정 보 보 호 학 회 지
제28권 제6호, 2018. 12

무기체계 소프트웨어 신뢰성 시험 현황 및 발전방향

이 태 호*, 백 옥 현*, 김 태 현*

요 약

무기체계의 대형화 및 복잡도가 증가함에 따라 무기체계에서 소프트웨어의 비중이 큰 부분을 차지하고 있으며 이로 인해 소프트웨어 결함에 따른 잠재적인 위험성도 증가하고 있다. 무기체계 분야에서는 무기체계 소프트웨어 개발 및 관리 매뉴얼을 통하여 개발 단계별 활동 및 검토사항들을 정의함으로써 단계별로 소프트웨어의 품질을 확보할 수 있도록 하는 한편, 소스코드 내의 잠재적인 결함을 체계적으로 제거하여 품질을 높일 수 있도록 소프트웨어 신뢰성 시험 제도를 도입하여 운영하고 있다. 본 글에서는 무기체계 소프트웨어 신뢰성 시험 현황에 대하여 소개하고 발전방향을 제시하고자 한다.

Ⅰ. 서 론

무기체계의 복합화·첨단화에 따라 소프트웨어가 차지하는 비중이 점점 높아지고 있으며, 소프트웨어로 구현되는 요구사항의 비중과 소프트웨어의 복잡도 및 규모가 점점 커지고 있는 추세이다. 최근 개발 완료된 A 유도무기 체계의 소프트웨어 규모는 8,500 KSLOC 수준이며, 현재 개발 중인 B 전투체계의 경우 2,800 KSLOC에 이르고 있다.

1991년 걸프전 당시 패트리어트 미사일 시간 측정 소프트웨어 결함으로 인한 요격 실패로 인해 인명 손실(미군 28명 사망, 97명 부상)이 발생한 사건은 대표적으로 알려진 무기체계 소프트웨어 결함 사례이다. 이처럼 무기체계 소프트웨어는 결함으로 인한 사고 발생 시 막대한 인명 손상과 재산상의 피해를 가져올 수 있기 때문에 무기체계 소프트웨어 품질에 대한 기준 강화가 요구되고 있다.

방위사업청에서는 무기체계 소프트웨어 품질향상을 위해 ISO/IEC 12207을 참조한 무기체계 소프트웨어 개발 프로세스를 정립하여 시행하고 있다[1]. 무기체계 소프트웨어 개발 프로세스는 소프트웨어 품질을 높이기 위한 활동의 일환으로 무기체계 소프트웨어에 잠재되어 있는 결함을 개발과정에서 사전에 제거하는 활동을 강조하고 있다. 방위사업청은 계획 수립 단계부터 잠재되어 있는 결함을 체계적으로 제거하도록 하는 활동을 소프트웨어 신뢰성 확보 활동으로 정의하고 무기체계 소프트웨어 신뢰성 시험을 제도화하여 운영하고 있다.

본 글에서는 무기체계 분야에 적용하고 있는 소프트웨어 신뢰성 시험에 대한 개요, 시험항목의 구성 및 내용 그리고 소프트웨어 신뢰성 시험의 주요 보완사항 및 발전방향에 대하여 살펴보고자 한다.

Ⅱ. 무기체계 SW 신뢰성 시험 개요

무기체계 소프트웨어 신뢰성 시험은 소프트웨어가 동작할 수 있는 다양한 경우의 수를 확인함으로써 소프트웨어가 일으킬 수 있는 결함을 식별하는 시험을 말한다. 무기체계 소프트웨어 신뢰성 시험이라는 용어는 무기체계 분야에서 고유하게 쓰이는 용어로 ISO 25000 시리즈에 정의된 품질특성 중의 하나인 '신뢰성(Reliability)' 이라는 용어와는 다소 차이가 있다. 이는 소프트웨어 테스팅 분야에서 쓰이는 정적·동적 시험 개념을 무기체계 분야에 적용하면서 붙여진 명칭이다.

