논문 2012-07-33

CPS를 위한 모델 기반 자율 컴퓨팅 프레임워크

(Model-based Autonomic Computing Framework for Cyber-Physical Systems)

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

Abstract : In this paper, we present the model-based autonomic computing framework for a cyber-physical system which provides a self-management and a self-adaptation characteristics. A development process using this framework consists of two phases: a design phase in which a developer models faults, normal status constrains, and goals of the CPS, and an operational phase in which an autonomic computing engine operates monitor-analysis-plan-execute(MAPE) cycle for managed resources of the CPS. We design a hierachical architecture for autonomic computing engines and adopt the Model Reference Adaptive Control(MRAC) as a basic feedback loop model to separate goals and resource management. According to the GroundVehicle example, we demonstrate the effectiveness of the framework.

Keywords : Cyber-physical systems, Model-based autonomic computing, CPS modeling, Autonomic computing framework

Ⅰ. 서론

컴퓨팅 기술은 시대의 변화에 따라 새로운 패러다임으로의 전환을 요구한다. 최근 기업의 비즈니스는 상품 지향적(good-centric)에서 서비스 지향적(service-oriented)으로 변화하고 있고 [1], 통신, 뱅킹, 쇼핑 등 기존의 일상 서비스들은 발전된 정보 기술 인프라와의 결합으로 인간과 인간의 상호작용 없이 이루어지고 있다. 또한 컴퓨팅 장치와 각종 시스템들의 폭발적인 확산으로, 소프트웨어 시스템들은 여러 시스템 자원들을 결합하고 있다. 이들 소프트웨어 시스템들은 다양한 시스템들이 연동되어 공통 목적을 수행하는 형태인 시스템들의 시스템(systems of systems) [2]을 넘어서, 초대규모 시스템(ultra-large-scale systems) [3], 사이버-물리 시스템(Cyber-Physical System, CPS)과 같은 형태로 진화하고 있다.

사이버-물리 시스템은 초대규모의 임베디드 시스템들(embedded systems)과 물리 시스템들(physical systems)이 연계되어 하나의 서비스를 제공하는 컴퓨팅 시스템이다 [4]. CPS에서는 다수의 이종 시스템들이 네트워크를 이루고, 각 시스템이 접해있는 복잡한 물리 환경과 상호 작용하기 때문에 신뢰성을 보장해야 한다. CPS의 고신뢰성을 보장하기 위해서는 시스템 설계 단계부터 물리 영역과 컴퓨팅 영역이 통합적으로 고려된 모델링과 시뮬레이션(M&S) 개발 기법이 필수적이다 [5].

또한, 분산된 다수의 시스템이 연계되어 서비스를 제공하는 CPS 운영 관점에서 볼 때, 시스템 관리의 복잡도는 종종 인간의 수용 능력을 넘어서며 [6], CPS 운영 시 주변 물리 환경은 극도로 복잡하고 지속적으로 변화하므로, 개발 단계에서는 시스템이 직면할 많은 문제 상황을 예측할 수 없다 [7]. 이와 같이 관리의 복잡성과 운영 환경의 불확실성에 대응하기 위해, 자가 관리(self-management)와

* 교신저자(Corresponding Author)
논문접수 : 2012. 08. 10., 채택확정 : 2012. 09. 06.
강성주, 전인걸, 박정민, 김원태: 한국전자통신연구원 임베디드소프트웨어연구부 CPS 연구팀
※ 본 연구는 지식경제부 및 한국산업기술평가관리원의 산업원천기술개발사업의 일환으로 수행하였음. [10035708, “고신뢰 자율제어 SW를 위한 CPS(Cyber-Physical Systems) 핵심 기술 개발”]

ISSN : 1975-5066

자가 적응(self-adaptation) 특성을 갖춘 소프트웨어 시스템의 개발과 운용, 관리를 위한 방법이 필요하다 [6]. 시스템에 자가 관리와 자가 적응 특성을 부여하는 방법은 시스템에 피드백 제어 루프(feedback control loop)를 갖추는 것으로, 이는 자율 컴퓨팅(Autonomic Computing)의 기초가 된다 [8].

자율 컴퓨팅은 IBM에서 제안한 컴퓨팅 패러다임으로, 기업용 애플리케이션 시스템 관리 시 기존에는 사람이 담당하던 관리 업무를 시스템이 자율적으로 수행하도록 하기 위해 고안되었다. IBM의 자율 컴퓨팅 참조 아키텍처에 따르면, 자율 컴퓨팅 시스템은 시스템의 자가 설정(self-configuring), 자가 치유(self-healing), 자가 최적화(self-optimizing), 그리고 자가 보호(self-protecting)를 위해 관리하고자 하는 시스템 자원(서버, 네트워크, DB, 애플리케이션 등)의 상태를 지속적으로 모니터(Monitor)하고, 이상 상태 발생 시 그 원인을 분석(Analyze)하고, 문제에 대한 해결 방법을 계획(Plan)한 후, 시스템에서 실행(Execute)한다 [8].

