Journal of the Korea Academia-Industrial Cooperation Society
Vol. 12, No. 12 pp. 5885-5894, 2011

http://dx.doi.org/10.5762/KAIS.2011.12.12.5885

유비쿼터스 홈 기반의 신뢰성 제공을 위한 오류 제어 시뮬레이터 개발

이시흥[1], 김홍윤[2*]
[1]한국인터넷진흥원 공공정보보호단, [2]한서대학교 컴퓨터공학과

Reliability of the Ubiquitous Home-Based Simulator Developed for the Eerror Control

Si-Heung Lee[1] and Hong-Yoon Kim[2*]
[1]**Public Security Division, Korea Internet & Security Agency**
[2]**Division of Computer Engineering, Hanseo University**

요 약 유비쿼터스 환경이 가장 먼저 실현될 것으로 예상되는 유비쿼터스 홈에 대한 연구가 지금까지 이루어져 왔다. 하지만, 유비쿼터스 홈의 복잡성으로 인하여 신뢰성의 오류 처리가 어렵다는 것이 현실이다. 본 논문에서는 이런 오류들의 검출 및 복구방법에 대하여 유비쿼터스 홈 오류 연구를 위한 시뮬레이터를 개발하여 다양한 실험을 통해 오류의 규칙과 진단 지식의 정확성에 대하여 알아보았고 그 결과를 토대로 향후 연구 과제에 대하여 논하였다.

Abstract Ubiquitous environment is expected to be result of researches and studies in ubiquitous home base on. However, because of the complexity of the ubiquitous home base on, the realization of error handling seems impossible. In this paper, we present how to detect and recover from these errors for the Ubiquitous Home utilizing a fault simulator developed for the study and through a variety of experiments, results were discussed and analyzed. Base on the said results, future works are also discussed.

Key Words : Ubiquitous Home, Simulator, Error Control, Error Detection

1. 서론

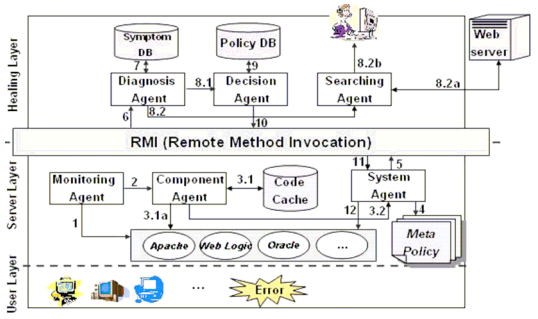
유비쿼터스 컴퓨팅이란 인간의 주변에 내장된 컴퓨터가 편리를 위해 스스로 동작하는 컴퓨팅 환경을 구축하는 것이다. 이 개념은 다양한 분야에 적용 가능하지만, 가정 내에서의 편리를 위한 유비쿼터스 환경(유비쿼터스 홈)이 제일 먼저 실현될 것[1]으로 예상됨에 따라, 유비쿼터스 홈에 대한 연구[2,3,4,5]가 활발히 진행되고 있다. 이러한 환경을 위한 유비쿼터스 홈은 다양한 기기들과 소프트웨어로 구성되기 때문에 유비쿼터스 홈 상에서 높은 수준의 신뢰성을 제공하기 쉽지 않다. 신뢰도를 향상시키기 위하여 유비쿼터스 홈 환경에서 발생하는 오류를 예측하여 미연에 방지하거나, 복구가 가능한 오류는 유비쿼터스 홈 시스템이 스스로 복구하는 기능이 필요하다. 하지만, 유비쿼터스 홈의 구성요소에서 발생하는 오류의 검출 및 복구에 관한 연구는 거의 없는 실정이다.

2. 관련연구

본 논문에서는 규칙을 생성하고, 생성된 규칙을 기반으로 오류 진단 지식을 얻기 위해, 유비쿼터스 홈에서 생성되는 로그 데이터를 수집하여 현재 상황을 분석하고 오류의 발생 여부를 파악한다. 만일, 오류를 발견 했을 경우 오류 발생과 관련 있는 로그 데이터를 분석하여, 규칙에 기반해 오류의 원인을 찾아 오류 진단 지식을 생성하는 방법과, 댁내에 설치된 가전기기들의 상태 정보를 알려주는 로그 데이터를 활용하여, 실시간으로 규칙을 생성

*교신저자 : 김홍윤(hykim@hanseo.ac.kr)
접수일 11년 09월 09일 수정일 11년 11월 06일 게재확정일 11년 12월 13일

하는 방법을 제안한다. 로그 데이터는 유비쿼터스 홈 구성요소의 상태가 변경될 때 발생하므로 유비쿼터스 홈에서 발생한 하나의 사건으로 간주될 수 있으며, 베이지안 네트워크[6] 혹은 은닉 마코프 모델(Hidden Markov Model)[7] 등 조건부 확률에 기반한 방법을 통해 두 사건 간의 규칙을 분석할 수 있다. 규칙 생성 방법을 연구하기 위해 사용되는 로그 데이터는 실제 유비쿼터스 홈 환경에서 발생하는 데이터를 이용하여 생성되는 것이 바람직하나, 유비쿼터스 홈의 물리적인 환경 구축이 선행되어야 하기 때문에 다양한 유비쿼터스 홈 환경이 반영된 다량의 데이터를 짧은 시간 안에서 얻기 어렵다. 다양한 환경이 반영된 다량의 데이터를 얻기 위해 컴퓨터 과학 분야에서는 시뮬레이터를 구축하는 방법이 일반적으로 사용되고 있다. 본 연구에서도 유비쿼터스 홈 구성요소를 다양하게 변경할 수 있고, 유비쿼터스 홈 서비스 동작을 모사하기 위한 이산 사건 시스템 기반의 시뮬레이터를 설계 및 구현해 생성된 규칙의 정확성과 오류 진단 지식 생성에 대한 실험 및 분석하였다. 전통적으로 컴퓨터 분야에서 오류와 관련해 매우 다양한 연구가 진행 되어 왔지만[8,9], 유비쿼터스 홈의 경우 그 시스템이 정립되어 가는 과정에 있어 오류 관련 연구는 아직 초기 단계에 있다. 현재까지 유비쿼터스 홈에서 활용 가능한 오류 관련 연구는 소프트웨어 오류를 대상으로 한 Proactive Self-Healing System(이하. 능동적 자기 치유 시스템)[10], 대규모 컴퓨터 시스템에서의 소프트웨어, 하드웨어의 오류를 대상으로 하는 Autonomic System[11], 그리고, 시뮬레이션 환경을 구축하고 사용자가 입력한 서비스를 기반으로 장치간의 규칙을 생성하고, 생성된 규칙으로부터 충돌을 탐지하는 CASS(Context Aware Simulation System for Smart Home) [12] 등이 있다.

