대한산업공학회/한국경영과학회 2004 춘계 학술대회
2004년 5월 21일 ~22일 전북대학교

POF기반한 내장형 소프트웨어의 테스트 전략

이상용1, 장중순, 장세헌, 고병각, *최경희, *박승규, **정기현, ***이명호
아주대학교 산업공학과 / *아주대학교 컴퓨터공학과
**아주대학교 전자공학과 / ***세명대학교 인터넷 정보공학부

Test case generation strategy and method of embedded software based on POF(Physics of Failure)

S.Y. LEE, J.S. Jang, S.H. Jang, B.G. Ko
*K.H. Choi, *S.K. Park, **K.H. Jung , ***M.H. Lee
Industrial Eng., Ajou Univ. / *Computer Eng., Ajou Univ.
** Electronics Eng., Ajou Univ. / ***Div. of Internet Information, Semyung Univ.

Abstract

It is still not sufficient for the famous embedded software test methods such as Finite State Machine, Software Cost Reduction and model coverage based test case generation, to pinpoint where bugs hang around and to figure out what makes the bugs. A new approach to ameliorate the drawbacks is proposed in this paper. In the approach, we define a generic model for embedded software. And we also define failure mechanism for embedded software, and a way to generate test cases based on it.

Keywords : *embedded software, test case generation, finite state machine*

1. 서론

내장형 시스템(embedded system)은 일반 desktop 컴퓨터와는 달리 sensor나 actuator등을 갖춘 H/W 장비을 제어하기 위하여 특수목적의 S/W를 내장한 시스템을 말한다. 내장형 시스템은 산업용 기기, 군사용 무기, 항공 우주산업용 기기 등 고가의 장비나 시스템들에서 주로 사용되었으나, 최근에는 가전제품의 다양한 기능을 제공하기 위하여 사용되기도 하고, 휴대용 단말기기나 게임기 등에도 활용되는 등 그 사용의 범위가 매우 빠르게 확산되고 있다.

산업용 기기, 군사장비 등의 고신뢰성을 요구하는 고가의 내장형 시스템들에 내장되는 S/W는 매우 엄격한 시스템 개발 방법론에 따라서 개발되는 것이 일반적이다. 특히, 내장형 기기의 품질이 내장된 S/W를 얼마나 엄격하고 완벽하게 테스트하는 가에 따라 좌우된다는 인식 아래 내장형 S/W를 테스트하는 방법도 달라지고 있다. 예를 들면, 코딩이 끝난 S/W에 테스트 케이스를 적용하여 오류를 발견하는 전략이 주로 사용되었으나 최근에는 S/W 설계 단계에서부터 테스트를 고려하는 개발 방법론도 제안되고 있다.

예를 들면, 내장형 S/W의 스펙(specification)을 설계단계에서부터 자연어로 기술하고 이를 바탕으로 개발하던 방법에서 벗어나 일정한 형식을 가진 프로그래밍 언어로 기술함으로써 개발코자하는 S/W의 기능을 정확히 할 것이 요구되고 있다. 대표적인 언어로는 MSC(message sequence chart) (ITU, 1999) 나 SDL(specification description language) (ITU, 1999), Charon(Alur, 2000)이나 SCR(software cost reduction) (Paul, 1996) 이 있다. 또한, S/W가 요구되는 기능을 수행하고 있는지를 테스트할 때에도 경험적 지식에 의존하여 임의로 작성된 테스트 시나리오를 실행함으로써 테스트하는 방법 대신에 S/W의 모델 모델로부터 그의 모든 기능을 커버할 수 있는 테스트케이스를 생성한 후 이를 개발된

1이상용, 수원시 영통구 아주대학교 팔달관 229 FAX:031-219-1610
E-mail:biztech.ajou.ac.kr

S/W에 적용하여 나온 결과는 재 정의된 spec으로부터 계산된 expected output과 비교하여 소프트웨어의 고장 유무를 판단한다. Statechart, FSM(Finite State Machine), FSM의 단점(산태나 전이의 수가 폭발적으로 증가하여 취급이 어려워 진다는 점)을 보완한 Extended FSM(David, 1996), Hierarchical FSM(Alur, 2001)등은 대표적인 모델링 기법에 속한다.