본 논문에서는 자율 컴퓨팅 참조 아키텍처를 기반으로 CPS를 위한 모델 기반 자율 컴퓨팅 프레임워크를 제안한다. 제안된 프레임워크는 CPS 개발 프로세스에 따라 모델링된 시스템의 목표 모델(goal model), 결함 모델(fault model) 및 정상 상태 조건(normal status constraints, NSC)에 기반하여, CPS 응용 프로그램의 상태 모니터링 및 시스템 목표 모델의 달성도에 따라 선택적으로 적응 정책을 수행하는 자율 컴퓨팅 관리자 구조를 제시하는 것을 핵심으로 한다.

본 논문은 다음과 같이 구성된다. 2장에서는 자율 컴퓨팅에 관련된 연구 동향을 기술하고, 3장에서는 IBM의 자율 컴퓨팅 참조 모델을 소개한다. 4장과 5장에서는 CPS 개발 프로세스를 통한 모델 기반 자율 제어 프레임워크의 설계와 운용 관점을 각각 설명한다. 6장에서는 적용 사례를 소개하고, 결론을 기술한다.

Ⅱ. 관련 연구

자율 컴퓨팅은 관리자가 부여한 관리 목적 아래 스스로의 상태를 진단하고 관리하는 컴퓨팅 시스템을 의미한다 [9]. IBM은 자율 컴퓨팅 개념을 제안하고, 참조 아키텍처 및 적용 시나리오를 제시했다 [8]. IBM은 또한 Blue Gene/L과 eLiza 프로젝트를 통해 시스템 컴포넌트의 예기치 않은 변경이나 실패에 자동적으로 대응하는 메인프레임에 대해 연구하였다 [10]. 하지만 이는 서버 관리 관점에서 사람의 역할을 시스템에 부여한 것으로, 물리 환경의 변화와 영향을 고려해야 하는 CPS에는 적용하기 쉽지 않다.

Cheng과 Garlan은 아키텍처 기반의 모델 기반 자가 적응 프레임워크인 Rainbow 프레임워크를 제안하였다 [11]. Rainbow는 아키텍처 설계 단계에서 문제 해결 전략 등을 모델링하고, 운용 중 문제 발생 시 적응 엔진을 통해 자가 치유한다. 이는 대규모 데이터 센터와, 비디오 컨퍼런싱 등의 서비스에 적용되었다. 그러나 시스템 적응 평가 시 서비스 환경에서의 제한된 지표(응답속도, 전송률)로만 평가되어, 시스템의 다양한 목표에 대해 만족도 평가를 요구하는 CPS에 적용하기에 적합하지 않다.

Hong과 Youn은 CPS를 위한 자율 컴퓨팅 프레임워크의 구조를 제안하였다 [12]. 제안된 프레임워크는 CPS에서 관리하고자 하는 애플리케이션 모듈에 대해 MAPE(Monitor Analyze Plan Execute) 과정과 적응도 평가 방법을 통해 자가 치유 기능을 수행하는 자율 컴퓨팅 엔진을 제안하였다. 그러나 제안 방법은 센서 등 물리 환경과 접해있는 하드웨어 자원 관리에 대한 고려 없이 애플리케이션 모듈의 상태 관리만을 수행해, 시스템의 목표 달성도의 정확한 평가 및 하드웨어와 소프트웨어 수준의 다양한 적응 정책을 수행하기 어렵다.

따라서 본 논문은 이러한 문제를 해결하기 위해 모델 기반 자율컴퓨팅 프레임워크를 제안한다. 제안된 프레임워크는 CPS의 목표 및 결함 모델링 기능과 작성된 모델을 기반으로 하는 계층화된 자율 컴퓨팅 엔진 구조를 특징으로 한다. 계층화된 자율 컴퓨팅 엔진은 모델링된 자원(소프트웨어 컴포넌트 및 하드웨어)의 오류 분석 및 결함 해결을 통한 자가 관리 기능과 CPS의 목표 기반의 만족도 달성을 위한 자가 적응 기능을 제공한다.

Ⅲ. 자율 컴퓨팅 참조 모델

본 장은 우리의 제안 프레임워크의 기반이 되는 IBM의 자율 컴퓨팅 참조 모델을 설명한다. IBM의 자율 컴퓨팅 참조 모델 [6, 8]은 서버 시스템의 자가 관리를 위한 모델로, 그림 1과 같이 계층적 구조로 되어 있다.

