한국산학기술학회논문지
Vol. 10, No. 10, pp. 2741-2749, 2009

스마트 홈 환경에서 규칙 기반의 오류 진단 지식 생성 방법

류동우[1*]
[1]중앙대학교 컴퓨터공학과

A Method for Generating Rule-based Fault Diagnosis Knowledge on Smart Home Environment

Dong-Woo Ryu[1*]
[1]**Division of Computer Science & Engineering, Chung-Ang University**

요 약 스마트 홈에서 발생하는 다양한 형태의 오류는 스마트 홈의 신뢰성을 저하시키기 때문에 스마트 홈에서 오류의 검출 및 복구를 위한 연구가 그 동안 진행되어 왔으나, 이들 대부분은 장치의 기능적 고장이나 소프트웨어의 오동작 등에 한정되어 있고, 장치간의 연관 관계에서 발생하는 오류에 대한 것은 없었다. 본 논문에서는 장치간의 연관 관계를 규칙으로 정의하고, 규칙의 만족 여부에 따라 컨텍스트를 두 집합으로 구분한 다음, 장치간의 연관 관계에서 발생하는 오류의 증상과 원인을 정의하는 오류 진단 지식 생성 방법을 제시한다. 향후, 스마트 홈에 적용하여 이 방법을 장치들의 연관성에 의해 발생하는 오류의 탐지와 그 원인의 식별이 실시간으로 가능하다.

Abstract There have been many researches to detect and recover from faults on smart home environment, because these faults should lower its reliability. while, most of these researches have addressed functional defects of devices or software malfunction, few attempts have been made to deal with faults which may occur due to the inter relationships among devices. In this paper, we define the relationships among devices as rules, and propose a method for generating fault diagnosis knowledge which defines the symptoms and causes of faults. We classify the contexts of devices into two sets, depending on whether it satisfies the rules or not. when this method is applied to smart home environment, it is feasible not only to detect faults that may occur due to the relationships among devices but to identify their causes at real time.

Key Word : Relationships, Contexts, fault diagnosis, Symptoms, Smart home

1. 서론

스마트 홈은 가사 활동, 엔터테인먼트 등 다양한 분야에서 사용자의 행동에 대해 능동적으로 반응하여, 사용자가 처한 상황에 적합한 서비스를 제공할 수 있을 것으로 기대된다. 예를 들어, 콜로라도 대학의 Adaptive House[1]는 거주자의 생활 패턴과 요구사항을 댁내에 설치된 센서를 통해 관찰하여, 관찰 결과에 따라 환경을 변화시키고, 거주자가 필요한 것들을 미리 제공한다. 마이크로 소프트의 Easy Living[2], 조지아 공대의 Aware Home[3] 등도 유사한 목적을 가지고 개발되고 있다. 국내에서는 거주자의 명령 없이 스스로 거주자의 행위를 배워, 앞으로의 상황을 예측해 지능적인 서비스를 제공하기 위한 연구[4]와 댁내에서 디지털 콘텐츠의 수요가 증가함에 따라 저작권 관리가 힘들어 질 것으로 보고, RFID를 이용해 디지털 콘텐츠의 저작권을 관리하려는 연구[5] 등이 있었다.

이러한 스마트 홈 환경을 달성하기 위해서는 IEEE 802.11과 PLC (Power Line Comunication) 등을 제공하는 댁내 망, 홈 서버 등 다양한 장치로 구성된 기본적인 인프라가 구축되고, 장치 간의 상호 연동을 위해 각 장치에 미들웨어가 탑재되어야 하며, HDTV와 같은 멀티미디어 가전, 냉장고, 에어컨과 같은 백색가전 등 댁내의 모든 장치가 상호 연동될 수 있어야 한다. 이와 같이, 스마트 홈

*교신저자 : 류동우(rdw2486@naver.com)
접수일 09년 09월 29일 수정일 09년 10월 12일 게재확정일 09년 10월 14일

은 다양한 지능형 장치와 네트워크 기술로 구성되고 복잡한 소프트웨어로 구현되어 있어, 스마트 홈이 사용자가 기대하는 수준의 신뢰성을 보장하는 것이 쉽지 않다. 또한 다양한 미들웨어를 통합[6]하여 스마트 홈의 구조를 일관성 있게 유지함으로써 신뢰성을 확보하고자하는 연구[7]는 있었지만 스마트 홈의 구성요소에서 발생하는 오류의 검출 및 복구에 관한 연구는 거의 없었다.

스마트 홈에서 오류란 구성요소들 중 일부가 본연의 역학을 수행하지 못하는 것으로 사용자에게 필요한 서비스가 제공되지 못하는 결과를 가져오며 이는 스마트 홈의 신뢰성을 저하시킨다. 스마트 홈은 댁내 환경에서 발생한 오류를 파악하여 사용자에게 통지하고 가능한 경우 스스로 오류를 복구해 서비스가 정상적으로 제공됨을 보장할 수 있어야 한다. 기존 오류 관련 연구[8-12]는 이를 위해 단순히 고장이 발생한 장치를 검색하거나 단일 장치에 여러 사용자가 상충되는 명령을 내린 상황을 해결하고자 하였다. 하지만 장치 자체에는 아무런 문제가 없지만 장치들 간의 연관 관계에 의해 오류가 발생하는 경우도 존재한다. 예를 들어, 에어컨을 켰는데 창문이 열려 있어 방의 온도가 내려가지 않으면 오류가 발생한 것으로 간주될 수 있다. 이러한 오류는 장치의 기능상 문제와는 달리 스마트 홈이 오류의 원인들을 분석해 장치를 조작함으로써 해결될 수 있다. 본 논문에서는 스마트 홈에서 장치들 간에 발생할 수 있는 인과 관계를 정의하는 규칙을 이용하여 오류의 증상과 원인을 정의하는 오류 진단 지식 생성 방법을 제안하고, 시뮬레이션을 통해 생성된 오류 진단 지식을 확인한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 연구에서 다루어졌던 오류와 오류의 처리 방법을 살펴보고 본 논문에서 다루게 될 오류에 대해 설명한다. 3장에서는 스마트 홈 구성요소간의 관계를 이용한 규칙 생성과 생성된 규칙으로부터 오류 진단 지식을 생성하는 방법에 대해 설명한다. 4장에서는 구성요소간의 연관 관계를 시뮬레이션 하기 위한 시뮬레이터를 설계하고, 이를 통해 오류 진단 지식 생성을 테스트한다. 마지막으로 5장은 결론과 향후 과제에 대해 기술한다.

