유비쿼터스 컴퓨팅에서 상황인식을 위한 컨텍스트 스크립트 언어 및 언어 처리기

심 춘 보[†] · 김 용 기[††] · 장 재 우[†††] · 김 정 기[††††]

요 약

상황인식 처리기술을 위한 응용 소프트웨어를 개발하기 위해서는 다양한 상황인식에 대한 판단과 그에 따른 적절한 처리를 모두 프로그램 언어로 기술해야 하며, 이는 무수한 프로그램 코드의 반복과 소프트웨어 생산성의 저하를 초래한다. 따라서 본 논문에서는 상황인식을 위해 요구되는 일련의 복잡한 과정을 간략하고 명료하게 기술하고 상황에 대한 정의를 미리 규격화된 구문으로써 표현함과 동시에 자동적으로 처리할 수 있는 상황인식을 위한 컨텍스트 스크립트 언어 및 언어 처리기를 구현한다. 제안하는 컨텍스트 스크립트 언어는 주어진 상황을 효율적으로 정의 할 수 있는 기능을 제공함은 물론 보다 다양한 상황을 범용적으로 표현할 수 있도록 설계한다. 아울러 제안하는 언어 처리기의 유용성을 보이기 위해, 상황인식에 근거하여 음악 재생 서비스를 제공하는 응용 시스템을 구축한다.

Context Script Language and Language Processor for Context-Awareness in Ubiquitous Computing

Choon-Bo Shim[†] · Young-Ki Kim[††] · Jae-Woo Chang[†††] · Jeong-Ki Kim[††††]

ABSTRACT

In order to develop an application software for context-awareness techniques, we should program both all decisions on variable context-awareness and appropriate process with some program languages. These cause a loss of software production and unlimited repetition of program code. In this paper, we implement a context script language and language processor which can simplify a series of involved process acquired for context-awareness and describe them clearly. In addition, it can represent a definitions of context as standard syntax as well as accomplish them automatically. The proposed context script language provides functionality which can not only define efficiently a given context but also describe a variety of context with general purpose. Also, for the usefulness of the language processor, we build an application system which can provide music play service based on context-awareness.

키워드 : 유비쿼터스 컴퓨팅(Ubiquitous Computing), 상황인식(Context-Awareness), 컨텍스트 스크립트 언어(Context Script Language Processor)

1. 서 론

'사람을 포함한 현실 공간에 존재하는 모든 대상물을 기능적, 공간적으로 연결해 사용자에게 필요한 정보나 서비스를 즉시에 제공할 수 있는 컴퓨팅 환경을 유비쿼터스 컴퓨팅[1-5]'이라고 정의하고 있다. 유비쿼터스 컴퓨팅 기술은 위치기반서비스(LBS), 텔레매틱스, 여러 사람과 정보 공유 및 협업이 필요한 사무실 등 우리 일상생활의 다양한 서비스 분야에 활용될 수 있다. 한편 상황인식(context-awareness) 기술[6-9]은 일상생활 곳곳에 편재된 센서 및 컴퓨터들이 수집한 각종 환경정보를 효과적으로 상호 공유하여 사용자 및 주변 환경의 컨텍스트(context)를 감지하고 분석해서 사용자가 필요로 하는 서비스를 효율적으로 제공하기 위함이며, 이는 사용자를 중심으로 하는 주변 환경과 사용자간 혹은 사용자와 장치간의 상호 운용성을 지능적, 자동적으로 선택하여 지원해 줌으로써 사용자로 하여금 정보 획득 및 실행을 보다 용이하도록 지원한다. 상황인식 기술은 '일상 환경 속에 편재된 언제 어디서나 이용 가능한 컴퓨팅 환경'인 유비쿼터스 컴퓨팅 환경에서 가장 중요한 요소 기술 중에 하나로 발전하고 있다. 그러나 유비쿼터스 컴퓨팅 환경에서 상황인식을 위한 사용자 및 주변 환경의 컨텍스트 즉 주어진 상황을 적절히 정의하고 표현하기 위한 상황 정의 언어와 관련된 연구는 그리 많지 않다. 유비쿼터스 컴퓨팅에서 상황인식을 위한 다양한 응용 소프트웨어를 개발하기 위해

† 준 회 원 : 부산가톨릭대학교 컴퓨터정보공학부 교수
†† 준 회 원 : 전북대학교 대학원 컴퓨터공학과
††† 종신회원 : 전북대학교 컴퓨터공학과 교수
†††† 정 회 원 : 한국전자통신연구원 임베디드S/W 연구단 선임연구원

논문접수 : 2004년 7월 20일, 심사완료 : 2004년 12월 6일

서는 주어진 상황을 보다 편리하고 용이하게 정의할 수 있는 상황 정의 언어가 필요하다.

이를 위해 본 논문에서는 유비쿼터스 컴퓨팅을 위한 상황인식 기반의 컨텍스트 정의 스크립트 언어 및 언어 처리기를 설계 및 구현한다. 제안하는 컨텍스트 정의 스크립트 언어는 일련의 복잡한 과정을 간략하고 명료하게 기술하고 주어진 응용에 따른 상황(컨텍스트)에 대한 정의를 미리 규격화된 구문으로써 편리하고 쉽게 표현할 수 있음은 물론 특정 시스템에 의존적이지 않으며 범용적으로 기술할 수 있는 장점을 가진다. 컨텍스트 정의 스크립트 언어의 구성 요소는 전체적으로 컨텍스트 객체의 정의와 컨텍스트의 정의 및 삭제 그리고, 컨텍스트가 구체화된 인스턴스의 삽입 및 삭제, 인스턴스의 시작 및 정지로 구성된다. 아울러 제안하는 스크립트 언어의 유용성을 테스트하기 위해 상황인식 시스템을 구현한다. 제안하는 시스템은 미들웨어와 컨텍스트 서버로 구성된다. 미들웨어는 블루투스[10-12] 무선 통신 기술을 이용하여 이동성을 지닌 이동 노드를 발견, 등록 및 실행하는 역할을 담당한다. 그리고 컨텍스트 서버는 주어진 이동 노드의 정보와 컨텍스트 정보를 효율적으로 데이터베이스 서버에 저장 및 관리하는 기능을 수행한다. 제안하는 시스템을 기반으로 컨텍스트 정보에 근거한 음악 재생 서비스를 제공하는 응용 시스템을 구축한다.

본 논문의 구성은 다음과 같다. 제2장에서는 유비쿼터스 컴퓨팅에서 상황인식 처리 기술과 관련된 연구들을 소개한다. 제3장에서는 상황인식을 위한 컨텍스트 정의 스크립트 언어와 언어 처리기에 대해서 자세히 기술한다. 제4장에서는 앞 장에서 설명한 상황인식 기반 컨텍스트 스크립트 언어 처리기를 위한 응용 시스템 및 테스트에 대해서 기술한다. 마지막으로 제5장에서는 결론 및 향후연구를 제시한다.