(1) 관리 자원(Managed Resource): 자율 컴퓨팅이 적용될 대상 시스템 내의 관리될 수 있는 소프트웨어 및 하드웨어 등의 자원(Managed Resources)이다.
(2) 터치 포인트(Touchpoint): 각 자원의 상태 값의 획득 및 정상 상태를 유지하기 위한 정책 실행

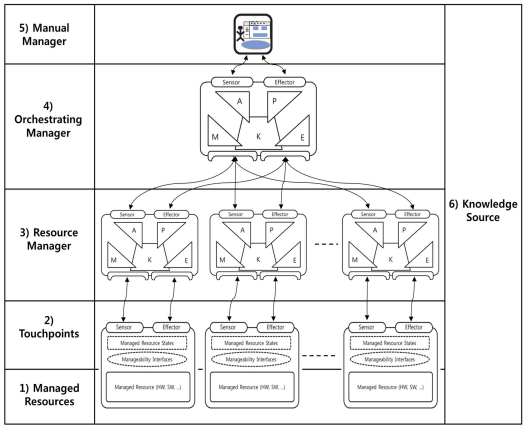

그림 1. 자율 컴퓨팅 참조 모델 [6, 8]
Fig. 1. Autonomic Computing Reference Architecture

을 위한 인터페이스이다. 예를 들어, CPU 자원의 상태 값은 BUSY, IDLE과 같은 상태 지표나 사용량(%)과 같은 측정 지표를 사용할 수 있다. 이와 같은 자원의 정보를 획득하기 위한 센서(sensor) 기능으로는 get함수, 로그 파일 저장, 이벤트에 대한 알림 기능(Notification) 등이 있으며, 자원의 정보를 변경하기 위한 이펙터(effector) 기능으로는 set함수, 설정 파일, 시스템 명령어 호출 등이 있다.

(3) 자원 관리자(Resource Manager): 터치포인트의 센서, 이펙터 기능들을 이용해 관리 자원의 자가 관리(자가 설정, 자가 치유, 자가 최적화, 자가 보호)를 수행하는 자율 컴퓨팅 관리자이다.

(4) 조정 관리자(Orchestrating Manager): 시스템 관리자가 지정하는 명령을 받아 하위 자원 관리자들을 이용해 해당 명령의 목표(시스템 차원의 관리)를 달성하는 것을 목적으로 한다.

(5) 매뉴얼 관리자(Manual Manager): 시스템은 조정/자원 관리자를 통해 자가 관리를 수행하지만, 시스템 관리자(인간)는 시스템의 상태 정보 수집과 변경을 위한 방법을 확보해야 한다. 매뉴얼 관리자는 시스템 관리자를 위한 인터페이스를 제공한다.

(6) 지식 저장소(Knowledge Source): 시스템이 자가 관리를 위한 각종 정보(시스템 이상 정보 및 해결 프로시져, 시스템 재설정 방법 등)를 저장한다.

본 장에서는 자율 컴퓨팅 참조 아키텍처를 설명하였다. 본 아키텍처를 기반으로 임의의 시스템에 자율 컴퓨팅 기능을 적용하기 위해서 다음과 같은 몇 가지 요구사항이 존재한다.

- 시스템에서 관리할 자원의 종류가 선정되어야 한다.
- 자원에서 관리할 상태에 대한 정의 및 상태 정보 수집과 변경을 위한 인터페이스가 확보되어야 한다.
- 관리 자원의 정보 및 발생할 문제의 종류와 해결 방법이 제공되어야 한다.
- 관리 자원의 자가 관리를 위한 자율 컴퓨팅 관리자와 시스템 전체의 상황을 인지하여 시스템이 목표대로 수행되고 있는지 평가 및 목표에 맞게 수행되도록 정책 실행을 담당할 자율 컴퓨팅 관리자의 형태로 계층화되어야 한다.

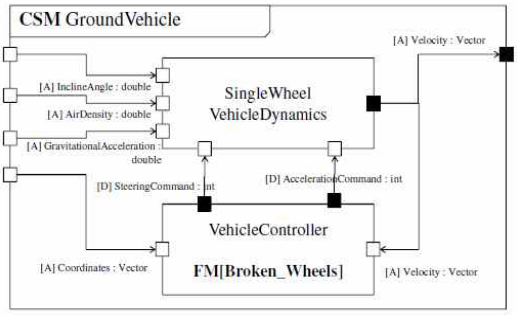

그림 2. CPS 구조 모델 (CSM)
Fig. 2. CPS Structure Model (CSM)

이와 같은 요구사항을 기반으로 4장에서는 CPS 모델에 기반한 자율 제어 프레임워크에 대해 설계와 운용 관점에서 각각 설명한다.

IV. 자율 컴퓨팅을 위한 CPS 모델링

ETRI에서 개발한 CPS 모델링 도구인 EcoPOD는 GUI 기반의 CPS 모델링 및 개발 도구로, 개발하고자 하는 CPS의 구조와 행위를 모델링할 수 있다. EcoPOD로 모델링 된 CPS의 구조 모델과 행위 모델은 ECML(ETRI CPS Modeling Language) 언어로 표현된다 [13]. 본 모델링 도구를 통해 CPS 설계 시 시스템의 자율 컴퓨팅을 위한 결함, 오류 및 목표 모델링을 할 수 있다.

