JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 27 No. 1, pp. 51-59, January 2022
https://doi.org/10.9708/jksci.2022.27.01.051

A study on Software Maintenance of Domestic Weapon System by using the Automatic Test Equipment

Il-Kwon Chae*

*Engineer, Naval R&D Center, Hanwha Systems, Gumi, Korea

[Abstract]

As the weapon system's dependence on software functions increased, software became a key factor in controlling the weapon system. In addition, as software development becomes more important domestically and internationally, software verification becomes an issue. The recent defense market has recognized this point and is demanding a plan for weapon system software maintenance. In this paper, we propose a weapon system software maintenance plan using Automatic Test Equipment. The specific method is to use a simulator to check the software function and identify failure cases. This is an effective way for developers to reduce the Total Corrective Maintenance Time(TCM) of the weapon system by reducing the time it takes to identify failure cases. It has been proven that the proposed Automatic Test Equipment can achieve software maintenance and excellent Maintainability and Operational Availability compared to the existing ones.

▸ **Key words**: Support Equipment, Automatic Test Equipment, Software Maintenance, Software Reliability, Simulator, Combat Management System

[요 약]

무기체계의 소프트웨어 기능 의존성이 높아짐에 따라 소프트웨어는 무기체계를 제어하는 핵심 요인으로 자리매김하였다. 또한 국내/외적으로 소프트웨어 개발이 중요해지면서 소프트웨어에 대한 검증이 쟁점이 되고 있다. 최근 방산시장에서 이러한 점을 인식하고 무기체계 소프트웨어 유지보수를 위한 방안을 요구하고 있다. 본 논문에서는 자동화시험장비(Automatic Test Equipment, ATE)를 이용하여 무기체계 소프트웨어 유지보수 방안을 제시한다. 그 구체적인 방안은 시뮬레이터를 활용하여 소프트웨어 기능을 확인하고 고장 케이스를 식별하는 것이다. 이는 개발업체가 고장 케이스를 식별하는 시간을 줄임으로써 무기체계 총 고장정비시간(Total Corrective Maintenance Time , TCM)을 감소시킬 수 있는 효과적인 방법이다. 제안된 소프트웨어 유지보수 기능 추가된 자동화시험장비(ATE)는 기존 대비 정비도 및 운용가용도(OA)가 개선됨을 입증하였다.

▸ **주제어**: 지원장비, 시험장비, 소프트웨어 유지보수, 소프트웨어 신뢰성, 시뮬레이터, 전투체계

• First Author: Il-Kwon Chae, Corresponding Author: Il-Kwon Chae
*Il-Kwon Chae (ik.chae@hanwha.com), Naval R&D Center, Hanwha Systems
• Received: 2021. 11. 25, Revised: 2021. 12. 22, Accepted: 2021. 12. 22.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

기존 국방 분야의 무기체계들은 하드웨어 기반의 시스템으로 개발됐으며, 하드웨어 성능 및 신뢰도에 중점을 두었다. 그러나 무기체계는 다양한 환경에서 운용되기를 요구받고 있으며, 복합적 기능을 갖는 무기체계 개발이 주를 이루고 있다[1]. 이런 기능을 구현하기 위해 무기체계 소프트웨어는 더욱 복잡, 정교해지고 있다. Fig. 1같이 무기체계의 소프트웨어 기능 의존성이 높아짐에 따라 소프트웨어는 무기체계를 제어하는 핵심 요인으로 자리매김하였다.

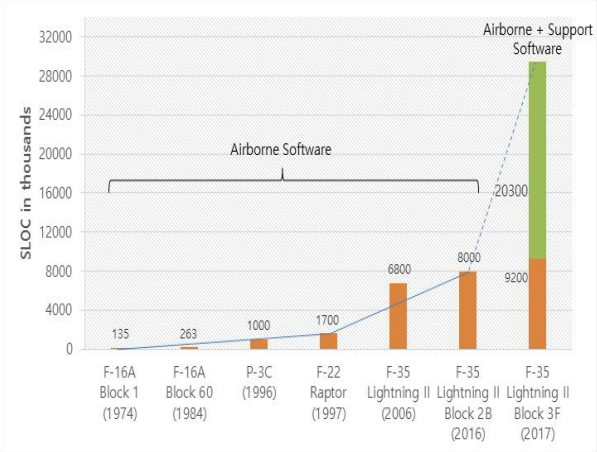

Fig. 1. Ratio of Software Functions on Weapon System[2]