2. 관련 연구

유비쿼터스 컴퓨팅에서 상황인식 처리 기술과 관련된 몇 가지 연구를 소개한다.

첫째, 애리조나 주립대학의 연구[6]는 유비쿼터스 컴퓨팅 환경에서 상황인식을 위해 각각의 장치로부터 끊임없이 상태를 파악하고 그 파악된 상태에 따라 미리 정해진 적절한 조치를 취하는 미들웨어를 제공하고 있다. 미들웨어는 기존의 RCSM(Reconfigurable Context-Sensitive Middleware)을 확장한 것이다. 아울러 각각의 상황을 정의하기 위해 CA-IDL(Context-enabled Interface Definition Language)이라고 하는 상황 정의 언어를 제공하고 있으며, 각 상황에 대한 Context tuple을 다음과 같이 정의하고 있다.

$$\text{Context tuple} : \langle a_1, \cdots, a_n, t_m \rangle$$

여기서 n은 # of unique contextual data는 나타내며, a_i는 value를 의미한다. 그리고 t_m : tuple creation time를 나타낸다. 예를 들어, Context tuple의 요소가 위치, 방향, 속도라고 가정한다면 그에 해당되는 tuple은 $\langle (x, y), d_i, m, t \rangle$이 된다. 이렇게 정의된 tuple에서 객체가 북쪽으로 향하고 있다는 것을 가정하면 $\langle (x, y), north, m, t \rangle$로 정의할 수 있고, 이러한 상황이 발생하게 되면, 그에 적절한 메서드를 호출하게 된다.

둘째, IRISA/INRIA의 연구[7]는 이동성을 가진 사용자에 대해서 사용자의 주변 환경 즉 상황인식을 이용하여 그에 대한 서비스의 레벨을 향상시키기 위해 Contextual Object (CO)에 기반을 둔 하부 구조(infrastructure)를 제안하였다. 즉, 사용자가 처해 있는 주변 환경 즉, 상황을 감지하여 그에 적합한 최적의 서비스를 제공하기 위해 CO를 다음과 같이 정의하고 있다.

$$CO(id) : \langle \text{Variant } V_x \text{ Attribute } A_i : \text{value, Atribute } A_j : \text{value, } \cdots \rangle$$

예를 들어, 웹 문서(Web Document)의 경우 $\langle V_1$; Location : L_1, Language : French, Browser : with frame$\rangle$, $\langle V_2$; Location : L_2, Language : English, Browser : without frame$\rangle$라는 컨텍스트 객체가 있다면, 사용자의 위치나 언어, 브라우저 타입에 따라 상황에 적합한 해당 Variant를 선택하여 그에 알맞게 웹 문서를 보여주게 된다. 비슷한 예로서, 사용자의 통신 환경이 High Bandwidth인지 Low Bandwidth인지에 따라 동일한 이미지 데이터라도 통신 Bandwidth에 적합한 최적의 해상도를 선택하여 전송한다.

셋째, AT&T 연구[8]는 실내(indoor)에서 사용자의 위치를 탐지 및 추적하여 사용자의 주변 환경에 맞는 상황인식 응용 서비스(Context-Awareness Application Service)를 제공하고 있다. 각 사용자는 배지(Bat unit)를 착용한 상태로 이동을 하고 있으며 각 해당 receiver에서는 각 사용자의 배지를 감지하여 위치를 파악한다. 이때 오직 하나의 receiver가 배지를 감지하는 것이 아니라 다수개의 receiver가 배지를 탐지하여 각 receiver와 배지 사이의 전파의 세기를 계산하여 사용자의 위치를 파악한다.

마지막으로, Lancaster GUIDE 프로젝트[9]는 영국의 Lancaster 도시를 방문한 여행객의 편의와 효율적인 여행 서비스를 제공하기 위해 여행객에게 Fujitsu TeamPad 7600이라는 휴대용 pen-based tablet PC를 휴대하게끔 함으로써 여행객의 위치 정보를 탐지 및 추적하여 도시의 여행객에게 여행 가이드를 하는 위치기반 서비스 시스템을 제안하였다. 휴대용 장치는 Window CE가 탑재된 Pen-based tablet PC로서 사용자가 위치해 있는 주변의 흥미있는 관광 명소에 대한 정보를 HTML 브라우저를 통해 브라우징 하도록 설계하였다. 사용자의 위치 탐지는 그도시의 각 지역마다 많은 셀(cell)들을 설치하고 또한 다수 개의 셀들을 관리하는 셀

서버(cell server)를 두어 각각의 셀들 간의 통신을 통해 가능하게 한다.

3. 상황인식을 위한 컨텍스트 스크립트 언어

유비쿼터스 컴퓨팅에서 상황인식을 지원하는 다양한 응용 소프트웨어를 개발하기 위해서는 주어진 상황(컨텍스트)을 보다 편리하고 쉽게 정의할 수 있는 컨텍스트 정의 언어가 필요하다. 따라서 본 장에서는 다양한 상황에 대해서 이를 효과적으로 표현할 수 있는 컨텍스트 정의 스크립트 언어 및 언어 처리기에 대해서 기술한다.

3.1 컨텍스트 스크립트 언어의 구성 요소

상황인식 처리 기술을 위한 응용 소프트웨어를 개발하기 위해서는 다양한 상황인식에 대한 판단과 그에 따른 알맞은 처리를 모두 프로그래밍 언어로 기술해야 하며 이러한 작업은 무수한 코드의 반복과 생산성의 저하를 초래한다. 이러한 일련의 복잡한 과정을 간략하고 명료하게 기술하고 상황에 대한 정의를 미리 규격화된 구문으로써 표현함과 동시에 자동적으로 처리할 수 있는 스크립트 언어가 요구된다. 이를 위한 언어로서 IRISA/INRIA의 CA-IDL[6]이라는 언어가 제안되었지만, 특정 시스템에 의존적이며, 범용적이지 못한 단점이 있다. 따라서 본 논문에서는 주어진 상황을 효율적으로 정의할 수 있는 기능을 제공함과 동시에 보다 다양한 상황을 범용적으로 표현할 수 있는 컨텍스트 정의 스크립트 언어를 설계하는 데 초점을 둔다.

컨텍스트 정의 스크립트 언어의 구성 요소는 전체적으로 컨텍스트 객체의 정의와 컨텍스트의 정의 및 삭제 그리고, 컨텍스트가 구체화된 인스턴스의 삽입 및 삭제, 인스턴스의 시작 및 정지로 구성된다.

