Transactions of KSAE, Vol. 22, No. 3, pp.68-74 (2014)

pISSN 1225-6382 / eISSN 2234-0149
DOI http://dx.doi.org/10.7467/KSAE.2014.22.3.068

ISO 26262 표준 기반의 소프트웨어 검증을 위한 소프트웨어 결함 주입 기법

이 상 호* · 신 승 환

현대오트론 주행제어팀

Software Fault Injection Test Methodology for the Software Verification of ISO 26262 Standards-based

Sangho Lee* • Seunghwan Shin

Driving Control Engineering Team, Hyundai-autron, Mtek IT Tower, 344 Pangyo-ro, Bundang-gu, Seongnam-si, Gyeonggi 463-400, Korea

(Received 31 October 2013 / Revised 29 January 2014 / Accepted 26 February 2014)

Abstract : As the number of ECUs (Electronic control units) are increasing, reliability and functional stability of a software in an ECU is getting more important. Therefore the application of functional safety standards ISO 26262 is making the software more reliable. Software fault injection test (SFIT) is required as a verification technique for the application of ISO 26262. In case of applying SFIT, an artificial error is injected to inspect the vulnerability of the system which is not easily detected during normal operation. In this paper, the basic concept of SFIT will be examined and the application of SIFT based on ISO26262 will be described.

Key words : Dynamic verification(동적 검증), FMEA(고장형태 영향분석), ISO 26262(기능 안전 표준), Robustness(강건성), SFIT(소프트웨어 결함 주입 테스트)

1. 배 경

최근 차량의 주행성능, 편의성, 안전성을 높이기 위한 차량 내 전자제어장치(ECU)의 급속한 증가 및 네트워크화로 인하여 자동차 내 전장부품에 대한 비중이 지속적으로 증가되고 있다. 컨설팅 업체인 맥킨지社의 보고서에 의하면, 자동차 제조원가에서 전장이 차지하는 비중은 2004년 19퍼센트에서 현재 20퍼센트 대 중반으로 높아진 데 이어 2015년엔 40퍼센트 수준으로 급성장 할 것으로 예상된다.[1] 이처럼 전자부품의 사용이 크게 증가하면서 전자적 오류에 따른 자동차 사고 위험성을 줄이는 것이 자동차 업계의 화두로 떠오르고 있다. 따라서 전자제어장치에 들어가는 소프트웨어의 품질 및 기능안정성이 자동차 산업에서 중요해지고 있으며 이를 만족시키고 신뢰성을 갖기 위한 소프트웨어 검증이 필요해지고 있다.

이러한 흐름에 맞춰 전자제어장치에 탑재되는 소프트웨어의 오류로 인한 사고방지를 위하여 자동차 기능 안전 국제 규격(ISO 26262)이 제정되었다. ISO 26262 표준은 명세, 설계, 구현, 통합, 검증, 인증에 이르는 개발 전 단계에서 최신 개발 방법 및 테스트 방법을 적용하도록 하고 있으며 기능 안전 규격을 준수하기 위한 각 단계에서의 요구 사항을 정의한다. 세부적으로는 시스템 레벨, 하드웨어 레벨, 소프트웨어 레벨에서 각 요구사항들이 정의되어있으며 다시 소프트웨어 레벨(ISO 26262, Part 6)에서 소프

*A part of this paper was presented at the KSAE 2013 Annual Conference and Exhibition

*Corresponding author, E-mail: sangho.yi@hyundai-autron.com

트웨어 개발 시 준수해야 할 요구사항과 소프트웨어 검증 시 준수해야 할 요구사항이 정의되어있다. 따라서 ISO 26262 표준을 기반으로 소프트웨어를 개발할 때에는 표준에 정의된 소프트웨어 검증 방법들을 수행해야 한다. 하지만 이처럼 ISO26262 표준 기반의 소프트웨어 검증의 중요성이 대두되고 있음에도 표준의 특성상 적용을 위한 실질적인 방법론에 대한 설명은 부족한 측면이 있다.

소프트웨어 결함 주입 기법(Software Fault Injection Test)이란 ISO 26262 표준의 적용에 있어 필요한 소프트웨어 검증 단계 중 하나로써 시스템에 인위적으로 결함을 주입하여 오류를 파악하고 시스템의 기능 안전상의 강건성을 검증하는 방법이다.[2] 이를 통해 시스템의 정상작동 시 쉽게 발견되지 않는 내부 결함을 파악하고 안전 기능이 예상치 못한 상황에서 적절하게 통제되고 견고성을 유지하는 지에 대한 검증이 가능하다.

