하거나 조건에 맞는 상태를 다시 소비자에게 보내는 역할을 하게 된다. 센서나 PDA에서 수집되는 정보는 정보가 들어오자마자 DB에 저장되고, 특정 조건에 맞는 정보는 질의/이벤트 핸들러에 의하여 바로 소비자로 보내야한다. 이러한 핸들러들과 DB사이에 역할을 분담하고 조율하는 기능을 하는 것이 컨트롤 핸들러이다.

소비자와 제공자의 통신은 [그림 5]에서와 같이 질의(Query), 응답(Response), 이벤트 탐지 요구(Event Detection Request), 이벤트 통지(Event Notification)의 4가지이다. 질의와 응답은 소비자가 요구하는 정보를 질의형식으로 보내서 응답을 받는 프로토콜이다. 예를 들어 방718에 있는 사람들 리스트를 보내달라고 요구하게 되면 질의/이벤트 핸들러와 컨트롤러를 통해 DB에 있는 정보를 다시 받게 된다. 반대로 이벤트 탐지 요구와 이벤트 통지는 어떤 특정 이벤트가 발생하면 소비자에게 통보를 받는 프로토콜이다. 예를 들어 이벤트 탐지 요구로 방718에 ID11번 사용자가 입장하게 되면 응답을 달라는 명령을 보내고, 컨트롤러에 의하여 센서로부터 정보가 들어오게 되면 질의/이벤트 핸들러를 통해 소비자

에게 ID11번이 방718에 들어왔는지를 알리게 된다.

### 3.3 DB의 구조

센서와 PDA로부터 들어오는 정보는 수시로 저장해야 한다. 이러한 정보를 효율적이고 직관적으로 분석하고 탐색하기 위하여 다양한 형태의 테이블을 이용하였다.

현재 가지고 있는 수집 장비는 빛 센서, 모션 탐지 센서, 사용자 ID를 수집하는 PDA이다. 우선 각 센서는 특정 위치에 고정되어 있고 사용자 ID와 신상명세 사항도 고정되어 있기 때문에 이러한 정보는 센서에 의해서 수집되어 변경되지 않는 고정 DB를 가지고 있다.

(고정 DB)

-사용자 정보 테이블: 사용자 ID / 이름 / 직급 등등.
-센서위치 테이블: 센서 ID / 방 번호.

각 센서별로 상태 변화와 시간을 기록한 DB가 필요하다. 현재 상태란 빛 감지 센서의 경우 해당 위치의 방에 불이 켜졌는지 꺼졌는지의 상태를 나타내고 모션 탐지 센서의 경우는 움직임이 있는지 없는지를 탐지하여 테이블에 기록한다. 사용자 테이블은 현재 방 번호를 기준으로 방에 있는 사용자 ID 리스트를 보관하게 된다. 이러한 상태 DB의 경우 테이블 갱신 단위가 시간이 된다. 즉 일정시간마다 주기적으로 갱신된다.

(상태 DB)

-빛 감지 센서 테이블: 센서 ID / 현재 상태 / 시간
-모션 감지 센서 테이블: 센서 ID / 현재 상태 / 시간
-사용자 테이블: 방 번호 / ID 리스트 / 시간

이벤트 DB는 특정이벤트가 발생할 경우 기록을 하게 되는 DB형태이다.

(이벤트 DB)

-빛 감지 이벤트 테이블: 방 번호 / 이벤트 타입 (ON/OFF) / 시간
-모션 감지 이벤트 테이블: 방 번호 / 이벤트 타입 (MOVE/NOT) / 시간
-사용자 이벤트 테이블: 방 번호 / 이벤트 타입 (Enter/Leave) / 사용자 / 시간

이러한 이벤트 DB의 경우 상태 DB를 참조하여 이벤트의 변화가 있을 경우 갱신하게 된다. 예를 들어 방707에 라이트가 오프 상태에서 온 상태로 바뀔 때 State DB를 참조하여 갱신하게 된다.

(수집 DB)

-센서 테이블: 센서 번호 / 센서에서 수집된 값 / 시간

수집 DB는 센서로부터 수집된 모든 자료를 수집된 값 자체로 일정시간동안만 보관하는 테이블이다. 예를 들어 빛 감지 센서를 보면, 센서에서 들어오는 값은 특정 범위의 숫자 값이지 라이트의 ON/OFF를 판단하는 것은 아니므로 일단 순수한 값 자체를 저장하고 컨트롤 핸들러가 적당한 상태를 판단한 후 이벤트 DB나 State DB에 기록하게 된다.

