오류트리를 이용한 오류 진단 및 추론 기법

고재헌*, 김영덕*, 민동욱*, 이현숙*, 김훈기*, 정석용*, 박정민*
*동양공업전문대학 전산정보학부
{episod2, jhko, appeal, hsrhee, kimhk, syjung, ya23ma}@dongyang.ac.kr

A Fault Diagnosis and Reasoning Method based on Fault Tree

Jaeheon Ko*, Yeongduck Kim*, Dongwook Min*,
Hyunsook Rhee*, Hoonki Kim*, Sukyong Jung*, Jeongmin Park*
*School of Computing and Information, DongYang Technical College

요 약

자가 치유 시스템은 자율 컴퓨팅의 개념 중 하나로 사람의 개입 없이 시스템의 이상상태를 인식하고 정상상태로 복귀 가능한 시스템을 의미한다. 발생한 오류는 또 다른 오류를 유발할 수 있고, 하나 이상의 원인이 되는 오류나 사건이 있을 수 있다. 따라서 오류의 원인에 따른 치유전략을 필요로 하며 발생한 오류에서부터 전이될 수 있는 오류에 대한 추론을 요구하게 된다. 따라서 본 논문에서는 오류의 인과관계에 따른 진단 및 추론 기법을 제안하고자 한다. 제안사항을 통해 오류트리를 기반으로 하여 발생한 오류의 원인이 되는 오류를 파악할 수 있으며, 오류의 원인에 따른 치유 전략을 계획 가능하고, 발생 가능한 오류의 추론이 가능하다.

1. 서 론

근래에 들어와 시스템의 복잡성에 대한 문제가 제기되고 있다. 그 해결책으로 대두되고 있는 자가 치유 시스템은 자율 컴퓨팅의 개념 중 하나로 시스템에서 발생될 수 있는 에러나 오류를 미리 예상하거나 감지하고 이러한 오류를 시스템 스스로 치유 혹은 수정함으로써 시스템의 오동작을 최소화하는 것을 의미[1]한다. 이를 달성하기 위해서는 가장 먼저 필요한 것이 오류의 원인을 파악하는 것이다. 발생한 오류는 또 다른 오류를 유발할 수 있고, 하나 이상의 원인이 되는 오류나 사건(event)이 있을 수 있다. 따라서 오류의 원인에 따른 치유전략을 필요로 하며 발생한 오류에서부터 전이될 수 있는 오류에 대한 추론을 요구하게 된다. 하지만 기존의 연구는 오류의 인과관계가 아닌 하드웨어적 외부상황을 통해 오류를 진단하므로 이전에 발생한 오류로 인한 결과적인 오류의 원인을 진단할 수 없다.

따라서 본 논문에서는 오류의 인과관계에 따른 진단 및 추론 기법을 제안하고자 한다.

1) 관리기(Manager)
- 오류 트리 및 오류 정보를 보관
- 평가기와 계획자에 의해 참조

2) 평가기(Evaluator)
- 오류 트리를 참조 하여 발생한 오류를 통해 원인을 파악
- 다른 오류로의 전이여부를 판단

3) 계획자(Planner)
- 각 오류의 원인에 따른 치유 전략을 계획

제안사항을 통해 오류트리를 기반으로 하여 발생한 오류의 원인이 되는 오류를 파악할 수 있으며, 오류의 원인에 따른 치유 전략을 계획 가능하고, 발생 가능한 오류의 추론이 가능하다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구를 설명하고, 3장에서는 제안 사항을 서술한다. 4장에서는 사례연구를 소개하고, 5장에서 결론을 맺는다.