현재 ISO 26262 표준에서는 소프트웨어 통합 검증 단계에서 결함 주입 기법을 ASIL C, D 등급에서 반드시 적용해야 한다고 정의하고 있으며 하위 등급에서도 적용을 권장하고 있다. 하지만 역시 기법의 적용을 위한 자세한 방법론에 대해서는 정의되지 않았다. 따라서 결함 주입 대상의 식별, 입력할 결함 데이터 추출, 결함 주입 방법 등 실제 적용을 위한 방법론에 대한 논의가 필요한 시점이다.

Methods		ASIL			
		A	B	C	D
1a	Requirements-based test[a]	++	++	++	++
1b	Interface test	++	++	++	++
1c	Fault injection test[b]	+	+	++	++
1d	Resource usage test[c, d]	+	+	+	++
1e	Back-to-back comparison test between model and code, if applicable[e]	+	+	++	++

Fig. 1 ISO 26262 Part6 Table 13. Methods for software integration testing

2. 관련 연구 동향

결함 주입 기법(Fault Injection Test)은 초기에 하드웨어의 강건성을 테스트하기 위한 방법으로써 하드웨어에 고의적인 결함을 인위적으로 일으키고 그에 따른 시스템의 의존성을 평가하는 검증기법으로 개발되었다.

그 후 소프트웨어 검증에 도입되어 주로 프로토콜, 커맨드 라인 파라미터, API를 대상으로 결함 주입을 통한 소프트웨어에 검증 기법에 대한 연구가 다수 이루어져왔으며 근래에 임베디드 소프트웨어에서의 결함 주입 기법의 연구가 활발해지고 있는 추세이다.

따라서 이미 기존에 소프트웨어 결함 주입 기법에 대한 다양한 논문들이 존재하고 있으며, 크게 컴파일 타임 주입 기법, 런타임 주입 기법으로 나뉘어 각각에 따른 세부 결함 주입 기법이 연구되었다.

최근 자동차 분야에서 소프트웨어 개발은 모델기반 개발(Model Based Development)로 진행된다. 따라서 소프트웨어 개발 시 소프트웨어 결함 주입을 모델링 하고 시뮬레이션 함으로써 소프트웨어의 강건성을 검증하는 연구가 진행되었다.[6]

다른 연구로는 소프트웨어에 주입하는 결함을 식별하기 위해서 시스템 수준에서 식별된 기능 안전 요구사항(Functional Safety Requirement)과 기술적 안전 요구사항(Technical Safety Requirement)를 분석하여 결함 주입 시험을 수행하는 방법에 관한 것이 있다.[7]

자동차 분야에서 진행되는 결함 주입 기법의 최근 연구는 모델을 사용하여 결함을 주입하거나 상위 시스템 요구사항에서 소프트웨어에 관한 결함을 식별하는 것으로 볼 수 있다. 하지만 기능안전에서 요구하는 결함 주입 시험은 소프트웨어 아키텍처 수준에서 결함을 식별하고 식별된 결함을 소프트웨어 주입하고 테스트하여 강건성을 확인해야 한다.

3. 본 론

본론에서는 결함 주입 기법을 수행하기 위한 결함 주입 대상의 식별, 입력할 결함 데이터 추출, 결함 주입 방법 등에 대한 실제 적용을 위한 방법론을 다루었다.

3.1 결함 주입 대상 식별

소프트웨어 결함 주입 기법을 적용하기 위해서는 먼저 결함이 주입될 대상 소프트웨어를 식별해야 한다. 시스템 전체에 대하여 결함을 주입하고 그에 대한 시스템의 반응과 결과를 관찰하는 것도 의미가 있을 수 있으나 이는 많은 시간적 비용적 낭비를 유발할 수 있다.

결함 주입 기법은 인위적인 결함을 주입하였을 때 시스템이 결함을 잘 처리하여 전체 시스템이 문제없이 정상 수행되는지를 살펴보는 강건성 테스트라고 볼 수 있다. 따라서 결함 주입 기법은 장애 허용 시스템(Fault Tolerant System)에 의한 강건설계가 구현된 소프트웨어 모듈에 대하여 수행하였을 때 의미가 있다. 만약 강건 설계가 이루어지지 않은 모듈에 대하여 원치 않는 결함데이터가 입력될 경우 잘못된 결과값이 리턴되는 것은 일견 당연한 현상이기 때문이다.

