Journal of The Korea Society of Computer and Information
Vol. 23 No. 12, pp. 115-122, December 2018
www.ksci.re.kr
https://doi.org/10.9708/jksci.2018.23.12.115

Process Improvement for Quality Increase of Weapon System Software Based on ISO/IEC/IEEE 29119 Test Method

Byung Hoon Park*, Yeong Geon Seo**

Abstract

As the proportion of software in weapon systems increases, the impact of software on the overall system is growing. As a result, software quality management becomes important, and related regulations and work manuals for quality assurance activities at each stage in the R & D process are becoming more sophisticated. However, due to the characteristics of the weapon system software that is developed as a customized form for the purpose of a specific mission, there are limits to specifying and definitizing the detailed requirements (upper and lower level) according to various operational concepts during the development process. Due to this, software modification (patch change, shape change due to upgrade, etc.) occurs on account of many defects and performance improvement in the mass production and operation stage after the development is completed. In this study, we analyze the characteristics of these weapon system softwares and propose quality improvement methods based on ISO / IEC / IEEE 29119 test method.

▸Keyword: Software quality, Software test, Software engineering, Weapon system, ISO/IEC/IEEE 29119

I. Introduction

무기체계는 유도무기, 항공기, 함정 등 전장에서 전투력을 발휘하기 위한 무기와 이를 운영하는데 필요한 장비, 부품, 소프트웨어 등 제반요소를 통합한 것으로서[1] 지휘·통제통신무기체계, 감시정찰무기체계, 기동무기체계, 함정무기체계, 항공무기체계, 화력무기체계, 모의분석훈련 소프트웨어 및 장비 등 그 밖의 무기체계로 분류된다[2].

산업 전 분야에서 4차 산업혁명의 기반 기술로써 인공지능, 빅데이터, IoT 등이 확산됨에 따라, 국방 분야에서도 이에 대응하기 위해 “4차 산업혁명시대에 걸맞은 방위산업 육성”을 국정과제로 채택하고[3] 세부 계획을 추진 중이며, 그 동안 단독(stand-alone)으로 운용되던 무기체계들이 상호 연동되어 운용됨에 따라 시스템 내 소프트웨어의 비중이 증가하며, 시스템 전체 성능을 좌우하거나 시스템에 미치는 영향성이 커지고 있다. 이에 따라 무기체계 소프트웨어 연구개발 시 품질보증 활동에 대한 관련 규정, 매뉴얼 등이 제정 및 개정되며, 세부 업무 절차나 활동들이 구체화되어 품질이 향상되고 있다. 그러나 여전히 무기체계 소프트웨어 개발 종료 후에 다양한 하자(error), 사용자 불만, 성능개선 등의 이유로 양산(제품생산) 단계 및 운용 단계에서 많은 수정(패치, 업그레이드 등을 통한 소프트웨어 형상변경)이 발생하고 있다. 앞서 무기체계 용어 정의에서도 보았듯이 소프트웨어 운용 및 탑재 플랫폼이 매우 다양하며, 복잡한 무기체계 내 포함된 소프트웨어이다 보니 그 원인도 단순하지 않다. 이러한 상황에서, 무기체계 소프트웨어의 품질 향상을 위해 그 동안 방위사업청 및 산·학·연 관련 기관들이 여러 가지 측면에서 연구를 수행하고 있으며, 관련 각종 컨퍼런스, 세미나 등도 정기적으로 수행되고 있다. 그 결과 소프트웨어 개발 프로세스의 성숙도가 많이 향상되어 개발기관들이 CMMI(Capability Maturity Model Integration), SP(Software Process) 등의 인증을 획득하였으며 개발 역량은 향상되고 있는 상황이다. 각 개발 단계의 산출물이

• First Author: Byung Hoon Park, Corresponding Author: Yeong Geon Seo
*Byung Hoon Park (bhpark@dtag.re.kr), Defence Agency for Technology and Quality
**Yeong Geon Seo (young@gnu.ac.kr), Dept. of Computer Science, Graduate School of CCBM and Engineering Research Institute in Gyeongsang Nat'l University
• Received: 2018. 11. 12, Revised: 2018. 12. 12, Accepted: 2018. 12. 13.

고도화되고, 요구사항, 설계, 시험항목 간 관리도 체계화되어가고 있다. 하지만, 이러한 개발 프로세스 성숙도의 향상이 직접적으로 실제 개발 완료된 무기체계 소프트웨어의 사용품질(quality-in-use)[4] 향상으로 이어지지는 못하고 있는 실정이다. 따라서 현재 상태에서 무기체계 소프트웨어 품질을 한 단계 더 향상시키기 위해서는 특화된 방안이 필요하며, 본 논문에서는 이에 대한 연구 내용을 기술한다.

II. Related Works

본 장에서는 사용자(군) 측면의 실질적인 품질향상을 위해 일반적인 소프트웨어가 아닌 무기체계 소프트웨어만의 특성을 분석하고, 현재 수행 중인 무기체계 소프트웨어 개발 시 수행 중인 각 단계별 품질보증 활동 분석을 통해 3장에서 제시할 품질 향상 방안에 대한 필요성 및 타당성을 설명한다.