소프트웨어 개발이 중요해지면서 소프트웨어에 대한 검증이 국내/외적으로 쟁점이 되고 있다. 대표적으로 소프트웨어 시간 계산 오류로 인한 미사일 방어 시스템 미동작, 항공 레이더 시스템 결함으로 인한 민간 항공기 격추 등의 사례가 있다. 이과 같이 막대한 인적, 금전적 손해를 일으키는 사고가 늘어나고 있다. 국내에서는 우주 발사체 나로호 발사 시 압력 측정 소프트웨어 결함으로 발사가 중지되는 일이 발생하였다. 국내 개발된 무기체계도 소프트웨어 오류로 인한 무기체계 고장이 점차 발생하고 있다.

해양 전투체계 체계진단 시 업체로 하드웨어 및 소프트웨어 정비 요청사항을 분석한 결과는 Fig. 2와 같으며 전체 고장의 18%가 소프트웨어에 대한 성능개선, 결함 수정 요청사항이다. 또한, 업체 정비 항목 중 연도별 소프트웨어 고장 비율은 Fig. 3과 같으며 무기체계 운용 중에 발생하는 소프트웨어 고장이 차지하는 비중이 점차 증가함을 알 수 있다.

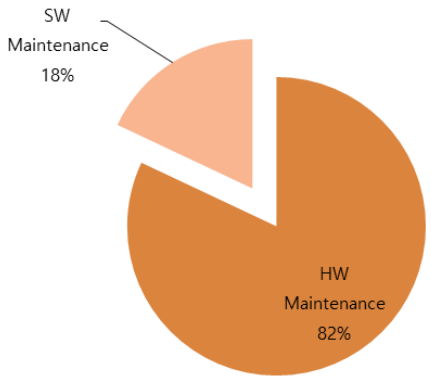

Fig. 2. Ratio of Hardware/Software Maintenance

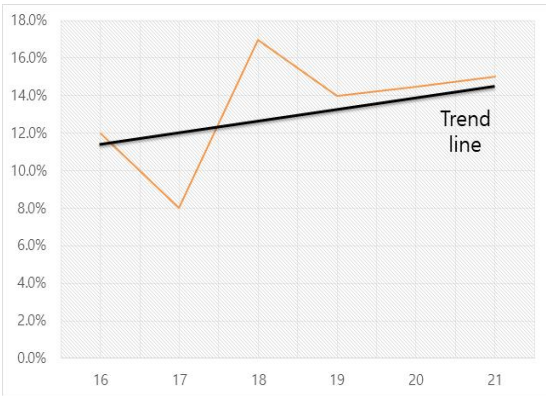

Fig. 3. Ratio of Software Failure per Year

소프트웨어 정비 요청으로 인한 조치사항을 자세히 살펴보면 Fig. 4와 같다. 소프트웨어 정비 요청사항으로 체계진단 기간 중 소프트웨어 오류 현상이 재현되지 않아 지속 관찰되는 경우가 65%를 차지하며, 소프트웨어 재설치 및 소프트웨어 업데이트 사항이 약 22%를 차지하고 있다.

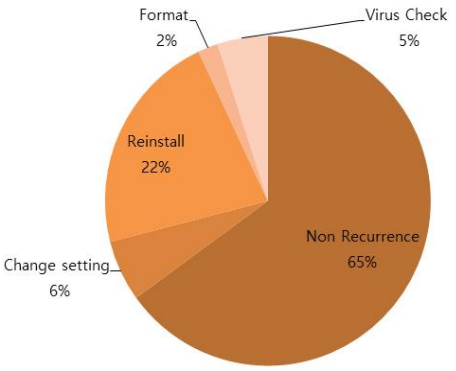

Fig. 4. Ratio of Software Failure Maintenance

이러한 흐름에 맞춰 방위사업청에서 제정한 방위사업규정[3] 및 무기체계 소프트웨어 개발 및 관리 매뉴얼[4]에는 연구개발 주관기관은 무기체계 소프트웨어 신뢰성을 확보하고, 유지보수 등 소프트웨어 군수지원을 원활하게 수행

하기 위하여 종합군수지원요소를 개발하여야 한다고 명시하고 있다. 이를 바탕으로 종합군수지원 11대 요소 중 지원장비는 소프트웨어 유지보수 도구(개발도구, 운용도구, 운영 지원 도구 등)와 정비에 필요한 소프트웨어를 확보해야 한다. 하지만 국내 무기체계 소프트웨어 유지보수를 위한 도구나 장비는 찾아볼 수 없으며, 소프트웨어 유지보수를 위한 세부적 기준과 규정이 정립되어 있지 않다.

