논문 2013-08-42

고신뢰 CPS를 위한 자율제어 시스템에 관한 연구

(A Research on Designing an Autonomic Control System Towards High-Reliable Cyber-Physical Systems)

박 정 민, 강 성 주, 전 인 걸, 김 원 태*
(Jeongmin Park, Sungjoo Kang, Ingeol Chun, Wontae Kim)

Abstract : Cyber-Physical system(CPS) is characterized by collaborating computational elements controlling physical entities. In CPS, human desire to acquire useful information and control devices anytime and anywhere automatically has increased the necessity of a high reliable system. However, the physical world where CPS is deployed has management complexity and maintenance cost of 'CPS', so that it is impossible to make reliable systems. Thus, this paper presents an 'Autonomic Control System towards High-reliable Cyber-Physical Systems' that comprise 8-steps including 'fault analysis', 'fault event analysis', 'fault modeling', 'fault state interpretation', 'fault strategy decision', 'fault detection', 'diagnosis&reasoning' and 'maneuver execution'. Through these activities, we fascinate to design and implement 'Autonomic control system' than before. As a proof of the approach, we used a ISR(Intelligent Service Robot) for case study. The experimental results show that it achieves to detect a fault event for autonomic control of 'CPS'.

Keywords : High-Reliability, Cyber-Physical Systems, Goal Model, Autonomic Control System

Ⅰ. 서 론

오늘날의 컴퓨팅 기술의 환경은 정보 기술 인프라, 다양한 컴퓨팅 장치, 소프트웨어, 그리고 시스템 자원들이 결합되어 운영되고 있다. 즉, 다양한 시스템들이 서로 연동되어서 공통의 목표(goal)를 달성하는 'Cyber-Physical Systems(CPS)'과 같은 형태로 진화하고 있다 (예를 들어, 스마트 하이웨이, 스마트 항만, Live-Virtual-Constructive 연동 시스템 등). 'CPS'는 센서, 네트워크, 소프트웨어, 하드웨어가 함께 결합된 대규모 임베디드 시스템을 의미한다. 이것은 크게 두 가지의 구성 요소를 가지는데 첫째, 소프트웨어의 도움으로 존재하는 '가상 환경(Cyber world)'과 둘째, 우리가 현실 세계를 직접 보고 만질 수 있는 '물리 환경(Physical world)'을 가진다. 즉, 'CPS'의 의미는 '가상 환경'을 통해 실제 '물리 환경'을 인지하고, 인지된 상황을 시뮬레이터의 시뮬레이션을 통해 실제 환경을 제어 할 수 있는 시스템이라고 정의 할 수 있다 [1].

다양한 환경에서 상호작용하는 'CPS'는 내부에 존재하는 여러 컴포넌트들의 결합과 연동작용들에 의해서 매우 복잡해져 가고 있다. 이러한 환경에서 발생하는 문제들을 인간이 감지, 분석, 해결하는 것은 이용 가능한 인적 자원과 효과적인 비용관리 측면에서 볼 때 명확한 제약이 있다. 심지어는 전체 컴퓨터 시스템의 오류 중 약 40%가 관리자의 오류에 의한 것이라고 한다[2]. 따라서 인간의 개입에 의존하는 현재 시스템 관리 방식의 개선을 위해 '자율제어 시스템(autonomic control system)'

*Corresponding Author (wtkim@etri.re.kr)
Received: 6 Sep. 2013, Revised: 4 Oct. 2013, Accepted: 18 Oct 2013.
J.M Park, I.G Chun, S.J. Kang, W.T. KIM: ETRI
※ 본 연구는 미래창조과학부 및 한국산업기술평가관리원의 산업원천기술개발사업의 일환으로 수행하였음. [10035708, 고신뢰 자율제어 SW를 위한 CPS(Cyber-Physical Systems) 핵심 기술 개발]

ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2013.8.6.347

은 매우 중요한 이슈가 되고 있다. ‘자율제어 시스템’을 개발하기 위해서는 관리 대상이 되는 ‘CPS’를 분석, 설계하는 것은 매우 중요하다. 숙련된 전문가들에 의해서 분석된 결과물과 설계된 산출물들을 기반으로, ‘CPS’를 모니터링하고, 평가하여, 전략을 계획하고, 계획된 전략을 실행하는 것은 매우 정형화되어 있는 방법이다[3]. ‘자율제어 시스템’에 대한 기존 전통적인 연구들 의 공통적인 장점들은 ‘자율제어 시스템’이 가져야할 기본 능력들을 잘 내재화 시키고 있다 (예를 들어, 모니터링-문제 분석-전략 수립-전략 실행).

그러나 기존 연구들은 보다 복잡한 환경을 가진 ’CPS’에 적용하기는 쉽지 않다. 그 이유는 1) ‘CPS’의 관리 목표(goal)가 무엇인지 명확히 알 수 없고, 2) ‘CPS’가 직면할 오류 상황들을 예측하기 쉽지 않다. 3) 방대한 ‘CPS’를 분석하고 설계해야 하는 분석의 노력이 매우 크다. 따라서 본 논문에서는 이러한 문제들을 해결하기 위해서 ‘CPS’에 적용 가능한 ‘자율제어 시스템’에 관한 설계 방법을 제안한다.

제안 방법론은 총 8단계로 1)오류 분석, 2)오류 이벤트 분석, 3)오류 모델링, 4)오류 상태 해석, 5)전략 결정, 6)오류 감시, 7)진단&추론, 8)전략 실행을 수행한다. 제안 사항들을 통해 고신뢰 ‘CPS’를 위하여 ‘CPS’가 직면하는 오류 상황들을 분석하고, ‘자율제어 시스템’의 설계를 보다 용이하게 할 수 있다. 또한 ‘CPS’에 대한 분석의 노력을 최소화하면서 관리 목표를 명확하게 설정할 수 있다. 제안 방법론의 사례 연구로써 우리의 이전 연구인 ‘지능형 서비스 로봇[4]’을 적용하고, 평가를 위해 오류 주입 기법[5]을 이용한다. 이를 통해 ‘자율제어 시스템’의 개발을 용이하게 하는 방법론의 유효성을 입증한다. 본 논문은 다음과 같이 구성된다. 2장에서는 전통적인 ‘자율제어 방법론’들과 연구 동향을 기술하고, 3장에서는 본 연구의 제안 방법론을 소개한다. 4장에서는 실험 및 평가를 설명하고 마지막 5장에서는 결론을 기술한다.