2. 관련연구

전통적으로 컴퓨터 분야에서 오류와 관련해 매우 다양한 연구가 진행되어 왔지만[8][9], 스마트 홈의 경우는 그 시스템이 정립되어 가는 과정에 있어 오류 관련 연구가 초기 단계에 있다. 현재까지 스마트 홈에서 활용 가능한 오류 관련 연구는 소프트웨어 오류를 대상으로 한 Proactive Self-Healing System[10], 대규모 컴퓨터 시스템에서의 소프트웨어 하드웨어의 오류를 대상으로 하는 Autonomic System[11], 그리고 사용자가 입력한 서비스를 기반으로 장치간의 규책을 생성하고 규칙으로부터 충돌을 탐지하는 CASS(A Context-Aware Simulation System for Smart Home)[12] 등이 있다.

Proactive Self-Healing System에서는 모니터링 에이전트, 컴포넌트 에이전트, 시스템 에이전트, 진단, 에이전트, 결정 에이전트 등 다양한 에이전트들이 협력하여 오류를 처리한다. 모니터링 에이전트가 응용프로그램이 생성한 로그나 운영체제의 에러 이벤트를 감지하면 컴포넌트 에이전트가 로그와 이벤트를 분류한 다음, 일관된 오류 메시지 형태로 변환해 시스템 에이전트로 전달한다. 시스템 에이전트는 오류와 관련된 시스템의 정보를 수집해 진단 에이전트가 오류의 원인을 분석할 수 있도록 지원한다. 진단 에이전트는 오류 증상을 저장하는 증상(symptom) DB를 검색해 오류의 원인을 검색하고, 결정 에이전트는 정책(Policy) DB와 원인에 따른 결정 테이블을 바탕으로 즉시 행동을 취하거나 오류 예방 작업을 수행한다. Proacive Self-Healing System은 다수의 에이전트로 구현되어 기능의 추가 혹은 변경이 유연해 구조상 유리하지만, 장치의 물리적인 오류나 장치간의 연관관계로 인해 발생하는 오류를 식별하지 못하고, 증상 DB, 정책 DB, 결정 테이블에 정의된 형태의 오류만 처리가 가능하다는 문제점이 있다.

Autonomic System은 엔터프라이즈 컴퓨팅에서 시스템이 동작하는 환경을 감시하고, 그 결과에 따라 시스템의 구성을 변경, 최적화 하는 등의 동작을 수행한다. Autonomic System은 변화에 맞추어 자신의 구성을 변화시키는 self-configuring, 오류를 해결하는 self-healing, 동작하는 환경에서 자신을 최적화하는 self-optimizing, 그리고 외부 침입에 대해 스스로를 보호하는 self-protecting 기능을 가진다. Autonomic System은 증상 DB를 이용하여 문제의 해결책을 찾는다. 증상 DB는 문제를 식별하기 위한 패턴과 문제를 해결하는 방법에 관한 정보를 저장한다. 그리고 Policy Manager는 검색된 해결책 주위에서 최적의 것을 선택한다. 예를 들어, 문제의 해결책으로 시스템을 재부팅 하는 방법과 스왑 공간을 늘리는 방법이 검색 되었다면, Policy Manager는 영업 시간에 컴퓨터를 재부팅 하면 안 되는 정책 때문에 후자를 선택한다. Autonomic System은 소프트웨어적인 문제뿐만 아니라, 시스템 구성요소의 물리적인 오류 등에도 대처할 수 있지만 역시 구성요소 간의 연관성으로 발생하는 문제는 해결하지 못한다.

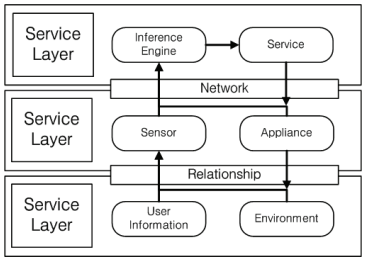
CASS는 일련의 조건과 결과로 이루어진 규칙을 기반

으로 센서가 생성한 컨텍스트와 규칙에포함되어 있는 조건을 비교해 규칙 간의 충돌을 검사할 수 있는 시뮬레이터이다. CASS는 생성된 컨텍스트와 규칙에 포함되어 있는 조건을 서로 비교해 규칙 간의 충돌을 검사할 수 있다. 규칙은 사용자가 입력한 서비스를 바탕으로 생성되며 특정 컨텍스트가 두 개 이상 규칙의 조건을 동시에 만족하고, 결과가 서로 상충될 때 충돌을 감지한다. CASS는 사용자가 입력한 서비스를 분석해 장치간의 연관 관계에서 발생하는 오류를 처리할 수 있다. 하지만 홈 네트워크 환경에서는 사용자가 입력한 서비스 외에도 에어컨과 온도 센서와 같이 물리적으로 연관성이 발생하는 경우도 존재한다.

