안정성평가 기준에 적합한 내장형 소프트웨어 단위시험 절차 방안

A Proposal for Unit Testing Procedure of Embedded Software Complied with Safety Assessment Criteria

장정훈† 이원택* 장주수**

Jeong-Hoon Jang Won-Taek Lee Ju-Su Jang

ABSTRACT

Recently, an important physical device of transportation, such as car, railroad, ship and aircraft has changed into electronic control unit. According to accident reports, the most of car accidents are caused by faults of embedded software loaded to computer control unit. The facts implies that the test to find defects in embedded software haven't performed sufficiently. As a result, it is necessary to establish the test procedures of embedded software based on safety assessment criteria. The objective of this proposal is to provide a unit test procedure complied with the safety assessment criteria for the embedded software. In addition, an effective unit testing procedure and defect analysis methods are proposed and a testing procedure using a safety criteria built-in tool is presented.

국문요약

최근 자동차, 철도, 선박, 항공기 등 교통수단의 핵심 장치가 CPU가 들어 있는 컴퓨터 제어장치로 전환되어 가고 있는 상황이다. 자동차 사고사례 중 상당 부분의 원인으로 컴퓨터 제어장치에 탑재된 내장형 소프트웨어의 결함이라고 분석한 보고서가 발표되고 있다. 내장형 탑재 소프트웨어에 대하여 결함 제거를 위한 시험이 충분하지 못한데서 문제가 있을 수 있다. 결국 이러한 내장형 소프트웨어의 시험 기준으로 안전성 평가 기준을 철저히 적용하고 이에 적합한 단위테스트 절차를 구축하는 것이 필요하다.

본 논문은 안전성평가 기준에 적합한 내장형 탑재 소프트웨어에 대한 단위테스트 절차에 대한 방안을 제시하는데 그 목적이 있다. 내장형 탑재 소프트웨어에 대한 테스트에는 단위테스트, 통합테스트, 시스템테스트 등이 있으며, 이중 가장 많은 결함을 발견하는 것은 단위테스트이다. 본 논문에서는 테스트계획, 테스트절차, 테스트케이스, 테스트시나리오, 테스트도구, 재테스트 기준 등에 대한 효과적인 단위테스트 준비 과정과 단위테스트를 수행하면서 발생하는 업무, 즉 결함식별, 결함분류, 결함분석, 결함원인, 결함조치 및 확인, 재테스트 실시, 테스트보고 등의 활동을 체계적으로 구축하는 방안을 제시한다.

또한 개발된 내장형 탑재 소프트웨어의 프로그램 소스를 일일이 눈으로 검사하여 결함을 발견하기란 쉽지 않다. 안전성평가 기준이 내장된 테스트 도구를 활용함으로써 내장형 탑재

† 정회원, 모아소프트, 신뢰성기술연구실, 책임연구원
E-mail : junghoonjang@moasoftware.co.kr
TEL : (02)420-3203 FAX : (02)407-3511

* 정회원, 모아소프트, 신뢰성기술연구실, 수석연구원

** 정회원, 모아소프트, 연구책임자

소프트웨어 단위테스트의 작업시간을 줄이고 결함발견 및 원인분석에 대한 자동화된 보고서를 얻음으로써 테스트 생산성 및 소스코드의 품질 향상을 달성할 수 있는 방안도 아울러 제시한다.

1. 서론

21세기 들어 자동차, 기차, 선박, 항공기 등의 교통수단에 전자장치로 만들어지는 부품이 주종을 이루며 과거의 기계식 장치를 대체하고 있다. 자동차의 경우 섀시 프레임의 개별제어, 통합제어 및 자율주행 제어로 이어져 탑승자의 안전을 제공하며 편리한 새로운 기능들이 추가되고 있다. 자동차 산업은 90년대 부품 모듈의 개별제어 수준에서 점차 통합 모듈 제어 방식으로 진화되고 있으며, 외관과 디자인 위주에서 점차 편의성이나 안전 등의 서비스 개발로 추진되고 있다. 미래 지능형 자동차는 편의와 안전 등의 서비스들이 더욱 발달할 것으로 전망된다.