1. 결함 모델링

결함 모델링은 CPS의 구조 모델에서 발생할 수

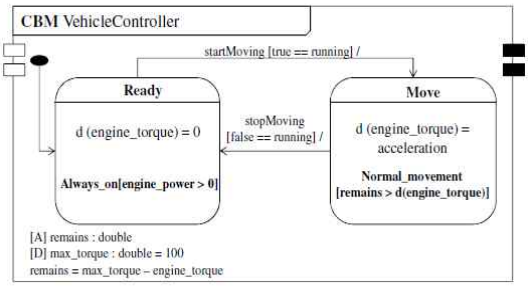

그림 3. CPS 행위 모델 (CBM)
Fig. 3. CPS Behavior Model (CBM)

있는 결함을 기술하는 것이다. 그림 2는 CPS의 구조 모델(CPS structure model, CSM)과 해당 구조에서 발생할 수 있는 잠재 결함(fault model, FM)을 표현한 것이다.

그림 2의 CSM에는 자가 적응 로봇 차량(GroundVehicle)의 바퀴의 역학적 구조를 모델링한 블록(SingleWheel)과 바퀴를 제어하는 블록(VehicleController)이 표현되어 있다. 바퀴 제어 블록에는 바퀴가 고장나는 모델:FM[Broken_Wheels]이 표현되며, 개발자는 정의된 결함 모델에 대해 적응 정책 및 치유코드를 작성한다.

2. 오류 모델링

오류 모델링은 CPS의 구조 모델의 행위 도중 발생할 수 있는 이상 상태를 기술하는 것이다. 그림 3은 CPS의 행위 모델(CPS Behavior Model, CBM)과 해당 행위 과정에 만족되어야 할 변수의 조건인 정상 상태 조건(Normal Status Constraints, NSC)을 표현한 것이다.

그림 3의 CBM은 그림 2의 'VehicleController'의 하위 모델로, 바퀴 제어 모델에서 표현될 수 있는 행위들을 상태 모델로 표현한 것이다. 예제에서는 대기 상태(Ready)와 이동 상태(Move)가 존재한다. 대기 상태는 차량의 바퀴가 움직이지는 않지만 엔진은 켜져 있는 상태이므로, 현 상태의 NSC는 Always_on [engine_power > 0]와 같이 표현된다. 이동 상태에서의 NSC은 Normal_movement [remains > d(engine_torque)], 즉 엔진의 토크가 일정 수준 이하에서 유지될 때로 정의한다.

3. 목표 모델링

목표 모델링을 통해 정의되는 목표는 개발하고

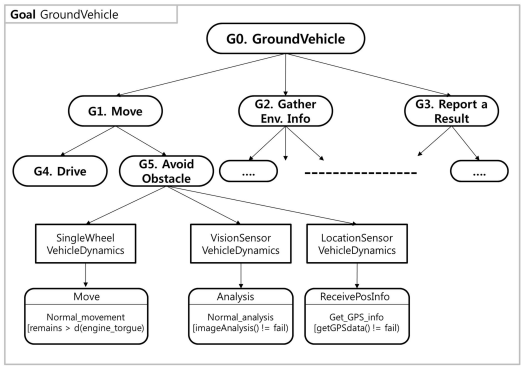

그림 4. CPS 목표 모델
Fig. 4. CPS Goal Model (CGM)

자 하는 CPS가 운용 중 만족해야 할 기능이나 성능에 대한 요구사항을 표현한 것이다. CPS의 목표 모델링 단계에서는 목표에 대한 정의, 해당 목표를 만족시키기 위해 연관된 CPS의 구조 모델(CSM) 및 해당 구조 모델의 행위 모델(CBM)에 대한 정상 상태를 트리 형태로 작성한다. 그림 4는 차량(GroundVehicle)에 대해 정의된 목표의 일부를 보여준다.

차량은 이동(G1), 환경 정보 획득(G2), 결과 보고(G3)의 목표를 가지고 있으며, 각각의 목표에는 하위 목표가 존재한다. 이동(G1) 목표 하위에는 운전(G4)과 장애물 회피(G5)의 목표가 있다. 장애물 회피 목표 달성에는 3개의 구조 모델(CSM) - 바퀴(Single Wheel), 비전 센서(Vision Sensor), 위치 센서(LocationSensor)가 연관되어 있다. 각 구조 모델에는 행위 모델(CBM)이 정의되어 있으며, 행위 모델 내부에는 해당 구조가 가질 수 있는 상태가 정의되어 있다. 그림 3의 바퀴(SingleWheel) 모델의 제어를 담당하는 'VehicleController'의 경우 'Ready' 상태와 'Move'상태가 존재한다.

결론적으로, 장애물 회피(G5)라는 목표 달성 여부의 평가를 위해서는 목표와 연관된 하위 구조모델의 상태(바퀴는 'Move' 상태, 비전 센서는 'Analysis' 상태, 위치 센서는 'ReceivePosInfo' 상태)를 만족하는지 점검해야 하고, 각 상태에 정의되어 있는 정상 상태 조건(NSC)을 만족하는지를 점검해야 한다.