이상에서와 같이 현재까지 홈 네트워크에서 발생하는 오류는 장치 혹은 소프트웨어의 기능적이 오류를 그 대상으로 하거나, 정상적인 상황에서 장치들을 사용하는 데 있어 상충되는 명령이 전달되는 상황을 해결하고자 하였다. 하지만, 장치, 소프트웨어 등이 모두 정상이고, 상충된 명령 또한 전달되지 않지만 오류가 발생하는 경우가 존재한다. 물론 CASS가 서비스를 통해 장치간의 연관성을 일부 파악하고 충돌 여부를 판단할 수 있지만 오직 사용자가 입력한 장치간의 서비스를 통해서만 얻는다는 문제점을 가진다. 예컨대, 에어컨과 보일러 등, 장치간의 물리적인 관계에서 발생하는 연관성에 대해서 CASS는 충돌 여부를 판단할 수 없다.

3. 장치간 규칙 및 오류 진단 지식의 생성

본 장에서는 스마트 홈에서 발생하는 장치간의 규칙을 알아보고 스마트 홈을 구성하는 장치들 간의 규칙 생성과 생성된 규칙을 기반으로 오류 진단 지식을 생성하는 방법을 설명한다. 일반적으로 스마트 홈은 센서를 통해 댁내 환경을 파악하고 가전기기를 통해 사용자에게 편의를 제공한다. 다시 말해, 스마트 홈을 구성하는 장치는 사용자 편의를 제공하는 가전기기와 상황 정보를 획득하는 센서로 구분된다. 댁내 상황을 인지하고, 사용자에게 서비스를 제공하는 관점에서 스타트 홈을 도식화 하면 그림 1 과 같이 세 가지의 서로 다른 계층으로 구성될 수 있다. 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 스마트 홈 시스템이 주변 환경의 변화를 인지하고자 할 때 사용되며, 가전기기는 스마트 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.



[그림 1] 스마트 홈 계층

이러한 스마트 홈 계층에서 규칙은 환경 계층과 서비스 계층에서 발생하며, “C이면 R이다.”와 같은 형태로 장치 계층 구성 요소 간의 인과 관계(Causality)를 의미한다. 여기서 "C"와 "R"은 각각 원인과 결과를 의미한다. 먼저 장치 계층과 서비스 계층의 동작을 살펴보면, 서비스 계층의 구성요소는 장치 계층의 센서와 가전기기의 상태를 입력으로 하여, 상황을 추론하고 적절한 가전기기를 동작시킨다. 여기에서 우리는 입력에 해당하는 가전기기들의 현재 상태를 원인으로, 그리고 서비스 제공을 위해 가전기기의 변경된 동작 상태를 결과로 규칙을 생성할 수 있다. 장치 계층과 서비스 계층의 구성요소에 대한 명세는 모두 스마트 홈 시스템이 직접 얻을 수 있고, 규칙 또한 이를 통해 역시 간단히 얻을 수 있다. 반면 환경 계층에서 발생하는 규칙은 물리적인 인과 관계를 포함하기 때문에 스마트 홈 시스템이 직접 파악하기 어렵다. 예를 들어, 스마트 홈 시스템이 창문이 열리면 온도가 내려간다는 규칙을 얻기 위해서는 적어도 규칙을 시스템 내에 내장하거나, 추론을 통해 파악할 수 밖에 없다.

스마트 홈 시스템은 센서의 측정치를 통해 댁내 환경을 파악하며, 환경이 변화했다는 것은 센서의 측정치가 일정 시간을 두고 변했다는 것을 의미하고, 센서의 측정치는 증가하거나 감소할 수 밖에 없다. 따라서 가전기기가 동작하는 동안 센서 측정치의 변화를 지속적으로 관찰하면 해당 가전기기가 환경에 미치는 영향을 파악할 수 있다. 하지만 다양한 가전기기들이 서로 상호작용하는 스마트 홈에서 시간적인 선후 관계에 의해서만 규칙을 파악하는 것은 한계가 있다. 예를 들어, 보일러와 에어컨이 동시에 동작하면, 온도에 대한 규칙이 적어도 하나의

가전기기에 대해서는 정상적으로 생성되지 않으며, 두 가전기기의 동작 강도에 따라 생성된 규칙이 다를 수도 있다. 물론, 가전기기 자체가 규칙을 내장하고, 있다면 가전기기가 댁내에 설치될 때 서비스 계층에서 규칙을 읽어들이면 된다. 그러나 이러한 방법은 규칙의 내장 방식의 표준화가 선행되어야 하며, 규칙이 내장될 수 없는 구식 장치(Legacy Device)들에 대해서는 규칙을 생성할 수 없다.

장치간의 연관 관계의 의한 오류 진단 지식은 이상에서 설명한 규칙을 기반으로 생성된다. 규칙의 원인이 발생했음에도 불구하고, 기대한 결과가 나타나지 않는 상황을 우리는 오류의 증상으로 볼 수 있으며, 규칙이 위배되었기 때문에 이 규칙의 원인이 가전기기의 오동작 원인이 될 수 있다. 오류의 원인은 오류 증상에 해당하는 규칙과 결과가 모순되는 다른 규칙을 통해 쉽게 추론 가능하지만, 서비스 계층에서 발생하는 장치들 간 관계의 복잡성과 장치의 특정 상태를 수집한 후에서야 오류의 원인이 파악될 수 있다.

3.1 장치간 규칙 생성

본 절은 장치 계층의 구성요소들을 이용해 간접적으로 규칙을 파악하는 방법을 설명한다. 장치간의 규칙 Rule은 C → R과 같은 형태로 표현되며, 원인 C가 만족되면 결과 R이 발생해야 함을 의미한다. 스마트 홈에서 규칙은 서비스 계층에서 서비스가 제공되는 과정과 환경 계층에서 가전기기가 물리적인 환경에 영향을 주는 과정에서 발생한다는 것은 이미 설명 하였다. 서비스는 모두 스마트 홈 시스템에 저장되므로 다른 작업 없이 얻을 수 있다.