Ⅱ. 관련연구

최근 애플리케이션이나 컴퓨팅 환경에 자율제어 방법론이 적용되고 있다. 본 장에서는 전통적인 ‘자율제어 방법론’들에 대한 특징 및 장점, 취약점들을 분석한다.

1. Adaptive Service Framework[6]

IBM과 CISCO가 공동으로 연구한 ‘Adaptive Service Framework(ASF)’는 크게 ‘관찰(Monitoring)’, ‘로그 변환(Translation)’, ‘여과(Filtering)’, ‘분석(Analysis)’, ‘진단 및 치유(Diagnosis)’, ‘피드백(Feedback)’ 6단계의 처리 프로세스를 제안 하였다. 자율제어 엔진을 통해 다양한 컴포넌트에서 생성되는 로그들을 분석하고, 문제 발생 시 이를 ‘자율제어’하는 특징을 가진다. 장점으로는 로그의 내용이 실시간으로 기록되어 데이터베이스로 구조화하기 쉽고 상향식 분석이 용이하다. 그러나 로그가 발생하지 않는 경우, 시스템의 내부 관점을 모니터링하고 분석해서 ‘자율제어’하는 방법론이 적용되지 않았다. 로그가 발생하지 않아도 시스템의 ‘자율제어’할 수 있는 방법론이 필요하다.

2. Heart Beating Approach[7]

미국 ‘Columbia University’에서 연구한 ‘Heart Beating Framework’는 컴포넌트의 주기적인 신호를 통해 컴포넌트의 상태가 정상인지 비정상인지를 판단하는 방법을 제안하였다. 이것의 모니터링 방법론은 하드웨어(timer)와 소프트웨어 (heartbeat generator)의 조합을 기반으로 컴포넌트가 주기적으로 모든 것이 정상동작 하고 있다는 것을 가리키는 “I am alive” 신호를 발생 시킨다. 이 신호의 부재는 시스템에 문제가 있다는 것을 나타낸다. 즉, ‘fault’나 ‘problem’이 있다는 것을 나타낸다. 소프트웨어에 의해 주기적으로 ‘heartbeat’ 신호가 오지 않는 경우, 임베디드된 프로세서들은 ‘reset / restart’를 할 수 있는 특징을 가진다. 장점으로는 ‘I am alive’ 신호를 ‘time testing’하여 컴포넌트 또는 시스템 전체의 비정상 컴포넌트를 감지하는데 용이하고, 시스템이나 컴포넌트를 감시하는 모니터가 내부 상태를 확인하지 않기 때문에 자가 적응 모듈의 리소스 소모량을 줄일 수 있는 장점이 있다. 그러나 컴포넌트의 내부 객체에서 발생한 오류를 탐지하기 어렵고 시스템의 기능적 행동을 중지 시켜 중요 데이터를 잃게 할 수도 있다. 이러한 문제를 해결할 수 있는 여러 가지 방법 중 하나로 ‘오류 모델링’이 요구될 수 있다.

3. UML기반의 자율제어 방법론[8]

우리의 이전연구 에서는 소프트웨어 설계단계에서 생성된 UML 모델을 이용하여 시스템의 외부 자원 환경과 내부 상태 환경을 파악할 수 있는 자가 치유 소프트웨어 구조를 제안 하였다. 배치모델에서

명세한 자원 조건을 기반으로 외부 자원 환경의 비정상적인 상태를 감지하고, 클래스 모델, 시퀀스 모델, 상태 모델, 활동 모델에서 분석된 XMI 정보들을 기반으로 변환된 감지 코드를 컴포넌트 내부에 삽입하여 컴포넌트의 내부 상태의 비정상적인 동작을 감지하는 특징을 가진다. 장점으로는 자율제어 소프트웨어를 구현하는 개발자의 분석의 노력을 줄이기 위해 목표시스템의 설계 모델을 이용하여 외부, 내부 상황의 제약조건을 자동 생성할 수 있는 장점이 있다. 그러나 설계 모델들에서 생성된 XMI 정보를 기반으로 컴포넌트의 실행환경과 내부 상태를 감지할 수 있으나 컴포넌트 간의 'communication fault', 'interaction fault'를 감지할 수 없으며, 가장 범용적으로 사용할 수 있는 범위가 매우 한정적이다. 이러한 문제를 해결하기 위해서는 보다 진보된 '자율제어' 방법론이 요구된다.

4. 모델 기반의 자율제어 방법론[9, 10]

미국 'Carnegie Mellon University'에서 연구된 가장 전통적인 자율제어 방법으로 모델 기반의 자율제어 방법론들은 시스템을 모니터링, 해석, 분석, 재구성하기 위해 사용될 수 있는 계층화된 구조적 분석 모델을 제안하였다. 이 방법론들은ADL(Architecture Description Language)을 기반으로 관리 대상이 되는 시스템의 외부화 관점에서 최적화된 자원 환경이나 프로세스를 모델링 한 것이다. 장점으로는 시스템의 수정을 일으키지 않고, 시스템의 내부정보 없이 자율제어 시스템 개발을 용이하게 한다. 그러나 자가치유 개발자가 관리 대상이 되는 시스템을 모델링하기 위해서 내부 정보 없이 외부 자원 환경에 대한 분석을 하는 것은 그 부하가 매우 높다. 이러한 문제를 해결하기 위해서는 관리 대상이 되는 시스템의 'Goal(목표)'를 모델링하여 관리 대상을 명확하게 하는 것이 요구될 수 있다.