## 4. 제안된 구조의 활용

이러한 시스템을 활용하기 위해서는 접근 경우 접근제어의 정책이 중요한 요소가 된다. 접근제어나 이벤트의 경우 우리가 설계한 시스템을 유지시키고 운영시키기 위하여 반드시 필요한 사항이다. 이러한 접근제어를 효율적으로 하기 위해서는 접근제어나 이벤트에 관한 명세 언어가 필요하게 된다. 접근제어 정책을 위한 명세 언어를 간단히 살펴보고 시스템을 활용하는 간단한 시나리오에 대하여 알아본다.

### 4.1 상황인식 접근제어

접근제어 정책에 대한 명세를 정의하기 위한 언어를 소개한다. 이 언어는 기본적인 6가지 정책 타입을 이용하여 설명한다. 이 6가지 타입은 다른 정책 명세 언어를 설명하는데 많이 사용되기도 한다.

- Basic Policies Authorize/Prohibit

```
Policy ( authorize | prohibit ) policy-Name "{"
        subject         object-Expression;
        target          object-Expression;
        operation       peration-Expression;
        [ when          constraint-Expression;]
        [ while         constraint-Expression;]
        [ entry         action-Expression;]
        [ exit          action-Expression;] "}"
```

어떠한 개체가 어떠한 목적으로 어떤 일을 하는지에 대하여 접근을 허락하는지 금지하는지를 결정한다. 이때 제약사항을 포함시킬 수 있다.

- Basic Policies Initiate/Terminate

```
Policy ( initiate | terminate ) policy-Name
[reuse-Flag] "{"
        [ subject       object-Expression;]
        target          object-Expression;
        operation       operation-Expression;
        on              event-Expression; "}"
```

특정 이벤트가 발생할 경우 이 이벤트의 제약사항과 맞으면 해당 정책을 중지하거나 초기화 하게 된다.

- Basic Policies Delegate

```
Policy delegate policy-Name[ forwardable-Flag ] "{"
        grantor         object-Expression;
        grantee         object-Expression;
        target          object-Expression;
        operation       operation-Expression;
        [ when          constraint-Expression;
        [ while         constraint-Expression;
        [ entry         action-Expression; ] "}"
```

Delegate는 다른 개체로 전송하는 권리를 얻기 위한 정책으로 사용된다. when, while 제약사항이 만족할 경우 grantor 대상이 target에 대하여 전송할 권리를 갖게 된다.

- Basic Policies Revoke

```
Policy revoke policy-Name "{"
        grantor         object-Expression;
        grantee         object-Expression;
        target          object-Expression;
        operation       operation-Expression;
        on              event-Expression; "}"
```

Revoke는 grantor가 전송을 철회하기를 원할 경우 사용된다.

상황인식의 제약과 이벤트에 대한 명세에 대하여 간단하게 알아본다.

- 제약사항과 이벤트에 대한 명세

제약사항에과 이벤트의 명세는 기본적인 개체의 정의를 이용하여 사용하였다. 기본적인 문법은 다음과 같다.

```
"("context-Object-Name
    {"("attr-Name attr-Expression")" } ")"
```

인자의 표현은 숫자나 문자 같은 상수로 표현할 수 있고 변수 혹은 관계식 표현이 가능하다. 아래는 기본적인 제약사항 명세의 예를 표현한 것이다.

```
(location(person kim)(locationName rm707))
(temperature(tempValue>40C)(locationName?X))
```

처음의 제약사항은 Kim이라는 이름을 가진 사람이 707호 방에 있을 경우 참이 된다. 두 번째는 방에 온도가 40C보다 높은 지역의 값 ?X가 있을 경우 참이 된다. 만약 참이 된다면 위의 상태에 만족되는 값을 가진 방의 이름이 변수에 반환된다.
정책명세 언어는 완벽하게 제약사항을 구성하기 위해 기본적인 논리 연산을 사용할 수 있다 : and, or, not.

- 이벤트 명세

이벤트는 한 지점에서 한번 센서에 정의되고 센서에서 발견되는 기본적인 이벤트와 어떤 제약사항의 표현값의 변화를 감지하는 것을 나타낼 수 있다. 제약사항의 변화 이벤트의 문법은 다음과 같다.

"("becomes constraint-expression ")"

다음은 기본적인 몇몇 이벤트의 예를 나타내고 있다.

```
(enters(person kim)(locationName 방707))
(becomes(temp(tempValue>40C)(locationName ?X)))
```

제약사항과 마찬가지로 이벤트의 경우에도 and, or, repeat, next와 같은 연산을 사용할 수 있다.