이를 ISO26262 표준에 따른 소프트웨어 개발과 접목시켜보면 기능안전표준에서는 작성된 소프트웨어 아키텍처에 잠재적인 결함이 없는지 분석하는 안전 분석을 반드시 수행해야 한다. 이를 위해 소프트웨어 아키텍처를 정의한 후 소프트웨어 FMEA (Failure Mode and Effects Analysis), FTA(Fault Tree Analysis)등의 과정을 통해 아키텍처상의 취약점을 분석하고 이를 보완하는 장애 허용 시스템이 추가된 소프트웨어 아키텍처를 다시 설계하는 방식으로 개발이 이루어진다. 따라서 ISO26262 표준 기반의 결함 주입 기법을 활용하기 위해서는 표준개발 프로세스에 따라 소프트웨어 안전분석을 통해 강건설계가 이루어진 소프트웨어 모듈에 대하여 결함주입을 활용하면 장애 허용 시스템이 잘 설계되었는지에 대하여 올바른 검증을 할 수 있을 것이다.

FMEA 결과는 테이블 형식으로 문서화 되므로 시스템의 고장과 원인과의 논리적 관계를 나타내기 어렵다. 개발초기에 개발하는 소프트웨어에 대한 이해 부족으로 FTA를 통한 분석은 최상위 고장으로부터 하향식으로 모든 원인을 찾기가 쉽지 않기 때문에, 중요한 고장 원인이 누락될 수 있는 위험이 있다.[8] 따라서 하드웨어나 기구처럼 시스템의 구조가 명확하지 않은 소프트웨어의 경우 FTA 보다는 FMEA를 사용하여 안전분석을 수행하는 편이 더욱 효과적이다. 아래에서 예시를 통해 FMEA를 통한 강건 설계가 이루어지는 방법을 나타내보겠다.

Fig. 1과 같이 레이더에서 입력신호를 받아 차량을 제어하는 적응형 순항제어 시스템을 단순하게 고려해보면 제어기에 대한 소프트웨어 아키텍처를 아래와 같은 DFD(Data Flow Diagram)으로 표현할 수 있을 것이다.[4]

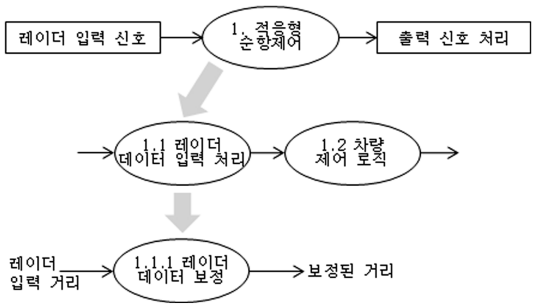

Fig. 2 Data flow diagram of SCC

위 시스템에 대하여 FMEA를 수행하기 위해서는 먼저 일어날 수 있는 결함을 식별해야 한다. SAE ARP 5580을 보면 총 6가지의 결함이 정의된다.[5]

1) 실행되지 않는 결함
2) 불완전하게 실행되는 결함
3) 실행 타이밍이 불안전해서 생기는 결함
4) 에러가 발생하는 결함
5) 리턴되지 않아서 낮은 우선순위의 인터럽트가 실행되는 것을 막는 경우
6) 올바르지 않은 우선순위를 리턴하는 경우

위 SAE 권고사항을 시스템의 1.1.1 레이더 데이터 보정 모듈에 대하여 적용해 보면 다음과 같은 결함의 발생을 예상할 수 있다.

1) 모듈이 실행되지 않음
2) 모듈이 불완전하게 실행됨
3) 모듈의 실행 타이밍이 느리거나 빠름
4) 모듈 실행 시 에러가 발생

레이더 데이터 보정 모듈이 실행되지 않거나 불완전하게 실행되는 경우 보정된 레이더 입력 값이 전해지지 않아 적응형 순항제어 기능이 오작동을 일으킬 수 있다. 또한 레이더 데이터 보정 모듈의 실행 타이밍이 느리거나 빠를 경우 이전에 처리된 데이터가 입력되거나 이후 처리될 데이터가 입력되어 마찬가지 오작동을 일으킬 수 있다. 이는 전체 시스템의 관점에서 잘못된 레이더 데이터 처리로 인하여 선행차량과의 거리 계산이 잘못되어 차량이 비정상적으로 제동이 걸리거나 가속이 가해져 선행차량과의 충돌 또는 급제동과 같은 사고를 일으킬 수 있다. 위 내용을 소프트웨어 FMEA 형식으로 Table 1