1. Weapon system software quality assurance regulations

앞서 기술된 것과 같이 무기체계에서도 소프트웨어를 이용한 기능 구현이 점점 증가하고 있으며, 전투기의 경우 소프트웨어로 구현되는 기능이 1960년대 생산된 F-4가 8%였지만 2007년에 생산된 F-35의 경우 90%가 소프트웨어에 의해 기능이 구현되었다[5]. 실제적으로도 그림 1과 같이 국방표준종합정보시스템(KDSIS) [방위사업청에서 운영하는 정보체계(Information System Software)로 무기체계 규격/표준 정보 등을 제공. 일반인(인터넷)에게는 일부 공개가능한 정보만 제공됨] 상에서도 개발 완료되어 국방 규격화된 무기체계 소프트웨어 기술자료 건수가 급격하게 증가함을 볼 수 있다(보안 상 실제 값 대신, 2008년 값을 100으로 기준하여 상대적인 값으로 변환한 값이다). 2017년에는 2008년 대비 14배 이상 증가하였음을 볼 수 있다.

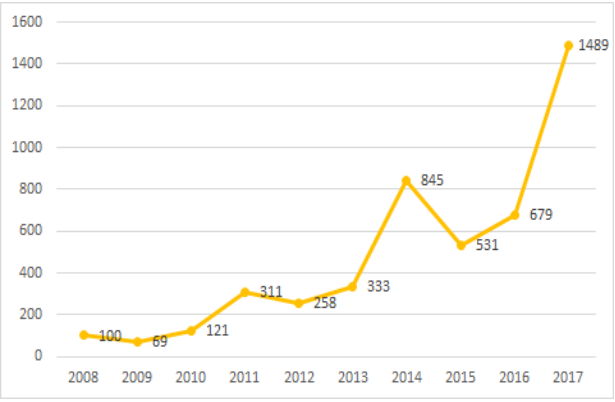

Fig. 1. Number of Defence Standardization Software of Weapon System(2008-2017)

이러한 흐름에 맞춰, 무기체계 소프트웨어 품질을 관리하고 향상시키기 위에 관련 규정 및 업무 매뉴얼이 고도화되고 있다. 이 작업은 담당 정부부처인 방위사업청과 출연기관인 국방기술품질원, 국방과학연구소에서 정기적으로 수행하고 있으며, 현재 유효한 관련 규정 및 업무 매뉴얼은 그림 2와 같다.

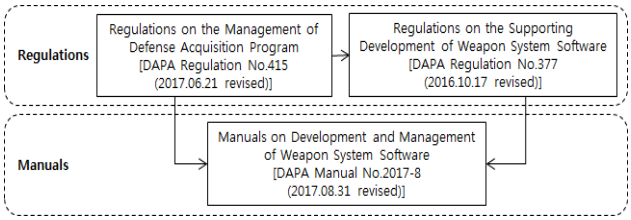

Fig. 2. Regulation & Manual of Software Quality Management of Weapon System

'방위사업관리규정'에서는 무기체계 소프트웨어 관리 항목이 있으며, 세부 규정을 참조하도록 되어 있다. 소프트웨어 관련 전용 규정인 '무기체계 소프트웨어 개발 지원에 관한 규정'에서는 각종 용어 정의와 정책, 제도와 관련된 사항이 있으며, 세부 활동은 매뉴얼을 따르도록 하고 있다. 실질적인 내용은 '무기체계 소프트웨어 개발 및 관리 매뉴얼' 상에서 제시되고 있는데, 주요 내용은 무기체계 소프트웨어 개발 프로세스 정의, 각 단계별 업무 절차, 품질보증 활동, 산출물 등이 제시되어 있으며, 코딩규칙, 취약점, 소스코드 메트릭, 코드 실행률(code coverage) 측정과 같은 신뢰성 시험이 제시되어 있다. 이를 바탕으로 무기체계 소프트웨어의 품질이 어느 정도는 향상되고 있지만, 여전히 개발 종료 후 양산 시 많은 소프트웨어 하자 및 성능개선 요소가 식별되어 수정이 발생한다. 이러한 현상의 원인은 복합적이겠지만, 주요한 원인 중 하나는 무기체계 소프트웨어 자체의 산업 특성에 따른 것이며, 본 연구에서 무기체계 소프트웨어 자체 특성에 따른 소프트웨어 완성 제품에 대한 품질 향상 방안 및 시범 적용 사례에 대해 알아보고자 한다.

2. The existing weapon system software development process and test activities at each phase

현재 무기체계 소프트웨어 개발 관련 규정 및 매뉴얼에서 정의하고 있는 개발 프로세스 상 각 단계별 소프트웨어 시험 관련된 활동은 표 1과 같다. 표준 개발 프로세스는 ISO/IEC 12207, 국방 CBD(Component Based Development) 방법론 등을 기반으로 만들어졌으며, 하드웨어를 포함한 시스템 형태로 개발됨에 따라 순차적 개발방법론(워터폴[5] 또는 V-모델[6])을 따르고 있다. 또한 각 단계별 산출물은 MIL-STD-498 규격을 따라, 각 단계별 소프트웨어 기술 문서를 작성하도록 되어 있다. 주로 수행하는 시험의 목적은 계약관계 상의 요구사항 충족 여부 위주로 수행되며, 일부 잠재적인 결함 제거를 위한 시험을 '소프트웨어 신뢰성 시험'[7]이란 용어로 재정의하여

Table 1. Weapon System Software Testing Activities of Each Software Development Phase

type	SW implementation	SW integration/ testing	system integration/ testing	test&evaluation (DT/OT)	production (initial/second)
test type	function	function/non-function	function	function	function
		SW reliability		SW reliability	SW reliability
test unit	unit(function/ CSU/CSC)	CSCI	System	System	CSCI/System
test criteria	verify function/performa nce of unit	satisfying requirements	satisfying requirements	satisfying requirements	satisfying requirements(based on KDS)
		eliminate potential defects		eliminate potential defects	eliminate potential defects
coverage type	-	requirements	requirements	requirements	requirements
		statement/branch/ MCDC		statement/branch/ MCDC	statement/branch/ MCDC
test organization	R&D organization	R&D organization	R&D organization	R&D organization/test evaluation team	quality assurance agency