기존 무기체계에는 소프트웨어 기능 점검이 가능한 장비가 없다. 소프트웨어 오류 발생 시, 소프트웨어 개발업체가 현상만 가지고 소프트웨어 오류 발생 케이스를 식별하는 방식은 총 고장정비시간(TCM)의 증가 및 무기체계의 정비도와 운용가용도(OA)를 낮춘다. 본 논문에서는 자동화시험장비(ATE)를 이용하여 무기체계 소프트웨어 유지보수 방안을 제시한다. 그 구체적인 방안은 시뮬레이터를 활용하여 소프트웨어 기능 점검과 소프트웨어 고장 케이스를 식별하는 것이다. 하드웨어 기능 점검 중심의 지원장비가 아닌 소프트웨어 기능 점검 및 유지보수가 가능한 지원장비를 개발에 목적이 있다. 이는 무기체계 구성품 시험에 소요되는 총 고장정비시간(TCM)을 감소로 정비도 및 운용가용도를 향상하는 효과가 있다.

본 논문의 구성은 다음과 같다. 2장에서 소프트웨어 신뢰성에 대한 이론적 배경, 사용 패턴 기반 테스트가 시스템의 평균고장시간(Mean Time to Failure, MTTF)에 미치는 효과를 서술하였다. 그리고 정비도와 운용가용도의 정의, 자동화시험장비 정의와 분류, 소프트웨어 유지보수를 위한 자동화시험장비(ATE)를 서술하였다. 3장에서는 EO/IR 점검장비 구성, 시험 구성, 사용자 패턴 기록을 위한 설계, 마지막으로 시험을 통한 소프트웨어 유지보수의 효과성을 검증하였다. 마지막 4장에서는 결론 및 추후과제로 본 논문을 마무리하였다.

II. Preliminaries

1. Overview of Software Reliability

1.1 Software Reliability Testing

소프트웨어의 결함은 많은 인명과 재산상의 피해를 가져올 수 있으므로 소프트웨어의 비중이 증가하는 만큼 소프트웨어 신뢰성 시험에 대한 인식과 중요성이 높아지고 있다.

방위사업청에서 제정한 무기체계 소프트웨어 개발 및 관리 매뉴얼에 따르면 소프트웨어 신뢰성 시험이란 소프트웨어 코드가 일으킬 수 있는 결함을 사전에 식별하여 제거하기 위한 시험을 말하며 정적시험 및 동적 시험으로 구분하고 있다. 소프트웨어 신뢰성 시험은 잠재적 결함을 최소화하기 위한 활동으로 소프트웨어 신뢰성 시험을 수행한다고 해서 소프트웨어의 잠재적 결함이 완전히 없어지는 것은 아니라고 명시하고 있다.

1.2 Software Reliability Testing Methods

소프트웨어 신뢰성 시험 기법은 목적이나 특징에 따라 구분되어 진다. 대표적으로 시스템 동작 여부에 따라 정적시험과 동적시험이 있다. 정적시험은 시스템 실행 없이 산출물 위주의 검토를 시행한다. 동적시험은 구현된 시스템을 실행시켜 테스트를 시행한다.

동적시험은 화이트박스(White-box testing) 테스트, 그레이박스 테스트(Gray-box testing), 블랙박스(Black-box testing) 테스트가 있다. 동적시험 테스트를 구분하는 기준은 정보 획득 대상으로 볼 수 있다.

화이트박스 테스트는 소프트웨어의 내부 구조(Structural) 혹은 코드-기반(Code-Based) 테스트하는 방식이다. 소프트웨어 내부 코드를 테스트하는 기법으로 개발자 관점의 단위테스트라 볼 수 있다. 또한, 구조 기반 테스트라고 볼 수 있다. 구조 기반 테스트는 제어 흐름 테스트, 데이터 흐름 테스트, 분기 테스트, 경로 테스트가 있다. 개발자가 내부 코드 동작을 추적할 수 있어서 동작의 유효성뿐만 아니라 코드를 꼼꼼하게 테스트할 수 있다. 무기체계 개발 기간에 화이트박스 테스트가 이루어진다.