V. 모델 기반 자율 컴퓨팅 프레임워크

본 장에서는 4장에서 소개된 자율 컴퓨팅을 위

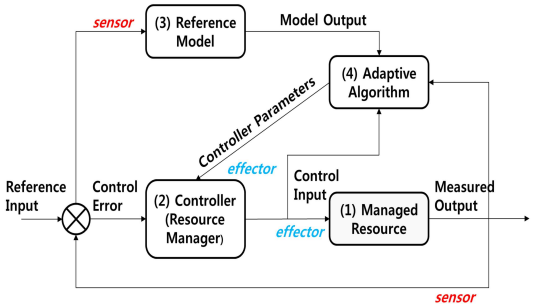

그림 5. 모델 참조 적응 제어 (MRAC)
Fig. 5. Model Reference Adaptive Control (MRAC)

한 CPS 모델들을 기반으로 CPS 운용 시 시스템의 자가 관리, 자가 적응 등을 수행하는 모델 기반 자율 컴퓨팅 프레임워크에 대해 설명한다.

1. MRAC 기반의 피드백 제어 루프

본 연구에서 제시하는 모델 기반 자율 컴퓨팅 프레임워크는 그림 5의 모델 참조 적응 제어(Model Reference Adaptive Control, MRAC) 루프를 기본 피드백 루프 모델로 한다 [14]. MRAC는 기본적으로 다음과 같이 네 개의 구성 요소가 연동되어 동작한다.

(1) 관리 자원(Managed Resource): 관리 자원은 시스템 또는 시스템에서 관리되고자 하는 모든 자원을 의미한다.
(2) 컨트롤러(Controller): 컨트롤러는 관리 자원의 출력 값을 'sensor'를 통해 모니터하고, 이상 범위의 출력 값이 나타날 경우 'effector'를 통해 입력 값을 적절히 조절하여, 정상 범위의 출력 값이 나오도록 한다.
(3) 참조 모델(Reference Model): 참조 모델은 관리 자원의 출력 값에 대한 이론적인 모델로 관리 자원의 출력 값과 이론값과의 차이가 발생할 경우 적응 알고리즘(Adaptive Algorithm)을 통해 컨트롤러를 제어할 수 있다.
(4) 적응 알고리즘(Adaptive Algorithm): 컨트롤러를 제어할 수 있도록 시스템에서 수행될 수 있는 방법들이다.

이에 따라 MRAC를 CPS에 적용하면 관리 자원은 자율 컴퓨팅 프레임워크를 통해 관리되는 CPS 애플리케이션 프로세스와 CPS에 장착된 하드웨어

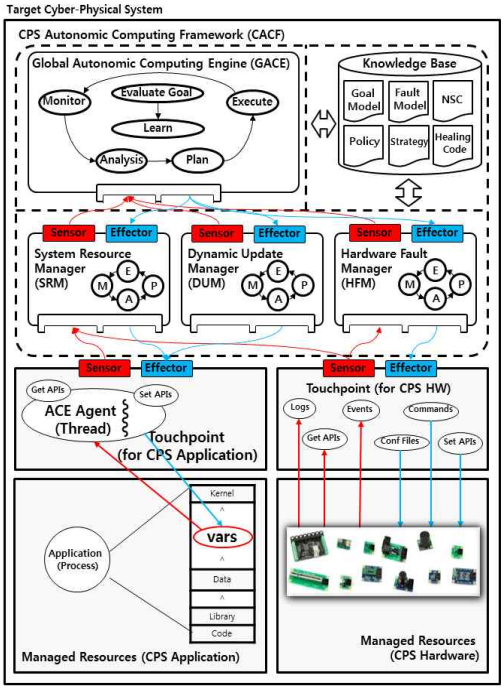

그림 6. CPS 자율 컴퓨팅 프레임워크 (CACF)
Fig. 6. CPS Autonomic Computing Framework (CACF)

가 이에 해당된다. 컨트롤러는 CPS 자원 각각의 상태 값을 획득하고, 상태가 원하는 상태에서 벗어날 경우 이를 변경하여 자가 관리를 수행한다.

컨트롤러는 기본적으로 관리 자원으로부터 측정된 출력 값이 모델링 단계에서 정의된 정상 상태 조건(NSC)의 범위에 있는지 점검한다. 하지만 때로는 이러한 조건이 변경되어야 할 경우가 있다. 'GroundVehicle'의 예에서, 차량의 바퀴 중 하나가 고장나서 차량 전체의 속력이 원하는 목표 수준에 도달하지 못하는 상황을 가정하면, 엔진 전체의 출력을 정상 상태 조건에 정의한 수준 이상으로 높여서 고장 나지 않은 나머지 바퀴들로 차량을 운행해야 한다. 이와 같은 방법으로 그림 5의 참조 모델은 CPS 전체 목표를 관리하며, 시스템의 기능이나 성능이 목표에 항상 부합하도록, 컨트롤러에 적응 알고리즘을 입력하는 역할을 담당한다. 적응 알고리즘은 대상 CPS의 목표를 만족하도록 컨트롤러를 제어할 수 있는 방법들로 그림 1의 지식 저장소에 저장되는 적응 정책들이다. 예를 들어 소프트웨어 시스템에서 제공할 수 있는 다양한 기법(재부팅, 소프

트웨어 업데이트 등)들이 이에 해당한다.