• Computer Software Unit : 분리되어 시험할 수 있는 최소단위
• Computer Software Component : 소프트웨어 형상항목(CSCI)의 구성요소
• Computer Software Configuration Item : 최종 사용 기능을 만족하고, 획득기관이 형상관리를 하기 위해 분리한 소프트웨어 집합체
• Korea Defense Standard : 국방표준

Table 2. Differences between Package and On-demand Software

package software (general)	on-demand software (dedicated)
- Developers identify and develop customer needs - On development phase test level determined on its own (optimization of test effort based limited resources and TTM) - Defect removal is optional after release(or when it can't) - Achieving a specific quality goal autonomously according to market economy logic	- User requests development by providing requirements - On development phase test level determined based contraction (can't various situations test by limited cost, term) - Defect can be removed after release(may have maintenance contract) - Forcing specific quality objectives such as contracts, laws, and regulations

수행하고 있다. 하지만, 전반적으로 대부분 요구사항 만족여부 위주로 수행되고 있음을 알 수 있다. 즉, 다양한 운용 시나리오나 환경을 고려한 시험은 사업적인 측면(기간, 비용, 군 협조 등)을 이유로 제한되는 경우가 많다.

3. Characteristics of weapon system software industry

무기체계 소프트웨어의 특성을 도출하기 위해 소프트웨어 분류 중 범용적인 목적의 일반 패키지 소프트웨어(상용 소프트웨어 및 상용 제품 내장 소프트웨어)와 특수한 목적을 수행하기 위해 사용자가 요구사항을 제공하여 개발하는 주문형 개발 방식의 전용 소프트웨어 분류를 사용하였다[8]. 그 둘 간의 주요한 차이점은 표 2와 같다. 무기체계 소프트웨어와 같은 주문형 소프트웨어의 특징을 살펴보면, 먼저 실제 사용자가 아닌 개발자가 요구사항의 명확화와 구체화에 한계가 있다. 개발 진행 중 이해관계자(소요군, 방위사업청, 국방과학연구소/시제업체, 산학연 전문가 등)가 참여하는 공식 설계검토회의가 몇 차례 있지만, 제한된 시간과 시각의 차이로 제한된다. 또한, 매우 다양한 플랫폼, 탑재형태, 운용환경 등으로, 직접적으로 유사 프로젝트 경험을 보유한 개발자가 없는 상황이 많은 것도 완벽한 요구사항 명세화가 어려운 이유 중 하나이다.

다음으로, 주문형 소프트웨어는 개발 중 품질 관리가 계약관계 상 요구사항(품질 요구사항 포함) 충족으로 제한된다. 계약 단계에서 구체적인 품질 수준을 예측하기 힘들며 그에 대한 비용 산정도 어렵다. 개발 중에는 전체 사업 비용 내에서 개발기관이 개발 및 품질보증(시험 포함)을 모두 수행하므로, 객관적인 리스크 분석 및 리소스 투입이 곤란하다. 반면, 장점은 제품 배포 이후에도 제품 유지보수(패치, 업그레이드 등)가 용이하며, 별도의 유지보수 계약관계로 수정 절차, 업무 범위가 명확하게 정립되어있는 경우가 많다. 이러한 특성을 바탕으로, 무기체계 소프트웨어는 개발단계에서 기본적인 계약상의 요구사항 충족여부를 확인하고, 개발완료 후 완성 제품에 대한 시험이 좀 더 효율적이라고 분석할 수 있다.

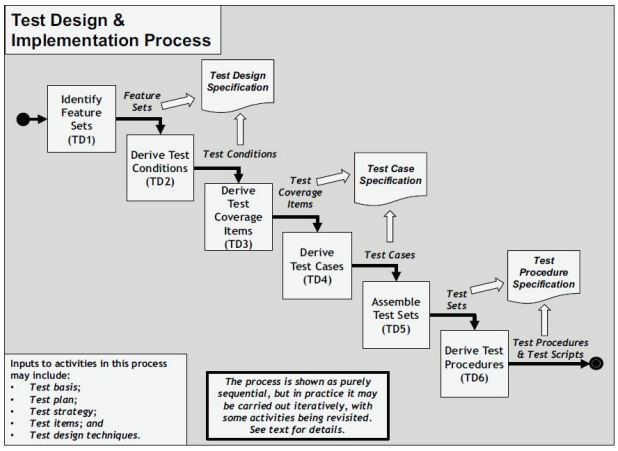

Fig. 3. Test Design and Implementation Process on ISO/IEC/IEEE 29119-2

4. Software test of ISO/IEC/IEEE 29119

ISO/IEC/IEEE 29119 시리즈는 소프트웨어 테스팅 관련 표준으로써, 이후 본 연구에서 제시하는 개선 방안에서 일부 프로세스 및 시험 방법을 활용하므로 간단하게 소개하고자 한다. ISO/IEC/IEEE 29119-2에서 그림 3과 같이 시험 설계 및 구현 프로세스를 제시하고 있다[9]. 또한, ISO/IEC/IEEE 29119-4에서 시험 설계 시 사용할 수 있는 시험 방법을 그림 4와 같이 분류하고 있다[10].