무기체계 소프트웨어 신뢰성 시험의 적용 대상 언어는 C, C++, Java, C#이며 소프트웨어의 개발 형태에 따라 시험 여부를 정하고 있다. 개발한 소프트웨어, 공개 소프트웨어, 자동생성 코드에 대해서는 SW신뢰성시험 대상으로 식별하며 기존에 무기체계 소프트웨어 신뢰성 시험을 수행한 소프트웨어와 상용 소프트웨어는 시험을 수행하지 않는다. 다만, 공개 소프트웨어에 대해서는 신뢰성이 확보된 SW로 판단될 경우 시험 대상에서 제외할 수 있으며, 자동생성 코드에 대해서도 타당한 사유가

* 국방과학연구소 공용기술센터 SW신뢰성기술실 (lth2094@add.re.kr, ohpaek@add.re.kr, tae_hyoun@add.re.kr)

있을 경우 제외할 수 있다.

본 장에서는 무기체계 소프트웨어 개발 프로세스 각 단계에서 수행되어야 하는 무기체계 소프트웨어 신뢰성 시험 활동에 대해 개략적으로 설명하며 실제 시험 항목에 대한 자세한 내용은 Ⅲ장에서 다룬다.

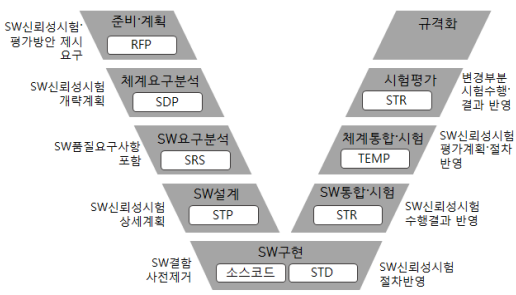
무기체계 소프트웨어 개발 프로세스에서 무기체계 소프트웨어 신뢰성 시험이 실제 수행되는 단계는 구현 및 시험 단계이다. 하지만 시험 수행을 위해서는 사업의 추진 준비 및 계획 단계에서부터 이와 관련된 사항을 고려한 활동들이 필요하다. [그림 1]은 무기체계 소프트웨어 개발 단계 별로 무기체계 소프트웨어 신뢰성 시험과 관련 활동을 요약한 그림이다.

먼저 사업 추진 준비 단계에서는 제안요청서에 무기체계 소프트웨어 신뢰성 시험 및 평가 방안을 제시하도록 요구해야 한다.

사업 착수 후, 체계 요구사항 분석 단계에서는 관련 산출물로 소프트웨어 개발계획서(SDP, Software Development Plan)를 작성한다. 이때 무기체계 소프트웨어 신뢰성 시험 적용대상 소프트웨어 형상항목(CSCI, Computer Software Configuration Item)을 식별하고 개략적인 무기체계 소프트웨어 신뢰성시험 계획을 포함하여 소프트웨어 개발계획서를 작성해야 한다.

소프트웨어 요구사항 분석 단계에서는 소프트웨어요구사항명세서(SRS, Software Requirement Specification)를 작성하며 이때 소프트웨어 품질 요소와 관련된 요구사항에 무기체계 소프트웨어 신뢰성 시험 관련 요구사항을 포함하여 작성한다.

무기체계 소프트웨어 신뢰성 시험의 상세한 계획은 소프트웨어 상세설계 단계에서 확정된다. 이때 시험 대상 소프트웨어와 기준, 일정, 방법, 도구, 환경 등 무기체계 소프트웨어 신뢰성 시험 계획을 구체화하여 소프트웨어통합시험계획서(STP, Software Test Plan)에 기술한다.



〔그림 1〕 개발단계별 SW신뢰성시험 관련 활동

소프트웨어 구현 단계에서는 코딩 규칙에 따라 소프트웨어를 개발하고 무기체계 소프트웨어 신뢰성 시험을 수행하는 등 소프트웨어의 결함을 사전에 제거하기 위한 활동을 지속적으로 수행한다. 또한 소프트웨어통합시험절차서(STD, Software Test Description)에 무기체계 소프트웨어 신뢰성 시험 절차를 반영한다. 이후 소프트웨어 통합 및 시험 단계에서 소프트웨어 통합시험결과서(STR, Software Test Report)에 무기체계 소프트웨어 신뢰성 시험 결과를 작성한다.