본 절에서는 참조 모델과 적응 알고리즘을 통해 컨트롤러를 제어하는 MRAC 기반의 피드백 제어 루프를 CPS에 적용하였다. 이를 통해 CPS가 내부 상태의 변화나 외부 물리 환경의 변화에 자가 적응할 수 있다.

2. 계층화된 자율 컴퓨팅 엔진

MRAC 기반의 피드백 제어 루프는 그림 6의 계층화된 자율 컴퓨팅 엔진구조를 갖는 CPS 자율 컴퓨팅 프레임워크(CPS Autonomic Computing Framework, CACF)와 CPS 관리 자원에 의해 구체화된다.

(1) CPS 자율 컴퓨팅 프레임워크(CACF): CACF는 목표 모델을 기반으로 자가 적응을 수행하는 전역 자율 컴퓨팅 엔진(Global Autonomic Computing Engine, GACE)과 GACE의 요청에 따라 관리 자원의 상태를 점검 및 변경하는 중간 관리자인 시스템 자원 관리자(System Resource Manager, SRM), 동적 업데이트 관리자(Dynamic Update Manager, DUM) 및 하드웨어 결함 관리자(Hardware Fault Manager, HFM), 그리고 CPS 모델링 단계에서 정의된 목표 모델, 결함 모델, NSC, 치유 코드 등이 저장되어 있는 지식 베이스(Knowledge Base)로 구성된다.

(2) CPS 관리 자원과 터치 포인트: CPS의 관리 자원은 크게 CPS 애플리케이션과 CPS 하드웨어로 나뉜다. CPS 애플리케이션은 CPS 개발자가 EcoPOD로 CPS의 구조(CSM), 행위(CBM) 등을 모델링한 후 해당 모델을 실제 구현 및 실행한 프로세스이다. CPS 하드웨어 자원은 CPS 내부 상태에 관한 것(CPU, 메모리 등)과 외부 환경에 관한 것(GPS, 자이로센서 등)등 매우 다양하며, CPS 하드웨어를 위한 터치포인트는 이들의 상태 값을 획득하고 변경하기 위해 사용된다.

전역 자율 컴퓨팅 엔진(GACE)은 CPS가 동작하면서 목표(예를 들어, 4장의 '장애물 회피')가 만족되도록 시스템을 유지한다. 이를 위해 GACE는 목표에 대한 주기적인 평가(Evaluate Goal), 즉 현재 시점의 목표와 관련된 하위 구조/행위 모델 및 CPS 하드웨어의 정상 상태에 대해 MAPE 과정을 수행한다. GACE는 MAPE 과정 수행 시 하위 중간 관리자를 이용한다.

그림 6의 중간에 표현된 SRM, DUM, HFM은 개별적인 MAPE 과정을 통해 관리 자원의 자가 관리를 수행한다. 첫 번째로 시스템 자원 관리자(SRM)는 CPS 애플리케이션의 상태를 관리한다. CPS의 정상 상태 판단을 위해서는 CPS 모델링 단계에서 정의된 NSC가 이용된다. CPS 애플리케이션을 위한 터치포인트인 ACE 에이전트는 SRM이 NSC 관련 변수값(vars)을 CPS 애플리케이션 프로세스로부터 획득하거나 변경하기 위한 Get/Set API를 제공한다.

두 번째로 동적 업데이트 관리자(DUM)는 CPS 애플리케이션의 버전을 관리한다. 외부 업데이트 서버와의 통신을 통해 애플리케이션의 최신 버전 여부를 점검하고, 업데이트가 존재할 때 시스템의 중단 없이 업데이트 정책을 수행한다. DUM을 통한 동적 업데이트가 필요한 이유는, 주변 물리 환경이 극도로 복잡하고 지속적으로 변화하는 CPS의 특성상 개발 단계에서는 시스템이 직면할 많은 문제 상황을 예측할 수 없기 때문이다.

세 번째로 하드웨어 결함 관리자(HFM)는 CPS 하드웨어를 관리한다. HFM은 지속적인 하드웨어의 상태 값 점검을 통해 오류 발생 여부를 점검하고, 오류 발생 시 자가 치유를 수행함과 동시에 해당 정보를 GACE에 보고하는 목적을 수행한다. 왜냐하면 중대한 시스템(critical system)의 경우 하드웨어의 고장은 치명적 결과로 연결될 수 있기 때문이며, 하드웨어 장치의 고장 발생 시 대처 방안 결정은 상위 관리자(GACE)에서 할 수 있기 때문이다.