3.1.1 컨텍스트 객체 정의

컨텍스트 객체는 상황인식 소프트웨어 주위의 모든 사물이나 장치를 나타내는 논리적인 개념이다. 다시 말해 개념적으로는 C++ 언어에서의 객체와 유사하지만 용법에 있어서는 C언어의 구조체와 비슷하다. 객체를 이루는 각각의 속성값들중 일부는 하드웨어적인 장치에 의해서 갱신되며, 그 값들은 뒤에서 언급할 컨텍스트의 조건절(condition clause)에서 사용된다. 컨텍스트 객체의 정의 용법은 (그림 1)과 같다. 진하게(bold) 표기된 문자는 예약어로서 대/소문자를 가리지 않는다. object_name과 element_name은 첫 문자로서 숫자나 특수문자를 사용할 수 없고, 대/소문자를 구별하며, 256자 이내여야 한다. 일반적인 변수 명명 규칙과 유사하다고 할 수 있다. data_type은 문자와 문자열 데이터 타입으로서 string이 있고, 수치 데이터 타입으로서 int, float, long, double, time, date가 있다.

```
ContextObject object_name (
        data_type element_name1,
        data_type element_name2,
        data_type element_name3
                :
                :
)
```

(그림 1) 컨텍스트 객체의 정의

3.1.2 컨텍스트의 정의 및 삭제

컨텍스트의 정의는 주어진 조건(condition)에 일치하는 상황이 발생하면 어떠한 동작(action)을 수행함을 규정하는 것으로 주어진 조건절과 그 조건절이 만족되었을 때 수행하는 동작을 나열한다. 여기서는 어떠한 컨텍스트의 틀(template)을 정의하는 것으로 실제 값은 컨텍스트 인스턴스가 추가될 때 동작이 가능한 형태가 된다. 즉, 데이터베이스에 비유를 한다면 실제 데이터베이스에 아직 내용은 들어있지 않은 테이블만 구성되어 있는 형태라고 볼 수 있다. 컨텍스트 정의 용법은 (그림 2)와 같다.

```
ContextDefine   context_name
Condition       (condition_expression) [during time_value]
Action          function_name
                [before/when/after time_value] or
                [from time_value to time_value]
Param           (param1, param2, ··· )
```

(그림 2) 컨텍스트의 정의

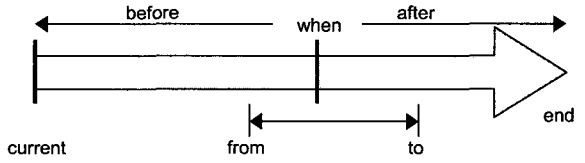
컨텍스트를 정의할 때 ContextDefine이 사용되며, 부가적으로 Condition, Action, Param 절이 함께 사용된다. 용법은 ContextDefine에 의해서 컨텍스트가 정의되며, Condition에 의해서 해당되는 조건이 결정되고, Action과 Param에 의해서 수행해야 할 함수가 호출되어 매개변수가 전달된다. 여기서 ContextDefine의 context_name은 컨텍스트의 이름으로서 앞에서 언급한 object_name과 이름 명명 규칙이 같다.

첫째, Condition의 (condition_expression)과 옵션 값인 [during time_value]에 대해서 살펴보면, condition_expression은 내부에서 괄호를 중첩해서 사용할 수 있으며, 조건절로서 표현된다. 조건절을 표현하기 위한 관계 연산자로서 양쪽의 변수사이에 등호(=)와 부등호(<, >)가 사용된다. 또한 필요한 연산자로서 논리연산자가 있다. 각 논리 연산자는 AND, OR, NOT이 있다. [during time_value]는 옵션 항으로서 생략이 가능하고, 만약 during time_value항이 추가되면 조건 절이 time_value의 시간만큼 지속 되어야 참이 된다. 여기서 time_value는 시간의 값으로서 표현되는데, 수치 뒤에 s,m,h 가 추가됨에 따라 각각 초, 분, 시간으로 구분된다. 예를 들어 1시간 12분 3초를 표현한다면, 1h12m3s로 표현 할 수 있다. 만약 "A라는 사람이 B 장소나 혹은 C 장소에 10초 동안 있게 된다면"을 정의하면, ContextObject

person이 이미 정의가 되었고 person의 속성으로서 location이 정의되었음을 전제로 할 때, Condition을 표현하면 아래와 같다.

Condition (((A_person.location = B_place) during 10s) or (A_person.location = C_place) during 10s)

둘째, Action절의 function_name/NULL과 [while/when/after time-value] or [from time_value to time_value]에 대해 설명한다. function_name은 조건이 만족되었을 경우 실행하는 함수의 이름이 기술된다. 호출하는 함수는 용도에 따라 미리 만들어져 준비된 것일 수도 있고, 사용자의 필요에 따라 새로 만들어서 사용하는 경우도 있다. 만약 함수를 호출할 필요가 없을 경우에는 function_name에 NULL을 사용한다. function_name 다음의 시간에 대한 옵션항은 함수를 수행함에 있어 시간에 따라 어떻게 실행할 것인지를 결정하며, 옵션 값으로서 생략이 가능하다. 어떠한 시점(when)을 기준으로 before는 그 시점 이전에 한번만 Action을 수행하고, after는 그 시점 이후에 한번만 Action을 수행하는 것이다. from, to는 어떤 정해진 시간동안 계속적으로 실행하는 것으로 (그림 3)에서 도식화하여 나타낸다.



(그림 3) Action에 대한 시간과의 관계

마지막으로 Param(param1, param2,…)은 Action에 의해서 실행되는 함수에 전달할 매개변수의 리스트이다. 만약 전달할 매개변수가 없을 경우에는 Param NULL로 표현할 수 있다.

컨텍스트가 정의 되었다면 필요에 따라 삭제하는 경우도 있다. 이미 정의된 컨텍스트의 삭제 방법은 (그림 4)와 같다. 컨텍스트가 삭제 될 때는 컨텍스트의 인스턴스도 모두 소멸되며, 컨텍스트 인스턴스가 활성화되어 계속적으로 조건을 체크하고 있는 상태 역시 모두 정지된다.

ContextDestroy	context_name

(그림 4) 컨텍스트의 삭제

3.1.3 컨텍스트 인스턴스의 삽입 및 삭제

컨텍스트가 정의되어 컨텍스트 인스턴스를 담을 수 있는 스키마가 만들어졌다면, 컨텍스트에 내용을 담아 구체화하는 것이 컨텍스트 인스턴스이다. 용법은 (그림 5)와 같다. 컨텍스트에 컨텍스트 인스턴스를 추가 할 때 ContextAdd, Condition, Param의 절로 표현한다. ContextAdd의 context_name에는 인스턴스가 삽입될 컨텍스트의 이름을 기술하며, instance_name에는 컨텍스트 인스턴스의 이름을 기술한다. Condition절과 Param절의 각 value에는 리터럴 즉, 상수가 기술된다. 컨텍스트 정의할 때 condition_expression의 조건 표현부에서 컨텍스트 오브젝트의 속성값과 '변수'가 비교되는 경우가 있는데, 이 '변수'에 해당하는 실제 값이 컨텍스트 인스턴스 삽입시 Condition절에 사용되는 각 condition_value이다. 같은 맥락으로 Param절의 parma_value의 각 값은 컨텍스트 정의시 Action에서 호출되는 함수의 Param에 해당되는 변수의 실제 상수값을 담고 있다.