그런데 자동차 사고사례 중 상당 부분의 원인으로 컴퓨터 제어장치에 탑재된 내장형 소프트웨어의 결함이라고 분석한 결과가 종종 보고되고 있다. 내장형 탑재 소프트웨어에 대하여 결함 제거를 위한 테스트가 충분하지 못한데서 문제가 있을 수 있다. 결국 이러한 내장형 소프트웨어의 테스트 기준으로 안전성 평가 기준을 철저히 적용하고 이에 적합한 단위테스트 절차를 구축하는 것이 필요하다.

본 논문은 안전성평가 기준에 적합한 내장형 탑재 소프트웨어에 대한 단위테스트 절차에 대한 방안을 제시하는데 그 목적이 있다. 내장형 탑재 소프트웨어에 대한 테스트에는 단위테스트, 통합테스트, 시스템테스트 등이 있으며, 이중 가장 많은 결함을 발견하는 것은 단위테스트이다. 본 논문에서는 테스트계획, 테스트절차, 테스트케이스, 테스트시나리오, 테스트도구, 재테스트 기준 등에 대한 효과적인 단위테스트 준비 과정과 단위테스트를 수행하면서 발생하는 업무, 즉 결함식별, 결함분류, 결함분석, 결함원인, 결함조치 및 확인, 재테스트 실시, 테스트보고 등의 활동을 체계적으로 구축하는 방안을 제시한다.

또한 개발된 내장형 탑재 소프트웨어의 프로그램 소스를 일일이 눈으로 검사하여 결함을 발견하기란 쉽지 않다. 안전성평가 기준이 내장된 테스트 도구를 활용함으로써 내장형 탑재 소프트웨어 단위테스트의 작업시간을 줄이고 결함발견 및 원인분석에 대한 자동화된 보고서를 얻음으로써 테스트 생산성 및 소스코드의 품질 향상을 달성할 수 있는 테스트 방안도 아울러 제시한다.

2장에서는 자동차와 항공기용 내장형 소프트웨어에 대한 안전성 기준으로 ISO/IEC 61508에서의 SIL(Safety Integrity Level)과 DO-178B에서의 소프트웨어 안전성 등급(Software Safety Level)을 살펴보고, 3장에서는 소프트웨어에 대한 테스트 관련 국제 표준을 알아본다. 4장에서는 내장형 소프트웨어에 대한 테스트 방안을 제시하고, 결론 및 향후 과제에 대하여는 5장에 정리한다.

† 정회원, 모아소프트, 신뢰성기술연구실, 책임연구원
E-mail : junghoonjang@moasoftware.co.kr
TEL : (02)420-3203 FAX : (02)407-3511
* 정회원, 모아소프트, 신뢰성기술연구실, 수석연구원
** 정회원, 모아소프트, 연구책임자

2. 내장형 소프트웨어와 안전성

2.1 Functional Safety

ISO/IEC 9126에서는 여러 가지 품질특성을 제시하고 있다[3]. 그중 안전성(Safety)에 대하여 포괄적으로 정의하고 있다. 자동차와 같은 전기전자적인 시스템에 관련해서 안전성에 대해 보다 구체적으로 정의하고 있는데, 전기전자적 안전성 관련 시스템과 관련된 안전성이란 재물이나 환경에 대한 피해의 결과로, 직간접적으로 야기되는 사람 건강에 대한 피해 혹은 물리적 상해에 대해 수용할 수 없는 수준의 위험이 없는 것이라고 정의하며, 인간의 안전성을 확보하기 위해 시스템은 반드시 기능적 안전성(functional safety)을 마련해야 한다고 명시하고 있다.

IEC 61508에서는 그러한 기능적 안전성에 대하여 “입력물에 대하여 올바르게 운영 작동되는 시스템 또는 장비에 따른 전체 안전성 중 일부분이다” 라고 정의하고 있다[1]. 일반적으로 장비의 고장이나 장애가 발생할 경우 그 영향이 사람에게 미치는 경우에 안전성에 문제가 있다고 하며, 안전성을 중요하게 고려하는 경우에는 사람에게 미치는 영향의 정도에 따라 안전성의 수준도 결정되기 때문이다[6,7].