블랙박스 테스트는 소프트웨어 내부 구조나 작동 원리를 모르는 상태에서 기능(Function)을 테스트하는 방식이다. 올바른 값과 올바르지 않은 값을 입력하여 올바른 출력이 나오는지 검사한다. 사용자 관점의 테스트 방법이라고 볼 수 있다. 또한, 명세 기반 테스트 방법이라 볼 수 있다. 명세 기반 테스트 기법으로는 동등 분할 기법, 경계값 분석 기법, 오류 예측 기법, 상태 전이 테스트 기법 등이 있다. 사용자의 경험을 바탕으로 기능이 누락되는 오류를 검출하기 쉽다.

그레이박스 테스트는 요구사항 명세서에 대한 이해 및 개발자와의 접촉을 통해서 체계의 내부 구조를 이해하는 테스트로서 기능적(Functional)인 블랙박스와 구조적(Structural)인 화이트박스 테스트가 융합된 테스트하는 방식이다.

1.3 Failure Curves for Software

Fig. 5는 시간의 흐름에 따른 하드웨어와 소프트웨어의 고장률을 표현한 그림이다. 하드웨어는 안정화 단계에 들어가면 일정한 고장률을 유지하기 때문에 일반적으로

Bathtub 곡선을 그린다. 소프트웨어는 개발단계에서 개발이 완료된 후 오류 수정이 완료되면 이론적으로 고장률이 매우 낮은 상태를 유지한다. 그러나 소프트웨어 수정 활동(유지보수 활동)을 수행함에 따라 추가적인 결함이 주입되어 고장률이 급격하게 상승하는 양상이 비번이 발생한다.

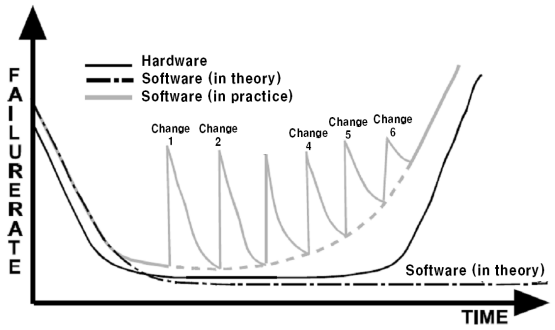

Fig. 5. Change of Failure Rate by Software and Hardware Maintenance

Table 1은 하드웨어 고장과 소프트웨어 고장에 관련된 특성을 비교한 표이다. 하드웨어 고장은 설계, 제조, 운용 중에 발생하며, 다양한 방법(육안 점검, 자체 점검 기능(Built In Test), 자동화시험장비(ATE) 등)을 통해 고장을 식별 할 수 있다. 반면에 소프트웨어 고장은 개발과정에서 발생하는 오류뿐만 아니라 고객의 요구사항 변경, 하드웨어 변경 등 다양한 요소에 의해서 발생한다. 소프트웨어 결함 발생원인, 조치 방안 및 결함에 의한 무기체계에 미치는 여파를 정확하게 감지할 수 없다.

1.4 Software Maintenance Classification

국내뿐만 아니라 미국에서도 무기체계 소프트웨어만을 관리하기 위한별도 조직은 없으며, 각 군에 소속된 소프트웨어센터 또는 무기체계 개발업체와 계약에 의해 소프트웨어 유지보수가 수행된다. 미 국방부(Department of Defense, DoD)는 무기체계 소프트웨어 유지관리를 위해 유지관리 유형을 4가지로 구분하고 있다[5].

■ 오류 수정(Corrective Sustainment)는 유지관리 대상 소프트웨어의 오류 진단 및 수정하는 활동이다.

■ 예방 정비(Preventive Sustainment)는 유지관리 대상 소프트웨어의 오류를 사전에 감지하는 활동이다.

■ 적응 정비(Adaptive Sustainment)는 유지관리 대상 소프트웨어가 새로운 환경에 적응하기 위한 활동으로 데이터 전환, 시스템 성능개선에 따른 프로그램 변경, 패키지 버전 상승에 따른 부분 개선 등이 해당한다.

■ 완전 정비(Perfective Sustainment)는 소프트웨어의 기능 향상하는 활동으로 유지관리 대상 소프트웨어에 대한 신규 기능 추가, 기능 변경, 기능 삭제 등이 해당한다.