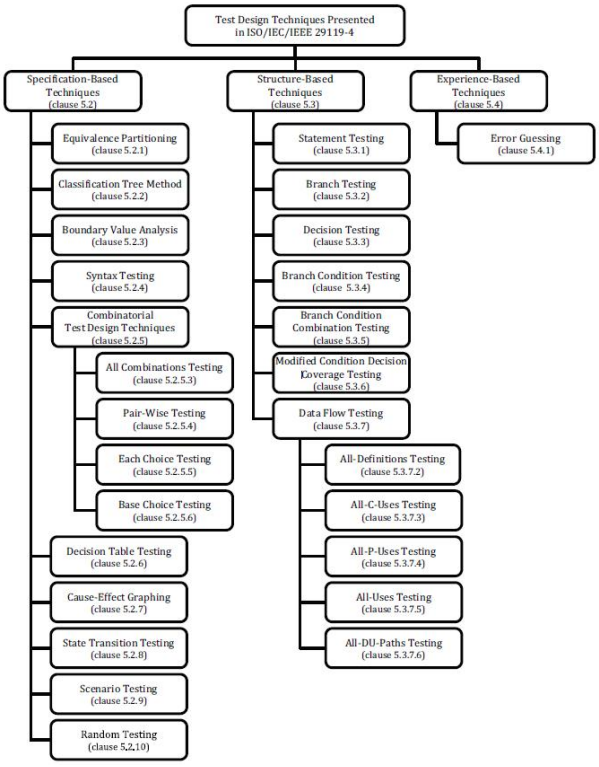

Fig. 4. Test Method on ISO/IEC/IEEE 29119-4

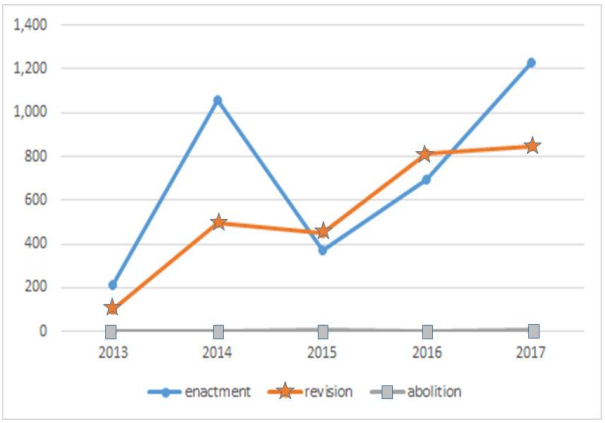

Fig. 5. Number of enactment, revision and abolition of weapon system software in 2013-2017

5. Current state of software configuration change on mass production phase

앞서 언급한 것처럼 개발단계 프로세스 고도화 및 각 단계별 테스트 활동이 강화되고는 있지만, 개발 종료 후 양산 및 운용 단계에서 소프트웨어 형상변경(수정)이 일어나는 빈도가 기존 대비 크게 나아지고 있지 않은 상황이다. 그림 5는 국방표준종합정보시스템 상에서 개발 완료된 무기체계 소프트웨어의 신규 제정, 개정, 폐기 발생 건수를 보여주고 있으며, 개정 건수가 꾸준히 증가하고 있는 것을 볼 수 있다(보안 상 실제 값 대신, 2013년 개정 건수를 100으로 기준하여 상대적인 값으로 변환한 값이다). 이를 신규 제정 대비 개정 건수의 비율로 계산해보아도, 의미 있게 향상된다고 보기 어려운 수치가 나온다.

III. Improvement of Software Product Quality based on ISO/IEC/IEEE 29119 Test Method

본 장에서는 지금까지 살펴본 무기체계 품질보증 활동 현황, 관련 표준, 특성을 바탕으로 이를 개선하기 위한 접근 방안을 정리하고, 구체적인 프로세스 개선 방안을 제시한다.

1. Approaches

현 무기체계 소프트웨어 품질관리를 위한 시험 수행 현황 문제점을 정리하면 다음과 같다. 먼저, 정상적인 운용 흐름 위주의 테스트 시나리오를 기반으로 시험을 수행한다(실제 운용 시 발생할 수 있는 다양한 상황을 고려한 테스트 시나리오 작성 미흡). 개발 중에 복잡하고 다양한 무기체계 시스템의 예외적인 시나리오 및 대안 운용 흐름을 모두 정의하여 요구사항으로 명세화하기 어려우며[11], 시스템의 기술적 난이도로 인해 정해진 개발기간 동안 요구하는 기능 구현 및 충족에 집중하게 된다. 다음으로, 모든 소프트웨어에 일괄적인 테스트 방법을 적용하고 있다. 어떤 소프트웨어는 구조적 커버리지(statement, branch, MD/DC)를 측정하는 시험 방법이 효과적이고, 어떤 소프트웨어는 명세 기반의 커버리지를 측정하는 시험, 또는 탐색적 테스팅 기법을 적용하는 것이 효과적일 수 있다. 세 번째로 시험 관련 데이터의 문서화 및 관리가 미흡하다. 개발 프로세스 고도화 및 관리로 각종 개발 데이터들은 문서화가 잘 수행되고 있으나, 테스트 케이스, 시나리오, 테스트 결과, 테스트 활동에 대한 효율(efficient), 효과(effective) 등이 명세화되어 관리되지 않는다. 이러한 문제점들을 해결하기 위해서, 현재 개발 프로세스 상 개발진행 중에는 프로세스 품질과 요구사항 충족 여부 위주로 수행하고, 복잡한 주문형 제작 형태의 무기체계 소프트웨어 특성을 고려하여, 개발 종료 후(초도 양산 또는 그 이전단계) 완성 제품에 대해 다양한 시험 방법 및 테스트케이스를 통한 시험을 추가하여 실질적인 품질 향상을 하고자 한다.