장치 계층에서의 규칙은 센서의 역할에 대한 명세를 통해 센서가 어떤 환경 정보를 가져오는 지 알 수 있으므로, 가전기기가 환경에 미치는 영향은 센서 값의 변화를 관찰함으로써 파악될 수 있다. 가전기기의 동작이 환경을 변화시킨다면 아래의 두 조건이 만족된다고 가정한다.

조건 1) 만일 어떤 가전기기 A가 센서 S의 값을 증가시킨다면

* 특정시점 t에서 S의 값을 S_t라 하면 $S_t > S_{t-1}$ 이다.
* 가전기기 A와의 거리가 d일 때 S의 값을 S_d라 하면 $S_d \geq S_{d'}$ 이다.($d \leq d'$)

조건 2) 만일 어떤 가전기기 A가 센서 S의 값을 감소시킨다면,

* 특정 시점 t에서 S의 값을 St라 하면 $S_t < S_{t-1}$ 이다.
* 가전기기 A와의 거리가 D일 때 S의 값을 S_d라 하면 $S_d \leq S_{d'}$ 이다.($d \leq d'$)

가전기기의 동작에 따라 위 두 조건을 지속적으로 감시하면 가전기기가 환경에 미치는 영향이 파악될 수 있다. 규칙 Rule은 원인 C, 결과 R, 반응시간 T_r, 한계치 L로 구성된다. 원인과 결과는 장치 계층의 구성요소들의 상태이며, T_r은 원인이 발생한 후부터 결과가 나타나기까지의 시간으로서 오류 진단 지식 생성 과정에서 오류를 판단하기 위한 제한 시간으로 활용 된다. L은 가전기기가 더 이상 환경을 변화시키지 못하게 될 때의 값으로 L을 벗어난 경우 규칙의 만족 여부를 확인 할 필요가 없다.

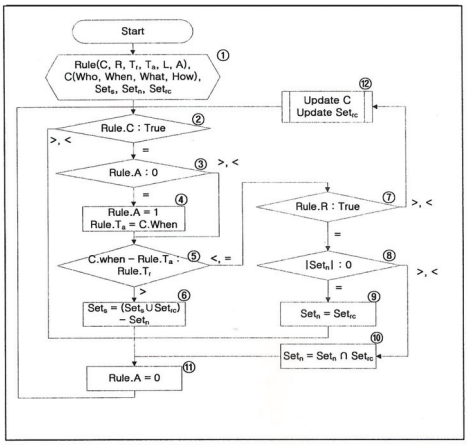
3.2 오류 진단 지식의 생성

본 절에서는 그림 2와 같은 순서도를 이용해 오류 진단 지식의 생성 방법을 설명한다. 오류 진단 지식은 특정 오류 증상을 일으키는 원인들을 찾기 위해 필요하며, 특정 증상 Sy와 Sy의 원인 집합 Sy_r에 대해 "Sy_r 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 T_r 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 Set_s와 어떠한 경우에도 오류의 원인이 될 수 없는(Not Suspected) 집합인 Set_n을 유지함으로써 오류의 원인을 식별한다. 집합 Set_n은 스마트 홈이 정상일 때의 컨텍스트들을 수집해 생성되며, 집합 Set_s는 오류 상황에서 발생한 컨텍스트들을 수집해 생성된다. 집합 Set_s의 컨텍스트들을 집합 Set_n의 컨택스트들이 이용해 제거되면, 원인 Sy_r이 식별될 수 있다.

기본적으로 어떤 규칙 Rule의 원인 C가 발생해도 결과 R이 발생하지 않는 것은 결과 R과 모순을 일으키는 또 다른 규칙 Rule'이 존재하기 때문이며 이는 규칙의 검색을 통해 원인을 파악할 수 있다. 하지만 원인 C에 해당하는 가전기기의 기능적 고장이 오류의 원인 될 수 있으므로 해당 가전기기에 관련된 모든 컨텍스트를 검색하는 것이 필요하다. 예를 들어, 가습기의 물이 부족한 상태라면 가습기가 동작해도 습도가 변하지 않을 것이다.

오류 진단 지식의 원인 집합은 그림 2의 순서도를 통해 생성된다. ①은 순서도에서 사용되는 데이터들의 선언이다. Rule은 규칙이며, 구성요소로서 원인(C), 결과(R), 반응시간(T_r), 한계값(L) 외에 활성화 여부를 판단하기 위해 활성화된 시간(T_a), 활성화 여부(A)를 부가적으로 가진다. C는 컨텍스트로서 발생 시간(When), 장치 이름(Who), 장치 속성(What), 변경된 값(How)으로 구성된다. 장치의 세부 정보는 장치 속성을 이용하여 표현된다.

Set_s, Set_n 그리고 Set_{rc}는 각각규칙 Rule에서 생성되는 오류 증상의 원인이 될 수 있는 컨텍스트 들의 집합, 어떠한 경우에도 원인이 될 수 없는 컨텍스트 들의 집합 그리고 장치 별로 가장 최근에 발생한 컨텍스트의 집합으로서 장치들의 가장 최근 상태를 나타낸다.



[그림 2] 오류 진단 지식 생성

②에서 Rule의 원인이 만족되면, ③과 ④의 과정을 통해 규칙을 활성화 하고 시간을 기록한다. ⑤에서 규칙의 제한시간을 확인해 오류 여부를 판단하고, 오류인 경우 ⑥에서 장치의 가장 최근 상태인 Set_{rc}에 속한 컨텍스트를 집합 Set_s에 추가하며, Set_n에 속한 컨텍스트를 삭제한다. ⑦에서 테스트 결과가 만족되면, ⑧,⑨ 그리고 ⑩에서 집합 Set_n을 조정한다. 그렇지 않을 경우, 컨텍스트 C와 Set_{rc}를 갱신하고(⑫) ②부터 다시 반복한다. ⑧에서 $|Set_n|$은 집합 Set_n의 크기이다. 마지막으로 집합들이 갱신되면, ②부터 모든 과정의 반복을 위해 ⑪에서 규칙이 비활성화 된다.