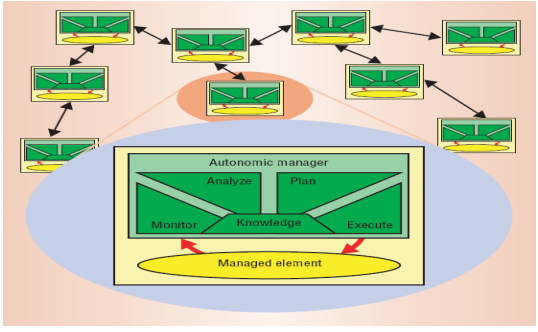


[그림 1] 능동적 자기 치유 시스템
[Fig. 1] Proactive Self-Healing System

Proactive Self-Healing System은 그림 1과 같이 모니터링 에이전트(Monitoring Agent), 컴포넌트 에이전트(Component Agent), 시스템 에이전트(System Agent), 진단 에이전트(Diagnosis Agent), 결정 에이전트(Decision Agent) 등으로 구성되어 각 에이전트들이 협력하여 오류 처리를 수행한다.

Proactive Self-Healing System은 다수의 에이전트로 구현되어 기능의 추가 혹은 변경이 유연해 구조상 유리하다는 장점이 있으며, 유비쿼터스 홈 미들웨어를 컴포넌트 에이전트에 등록하는 것으로 운영체제 및 소프트웨어에서 발생하는 오류 처리가 가능하지만, 장치의 물리적인 오류나 장치간의 연관관계에서 발생하는 오류를 식별하지 못하고, 증상 DB, 정책 DB, 결정 테이블에 정의된 형태의 오류만 처리 가능하다는 문제점이 있다.

Autonomic System은 다수의 컴퓨터가 네트워크를 통해 복합적으로 연결된 환경에서 새로 추가된 장비에 대한 자가 구성, 시스템 최적화, 그리고 오류 처리를 지원하는 기업 환경을 위한 컴퓨팅 시스템이다. Autonomic System의 오류 관련



[그림 2] 오토노믹 시스템의 오토노믹 관리자
[Fig. 2] Automatic system, Automatic manager

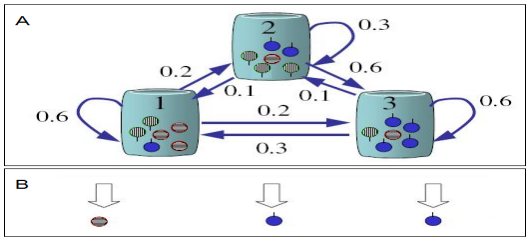
구성 요소로는 Automatic Manager와 Policy Manage가 있다. Autonomic System의 Autonomi manager는 그림 2와 같이 Monitor 단계에서 시스템의 각 노드들에 대한 지속적인 감시를 통해 노드들의 소프트웨어 및 하드웨어 오류를 탐지한다. Autonomic System은 소프트웨어적인 문제뿐만 아니라, 시스템 구성요소의 물리적인 오류 등에도 대처할 수 있지만 역시 구성요소 간의 규칙으로 발생하는 문제는 해결하지는 못한다는 단점이 있다.

CASS(Context Awareness Simulation System)는 일련의 조건과 결과로 이루어진 규칙을 기반으로 동작하며 센서가 생성한 컨텍스트와 규칙에 포함되어있는 조건을 비교해 규칙 간의 충돌을 검사할 수 있는 시뮬레이터이다.

현재까지 발생하는 오류는 기기 혹은 소프트웨어 기능적인 오류를 그 대상으로 하거나, 정상적인 상황에서 기

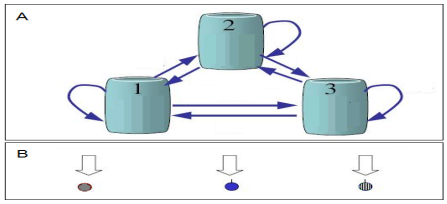
기들을 사용하는 데 있어 상충되는 명령이 전달되는 상황을 해결하고자 하였다. 하지만, 기기, 소프트웨어가 모두 정상이고, 상충된 명령 또한 전달되지 않지만 오류가 발생하는 경우가 존재한다. 물론 CASS가 서비스를 통해 장치간의 규칙을 일부 파악하고 충돌 여부를 판단할 수 있지만, 유비쿼터스 홈 구성요소에 대한 모든 규칙을 사용자가 직접 입력하는 것은 유비쿼터스 환경에 위배된다. 가장 최근에 유비쿼터스 홈 구성요소 간의 규칙 기반 연구에서는 사용자로부터 직접 입력을 요구[13]하거나 전문가 집단에 의해 온톨로지 언어 등으로 미리 기술[14]되는 것이 일반적이며, 아직까지 알고리즘에 의해 유비쿼터스 홈 구성요소들 간의 규칙을 생성하는 연구는 전무하다. 따라서, 오류 생성 규칙을 생성하기 위해서는 유비쿼터스 홈의 다양한 구성요소의 모든 상태 조합을 검토해야 하므로 빠른 시간 안에 각 구성요소의 규칙을 분석하기 어렵다. 확률에 의한 추론 기법인 은닉 마코프 모델(Hidden Markov Model)[7]과 베이지안 네트워크[6]를 소개하고, 이들을 사용해 규칙을 생성하는 방법에 대해 설명한다.

먼저, 은닉 마코프 모델의 이해를 위해서 그 기반이 되는 마코프 모델에 대해 알아볼 필요가 있다. 마코프 모델은 연속적으로 발생하는 사건이 그 연속성 상에서 규칙을 가지는 경우 적용될 수 있는 확률 모델로 마코프 모델은 각 사건들이 발생할 확률이 상태에 따라 다르다고 가정한다. 아래 그림 3은 마코프 모델의 한 예로서 사건은 각각 빨간 공(), 파란 공(), 녹색 공()에 대한 선택(B)이고, 상태가 변경(A)될때 마다 한번씩 발생하며, 상태 1, 2, 3에 따라 다른 확률을 가진다. 상태 1에서 빨간 공을 선택할 확률은 3/6 이며, 파란 공을 선택할 확률은 1/6이다. 파란색 화살표는 상태간 전이 확률이며 사건이 발생하면 반드시 상태가 변경된다. 마코프 모델의 목적은 상태에 따라 사건이 발생할 확률과 상태 간의 전이 확률이 사전에 정의되었을 때, 특정 사건이 발생한 상황에서 다른 사건의 발생을 사전에 예측하는 것이다.