이러한 모델을 바탕으로 테스트케이스를 생성하고 이를 개발된 S/W에 적용하는 전략은 S/W의 오류를 검증하는데 매우 탁월한 효과를 가지고 있음이 알려진 바 있다(Blk 2001). 그러나 모델기반의 테스트케이스 생성기법들은 무수히 많은 테스트 케이스 중 모델을 커버할 수 있는 최소한의 테스트 케이스로 오류들을 찾아내기 위한 방법론들을 제시하고 있어서 모델이 정확하지 않다면 오류를 간과할 수 있으며, 실제 오류를 찾았다 하더라도 그 원인파악이 매우 어렵다. Giri(2003)는 이러한 단점을 극복하기 위하여 브레인 스토밍기법을 사용하여 발생할 수 있는 오류들을 분류하고 분류를 토대로 테스트를 수행하면 오류에 좀더 구조적으로 접근 가능하고 테스트수행에 드는 시간도 줄일 수 있고 말하고 있다. 그러나 Giri의 방법은 체계적 분류에 의한 방법이라기 보다는 ad hoc 방법에 의하여 오류를 분류하였기 때문에 해당 S/W가 그의 스펙을 완벽히 만족하는지 검증하기에는 보완이 필요하다.

본 논문에서는 Giri의 브레인 스토밍 방법의 장점은 살리되, 단점을 보완하기 위하여 ad hoc 브레인 스토밍 대신에 내장형 S/W에서 발생할 수 있는 고장메커니즘(failure mechanism)을 정의하고 분류하여, 고장 메커니즘에 따라 해당 고장을 테스트하는데 적합한 케이스를 생성하는 방안을 제시하고자 한다. 고장 메카니즘이란 고장에 이르는 과정을 가리킨다. H/W의 고장이나 신뢰성에서는 많이 사용되어 왔으나 S/W분야에서는 아직 정의되지 안고 있다.

본 논문에서 제안하는 방법은 내장형 S/W의 블랙박스(black box) 테스트를 전제로 한다. 블랙박스 테스트는 S/W의 코드를 기반으로 이루어지는 화이트박스(white box) 테스트에 의하여만 발견되는 S/W의 논리적 결함에 따라 발생하는 고장의 발견에 어려움이 있다는 단점이 있다. 그러나 화이트박스 테스트보다는 간단하다는 장점과 입력에 대한 출력만으로 S/W의 기능 테스트가 가능하고, S/W의 소스코드를 모르더라도 테스트가 가능하여 다양한 내장형 S/W의 테스트에도 응용이 가능하다는 장점이 있다.

본 논문의 구조는 아래와 같다. 2절에서는 내장형 S/W에서 발생할 수 있는 고장 메커니즘에 대해서 정의하고, 3절에서는 고장메니즘을 찾을 수 있는 방법론 중 소프트웨어 모델링 기법인 FSM을 내장형 시스템에 적용할 경우 적합한 상태의 정의 방법에 대해서 제안하고, 4절에서는 2절에서 정의한 각각의 고장 메커니즘에 대한 테스트 방법론들을 제시한다.

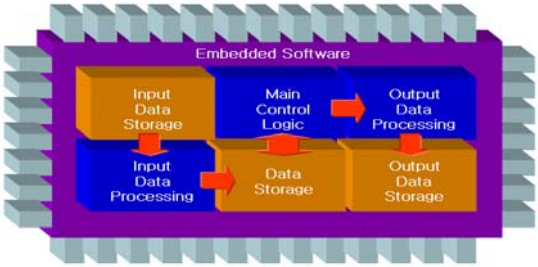

Figure 1. Embedded software structure

2.내장형 S/W 모델링

내장형 시스템은 일반적으로 크게 세가지 요소로 구분된다. 하나는 S/W를 탑재한 시스템은 사용자의 조작 또는 환경 변화를 입력받는 입력 요소이다. 내장형 시스템의 입력 요소로는 버튼이나 키보드처럼 사용자가 직접 입력토록 허용하는 요소와 환경을 스스로 인지할 수 있는 센서 등이 있다. 입력 요소들은 일반적으로 아날로그 시그널의 형태로 ADC (analog to digital converter)를 통하여 디지털 값으로 변화되어 입력된다.