5장의 'GroundVehicle' 고장의 예시에서 보듯 계층화 된 자율 컴퓨팅 관리자간의 상태 판단 기준과 정책 실행의 우선순위는 상위 관리자인 GACE에 있다. 왜냐하면 CPS의 목표를 통해 더 많은 관리 자원의 정보를 판단할 수 있기 때문이다. 이와 같은 계층 구조를 통해 CPS 자율 컴퓨팅 프레임워크는 CPS 전반에 걸쳐 자가 치유 및 자가 적응 정책을 수행할 수 있다.

Ⅵ. 구현 및 적용 사례

본 논문에 소개된 자가 적응 차량(Ground Vehicle)은 ETRI CPS 모델링 도구인 EcoPOD로 모델링되고 리눅스 상에 운용되도록 개발된 차량 로봇이다. 자가 적응 차량의 구조 모델(CSM), 행위 모델(CBM), 결함 모델(FM), 오류 모델(NSC), 및 목표 모델(CGM)이 각각 ECML 언어로 표현되었으

표 1. 정상 상태 조건
Table 1. Normal Status Constraints

Category	NSC name	CBM/state	Condition
Move	Move_forward	VehicleController/ Tracking/Forward	Rotated_angle = 0
Turn	Normal_rot	VehicleController/ Tracking/Forward	Compass_sensor = normal
Power	Battery_remain	VehicleController/ Ready/WaitComm	Power_remain > threshold
...	...	...	...
Move	Normal_mov	VehicleController/ Tracking/Forward	Current_location = desired_loc
Sensor	**Obj_recognition**	**VehicleController/ Tracking/Sensing Obj**	**Detected_Object = normal**

표 2. 결함 모델
Table 2. Fault Model

FM name	CSM	substructure	Status
BrokenMotor	GroundVehicle	Right-side Motor	Revolution < threshold
WheelBroken	GroundVehicle	Location Sensor	getGPSdata() = fail
...	...	...	...
UnknownObj	**GroundVehicle**	**Vision Sensor**	**imageAnalysis() = fail**

며, 모델들이 저장되는 지식베이스는 SQLite 데이터베이스가 사용되었다. 자가 적응 차량의 자율 컴퓨팅 적용 시나리오는 ETRI CPS 시뮬레이션 도구인 EcoSIM에서 시뮬레이션 되었다.

차량은 EcoSIM의 시뮬레이션 전장(Map) 위에서 그림 4의 목표 모델들을 기반으로 자율 주행한다. 자율 주행 중 사전에 정의된 오류 모델에 해당하는 상황 발생 시 자율 컴퓨팅 기능이 수행된다. 예를 들어 주행 중 돌발적으로 나타나는 장애물에 대한 회피 정책을 수행하는 목표인 '장애물 회피(G5)'의 경우, 차량은 표 1의 NSC들에 대한 점검을 수행하다가 장애물 발견 시 NSC를 위반함을 인식한다. (상태명: SensingObj, NSC명: Obj_recognition, 조건: Detected_Object = normal)

장애물이 비전 센서가 인식할 수 있는 벽, 지형지물 등 알려진 물체일 경우 차량은 문제없이 동작하지만, 그렇지 않을 경우 자율 컴퓨팅 엔진은 해당 상황을 결함으로 판단하고 표 2의 결함 모델을 조회한다.(결함 모델명: UnknownObject, 결함 상태: image Analysis() = fail).

표 3. 적응 정책
Table 3. Adaptation Policy

Policy	FM name	Strategy name	Parameters
StopMoving	BrokenMotor	Kill	(process ID)
ChangeTorque	BrokenMotor	ChangeValue	({var, val})
...	...	...	...
ChangeMission	WheelBroken	SWUpgrade	(SW ID)
UpgradeSW	**UnknownObj**	**SWUpgrade**	**(SW ID)**

표 4. 적응 전략
Table 4. Adaptation Strategy

Strategy	Action	Type
ChangeValue	Change internal values	Internal
Kill	Kill Process	External
...	...	...
SWUpgrade	Update current SW to new SW	External
UpgradeSW	**UnknownObj**	**SWUpdate**

결함 모델의 조회 결과 해당 장애물의 종류를 판별할 수 있는 새로운 소프트웨어의 업그레이드가 필요함을 표 3의 적응 정책으로부터 결정하고, 표 4의 적응 전략을 통해 소프트웨어 업데이트를 실행한다. 이때 동적 업데이트 관리자(DUM)가 이용된다.

결과적으로 자가 적응 차량은 기존에 파악되지 않은 새로운 형태의 장애물을 인지하고, 해당 장애물을 피하거나, 제거하는 등 장애물 회피에 대한 목적을 수행할 수 있다.