[그림 3] 마코프 모델의 예
[Fig. 3] Example of Markov Model

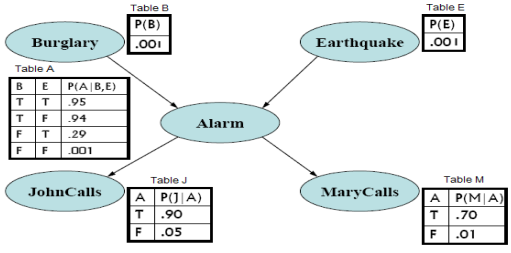
은닉 마코프 모델은 그림 4와 같이 사건들의 순서열(B)이 주어 졌을 때, 동일한 순서열을 생성하는 마코프 모델(A)를 구축한다. 다시 말해, 은닉 마코프 모델은 사건들의 순서열 만으로 마코프 모델은 상태의 개수, 각 상태간의 전이 확률, 그리고 각 상태에서 사건들의 발생 확률을 추론 하는 것이 목적이다.



[그림 4] 은닉 마코프 모델
[Fig. 4] Hidden Markov Model

은닉 마코프 모델이 사건들에 대한 정보를 올바르게 학습해 마코프 모델과 동일한 모델을 구축해 내면, 마코프 모델의 목적이 이미 알고 있는 상태에서 사건들에 대한 확률 정보를 이용하여 앞으로 일어날 사건을 예측하는 것이므로 앞으로 일어날 사건들에 대한 확률을 얻는 것 또한 가능하다.

베이지안 네트워크는 그림 5와 같이 사건들의 전이 확률을 표현하는 방향성 그래프로서 베이즈 정리에 기초해 어떤 사건이 발생했을 때, 그 원인에 대한 확률적 분석이 가능하다.



[그림 5] 베이지안 네트워크의 예
[Fig. 5] Example of a Bayesian network

베이지안 네트워크는 어떤 사건에 대한 원인 분석이나 의사 결정에 사용되기도 하지만, 어떤 사건들의 순서열이 주어졌을 때, 해당 네트워크를 통해 특정 사건이 생성될 확률을 계산하는데 사용될 수 있다. 즉, 은닉 마코프 모델과 마찬가지로 패턴 인식과 패턴 분류에 이용될 수 있다. 또한, 베이지안 네트워크의 각 정점과 호는 확률적 관계를 가지고 있으므로 높은 확률을 갖는 각 정점들 즉, 사건들 사이에는 어떠한 규칙을 가진다고 볼 수 있다. 이들

방법이 유비쿼터스 홈 환경에 활용되기 어려운 점은 두 사건이 실제로 규칙을 갖는다 하더라도 원인에 해당하는 사건이 발생한다고 해서 즉시 다음 사건이 발생하는 것이 아니라는 점이다.

3. 규칙 생성 방법

유비쿼터스 홈 상황에서 오류를 포함하고 있는지를 파악하기 위해 사용되는 규칙의 생성방법에 대하여 알아보자면, 오류 진단 지식은 유비쿼터스 홈에서 발생 가능한 오류 증상과 오류에 대한 원인을 나타내며, 유비쿼터스 홈 시스템에서 발생하는 로그 데이터를 분석하여 오류 진단 지식을 생성한다. 또한, 오류 진단 지식 생성 모듈은 오류 증상을 파악하기 위해 규칙을 이용하는데, 이 규칙은 유비쿼터스 홈이 오류 없이 동작할 경우 반드시 발생해야 하는 사건으로 정의한다. 규칙의 형식은

IF (Conditoon_Event) THEN (Result_Event)로 표현된다. Condition_Event와 Result_Event는 유비쿼터스 홈에서 발생하는 로그 데이터를 분석하여 생성된 사건을 나타내며, 사건의 발생 순서 및 인과 관계에 따라 Condition_Event와 Result_Event로 분류된다.

Condition_Event는 Result_Event보다 먼저 유비쿼터스 홈에서 발생한 사건으로서 Result_Event가 발생하기 위한 조건이 되고, Result_Event는 Condition_Event가 발생한 후 결과로서 발생해야 하는 사건이다. 만약 유비쿼터스 홈에서 Condition_Event가 발생 하였으나 Result_Event가 발생하지 않았거나, 혹은 Condition_Event가 발생 하지 않았으나 Result_Event가 발생 했다면 유비쿼터스 홈에 문제가 있음을 파악할 수 있다. 그리고, 오류 증상 파악방법에서 오류 학습 모듈은 Context에 드러나는 경우와 그렇지 않은 경우에 대하여 판단 한다. 즉, 전달된 context로부터 분석한 상황이 오류 탐지 규칙에 만족하면 정상인 사례로 판단하고 그렇지 않으면 오류사례로 판단한다. 오류 진단 지식은 많은 오류를 추정해 한 집합으로 모으고 오류의 원인으로 간주될 수 없는 context를 제거함으로써 생성된다. 오류 진단 규칙은 오류 증상 파악, 오류 원인 식별 그리고 오류 진단 규칙 생성의 과정을 통해 이루어진다. 오류 진단 규칙 생성 모듈에서는 규칙 생성을 위하여 유비쿼터스 홈에서 발생 하는 사건에 대해 조건부 확률을 적용하여 사건 간의 인과 관계를 파악한다. 발생된 사건 사이 조건부 확률을 적용하여 규칙을 생성하는 방법은 다음과 같다.

조건부 확률은 특정 사건 X1, X2에 대해 X2가 발생한 상황에서 X1이 발생할 확률을 의미하고 P(X1|X2) = P(X1∩X2) / P(X2) 으로 표기 된다. 단순히 조건부 확률만으로 두 사건 X1, X2 사이의 규칙을 알아내기는 어렵다. 조건부 독립은 특정 사건 X1, X2가 서로에 대해 어떠한 영향도 미치지 못하는 것을 의미하며, 오직 P(X1|X2) = P(X1) 이고, P(X2|X1) = P(X2) 일 때, 두 사건 X1, X2은 서로 조건부 독립이라 정의 된다.