5. 공통적인 문제점

관련 연구들을 통해 분석된 것은 전통적인 자율제어 방법론들은 각기 다른 관점에서 독립적으로 구현되어져 왔기 때문에 CPS 환경에 있는 'critical system' 산업에 적용될 수 없는 공통적인 단점이 있다. 따라서 적용될 수 있는 범위 내에서 CPS가 달성되어야 할 '목표'를 잘 설계하면 전통적으로 연구된 '자율제어 방법론'들이 부분적으로 통합되어 관련 산업에 큰 기대효과를 부여할 수 있다.

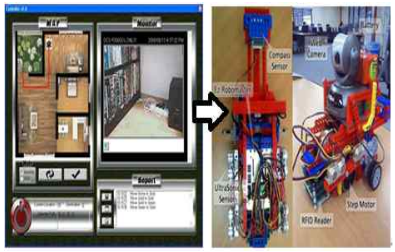

그림 1. '사용자 애플리케이션'과 'ISR'[4]
Fig. 1 User Application & ISR[4]

Ⅲ. 자율제어 시스템 구조 및 프로세스

본 장에서는 제안 사항을 위해 CPS의 환경을 설명하고, 고신뢰 CPS를 위한 자율제어 시스템 개발 프로세스와 자율제어 시스템 구조를 설명한다.

1. 'CPS' 환경

본 논문에서는 선행 연구로 제작된 ISR(Intelligent Service Robot)[4]을 이용한다. ISR은 RFID 기반의 위치 탐색 로봇이다. 이것의 사용자는 거동이 불편한 장애인이나 노약자로 로봇을 이용해서 집안을 탐색할 수 있다. 서론에서 언급된 'CPS'의 의미처럼 우리는 ISR을 '가상 환경(cyber Space)' 과 '물리 환경(physical space)로 나누었다. 그림 1에 표현된 것처럼 ISR의 가상 환경으로는 '사용자 애플리케이션'이고, 물리 환경은 실제 로봇인 'ISR'을 의미한다. 사용자가 '사용자 애플리케이션(가상 환경)'의 명령 버튼이나 드로잉 모드를 통해 'ISR(실제 물리 환경)'에게 위치 이동 명령을 내리면, 로봇은 전송 받은 위치로 이동하고, 사용자는 이동된 로봇으로부터 영상을 확인할 수 있다. 우리는 이러한 환경을 'CPS'라고 가정하였다.

2 자율제어 시스템 프로세스

지난 수년간의 전통적인 자율제어 연구들에 의하면 '기계학습', '코드자동생성', '확률', '추론'과 같은 방법으로 '자율제어'를 위한 '지식'들을 습득하고 확장해왔다. 그러나 그들은 대부분 방대한 컴퓨팅 환경을 다루려고 노력했고 그 결과 후속 연구 방향이 단절되는 현상이 이루어져 왔다. 또한 '자율제어'에 관한 비전만 제시했을 뿐 실제로 어떤 방법이 가장 '자율제어'에 가까운지를 제시하지 못했다. 따라서 본 절에서는 'CPS'를 위한 자율제어 개발 프로세스를 설명하는 것과 동시에

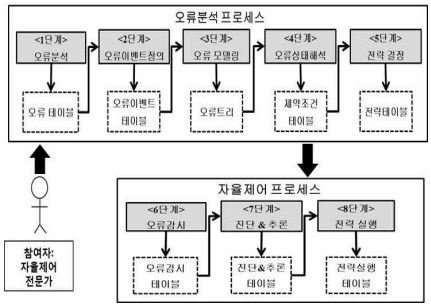

그림 2. 오류분석&자율제어 프로세스
Fig. 2 Process for Autonomic Control

자율제어 시스템을 위해 분석하고 모델링해야 하는 과정들을 알아보고자 한다. 자율제어 시스템의 기초 중에 하나는 '오류 모델'을 명세화 하는 것이다[11-13]. '오류 모델' 없이는 문제 상황에 대해서 스스로를 '자율제어' 할 수 있는 방법이 존재하지 않는다. 따라서 그림 2와 같이 자율제어 시스템의 프로세스는 '오류분석 프로세스'와 '자율제어 프로세스'로 나누어져 개발해야 할 필요가 있다. 본 절에서는 자율제어 시스템 프로세스를 위한 8단계를 상세히 설명한다.

<1단계> 오류 분석(Fault Analysis): 전통적인 연구들[11-13]에 의하면 자율제어 시스템을 개발하기 위해서는 "시스템은 반드시 자신의 문제에 대하여 알아야 한다." 라고 정의하며 '오류 분석'의 중요성을 매우 강조하고 있다. 따라서 1단계에서는 CPS의 정상 동작을 분석하여 발생 가능한 오류들을 추출하고 정의하여 자율제어를 위해 요구되는 데이터로 이용된다. 오류 모델 분석가는 경험에 의해 오류를 분석해야 하기 때문에 관리 대상이 되는 시스템의 정상 동작을 반드시 잘 알아야 한다[11-13]. 분석가에 의해 추출된 오류들은 발생위치에 따라 구분되어 '오류 테이블(fault table)'로 표 1처럼 작성된다. 표 1은 ISR과 '사용자 애플리케이션(UA)'에서 일어날 수 있는 오류에 따라 구분 지었으며, CPS를 구성하는 컴포넌트 별로 오류를 추출하여 정리한다. 특이한 사항은 ISR과 UA에서 공통적으로 일어나는 오류 상태가 존재 한다. 우리는 이러한 문제를 연관성 문제라고 정의한다. 예를 들어 Table1에서 물리환경과 가상환경에 공통적으로 존재하는 '데이터 수신 불가능', '잘못된 이동', '이동 불가'와 같은 것들이 있다. 이러한 연관된 문제에 대한 상태들과 관련된 컴포넌트들 간의 상호작용들을 잘 분석하면 CPS에서 일어나는 문제들

표 1. 오류 테이블
Table 1. Fault Table

구분	컴포넌트	오류 상태
ISR (물리 환경)	컴퍼스 센서	컴퍼스 센서 비정상, 잘못된 이동
	초음파 센서	초음파 센서 비정상, 충돌
	스텝 모터	경로이탈, 잘못된 이동
	블루투스 모듈	데이터 송수신 불가능
	RFID tag	이동불가, 잘못된 이동
	…	…
UA (가상 환경)	카메라	영상 데이터 수신 불가능
	좌표선택 모듈	잘못된 이동, 경로이탈, 이동 불가
	…	…

의 '연관성 분석'이 가능할 것이다(본 연구에서는 '연관성 분석'에 관한 것은 향후 연구로 예정되어 있음).