ContextInsert	instance_name **Into** context_name
Condition	(condition_value1, condition_value2,…)
Param	(param_value1, param_value1,…)

(그림 5) 컨텍스트 인스턴스의 정의

컨텍스트 인스턴스의 삭제는 (그림 6)과 같다. context_name에는 삭제할 인스턴스가 있는 컨텍스트 이름을 기술하고, instance_name에는 삭제할 컨텍스트 인스턴스 이름을 기술한다.

ContextDelete	instance_name **In** context_name

(그림 6) 컨텍스트 인스턴스의 삭제

3.1.4 컨텍스트 인스턴스의 시작 및 정지

컨텍스트의 정의와 컨텍스트 인스턴스의 삽입이 완료되면 실제 그 인스턴스를 활성화하여 계속적으로 조건의 상태를 판단하고 해당되는 행동을 취하도록 하는 것이 컨텍스트 인스턴스의 시작이다. 그에 반해 컨텍스트 인스턴스의 상황판단을 멈추도록 하는 것이 컨텍스트 인스턴스의 정지로 그 용법은 (그림 7)과 같다.

ContextStart	instance_name **In** context_name
ContextStop	instance_name **In** context_name

(그림 7) 컨텍스트 인스턴스의 시작과 정지

context_name에는 시작하거나 정지할 컨텍스트 인스턴스가 있는 컨텍스트 이름을 기술하고, instance_name은 시작하거나 정지할 컨텍스트 인스턴스를 기술한다. 어떤 컨텍스트의 모든 인스턴스를 시작하거나 정지할 경우에는 instance_name에 별표(*)를 사용한다.

3.1.5 컨텍스트 스크립트 언어의 예제

앞에서 언급한 각 용법을 이용해 간단한 예제를 소개한다. 예를 들어, 주어진 컨텍스트가 각 사람에 따라서 정해진 음

악이 있고, 이 사람들이 각 방을 이동하는데, 만약 누군가 방에서 2초이상 이상 대기하게 된다면, 1초 후에 정해진 음악을 재생하는 것을 표현하면 (그림 8)과 같다.

```
ContextObject person (
        string name,
        string location
);

ContextDefine music
Condition ((person.name = someone_name) AND (person.location =
someone_location) During 2s)
Action PlayMusic When 1s
Param (music_name)
```

(그림 8) 컨텍스트 객체와 컨텍스트의 정의의 예

컨텍스트 객체와 컨텍스트가 위와 같이 정의되었을 때, ***'김철수'***라는 사람이 ***'7401'***호의 방에 있을 때는 ***'아리랑'***을 재생하고, ***'이영희'***라는 사람이 ***'7429'***호의 방에 있을 때는 ***'도라지'***를 재생할 경우에는 (그림 9)와 같이 표현한다.

```
ContextInsertkim Into music
Condition('김철수', '7401')
Param ('아리랑') ;

ContextInsert lee Into music
Condition ('이영희', '7429')
Param ('도라지') ;
```

(그림 9) 컨텍스트 인스턴스의 삽입의 예

(그림 8)에서의 someone_name, someone_location, music_name이 인스턴스에 의해서 구체화 되었다. 이 인스턴스의 각 조건을 주기적으로 체크하도록 하기 위해서는 (그림 10)과 같이 표현한다.

```
ContextStart * In music
```

(그림 10) 컨텍스트 인스턴스의 시작

ContextStart 구문 사용시 별표(*)를 사용함으로써 music의 모든 인스턴스의 조건이 체크되도록 한다.

3.2 컨텍스트 스크립트 언어의 구문 구조

컨텍스트 스크립트 언어에서 사용되는 명령어를 위한 정규 표현식을 이용한 구문 구조에 대해서 기술한다.

3.2.1 스크립트의 예약어

컨텍스트 스크립트 언어에서 사용되는 예약어는 명령어, 부가 명령어, 시간 명령어, 관계 연산자, 논리 연산자, 상수값, 주석기호 그리고, 데이터 타입이 있다. 각각의 자세한 설명과 내용은 <표 1>과 같다.

〈표 1〉 스크립트의 예약어

종 류	심 볼	역 할	예약어
명령어	command	스크립트 명령어	ContextObject, ContextDefine, ContextDelete, ContextInsert, ContextStart, ContextStop
부가 명령어	command_ad	스크립트 명령어를 위한 추가적인 명령어	Condition, Action, Param, in, into
시간 명령어	command_time	스크립트 명령어를 위한 시간 관련 추가 명령어	during, when, while, after, period
관계 연산자	re_op	대소 비교를 위한 연산자	=, <>, <,>, <=, =>
논리 연산자	lo_op	논리 연산을 위한 연산자	and, or , not, AND, OR, NOT
상수값	const	스크립트에서 사용되는 상수값	NULL, TRUE, FALSE, null, true, false
데이터 타입	data_type	스크립트에서 사용되는 변수 데이터 타입	string, int, float, long, double, time, date

3.2.2 명령어의 구문 구조

여기에서는 명령어의 실제적인 구문 구조를 표현한다. 편의상 공백문자(white space)는 구문 구조의 가독성을 위해서 심볼로 표현하지 않고 실제 공백문자로서 표현한다.

○ ContextObject

컨텍스트 객체를 이루는 데이터 타입과 변수 이름으로 표현되는 각 element를 예약어 심볼인 data_type과 정규 표현식으로 표현되는 variable로 나타하면 다음과 같다.

element : data_type variable

전체적인 컨텍스트 객체를 정의하는 문장을 위에서 정의한 element를 이용하여 정의하면 다음과 같다.

command context_name [( ] element ([,]element) * [)]

○ ContextDefine

컨텍스트를 정의하는 ContextDefine의 Condition절에 있어서, 조건을 표현하는 조건절(condition_expression)을 위해 단순 비교를 위한 조건식을 정의해야 해야 하고, 정의된 조건식이 논리 연산자에 의해 중첩되는 조건절을 정의해야 한다. 즉, 조건식은 컨텍스트 객체의 속성값과 일반 변수의 비교, 컨텍스트 객체의 속성값과 다른 컨텍스트 객체의 속성값과의 비교, 일반 변수끼리의 비교, 일반 변수와 컨텍스트 객체의 속성값과의 비교를 위한 조건식으로 구성되며, 이 조건식이 중첩되어 조건절을 구성한다. 이러한 표현은 다음과 같다.

condition : context_name[.]context_obj_name re_op var |
context_name[.]context_obj_name re_op context_nam
e[.]context_obj_name |
var re_op var |
var re_op context_name[.]context_obj_name

condition_expression : [(] condtion[lo_op condition]* [)]