체계통합 및 시험 단계에서는 시험평가 준비를 위해 시험평가기본계획(TEMP, Test and Evaluation Master Plan)에 무기체계 소프트웨어 신뢰성 시험을 개발시험평가항목으로 포함 하고 시험평가 활동을 수행한다. 이후 개발/운용시험평가 과정에서 소프트웨어의 형상이 변경되었다면 변경된 부분에 대해서 무기체계 소프트웨어 신뢰성 시험을 수행하고 관련 산출물에 결과를 반영한다.

Ⅲ. 무기체계 SW 신뢰성 시험의 구성 및 내용

본 장절에서는 무기체계 SW 신뢰성 시험의 구성 및 내용에 대하여 간단히 살펴보도록 한다. 본 장절의 내용은 2018년 11월에 개정된 무기체계 소프트웨어 개발 및 관리 매뉴얼을 기준으로 작성되었다[1]. 매뉴얼에서 정의하고 있는 무기체계 SW 신뢰성 시험은 크게 SW 정적시험과 SW 동적시험으로 구성된다.

3.1. SW 정적 시험

일반적으로 SW 정적 시험이라 함은 코드를 실행하지 않은 상태에서 수행하는 시험을 말하며 이는 크게 검토(Review) 활동과 정적 분석(Static Analysis) 활동으로 구분된다. 여기서 검토 활동이라 함은 사람이 수기로 문서상의 오류를 찾아 수정하는 활동을 말하며 정적 분석 활동은 주로 도구를 이용하여 코드 상의 구조적 결함을 찾아 수정하는 활동을 말한다.

방위사업청의 무기체계 소프트웨어 개발 및 관리 매뉴얼에서는 앞서 언급한 두 가지 정적 시험 활동 중 도구를 이용하여 수행하는 정적 분석활동을 주된 시험 활동으로 요구하고 있으며 세부 활동으로 코딩규칙 검증,

취약점 점검, 소스코드 메트릭 점검 활동을 요구하고 있다[1]. 각 세부 항목에 대한 내용은 이하 장절에서 설명하도록 한다.

3.1.1. 코딩 규칙 검증

무기체계 소프트웨어 코딩 규칙이란 무기체계 소프트웨어의 품질 및 신뢰성 향상을 위해 프로그램 코딩 시 개발자가 준수해야 하는 규칙을 말한다. 이는 2011년 제정된 방위사업청의 무기체계 소프트웨어 개발 및 관리 지침[2]에서 처음 소개되었으며 적용 범위 및 규칙과 관련된 내용이 현재까지 지속적으로 발전되어오고 있다.

2011년에 처음 제시된 무기체계 소프트웨어 코딩 규칙은 주석 및 명명 규칙 외에 C와 C++ 언어에 대해 총 65개의 프로그램 작성 규칙으로 이루어져 있다. 이는 C와 C++ 언어 공통으로 적용 가능한 규칙 45개, C 전용 규칙 5개 그리고 C++ 전용 규칙 15개로 구성되어 있다. [표 1]은 초기 프로그램 작성 규칙에 대한 간략한 분류표이다.

이 후 2016년에 개정된 무기체계 소프트웨어 개발 및 관리 매뉴얼[3]에서는 코딩 규칙 적용 언어의 범위를 C, C++, C# 그리고 JAVA로 확대하였으며 이를 위해 기존의 65개 코딩 규칙이 아닌 다양한 분야에서 많이 적용되고 있는 코딩 규칙 항목들을 선별하여 이들 중 체계 사업의 특성에 가장 적합한 항목을 준수하도록 내용이 수정되었다. 다만 부품국산화 등 상기 항목에 대한 적용이 제한되는 사업의 경우에는 기존 65개 코딩 규칙과 스타일 관련 규칙 1개를 추가하여 총 66개의 규칙을 준수토록 요구하고 있다. [표 2]는 최신 매뉴얼에서 요구하고 있는 코딩규칙 항목에 대한 표이다.