2. Improvement process

제안하는 개선 프로세스는 그림 6과 같으며, 이를 적용하는 단계는 그림 7과 같이 개발 완료되어 규격화되어 있는 소프트웨어를

Process	Identify target System →	Drive Test Conditions →	Drive Test Cases →	Test Execution →	Post processing
Input	- Quality management data on pre-phase (Development)	- Software development documents - Software list & risk	- Test basis - Test Oracle (test criteria)	- Test cases - Test environments	- All test documents & data
Output	- System under test	- Software under test (CSCI, function, interface, performance, etc.)	- Test cases (input, expected value, pre/post conditions)	- Coverage - Test effectiveness, efficiency	- Test data DB(defects, improvements item) - Technical change/ performance improvements

Fig. 6. Test process on weapon system software product

대상으로 양산 초기(초도 양산) 시점에 추가적인 품질 보증 활동인 명세 기반 및 경험 기반 테스트를 조합하여 수행하는 것이다. 이 때, 실질적인 테스트 설계 및 구현 프로세스는 앞에서 언급한 ISO/IEC/IEEE 29119-2의 프로세스를 활용하며, 소프트웨어 특성에 적합한 테스트 방법은 ISO/IEC/IEEE 29119-4를 활용한다.

세부적인 적용 프로세스는 다음과 같다. 단계별로 살펴보면, 먼저 첫 번째 대상 시스템 선정 단계에서는 개발 시 수행한 품질관리 정보(제품 리스크 분석 등급, 품질 수준 등)를 기반으로 양산단계 시 추가적인 품질보증이 필요한 대상 체계를 선정한다. 일반적으로 리소스(인력, 비용)는 한정적이므로, 모든 무기체계를 대상으로 추가 품질보증을 수행할 수는 힘들 것이며, 리스크 기반으로 대상 시스템을 선정한다. 두 번째 단계는 전 단계에서 정해진 대상 시스템 내에서 테스트 컨디션, 즉 검증을 하고자 하는 컴포넌트나 시스템 항목, 이벤트 등을 선택하는 것이다[2]. 선정된 시스템 내에서 다시 한 번 구체적인 시험대상 소프트웨어, 국방 분야에서는 CSCI라는 단위로 구분된다. 시스템 내에는 주 장비에 탑재되어 핵심적인 기능을 수행하는 CSCI도 있고, 지원 장비에 탑재되는 CSCI도 있으므로, 실제 대상 CSCI를 선정하고, 그 중에서 세부적으로 시험할 항목을 선정한다. 세부적인 대상은 기능, 인터페이스, 모듈 등이 될 것이다. 세 번째 단계에서는 선정된 대상에 대한 테스트 케이스를 도출하는 것이다. 이 때 테스트 베이시스로 기 규격화된 소프트웨어 기술문서(SRS, SDD, STD, SPS 등)를 활용하며, 또한 유사 시스템 개발자 및 사용자의 경험을 활용한다. 네 번째 단계는 도출한 테스트 케이스를 갖고 실제 시험을 하는 단계이다. 시험 진행 중에 시험 내용을 문서화하고, 커버리지를 측정하며 테스트 효과 및 효율을 측정한다. 마지막 단계는 후처리로 앞 4단계의 각종 입출력 자료들을 문서화하고 정리하여 관리되도록 하는 것이다. 필요 시 개발기관 및 유관기관과 협의하여 식별한 사항을 실제 시스템에 반영토록 추진할 수도 있다. 그리고 도출된 성능 개선 요소들을 바탕으로 유사 시스템 개발 시에 그 정보를 활용할 수 있도록 제공할 수 있을 것이다.

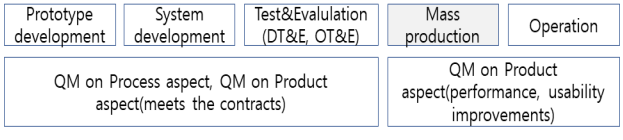

Fig. 7. Phase of Improvement Method

IV. Experiment and Evaluation

1. Experimental Method

제안한 개선 프로세스의 효과를 확인하기 위하여 실제 무기체계 소프트웨어로 실험을 수행하였다. 국방 분야 무기체계 특성상 제시한 방안이 관련 절차를 통해 실질적으로 규정, 업무 매뉴얼에 반영되기 전까지는, 실제 프로세스에 적용하기는 힘들기 때문에, 두 가지 방식으로 실험을 수행하였다. 첫 번째는 기 수행된 양산단계 소프트웨어 수정(기술변경을 통한 개정) 사례를 분석하여, 제시한 개선 프로세스를 통해 해결이 가능함을 역으로 입증하였으며, 두 번째는 특정 무기체계를 선정하여 본 프로세스 대로 진행하여 성능개선 요소를 도출하였다. 도출된 사항은 개발 시 요구사항은 충족하였지만, 운용 중 발생할 수 있는 예외처리 및 성능향상과 관련된 것으로써, 본 사항을 실제로 수정 반영할 지는 기술 외적인 사업적인(비용/일정 등) 부분이므로 추후 협의를 통해 진행할 예정이다.

