Software-in-the-Loop 시뮬레이션 기반의 임베디드 소프트웨어의 견고성 테스팅

정수용°, 백태산*, 황병일*, 이우진*
*°경북대학교 컴퓨터학부
e-mail : kyo1363@naver.com, sans79@gmail.com, halove2002@gmail.com, woojin@knu.ac.kr

A Testing on Robustness of Embedded Software Based on Software-in-the-Loop Simulation

Sooyong Jeong°, Tae-San Baek*, Bueng Il Hwang*, Woo Jin Lee*
*°School of CSE, Kyungpook National University

● Abstract ●

소프트웨어의 작동 중에 중대한 고장 없이 유연하게 대처할 수 있는 성질은 견고성이라 불리며 임베디드 소프트웨어에서 중요하게 여겨진다. 본 논문에서는 개발 중인 임베디드 소프트웨어의 견고성을 조기에 검증하기 위하여, 실물 시스템을 이용한 테스트가 갖는 물리적 한계를 극복할 수 있는 Software-in-the-Loop 시뮬레이션을 이용하여 개발 중인 PC에 견고성 테스팅 환경을 갖추는 방법을 제시한다. 제시한 방법은 소프트웨어의 견고성을 표현할 수 있게끔 만드는 테스트 케이스를 생성하고, 가상 시뮬레이션 환경을 구성하여 테스트 케이스를 실행함으로써 소프트웨어의 견고성을 객관적인 수치의 형태로 나타내는 방법을 보인다.

키워드: 견고성(robustness), 소프트웨어 테스팅(software testing), SiL 시뮬레이션 (Software-in-the-Loop simulation)

I. Introduction

임베디드 소프트웨어는 시스템의 센서 및 액츄에이터를 통하여 주변 환경 및 타 시스템과의 상호 작용을 수행하며 작동한다. 그러므로 일반적인 응용 소프트웨어에 비하여 외부로부터 비정상 입력을 포함하는 시나리오에 노출되기 쉬우며, 오류가 발생할 수 있는 여지가 크다. 뿐만 아니라 임베디드 소프트웨어의 오류(fault)로 인한 고장 상황(failure)이 일으키는 잘못된 액츄에이터의 작동이 시스템 하드웨어의 고장 및 인적/물적 상해를 가하는 경우가 발생할 수 있다. 소프트웨어가 상해 등을 가할 수 있는 중대한 고장을 나타내지 않고 유연하게 대처하는 성질을 견고성이라 부르며, 인적/물적 이해관계에서 직접적인 영향을 주는 임베디드 소프트웨어에 있어서 견고성은 중요한 품질의 척도이다.

본 논문에서는 임베디드 소프트웨어의 소스 코드를 Software-in-the-Loop(SiL) 시뮬레이션 및 가상 프로토타입을 이용하여 PC 환경에서 테스트하는 기술을 기반으로 확장함으로써 코드의 견고성을 측정 및 수치화하는 방법을 보인다. 시스템의 요구 사항으로부터 견고성을 측정하기 위한 테스트 케이스를 생성 및 실행하고, 해당 실행 결과로부터 견고성을 측정하는 방법을 제시한다.

II. Preliminaries

1. Related works

1.1 Software-in-the Loop 시뮬레이션

임베디드 소프트웨어를 테스트하기 위한 방법으로는 실물 시스템을 이용하거나, Hardware-in-the-Loop (HiL) 또는 Software-in-the-Loop(SiL) 시뮬레이션을 이용하는 방법 등이 있다. 이 중에서 SiL 시뮬레이션은 센서나 액츄에이터, 동역학 모델뿐만 아니라 소프트웨어가 실행될 제어기까지 PC 내부의 시뮬레이션으로 이루어지기 때문에 테스팅 중 시스템의 오류로 인한 물리적 사고를 피할 수 있다[1]. 그리고 제어기가 상대적으로 처리속도가 빠른 PC를 통하여 시뮬레이션되기 때문에 결과를 빨리 얻을 수 있다.