〔표 1〕 2011년 지침 기준 프로그램 작성 규칙 분류〔2〕

규칙 구분		개수
공통	스타일	11
	초기화	3
	식별자	4
	조건식	5
	변환	8
	포인터/배열	5
	연산자	9
C 언어 전용		5
C++ 언어 전용		15

[표 2]에 제시된 코딩 규칙 항목은 기존 65개 코딩 규칙보다 많은 코딩규칙을 포함한다. 특히 C, C++언어와 관련된 MISRA-C, MISRA-C++은 자동차 분야에 적용되는 규칙으로 해당 규칙을 모두 준수하기 위해서는 기존 65개 규칙을 준수하였을 때보다 상당히 많은 비용과 일정이 요구된다. 이를 위해 방위사업청 매뉴얼에서는 [표 2]에 제시되어 있는 코딩 규칙 항목 준수 시 조정이 필요한 항목에 대하여 사유를 제시하고 사업관리회의를 통해 이에 대한 결정을 확정하도록 하고 있다[1]. 무한한 시간과 비용은 현실에서는 존재하지 않는다. 제한된 자원을 가지고 최적의 성과를 내기 위한 의사결정을 다루는 소프트웨어 공학 경영[7] 관점에서 볼 때, 사업 담당자는 사업의 일정 및 비용을 고려하여 필요에 따라 코딩 규칙 준수 시 조정이 필요한 항목을 선별하고 적용 범위를 조정할 필요가 있다.

〔표 2〕 최신 무기체계 코딩 규칙 항목〔1〕

언어	규칙 명
C	MISRA(Motor Industry Software Reliability Association)-C : 2012 [4]
C++	MISRA-C++ : 2008 [5]
	JSF++(Joint Strike Fighter Air Vehicle C++) [6]
C#	C# Coding Convenstions (MicroSoft)
	.Net Framework Design Guideline(MicroSoft)
JAVA	Code Convenstions for the JAVA Programming Language(Oracle)

3.1.2. 취약점 점검

무기체계 소프트웨어 분야에서 취약점 점검이란 미국 국토안보부(국가사이버보안국) 산하의 비영리 공익단체인 MITRE가 제공하는 공통적인 취약점 목록(CWE, Common Weakness Enumeration)을 점검하는 활동을 말한다. MITRE에서는 CWE 제공을 위한 사이트(http://cwe.mitre.org)를 운영 중이며 지속적으로 공통적인 취약점 목록을 최신화 해나가고 있다.

취약점 점검 활동은 코딩 규칙 검증과 마찬가지로

2011년 제정된 방위사업청의 무기체계 소프트웨어 개발 및 관리 지침에서 처음 소개되었다[2]. 초기에 이 활동의 명칭은 '실행시간오류(Runtime error) 검출'이었으나 2016년 개정된 무기체계 소프트웨어 개발 및 관리 매뉴얼에서 '취약점 점검'으로 명칭이 수정되었다[3].

2011년에 처음 제시된 취약점 점검 활동은 C와 C++ 언어에 대하여 각 언어에 대한 공통적인 취약점을 목록화한 CWE-658와 CWE-659을 준수하도록 요구하고 있다. 2016년 이후 부터는 코딩 규칙 검증과 마찬가지로 적용 언어를 확대하여 JAVA에 대한 공통 취약점 항목인 CWE-660까지 준수토록 요구하고 있다. 다만 C#과 관련된 공통 취약점 목록은 제공하고 있지 않아 취약점 점검 활동의 적용 대상 언어는 C, C++ 그리고 JAVA로 한정되어 있다.

3.1.3. 소스코드 메트릭 점검

무기체계 소프트웨어 분야에서 소스코드 메트릭이란 소프트웨어의 복잡도를 감소시키고 유지보수성을 높이는 등 소프트웨어의 품질 향상을 위해 준수해야하는 소스코드 품질 측정 지표를 말한다. 소스코드 메트릭은 2016년 개정된 무기체계 소프트웨어 개발 및 관리 매뉴얼에서 처음 요구되었으며 이에 대한 종류 및 제한 값은 [표 3]과 같다.