결국 집합 Set_s에서는 오류일 가능성이 있는 모든 컨텍스트를 보관하고, 집합 Set_n에서는 절대로 오류의 원인이 될 수 없는 컨텍스트를 유지하게 된다. 즉, 오류 진단 지식은 가능한 많은 오류를 추정해 집합 Set_s에 유지하고, 집합 Set_n를 통해 오류의 원인으로 간주될 수 없는 컨텍스트를 제거함으로써 생성된다.

여기서 주목할 점은 Set_n은 정상적인 상황 즉, 규칙이 만족될 때 구축된다는 것이다. 예를 들어, 만일 규칙 Rule(C→R)과 Rule'(C'→˜R)이 발견 되었다면, Rule에 대한 오류의 증상은 C^˜R이고, Rule'의 증상은 C'^R 이다. C와 C'이 동시에 발생한다 하더라도 R 혹은 ˜R은 반드시 참이므로 두 규칙이 동시에 활성화 되면 C와 C' 둘 중 적어도 하나의 원인은 상대방의 집합 Set_n에 추가 된다. R이면 Rule이 만족되어 Rule의 집합 Set_n에 C'이 추가되고, ˜R이면 Rule'이 만족되어 Rule'의 집합 Set_n에 C가 추가되게 된다. 차후 증상 C^˜R이 발생하더라도 이미 C'이 Rule의 집합 Set_n에 포함 되어 있으므로 C'은 Rule의 원인에 추가될 수 없게 된다. 원인은 집합 Set_s와 set_n의 차집합으로 얻어지고 C'은 이미 집합 Set_n에 추가되어 있기 때문이다. 이는 Rule'에 대해서도 동일하다.

따라서, 집합 Set_n가 Set_{rc}로부터 점차 확장되는 것은 오류의 원인이 되는 컨텍스트가 집합 Set_n에 포함되어 오류 진단 지식에서 누락되는 경우가 발생한다. 따라서 ⑨와 ⑩에서처럼 교집합을 통해 Set_n을 줄여 나가는 것이 필요하다.

4. 테스트

테스트는 가능한 실제의 장치를 대상으로 실시되는 것이 이상적이지만, 구성요소 변경과 시간당 데이터 생성량에 있어 효과적이지 못하다. 따라서, 본 논문에서는 시뮬레이션을 통해 오류 진단 지식 생성 방법을 테스트한다. 본 장에서는 제3장에서 언급한 홈 네트워크의 계층과 각 구성요소의 동작을 구현하기 위해 시뮬레이터를 설계하고, 연관성에 의해 발생할 수 있는 오류에 대한 사례를 제시하며, 그에 대한 테스트 결과를 분석한다.

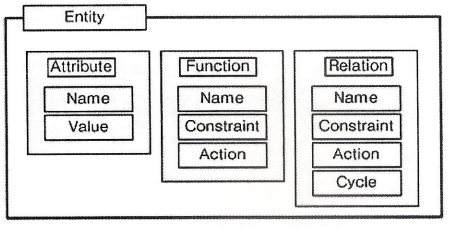
4.1 시뮬레이터의 설계

본 절에서는 제3장에서 설명한 오류 진단 지식 생성 방법의 동작을 확인하기 위해 컨텍스트 생성 모듈과 오류 진단 지식 생성 모듈로 구성된 스마트 홈 시뮬레이터를 설계한다. 컨텍스트 생성 모듈은 스마트 홈의 각 계층에 속하는 구성요소들을 속성, 기능 그리고 관계로 표현되는 엔티티로 추상화하고, 시뮬레이션을 통해 오류 진단 지식 생성에 필요한 컨텍스트를 생성한다. 오류 진단 지식 생성 모듈은 3.2절에서 설명한 오류 진단 지식 생성 방법을 구현한 것으로 특정 스마트 홈 모델에 대해 오류 진단 지식을 생성할 수 있으며, 독립적인 모듈로서 스마트 홈에 내장되어 실시간으로 오류 진단 지식을 생성하는 데 사용될 수 있다.

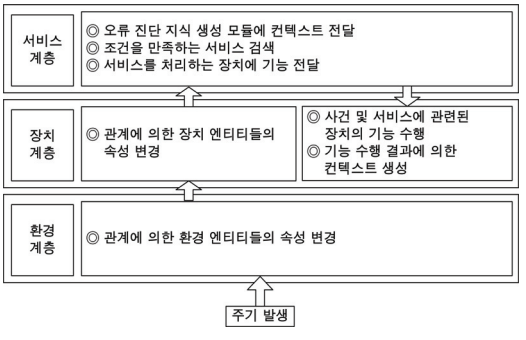
시뮬레이터는 스마트 홈의 댁내 환경 정보들을 어떠한 값으로 표현할 수 있어야 하며, 스마트 홈을 구성하는 환경과 장치 모두를 엔티티로 정의한다. 엔티티는 속성으로 표현되며, 엔티티사이에 정보를 전달하기 위해 기능과 관계를 가지고 있다. 예를 들어, 사용자 혹은 서비스가 장치를 켜면 해당 장치의 전원을 표현하는 정보가 '꺼짐'에서

‘켜짐’으로 변경되는 것처럼 장치는 기능을 이용해 자신의 속성을 변경할 수 있다. 반면에 댁내의 온도, 습도 등에 해당하는 환경은 기능을 가지지 않으며 시간과의 관계 혹은 관련 엔티티들이 가진 정보와의 관계에 따라 변경된다. 예컨대, 온도 값은 시간에 따라 변경되기도 하지만 에어컨의 동작에 따라 값이 변경될 수도 있다.