Table 1 Result of software FMEA

1차 아이템	1.1 레이더 데이터 입력 처리
결함	레이더 데이터 보정이 안 되어 잘못된 레이더 입력 처리 값을 전달
고장	선행차량과의 거리 계산이 잘못 되어 충돌 발생 가능
원인	레이더 데이터 보정 모듈이 실행되지 않거나 불완전 실행
2차 기능	1.1.1 레이더 데이터 보정
필요 조치	레이더 데이터 보정 모듈의 모니터링 메커니즘을 추가하여 보정 모듈이 정상작동 하는지 검사함

과 같이 정리할 수 있다.

이와 같이 고장과 결함을 식별한 뒤 이 결함을 제거하는 대책을 수립하고 해결하는 과정이 뒤 따르게 된다. 이를 통해 기능 안전 측면에서 소프트웨어의 오작동에 따른 위험성을 줄일 수 있는 강건설계가 이루어지게 된다.

3.2 결함 주입 데이터

결함 주입 기법을 활용하기 위해서는 어떠한 입력 데이터를 결함으로 주입해야 하는가? 결함 주입 기법에서 입력 되는 결함 데이터에는 크게 2가지 방식이 있다. 무작위(Random) 데이터와 부분 유효(Semi-Valid) 데이터 입력 방식이다.

무작위 데이터 입력 방식은 입력 횟수가 많아질수록 더 많은 검증의 정확도를 가질 수 있으나 검증 대상 소프트웨어의 특성을 고려하지 않기 때문에 입력 단계에서 폐기되거나 의미 없는 검증 결과들이 다수 발생할 수 있다. 예를 들면 계층구조로 이루어진 모듈에서 하위 모듈에 대해 결함 주입을 하고자 하는 경우, 무작위 데이터 입력 방식을 사용하는 경우 상위함수에서 이미 무작위 데이터가 폐기되는 상황이 발생할 수 있다. 따라서 무작위 데이터 입력 방식은 경우에 따라 비효율적일 수 있다.

부분 유효 데이터 입력 방식은 검증 대상 소프트웨어의 특성을 분석하여 입력될 결함 데이터를 결정하는 방식이다. 따라서 먼저 대상 소프트웨어의 입력 데이터들에 대한 분석이 필요하다. 본 논문에서는 이 과정을 앞서 언급된 소프트웨어 FMEA 과정과 접목하는 방법론을 제안한다.

앞서 예시를 통해 살펴봤듯이 소프트웨어 아키텍처 설계 과정에서 소프트웨어 FMEA를 수행하려면 소프트웨어의 특성과 나타날 수 있는 결함의 유형들을 식별해야 한다.

소프트웨어 FMEA 과정에서 위 6가지 유형 중 대상 소프트웨어에서 일어날 수 있는 결함을 식별하고 해당 결함에 의해 발생할 수 있는 고장(Failure effects)과 발생하는 원인(cause)을 식별하는 과정을 거친다. 또한 각 결함에 의해 발생하는 리스크에 대해 HARA(Hazard Analysis & Risk Assessment)에서 발생확률, 제어 가능성, 심각도를 평가하여 리스크의 우선순위를 정하는 과정을 거친다.

이와 같은 ISO 26262 표준 프로세스와 FMEA를 통해 설계된 소프트웨어라면 이미 위의 과정에서 각각의 검증대상 소프트웨어 발생할 수 있는 결함들이 식별되어 있다고 볼 수 있다. 또한 개별 결함이 미치는 리스크에 대하여 우선순위가 정해져 있으므로 우선순위가 높은 리스크에 대하여 검증의 강도를 부여한다면 좀 더 높은 신뢰성을 갖춘 검증 결과를 얻을 수 있을 것이다.

3.3 결함 주입 방법

식별된 결함을 주입 하는 방법에는 결함 주입 시기에 따라 컴파일타임 레벨과 런타임 레벨의 2가지 방법이 가능하다.