유비쿼터스 홈에서 어떤 두 이벤트 E1과 E2가 순차적으로 발생 했을 때, 발생 순서에 따라 규칙 Rule1: E1 → E2 혹은 규칙 Rule2: E2 → E1이 생성 가능하다. 그러나 이러한 사건들 간에 인과 관계가 있는지의 여부는 알 수 없기 때문에 조건부 확률을 이용하여 파악한다. 다음에서 조건부 확률의 개념 및 이를 적용하여 규칙을 생성하는 방법에 대하여 설명한다.

・사건 E1과 E2 간에 규칙 분석:

만약 P(E2|E1) ≠ P(E2) 혹은 P(E1|E2) ≠ P(E1)이 발견되면, 사건 E1과 E2와는 서로 규칙이 있다. 그러나 일반적으로 두 확률 P(E1)과 P(E1|E2)의 값이 일치하지 않기 때문에 P(E1) = P(E1|E2)인지 여부를 결정하기 위한 임계치가 필요하다.

・사건 간 규칙 및 인과 관계를 결정하는 임계치 P value의 설정:

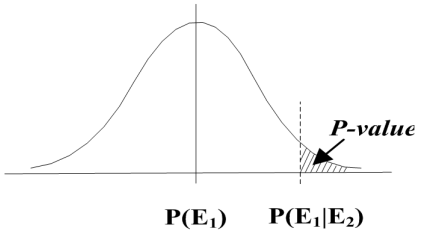
수식(1)은 P(E1) = P(E1|E2) 에 대한 임계치를 정의하는 P value를 결정하기 위한 식이다.

$$P-value = P\left(Z > \frac{|P(E_1 \mid E_2) - P(E_1)|}{\sqrt{\left(\frac{P(E_1)(1-P(E_1))}{n}\right)}} \right)$$

수식(1), where n=전체 관측 횟수

Equation (1), where n = whole number of observations

P value는 그림 6과 같이 평균이 P(E1), 표준 편차가 P(E1)(1 P(E1))인 표준 정규 분포를 가정하고, 가설 P(E1) = P(E1|E2)에 대한 단측 테스트(one tailed tests)에 사용된다. 일반적으로 P value가 0.1보다 작으면 가설은 기각되고, 두 값은 같지 않다고 본다.



[그림 6] P-value를 이용한 규칙 결정 방법
[Fig. 6] P-value determined using the rules, how

4. 오류 진단 지식 생성 방법

오류 진단 지식은 특정 오류 증상을 일으키는 원인들을 찾기 위해 필요하며, 특정 증상 Sy와 Sy의 원인 집합 Syr에 대해 “Syr 때문에 Sy가 발생한다.”와 같은 형식을 가진다. 오류 진단 지식 생성을 위해 규칙을 지속적으로 수집하고, 규칙이 위배될 때 오류가 발생했음을 인지한다. 즉, 오류 진단 지식의 증상 S는 규칙 Rule에 대해 ~ Rule이다. 규칙 Rule의 원인 C가 발생해도 제한 시간 Tr 내에 결과 R이 발생하지 않으면 오류가 발생한 상황으로 간주될 수 있다. 오류 진단 지식은 특정 오류 증상 S에 대해 오류의 원인으로 추정되는(Suspected) 컨텍스트들의 집합인 Sets와 어떠한 경우에도 오류의 원인이 될 수 없는(Not Suspected) 집합인 Setn을 유지함으로써 오류의 원인을 식별한다. 집합 Setn은 유비쿼터스 홈이 정상일 때의 컨텍스트들을 수집해 생성되며, 집합 Sets는 오류 상황에서 발생한 컨텍스트들을 수집해 생성된다. 집합 Sets의 컨텍스트들을 집합 Setn의 컨텍스트들을 이용해 제거되면, 원인 Syr이 식별될 수 있다.

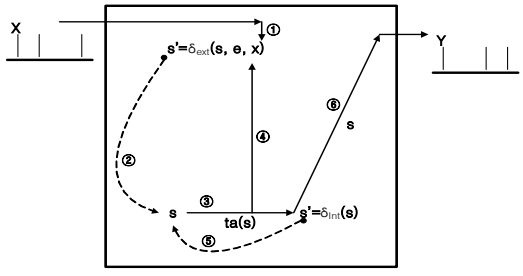
유비쿼터스 홈과 관련된 표준화를 비롯하여 관련 기술들의 상호호환성 등이 충분히 고려되지 않아 유비쿼터스 홈의 상용화 시 문제가 되고 있다. 특히 고려해야 할 많은 종류의 홈 네트워킹 방법과 대상 그리고 인식해야 할 상황이 매우 다양한 형태로 존재하기 때문에 새로운 기술이나 제품의 연구·개발, 기존에 개발된 제품들의 실제 댁내 환경 적용 또는 새로운 서비스 개발 및 시험을 하는데 있어 비효율성의 원인이 되고 있다. 따라서 실제 유비쿼터스 홈이나 홈 장치를 구현환경을 제공하는 것은 비용이나 시간 그리고 노력의 측면에서 효율적이며 필요한 일이라고 할 수 있다. 이를 위해 개발된 유비쿼터스 홈 시뮬레이터로는 SensorSim(A Simulation Framework for Sensor Networks)[15], SENS(Sensor, Environment and Network Simulator)[16], UbiWise[17], TATUS[18], CASS (Context Aware Simulation System)[12], 그리고 ubiHome 3D Simulator[19] 등이 있다.

5. 시뮬레이터 설계 및 구현

시뮬레이터는 유비쿼터스 홈 환경을 유연하게 변경할 수 있어야하고 쉽게 확장될 수 있어야 하나, 위의 예시된 시뮬레이터들은 확장성과 유연성이 부족하다는 문제점이 있다. 특히, 유연성은 모든 시뮬레이터들의 공통이다. 이 문제를 해결하고 신뢰성 있는 로그데이터의 출력을 위하여 시뮬레이터 구축방법이 필요하다. 이 구축방법으로는 DEVS(Discrete Event System Specification)라는 모델링 방법으로 유비쿼터스 홈과 같은 이산 사건 시스템의 동작과정을 추상화하여 수학적으로 기술하는 틀을 제공하고, 시스템을 계층적으로 모듈화하여 표현하는 수학적으로 검증된 모델링방법론이다. 유비쿼터스 홈 시뮬레이터의 설계에서 DEVS는 이산사건 시스템의 동작을 추상화하여 수학적으로 기술하는 틀(framework)을 제공하며, 시스템을 계층적으로 모듈화하여 표현한다. 이를 위하여 DEVS는 2개의 모델 클래스인 Atomic model과 Coupled model로 정의된다. Atomic model은 더 이상 분해될 수 없는 컴포넌트를 나타내며, Coupled model은 서브 모델들의 결합체로서 이들 서브 모델들의 결합 관계를 표현한다.