일반적으로 소프트웨어 신뢰성 메트릭과 관련된 국제 표준에서는 평균 고장 시간(MTBF, Mean Time Between Failure), 고장률(Failure Rate) 등과 같은 시스템 수준의 메트릭을 소프트웨어 신뢰성 관련 메트릭으로 제시하고 있다[8][9]. 반면 무기체계 SW 신뢰성 시험 항목인 소스코드 메트릭 점검 활동은 시스템 수준의 메트릭보다 소스코드 수준의 메트릭에 더 중점을 두고 있다는 특징을 갖는다.

〔표 3〕 무기체계 소스코드 메트릭

메트릭 명	제한 값
Cyclomatic Complexity	20 이하
Number of Call Levels	6 이하
Number of Function Parameters	8 이하
Number of Calling Functions	8 이하
Number of Called Functions	10 이하
Number of Executable Code Lines	200 이하

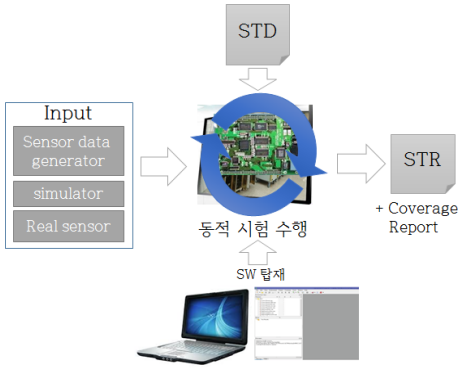
매뉴얼에서 요구하는 소스코드 메트릭 종류는 총 6가지이며 이들은 C언어와 같은 절차지향 프로그래밍 언어(Procedural Programming Language)에는 적합한 규칙이지만 C++, C#, JAVA 등과 같은 객체지향 프로그래밍 언어(Object-Oriented Programming Language)에서는 다소 적용하기가 어려운 항목들이다. 이러한 문제를 해결하기 위해 방위사업청 매뉴얼에서는 [표 3]에 제시된 제한 값 적용이 제한되는 경우 연구개발주관기관에서 타당한 사유를 제시하고 이에 대해 사업관리회의를 통해 의사결정을 할 수 있음을 명시하고 있다.

소스코드 메트릭의 제한값은 일반화가 어려우며 조직, 사업의 특성 및 개발환경에 의존적이다[10][11]. 소스코드 메트릭의 제한 값과 관련된 기존의 연구문헌에서는 제한값을 결정하는 대표적인 방법으로 과거 유사 사업들의 결과를 바탕으로 제한 값을 결정하는 방법을 제시하고 있다[11]. 하지만 소스코드 메트릭 점검 활동은 무기체계 SW 정적 시험 항목 중 가장 최근에 추가된 활동으로 아직까지는 이를 적용한 사업이 많지 않다. 따라서 소스코드 메트릭 점검 활동이 앞으로 더욱 의미있는 활동이 되기 위해서는 여러 사업의 데이터 축적과 이에 대한 분석 활동이 필요하다.

3.2. SW 동적 시험

일반적으로 SW 동적 시험이라 함은 소프트웨어가 컴파일 되고 실제 동작하는 상태에서 수행하는 시험을 말한다. SW 동적 시험의 주요 활동은 소프트웨어에 특정 테스트 케이스 입력 값을 주입하였을 때 나오는 결과 값이 기대 값과 동일한지를 확인하는 활동이다. 이외에도 SW 동적 시험에는 코드 실행률 혹은 요구사항 실행률을 분석하는 실행률 분석 활동(Coverage Analysis) 등이 포함된다.

방위사업청의 무기체계 소프트웨어 개발 및 관리 매뉴얼에서는 [그림 2]와 같이 실제 하드웨어에 소프트웨어를 탑재하여 요구사항 기반으로 작성된 시험 절차에 따라 시험을 수행하였을 때 얻어지는 소스 코드 실행률을 점검하는 활동을 주된 SW 동적 시험 활동으로 정의하고 있다[1]. 2011년 제정된 방위사업청의 무기체계 소프트웨어 개발 및 관리 지침[2]에서 처음 소개된 이 활동은 초기에는 단순히 소스코드의 구조적 실행률 달성 여부를 판단하는 활동에만 초점이 맞추어져 있었으



〔그림 2〕 소프트웨어 동적 시험

나 최근에는 테스트케이스 입력 값에 대한 기대 값과 실제 결과 값의 비교 활동 또한 중요시 되고 있다. 이는 소프트웨어 안전성과 관련된 국제 표준에서 제시하는 요구사항 기반의 시험 활동 내용과 유사하다고 할 수 있다[12][13].