2. 관련연구

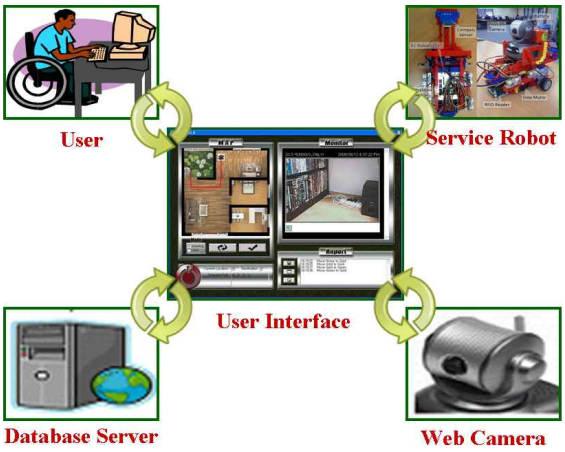
2.1 네트워크 기반의 지능형 서비스 로봇

우리는 선행 연구로 지능형 서비스 로봇을 제안했다[2]. 제안 로봇은 EZ-Robomaster[3]와 Lego Brick을 이용하여 제작하였고, 사용자 애플리케이션과 이동 로봇, 웹 카메라, RFID 시스템으로 구성되어 다음과 같은 기능을 수행한다.

1) 격자형으로 구역화 된 RFID Tag를 이용하여 로봇의 위치를 인식

2) 컴퍼스 센서를 이용하여 로봇의 방향을 인식

3) 전방 초음파 센서를 이용하여 장애물 인식
4) 사용자 애플리케이션을 통해 로봇을 제어



[그림 1] 시스템 구성도

그러나, 사람들과 가장 가까운 곳에서 작동하는 로봇이기 때문에 로봇의 오작동이 발생할 경우 안전과 직결될 수 있다. 이러한 문제점을 최소화하기 위해 본 논문에서는 우리의 이전 연구를 목표시스템으로 적용하여 오류 진단 기법을 실험한다.

2.2 오류 이벤트 모델을 이용한 모델 기반 진단 기법

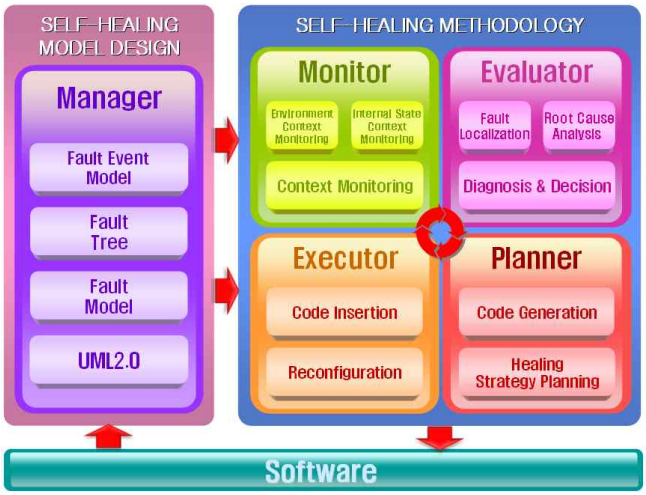
진단 시스템들의 중요한 성능 중에 하나는 오류(fault)와 증상(symptom) 사이의 원인 관계를 추론하는 것이다. 따라서 “오류 이벤트 모델을 이용한 모델 기반 진단 기법[4]”에서는 컴포넌트의 정상상태(normal state)에서 오류상태(faulty state)로 이르는 상태“오류를 오류 이벤트(fault event)라 정의하고, 오류 이벤트 모델(fault event model)기반의 진단 기법을 제안한다.

1) 제안 방법론은 오류 이벤트의 원인 관계를 제공하여 컴포넌트가 오류 상태에 이르는 이유와 방법을 설명할 수 있다.
2) 오류 이벤트 모델 기반의 추론은 의도된 행동 모델을 기초로 한 전통적인 방법론을 보완할 수 있다.
3) 전통적인 방법론과 추론기반의 오류 이벤트 모델을 구성하는 추론 시스템의 구조를 나타낸다.
4) 통합된 추론 시스템은 전통적인 방법론에 의해서 감지된 오류들을 기초로 더 중요한 원인들을 나타낼 수 있다.

본 논문에서는 위의 기법에 기반을 둔 오류 이벤트 모델과 오류 트리를 이용해 목표 시스템의 오류의 원인을 추론하고 결과를 진단하는 오류 평가에 적용 하였다.