위에서 정의된 조건절의 구문 규칙을 이용하여 Cotext Define의 구문을 정의하면 다음와 같다.

command context_name
command_ad condition_expression (command_time time_value){0,1}
command_ad func_name (command_time timevalue){0,1}
command_ad [(] func_name ([,]func_name])* [)]

- **ContextDestroy**

정의된 컨텍스트를 삭제하는 CotextDestroy는 다음과 같다.

command context_name

- **ContextInsert**

컨텍스트 인스턴스를 삭제하는 ContextInsert의 구문 구조는 아래와 같다. Condition과 Param절에 실제 값이 들어가는 구조로서 일반상수가 1번 이상 반복되는 구조이고, Condition과 Param절의 실제 구문 구조는 다음과 같다.

command context_ins_name command_ad context_name
context_ad [(] value_const ([,]value_const)* [)]
context_ad [(] value_const ([,]value_const)* [)]

- **ContextDelete**

앞에서 언급한 ContextDelete는 컨텍스트 자체를 삭제하는 명령어의 구문 구조이고, 여기서 언급하는 것은 컨텍스트 인스턴스를 삭제하는 명령어의 구문구조로서, 구문구조의 형태에 따라 수행하는 역할이 다르다. 구문구조는 다음과 같다.

command context_ins_name command_ad context_name

- **ContextStart**

이미 정의된 컨텍스트 인스턴스들을 활성화시키는 명령어로서 명령어 가운데 Period절에 해당하는 부분이 옵션으로서 생략이 가능하다.

command context_ins_name command_ad context_name
(command_time timevalue){0,1}

- **ContextStop**

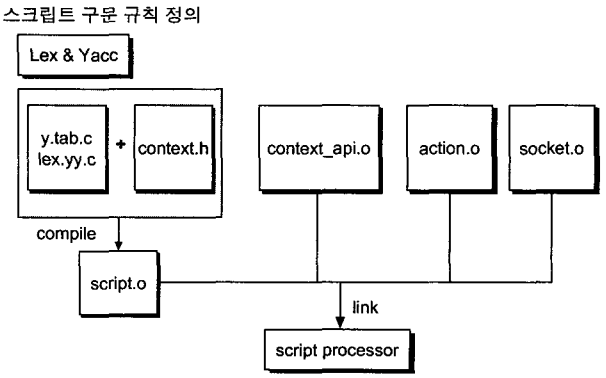
ContextStop의 구문구조는 다음과 같다.

command context_ins_name command_ad context_name

3.3 컨텍스트 스크립트 언어 처리기

컨텍스트 스크립트 언어 처리기는 정의된 구문에 따라 생성된 스크립트 언어 파일을 문법 분석 및 구문 분석을 통해 처리하는 프로그램이다. 컨텍스트 스크립트 처리기에 대해서 전체적인 구조와 스크립트의 처리 과정을 기술한다.

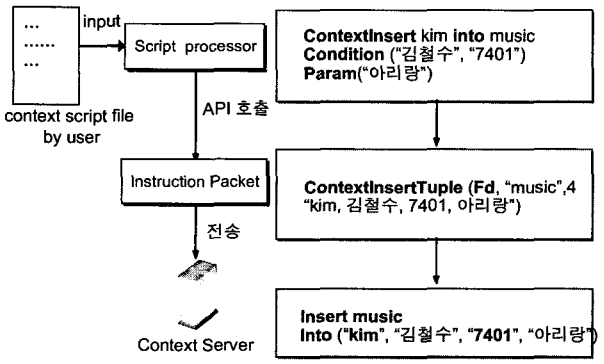
먼저, 컨텍스트 스크립트 언어를 처리할 수 있는 언어 처리기의 전체적인 구조는 (그림 11)과 같다.



(그림 11) 컨텍스트 스크립트 언어 처리기의 전체 구조

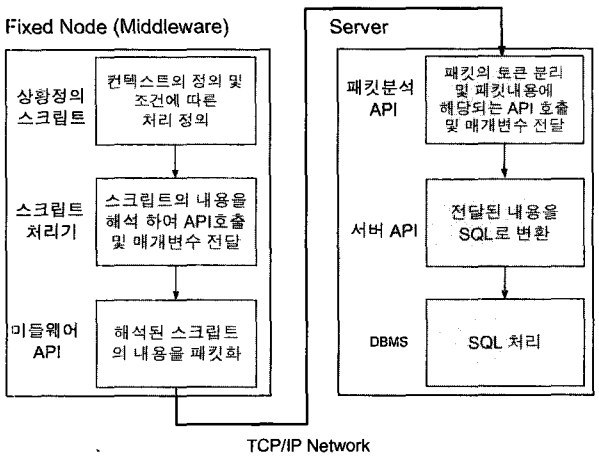
여기서, script.o는 Lex와 Yacc를 통해 상황정의 스크립트 언어의 문법 및 구문 규칙을 정의하여 컨텍스트 API를 호출하는 내용이 내재되어 있다. context_api.o에는 스크립트에서 기술된 모든 표현을 처리하는 API가 기술되어 있고, action.o에는 어떠한 조건을 만족했을 때 수행하는 함수들 즉, 컨텍스트 정의시 Action절에서 호출하는 함수들의 실제 내용이 기술된다. socket.o에는 스크립트가 처리 종료 후 내용을 서버 측으로 전송할 때 사용하는 소켓 관련 API가 포함된다. 앞에서 언급한 모듈이 모두 링크되어 하나의 실행 파일이 되어 컨텍스트 스크립트 언어를 처리하는 프로그램으로서 동작한다.

컨텍스트 스크립트 언어는 스크립트 언어 처리기를 통해 번역되어 스크립트의 내용을 수행할 수 있는 API가 호출되어 서버로 전달되며, 이는 (그림 12)와 같다.



(그림 12) 컨텍스트 스크립트 언어의 처리 과정

스크립트 언어 처리기는 고정노드(fixed node)에서 미들웨어로서 수행되며 스크립트 언어가 처리되는 일련의 과정을 살펴보면, 앞에서 언급한 구문에 따라 기술된 컨텍스트 스크립트 언어는 스크립트 언어 처리기를 통해 스크립트의 내용이 해석되어 API가 호출된다. 이 API는 해석된 API의 내용을 패킷화하여 소켓을 통해 컨텍스트 서버로 전달한다. 컨텍스트 서버는 이 패킷을 수신받고 해석하여, 서버측의 DBMS관련 API를 수행해서 스크립트의 처리 내용을 데이터베이스에 저장한다. 컨텍스트 스크립트가 고정노드에서 서버까지 처리되는 과정을 나타내면 (그림 13)과 같다.