본 적용 사례를 통해 CPS 자율 컴퓨팅 프레임워크의 몇 가지 장점을 확인할 수 있다. 첫 번째로 시스템의 자원에서 발생할 수 잠재적인 결함과 오류, 해결 정책 및 전략을 지식베이스를 통해 관리함으로써 프로그램 코드 내에 조건문이나 예외처리를 통해 오류를 인지하고 대응하는 방법에 비해 결함, 오류, 정책 및 전략의 동적 추가 및 상관 관계 변경 등이 가능해져 보다 유연성 있는 시스템 개발과 운영이 가능해진다. 두 번째로 CPS의 시스템 요구사항을 기반으로 작성된 목표 모델의 정상 상태를 관리하는 전역 자율 컴퓨팅 엔진을 통해 시스템의 요구사항을 동적으로 테스팅하는 효과를 얻을 수 있다. 세 번째로 동적 업데이트 관리자를 통한 운영 중 애플리케이션의 기능 업데이트를 통해 시스템 개발 단계에서 예측할 수 없는 상황에 대한 자가 적응도를 높일 수 있다.

Ⅶ. 결론

본 논문에서는 사이버-물리 시스템(CPS) 환경에서, 시스템의 자가 관리 및 자가 적응 기능의 제공을 위한 모델 기반 자율 컴퓨팅 프레임워크를 제안하였다.

제안 프레임워크를 통한 CPS 개발은 결함 모델, 오류 모델 및 목표 모델을 정의하는 설계 단계와 자율 컴퓨팅 관리자를 중심으로 모니터링(Monitoring)-분석(Analysis)-계획(Plan)-실행(Execute) 과정을 수행하는 'MAPE' 단계로 이루어져 있다. 또한 제안 프레임워크는 모델 참조 적응 제어(MRAC) 루프를 기반으로 계층화된 자율 컴퓨팅 엔진 구조를 통해 CPS의 시스템 차원의 목표 만족 상태와 각 자원 관리자가 관리하는 자원의 정상 상태를 만족시킬 수 있는 장점을 가지고 있다. 즉, CPS 환경에서 발생하는 여러 상황에 대해 사람의 개입 없이 자가 관리할 수 있으며, 주변 물리 환경의 변화 시 새로운 소프트웨어의 업데이트 등을 통해 자가 적응이 가능하다.

본 연구 결과는 향후 다양한 CPS에 적용될 예정이지만, CPS의 구조, 행위, 결함 등의 모델링은 도메인 전문가의 역량에 많은 영향을 받는다. 따라서 본 연구의 향후 계획은 도메인 전문가의 모델 개발을 지원하는 주제, 예를 들어 모델 재사용성 향상, 결함에 대한 적응 정책 및 전략의 학습 기법 등에 초점이 맞추어질 것이다. 또한 자율 컴퓨팅 프레임워크의 각 중간 관리자(SRM, DUM, HFM), 그리고 지식 베이스의 구조 및 성능 향상에 대한 추가 연구도 진행될 것이다.

참고문헌

[1] R. Kazman, H.M. Chen, "The Metropolis Model: A New Logic for the Development of Crowdsourced Systems," Communications of the ACM, pp.78-84, 2009.

[2] SEI Software-Intensive Systems (ISIS), "Integration of Software-Intensive Systems Initiative: Addressing System-of-Systems Interoperability," 2007.

[3] P. Feiler, R.P. Gabriel, J. Goodenough, R. Linger, T. Longstaff, R. Kazman, M. Klein, L. Northrop, D. Schmidt, K. Sullivan, K. Wallnau, "Ultra-Large-Scale Systems - The Software Challenge of the Future," TR, 2006.

[4] E.A. Lee, "Cyber physical systems: Design challenges," Procdeengs on ISORC'08, 2008.

[5] 김진명, 이해영, 전인걸, 김원태, 박승민, "CPS 모델 신뢰성 검증을 위한 HLA/RTI 기반의 하이브리드 시뮬레이션 프레임워크," 대한임베디드공학회 추계학술대회, 2010.

[6] H. Muller, H. Kienle, U. Stege, "Autonomic Computing Now You See It, Now you Don't," LNCS, Vol. 5413, pp. 32-54, 2009.

[7] C. Lee, H. Youn, I. Chun, E. Lee, "A Runtime Evaluation Methodology and Framework for Autonomic Systems," Int. J. on Network Security , Vol. 3, No. 2, pp.21-25, 2012.

[8] IBM, "An Architectural Blueprint for Autonomic Computing 3rd Edition," IBM Autonomic Computing White Paper, 2005.

[9] J.O. Kephart, D.M. Ches, "The Vision of Autonomic Computing," IEEE Computer, Vol. 36, No. 1, pp.41-50, 2003

[10] IBM Research Blue Gene Project, http://www.research.ibm.com/bluegene/index.html

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

[12] I. Hong, H. Youn, I. Chun, E. Lee, "Autonomic Computing Framework for Cyber-Physical Systems," Proceedings on Int. Conf. on Advances in Computing, Control, and Telecommunication Technologies, pp.148-151, 2001.

[13] I. Chun, J. Kim, H. Lee, W. Kim, S. Park, E. Lee, "Faults and Adaptation Policy Modeling Method for Self-adaptive Robots," UCMA'11,, 2011.

[14] A. Karl, B. Wittenmark, "Adaptive Control," Dover Books, 2008.

저 자 소 개

강 성 주

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

전 인 걸

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

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

김 원 태

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