<2단계> 오류 이벤트 정의(Fault Event Definition): 2단계에서는 오류 이벤트를 정의한다. 시스템은 어떤 오류 상태에서 특정 행동을 수행 하는 경우 다른 오류 상태로 전이하게 된다. 현재의 단계에서는 1단계에서 작성된 '오류 모델'을 기반으로 발생 '원인'과 '결과'를 관계로 정의한다. 또한 원인에서 결과로 전이하게 하는 이벤트를 '오류 이벤트'로 정의하고 최종적으로는 표 2처럼 '오류 이벤트 테이블(Fault Event Table)'을 산출한다. 잘못된 이동(mismove)의 경우 '회전된 각도가 부정확' 하거나 '왼쪽과 오른쪽모터의 회전수가 다름'인 상태에서 이동을 하게 되면 '도착한 좌표가 비정상'인 상태로 전이하게 된다.

<3단계> 오류 모델링(Fault Modeling): 3단계에서는 오류들의 인과 관계에 의한 오류 트리를 모델링 한다. 이전 단계에서 수행된 '오류 이벤트 테이블'을 참조하여 오류의 '원인'과 '결과'를 관계 시킨다(사각형은 '결과' 상태이고 타원은 '원인' 상태가 되도록 구성). 관계된 오류 상태들은 하나의 트리와 같은 형태를 가진다. 오류 트리는 '원인' 과 '결과'의 형태로 구성되어 있으며 '원인'을 통해 '결과'를 도출하는 것을 '진단'이라 하고, '결과'를 통해 '원인'을 도출하는 것을 추론이라고 한다. 그림3은 ISR의 '오류 트리'를 나타내고 있다. 예를 들어, 'rotateAngle==incorrect'가 발생한 경우, 이것은 컴퍼스 센서로 인한 문제(compass sensor == abnormal)이며, 이것이 원인이 되어 잘못된 이동 'arrival location error'를 유발하게

표 2. 오류 이벤트 테이블
Table 2. Fault Event Model

오류 이벤트	원인	시스템 동작	결과
Mismove	rotateAngle ==incorrect	move()	arrival location error
Collision	ultrasonic ==abnormal	move()	arrival location error
Curved path	step motor ==abnormal	move()	arrival location error
Unread tag	RFID tag ==abnormal	move()	arrival location error
Irregular Rotation	compass ==abnormal	rotate()	rotated angle == incorrect
Lack of power	Battery Power < threshold	move()	step motor ==abnormal
Magnetic field	Environment Problem	rotate()	Compass ==abnormal
Broken	Environment Problem	move()	Mission Failure
		rotate()	
...	...	...	...

된다. 진단과 추론의 정확성은 2단계에서 수행된 오류 이벤트에 대한 분석이 세밀할수록 높아진다. 이러한 '진단'과 '추론'의 조합으로 '원인'과 '결과'를 찾아 낼 수 있으며, 이를 기반으로 다양한 인공지능 기법[12]들을 통해 새로운 오류 이벤트가 추가 되고, 트리가 확장될 수 있다.

<4단계> 오류상태해석(Faulty State Interpretation): 4단계에서는 자율제어 시스템이 오류상태를 해석할 수 있는 조건들을 정의하여 표 3과 같이 제약 조건 테이블(Constraint Table)로 작성한다. 제약 조건 테이블(Constraint Table)은 오류 상태를 해석하기 위해 사용되며, 만약 제약 조건 테이블이 없다면 오류 상태를 측정하는 기준이 모호할 것이다. 따라서 자율제어 시스템을 위해 반드시 필요하다. 또한 각 오류가 '정상' 상태에서 '비정상' 상태로 전이 되는 이벤트를 발견하기 위해서도 사용될 수 있다. 예를 들면, ISR의 이동 경로 중 목적 좌표(destination)와 현재 도착 좌표(current_position)가 다를 경우 잘못된 이동(arrived location error)이 발생되었고, 'mismove'이벤트가 발생함을 알 수 있다.

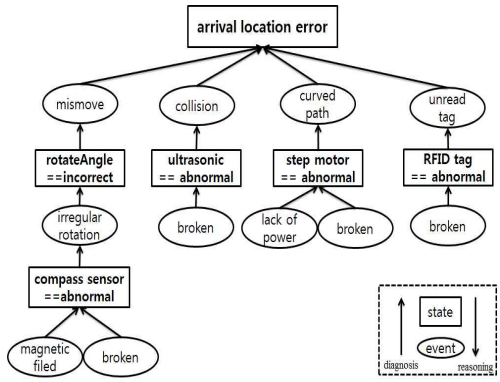

그림 3. '오류트리'
Fig. 3 Fault Tree

표 3. 제약 조건 테이블
Table 3. Constraint Table

오류 상태	제약조건
arrival location error	destination != current_position
ultrasonic == abnormal	changed_distance == 0 OR changed_distance > ULTRA_AVG
step motor == abnormal	(moving) motorSpeed1 ≠ motorSpeed2
RFID tag == abnormal	destination ID ≠ current_position ID
compass sensor == abnormal	changed_angle > COMPASS_AVG
...	...