Table 3. Case Analysis Result of Weapon Software Modification

Target system /equipments	Cause of defect	Test method	Analysis results
KOOO/OO equipment	undefined input	specification-based drive TCs	can be detected by analysis input range on design description & interface documents
KO OOO display unit	insufficient exception processing	specification-based drive TCs	can be detected by analysis task priority on design description documents
OOO armoured car/OO box	insufficient operational scenario	experience-based drive TCs	can be detected by test expert of imbedded system
OOO vehicle/OO control uint	insufficient operational scenario	experience-based drive TCs	can be detected by developer & operator of similar system

2. Analysis results of the existing software configuration changes

먼저 기존에 수행되었던 소프트웨어 형상변경 내용을 바탕으로 본 연구에서 제시하는 프로세스를 적용하였다면, 사전 식별이 가능하였을 것으로 판단할 수 있는 사례들을 분석하여 제시한다(표 3). 첫 번째는 KOOO OO통제장치 OOOO측정기의 자체점검 결과 전시 불량 개선 건으로 인터페이스 설계 문서상 통신상태의 값이 완벽하지 않아서 발생한 내용으로, 미정의된 값이 수신되었을 때 원치 않는 수행을 하였다(그림 8).

두 번째는 KO OOO종합전시기의 화면 멈춤 현상 개선 건으로, 기동 간 간헐적으로 전시기의 화면이 정지되고 화면 구성이 흐트러지는 현상이 발생하였다(그림 9).

세 번째는 OOO장갑차 OO회로망상자 비상등 스위치의 간헐적 미동작 현상 개선 건으로 스위치 상태 변경을 인지하는 소프트웨어 타이머에 대한 내용이다.

네 번째는 OOO차량 OO제어기로 요구된 기능이긴 했지만, 운용 상황에 따라 불필요한 동작이 수행되어 운용 상 불편한 사항이 있었다.

위 4건의 종합적인 결과는 표 4와 같으며, 모두 앞 장에서 제시한 프로세스를 통해 문제점들이 모두 시험되었을 것으로 분석되었다.

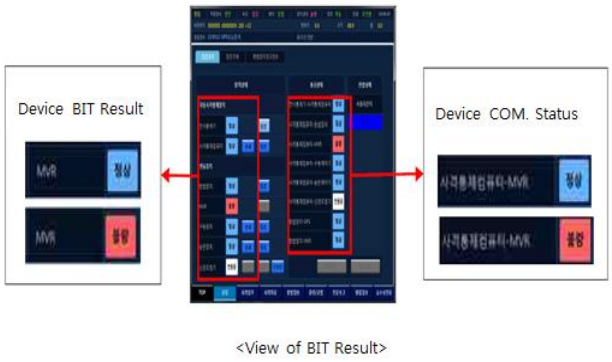

Fig. 8. Modification Case 1 of CSCI - BIT(Built-in-Test) Result Error

Fig. 9. Modification Case 2 of CSCI - Display unit Freezing

Table 4. Deriving Test Condition

No.	Identifier	Requirements
		Description
1	S-DFSA-009	- should analyze PRI, frequency modulation type - PRI modulation type : jitter, Dwell&Switch(lv.8), Stagger(lv.8), pattern(sign, triangle, etc.) - frequency modulation type agile, hopping(lv.8), pattern(sign, triangle, etc.)
2	S-DFSA-017	- should find direction of RF radiator - calculate the angle(AOA) by rotating the spinning-antenna

• Pulse Repetition Interva : 펄스 반복 간격. 레이더에서 펄스형태의 신호 방사 시 펄스 간 간격 시간
• Angle Of Arrival : 도래 방위. 레이더 펄스를 수신한 방위각(azimuth)

Table 5. Test Case Design

No.	type	test cases(TC)		remarks
		input	expected value	
1	analysis of PRI, frequency modulation type	PRI modulation type : lv.8 stagger	- PRI modulation type : 0x020000000 PRI each lv.(us)	input in the range
2		PRI modulation type : lv.9 stagger		out-of-range input
3		PRI modulation type : lv.16 stagger		
4		frequency modulation type : lv.8 hopping	- frequency modulation type : 0x020000000 frequency each lv.(Mhz)	input in the range
5		frequency modulation type : lv.9 hopping		out-of-range input
6	find direction of RF radiator	without noise	AOA : 200(20 deg.)	no specific requirements
7		added noise		
8		count of pulses : 1800	2200(220 deg.)	
9		count of pulses : 72		

3. Results applied to actual weapon system software

본 연구에서 제시한 프로세스를 실제 개발 완료되어 초도양산 전인 무기체계에 적용하여 그 효과를 확인하였다. 먼저, 첫 번째 단계로 시험대상 시스템 선정은 기본적으로 해당 무기체계에 대한 운용 및 개발 경험자를 섭외할 수 있고, 동적시험 환경을 구성에 필요한 장비를 확보할 수 있는 체계로 선정하여 훈련함 용 OOO훈련지원체계(OOO-OOO)로 선정하였다.

두 번째로 테스트 컨디션은 선정한 체계 내에서 핵심적인 기능을 수행하는 방탐 및 신호분석 CSCI로 도출하였다. 소프트웨어 목록 명세서를 통해 체계 내 전체 소프트웨어 목록을 확인하고, 소프트웨어 시험 계획서에서 각 소프트웨어 형상항목 별 위험도를 참조하여 선정하였다. 해당 CSCI 내에서는 소프트웨어 요구사항 명세서를 통해 가장 주요한 기능 요구사항을 2개 식별하여 시험항목으로 선정하였다. 세 번째 단계로 각 항목별 세부 테스트 케이스를 도출하였다(표 5). 요구사항 명세서의 요구사항 설명, 유즈 케이스 시나리오와 설계기술서의 클래스와 시퀀스 다이어그램을 통해 결함을 일으키거나, 성능개선을 할 수 있는 테스트 케이스를 4건 도출하였다(그림 10).