두 번째 요소로는 입력 요소에 의하여 입력되고 저장된 정보를 분석하고 처리하여 하드웨어 요소들을 구동시키는 위한 제어명령을 생산하는 제어부 요소이다. 입력 요소를 통하여 입력된 값들이 S/W로 구현된 알고리즘을 통하여 연산된 후 하드웨어를 구동하며, 내장형 S/W의 고장 메카니즘이 가장 많이 관여하는 요소이다.

세번째 요소로는 H/W 요소에 해당하는 출력 요소가 있다. 출력 요소에는 특정 H/W를 구동시키는 엑츄에이터(actuator)와 정보를 단순히 디스플레이(display)시켜주는 출력장치 등이 이에 속한다.

이러한 내장형 시스템에서의 일련의 데이터 흐름을 고려하여 볼 때, 내장형 소프트웨어의 전형적인 구조는 Figure 1과 같이 표시할 수 있다.

위 그림에서 데이터 저장소(data storage)는 데이터를 영구적으로 혹은 임시로 저장할 수 있는 메모리영역을 의미한다. 입력데이터 저장소는 입력 요소로부터 디지털로 변환된 값들을 저장하고 영역이다. 일반적으로 디지털로 변환된 데이터들이 매우

급격히 변화할 수 있기 때문에 버퍼로부터 저장하고 적당한 간격에 따라 샘플링을 수행하여 사용한다. 또한 주 제어로직 내부에서 사용할 수 있는 데이터로 변환을 시키며, 때에 따라서는 급격한 변동에 대한 보정을 수행한다. 만일 입력장치의 고장을 인지한 경우에는 fail-safe기능에 의거 특정 값으로 설정되거나 무시되기도 한다. data storage는 입력 데이터처리로부터 얻어진 데이터와 주제어로직에서 사용하는 각종 변수 및 데이터들이 저장되는 공간이으로써 주 제어로직과 빈번한 입출력이 수행된다. 주 제어로직은 내장형 시스템이 구동되어야할 스펙이 S/W로 구현된 것으로써 입력된 데이터를 연산하여 스펙이 정의하고 있는 결과를 산출한다. 출력데이터 처리는 주 제어로직으로부터 계산된 결과값에 따라 어떤 장비를 구동 또는 정지 시킬 것인가를 결정하고 출력 시그널을 낼 수 있는 마이콤과 연결된 메모리영역에 값을 쓰게 된다. 또한 출력 디바이스의 고장을 인지하여 fail-safe기능에 의해 구동하게 된다.

3.내장형 S/W 고장메커니즘

위와 같은 구조를 따르는 내장형 S/W의 고장 원인(failure cause)는 각 요소들의 고장에 기인한다고 보는 것이 합리적이다. 예를 들어 입력 H/W의 고장은 입력 데이터 저장소에 고장을 일으키고, 나아가서 고장난 데이터를 직접적으로 사용하는 S/W는 데이터 저장소는 물론 출력 데이터 저장소의 고장을 일으키게 된다. 출력 데이터 저장소의 고장은 H/W의 올바른 작동을 방해하여 내장형 시스템의 고장을 사용자가 인식하게 만든다. 따라서 내장형 S/W를 테스트하기 위하여 이러한 고장 메카니즘을 고려하지 않고 S/W 만의 기능을 테스트하는 경우에는 올바르게 내장형 S/W를 테스트 할 수 없다.