<5단계> 전략 결정(Maneuver Decision): 전략 결정 단계에서는 진단된 증상에 대한 적응행위자 및 전략을 미리 정의 한다. 표 4는 전략 테이블(Strategy Table)을 작성해 놓은 것이다. 4단계의 산출물인 '제약 조건 테이블'을 통해서 ISR의 오류 종류를 인식하게 되면, 발생한 오류를 해결 하기 위해 적응 전략을 요구하게 된다. 이러한 오류 상태들의 자율제어 전략을 정의 한 것이 바로 전략 테이블(Strategy Table) 이다. 전략 테이블은 ISR의 각 오류 상태에 따른 자율제어 전략과 제어의 행위자를 정의 한다. 예를 들면, 잘못된 이동(arrival location error)이 발생 할 경우, ISR은 '사용자 애플리케이션'에게 '경로 재 연산 요청'을 할 수 있다. 초기에 미리 설정해 놓은 전략들은 다양한 문제 상황들을 통해 진화될 수 있다.

<6단계> 오류 감시 (Fault Monitoring): 오류 감시 단계에서는 이전 단계들에서 생성된 '오류 테이블',

표 4. 전략 테이블
Table 4. Strategy Table

오류 상태	적응 전략	행위자
arrival location error	경로 재연산 요청	ISR
rotated angle == incorrect	측면 초음파 센서로 평행거리 유지	ISR
ultrasonic sensor == abnormal	대체 센서로 채널 변경	ISR
RFID tag == abnormal	Tag 고장 위치 좌표 전송 후 경로 삭제	ISR
compass sensor == abnormal	이동 경로 삭제	사용자 애플리케이션
Broken	경보음	관리자 호출
...	...	...

표 5. 오류 감시 테이블
Table 5. Strategy Table

오류 이벤트	오류 추적 조건	환경 타입
Mismove	destination != current_position	ISR (물리환경)
Collision	changed_distance == 0 OR changed_distance > ULTRA_AVG	ISR (물리환경)
Curved path	(moving) before_angle ≠ current_angle	ISR (물리환경)
...	...	...

표 6. 진단 & 추론 테이블
Table 6. Diagnosis& Reasoning Table

오류이벤트	진단상태	추론상태	전략
Mismove	arrival location error	rotateAngle == incorrect	M1
irregular rotation	rotateAngle == incorrect	compass sensor == incorrect	M2
Collision	arrival location error	ultrasonic ==abnormal	M3
Curved path	arrival location error	step motor == abnormal	M4
Unread tag	arrival location error	RFID tag ==abnormal	M5
broken	compass sensor == incorrect	...	M6
broken	ultrasonic == incorrect	...	M7
...	...	...	...

표 7. 전략 실행 테이블
Table 7. Maneuver Execution Table

전략 식별자	오류이벤트	실행 전략
M1	Mismove	측면 초음파 센서로 평행거리 유지
M2	Collision	대체 센서로 채널 변경
M3	Curved path	파워교체(관리자 호출) 또는 모터 스피드 조절
M4	Unread tag	Tag 고장위치 전송 후 경로 삭제
...	...	...

‘오류 이벤트 테이블’, ‘제약 조건 테이블’과 같은 산출물들을 기반으로 CPS의 비정상적인 문제들을 추적하는 단계이다. 표 5는 ‘오류 감시’ 단계에서 수행하는 내용이다. ‘오류 추적 조건’에 트리거 반응이 일어나면 ‘오류 이벤트’가 일어났음을 결정하고 오류 데이터를 ‘write’한다.

<7단계> 진단&추론 (Diagnosis& Reasoning): 진단&추론 단계에서는 감지한 오류들에 ‘오류 트리(그림 4 참조)’를 참조하여 발생된 ‘오류이벤트’에 대한 ‘진단’과 ‘추론’을 수행 한다. 표 6은 진단&추론 테이블을 나타내고 있다. 주어진 ‘오류 이벤트’에 대한 결과(진단)가 무엇인지, 그 결과에 대한 원인(추론)이 무엇인지를 표현한다. 이러한 데이터를 표현하고 관련 전략 식별자(M1, M2, M3, ..., Mn)들을 연결하는 것이 가능하다. 표 6의 진단 테이블에서 오류이벤트에 대한 추론 상태는 M:N의 매핑이 될 수 있다. 예를 들어 오류이벤트 ‘Mismove’의 추론 상태는 (<3단계>에서 수행된 ‘오류 트리’ 참조) ‘rotateAngle==incorrect’와 ‘compass sensor==abnormal’ 이 될 수 있고, 오류이벤트 ‘broken’의 진단상태와 추론상태도 다양하게 존재할 수 있다. 본 연구에서는 오류이벤트에 대한 M:N 관계의 진단상태 및 추론상태를 판별하기 위해 ‘오류이벤트 기반의 진단 & 추론 알고리즘[12]’을 사용하였다.

<8단계> 전략 실행(Maneuver Execution): 전략 실행 단계에서는 표 6의 M1, M2 와 같은 식별자들을 통해서 전략들 식별하고, 같은 진단상태 이더라도 다른 전략이 실행될 수 있다. 표 7은 실행된 전략들을 나타내고 있다. 또한 ‘오류 이벤트’가 다르고 ‘추론 상태’가 같은 경우도 다양한 전략을 실행할 수 있다.

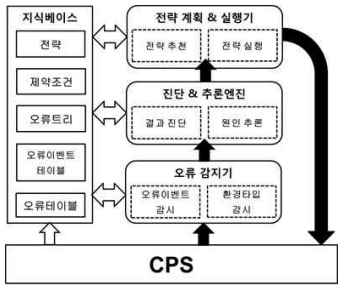

그림 4. 자율제어 시스템 구조
Fig. 4 Autonomic Control Architecture

3. '자율제어 시스템 구조'

자율제어 시스템 구조는 자율제어를 위한 기초 정보인 '지식 베이스'와 1-5단계까지 수행된 산출물들('오류 이벤트' '오류 트리', '제약 조건', '전략')을 기반한 '오류감지기', '진단&추론엔진', 그리고 '전략 계획&실행기'의 네 부분으로 구성되며, 전체적인 구조는 그림4와 같다.