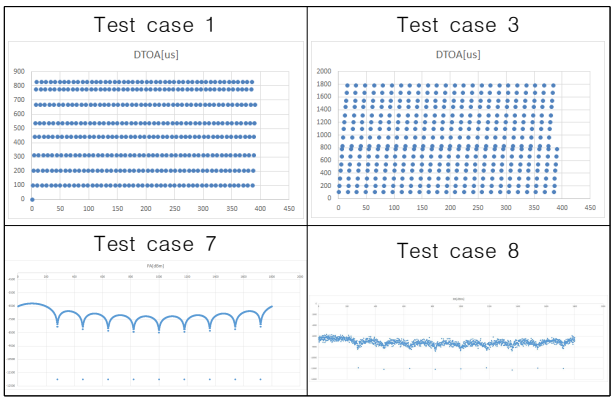

Fig. 10. Test Cases(normal-exception)

네 번째 단계로 위 테스트 케이스로 테스트 시나리오를 작성하여 실제 시험을 수행하였다. 시험환경은 PC용 VxWorks 시뮬레이터 상에서 소프트웨어를 실행하였으며, 입력으로 레이더펄스 데이터(PDW)는 엑셀 및 매트랩을 사용하여 파일형태로 만들어 입력하였다.

-운영체제 : Microsoft Windows 7 32bit
-통합개발환경(IDE) : Wind River Workbench 3.3 (VxWorks 6.9)
-입력데이터 생성 : Microsoft Excel, Matlab

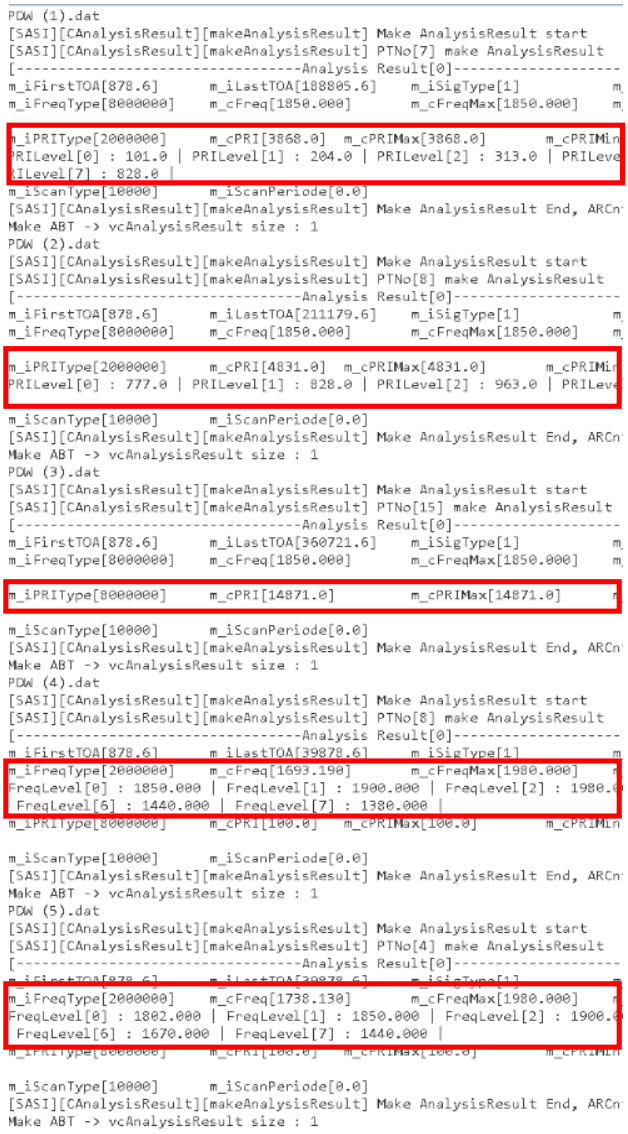

Fig. 11 Test Result-PRI, Frequency Modulation Type

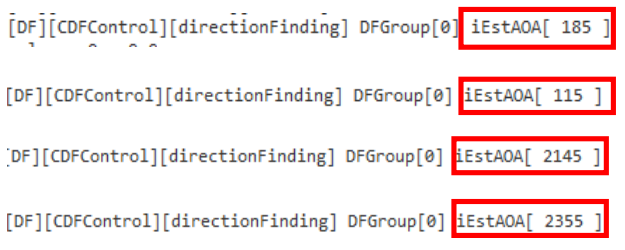

Fig. 12. Test Result - Direction Detection

시험 결과는 그림 11, 그림 12와 같다. 시험 결과 TC1은 8단 스태거 및 각 PRI 단 값이 정상적으로 분석되었고, TC2와 TC3은 각각 신호형태 및 PRI 값이 잘못 분석되었다.

다음 항목의 시험 결과도 TC6과 TC8은 오차 범위 내에서 방위값이 측정되었으며, TC7과 TC9는 오차 범위 이상으로 측정된 것을 볼 수 있었다. 시험 효과는 성능개선 요소 2건 도출, 효율성은 특정 시간 내 결함 및 성능 개선 항목 도출 건수를 하여 0.3건/시간 이다. 하지만, 이는 테스트 참가자의 숙련도에 따라 달라질 수 있다.

결론적으로, 시험 결과 정상적인 요구사항에 명시된 테스트 케이스는 모두 충족하였지만, 정의되지 않은 입력에 대해서는 올바른 결과(실제로는, 비정상적인 입력에 대한 동작 자체가 정의되지 않음)가 나오지 않음을 확인하였다. 이러한 항목들은 운용 개념에 따라 개선이 필요할 수도 있는 항목이며, 사용자 입장에서는 품질이 떨어진다고 생각할 수 있는 항목들이며 본 연구에서 향상시키고자 하는 사용자 품질로 볼 수 있다.