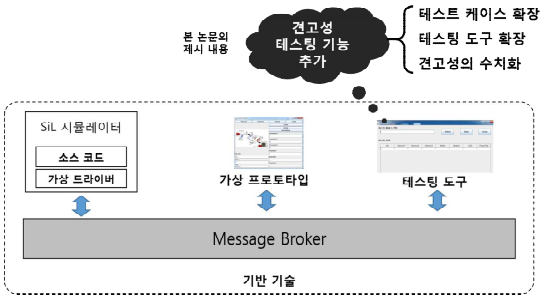
1.2 견고성 테스팅

소프트웨어에 있어 견고성은 '정상적인 작동 범위를 벗어나더라도 중대한 문제를 일으키지 않고 유연하게 대처할 수 있는 정도'로 정의하고 있다[2].

임베디드 소프트웨어의 견고성 테스팅을 위한 방안으로 QEMU 에뮬레이터[3]를 이용하는 기법이 제안되었으나[4], 해당 논문은 QEMU를 지원하는 시스템에 대하여만 적용 가능하며, 구체적이고

객관적으로 견고성을 표현하는 방법을 제시하지 않았다는 한계점이 있다.

본 논문에서는 기존의 방법이 대상 시스템에 구애받는 문제점을 최소화하는 SiL 시뮬레이션 기법을 이용하고, 테스팅 결과로부터 견고성을 수치로 얻는 방법을 제시한다.



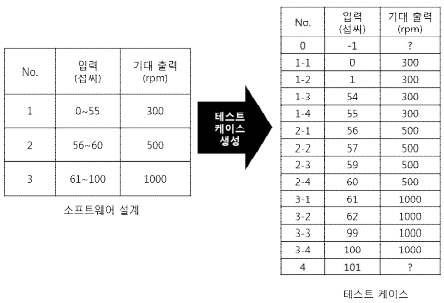
SiL 기반 소프트웨어 견고성 측정 방법 개요

1. 개 요

임베디드 소프트웨어의 견고성을 SiL 시뮬레이션으로 테스트하는 과정은 크게 세 가지로 나누어진다. 첫 번째로 견고성을 테스트하기 위한 요소를 포함하도록 테스트 케이스를 생성한다. 두 번째로 SiL 시뮬레이션 및 가상 프로토타입을 기반으로 임베디드 소프트웨어를 대상 시스템을 대신하여 개발용 PC에서 구동하는 환경을 구현한다. 마지막으로 생성한 테스트 케이스를 가상 시뮬레이션 환경에서 실행하고, 결과물을 분석하여 코드의 견고성을 측정한다. 그림 1은 본 논문에서 제시하는 과정을 도식화한다. 소스 코드는 가상 프로토타입과 가상 드라이버로써 임베디드 시스템을 대신하여 PC에서 시뮬레이션 및 테스팅이 이루어지며, 가상 프로토타입 기반 테스팅 도구에 상기의 견고성 테스팅을 위한 기능들이 추가된다.

2. 테스트 케이스 생성 과정

테스팅 과정에서 견고성에 대한 내용을 얻기 위해서는 견고성을 측정하기 위한 요소가 포함되는 테스트 케이스의 구현이 필요하다. 이 과정에서는 시스템의 설계 문서가 요구된다. 설계 문서는 유스케이스 및 시나리오와 같이 소프트웨어의 작동을 위한 구체적인 가이드라인을 제공할 뿐만 아니라, 하드웨어의 성능과 같은 제약사항을 포함하고 있으므로, 이들을 조합하면 견고성 검증 목적의 테스트 케이스를 얻을 수 있다.