Table 1은 내장형 S/W의 고장을 발생시킬 수 있는 고장 메커니즘을 고장의 발생 가능 장소와 원인을 정리한 것이다. Table 1이 모든 내장형 S/W의 고장을 대변한다고는 볼 수 없으나 적어도 단순한 ad hoc에 의존하여 테스트 케이스를 만든 것 보다는 체계적이라 할수 있다. 특히 내장형 S/W의 고장의 경우 입출 처리과정에서 발생할 수 있는 고장 메커니즘과 주 제어로직의 일부 고장 메커니즘들은 각각의 고장 메커니즘을 고려한 테스트 케이스를 실행시킴으로써 고장발생의 원인을 쉽게 알아낼 수 있다. 다만, 알고리즘간의 충돌, 자원의 고갈 및 시간관련 고장메커니즘들은 복수의 테스트 케이스들의 조합을 통하여 고장의 원인을 알아낼 수 있다. 이러한 테스트 케이스들의 조합은 기존의 소프트웨어 테스트 모델링 방법인 FSM에 의한 테스트 케이스 생성 방법들을 적용할 수 있다.

Table 1. Failure mechanism of Embedded S/W

failue site		failure mechanism
입력 데이터 처리	AD Sampling	샘플링 오류
	AD 변환	연산 오류
		참조 오류
	Fail-Safe	입력기기 고장시 처리부재
	Interrupt	Interrupt후 재수행 불가
	Polling	데이터를 가져오지 못함
출력 데이터 처리	출력포트 지정	연산결과:구동메모리 대응
	출력중단	시간 부정확
		Feedback 제어 오류
	Fail-Safe	출력기기 고장시 처리부재
주 제어 로직	연산	잘못된 연산자의 사용
		0으로 나눔
		이상치의 처리부재
	데이터 처리	잘못된 참조
		Default값의 미지정
		오버플르우 or 언더플로우
	분기제어	조건의 불만족
		하드웨어 구동점오차 부재
		잘못된 서브루틴 호출
	루프제어	탈출조건 부재
		사용변수의 메모리 미해제
	폴링	I/O 오류
	알고리즘	동일 리소스를 사용하는 알고리즘간의 충돌

4. POF기반 테스트 전략 및 테스트 방법

본 절에서는 내장형 S/W의 모델과 고장메커니즘을 이용한 체계적인 테스트 케이스 설계 및 생성 전략을 기술한다.

4.1 S/W 모델 구성 요소별 테스트

앞절의 고장 메카니즘에서 살펴 본 바와 같이 내장형 S/W를 구성하는 요소 중에서 어느 하나라도 고장이 발생하면, 그 고장은 다른 요소의 고장으로 이어지게 된다. 따라서 S/W의 모든 요소를 일괄로 테스트 하는 것 보다는 요소별 테스트를 수행하여 한 곳의 고장이 다른 곳으로 파급되지 않도록 하는 것이 중요하다. 본 논문에서는 내장형 S/W의 모델의 각 요소별로 divide-and-conquer 전략에 따라 테스트를 수행 할 테스트 케이스를 생성하는 것이 매우 경제적이며, 타당한 전략이라고 하겠다.

예를 들면, 입력요소의 하나인 센서에 고장이 발생하더라도 입력저장소에 고장이 발생하지 않도록 구현하는 fail-safe 기능은 좋은 예이다. 이처럼 입력 센서의 fail-safe 기능을 테스트하기 위하여는 입력 저장소로의 다양한 입력과 입력요소의 고장에 대한