[표 1] 오토믹 모델
[Table 1] Atomic model

M = <X, Y, S, δ_{ext}, δ_{int}, ta, λ>, where X : 입력 사건 집합 Y : 출력 사건 집합 S : 모델의 상태 집합 δ_{ext} : Q x X→ S : 외부 상태 변경 함수, where Q = {(s, e) | s∈S, 0≤e≤ta(s)} δ_{int} : S → S : 내부 상태 변경 함수 ta : S → $R^{+}_{0,\infty}$: 시간 진행 함수 λ : S → Y : 출력 함수

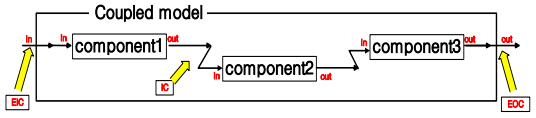


[그림 7] Atomic model의 도식화
[Fig. 7] The graphs of automatic model,

그림 7은 Atomic model에서의 상태 변경 과정을 도식화한 것으로 X는 입력 사건, Y는 출력 사건, s는 모델의 현재 상태, s’는 현재 상태 s에서 외부 변경 함수(δext) 혹은 내부 변경 함수(δint)에 의해 변경된 모델의 상태, ta(s)는 모델이 상태 s를 유지해야 하는 시간, 그리고 e는 상태 s를 유지한 시간을 나타낸다.

[표 2] 결합 모델
[Table 2] Coupled model

N = <X, Y, D, {M_i}, EIC, EOC, IC, Select>, where X : 입력 사건 집합 Y : 출력 사건 집합
N = <X, Y, D, {M_i}, EIC, EOC, IC, Select>, where X : 입력 사건 집합 Y : 출력 사건 집합 D : 구성모델의 이름 집합 {M_i}: N의 서브 모델 집합을 의미하며, i ∈ D에 대한 M_i는 Atomic 혹은 Coupled model EIC : 외부 입력 결합관계 EOC : 외부 출력 결합관계 IC : 내부 결합관계 Select : Coupled model을 구성하는 여러 개의 Atomic model 혹은 Coupled model인 DEVS model들이 같은 시간에 수행되려 할 때, 이들을 순서화 시키는 함수

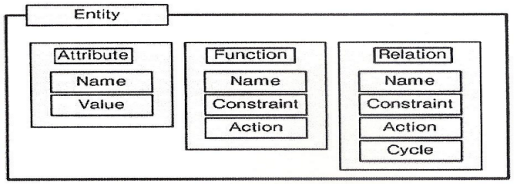


[그림 8] Coupled model의 도식화
[Fig. 8] The graphs of Coupled model

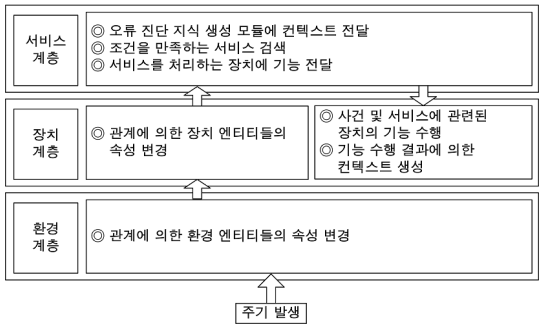
Coupled model은 서브 모델들이 어떻게 연결되어 있는가를 기술하는데 사용되고, Atomic model 간의 결합관계, Atomic model과 Coupled model의 결합관계, 혹은 Coupled model 간의 결합관계를 나타낸다. 만약 DEVS model인 여러 개의 Atomic model이 상위 계층인 Coupled model을 형성하려 한다면, Coupled model의 컴포넌트는 Atomic model로만 이루어진다. Atomic model과 Coupled model이 혼재하여 상위 계층인 Coupled model을 형성하려 한다면, 상위 계층의 Coupled model은 Atomic model과 Coupled model을 컴포넌트로 포함하게 된다. 이와 마찬가지로 상위 계층 Coupled model은 Coupled model만을 컴포넌트로 포함할 수 있다.

그림 8에서 component1, component2, component3은 표 2에서 언급한 D의 각 원소에 해당하는 구성모델의 이름으로서, Atomic model 혹은 Coupled model이다. 따라서 DEVS model은 Coupled model이 또 다른 Coupled model을 서브 모델로 포함하는 계층 구조를 이루게 된다. Coupled model의 component1, component2, component3 모두 component라는 이름의 Atomic model이라면, 표 2의 {Mi} = {component, component, component}가 된다. M1=component, M2=component, M3=component로 표현할 수 있다.

DEVS 기반으로 모델링하기 전에 유비쿼터스 홈 구성요소들을 모델링해야 한다. 제안된 시뮬레이터는 유비쿼터스 홈의 각 계층에 속하는 구성요소들을 속성, 기능 그리고 관계로 표현되는 엔티티로 추상화하고, 시뮬레이션을 통해 오류 진단 지식 생성에 필요한 로그 데이터를 생성한다. 시뮬레이터는 유비쿼터스 홈의 댁내 환경 정보들을 어떠한 값으로 표현할 수 있어야 하며, 유비쿼터스 홈을 구성하는 환경과 장치 모두를 엔티티로 정의한다. 엔티티는 속성으로 표현되며, 엔티티사이에 정보를 전달하기 위해 기능과 관계를 가지고 있다. 엔티티는 그림 9와 같은 구조를 가지며, 속성(Attribute)으로 자신이 가진 정보를 표현하고, 기능(Function)과 관계(Relation)를 이용해 자신의 속성이 어떻게 변경되는 지를 기술한다. 속성은 이름과 값으로 구성되어 엔티티가 가질 수 있는 정보를 표현한다. 기능을 통해 시뮬레이터는 사용자 혹은 서비스에 의한 조작을 시뮬레이션 하며, 이름, 제약 조건, 행동으로 구성된다. 제약 조건은 행동이 수행되기 위한 조건들이며, 행동은 모든 제약조건이 만족될 때 어떻게 속성이 변경되는 지를 기술한다. 관계는 수행 주식(Cycle)가 포함된 기능으로서 시간에 따른 변화 등 특정 사건 발생 여부와 관계없이 수행 주기마다 제약 조건을 검사해 행동 수행 여부를 결정한다. 매 주기마다 시뮬레이터는 그림 10과 같은 동작을 수행하며, 수행 중에 발생하는 Context[20]로부터 로그 데이터를 생성한다