- **지식 베이스 & 오류 감지기**

'지식 베이스'는 자율 제어를 위해 제공 되어야 하는 기본 데이터이다. 이것은 오류 분석가에 의해서 분석된 데이터 이다. '오류 감지기'는 지식 베이스의 기본 정보들을 기반으로 오류 이벤트들을 감지하고, 감지된 오류 정보를 진단&추론엔진에 전달한다(6단계 수행).

- **진단&추론엔진**

진단&추론엔진은 오류 감지기로부터 수신한 오류정보를 오류 테이블 및 오류 트리에 정의된 오류정보와 비교하여 오류의 종류를 분석, 진단과 추론을 한다. 전략의 실행이 필요하다고 판단되는 경우 전략 계획&실행기에게 오류에 대한 진단 및 추론 정보들을 전송한다(7단계 수행).

- **전략 계획&실행기**

전략 계획&실행기는 진단&추론엔진으로부터 수신한 정보를 '전략 실행 테이블'에 기반 하여 적응 전략을 계획하고 실행한다(8단계 수행).

Ⅳ. 실험 및 평가

본 장에서는 실험 및 평가를 위해 우리의 이전 연구인 ISR[4]에 오류 주입 기법[14]을 사용한다. 오류 주입 기법은 시스템의 신뢰성과 강건성을 테스트하는 방법으로 발생 가능한 오류를 주입하여 오류에 대처하는 방법을 분석 하기 위해 사용된다. 표 8은 CPS의 물리환경인 센서와 액츄에이터에 '오류 주입'을 수행한 후, 자율제어의 '오류 감지기'가 추적한 '오류 이벤트'를 나타내고 있다. 자율제어 방법론을 실험하기 위해서 우리는 두 개의 시나리오를 위한 오류주입 실험 절차를 수행했다. 첫째로 '전방 초음파 센서 비정상', 둘째로 '경로 이탈'에 관한 것이다.

표 8. 오류 주입 테이블
Table 8. Fault Injection Table

구분	오류주입	오류이벤트
RFID	RFIDSensorState ==Off	Unread tag
Compass sensor	rotatedAngle ==-1	Irregular Rotation
ultrasonic sensor	ultrasonic SensorState==Off	Collision
left wheel	MotorSpeed ==0	Curved path
right wheel	MotorSpeed ==50	Curved path

1. 오류주입: '초음파 센서 비정상'

정상적으로 이동 중인 'ISR'에 'ultrasonic SensorState==Off' 와 같은 오류를 주입하여 고의적으로 초음파 센서가 정상적인 값을 읽을 수 없도록 한다. 따라서 'ISR'은 장애물을 인식하지 못하는 결과를 초래한다. 이러한 문제를 해결하기 위해서 자율제어 프로세스가 동작하게 되며, 자율제어 시스템의 주요 컴포넌트들인 '오류 감지기', '진단&추론기', '전략 계획&실행기'가 각자의 역할을 통해 자율제어를 한다. 1)'오류 감지기'는 초음파센서가 읽어들이는 값의 변화량을 측정하면서 제약 조건에 위배된 오류 이벤트 'Collision'을 진단&추론엔진에 전송한다. 2)진단&추론엔진은 '오류 트리(그림3참조)'를 통해 'collision'의 발생 원인은 'broken'으로 인한 'ultrasonic sensor==abnormal'을 추론하고, 결과는 'arrival location error'임을 진단한다. 3) 전략 계획&실행기는 진단과 추론된 정보를 기반으로 '대체 센서로 채널 변경(표 7 참고)' 전략을 추천하며 실행한다. 자율제어를 수행한 후 ISR은 정상적으로 장애물을 탐지 하게 된다. 초음파 센서가 전방의 장애물을 정상적으로 인식하는 것을 확인하면 성공을 의미한다. 그림5는 자율제어 시스템이 수행된 기록이다.

```
<AUTONOMIC CONTROL SYSTEM-START>
LOADING…FAULT MODEL…CHECKING
.....
FAULT INJECTION-F001: ultrasonic SensorState==OFF

FAULT DETECTOR
Inspecting…ULTRA_SONIC_SENSOR : <abnormal status> 13:51:41
Fault event: collision

DIGNOSIS&REASONNING ENGINE
Causative event: broken 13:51:46
Causative state: ultrasonic sensor == abnormal 13:51:50
Resultant state: arrival location error 13:51:52

EXECUTOR
Maneuver 1: use alternative a ULTRA_SONIC_SENSOR 13:52:03
Maneuver 2: …
Result: ULTRA_SONIC_SENSOR <normal status> 13:52:10
.....
FAULT INJECTION-F002: motorSpeed == 0
…
FAULT DETECTOR
…
DIGNOSIS&REASONNING ENGINE
…
EXECUTOR
…
<AUTONOMIC CONTROL SYSTEM-END>
```

그림 5. 자율제어 수행 로그
Fig. 5 Autonomic Control Log

2. 오류주입: '스텝 모터 비정상'

직진 이동 중인 'ISR'에 'motor Speed==0' 과 같은 오류를 주입하여 양쪽 모터 중 한 쪽 모터의 회전수를 고의적으로 감소시킨다. 따라서 'ISR'은 정상적인 직진 이동을 불가능 하게 되며 자율제어 프로세스가 요구된다. 1) '오류 감지기'는 양쪽의 모터 스피드의 변화량을 측정하면서 제약 조건에 위배된 오류 이벤트 'Curved path'를 진단&추론 엔진에게 전송한다. 2)진단&추론엔진은 정보를 전송받아 오류 트리(그림3 참조)를 통해 해당 'Curved path'가 발생한 원인은 'lack of power'로 인한 'step motor==abnormal'을 추론하고, 결과는 'arrival location error'임을 진단한다. 3)전략 계획&실행기는 진단과 추론된 정보를 기반으로 '모터의 스피드 조절' 전략과 '파워 교체' 전략을 추천하며 실행한다. 두 가지 전략을 통해 정상상태로의 회복이 가능하면 자율제어 프로세스의 성공을 의미한다.