방위사업청의 매뉴얼에 제시된 SW 동적 시험의 종류로는 문장 실행률(Statement Coverage), 분기 실행률(Branch Coverage), MC/DC 실행률 (Modified Condition and Decision Coverage)이 존재한다. 사업 담당자는 개발하고자 하는 소프트웨어에서 발생할 수 있는 결함의 발생빈도, 영향성 및 제어가능성을 평가하여 매뉴얼에 제시된 등급 분류표에 따라 해당 등급에 맞는 실행률을 선정해야 한다. [표 4]는 매뉴얼에서 제공하고 있는 SW 동적 시험을 위한 등급 분류표이다.

〔표 4〕 SW 동적 시험을 위한 등급 분류표

결함 영향도 (Severity)	결함 발생빈도 (Exposure)	결함 제어가능성 (Controllability)		
		C1	C2	C3
S1	E1	S	S	S
	E2	S	S	S
	E3	S	S	S
	E4	S	S	S
S2	E1	S	S	S
	E2	S	S	S
	E3	S	S	S
	E4	S	S	B
S3	E1	S	B	B
	E2	B	B	B
	E3	B	B	B
	E4	B	B	M
S4	E1	B	B	M
	E2	B	M	M
	E3	B	M	M
	E4	M	M	M

* S:문장(Statement), B:분기(Branch), M:MC/DC

결함 발생 빈도	
수준	내용
E1	매우 낮은 확률
E2	낮은 확률
E3	중간 확률
E4	높은 확률
결함 영향도	
수준	내용
S1	무시 가능
S2	사소함
S3	심각함
S4	치명적
결함 제어 가능성	
수준	내용
C1	간단히 제어가능
C2	보통의 경우 제어가능
C3	제어하기 어렵거나 불가능

[표 4]에서 보는 바와 같이 SW 동적 시험을 위한 등급 분류 기준은 정량적이기 보다는 정성적인 기준이라고 할 수 있다. 따라서 사업 담당자의 등급 분류 활동이 무엇보다 중요하며 이를 위해서 시스템 수준에서부터 소프트웨어 수준까지 결함의 유형 및 영향도에 대한 세부적인 분석이 요구된다.

Ⅳ. 발전방향

지금까지 무기체계 소프트웨어 신뢰성 시험의 현황에 대해 살펴보았다. 정적 시험을 수행하면서 개발자가 발견하지 못했던 결함을 사전에 식별하고, 동적 시험을 통해서 데드 코드(dead code) 발견, 적절한 예외처리 상황 등 개발자가 미처 인지하지 못한 다양한 경우에 대한 시험 수행으로 소프트웨어 품질 향상에 기여함을 확인할 수 있었다.

그러나 무기체계 소프트웨어 신뢰성 시험이 보다 실질적인 소프트웨어 품질 향상에 기여하는 제도가 되기 위해서는 몇 가지 사항이 보완되어야 한다. 이 장에서는 다양한 무기체계에 대한 소프트웨어 신뢰성 시험을 수행하는 과정에서 인식한 보완사항 및 발전방향을 제시하고자 한다.

첫째, 시험 수행을 위한 충분한 예산, 기간, 인력이 반영되지 않고 있다. 사업 준비단계에서 시험의 범위와 요구 수준이 구체화되지 않아 시험에 소요되는 예산 및

기간에 대한 계획수립에 어려움이 있다. 사업계획 수립 시부터 시험의 범위와 수준을 구체화하는 노력이 필요하다. 또한, 소프트웨어 신뢰성 시험을 준비할 수 있는 예산 및 기간 등이 충분히 반영될 수 있는 정책적인 지원이 필요하다.