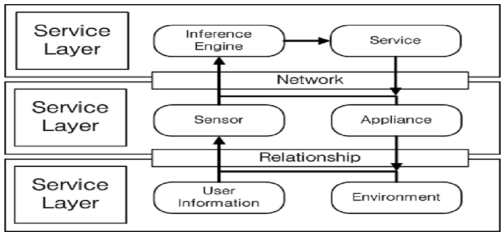


[그림 9] 엔티티 구성요소
[Fig. 9] the entity component



[그림 10] 시뮬레이터 동작 과정
[Fig. 10] simulator course of action

설계 및 구현된 시뮬레이터는 유비쿼터스 홈 상에서 오류 관련 연구를 수행할 수 있도록 오류 생성 정책을 제공한다. 우선, 댁내 상황을 인지하고, 사용자에게 서비스를 제공하는 관점에서 유비쿼터스 홈을 도식화 하면 그림 11과 같이 세 가지의 서로 다른 계층으로 구성될 수 있다.



[그림 11] 유비쿼터스 홈 계층
[Fig. 11] Ubiquitous Home hierarchy

환경 계층은 온도, 습도, 사용자의 위치 등 댁내의 물리적인 환경을 나타낸다. 이러한 댁내 환경 정보는 장치 계층의 센서를 통해 수치 데이터로 변경되며, 서비스 계층에서 서비스를 제공하기 위해 필요한 정보로서 활용된다. 장치 계층은 센서(Sensor), 가전기기(Appliance)등으로 구성된다. 센서는 유비쿼터스 홈 시스템이 주변 환경의 변화를 인지하고자 할 때 사용되며, 가전기기는 유비쿼터스 홈이 사용자에게 편의를 제공하기 위해 사용된다. 마지막으로 서비스 계층의 구성요소들은 장치 계층으로부터 전달 받은 데이터를 분석해 현재 상황을 파악하기 위한 추론 엔진(Inference Engine)과 상황에 따라 가전기기를 동작시키는 서비스로 구성된다.

또한, 유비쿼터스 홈 상에서의 오류를 네트워크 환경에서 정보 전달이 이루어지는 관점에서 본다면 아래와 같이 모두 네 종류의 오류가 발생할 수 있다.

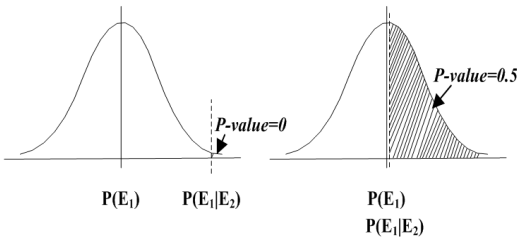
- 환경 계층에서 장치 계층으로 정보가 전달되는 과정이 이루어지지 않는 오류.
- 장치 계층이 컨텍스트를 서버계층에 전달하지 못하는 오류.
- 장치 계층에서 컨텍스트를 전달했음에도 불구하고 서버가 이를 인식하지 못하는 오류.
- 서비스 추론 혹은 오류 추론에 실패하는 오류.

위의 오류 중에 첫 번째와 두 번째 오류는 장치의 특정 기능에 고장이 발생했을 때 나타난다. 세 번째 오류는 네트워크상의 오류이거나 서버에 고장이 발생한 경우이다. 네 번째는 서버에서 구동되는 소프트웨어들에 문제가 발생한 경우이다. 즉, 오류를 발생하는 주체는 장치의 기능, 네트워크, 서버이다. 하지만 서버들 중 가장 핵심적인 두 부분은 시뮬레이터 내부에 구현되지 않으므로 서버에 대한 오류는 고려하지 않는다. 기능 오류는 장치의 특정 기능이 동작하지 않는 것이다. 기능에 오류가 발생하면 해당 장치는 오류가 발생한 시점부터 복구되기 전까지 해당 기능은 어떠한 작업도 수행하지 않는다. 또한 관련 로그 데이터 역시 생성되지 않는다. 네트워크 오류는 장치 계층과 서버계층 사이에 정보가 정상적으로 전달되지 못해 서비스 제공이 이루어지지 않는 것이며 연결 오류, 전송 실패, 전송 지연이 있다.

오류학습 시뮬레이터를 실행하면 Device, Environment, Service 입력파일로부터 시뮬레이션 하기 위한 환경을 생성한고 시뮬레이터는 배치파일의 명령에 따라 실행된다. 사용자는 시뮬레이터를 실행한 다음 배치파일을 선택하여 시뮬레이션을 시작한다. 시뮬레이터가 이미 검증된 DEVS를 기반으로 설계 및 구현되었기 때문에, 입력한 배치 명령과 일치하는 Context를 얻을 수 있었다. 이는 Context들로부터 생성되는 장치 상태 로그 데이터 역시 정확하리라는 사실을 뒷받침 해 준다고 볼 수 있다.

6. 실험 및 분석

시뮬레이터에 의해 생성된 로그 데이터를 입력으로 제안된 규칙 생성 방법에 대한 테스트를 수행하고, 테스트에 사용된 로그 데이터와 성능 지표에 대해 설명한다. 끝으로, 테스트에 사용된 시나리오와 테스트 결과에 대해 분석한다. 비교 대상이 확률에 기반하여 두 사건 간의 규칙을 분석하기 때문에 생성되는 규칙의 수는 허용되는 확률에 비례한다. 만일 허용되는 확률이 0이면 모든 사건들에 대해 규칙이 생성되며, 1일 경우 규칙은 생성되지 않는다. 제안하는 방법에서는 p value를 이용한 단측 테스트(One tailed test)를 수행하므로 규칙에 대한 확률 범위는 [0, 0.5] 이고, 허용 확률은 $1-(p \bullet value)\times 2$ 로 계산된다. 만일, 허용 확률을 0으로 설정하고자 할 때는 그림 12 좌와 같이 p value를 0.5로 설정 하고, 허용 확률을 1로 설정하고자 할 때는 그림 12 우와 같이 p value를 0으로 설정한다.



[그림 12] 허용 확률에 대한 p-value
[Fig. 12] allowing for the probability p-value

규칙의 질(quality)은 생성된 규칙이 가지는 확률의 평균과 표준편차를 이용해 얻을 수 있고, 만일 평균이 낮거나 표준 편차가 크면 허용 확률을 높일 때 생성된 규칙의 수가 급격히 줄어들 수 있다. 각 규칙 생성 방법의 규칙 생성 시간은 처리하는 로그 데이터의 수에 비례하여 증가한다.