먼저 컴파일타임레벨에서 결함을 주입하는 경우는 이미 작성된 소프트웨어의 코드를 변형하여 컴파일 후 실행시켜 변형된 코드에 의한 시스템의 결과를 관찰하는 것이다. 실제로 이미 구현된 소프트웨어의 코드가 변형되는 경우는 존재할 수 없지만, 소프트웨어가 실제 ECU에 탑재되어 동작되는 경우 레지스터 값의 변형이 일어나거나 행(Hang)이 걸려 처리되는 값의 지연이 일어날 수 있다. 이러한 현상을 코드 단위에서의 수정을 통해 모사하여 검증하는 것이다. 이때 코드의 변형에는 뮤테이션과 제너레이션(Generation) 두 가지 방법이 있다.[3)]

뮤테이션은 기존의 올바르게 작성된 코드에 대하여 일부를 임의적으로 변형시키는 방법이다. 앞서 예를 들었던 레이더 데이터 보정 모듈에서 결함이 나타났다고 가정해보자. 아래의 Table 2와 같이 코

Table 2 Code mutation examples

변경 전	radar_data = radar_input +100;
변경 후	radar_data = radar_input -100;

드의 일부에 간단한 변형을 줌으로써 소프트웨어의 결함을 유발시킬 수 있다.

실제로 소프트웨어가 작동하는 환경에서는 내외부의 예상치 못한 특정 원인에 의하여 레지스터 값의 변형이 일어날 수 있다. 코드 뮤테이션을 통하여 이러한 레지스터 값의 변형과 같은 경우를 모사하는 것이다.

제너레이션은 기존의 코드에 새로운 코드를 삽입하여 소프트웨어의 동작특성에 변화를 주는 것이다. 예를 들어 레이더 데이터 보정 소프트웨어 모듈의 결함 유형 중 실행 타이밍이 불안전해서 생기는 결함이 있을 수 있다. 이는 어떠한 실행함수에서 빠져 나오지 못하거나 행에 걸려 지연이 발생하는 경우를 생각해 볼 수 있다. 이런 경우를 모사하기 위하여 아래의 Table 3과 같이 기존의 코드에 임의의 1초간의 지연을 발생시키는 함수를 만들어 코드에 삽입할 수 있다.

만약 위와 같은 코드가 삽입된다면 시스템에 지연이 발생하여 결과값을 처리하는데 문제가 생길 것이다. 하지만 이러한 결함 유형에 대하여 제대로 된 강건 설계가 이루어진 소프트웨어일 경우 올바른 타이밍에 결과값이 들어오지 않는다고 판단되면 시스템 전체 처리속도를 늦추거나 다른 경로를 통해 올바른 결과값을 받아들이는 등의 결함 처리 방식을 통해 문제를 해결 할 것이다.

두 번째로 런타임 레벨에서 결함을 주입하는 경우는 소프트웨어를 실행시키며 디버거를 통해 결함을 주입하는 방법이다. 기본적인 결함주입의 개념은 앞서 기술한 코드단위에서의 결함주입과 동일하

Table 3 Code generation examples

변경 전	int radar_data; radar_data = radar_input + const ; return radar_data;
변경 후	int radar_data; radar_data = radar_input + const ; *wait (1000);* return radar_data;

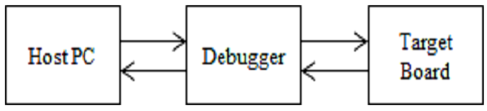

Fig. 3 SFIT environment using debugger

지만 코드가 실행되는 상태에서 디버거를 통해 입력값들을 변경하거나 임의의 시점에서 실행을 지연시키며 그에 따른 결과값을 관찰하여 결함 주입에 따른 결과를 검증할 수 있다. 이때 타깃 보드의 유무에 따라 코드만을 실행하여 컴파일러의 디버깅 상태에서 변수값들을 조정하거나 Fig. 3과 같이 타깃 보드에 소프트웨어를 탑재하여 Trace32와 같은 디버거를 통해 결함을 주입할 수 있다.

특히 타깃 보드에 따라 메모리의 크기, 레지스터 설정 등 소프트웨어의 실행환경이 다르므로 타깃 보드에 소프트웨어를 탑재하여 디버거 장비를 이용하는 경우 좀 더 실제 시스템의 실행 환경에 가까운 검증이 가능하다.

3.4 결함 주입 테스트 프로세스

결함 주입 테스트 프로세스는 Fig. 4와 같이 4단계로 나뉜다.