이러한 제약사항과 이벤트의 명세와 상황인식 접근제어 정책은 위 3장의 구조에서 Connection Interface와 Query/Event 핸들러 사이의 통신과 관련되어 있다. [그림 5]를 보게 되면 Query(질의)와 Response(응답)과의 통신이 제약사항의 명세를 이용하게 되고 Event Detection Request와 Event Notification은 이벤트 명세를 이용하게 된다.

4.2 시스템의 응용

이러한 유비쿼터스 상황 인신 응용 시스템을 가지고 응용할 수 있는 시나리오를 구성하였다.

1) 빛 감지 센서가 작동한 후, 모션 감지 센서가 작동한 경우: 정상적인 상황으로 인식하고 개인 시스템을 ON할 수 있는 권한이 주어진다(가상 시뮬레이션)

2) 빛 감지 센서가 작동하지 않고, 모션 감지 센서만 작동할 경우: 부적절한 상황으로 경보 발생 메시지. 이 경우는 해당 방에 불순한 의도를 가지고 침입한 상황으로 가정(가상 시뮬레이션)

3) 빛 감지 센서만 작동중이고 모션 감지 센서가 장시간 작동 안할 경우: 자리를 비운 것으로 인식하고 해당 방의 등을 끌 것을 알리는 메시지를 출력한다

이러한 상황은 빛 감지와 모션 감지 센서만을 활용한 예이고 여기에 PDA를 통한 사용자 인증을 추가 할 수 있다.

4) PDA를 이용한 동일한 사용자 ID가 양쪽 방에 동시에 접근되는 경우: 양 쪽 모두 시스템 이용 권한을 없애고 경보를 울리게 한다.

5) 방을 사용하는 사용자 ID가 하나도 없으나 빛 감지 센서가 ON상태로 인식하고 있을 경우: 관리자에게 해당 방의 등을 끌 것을 지시(혹은 자동 소등).

6) 방을 사용하는 사용자 ID가 하나도 없으나 모션 감지 센서가 ON상태로 인식하고 있을 경우: 부정한 침입이 이루어 진 상황으로 인식.

또 다음과 같은 이벤트 통보와 질의 요청을 할 수도 있다.

- 방1에 있는 사람의 총 수 혹은 ID 리스트 요구
- 사용자 ID5번이 현재 있는 방 번호
- 방2에 사용자 ID8번이 있는지 확인 요구
- 사용자 ID6번이 방2에 진입할 경우 통보
- 사용자 ID7번이 방1에 있으나 모션 감지 센서가 동작하지 않을 경우(해당 사용자를 자리 비움으로 가정)

접근제어 정책 명세와 이벤트 명세 언어를 이용하여 위와 같은 다양한 시나리오 상황을 정의하고 응용하는데 사용할 수 있다.

## 5. 결론

유비쿼터스 컴퓨팅 환경은 현재 무선 네트워크와 모바일 네트워크 환경과 함께 발전하고 있으며, 일부에서 실제 유비쿼터스 컴퓨팅 시스템이 우리 생활에 사용되고 있고, 앞으로는 더 많이 이용 될 것이다. 이러한 유비쿼터스 컴퓨팅 시스템을 여러 분야에 적용시키기 위해서 실제 유비쿼터스 컴퓨팅 시스템을 설계, 구축하고 적절한 시나리오에 맞추어서 동작시켜봄으로써 유비쿼터스 시스템의 발전에 도움을 줄 수 있을 것이다. 또한 이러한 시스템을 설계하기 위하여 센서, DB, PDA등의 여러 장비를 실제 다뤄 볼 수 있고, 이러한 시스템을 보다 안전하게 제어하는 기술을 습득하고 이러한 시스템을 실제 생활에 적용시키는 시나리오에 대하여 정리 분석하였다.

## <참고문헌>

[1]J.I. Hong & J.A., "An 구조 Approach to 상황 인식 Computing," HCI Journal 16(2-3),2001

[2]G. Judd & P. Steenkiste, "Providing Contextual Information to Pervasive Computing Application," IEEE PerCom'08,2003

[3]J. Al-Muhtadi et al, "Cerberus: A 상황 인식 Security Scheme Smart Spaces," IEEE PerCom'03, 2003

[4]Young-Chul Shim, "Distributed Processing of 상황 인식 접근제어 in 유비쿼터스 Computing Environments," ICCSA 2005