2.2 IEC 61508 과 Software Integrity Level (SIL)

SIL이란, 기능적 안전성 보장수준이며 안전성이 보장되어야 하는 시스템의 신뢰성 수준에 대한 통계적 기준이다. 즉, 안전성 보장이 요구되는 시스템을 운영할 때 발생한 재앙의 발생 건수가 일정 기준 시간 이상 되어야 함을 의미한다. 예를 들어 [도표 1]에서 보는 바와 같이 안전성 수준인 SIL-3은 발생 가능성이 10^{-8} 에서 10^{-7} 범위 안에 있어야 한다.

[도표 1] SIL 기준

Software Integrity Level	Probability of a dangerous failure per hour
SIL-4	$> 10^{-9}$ to $<10^{-8}$
SIL-3	$> 10^{-8}$ to $<10^{-7}$
SIL-2	$> 10^{-7}$ to $<10^{-6}$
SIL-1	$> 10^{-6}$ to $<10^{-5}$

IEC 61508은 그러한 SIL에 대한 4가지 등급별 기준과 각 기준별 달성해야 할 요구사항을 기술하고 있다. IEC 61508 안전 표준은 자동차를 비롯한 항공, 원자력발전 시스템, 기차, 의료 등 안전성 결정적 시스템에서 기본적으로 준용하는 가장 포괄적인 표준이며, 각 도메인에서는 IEC 61508 표준을 근간으로 각 도메인에 최적화된 형태의 특화된 안전성 평가 모델을 제시하고 적용하고 있다.

2.3 DO-178B

2.3.1 DO-178B의 목적

DO-178B는 항공기 관련 부품 및 탑재시스템의 소프트웨어 생산 가이드라인으로서 감항(Airworthiness) 요구사항을 충족하기 위한 기준들을 제시하고 있다[8]. 가이드라인의 대상

이 되는 소프트웨어는 그 목표 기능을 정상적으로 수행함과 동시에 안전에 대한 신뢰성을 보장해야 한다. 감항증명(Airworthiness Certification)은 항공기의 사고방지를 위해 운항에 적합한 자체 안정성과 신뢰성을 갖고 있는지에 대한 증명이다. 대부분의 감항 규정은 소프트웨어에 대한 직접 언급보다는 장비 또는 기체 자체의 안전성이나 설치, 운영과 관련된 조항들을 다루고 있는데 비해 DO-178B는 오직 소프트웨어의 안전성에 대한 인증 기준만을 제시한다.

DO-178B의 주요 내용은 다음의 세 영역으로 구분된다.

1) 소프트웨어 수명주기 프로세스의 목적

2) 목표 달성을 위한 설계 고려사항 및 활동 내용 기술

3) 목표 충족 여부를 결정할 수 있는 증빙 기준의 기술

문서 구조를 간략히 살펴보면 소프트웨어 개발과 관련된 시스템적 측면 (Section 2), 항공기 및 엔진 인증의 개요 (Section 10) 그리고 소프트웨어 수명주기 프로세스(Section 3 ~ 12)에 대한 내용으로 구성된다. DO-178B에서 다루는 수명주기는 계획, 개발, 검증, 형상관리, 품질보증, 인증 등 각 프로세스에 해당하는 개발 기준 및 활동들을 기술하고 있으며, 수명주기 데이터 및 기타 고려사항들을 별도로 다루고 있다.

2.3.2 DO-178B에서의 소프트웨어 안전성 등급

DO-178B는 시스템 안전평가 프로세스를 통해 컴포넌트의 적정 등급 결정하고 있다. 문서 자체는 소프트웨어만을 가이드라인의 범위에 두고 있지만 그 소프트웨어의 등급을 결정하는 것은 시스템 수준의 안전평가 프로세스인 것이다. 안전평가 과정을 거친 안전필수 소프트웨어는 평가 결과로 선정된 원래 목표 등급보다 더 높은 등급으로 개발하는 것이 바람직하다. 수명주기 후반의 등급 상향조정은 더욱 어렵기 때문이다.