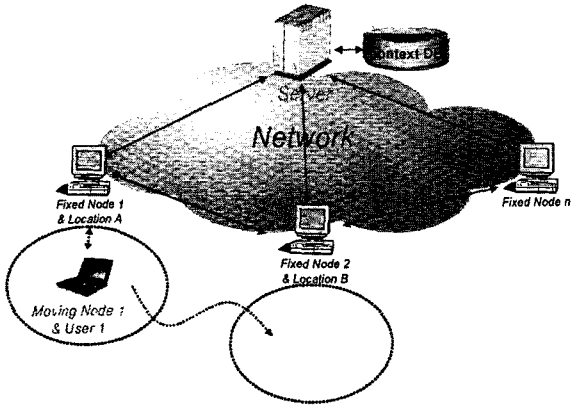
(그림 13) 컨텍스트 스크립트 언어의 전체적인 처리 과정

4. 응용 시스템 및 테스트

4.1 응용 시스템

본 논문에서 제안한 컨텍스트 스크립트 언어와 언어 처리기의 유용성을 보이기 위해 주어진 상황에 적합한 음악 재생 서비스를 제공하는 응용 시스템을 구축한다. 시스템은 크게 컨텍스트 서버(context server), 고정노드(fixed node), 이동노드(moving node)의 3개의 컴포넌트로 구성되어 있다. (그림 14)는 상황인식을 위한 시스템의 환경을 도식화 한 것이다. 먼저, 컨텍스트 서버는 원격 객체 정보와 컨텍스트 정보를 각각 객체 데이터베이스 및 컨텍스트 데이터베이스에 저장하고 검색하는 역할을 담당한다. 둘째, 고정노드는 미들웨어 역할을 담당하며, 블루투스 장치를 사용하는 환경에서 상황인식 처리를 위한 원격 객체 발견, 등록, 실행을 수행하는 모듈과 상황정의 스크립트 및 처리기로 구성되어 있다. 마지막으로, 이동노드는 원격 객체(컨텍스트 객체)로서 고정노드와 블루투스 통신을 통하여 데이터를 교환하며, 미리 정의된 상황에 따라 미들웨어와 상호작용을 통해 내장된 프로그램을 실행하거나 지정된 처리를 수행한다. 한편 서버와 고정노드 사이는 TCP/IP를 통한 유선 네트워크로 구성되어 있고, 고정노드와 이동노드 사이에는 무선 통신인 블루투스를 이용하여 데이터를 교환한다. 응용 시스템에서 사용자는 하나의 이동노드를 소유하고 있으며, 사용자의 위치에 따라 고정노드는 사용자의 접근을 감지해 사용자에 대해 미리 정의해 둔 음악을 연주하는 역할을 수행한다. 이 때, 음악은 사용자별로 서로 다르게 정의되어 있으며 같은 사용자에 대해서도 오전, 오후, 야간과 같은 시간대에 따라 다른 곡이 정의되어 있다. 따라서 응용 시스템은 각 사용자에 따라 정의된 음악을 연주함에 있어서 사용자 1이 장소 A에서 장소 B로 이동한다고 가정할 때, 장소 A에서 사용자 1를 위한 음악이 연주되고 있다면, 사용자 1이 장소 B로 옮겨감에 따라 장소 A의 음악이 정지되고 장소 B에서 사용자 1을 위한 음악이 연주된다. 한편, 고정노드는 사용자를 구분하여 사용자에 따른 미리 정의된 음악을 연주하며, 사용자가 고정노드 영역에 들어온 시각을 파악하여 시간대에 따라 정의된 음악을 연주한다.

한편 음악 연주 응용 시스템을 위한 컨텍스트 서버의 사용자 데이터베이스 스키마와 튜플 예들은 <표 2>와 같다. 각 필드는 User_ID, User_Name, Location, Music_M, Music_A, Music_E 로 구성된다. 여기서, User_ID는 각 사용자의 식별자로서 각 사용자를 구분하기 위한 주 키(primary key) 역할을 담당하며, User_Name은 각 사용자 이름을 나타낸다. Location은 각 사용자의 위치로서 미리 지정된 값이 아니며, 고정노드가 이동노드의 위치를 파악했을 때 사용자의 위치에 따라 빈번히 갱신되는 값이다. Music_M, Music_A, Music_E는 각각 사용자가 선호하는 음악을 위한 파일명으로서 오전(morning), 오후(afternoon), 저녁(evening)에 따른 음악 파일명에 대응된다. 컨텍스트 서버는 이와 같은 데이터베이스 스키마에 따라 사용자에 대한 음악 파일들을 유지하고 있으며, 미들웨어의 질의를 처리하고, 미들웨어의 파일 요청에 따라 파일들을 미들웨어로 전송한다. 즉, 이동노드를 소유한 사용자가 고정노드에 접근했을 때 이동노드는 주위의 가까운 고정노드를 탐색하고, 고정노드에 접속한 후 고정노드는 컨텍스트 서버에 등록된 사용자인지 질의를 수행한다. 고정노드는 사용자 인증절차를 거친 후 사용자에 대해 미리 정의된 음악을 컨텍스트 서버로부터 요청하여, 다운로드 받아 음악 파일을 수행한다. 블루투스 장치는 블루투스 Version1.1/Class1의 스펙(specification)을 만족하며 USB 인터페이스를 통하여 PC에 접속된다. 전체적인 응용 시스템의 환경은 시스템 클럭 866Mhz 펜티엄-Ⅲ 시스템에 64MB의 메모리를 탑재하였다. 운영체제는 Redhat Linux 7.3(kernel verion 2.4.20)이고, 블루투스 통신을 위한 장치 드라이버 및 프로토콜 스택은 affix 2.0.2, 컴파일러로서 GCC 2.95.4를 사용하였다. 아울러 음악 재생 서비스를 위해 앞에서 언급한 컨텍스트 객체 정의 및 컨텍스트 인스턴스의 삽입, 컨텍스트 인스턴스의 시작 등 컨텍스트 스크립트 언어가 제공하는 모든 명령어를 이용하여 응응 서비스를 구동하도록 하였다.



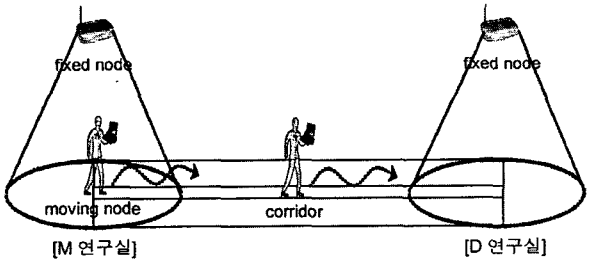
(그림 14) 응용 시스템의 전체 구조

〈표 2〉 컨텍스트 서버의 데이터베이스 스키마와 튜플의 예

User_ID	User_Name	Location	Music_M	Music_A	Music_E
1	'김철수'	NULL	filename11	filename12	filename13
2	'이영희'	NULL	filename21	filename22	filename23