2. The Adam's Study

아담의 연구에 따르면 운영 체제, 언어 컴파일러, 데이터베이스 시스템 등을 포함한 대규모 IBM 소프트웨어 제품에서 데이터 수집하여 분석한 결과 두 가지 결론을 내렸다.[6]

■ 조사된 다양한 소프트웨어 제품들의 장애율 분포가 균일하다.

■ 장애율의 평균고장시간(MTTF)이 5000년부터 19월 사이에 큰 차이가 나게 펴져 있다. 평균고장시간(MTTF)을 나타낸 Fig. 6에서 보이듯이 33.29%의 에러는 5000년의 평균고장시간(MTTF)을 가지고 0.51%의 에러는 19개월 평균고장시간(MTTF)을 가진다.

Table 1. Comparisons of specification between hardware and software

	Hardware	Software
Failure cause	Failure occurrence with designing, manufacturing, operating	Error on the development Customers needs Addition of function, change of hardware
Failure symptoms	Detectable	Non detectable
Prediction on failure	Predictable through analysis of RAM	Difficulties in predicting for error Only possible to grasp by addition of function, change of hardware
Maintenance task	Restoration to the original state with exchange of component	System operations and maintenance support task Software modification tasks
Failure recurrence	Replicable same failure	Modified error do not appear again High probability of occurrence with improper maintenance

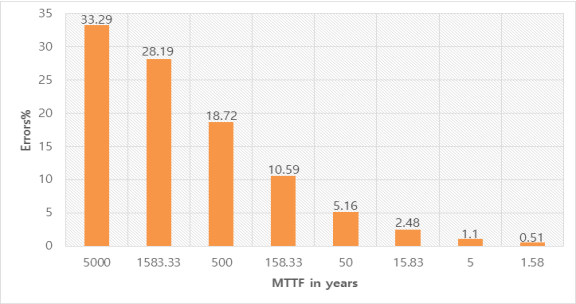

Fig. 6. Distributions of Errors in Percent among Mean Time to Failure(MTTF)

높은 장애율을 가지는 2%~5%의 에러만 제거하여도 소프트웨어 제품이 자신의 생명주기 동안 무결점에 가까운 상태로 가동될 수 있다[7]. 평균고장시간(MTTF)이 5000년이나 되는 매우 낮은 장애율을 가진 33.29%의 에러를 발견하기 위해 커버리지 테스트를 하는 것 보다 주로 사용되는 부분에 집중한 사용 패턴 기반 테스트를 통해 높은 장애율의 에러를 발견하고 수정하는 편이 시스템의 평균고장시간(MTTF)에 긍정적인 영향을 미칠 수 있다.

3. Classified Time of Combat Management System

가용도란 정비 가능한 시스템이 어떤 사용조건에서 규정시간에 정상적인 기능을 유지하고 있는 확률을 말하고 고유가용도, 성취가용도, 운용가용도로 분류할 수 있다[8]. 상세한 시간 분류 기준은 Table 2에 보인다.

운용가용도(Operational Availability, OA)란 무기체계가 실제의 운용환경과 규정된 조건에서 사용될 때 임의의 시점에서 만족스럽게 작동될 확률이다[8]. 즉, 행정/군수지연시간(Total Administrative & Logistics Delay Time, TALDT), 지원인력이동 시간과 같은 간접적인 요인을 모두 고려한 운용가용도(OA)를 의미한다. 운용가용도(OA)를 산출 식은 (1)과 같다.

$$A_0 = \frac{OT+ST+AT}{OT+ST+AT+TPM+TCM+TALDT} \quad (1)$$

Table 2. Classified time of Combat system management[9]

Total Time	Total Up Time	Operating Time(OT)	
		Non Operating Time	Standby Time
			Alert Time
	Total Down Time	Total Maintenance Time	Total Corrective Maintenance Time
			Total Preventive Maintenance Time
		Total Administrative & Logistic Delay Time	Administrative Delay Time
			Logistics Delay Time

정비도란 무기체계가 고장 발생 시 규정된 정비요원이 가용한 절차 및 자원을 이용하여 주어진 조건하에서 주어진 시간 내에 체계를 정비하여 그 성능을 규정된 상태로 원상 복구하기 위한 시간을 말한다[8]. 무기체계의 모든 정비활동에 소요되는 총 정비시간(Total Maintenance Time, TMT)은 총 예방정비시간(Total Preventive Maintenance Time, TPM)과 총 고장정비시간(TCM)의 합이다. 총 고장정비시간(TCM) 산출 식은 (2)와 같다.