결함 식별 단계에서는 소프트웨어 FMEA결과에서 결함 발생 시 소프트웨어를 강건하게 유지하기 위해서 소프트웨어 로직을 보강한 결함을 소프트웨어에 주입하는 결함으로 식별한다.

예를 들어 Table 1에서 식별된 결함인 레이더 데이터 보정 올바르게 처리되지 않은 경우 필요조치로 보정 모듈을 추가하는 작업을 하였다. 이 결함은 필요조치를 취했기 때문에 결함 주입 테스트를 통하여 제대로 된 결함 대응 로직이 구현되었는지 확인해야 한다. 따라서 이 결함은 결함 주입 테스트의 결함으로 식별한다.

결함 주입 단계에서는 결함 식별 단계에서 식별된 결함을 소프트웨어에 주입한다. 결함을 주입하기 위해서는 Fig. 5와 같은 소프트웨어 추적 매트릭스(Software Traceability Matrix)를 사용한다. 소프트

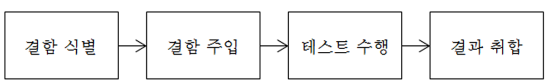

Fig. 4 Fault injection test process

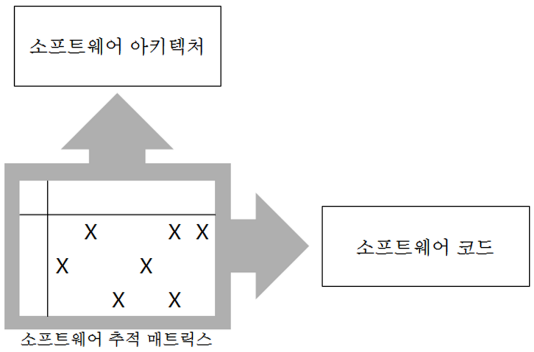

Fig. 5 Software traceability matrix

웨어 추적 매트릭스란 소프트웨어 요구사항에서 소프트웨어 코드까지 연관관계를 매트릭스 형태로 표현한 것으로써 아키텍처 상에서의 모듈이 어떤 소프트웨어 코드로 구현되었는지 확인할 수 있다. 따라서 결함 식별 단계에서 결함을 주입하는 모듈은 소프트웨어 추적 매트릭스를 사용하여 어떤 코드에서 구현되었는지 확인할 수 있다.

코드에 결함 주입은 코드를 변형하는 뮤테이션이나 제너레이션 기법을 사용하여 식별된 결함을 해당 모듈에 삽입한다.

테스트 수행은 결함주입 단계에서 결함을 주입한 소프트웨어를 대상으로 소프트웨어 기능 검증에서 사용하는 테스트케이스를 입력하여 진행한다. 정상적인 소프트웨어의 경우에는 기능 검증에서 사용한 테스트케이스를 입력하였을 때는 예상 결과를 모두 만족할 것이나, 결함을 주입한 소프트웨어의 경우 해당 결함을 처리하기 위해 기능 검증에서 예상한 결과와는 다르게 나올 수 있다. 이러한 결과를 결과 취합 단계에서 기록하여 하나의 결함 주입 테스트를 종료한다.

이상의 총 4단계를 보안조치를 한 모든 결함에 대하여 수행하여 완료하고 다음 결과분석으로 넘어간다.

3.5 결과 분석

끝으로 결함주입기법을 통해 얻은 결과를 분석하는 방법에 대한 논의가 필요하다. 본 논문에서는 결함주입기법의 대상을 소프트웨어 FMEA를 통해 강건 설계가 이루어진 소프트웨어 모듈에 한정하였다. 따라서 입력되는 결함 역시 FMEA과정에서 식별된 결함에 대하여 이루어졌으며 검증 결과는 결함의 처리가 잘 이루어져 전체 시스템의 동작에 영향이 미치지 않는지를 판단하는 방식으로 이루어지면 될 것이다.

앞서 들었던 '레이더 데이터 보정 모듈'을 예로 들어보면 레이더 입력데이터 보정이 불완전하게 이루어지는 결함에 대하여 보정 처리된 입력 값의 레지스터 값에 변형을 주어 결함을 주입하였다. 이러한 결함의 가능성에 대하여 강건 설계가 제대로 이루어졌다면 결함의 주입에 대하여 모니터링 모듈이 정상적으로 작동하여 결함을 제거하거나 데이터를 다른 방식으로 보정하여 전체 시스템이 정상 작동해야 할 것이다. 따라서 결함주입기법의 결과는 결함에 의해 전체 시스템의 안정성에 문제가 발생하는가의 여부를 살펴보면 될 것이다.