둘째, 테스팅 도구를 활용하여 시험 수행 시 적게는 몇백개에서 많게는 수십만개까지의 수많은 정탐(true positive, 실제 오류가 존재해서 보고하는 경우)과 오탐(false positive, false alarm이라 불리기도 함. 실제 오류가 존재하지 않지만 오류라고 보고하는 경우)이 발생한다. 개발자는 이들 경고(Alarm)에 대한 분석 및 코드수정 활동을 수행하는데 많은 시간을 투자하게 된다. 정탐에 대한 코드 수정은 테스팅의 긍정적인 효과이지만, 불필요한 오탐이 많을 경우 시험평가 합격을 위해 오탐에 대한 설명자료 작성 등 불필요한 소명 노력이 과다하게 발생하게 된다. 도구 업체에서는 오탐을 줄이기 위한 기술개발을 지속적으로 수행해야 하며, 개발조직에서는 정탐에 대한 수집·분석 활동을 수행하여 개발자에게 코딩 가이드를 제공함으로서 정탐을 사전에 감소시키고 코드 품질을 높일 수 있는 노력이 필요하다.

셋째, 소프트웨어 신뢰성 시험은 테스팅 프로세스를 내재화하기 위한 목적을 가진 제도이다. 그러나 개발시험평가 항목에 포함되어 있어서 자발적인 품질 향상 활동이 아닌 오탐에 대한 소명 노력 과다 발생을 우려하여 시험평가 통과만을 목적으로 경고를 모두 없애기 위해 불필요한 혹은 부적절한 우회 조치를 취하여 오히려 코드 품질을 떨어뜨리는 경우가 종종 있다. 도구별로 오탐유형 분류 및 표준 오탐 사유서 제공, 소프트웨어 신뢰성 시험결과 확인 방법 합리화 등을 통해 제도의 역효과가 나타나지 않도록 도구사 및 기술지원 기관의 노력이 필요하다.

넷째, 소프트웨어 테스팅의 내재화를 위해서는 적시에 개발 소프트웨어를 테스팅 할 수 있는 환경이 필요하다. 민간 및 일부 대기업에서는 자동화된 빌드-테스트 환경(Continuous Integration)을 구축하고 수행하고 있지만, 중소기업 등 상당수의 개발조직은 고가의 테스팅 도구를 획득·유지하는 데 한계가 있다. 이들이 개발과정에서 적시에 테스팅을 수행할 수 있도록 GOTS 제공, 공동 활용 테스팅랩 운영 등 테스팅 활동 내재화를 위한 기반환경 지원 체계가 필요하다.

다섯째, 소프트웨어 신뢰성 시험 기준이 대상 체계의 특성과 다양한 개발환경을 고려하고 있지 않다. 무기체계 소프트웨어는 임베디드 시스템에 장입되는 형태, 사용자 인터페이스가 많은 콘솔 등 다양한 형태로 개발되며 사용되는 분야도 지상, 해상·수중, 항공 등 다양한 플랫폼이 존재한다. 하지만, 코딩규칙은 개발 언어별로 표준이 정해져 있으며 조정에 대한 유연성에 한계가 있다. 또한, 소스코드 메트릭의 경우 제시하는 제한값의 획일성으로 인해 객체지향언어 사용 사업 등 적용이 어려운 경우가 발생한다. 소프트웨어 유형 및 무기체계 분야별로 코딩규칙 적용에 대한 가이드가 필요하며, 소스코드 메트릭 값에 대한 수집 및 분석을 통하여 소프트웨어 특성에 맞는 적합한 메트릭 및 제한값 제시가 필요하다.

여섯째, 소프트웨어 개발 시 생산성을 높이기 위해 사용하는 공개소프트웨어, 모델기반개발 방법론 적용을 통한 자동생성코드 등에 대해서도 신뢰성이 확인되지 않은 경우 신뢰성 시험 수행을 수행해야 한다. 이로 인해 공개소프트웨어 사용이 불가능하거나, 모델기반개발 방법론 적용이 제한되는 현상이 발생한다. 모델기반 개발 방법론 적용 시 모델 검증을 통한 소프트웨어 신뢰성 시험 수행 등 개발 패러다임 변화에 대응하는 소프트웨어 신뢰성 시험 수행방안에 대한 지속적인 연구 및 제도 발전이 필요하다.