$$TCM = \sum_{i=1}^{N} TF_i \times ET_i \quad (2)$$

i : 장비 고장정비 행위 수
N : 장비 고장정비수의 총합
TF_i : 고장 정비업무의 빈도수
ET_i : i번째 고장의 정비시간

또한, 평균수리시간 (Mean Time To Repair, MTTR)은 정비도 분석을 통해 산출된다. 무기체계가 고장 발생 시 수리 및 복구하기 위해서 소요되는 시간들을 평균한 것이다. 산출 식은 (3)과 같다.

$$MTTR = \frac{\sum_{i=1}^{N} TF_i \times ET_i}{\sum_{i=1}^{N} TF_i} \quad (3)$$

i : 장비 고장정비 행위 수
N : 장비 고장정비수의 총합
TF_i : 고장 정비업무의 빈도수
ET_i : i번째 고장의 정비시간

임의의 무기체계의 총 고장정비시간(TCM)은 135.34로 산출되며, 평균수리시간(MTTR)은 1.427로 산출된다.

4. Automatic Test System

4.1 Definition of Automatic Test System

미 국방부(DoD)에서는 ATS(Automatic Test System)는 시험대상장비의 요구 기능에 대한 성능을 컴퓨터를 이용하여 자동으로 시험하고 분석하는 시스템 체계를 말한다. ATS는 자동화시험장비(ATE)와 TPS(Test Program Set)로 구성된다. 자동화시험장비(ATE)란 컴퓨터를 이용해서 복잡한 시험장비(Digital Multimeter, Signal Analyzer, Signal Generator, etc.)들을 제어해서 시험대상장비의 규격조건을 시험하고 분석할 수 있는 장비를 말

한다. TPS는 일반적으로 Test program software, 시험대상장비와 통신을 위한 인터페이스 장비 및 치구 등으로 이루어진다. 국내에서는 통용되고 있는 자동화시험장비(ATE)의 개념은 자동화시험장비(ATE) 장비와 TPS를 동시 개발한다는 개념이 맞을 것이다. 본 논문에서 자동화시험장비(ATE)란 미 국방부(DoD)에서 정의하는 자동화시험장비(ATE) 및 TPS의 개념을 동시에 포함된 것으로 한다.

자동화시험장비(ATE)의 목적은 무기체계가 기능을 발휘할 수 없을 때, 무기체계의 고장 유무를 판단하고 정비할 수 있는 기능을 제공하는 것이다.[10]

4.2 Classification of Automatic Test Equipment

자동화시험장비(ATE)는 크게 두 가지로 구분한다. 형태에 따라 이동형 자동화시험장비(ATE), 고정형 자동화시험장비(ATE) 구분한다.

이동형 자동화시험장비(ATE)는 장비의 크기와 중량을 소형 및 경량화로 운반이 쉬운 장비이다. 외형적으로 표준화된 기준은 없으며 소요군의 요구사항에 맞게 제작되어진다. 시험대상장비에 대한 LRU(Line Replacement Unit) 단위 하드웨어 기능 검사 및 고장진단 기능을 보유하고 있다.

고정형 자동화시험장비(ATE)는 데스크형, 선반형이 있다. 고정형 자동화시험장비(ATE)는 시험대상장비에 대한 LRU 단위 하드웨어 기능 검사 및 하부 SUR(Shop Replacement Unit) 단위 고장진단/고장배제 기능을 보유하고 있으며, 시험대상장비가 다양한 통합형 자동화시험장비(ATE)이다. Fig. 7과 같이 소요군에 따라 무기체계 정비단계가 구분되어 진다. 정비단계는 3단계 및 5단계로 구분되어 진다. 본 논문에서는 3단계로 구분한다.