테스트 케이스의 입출력쌍 생성 과정

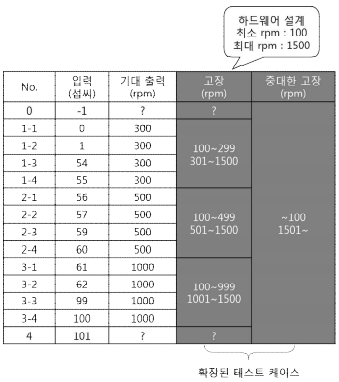
2.1 입력 및 출력의 출력값 생성

소프트웨어의 설계 문서는 임베디드 시스템의 각 기능을 제어하기 위한 조건과, 그 조건에 따른 결과의 기댓값 등을 포함하고 있다. 이를 바탕으로 경계값 분석 기법[5]을 이용하여 제어 요소의 조건별 경계값과 중간값을 취하면 테스트 케이스의 입력값과 출력의 기댓값으로 이루어진 쌍을 얻을 수 있다. 그림 2는 소프트웨어 설계 상의 입력 범위 및 결과의 기댓값으로부터 경계값 분석 기법으로 입출력에 대한 테스트 케이스를 생성한 예시이다.

설계로부터 정상적으로 발생할 수 있는 입출력들의 경계를 구분하고, 경계선 사이에 있는 값과 경계선 안쪽에 위치한 가장 바깥값을 테스트 케이스로 만든다. 그림 2의 1-1번 ~ 3-4번 테스트 케이스가 본 과정에서 생성된다. 그리고, 모든 정상 입력의 경계를 벗어나면서 가장 경계의 가까이에 위치하는, 비정상 입력에 대한 경우를 테스트 케이스로 만든다. 정상적인 입력이 아니므로 통상적으로 출력에 대한 예상을 하지 않을 것이다. 그러므로 본 입력에 대한 기댓값은 don't care로 한다. 그림 2의 0번, 4번 테스트 케이스가 이에 해당한다.

2.2 소프트웨어의 고장 분류 및 테스트 케이스 확장

하드웨어의 성능 사양을 기록한 설계 문서는 작동 범위 및 한계를 기록하고 있으므로, 하드웨어가 수용 가능한 동작의 범위를 표현하게 된다. 이로부터 소프트웨어의 오류로부터 나타날 수 있는 고장 행동의 종류를 분류할 수 있으며, 소프트웨어가 고장에 어떻게 대처하는지 판단할 기준을 만들 수 있다. 소프트웨어의 오류로 인하여 하드웨어 동작이 기댓값을 벗어나는 고장 상황이지만, 하드웨어의 설계 상 수용 가능한 동작의 범위 내이므로 중대한 문제를 일으키지는 않을 경우는 '고장'으로 분류한다. 반면 하드웨어가 수용 가능한 범위 내에서 작동하지 않을 경우는 시스템의 고장뿐만 아니라 주변 환경에 대한 상해를 가할 수 있으므로 이는 '중대한 고장'으로 분류한다.



고장 판단에 대한 테스트 케이스 확장

그림 3은 하드웨어 설계로부터 나타날 수 있는 고장 및 중대한 고장 상황을 판단하고, 이것을 그림 2의 테스트 케이스에 확장한 예시이다. 설계 문서로부터 출력 부하(팬)의 회전이 100~1500rpm 내에서 작동해야 한다는 사실을 바탕으로, 고장 상태를 구분한다. 팬이 기댓값 작동 범위를 벗어나지만 100~1500rpm 사이의 정상 범위 내라면 '고장'으로, 팬이 100rpm 미만 또는 1500rpm 초과로

작동하면 '중대한 고장'으로 구분하고, 이를 테스트 케이스에 추가하였다.