Ⅴ. 결 론

지금까지 무기체계 소프트웨어 신뢰성 시험 현황에 대하여 살펴보고 신뢰성 시험 제도 시행상의 주요 문제점 및 발전방향에 대해 살펴보았다. 2011년 제도 시행 이후 소프트웨어 테스팅에 대한 필요성이 이제는 다양한 이해관계자에게 어느 정도 공감되었다고 판단되며 무기체계 소프트웨어 개발 수행중에 테스팅을 내재화하려는 노력을 수행하고 있다.

소프트웨어 신뢰성 시험이 보다 실질적인 소프트웨어 품질 향상에 기여하는 제도가 될 수 있도록 국내 무기체계 소프트웨어 개발 환경을 고려한 프로세스 개선, 테스팅 관련 기술 연구, 수행할 수 있는 환경 구축 등 고도화/전문화를 위한 지속적인 노력이 필요하다.

소프트웨어 신뢰성 시험을 통하여 무기체계 소프트웨어의 품질향상에 많은 기여를 할 수 있도록 개발조직, 정책입안자, 기술지원 기관, 테스팅 도구 회사 등 다양한 이해관계자의 협업과 노력을 기대한다.

참 고 문 헌

[1] 방위사업청, “방위사업청 매뉴얼 제2018-7호 무기체계 소프트웨어 개발 및 관리 매뉴얼”, 2018.

[2] 방위사업청, “무기체계 내장형 소프트웨어 획득 및 관리 실무 지침서(폐지되었음.)”, 2011.

[3] 방위사업청, “방위사업청 매뉴얼 제2016-4호 무기체계 소프트웨어 개발 및 관리 매뉴얼(개정되었음)”, 2016.

[4] MISRA, “MISRA-C:2012”, 2012.

[5] MISRA, “MISRA-C++:2008”, 2008.

[6] Lockheed Martin Corporation, “Joint Strike Fighter Air Vehicle C++ Coding Standards For the System Development and Demonstration Program”, 2005.

[7] Boehm, Barry W. Software engineering economics. Vol. 197. Englewood Cliffs (NJ): Prentice-hall, 1981.

[8] IEEE Computer Society, “IEEE Std 982.1-2005 IEEE Standard Dictionary of Measures of the Software Aspects of Dependability”, 2005.

[9] ISO/IEC, “ISO/IEC 25023 Systems and software engineering – Systems and software Quality Requirements and Evaluation(SQuaRE) -- Measurement of system and software product quality”, 2016

[10] Steffen Herbold et al., “Calculation and optimization of thresholds for sets of software metrics”, Empirical Software Engineering, 2011.

[11] Tiago L. Alves et al., “Deriving Metric Thresholds from Benchmark Data”, 26th IEEE international Conference on Software Maintenance, 2010.

[12] RTCA, “DO-178C Software Considerations in Airborne Systems and Equipment Certification”, 2011.

[13] ISO, “ISO 26262 Road vehicles – Functional safty”, 2011.

〈저자소개〉

이 태 호 (Taeho Lee)
1995년 2월 : 한국과학기술원 화학과/경영정책학과 졸업
1997년 2월 : 한국과학기술원 테크노경영대학원 경영공학과 석사
2013년 8월 : 한국과학기술원 정보통신공학과(SW공학전공) 박사
1997년 ~ 현재 : 국방과학연구소 SW신뢰성기술실 실장/책임연구원
관심분야 : SW공학, SW품질, SW 프로덕트라인

백 옥 현 (Ockhyun Paek)
2000년 2월 : 충북대학교 정치외교학과 졸업
2002년 2월 : 충북대학교 전자계산학과 졸업
2002년 ~ 현재 : 국방과학연구소 SW신뢰성기술실 선임연구원
관심분야 : SW 공학, SW 제품라인 공학, SW 품질

김 태 현 (Taehyoun Kim)
2012년 2월 : 아주대학교 정보 및 컴퓨터공학부 졸업
2014년 2월 : 한국과학기술원 전산학과 졸업
2014년 ~ 현재 : 국방과학연구소 SW신뢰성기술실 선임연구원
관심분야 : SW공학, SW품질, SW 프로덕트라인