본 도구의 성능은 다음과 같은 두 가지 평가 항목에 대해 평가 된다.

① 생성된 규칙의 정확성(CGR: Correctness of Generated Rules)

생성 되어야 할 규칙들이 도구를 통해 생성된 규칙들에 포함될 확률로서 생성되어야 할 규칙의 집합을 Setg, 도구를 통해 생성된 규칙들 집합을 Setr, 집합 Set의 크기를 |Set| 라고 정의하면 아래 수식(2)와 같이 계산 된다.

$$CGR = \frac{|Set_g \cap Set_r|}{|Set_g|}$$

수식(2)
Equation (2)

② 잡음 비율(NR: Noise Ratio)

NR은 생성된 규칙들 중 잘못된 규칙의 비율로서 아래 수식(3)과 같이 계산 된다.

$$NR = \frac{Set_r - |Set_g \cap Set_r|}{|Set_r|}$$

수식(3)
Equation (3)

따라서, CGR은 허용 확률에 반비례하고, NR은 허용 확률에 비례한다.

테스트는 제안하는 규칙 생성 방법의 허용 확률에 따른 규칙 생성 수와 단위 시간당 로그 데이터 처리량을 평가하고, 그 평가 결과를 분석하여 효과적인 허용 확률을 찾는다. 본 절에서는 규칙 생성 수와 로그 데이터 처리량, 그리고 허용 확률을 얻기 위한 시나리오와 테스트 결과에 대해 설명한다.

규칙 생성 수 및 시간당 로그 데이터 처리량 허용 확률에 따른 규칙 생성 수와 단위 시간당 로그 데이터 처리량의 평가는 다음과 같은 환경에서 진행되었다. 약 3만건의 로그 데이터를 대상으로 시뮬레이터 상에 존재하는 15개의 정보 가전에 대해 규칙을 생성한다.

효과적인 허용 확률을 분석하기 위해 유비쿼터스 홈 각 구성요소들에 대한 규칙을 개별적으로 평가하는 것은 의미가 없다. 성능 테스트에서 시뮬레이터가 사용되므로 입력을 구성할 때, 개별적 규칙에 대해서는 각 구성요소들의 시간에 따른 상태를 정확히 예측할 수 있고, 어느 정도 허용 확률이 나타날 지 미리 알 수 있다. 따라서 단일 규칙에 대한 테스트 보다는 유사한 시간대에 여러 규칙이 복합적으로 발생할 경우 CGR과 NR의 변화를 분석할 필요가 있다. 잡음(Noise)에 해당하는 데이터는 처음 생성된 로그를 통해 분석된 결과 중 허용 확률이 0.5 이하인 실제로는 규칙이 없는 규칙이 포함된 데이터가 사용되었다. 규칙 및 규칙에 대응되는 시나리오 수는 다양하므로, 모든 조합에 대해 CGR, NR을 얻는 것은 사실상 불가능하다. 따라서, 한번의 테스트에서 규칙들을 랜덤하게 조합해 가며 모두 100회에 걸친 테스트가 수행 되었고, 그 수행 결과를 토대로 NR/CGR이 최소가 되는 허용 확률을 얻었다. 오류 진단 규칙 생성 테스트는 제안하는 규칙 생성 방법의 허용 확률에 따른 규칙 생성 수와 단위 시간당 로그 데이터 처리량을 평가하고, 그 평가 결과를 분석하여 효과적인 허용 확률을 찾는다. 테스트 결과 규칙이 적어도 하나 이상 생성되는 최대 허용 확률은 0.9955이며, 모든 규칙을 생성하는 허용 확률은 0으로 이다. 허용 확률에 따른 규칙 생성 수는 대부분의 의미 없는 규칙은 낮은 허용 확률에서 생성되며, 분포가 매우 밀집되어 있다는 것을 알 수 있다. 따라서, 허용 확률을 낮게 조절해도 의미 없는 규칙들을 충분히 걸러낼 수 있음을 알 수 있다. 본 연구에서의 실험에 사용된 시뮬레이터는 이산 사건 시스템을 기반으로 설계 및 구현되어 입력 사건에 해당하는 배치(Batch)에 따라 출력 사건에 해당하는 로그 데이터를 정확히 생성한다. 규칙은 로그 데이터를 사건으로 치환하고 각 사건들 간의 발생 확률에 의해 얻어진다. 각 사건들 모두가 서로 연관성이 있을 것이라 가정하고 허용 확률에 따라 규칙 생성 수를 조절하는 것이 가능하도록 하였다. 오류 진단 지식의 생성에 대해서는 특정 시나리오에 대해 예측되는 오류 진단 지식과 실제로 생성된 오류 진단 지식의 비교하는 방법으로 실험

하였고, 오류의 원인이 발생하면 적어도 한 번 이상의 규칙이 위배되어야 해당 원인이 오류 진단 지식에 포함 된다는 결과를 얻었다. 이는 제안된 방법은 규칙의 만족 여부를 기준으로 오류의 발생을 판단하기 때문이며, 오류의 원인이 발생한 후 실제로 그 오류가 발생하면, 오류의 원인을 정확히 찾을 수 있다.

7. 연구 결론 및 연구 과제

본 논문에서는 유비쿼터스 홈이란 댁내 가전기기이 거주자에게 지속적인 서비스를 제공해야 한다는 관점에서, 오류가 발생했을 때 이를 유비쿼터스 홈 시스템 스스로가 회복할 수 있어야 함을 언급하였다. 이에 유비쿼터스 홈을 구성하는 장치들의 상태 변화를 알려주는 로그 데이터를 이용해, 인과 관계로부터 규칙 및 생성된 규칙을 기반으로 오류 진단지식을 생성하는 방법을 제안하고, 시뮬레이터를 이용해 생성된 로그 데이터를 바탕으로 진행한 실험을 통해, 규칙과 오류 진단 지식의 생성과 정확성을 확인하였다. 그 결과, 각 사건들 모두가 규칙 생성의 신뢰도에 있어서는 신뢰 계수를 0.1로 할 때, 허용 확률이 41.1%이상이면, 약 83.7%의 규칙 생성 정확도와 약 27.1%의 잡음비율을 보여준다는 실험 결과를 얻었고, 오류의 원인이 발생했음에도 불구하고 오류가 발생하지 않는 사례를 제외한 사례들에 대해서는 해당 오류에 대한 진단 지식을 구축할 수 있다는 결론을 얻었다. 유비쿼터스 홈 상에는 다양한 장치와 서비스가 존재하고 복잡한 소프트웨어로 구성되기 때문에 오류를 탐지하고 오류를 진단하기가 쉽지 않다. 본 연구는 유비쿼터스 홈의 신뢰성을 위해 오류 탐지와 오류 진단 지식을 생성할 수 있는 도구를 개발한다. 유비쿼터스 홈 상에서 오류를 발견 했을 경우, 개발된 도구를 이용하여 오류 발생과 관련 있는 Context를 분석하고, 오류 원인을 찾아 유비쿼터스 홈 장애 처리의 해결책으로 제시할 수 있다.