3. 테스트 결과를 기반으로 한 견고성의 수치화

SiL 테스팅에 사용한 테스트 케이스 및 그 실행 결과, 확률을 바탕으로 하여 시뮬레이션한 소프트웨어의 견고성을 수치화하는 방법을 제시한다.

$$Robustness = 1 - \sum_{x \in Q} \left(P(x) \cdot \frac{F_S(x)}{S(x) + F(x)} \right)$$
$$Q : \{x \mid x = \text{발생 가능한 모든 입력}\}$$

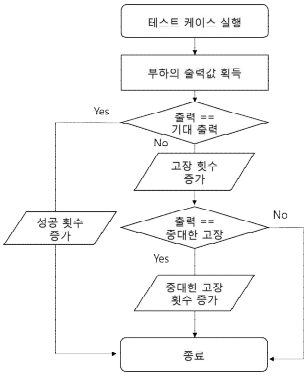
견고성 계산을 위한 수식

견고성 수치화를 위하여 필요한 수치 및 기호

Symbol	Description
P(x)	Probability of the input x
S(x)	The number of succeeded test cases at input x
F(x)	The number of test cases make failure at input x (Includes serious failures)
FS(x)	The number of test cases make serious failure at input x

표 1은 견고성의 측정을 위하여 사용하는 수치를 기호화한 것이다. P(x)는 모든 경우의 수 중에서 입력이 x로 일어날 확률을 뜻하며, 자주 일어나는 시나리오가 견고성의 계산에 있어서 더 큰 비중을 갖도록 만드는 가중치로써 작용한다. 입력 x에 대하여 성공한 테스트 케이스의 수 S(x)와 실패한 테스트 케이스의 수 F(x)를 각각 더하여 해당 입력의 전체 테스트 케이스 수를 구하고, 이 중에서 중대한 고장을 유발한 테스트 케이스의 수 Fs(x)의 비중을 계산한다. 이를 확률 P(x)와 곱하면 '입력이 x가 되고, 중대한 고장이 테스트에서 발견될 확률'이 된다. 이를 모든 경우의 수에 대하여 합산하면 '전체 테스트 케이스에서 중대한 고장이 테스트에서 발견될 확률'이 된다. 견고성은 전체 입력에 대하여 중대한 고장이 발생하지 않는 확률이므로, 전체의 확률 1로부터 중대한 고장이 발생할 확률을 뺀 수치로 그림 4와 같이 표현할 수 있다.

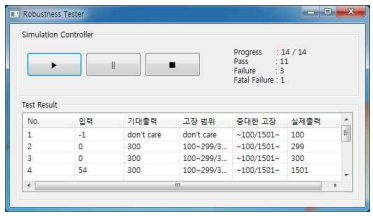
$\sum_{x \in Q} P(x) = 1$과 $\frac{F_S(x)}{S(x) + F(x)} \leq 1$이므로, 그림 4의 수식은 $0 \leq Robustness \leq 1$의 범위 내에서 나타나게 된다.



테스트 케이스에 대한 출력 및 결과 처리

4. SiL 시뮬레이터 및 가상 프로토타입을 기반으로 한 테스터 구현

임베디드 소프트웨어에 대한 SiL 시뮬레이션 및 가상 프로토타입을 이용한 테스트 기술에 대한 연구가 이루어지고 있다[6]. 해당 연구에서는 센서 및 액츄에이터를 다루는 장치 드라이버 코드를 PC 상의 입출력으로 대체할 수 있게끔 만드는 가상 드라이버로 설정하고, 이를 PC 상의 SWF 이미지로 구성된 가상 프로토타입과 연결함으로써 PC에서 임베디드 소프트웨어를 SiL 시뮬레이션으로 구현하는 아이디어를 제시하고 있다. 시뮬레이션의 테스트 자동화를 위하여 표의 형식으로 만들어진 테스트 케이스를 불러와 실행하고, 테스트 케이스의 입력에 대하여 원하는 출력이 나타나는지의 여부에 따라 테스트의 성공/실패를 나타낸다. 본 논문에서는 기존의 성공/실패 여부만을 표시하는 테스팅 방법을 확장하여 견고성 측정을 위한 테스트 케이스를 자동으로 수행하는 방법을 제시한다. 기존의 SiL 기반 테스터의 테스트 케이스는 유스 케이스 및 특정 타이밍에 대한 입력과 출력의 기댓값의 쌍으로 이루어져 있으나, 본 논문의 테스트 케이스는 출력 및 결과의 처리 절차가 추가적으로 요구된다. 그림 5는 테스트 케이스 출력에 대한 결과 처리 절차를 나타낸다.