3. 제안 사항

제안 사항을 적용하기 위해 자가 치유 시스템은 다음 [그림 2]와 같은 구조로 구성되어야 한다. 관리기(Manager)는 오류 트리 등 오류의 정보를 저장하고 있으며, 다른 구성요소에 의해 참조된다. 감지기(Monitor)가 오류를 감지하고, 평가기(Evaluator)는 오류의 원인을 진단 및 평가한다. 계획자(Planner)는 오류의 원인에 따른 치유전략을 계획하고 실행기(Executor)가 이를 실행하여 치유하는 구조이다. 본 논문에서는 오류 트리를 이용한 오류의 진단 및 추론 기법을 제안하므로 관리기, 평가기, 계획자를 다룬다.

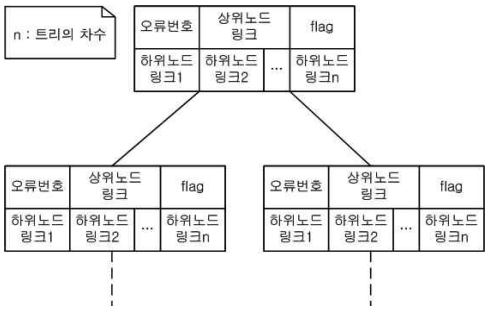


[그림 2] 자가 치유 시스템 구조

3.1 관리기

관리기는 목표 시스템과 자가 치유 시스템 사이에 정적으로 존재하며, 오류들의 인과 관계에 의해 정의된 오류 트리와, 모든 오류들의 정보(예를 들면, 오류 번호나 치유 전략 등)를 보관하고 있다. 이러한 정보들은 평가기, 계획자에 의해 참조된다.

각 오류는 오류 정보 배열에 오류 번호와, 치유전략, 제약조건, 오류 트리의 해당 노드를 가리키는 포인터가 저장되어 있으며, 오류 트리는 [그림 3]과 같이 각 노드마다 오류 번호, 평가기가 검색했음을 알리는 플래그, n개의 하위 노드 링크, 1개의 상위 노드 링크로 구성된다.



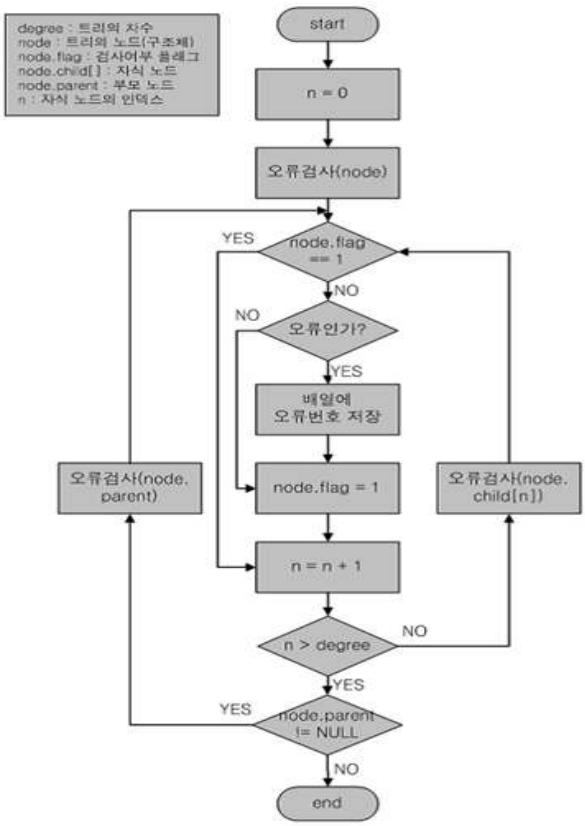
[그림 3] 오류 트리 구조

오류트리는 하위로 내려갈수록 구체적인 오류로 구성되며, 하위노드는 부모노드에 대한 원인이 될 수 있는 오류로 구성된다.

이 정보들은 개발자에 의해 정적으로 정의되어 관리기에 보관되며 시스템이 실행 중에 데이터가 변하지 않는다.

3.2 평가기

평가기는 감지기로부터 수신한 오류정보를 오류 테이블 및 오류 트리에 정의된 오류정보와 비교하여 오류의 발생 원인을 진단하고 발생 가능한 오류를 추론하며 계획자로 오류 원인정보를 전송하는 역할을 [그림 4]와 같은 단계로 수행한다.