4.2 테스트

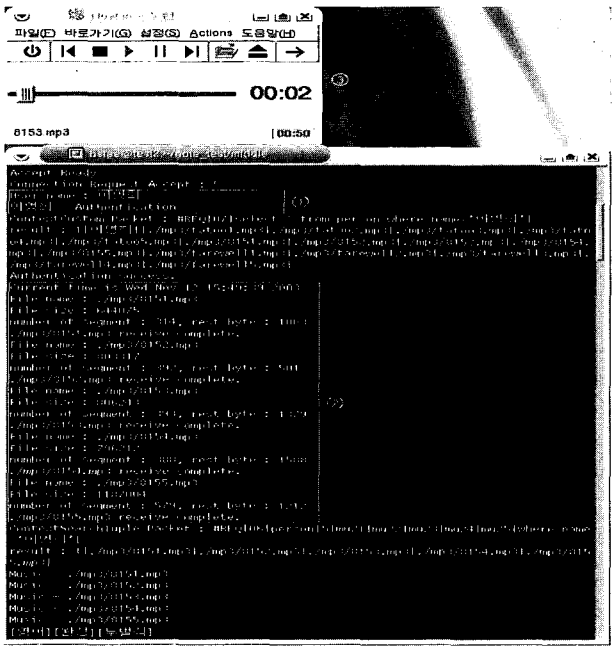
본 논문에서는 구현된 응용 시스템이 정상적으로 동작하는지 테스트한다. 테스트 항목은 이동노드를 소유한 사용자가 고정노드로 접근하거나 멀어질 때, 각각 다른 사용자가 고정노드에 접근할 때 미들웨어의 동작, 사용자가 고정노드에 접근할 때 시간대별 미들웨어의 수행동작 등 다양하다. 단, 본 논문에서는 이동노드를 소유한 사용자가 고정노드로 접근할 때와 다른 고정노드로 멀어져 갈 때에 대한 가장 일반적인 테스트를 수행한다. (그림 15)는 응용 시스템의 테스트 환경을 보여준다. 테스트를 위한 장소로는 대학원 내의 'D 연구실'과 'M 연구실'을 이용했으며, 각 장소에는 블루투스 이동노드를 감지할 수 있는 고정노드가 설치되어 있다. 사용자는 이동노드를 가지고 'D 연구실'에서 'M 연구실'로 이동하거나 'M 연구실'에서 'D 연구실'로 이동을 하며, 각 상황에 대해서 정상적으로 동작하는지 테스트한다.



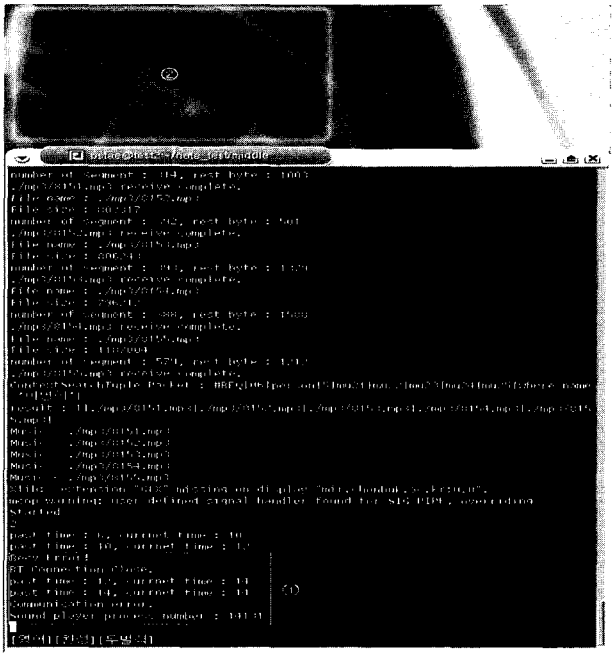
(그림 15) 응용 시스템을 위한 테스팅 환경

• 사용자의 고정노드로의 접근 시에 대한 테스트

이동노드를 소유한 사용자가 고정노드로 접근했을 때의 동작 화면은 (그림 16)(a)와 같다. (그림 16)(a)에서 ①은 사용자 접근 감지와 함께 사용자의 이름을 이동노드로 부터 획득한 모습이고, 아울러 인증절차를 수행하는 것을 확인할 수 있다. 인증절차가 끝나면 사용자에 따라 정의된 음악을 검색하고, ②의 모습과 같이 시각에 따른 음악을 서버로부터 다운로드 받는다. 다운로드가 완료되는 시점에서 ③과 같은 사운드 플레이어가 수행되어 음악이 연주되는 것을 확인할 수 있다.



(a) 접근 시 동작화면



(b) 멀어져 갈때의 동작화면

◉ **고정노드로부터 사용자가 멀어져갈 때에 대한 테스트**

(그림 16)(b)은 사용자가 고정노드로부터 멀어져 갈 때, 미들웨어가 수행하는 동작 화면을 나타낸다. 사용자가 고정노드로 부터 멀어져서 더 이상 블루투스와 무선 통신이 블가능하다고 미들웨어가 감지하면, (그림 16)(b)의 ①과 같이 에러 메시지를 출력하게 되며, 현재 음악을 연주하고 있는 사운드 플레이어의 프로세스 번호를 탐지하여 프로세스를 종료시킨다. (그림 16)(b)의 ②는 사운드 플레이어의 프로세스가 종료되어 사라진 모습을 나타내고 있다. 이와 같이 이동노드의 고정노드로의 접근 및 멀어지는 상황인식 처리 기술에 대해, 'M 연구실'와 'D 연구실'를 기반으로 블루투스 장치의 전파가 도달하는 범위가 서로 인접해 있을 때, 사용자의 장소 이동에 따라서 (그림 16)(a)와 같이 한 쪽 장소('M 연구실')에서는 음악이 재생되고, 다른 한 쪽 장소('D 연구실')에서는 (그림 16)(b)와 같이 음악이 정지됨을 확인할 수 있다.

5. 결론 및 향후 연구

본 논문에서는 유비쿼터스 컴퓨팅을 위한 상황인식 기반의 컨텍스트 정의 스크립트 언어 및 언어 처리기를 설계 및 구현하였다. 제안하는 컨텍스트 정의 스크립트 언어는 일련의 복잡한 과정을 간략하고 명료하게 기술하고 주어진 응용에 따른 상황에 대한 정의를 미리 규격화된 구문으로써 편리하고 쉽게 표현할 수 있음은 물론 특정 시스템에 의존적이지 않으며 범용적으로 기술할 수 있는 장점을 가진다. 반면에 특정 시스템에 의존하지 않기 때문에 최적화가 쉽지 않아 수행 속도가 떨어질 수 있으며 모든 응용을 표현하는데 한계가 있을 수 있다. 컨텍스트 정의 스크립트 언어의 구성 요소는 전체적으로 컨텍스트 객체의 정의와 컨텍스트의 정의 및 삭제 그리고, 컨텍스트가 구체화된 인스턴스의 삽입 및 삭제, 인스턴스의 시작 및 정지로 구성되어 있다. 아울러 제안하는 스크립트 언어의 유용성을 테스트하기 위해 블루투스 무선 통신 기술을 이용하여 이동성을 지닌 이동 노드를 발견, 등록 및 실행하는 상황인식 기반의 미들웨어와 주어진 이동 노드의 정보와 컨텍스트 정보를 효율적으로 데이터베이스 서버에 저장 및 관리할 수 있는 컨텍스트 서버로 구성된 상황인식 시스템을 구성하고 이를 토대로 컨텍스트 정보에 근거한 음악 재생 서비스를 제공하는 응용 시스템을 구축하였다. 앞으로 향후 연구로는 상황인식을 위한 다양한 응용 서비스에 제안하는 컨텍스트 정의 스크립트 언어를 적용함으로써 유용성을 테스트하는 것이다.