엔티티는 그림 3과 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 전원 상태, 볼륨, 온도 등의 정보를 표현한다. 기능을 통해 시뮬레이터는 사용자 혹은 서비스에 의한 가전기기의 조작을 시뮬레이션 하며, 이름, 제약 조건, 행동으로 구성된다. 제약 조건은 행동이 수행되기 위한 조건들이며, 행동은 모든 제약 조건이 만족될 때 어떻게 속성이 변경되는 지를 기술한다. 이를테면, "Power On"이라는 이름의 기능은 "Power" 속성 값이 “Off"이고 ”Power Supply" 속성 값이 "Sufficient"임을 제약 조건으로 하며 ”Power" 속성을 "On"으로 변경하는 행동을 가질 수 있다. 관계는 수행 주기(Cycle)가 포함된 기능으로서 시간에 따른 온도 변화 등 특정 사건 발생 여부와 관계없이 수행 주기마다 제약 조건을 검사해 행동 수행 여부를 결정한다. 매 주기마다 시뮬레이터는 그림 4와 같은 동작을 수행한다.



[그림 3] 엔티티 구성요소



[그림 4] 시뮬레이터의 계층별동작

4.2 사례 연구

본 절에서는 4.1절에서 설계된 시뮬레이터를 이용해 오류 진단 지식의 생성과 이의 정확성을 테스트 한다. 본 논문에서 대상으로 하는 스마트 홈의 오류는 장치간의 연관성에 의해 발생하며, 장치간의 연관성은 규칙으로 정의된다. CASS를 통해 이러한 규칙성을 사용자로부터 입력 받아 규칙 간의 충돌 여부를 찾아내는 시뮬레이션 환경 CASS[12]가 이미 구축된 적이 있다. 이를 통해 사용자는 자신이 입력하는 서비스에 의해 발생하는 장치들 간의 규칙에 대한 충돌 여부를 확인하여 이를 미연에 방지할 수 있다. 하지만, 3.1절에서 설명한 장치가 물리적 환경에 영향을 주는 과정에서 발생하는 규칙에 대해서는 사용자가 미리 알거나 충돌이 발생한다고 해서 그 중 하나를 임의로 제거하는 것이 불가능하다. 따라서 실시간으로 충돌이 발생한 규칙에 대해 증상을 파악하고 원인을 분석하는 기능이 요구된다.

그러나 규칙에 의해 정의되는 오류의 증상과 그 원인은 스마트 홈에서 발생 가능한 규칙들에 의존적이며, 규칙은 스마트 홈의 구성에 따라 다르게 생성될 수 있다. 또한 스마트 홈은 사용자의 요구에 따라 다양한 방법으로 구성될 수 있으므로 모든 스마트 홈에 대한 오류 진단 지식을 테스트하는 것은 사실상 불가능하다. 따라서, 모든 규칙에 대해 테스트하는 것은 큰 의미가 없고, 오류가 발생하는 형태를 가정하고 테스트를 수행하는 것이 효과적이다.

본 논문에서 찾고자 하는 오류의 형태는 모순이 되는 규칙들의 조건이 동시에 만족되는 경우이거나 장치의 특정 상태로 인해 기능이 수행되지 못하는 경우이다. 구체적인 예로서, 온도를 올리는 보일러와 온도를 내리는 에어컨이 동시에 동작하는 상황이 전자에 해당하며, 에어컨의 냉매가 부족하여 에어컨이 정상동작 함에도 불구하고 온도에는 아무런 영향을 주지 못하는 상황이 후자에 해당한다. 본 장에서 시나리오에 앞서 설명한 두가지 형태의 오류를 모두 포함하는 하나의 오류 진단 지식 생성 시나리오에 대해서 테스트를 수행하였다.

시나리오는 표 1과 같이 환경 계층의 온도를 매개로 하여 장치 계층의 에어컨, 보일러, 그리고 창문 사이에서 발생하는 연관 관계에 의한 오류와 그 원인을 찾기 위해 가전기기들을 동작시키는 일련의 사건들로 구성되었다. 시뮬레이션 환경에서 찾기 위해 가전기기들을 동작시키는 일련의 사건들로 구성되었다. 시뮬레이션 환경에서 보일러의 동작과 창문의 열림은 지속적으로 온도를 올리고, 에어컨의 동작은 온도를 내리도록 설정 되어 있다. 이는 “보일러가 동작하면 온도는 올라간다.”, “창문이 열리면 온도는 올라간다.” 그리고 “에이컨이 동작하면 온도는 내려간다.”라는 세 규칙을 유도할 수 있으며, 모순이 되는

규칙들의 조건이 동시에 만족되는 경우가 테스트될 수 있다. 또한, 에어컨의 동작은 “냉매” 속성의 “충분함”을 제약 조건으로 포함하여 장치 속성의 특정 상태에 의해 행동을 수행하지 못하고 결과적으로 장치의 특성 상태에서 기능이 수행되지 못하는 경우가 테스트된다.

[표 1] 오류 진단 지식 생성 시나리오

09년 05월 26일 06시 00분 : 시뮬레이션 시작

09년 05월 26일 06시 01분 : 보일러 동작
09년 05월 26일 06시 20분 : 에어컨 동작

09년 05월 26일 06시 40분 : 보일러 정지

09년 05월 26일 07시 01분 : 창문을 오픈
09년 05월 26일 07시 02분 : 창문을 닫음

09년 05월 26일 07시 15분 : 인위적으로 에어컨의 냉매를 제거

09년 05월 26일 07시 24분 : 시뮬레이션 종료.