[그림 4] 평가기 프로세스

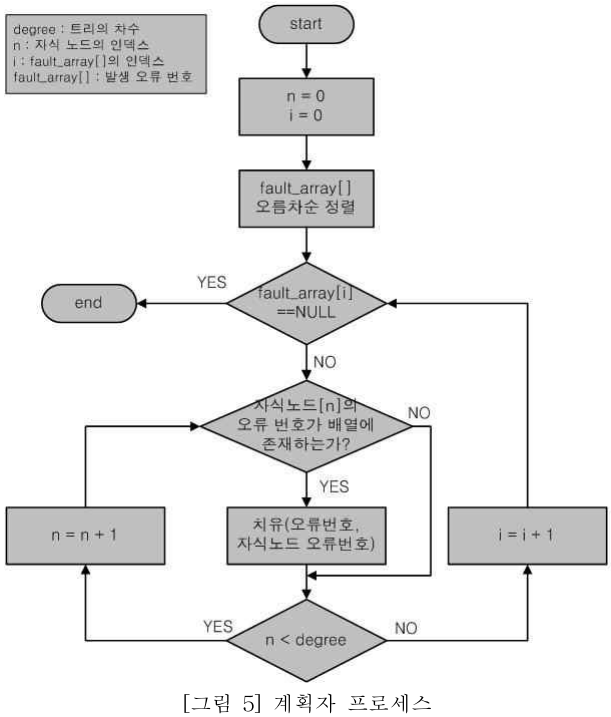
감지기로부터 전송받은 오류번호로 오류 정보 테이블에서 트리 노드의 주소를 얻는다. 해당 노드의 제약 조건을 검사하여 오류여부를 판단하고, 오류인 노드는 오류번호를 오류배열에 저장한 후 검사한 노드의 플래그를 true로 변경한다.

이와 같은 방법으로 하위노드의 오류를 검사하며, 오류라고 판단된 노드는 오류번호를 배열에 추가 한다. 하위 노드의 검색이 끝났으면 현재 오류로 인해 발생할 수 있는 상위노드의 오류를 추론 가능하므로 상위노드를 재귀적으로 호출한다. 상위노드가 오류라고 판단되는 경우 원인을 진단하기 위해 다시 하위노드를 검사 하게 된다. 이때, 플래그가 true인 노드와 오류가 아니라고 판단된 노드의 하위는 검사를 생략한다.

오류 트리의 모든 노드를 검사하여 시스템에 발생된 오류 정보를 수집한 평가기는 발생한 오류 번호를 담고 있는 오류 배열을 계획자로 전송한다.

3.3 계획자

계획자는 평가기로부터 수신된 오류의 발생 원인에 기반을 두어 [그림 5]와 같은 단계를 거쳐 치유 전략을 결정하고 실행기로 전송한다.



[그림 5] 계획자 프로세스

계획자는 오류트리의 상위노드부터 치유를 하기위해 평가기로부터 전송받은 오류배열을 오름차순으로 정렬하고, 정렬되어진 오류배열에서 순차적으로 해당 오류의 하위노드(원인)가 배열에 저장되어있는지

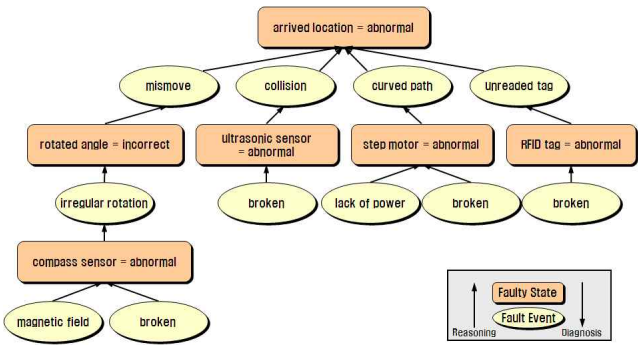
검색한다. 원인이 되는 하위노드를 발견했다면, 관리기의 오류 정보를 참조하여 해당오류의 번호, 원인 오류의 번호를 인자로 해당하는 치유전략 함수를 실행기로 전송한다. 같은 방법으로 다음 오류에 대한 치유전략을 결정하여 실행기로 전송한다.