소프트웨어가 둘 이상의 시스템 실패조건에 영향을 줄 경우, 가장 높은 등급으로 선정되는데 진화적 시스템 설계 적용 시 소프트웨어 등급은 프로세스를 수행하면서 수정 가능하다. 시스템 기능이 둘 이상의 소프트웨어 컴포넌트로 병렬 개발된 경우 (둘 이상의 소프트웨어 컴포넌트가 비정상 작동해야 실패 조건이 발생) 최소한 하나의 컴포넌트는 시스템 내부 최상위 등급으로 지정되어야 하며 직렬 개발 (하나의 컴포넌트만 실패해도 실패조건이 발생)인 경우 모든 컴포넌트가 최상위 등급으로 지정되어야 한다.

DO-178B는 시스템의 실패 조건을 다음의 5가지로 나누고 있는데 소프트웨어의 실패가 시스템의 어떤 실패 조건을 유발하는가에 따라 소프트웨어의 등급이 결정된다.

1) Catastrophic: 지속적으로 안전한 비행 또는 착륙 불가

2) Hazardous/Severe-Major: 시스템 안전성 또는 기능성의 격감, 승무원의 정상적 임무 수행 불가

3) Major: 항공기의 성능 또는 승무원 임무 수행능력 저하로 인한 항공기 안전성 감소

4) Minor: 심각한 안전성 저해 요건은 아니지만 안전성의 약간의 감소 또는 불편 초래

5) No Effect: 항공기 성능, 승무원 업무 부하에 영향을 주지 않음

이에 따라 DO-178B가 정의하는 소프트웨어 등급 또한 5단계로 나누어진다. [도표2]는 시

스템의 실패 조건과 그와 연관된 소프트웨어의 등급간 관계를 나타낸 것이다. 즉, 특정 소프트웨어의 실패가 “Catastrophic” 실패 조건을 유발할 수 있다면 그 소프트웨어의 등급은 “Level A”에 해당한다.

[도표 2] 소프트웨어 안전성 등급 기준

Software Safety Level	시스템 실패 조건
A	Catastrophic
B	Hazardous/Severe-Major
C	Major
D	Minor
E	No Effect

2.3.3 DO-178B에서의 소프트웨어 테스트 프로세스

DO-178B의 정의에 따르면 검증(verification) 프로세스는 소프트웨어 개발 과정의 에러를 탐지하고 보고하는 과정이다. 단, 에러의 제거는 검증이 아니라 개발 프로세스에 해당하는 활동이다. DO-178B에 명시된 검증 활동은 리뷰(review) 및 분석, 테스트 케이스의 생성과 테스트 절차 수행 등으로 구성되는데 소프트웨어 요구의 구현물과 각 검증 활동 간에는 추적성(traceability)이 보장되어야 한다. 이 때 요구사항과 테스트 케이스 간에는 “Requirement-Based Coverage 분석”을, 그리고 코드와 테스트 케이스 간에는 “Structural Coverage 분석”을 테스트의 요건으로 제시하고 있다.

DO-178B에 명시된 대로 리뷰나 분석 활동을 수행하기 위해서는 피인증자(조직)가 소프트웨어의 요구사항과 아키텍처, 그리고 소스코드에 대한 정확성(accuracy), 완결성(completeness) 및 검증가능성(verifiability)을 제공해야 한다. 테스트 케이스의 생성에 있어서는 내부적인 일치성(consistency)과 요구사항의 완결성(completeness)에 대한 심층적인 분석이 필요하며 테스트 절차를 수행함에 있어 요구사항의 충족(compliance) 여부를 시연할 수 있어야 한다.

일반적인 테스팅 방법론으로는 대상 소프트웨어 내에 에러가 없다는 사실을 증명할 수 없고, 한 번 인증 받은 소프트웨어가 다른 시스템의 일부로 재사용되거나 이미 제품화된 소프트웨어가 약간이라도 변경이 진행된 채로 시스템 내부에 다시 적재되었을 때 시스템 전체를 다시 인증받아야 할지 아니면 바뀐 소프트웨어만 재검증하면 되는지 그 검증 범위에 대한 명확한 해답을 제시하기가 어렵다.

DO-178B 레벨 A 소프트웨어에 대한 코드 검증의 경우 모든 라인에 대해 Modified Condition/Decision Coverage(MC/DC)로 검증되어야 하는데 이러한 테스트는 코드가 복잡도에 따라, 그 노력이 매우 클 수도 있다. 이러한 노력의 양 즉, 비용, 일정의 부담, 그리고 그에 상응하는 완전성을 확보해야 하는 요구에 대하여 소프트웨어 개발 업체들은 아직 이렇다할 해결책이 없는 실정이다.