그림 6은 ISR과 상호작용하는 사용자 애플리케이션을 나타내고 있다. 사용자 애플리케이션의 기능 중에 '오른쪽 상단'에 있는 'State Monitor'는 ISR을 구성하고 있는 컴포넌트들의 문제 상태를 심각한 정도에 따라 색상으로 표시한다. 녹색은 정상 상태, 노란색은 경고 상태, 빨간색은 비정상 상태를 의미한다. 각 컴포넌트를 클릭하면 해당 센서의 오류 발생 정보를 확인할 수 있다.

3. 평가

본 연구에서는 자율제어 시스템의 정량적 평가

그림 6. 사용자 애플리케이션
Fig. 6 Autonomic Control Architecture

를 위해서 주입된 오류의 정확도를 평가 하였다. 표 8의 오류 주입 테이블의 리스트를 추가하여 총 5회에 걸쳐 오류 주입이 이루어졌고, 횟수가 증가할 때마다 10개씩 증가 시켰다. 개별 '오류 탐지율'과 전체 '오류 탐지율'을 측정하기 위해서 (1),(2)와 같이 주어진 함수를 사용하였다.

$$\text{오류탐지율} = \frac{\text{오류탐지개수}}{\text{오류주입 개수}} * 100 \qquad (1)$$

$$\text{전체오류탐지율} = \frac{\sum_{i=0}^{n} \text{오류탐지율}(i)}{\text{실험 빈도수}} \qquad (2)$$

그림 7은 전체 오류 탐지율이 '100%'로 '오류 주입 개수'와 '오류 감지 개수'가 같음을 알 수 있다. 실제로 그림 7의 경우, 오류 주입의 형태는 '물리환경'에 관한 것이다. 즉 'ISR'에 관한 것이다. 따라서 '가상환경'과의 연관성에 관한 '오류 주입'은 이루어지지 않았다. 이러한 경우는 '오류 트리'의 정보가 정확하여 '진단'과 '추론'의 시간이 짧아 신속한 관리자의 호출이나 전략 대응이 가능하다.

그림 8는 1~5회 걸쳐 '오류탐지율'을 산출하여 총92%의 '오류탐지율'을 보였다. 그림 8의 경우, 오류 주입의 형태는 '가상환경'과 '물리환경'과의 연관성 문제와 관련된 것이다. 그림에서도 알 수 있듯이 '오류 주입 개수'가 증가할 때마다 '오류탐지율'이 100%~88%까지 계속적으로 감소하고 있다. 즉, 오류 탐지 개수가 감소하고 있는 것이다. 이 실험은 의도된 물리환경과 가상환경의 연관성에 관한 실험이다. 이것의 의미는 물리환경에 의도된 오류를 주입하고 오류를 관찰 했을 때 가상 환경의 오류도 존재할 수 있음을 의미한다. 의도된 물리 환경의 오류 주입이 없다면, 가상 환경의 오류 존재

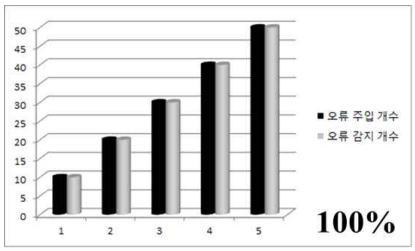

그림 7. '물리환경'의 오류탐지 정확도
Fig. 7 Accuracy for Fault Detection

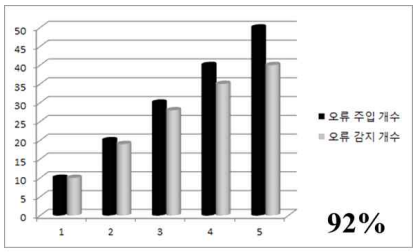

그림 8. 오류탐지 정확도
Fig. 8 Accuracy for Fault Detection

가 없다는 것은 아니다. 즉, 가상환경에 관한 오류 또한 개별적으로 존재할 수 있다(예를 들어, 통신의 문제, 좌표 미설정의 문제 등). 1단계(오류 분석 단계)에서도 언급했듯이 '가상환경'인 '사용자 애플리케이션'과의 연관성 문제로 일어나는 오류 상태로 인해서 자율제어 시스템의 '오류 탐지기'가 '오류 트리'를 해석할 때 충돌을 일으키게 된다. 예를 들어 표 1에서 '이동 불가'라는 오류 상태가 있을 때 이것이 '물리환경'의 문제인지 '가상 환경'의 문제 인지를 탐지하기 어렵다. 이 문제는 단순히 '오류 감지기'의 성능 문제가 아니다. 이것은 CPS의 '관리 목표'를 명확하게 하지 않은 분석가의 문제이다. 따라서 CPS가 달성해야 하는 추가적인 '목표 모델'을 설계해야 한다. '목표모델'과 '오류트리'를 연계하면 92%의 오류 탐지율을 보다 높게 개선하는 것이 가능하다.

그림 9은 '목표모델(직선주행)'과 '오류트리(직선주행 위배)'를 연계한 것이다. 즉, 목표모델은 런타임 시에 CPS가 성취되어야 하는 목표를 의미하는데, 이 목표를 위배 하였을 때 '자율 제어'를 요구하고, 목표에 위배 되지 않도록 전략을 수행할 수 있다. '직선주행' 목표모델의 제약조건(constraint)들을 잎-노드에 연결하여, 연결된 제약조건들이 위배되는 경우, 오류트리에서 '오류이벤트'를 찾아 '진단'과 '추론'을 산출한다. 그림 9과 관련 연구는 '오류탐지율'을 개선하기 위

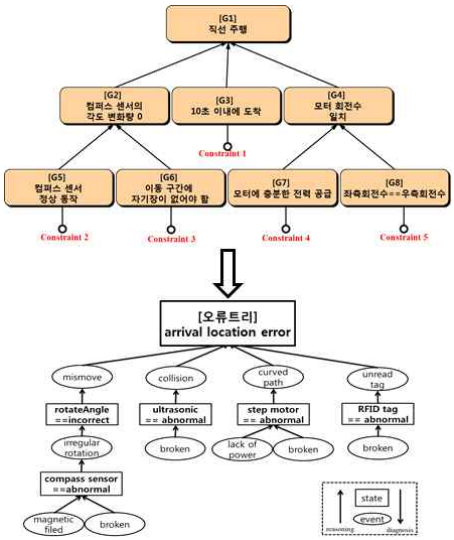

그림 9. 목표모델과 오류트리의 연계
Fig. 9 Connection of Goal Model and Fault Tree

해 현재 진행 중이며, 목표모델링을 위한 도구를 설계 중에 있다.