견고성 테스팅 도구의 실행 예시

테스트 케이스가 실행됨에 따라 입력이 시뮬레이터로 주입되고, 시뮬레이터는 주입된 입력에 따른 부하의 출력을 표현한다. 테스터는 부하의 출력을 획득함으로써 성공/실패의 여부를 가린다. 만약 테스트가 실패한다면, 소프트웨어의 오류로 인하여 외부로의 행동이 나타난 것이므로 고장으로 취급하고, 고장의 발생 횟수를 증가한다. 그리고 출력값을 중대한 고장 의 판단 기준에 대입하여 비교함으로써, 본 출력이 중대한 고장을 의미하는지 판단한다. 중대한 고장이라고 판단

될 경우 중대한 고장이 발생한 횟수를 증가시킨다. 본 테스트를 통하여 테스트의 성공 횟수, 테스트 중에 코드의 오류로 인하여 고장을 유발한 횟수, 그리고 이 중에서 중대한 고장을 유발한 횟수를 결과물로 얻게 된다. 그림 6은 기능이 확장된 테스팅 도구의 실행 예시를 보인다.

IV. Conclusions

본 논문은 기존에 연구된 SiL 시뮬레이션 기반의 테스팅 기법에 견고성 측정 기능을 추가하는 방안을 제시하였다. 하드웨어 및 소프트웨어 설계로부터 견고성 측정을 위한 테스트 케이스를 설계하고, 임베디드 소프트웨어 코드 중 장치 드라이버에 해당하는 부분을 가상 드라이버로 대체하는 시뮬레이션 및 가상 프로토타입 기반 테스트 도구에 본 테스트 케이스를 실행시킨다. 실행 결과로부터 견고성 계산 공식을 이용하여 소프트웨어의 견고성을 표현한다. 제시한 방법으로써 소프트웨어의 견고성을 객관적인 수치값으로 나타내어 테스트의 객관성을 높이고, 차후 소프트웨어 품질 향상에 기여할 수 있을 것으로 기대한다.

Acknowledgement

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 ICT융합고급인력과정지원사업 (IITP-2015 -H8601-15-1002) 및 교육부 및 한국연구재단의 BK21 플러스 사업(경북대학교 컴퓨터학부 Smart Life실현을 위한 SW인력양성사업단)으로 지원된 연구임 (21A20131600005)

References

[1] A. Bayha, F. Gruneis, and B. Schatz, “Model-based software in-the-loop-test of autonomous systems,” Proceedings of the 2012 Symposium on Theory of Modeling and Simulation - DEVS Integrative M&S Symposium, No. 30, 2012.

[2] M. Pezze, and M. Young, “Software Testing and Analysis:Process, Principles, and Techniques,” pp.45, 2008.

[3] QEMU - open source processor emulator, http://wiki.qemu.org/Main_Page

[4] M. D’Angelo, A. Ferrari, O. Ogaard, C. Pinello, and A. Ulisse, “A Simulator based on QEMU and SystemC for Robustness Testing of a Networked Linux-based Fire Detection and Alarm System.”, Proceeding of the Conference on Embedded Real Time Software and Systems - ERTS2, No. 4B.3, 2012.

[5] K. K. Aggarwal, and Y. Singh, “Software Engineering,” pp.352~357, New Age International Publishers, 2005.

[6] H. Ryu, S. Jeong, S. Lee, J. Kim, H. Park, S. Lee, and W. Lee, “A Testing Technique based on Virtual Prototype for Embedded Software,” IEMEK Journal of Embedded Systems and Application, Vol. 9, No. 6, 2014.