그러나 FAA/DER과 같은 인증기구 입장에서 시스템 개발 비용이나 일정 등은 안전성 인증에 있어 그다지 중요한 요소가 아니다. 안전성의 보장만이 인증의 가부를 결정하는 기준인 것이다.

3. 테스트 프로세스 관련 표준

3.1 IEEE 829: Software Test Documentation

IEEE 829는 일반적인 SW 테스트 프로세스와 산출물에 대한 표준을 제시하고 있다. 이러한 산출물을 생성하기 위해서는 일련의 테스트 활동을 수행하여야 하며 이는 곧 테스트 프로세스를 정의하고 있다는 것과 같다. [도표 3]에서는 IEEE 829에서 요구하고 있는 산출물과 이에 상응하는 테스트 활동을 보여주고 있다.

[도표 3] 소프트웨어 테스트 산출물

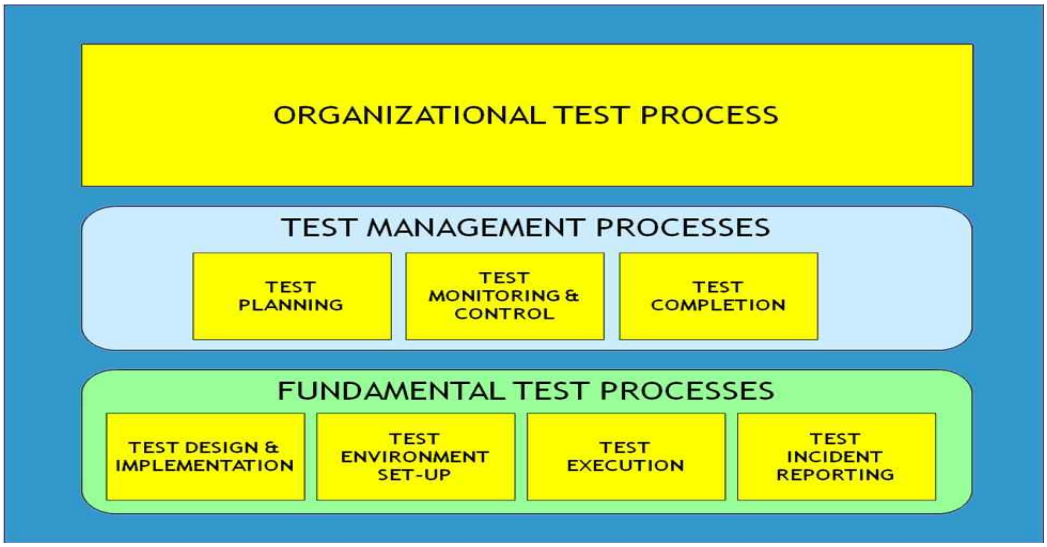
Software Test Documents	관련 활동
Test Plan	테스트 계획 수립
Test Design Spec	테스트 설계
Test Case Spec	테스트 케이스 작성
Test Procedure Spec	테스트 절차 정의
	테스트 실행
Test Log	테스트 결과 정리
Test Incident Report	테스트 결함 분석
Test Summary Report	테스트 결과 보고

내장형 소프트에어가 안전성 기준을 만족하기 위해서는 테스트 실행과정이 좀더 구체적이고 상세화되어야 하며, 검증해야 할 기준 또는 달성해야할 테스트 목표가 매우 중요하다. 즉, 테스트 케이스를 생성하는 경우에 요구사항 충족여부, 설계 충족여부, 소스코드의 정확성 및 일치성 등 기본적인 요건뿐만 아니라 앞서 DO-178B에서의 검증요건인 각종 Test Coverage를 만족하는 테스트가 수행되어야 한다.