Table 2. Test Case Generation based on Failure Mechanism

Failure Site		Failure Mechanism	Test Case	방법
입력 데이터 처리	AD Sampling	샘플링 오류	빠르게 변화하는 입력 데이터	단위
	AD 변환	연산자 오류	일반 데이터	단위
		참조 오류	AD와 변환테이블간에 대응되는 모든 데이터	단위
	Fail-Safe	입력기기 고장시 처리부재	입력기기의 물리적인 연결 제거	단위
	Interrupt	Interrupt후 재기동 못함	Interrupt 데이터	단위
	Polling	데이터를 가져오지 못함	일반 데이터	단위
출력 데이터 처리	출력포트 지정	연산결과 : 구동메모리 대응	구동 메모리맵을 모두 커버할 수 있는 데이터	단위
	출력중단	시간 부정확	일반 데이터 반복	단위
		Feedback 제어 오류	일반 데이터	단위
	Fail-Safe	출력기기 고장시 처리부재	출력기기의 물리적인 연결 제거	단위
주 제어 로직	연산	잘못된 연산자의 사용	일반 데이터	단위
		0으로 나눔	스펙에 정의된 연산식을 고려한 데이터	단위
		이상치의 처리부재	이상치	단위
	데이터 처리	잘못된 참조	이상치	단위
		Default값의 미지정	시스템 On/Off	단위
		오버플르우 or 언더플로우	이상치	단위
	분기제어	조건의 불만족	조건의 분기점에 해당하는 데이터	단위
		하드웨어 구동점간 오차 부재	하드웨어 구동점을 지나는 상/하 연속 데이터	단위
		잘못된 서브루틴 호출	서브루틴을 호출할 수 있는 데이터	단위
	루프제어	탈출조건 부재	스펙에 정의된 탈출 조건 불만족 데이터	단위
		사용변수의 메모리 미해제	루프를 발생시킬 수 있는 일반데이터	FSM
	알고리즘	동일 리소스를 사용하는 알고리즘간의 충돌	공용 리소스 알고리즘간의 경쟁데이터	조합
			제어로직간의 전이를 발생시키는 데이터	FSM

테스트로 전체적인 내장형 S/W의 입력요소인 해당 센서에 대한 테스트로는 충분하다. 그리고, 이러한 입력 저장소의 테스트는 다음 요소인 입력 데이터처리부의 테스르를 입력 저장소가 한정하는 값에 대한 테스트가 가능한 케이스를 생성하면 된다.

4.2 요소간 결합에 대한 논리적 테스트

divide-and conquer 방식에 의한 내장형 S/W 요소별 테스트 케이스에 의하여 테스트가 완료된 후, 발생 가능한 입력 값에 대하여 S/W가 스펙을 준수하는 값을 출력하는가를 테스트 할 필요가 있다. Sean은 요소별로는 올바르게 행동한다고 하더라도 요소들이 관계에 의하여 발생하는 다양한 고장을 열거하면서 왜 어디에서 고장이 발생하는 가를 지적하고 있다. 고장으로 다양한 고장이 발생하기도 한다.(Sean 2003)

5. 결론

본 논문에서는 내장형 S/W의 모델과 고장메카니즘, 그리고 그에 기반하여 테스트 전략을 설계하는 전략에 대하여 기술하고, 이를 Table2과 같이 정리하였다. Table2에 기술된 고장 메카니즘과 테스트 전략이 완벽하지는 않지만 테스트 목적을 고장메커니즘으로 좀더 세분화하여 테스트케이스를 생성 함으로써 단순한 모델기반의 테스트케이스 생성에 비해 오류 검증능력을 높일 수 있다. 향후 내장형시스템에 대한 다각적인 고장메커니즘의 분류를 통해서 사용 환경에 적합한 테스트케이스 생성에 대해 연구하고자 한다.

Reference

ITU-T Z.100(1999) Formal Description Languge : Specification and Description Language (SDL).

ITU-T Z.120(1999) Formal Description Languge : Message Sequence Chart (MSC).

R. Alur, R. Grosn, Y. Hur, V. Kumar, LS. Lee(2000), Modular Specification of Hybrid system in CHARON, Computational and Control, LNCS1790, pp 6-19

E. A. Paul, E. B. Paul, M. William(1998), Using Model Checking to Generate Tests from Specifications, ICFEM

David Lee(1996), Principles and Methods of Testing Finite State Machines -A Survey, *Proceedings of the IEEE*, Vol. 84, No. 8.

V. Giri(2003), Bug Taxonomies : Use Them to Generate Better Tests, STAR EAST

M. Blackburn, R. Busser, A. Nauman, R. Knickerbocker(2001), Mars Polar Lander Fault Identification Using Model-based Testing, Proceedings in IEEE/NASA 26th Software Engineering Workshop

Sean M Beatty(2003), Where testing fails, Embedded system programming, URL : http://www.embedded.com/showArticle.jhtml?articleID=12800625