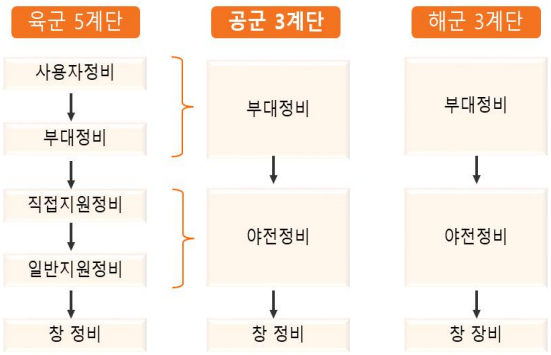

Fig. 7. Maintenance Steps on Weapon System

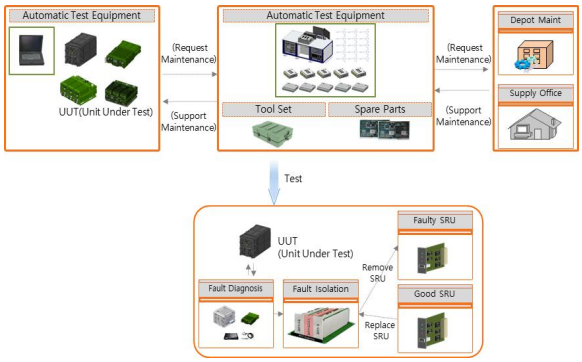

Fig. 8. Maintenance Steps on Weapon System

Fig. 8은 3단계 정비 절차를 세부적으로 표시하였다. 부대정비는 시험대상장비를 사용하는 부대 현장에서 부대 자동화시험장비(ATE)를 활용하여, LRU 단위로 고장 유무를 판단한다. 고장으로 식별된 시험대상장비는 LRU 단위로 교환된다. 부대 자동화시험장비(ATE)는 주로 이동형 자동화시험장비(ATE)를 운용한다.

야전정비는 부대정비 다음 단계로 고장 식별된 LRU를 야전 자동화시험장비(ATE)를 활용하여, LRU단위 또는 SUR 단위로 고장 유무를 판단한다. 고장으로 식별된 시험대상장비를 LRU 단위 및 SRU 단위로 교환된다.

창정비는 야전정비의 다음 단계로 고장으로 식별된 LRU 또는 SRU를 구성하는 하부 구성품의 고장을 판단하는 단계이다. 점검 결과 부품 교체 후 재사용 불가능한 SRU는 폐기한다. 야전장비 와 창장비에서는 고정형 자동화시험장비(ATE)를 운용하고 있다.

4.3 ATE for Software Maintenance

소요군에서 사용되고 있는 자동화시험장비(ATE)는 시험대상장비의 하드웨어 규격 만족 여부를 자동으로 시험하고 분석하는 기능만 하고 있다. 시험대상장비의 요구 기능에 대한 성능을 자동으로 시험하고 분석하는 기능은 없다. 소프트웨어 기능 점검 및 유지보수를 위한 자동화시험장비(ATE)는 다음과 같은 기능이 요구되어 진다.

첫 번째, 다양한 테스트 케이스를 시험할 수 있어야 한다. 하드웨어 규격을 확인하기 위해서는 고정된 한 가지 테스트 케이스로 충분하다. 하지만 소프트웨어로 구현된 기능 고장진단하고 고장을 사전 감지하기 위해서는 다양한 테스트 케이스를 만들 수 있어야 한다. 테스트 케이스를 다양하게 만들고 테스트하기 위해서는 자동화시험장비(ATE)는 시험대상장비를 제어할 수 있는 시뮬레이터가 필요하다.