3.2 ISO/IEC 29119: Software Test Standard

ISO/IEC 29119는 SW 테스트 프로세스에 대한 표준을 제시하고자 만들어졌다[2]. 크게 4개 Part로 나누어져 있는데, Part 1에서는 기본 개념과 용어 정의, Part 2에서는 테스트 프로세스, Part 3에서는 테스트 문서, 그리고 Part 4에서는 테스트 기법들을 소개하고 있다. Part 2의 테스트 프로세스는 테스트 관리 프로세스와 테스트 실행 프로세스로 구분하고 이들 상호간의 작용을 나타내고 있다. [그림 1]에서는 ISO/IEC 29119에 대한 전체적인 구성을 보여주고 있다.

ISO/IEC 29119의 표준은 역시 전체적인 절차와 산출물, 기법 등에 대하여는 내장형 소프트웨어에 대한 테스트 프로세스에도 적용할 수 있지만, 소프트웨어의 안전성을 보장하는 테스트 절차로는 미흡하다. 단순한 테스트 활동보다는 소프트웨어의 안전성을 보장할 수 있는 테스트 케이스에 대한 요건을 구체적으로 제시하는 기준을 표준화하는 것이 필요하다. 또한 내장형 소프트웨어가 갖는 도메인에 따라 좀더 특성에 맞는 테스트를 수행하도록 하드웨어 탑재 후의 소프트웨어 실행환경에서의 테스트에 대한 요건도 필요하다.



[그림 1] ISO/IEC 29119 테스트 프로세스 구성도

3.3 TMMI (Test Maturity Model Integration)

TMMI는 소프트웨어 테스트조직의 성숙도를 평가하고 프로세스를 개선하기 위한 목적으로 개발된 모델로, 5단계 테스트 조직 성숙도로 평가된다. SW는 보통 분석/설계-개발-테스트의 과정을 통해 완성된다. 이때 SW의 품질과 안정성을 보장하기 위해서는 수준 높은 테스트 역량이 필수다. 그 동안 SW개발 전반에 대한 품질인증으로는 CMMI(Capability Maturity Model Integration)가 널리 사용돼 왔다. 하지만 CMMI는 소프트웨어 품질 관리의 핵심인 테스트 분야의 평가에 취약하다는 단점이 지적돼 왔다. TMMI는 이러한 CMMI의 취약점을 보완하기 위해 개발되었다. TMMI의 평가항목은 [도표 4]에 나타나 있다.

[도표 4] TMMI 인증 등급별 프로세스 영역

Level	프로세스 영역
Level 2 (Managed)	테스트 정책과 전략
	테스트 계획
	테스트 모니터링 및 통제
	테스트 설계 및 수행
	테스트 환경
Level 3 (Defined)	테스트 조직
	테스트 교육 훈련
	테스트 수명주기와 통합
	비기능 테스팅
	동료검토
Level 4 (Management and Measurement)	테스트 측정
	소프트웨어 품질평가
	발전된 동료검토
Level 5 (Optimization)	결함예방
	테스트 프로세스 최적화
	품질 제어

그러나 TMMI 인증을 받는다는 것은 테스트 조직의 성숙도와 역량을 평가하는 것이며 이 조직이 개발하는 소프트웨어에 대한 품질 수준을 보장하는 것은 아니다. 소프트웨어의 안전성은 조직의 역량에 따라 좌우되기는 하지만 해당 소프트웨어 자체에 대한 안전성 테스트를 실시하여 합격해야만 하는 것이다. 따라서 TMMI 테스트 평가 모델은 테스트 프로세스에 대한 요건만 제시할 뿐 개발된 소프트웨어의 안전성을 보장하는 기준으로는 부족하다.

4. 내장형 소프트웨어 테스트 방안

앞서 알아본 내장형 소프트웨어에 대한 안전성 기준 및 표준 테스트 프로세스는 소프트웨어에 대한 직접적인 안전성을 보장해주지 못한다. 본 논문에서는 이러한 문제를 해결하고자 내장형 소프트웨어의 안전성 기준을 탑재하고 있는 테스트 툴인 LDRA의 주요 기능을 설명함으로써 내장형 소프트웨어에 대한 테스트 방안을 제시하고자 한다[4].

4..1 Requirements Workflow

요구사항은 만족하는 소프트웨어가 개발될 수 있도록 요구사항 추적성을 지원하고 요구사항 적합성을 검증하며 또한 소스 및 실행코드의 요구사항 Coverage를 제공해준다.