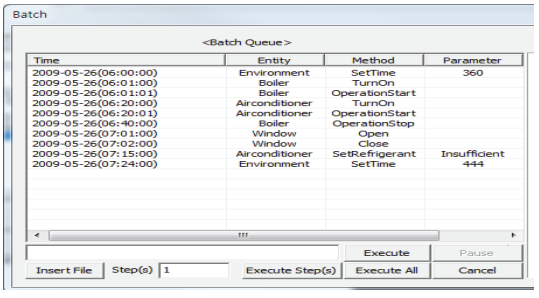
오류 진단 지식 생성 시나리오를 통해 우리는 “에어컨이 동작하면 온도는 내려간다.”라는 규칙에 대한 오류 진단 지식을 생성할 수 있으며, 에어컨(Aironditioner), 보일러(Boiler), 창문(Window), 그리고 냉매(Refrigerants)에 대해 표 2와 같은 오류 진단 지식이 생성될 것으로 예상된다.

[표 2] 예측되는 오류 진단 지식

증상	원인
에어컨이 동작 중이다. 하지만 온도가 내려가지 않는다.	보일러가 동작 중이다.
	창문이 열려 있다.
	에어컨의 냉매가 부족하다.

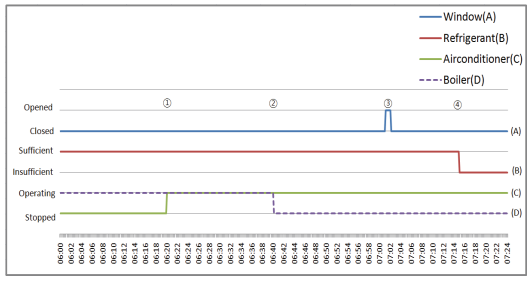
4.3 결과 분석

아래 그림 5는 시뮬레이터에 입력된 4.2절 표 1의 시나리오이다.

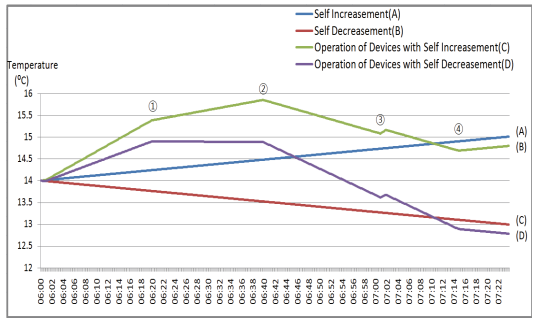


[그림 5] 시뮬레이터 상의 시나리오

4.2절의 테스트 시나리오는 장치 조작에 대해서만 언급되어 있다. 하지만, 온도는 시간에 따라 항상 같은 온도를 유지하지 않고, 장치의 동작되는 독립적으로 시간에 따라 변한다. 따라서 테스트는 같은 시나리오에 대해 두 번에 걸쳐 실시되었다. 첫 번째 테스트는 어떠한 가전기기도 동작하지 않으면, 온도가 증가하도록 설정되어 있고, 두 번째 테스트는 어떠한 가전기기도 동작하지 않으면, 온도가 감소하도록 설정되어 있다. 그림 6과 그림 7은 각각시뮬레이션 과정에서 발생한 컨텍스트로서 시간에 따른 각 가전기기들의 상태 변화와 온도 변화를 보여주는 차트이며, 시간은 “TT:MM"의 형태으로 표시되었다.



[그림 6] 시간에 따른 가전기기들의 상태 변화



[그림 7] 시간에 따른 온도 변화

그림 7의 Operation of Devices with Self Increasement에서 에어컨이 동작하는 동안 온도가 내려가지 않는 경우는 모두 세 번에 걸쳐 각각 다른 원인에 의해 발생한다. 아래 표 3은 시점 ①,②,③ 그리고 ④에서 집합 Set_s와 Set_n의 원소들이다. 최종 오류 진단 지식은 표 2에서 예측한 오류 진단 지식과 정확히 일치한다. 컨텍스트는 (장치, 속성, 값)의 형태로 기술되었다.

그림 7의 Operation of Devicesement의 경우, 에어컨 동작 후 창문이 개방되었을 때를 제외하고 규칙 “에어컨이 동작하면 온도는 내려간다.”를 만족 시키므로 각 시점

에서 표 4와 같은 집합이 생성 되며 최종 오류 진단 지식의 원인은 '보일러가 동작 중이다.'와 '창문이 열려있다.'로서 지식이 완전하게 생성되지 못함을 보여준다. 표 3과 비교할 때, 오류의 원인을 추정하는 집합 Set_s가 시점 ③에 단 한번 생성되고, 집합 Set_n는 정상적으로 동작하는 시점 ①에 생성되며, 보일러가 정지하는 시점 ②와 냉매가 제거되는 시점 ④에서 관련 보일러와 냉매에 관련된 컨텍스트가 교집합으로 제거된다. 만일, 관련 컨텍스트인 (Boiler, State, Stopped)와 (Airconditioner, Refrigerant, Insufficient)가 Set_n에 포함된다면 추후 그림 6의 Operation of Devices with Self Increasement에 해당하는 조건이 주어지더라도 보일러와 에어컨의 냉매 상태는 절대로 집합 Set_s에 포함되지 못한다. 이는 제3장에서 설명한 바와 같이 집합 Set_n을 합집합 연산을 통해 늘려가면 결국 절대로 오류 진단 지식이 생성될 수 없음을 의미한다.