향후에는 현실세계의 다양한 디바이스, 센서 그리고 사용자간의 영향을 주는 상관관계를 분석하여 Context의 신뢰도를 높이는 연구가 필요하며 또한, 규칙의 정확성을 높이기 위한 방법과 오류 해결을 위한 정확한 원인 선택 방법에 대한 연구가 필요하다. 현재는 댁내라는 환경만을 고려하였으며, 대상 장치들에 오류가 발생해도 거주자 입장에서는 심각한 영향을 받지는 않는다. 그러나, 최근 들어 의료 장비들이 댁내에서 활용되기 시작하고 있으며, 유비쿼터스 헬스케어 등이 주목 받고 있다. 의료 장비들은 기존의 가전기기들에 비해 오류에 민감하기 때문에, 앞으로 유비쿼터스 의료 장비들 사이에 발생하는 오류에 대한 문제에 대한 연구가 필요할 것으로 본다.

References

[1]H. Oh, et al., "A Study of Context Aware Technology in Smart Home," Proceedings of the Annual Conference of the JSSD, Vol. 52, pp. 308 309, 2005.

[2] “The Adaptive House Boulder, Colorado,” http://www.cs. colorado.edu/~mozer/house/

[3] “Easy Living,” http://research.microsoft.com/easyliving/

[4] “Aware Home Research Initiative,” http://www.aw arehome.gatech.edu

[5] H. J. Lee and S. J. Kim, “A Standard Method Based User Oriented Integrated Architecture for Supporting Interoperability among Heterogeneous Home Network Middlewares,” IJSH, Vol. 1, No. 1, pp.58 64, 2007.

[6] D. Heckerman, "A tutorial on learning with Bayesian networks," Learning in Graphical Models, MIT Press, Vol. 1, pp. 301 354, 1998.

[7] L. Rabiner, B. Juang, “An introduction to hidden Markov models,” ASSP Magazine, IEEE, Vol. 3, No. 1, pp. 4 16, 2003.

[8] C. Angeli, and A. Chatzinikolaou, “On Line Fault Detection Techniques for Technical Systems: A Survey,” International Journal of Computer Science & Applications I(1), pp. 12 – 30, 2004.

[9] P. C. Utton and E. Scharf, “FDS for Fault Diagnosis in the Connected Home,” IEEE Communications Magazine, Vol. 42, No. 11, pp. 128 134, Nov. 2004.

[10] J. Park, et al., “Proactive Self Healing System based on Multi Agent Technologies,” Proceedings of the Third ACIS Int'l Conference on Software Engineering Research, Management and Applications, pp. 256 263, 2005.

[11] W. E. Walsh, et al., “An Architectural Approach to Autonomic Computing,” Proceedings of the International Conference on Autonomic Computing, pp.29, 2004.

[12] J. S. Park, M. K. Moon, S. G. Hwang, and K. H. Yeom, "CASS: A Context Aware Simulation System for Smart Home," Proceedings of the 5th ACIS International Conference on Software Engineering Research & Applications, pp. 461 467, 2007.

[13] M. Kumar, B. A. Shirazi, S. K. Das, B. Y. Sung, D. Levine, and M. Singhal, “PICO: A Middleware Framework for Pervasive Computing,” IEEE Pervasive

Computing, Vol. 2, No. 3, pp. 72 79, 2003.
[14] T. Gu, et al., “An Ontology based context Model in Intelligent Environment,” In Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, pp 270 275, 2004.
[15] P. Gray, D. Salber, "Modeling and using sensed context in the design of interactive applications," In Porceedings of 8th IFIP Conference on Engineering for Human Computer Interaction, Toronto, 2001.
[16] P. Sung, S. Andreas and B. S. Mani, “SensorSim: A Simulation Framework for Sensor Networks,” In Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems. Boston, Massachusetts, United States: ACM Press, pp. 102 111, 2000.
[17] S. Sundresh, W. Kim, G. Agha, “SENS: A Sensor, Environment and Network Simulator,” In Proceedings of IEEE/ACM Annual Simulation Symposium, 2004.
[18] J. J. Barton, V. Vijayaraghavan, “UBIWISE, A Simulator for Ubiquitous Computing Systems Design”, HP Labs Technical Report, Hewlett Packard Company, 2003.
[19] E. O’Neill, et al., “A Testbed for Evaluating Human Interaction with Ubiquitous Computing Environments,” In Proceedings of the 1st International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities (TRIDENTCOM 2005), IEEE Computer Society Press, Los Alamitos, California, pp. 60 69, 2005.
[20] D. W. Ryu, “A Method for Generating Rule based Fault Daignosis Knowledge on Smart Home Environment,” Journal of The Korea Academia Industrial Cooperation Society, Vol. 10, No. 10, pp. 2741 2749, 2009.

이 시 흥(Sl-Heung Lee) **[정회원]**

- 2002년 2월 : 대전대학교 컴퓨터통신동학과 (공학석사)
- 2009년 3월 ~ 현재 : 한서대학교 디지털포렌직학과 (박사과정)
- 1984년 5월 ~ 1999년10월 : 한국전자통신연구원 선임연구원
- 1999년10월 ~ 2001년 4월 : 한국과학기술정보연구원 선임연구원
- 2001년 5월 ~ 현재 : 한국인터넷진흥원 수석연구위원)

<관심분야>
정보보호, 정보통신, 임베디드SW, 유비쿼터스컴퓨팅, 디지털포렌직

김 홍 윤(Hong-Yoon Kim) **[정회원]**

- 1984년 2월 : 인하대학교 대학원 전자계산학과 (이학석사)
- 1996년 2월 : 인하대학교 대학원 전자계산학과 (이학박사)
- 1995년 3월 ~ 현재 : 한서대학교 컴퓨터공학과 교수

<관심분야>
센서네트워크, 디지털포렌직,