Appendix

A Grammar Specification(BNF) of the Context Script Language.

```
<context> := contextobject <context_name> "(" <data_type> <atom>
   { "," <data_type> <atom> } ")" " ; "
| contextdefine <context_name> condition "(" <condtion_exp1> {
   <lo_op> <condition_exp1> } { during <time_value> } ")" action
   <action>  <command_time2> param <param1>
| contextdelete <delete_exp>
| contextinsert <atom> into <context_name> codition <condtion_exp2>
   param <param2>
| contextstart <instance_name> in <context_name>
| contextstop <instance_name> in <context_name>

<context_name> := <variable> | <atom>
<instance_name> := <atom> { "," <atom> } | <instance_all>
<instance_all> := "*"
<condtion_exp1> := "(" <atom> "." <atom> <re_op> <atom> "."
   <atom> { <during> <time_value> } ")"
<condtion_exp2> := "(" "'" <atom> "'" { "," "'" <atom> "'" } ")"
<action> :=  <variable>
<param1> := "(" <atom> { "," <atom> } ")"
<param2> := "(" "'" <atom> "'" { "," "'" <atom> "'" } ")"
<command_time1> = "before" | "when" | "after"
<command_time2> = <command_time1> <time_value> | "from"
   <time_value> to <time_value>
<delete_exp> := <context_name> | <atom> "in " <context_name>
<re_op> := "=" | "<>" | "<" | ">" | "<=" | "=>"
<lo_op> := "and" | "or" | "not" | "AND" | "OR" | "NOT"
<variable> := [A-Z][a-zA-Z0-9]*
<atom> := [a-z][a-zA-Z0-9]*
<number> := [1-9][0-9]*
<time_value> := { <number> "h" } { <number> "m" } { <number> "s" }
<datatype> := "string" | "int" | "float" | "long" | "double" | "time" |
   "date"
```

참 고 문 헌

[1] M. Weiser, "Some Computer Science Issues in Ubiquitous Computing," Commnunications of the ACM, Vol.36, No.7, pp.75-84, 1993.

[2] 김완석 외7명, "유비쿼터스 컴퓨팅 기술과 인프라 그리고 전망", 한국정보처리학회 유비쿼터스 컴퓨팅 특집, 제10권 제4호, 2003.

[3] G. Banavar, A. Bernstein, "Issues and challenges in ubiquitous computing : Software infrastructure and design challenges for ubiquitous computing applications," Communication of ACM, 2002.

[4] C. D. Kidd, R. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. Mynatt, T. E. Starner and W. Newstetter, "The Aware Home : A Living Laboratory for Ubiquitous Computing Research," Proc. of the 2nd Int'l. Workshop on Cooperative Buildings, 1999.

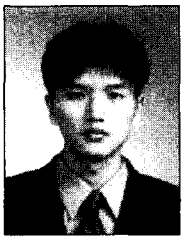
[5] 윤희용, "유비쿼터스 컴퓨팅 미들웨어 기술", 대한전자공학회지, 제30권 제11호, 2003.

[6] Stephen S. Yau, Fariaz Karim, "Context-Sensitive Middleware for Real-Time Software in Ubiquitous Computing Environments," Fourth International Symposium on Object-Oriented Real-Time Distributed Computing, pp.163-170, 2001.

[7] P. Couderc, A. M. Kermarrec, "Improving Level of Service for Mobile Users Using Context-Awareness," 18th IEEE Symposium on Reliable Distributed Systems, pp.24-33, 1999.
[8] A. Harter, A. Hopper, P. Steggles, A. Ward, P. Webster, "The anatomy of a Context-aware application," Wireless Networks Vol.8, Issue 2/3, pp.187-197, 2002.
[9] K. Cheverst, N. Davies, K. Mitchell, A. Friday, "Experiences of developing and deploying a context-aware tourist guide : the GUIDE project," Proceedings of the sixth annual international conference on Mobile computing and networking, pp.20-31, 2000.
[10] "Bluetooth Version 1.1 Profile," http://www.bluetooth. com.
[11] J. Bray, C. F. Sturman, "Bluetooth : Connect Without Cables," 홍릉과학출판사, 2001.
[12] "Affix : Bluetooth Protocol Stack for Linux," http://affix. sourceforge.net.

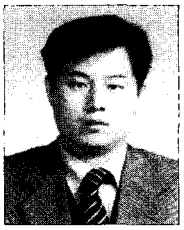
심 춘 보

e-mail : cbsim@cup.ac.kr
1996년 전북대학교 컴퓨터공학과(공학사)
1998년 전북대학교 컴퓨터공학과(공학석사)
2000년 전북대학교 컴퓨터공학과(공학박사)
2004년~현재 부산가톨릭대학교 컴퓨터정보공학부 전임강사
관심분야 : 멀티미디어 DB, 멀티미디어 IR, 유비쿼터스 컴퓨팅 등



김 용 기

e-mail : ykkim@dblab.chonbuk.ac.kr
2002년 전북대학교 컴퓨터공학과(공학사)
2003년~현재 전북대학교 컴퓨터공학과 석사과정
관심분야 : 유비쿼터스 컴퓨팅, 데이타베이스, LBS 등



장 재 우

e-mail : jwchang@dblab.chonbuk.ac.kr
1984년 서울대학교 전자계산기공학과(공학사)
1986년 한국과학기술원 전산학과(공학석사)
1991년 한국과학기술원 전산학과(공학박사)
1991년~현재 전북대학교 컴퓨터공학과 교수
관심분야 : 데이타베이스, 정보검색, 유비쿼터스 컴퓨팅, 상황인식 등

김 정 기

e-mail : jkk@etri.re.kr
1992년 전북대학교 컴퓨터공학과(공학사)
1994년 전북대학교 컴퓨터공학과(공학석사)
1999년 전북대학교 컴퓨터공학과(공학박사)
1996년~현재 한국전자통신연구원 임베디드S/W 연구단 선임연구원
관심분야 : 임베디드 S/W, 유비쿼터스 컴퓨팅, 병렬 정보검색 등