4.2 Design Review

내장형 소프트웨어의 call graph와 data flow graph를 제공하여 자동적으로 함수 및 기능간 인터페이스를 검증할 수 있게 한다. 또한 run time error를 찾아내는 기능을 제공한다.

4.3 Code Review

프로그래밍 언어에 대한 표준 룰을 검사하여 준다. 대표적인 코딩 룰인 MISRA-C/C++를 내장하고 있어 자동차 용 전장 소프트웨어의 안전성 테스트에 적합하며, 개발자가 임의로 코딩 룰을 생성하는 것도 가능하다[5].

4.4 Quality Review

소스 코드에 대한 분석을 통하여 testability, complexity 등의 지표를 제공한다. 지표에 대한 어느 정도의 수준이 달성되어야만 기능 테스트를 진행할 수 있다.

4.5 Unit Test

개별 기능에 대한 테스트 환경, 테스트 케이스 생성, 테스트 이력관리가 가능하며 structural coverage에 대한 수준을 제공한다.

4.6 Target Testing

내장형 소프트웨어에 대한 타겟 테스팅이 지원되며, 실행 프로그램이 하드웨어에 탑재된 상태에서의 테스트가 가능하다.

4.7 Test Verification

여러 가지 coverage에 대한 지표가 제공된다. 특히 MC/DC 등 DO-178B의 항공용 내장형 소프트웨어 안전성 A 등급에 해당하는 테스트 및 검증 기준을 모두 만족시키는 테스트를 수행할 수 있다.

4.8 Test Manager

전체적인 테스트 프로세스를 관리한다. 테스트 절차와 테스트 케이스에 대한 베이스라인 관리를 통하여 일정 시점 이전으로의 회귀 테스트가 가능하다.

5. 결론 및 향후 과제

본 논문에서는 안전성평가 기준에 적합한 내장형 탑재 소프트웨어에 대한 단위시험 절차에 대한 방안을 제시하였다. 내장형 탑재 소프트웨어에 대한 테스트를 위해서는 테스트계획, 테스트절차, 테스트케이스 작성, 테스트 시나리오 작성, 테스트 도구 준비 및 활용, 재시험기준 등에 대한 효과적인 테스트 프로세스가 필요하며, 이에 대한 테스트 관련 국제 표준을 적용하면 된다. 다만 안전성을 만족하는 소프트웨어를 개발하기 위해서는 테스트 프로세스 보다는 테스트 기준이 더 중요하다, 안전성 등급에 따른 테스트 기준은 달라지지만 최고 수준의 안전성을 확보할 수 있는 테스트에는 많을 노력과 비용이 수반된다.

이러한 안전성 기준에 적합한 내장형 소프트웨어의 단위 테스트 방안으로는 효과적인 테스트 툴을 활용하는 것이 중요하며 효율적인 테스트 업무 수행을 위해서도 필요하다. 본 논문에서는 예를 들은 LDRA 테스트 툴이 갖고 있는 기능들은 이러한 두 가지 목적을 모두 달성시켜줄 수 있는 좋은 툴로 판단하여 이러한 툴을 활용한 테스트 방안 수립을 제시한다.

그러나 테스트 툴을 활용하는 개발자 또는 테스터의 역량 또한 중요하다. 툴이 갖고 있는 좋은 기능을 충분히 사용활 수 있도록 꾸준한 노력과 경험이 필요하다.

참고문헌

1. IEC 61508, IFunctional Safety of Electrical/Electronic/Programmable Electronic Safety Related Systems. IEC, Geneva., 2000.
2. ISO/IEC 29119, Software Test Standard, ISO 2009
3. ISO/IEC 9126, Software Engineering - Product Quality, ISO/IEC TR 9126-4, 2004
4. LDRA Tools Presentation v16.3, LDRA, 2009
5. MISRA. Development Guidelines for Vehicle Based Software. November 1994.
6. NASA. Software Safety: NASA Technical Standard NASA-STD-8719.13B. July 2004.
7. NASA. Software Safety Guidebook, NASA-GB-8719.13, March 2004.
8. RTCA. SW Considerations in Airborne Systems and Equipment Certification RTCA/DO-178B. 1994.