Ⅵ. 결론

본 연구에서는 자율제어 시스템을 위해 1)오류 분석, 2)오류 이벤트 분석, 3)오류 모델링, 4)오류 상태 해석, 5)전략 결정, 6)오류 감시, 7)진단&추론, 8)전략 실행 단계들을 제안하였다. 제안 사항들을 통해서 'CPS'가 직면하는 오류 상황들을 분석하고, 설계하여 오류를 탐지하고 그것에 대응하는 전략 적용이 가능했다. 그러나 '물리환경'과 '가상환경'의 연관성에 관한 '오류 탐지'의 정확도를 개선해야할 필요가 있다. 이것은 오류 분석 및 대응 전략의 과정을 분석가가 경험에 의해서 모든 것을 수행해야 함을 의미하는 것이 아니다. CPS가 달성해야 하는 목표를 추상화하고, 이를 '목표모델'로 표현하여 '오류트리'와 연계하면 현재 연구의 문제를 개선할 수 있다. 따라서 현재 연구와 '목표모델'과의 연계를 통해 오류의 우선순위를 정하여 세분화된 문제 결정 수준이 분류 되도록 향후 연구에서 다를 예정이다.

References

[1] X. Li, C. Qiao, X. Yu, A. Wagh, R. Sudhaakar, S. Addepalli, "Toward Effective Service Scheduling for Human Drivers in Vehicular Cyber-Physical Systems," IEEE Transactions On Parallel and Distributed Systems, Vol. 23, No. 9, pp.1775-1789, 2012.

[2] J.H. Lee, J.M. Park, G.J. Yoo, E.S. Lee "Goal-Based Automated Code Generation in Self-Adaptive System," Journal of Computer Science and Technology, Vol. 25, No. 6, pp.1118-1129, 2010.

[3] I. Chun, J. Park, W. Kim, W. Kang, H. Lee, S. Park, "Autonomic Computing Technologies for Cyber-Physical Systems," Proceedings of Interational Conference on Advanced Communication Technology, 2010.

[4] J.M. Park, I.G. Chun, H.Y. Lee, W.T. Kim, and Eunseok Lee. "Intelligent Service Robot and Application operating In Cyber-Physical Environment," Lecture Notes in Electrical Engineering, Vol. 181, No. 01, pp.301-310, 2012.

[5] J.M. Park, S.J. Kang, I.G. Chun, W.T. Kim "An Approach to Generating Goal Model for Cyber-Physical Systems," Proceeding of International Symposium on Embedded Technology, 2013.

[6] Cisco Unified Communications Manager Adapter, http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=%2Fcom.ibm.itim_pim.doc%2Fciscouni%2Finstall_config%2Fciscouni_html_mstr.htm

[7] R. sterritt, D.F. Bantz, "Personal Autonomic Computing Reflex Reactions and Self-healing," IEEE Transactions on Systems Man and Cybernetics - PART C: applications and reviews, Vol. 36, No. 3, pp.219-228, 2007.

[8] J.M. Park, H.S. Youn, E.S. Lee, "An Autonomic Code Generation for Self-Healing," Journal of Information Science Engineering, Vol. 25, No. 6, pp.1753-1781, 2009.

[9] D. Garlan, S.W. Cheng, A.C. Huang, B. Schmer P. Steenkiste, "Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure," IEEE Computer, Vol. 37, No. 10, pp.46-54, 2004.

[10] D. Garlan, B. Schmerl, "Model-based adaptation for self-healing systems", Proceedings of Workshop on Self-Healing Systems, pp.27-32, 2002.

[11] P. Koopman, "Elements of the Self-healing System Problem Space," Proceedings of ICSE WAD, 2003.

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

[13] http://www.ibm.com/developerworks/tivoli/autonomic.html

[14] L. Cheung, "Early Prediction of Software Component Reliability," In Proceedinsgs of ICSE, 2008.

저 자 소 개

박 정 민

2003년 한국산업기술대학교 컴퓨터공학과 학사.
2005년 성균관대학교 컴퓨터공학과 석사.
2009년 성균관대학교 컴퓨터공학과 박사.
현재, 한국전자통신연구원 선임연구원.
관심분야: CPS, M&S, 자율제어
Email: jmpark23@etri.re.kr

강 성 주

2003년 한양대학교 전자과 학사.
2005년 한양대학교 전자과 석사.
2010년 한국과학기술원, 카네기멜론대 컴퓨터과학과 석사.
현재, 한국전자통신연구원 선임연구원.
관심분야: CPS, M&S, 자율제어
Email: sjkang@etri.re.kr

전 인 걸

1996년 성균관대학교 컴퓨터공학과 학사.
1998년 성균관대학교 컴퓨터공학과 석사.
2010년 성균관대학교 컴퓨터공학과 박사.
현재, 한국전자통신연구원 선임연구원.
관심분야: CPS, M&S, 자율제어
Email: igchun@etri.re.kr

김 원 태

1994년 한양대학교 전자과 학사.
1996년 한양대학교 전자과 석사.
2000년 한양대학교 전자과 박사.
현재, 한국전자통신연구원 책임연구원.
관심분야: CPS, M&S, 통신미들웨어
Email: wtkim@etri.re.kr