[표 3] 시점에 따른 집합의 변화

시점	집합 Set_s	집합 Set_n
①	(Boiler, State, Operating) (Window, State Closed) (Airconditioner, State, Operating) (Tirconditioner, Refrigerant, Sufficient)	ϕ
②	(Boiler, State, Operating)	(Window, State, Closed) (Airconditioner, State, Operating) (Airconditioner, Refrigerant, Sufficient)
③	(Boiler, State, Operating) (Window, State, Opened)	(Window, State, Closed) (Airconditioner, State, Operationg) (Airconditioner, Refrigerant, Sufficient)
④	(Boiler, State, Operating) (Window, State, Opened) (Airconditioner, Refrigerant, Insufficient)	(Window, State, Closed) (Airconditioner, State, Operating) (Airconditioner, Refrigerant, Sufficient)

[표 4] 완전하지 못한 오류 진단 지식

시점	집합 Set_s	집합 Set_n
①	ϕ	(Boiler, State, Operating) (Window, State Closed) (Airconditioner, State, Operating) (Airconditioner, Refrigerant, Sufficient)
②	ϕ	(Window, State Closed) (Airconditioner, State, Operating) (Airconditioner, Refrigerant, Sufficient)
③	(Boiler, State, Stopped) (Window, State, Opened)	(Window, State Closed) (Airconditioner, State, Operating) (Airconditioner, Refrigerant, Sufficient)
④	(Boiler, State, Stopped) (Window, State, Opened)	(Window, State Closed) (Airconditioner, State, Operating)

이상에서 4장에서 설계된 시뮬레이터를 바탕으로 생성된 오류 진단 지식의 정확성을 테스트 하였다. 그림 6의 그래프 D와 생성된 각 집합을 통해 오류의 원인이 발생하면 적어도 한번은 규칙이 위배되어야 해당 오류가 Set_s에 포함될 수 있으며, 이후 Set_n에는 적어도 오류의 원인이 아닌 컨텍스트가 적어도 한 번씩은 나타날 필요가 있다는 것을 알 수 있다.

5. 결론 및 향후 연구

본 논문에서 스마트 홈을 구성하는 장치들의 인과 관계에 대한 규칙을 기반으로 오류 진단지식을 생성하는 방법을 제안하고, 시뮬레이션을 통해 오류 진단 지식의 생성과 정확성을 확인하였다. 스마트 홈은 다양한 장치와 댁내 망으로 구성되고 복잡한 소프트웨어로 구현되기 때문에 오류를 탐지하고 원인을 진단하기 어렵다. 또한, 기존의 오류 처리 기술로는 장치간의 연관 관계에서 발생하는 오류를 처리할 수 없었다. 일반적으로 장치의 물리적인 고장은 스마트 홈 시스템이 직접적으로 해결하는 것이 어렵지만 모든 장치의 동작이 정상인 경우에도 발생하는 장치 간의 연관 관계에 의한 오류는 장치의 동작을 조절함으로써 해결될 수 있다.

오류 진단 지식이 스마트 홈에서 활용되기 위해서는 우선적으로 규칙이 정확하게 생성되어야 하며, 오류 발생 시 이를 해결하기 위한 방법이 필요하다. 시간과 위치를 고려한 규칙은 비교적 정확하게 생성될 수 있지만, 동일한 장소에서 모순을 일으키는 장치가 동시에 동작하면 규칙 생성이 부정확할 수 밖에 없다. 또한, 본 논문에서 제안된 오류 진단 지식은 단순히 오류의 증상과 그 원인 기술에 그치고 있으므로, 오류 해결을 위해서는 원인들 중 제거되어야 할 대상을 선택해야 한다. 따라서, 규칙의 정확성을 높이기 위한 방법과 오류 해결을 위한 정확한 원인 선택 방법에 대한 연구가 필요하다.

참고문헌

[1] “The Adaptive House Boulder, Colorado", http://www.cs.xolorado.edu/ ~ mozer/house.

[2] "Easy Living", http://research.microsoft.com/easyliving.

[3] "Aware Home Research Initiative", http://www.awarehome, gatech.edu.

[4] J. Song, K. Oh, "System Architecture of Ubiquitous House based on Human Behavior", Journal of the Korean Academic Industral Society, Vol. 9, No. 5, pp.1304-1310, 2008.

[5] E. Kim, et. al., "A Study on Convenient Move of Didigal Contents Between Devices Using RFID in Home Network", Journal of the Korean Academic Industrial Society, Vol. 10, No. 2, pp.351-357, 2009.

[6] Hak Jin Lee, Sung Jo Kim, "A Standard Method-Based User-Oriented Integrated Architecture for Supporting Interoperability among Heterogeneous Home Network Middlewares", IJAH, Vol 1, No. 1, pp.58 -64, 2007.

[7] K. Moon, et al., "Universal Home Network Middleware Guaranteeing Seamless Interoperability among the Heterogeneous Home Network Middleware", Consumer Electronics, IEEE Transactions on, Vol.49, No. 3, pp.546-553, 2003.

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

[9] Peter Covington Utton, "FDS for Fault Diagnosis in the connected Home," Thesis for the Degree of Doctor of Philosophy in Electronic Engineering, University of London, 2006.

[10] Jeongmin 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] William E. Walsh, et al., "An Architectural Approach to Autonomic Computing", Proceedings of the International Conference on Autonomic Computing, pp.2-9, 2004.

[12] JoonSeok Park, Mikyeong Moon, Seongjin Hwang, and Keunhyuk Yeom, "CASS: A Context-AwareSimulation System for Smart Home," Proceedings of the 5th ACIS International Conference on Software Engineering Research & Applications, pp.461-467, 2007.

류 동 우(Dong-Woo Ryu) **[정회원]**

- 2004년 8월 : 중앙대학교 정보대학원 컴퓨터공학과 (공학석사)
- 2007년 8월 : 중앙대학교 일반대학원 컴퓨터공학부 (박사수료)
- 1995년 5월 ~ 2004년 2월 : 가톨릭대학교 중앙의료원
- 2004년 3월 ~ 현재 : 가톨릭대학교 의정부성모병원

<관심분야>
모바일, 홈네트워크, 정보보안, 소프트공학, U-Health, 유비쿼터스