결함의 주입과 상관없이 전체시스템이 정상적으로 작동한다면 FMEA과정을 통한 강건설계가 제대로 이루어졌음을 검증할 수 있고 만약 결함 주입에 의하여 전체 시스템의 안정성에 문제가 생겼다면 원인을 파악하고 추가적인 아키텍처 설계를 통해 강건 설계가 이루어지는 과정이 다시 수행되어야 한다. 이러한 과정의 반복을 통해 좀 더 기능 안전을 만족하는 시스템의 구현이 가능할 것이다. 코드 뮤테이션의 경우 코드 변형에 따른 원인 분석이 명확하지 않을 수 있으나 본 논문에서는 하나의 결함에 대해서 코드 뮤테이션을 수행하여 테스트를 수행하기 때문에 결함과 관련된 코드 수정을 쉽게 확인할 수 있다.

4. 결 론

결함 주입 기법은 지금까지 소프트웨어 테스팅을 보완하는 보조적인 검증 방법으로 많은 연구가 이루어져 왔으며 결함 주입이 가능한 인터페이스, 결함 유발 데이터 형성 방법에 대하여 많은 논의가 이루어져 왔다. 하지만 이제 ISO 26262 표준 검증 프로세스에 결함 주입 기법이 포함되며 이제는 필수적인 테스팅 기법이 되고 있다. 이에 기존에 연구된 많은 연구들에 대하여 ISO 표준 개발 프로세스와 접목된 결함 주입 기법의 방법론에 대한 정리가 필요하다.

본 논문에서는 이에 ISO 26262 표준 개발 프로세스 항목 중 소프트웨어 FMEA와 결합하여 결함 주입 기법이 수행 되어야 할 대상, 결함의 식별, 결함 주입 방법을 장애 허용 시스템에 의한 강건 설계가 이루어진 소프트웨어 모듈에 한정하여 결함 주입 기법을 수행하는 방법론에 대하여 서술하였다. 이를 통해 검증 대상과 방법론을 좀 더 명확하게 확정지을 수 있다. 또한 이러한 방법은 소프트웨어 아키텍처에 대한 분석과 코드 단위의 분석을 통해 결함 주입 기법을 활용하므로 화이트박스 테스팅 기법을 바탕으로 한다고 할 수 있어 기존의 블랙박스 테스팅 기법을 주로 사용했던 결함 주입 기법과의 차별성을 꾀하였다.

하지만 아직 주입된 결함에 대한 결과에 대한 정확한 측정 방법을 체계화하고 명시적으로 나타내는 방법에 대한 연구는 부족한 듯하다. 앞으로 이 부분에 대한 활발한 연구가 더욱 진행되어야 할 것이다.

References

1) Mckinsey & Company, Analysis of the Global Dimensional Metrology Market in Electronic Manufacturing, http://www.mckinsey.com, 2007.
2) R. Thorhuus, Software Fault Injection Testing, M. S. Thesis, KTH, Royal Institute of Technology, Stockholm, 2000.
3) K. Kim, Y. Choi, J. Yang and S. Hong, Software Security Testing using Fault Injection, Information Security Academic Journal, National Security Research Institute, 2006.
4) S. Shin and S. Cho, Smart-car Software Engineering, Acon, 2013.
5) SAE International, SAE ARP5580: Recommended Failure Modes and Effects Analysis (FMEA) Practices for Non-automobile Applications, http://standards.sae.org/arp5580, 2001.
6) R. Rana, Improving Fault Injection in Automotive Model Based Development using Fault Bypass Modeling, Computer Science & Engineering, M. S. Thesis, Chalmers, University of Gothenburg, Gothenburg, Sweden, 2013.
7) R. Rana, Increasing Efficiency of ISO 26262 Verification and Validation by Combining Fault Injection and Mutation Testing with Model Based Development, Computer Science & Engineering, M. S. Thesis, Chalmers, University of Gothenburg, Gothenburg, Sweden, 2013.
8) Z. Hong, Integrated Analysis of Software FMEA and FTA, International Conference on Information Technology and Computer Science, Beighang Universty, Beijing, 2009.