4. 사례 연구

본 절은 앞서 기술한 진단 및 추론 기법을 우리의 이전 연구인 네트워크 기반의 지능형 서비스 로봇에 적용시켜 시스템이 오류 발생 시 오류의 발생 원인을 진단하고, 이후 발생 가능한 오류를 추론하는 과정을 기술한다.

4.1 관리기의 구현

관리기는 평가기와 계획기의 실행에 참조되는 오류 트리, 오류 이벤트 테이블, 치유 전략 테이블 등이 정의된다. 목표 시스템의 오류 트리는 [그림 6]와 같이 구성된다.



[그림 6] 오류 트리

4.2 진단 및 추론 기법

진단 및 추론 기법이 내장된 목표시스템은 오류가 발생한 경우 평가기를 통해 오류의 원인을 진단하고, 원인에 따른 치유 전략을 실행할 수 있다. 또한 이 후 발생 가능한 오류를 추론하여 이에 대비할 수 있다.

예를 들어, 감지기로부터 회전각도 부정확함(rotated angle = incorrect)오류가 발생했음을 전송받은 경우, 평가기는 하위 노드인 컴퍼스 센서 비정상(compass sensor = abnormal)을 검사하여 회전각도 부정확함 오류의 발생 원인이 컴퍼스 센서의 비정상인 것으로 진단하고, 상위 노드인 잘못된 이동(arrived location = abnormal)이 발생할 수 있음을 추론한다. 오류 트리의 운행을 완료한 평가기는 컴퍼스 센서의 오류로 인해 회전각도 부정확함 오류가 발생했음을 계획자로 전송한다. 계획자는 수신한 오류 원인 정보를 통해 컴퍼스 센서로 인한 회전은 잘못된 이동을 유발할 수 있음 인식하고 대체 센서인 초음파 센서를 통해 회전하는 치유 전략을 실행한다.

5. 결론

본 연구에서 제안한 오류트리를 이용한 오류 진단 및 추론 기법은 다음과 같은 장점을 갖는다.

- 발생 오류의 원인을 진단 가능
- 원인에 따른 치유 전략 결정
- 발생 가능한 오류를 예측하여 대비 가능

본 기법이 적용된 목표 시스템은 오류 트리에 정의된 오류 상태들의 인과 관계에 따라 오류를 진단 및 추론함으로써 발생한 오류에 대해 일괄적인 대응이 아닌 오류의 원인에 따른 치유를 수행하는 유연함을 보였고, 오류가 발생된 시점에서 이 후 발생할 오류를 추론하여 시스템의 강건성이 높아졌다.

그러나, 본 기법은 오류 트리에 정의되지 않은 오류에 대해서는 대응 방안이 없다는 문제점이 있다. 이미 제작된 시스템에서 새로운 오류가 발생할 때마다 지속적으로 업데이트를 하는 것은 관리에 대한 요구도가 높아지므로 비효율적이다. 이 문제를 해결하기 위해서는 시스템 스스로가 예상하지 못한 오류를 자동 확장하는 방안이 필요하다. 이는 향후 연구에서 다룰 예정이다.

참고문헌

[1] IBM, “IBM and Autonomic Computing ", http://www-03.ibm.com/servers/autonomic/

[2] 고재헌, “Ez-Robomaster를 이용한 네트워크 기반의 서비스 로봇 시스템의 설계와 구현” 한국정보과학회 제35회 추계학술대회 논문집 2008. 10.

[3] EASYTECH, http://www.ezlab.com

[4] Yoshinobu Kitamura, "A Model-based Diagnosis with Fault Event Models", Proc. of Pacific Asian Conference on Expert Systems, pp.322-329, 1997

[5] 김재선, “자기적응형 소프트웨어를 위한 목표 기반의 외부상황 평가 기법” 정보과학회 논문지, 제 33권 제3호, pp.316-334, 2006.

[6] Philip Koopman, "Elements of the Self-Healing System Problem Space"

[7] Michel E. Shin, "Detection of anomalies in software architecture with connectors" Science of Computer Programming 61 2006.