IV. Conclusions

미래 전장 환경이 첨단 복합화로 진화됨에 따라 무기체계에서 소프트웨어를 이용한 기능구현이 증가하고 있으며, 무기체계 소프트웨어 개발과 검증의 중요성 역시 증가하고 있다. 개발 진행 중 각 단계별 산출물 관리와 품질보증 활동을 강화하는 프로세스 측면의 품질 향상은 어느 정도 성숙단계에 들었다고 볼 수 있다. 하지만, 소프트웨어는 특성상 개발 완료(요구사항 충족)가 100% 완벽한 제품이라는 것은 아니고, 실제 배포/ 및 사용 후에 구체화되거나 개선사항이 많이 도출된다. 상용 소프트웨어나 게임 소프트웨어 등도 최근에는 베타 테스트나 클라우드 테스트를 많이 하는 추세이다. 특히, 개발 진행 중에 사용자가 설계 수준의 요구사항을 제시하기 어렵고, 개발자 또한 상세한 운용 시나리오를 예측하기 어려운 무기체계 소프트웨어의 경우는 더 그러하다.

본 연구에서는 위와 같은 특성에 따라 무기체계 소프트웨어의 특성에 따른 개발 종료 후 완성제품에 대한 테스트 추가로 현 프로세스 개선안을 제시하였고, 기 데이터 분석 및 자체 시험적용을 통해 그 효과를 확인하였다. 앞으로 더 많은 데이터 분석과 개선 프로세스의 세분화를 진행하고, 이를 바탕으로 관련 규정, 업무 매뉴얼에 반영되도록 제안할 것이다. 또한, 이 개선안을 효과적으로 적용하기 위해서는 몇 가지 해결해야 할 문제들이 있다. 첫 번째는 해당 무기체계 분야 및 소프트웨어 테스트 전문가 확보가 필요하며, 두 번째는 시험 시설 및 장비 확보가 필요하다. 무기체계 소프트웨어의 경우 대부분 내장형 시스템이며, 운용 플랫폼이 항공/함정/지상/모바일 등으로 다양하며, 탑재 플랫폼 또한 FPGA/DSP/회로카드 조립체형 컴퓨터/일반 PC 등으로 다양하므로, 시험환경 구축이 필요하다. 세 번

째로 성능개선 요소를 도출했을 때, 그것을 반영하기 위한 후속 조치 절차를 마련해야 한다. 이러한 문제점들을 해결해나가며, 적용하여 실질적인 사용자(군) 측면의 품질향상을 이루도록 노력해야 할 것이다.

REFERENCES

[1] G. M. Choi, Y. H. Kim and G. I. Woo, "Software Reliability Assurance for Weapon System Quality Improvement," Defense & Technology, Vol. 404, pp. 74-85, 2012.

[2] Ministry of Defense, Defense Business Act, Article 3 (Justice), Law 15344 ('18 .1.16. Amended), 2018.

[3] Ministry of Defense, Enforcement Decree of the Defense Business Act, Article 2 (Classification of Weapon System), Presidential Decree No. 28799 (Revised Apr. 18, 2018), 2018.

[4] S. W. Noh, “Analysis and Improvement Plan of Defense Technology Planning for the 4th Industrial Revolution”, Journal of Academia-industrial Technology, Vol. 19, No. 4, pp. 552, 2018.

[5] Roger S. Pressman, “Software Engineering: A Practitioner's Approach“, The McGraw-Hill Companies, pp. 47-48, 2017.

[6] S. Balaji, and M. Sundararajan Murugaiyan, Waterfall vs V-Model Vs Agile: A Comparative Study on SDLC, International Journal of Information Technology and Business Management 29th, Vol. 2, No. 1, pp. 28, 2012.

[7] Defense Acquisition Program Administration, “Weapon System Software Development and Management Manual”, No. 2017-8 (Amended on August 31, 2017), 2017.

[8] M. S, Yoon and S. B. Park, “An Empirical Study on the Quality Requirements Priorities of Packaged Software Using the AHP”, Internet E-commerce Research, Vol. 8, No. 2, pp. 2, 2008.

[9] ISO/IEC, Systems and Software Engineering -Systems and Software Quality Requirements and Evaluation(SQuaRE) — System and Software Quality Models, pp. 3, 2011(E).

[10] ISO/IEC/EEE, Software and systems engineering—Software Testing—Part 4 : Test techniques, pp. 6, 2015(E).

[11] Y. G. Seo and etc, “Design and Implementation of Component-based Configuration and Data Management System for Weapon System R&D”, Journal of KSCI, Vol. 13, No, 7, pp. 127-138, 2008.

Authors

Byung Hoon Park received a BS degree from computer engineering of Hongik univ. in 2006. During 2005 - 2015, he worked in LIGNex1. He has been studying at graduate school of GNU(Gyeongsang Nat'l Univ.) since 2017. Now he has been working for DTAQ, dept. of software quality assurance since 2015. His research interests include software testing, software engineering and ICT convergence.

Yeong Geon Seo received MS and Ph.D. degrees from computer science of Soongsil univ. in 1987, 1989 and 1997, respectively. During 1989 - 1992, he worked in Trigem computer inc. developing 4GL(XL/4). And now he has been working for GNU, dept. of computer science and graduate school of CCBM since 1997, and Engineering Research Institute since 2011. His research interests include medical imaging, software testing and computer network.