두 번째, 소프트웨어 사용 패턴과 통신 데이터를 수집하는 것이 가능해야 한다. 소프트웨어를 제대로 유지 보수하

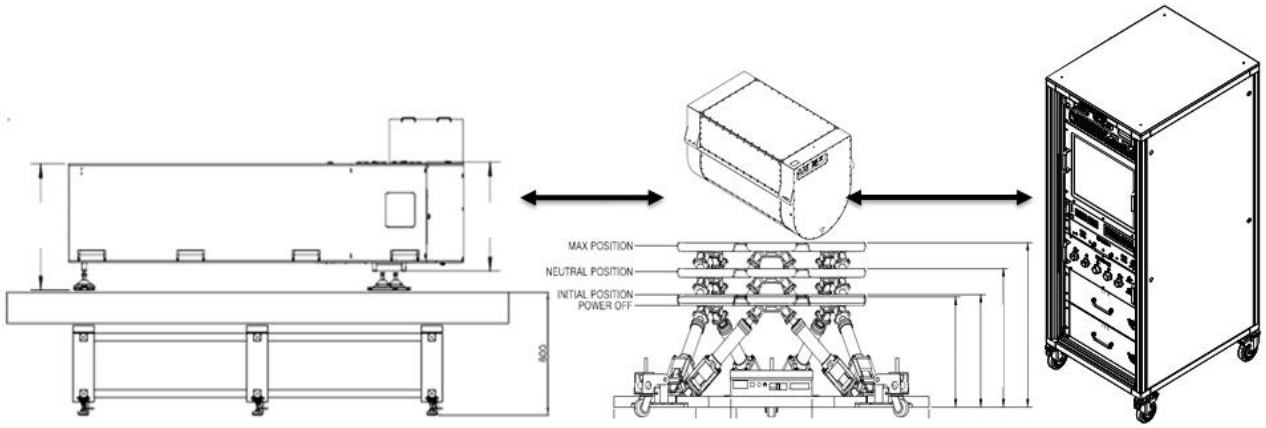

Fig. 9. EO/IR Automatic Test System Configuration

려면 소프트웨어 유지보수 활동에 필요한 적절한 정보가 제공되어야 한다. 예를 들어, 소프트웨어 비정상적 멈춤이 발생한 경우 어떤 모듈에서 비정상 멈춤이 발생했는지, 관련된 모듈들은 어떤 것들인지, 비정상 멈춤 발생 시 어떤 순서로 관련 모듈들이 호출되었는지 등의 정보를 알아야 한다. 또한, 소프트웨어 유지보수에서 사용 패턴에 기반을 둔 테스트를 수행하려면 소프트웨어가 주로 사용되는 방식을 반영한 테스트 케이스를 생성해야 한다. 이를 위해서는 소프트웨어의 실제 사용 패턴에 대한 정보를 자동화시험장비(ATE)가 기록해야 한다.

III. The Proposed Scheme

1. EO/IR ATE

1.1 EO/IR ATE Configuration

EO/IR 점검장비는 Fig. 9와 같이 점검장비, 6자유도 모사기, 시준기로 구성이 되어 있다. 항공기 요동을 위한 6자유도 모사기, 무한 광을 모사를 위한 광학 시준기로 구성되어 있다. 여기서 시준기는 EO 광원 생성을 위한 적분구와 IR 표적 열원을 위한 흑체가 포함되어 있다. 기존 EO/IR 점검장비 구성과 같다.

1.2 EO/IR Test Configuration

시험 구성은 Fig. 10과 같이 하드웨어 기능 점검 및 소프트웨어 기능 점검으로 구분하였다. 소프트웨어 기능 점검은 하드웨어 기능 점검 완료 후 소프트웨어 기능 점검을 수행한다. 하드웨어 기능 점검으로는 전원 단락 및 전원 입·출력 점검, 통신 점검 및 BIT(Built In Test) 점검을 수행한다. 소프트웨어 기능 점검은 동일한 운용 환경을 만들기 위해 시준기 및 6 자유도 모사기를 구동 후, 시험대상 장비를 제어하기 위한 시뮬레이터를 이용하여 소프트웨어 기능을 점검을 시행한다.

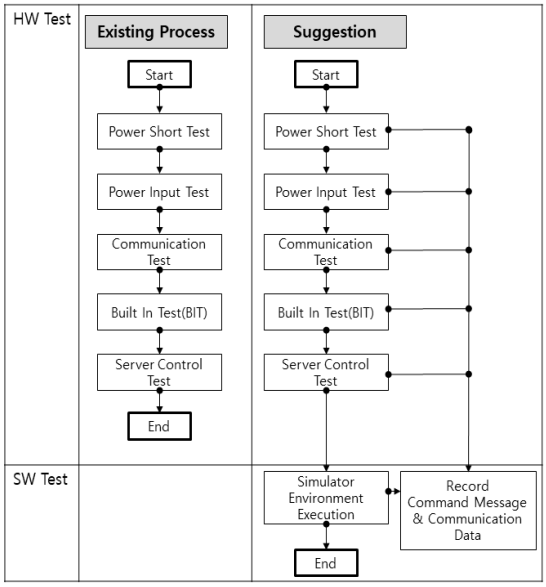

Fig. 10. EO/IR Test Configuration

1.3 Design of User Pattern Record

사용자 패턴을 기록하기 위해 Fig. 11과 같이 자동화시험장비(ATE) 소프트웨어 제어 명령과 시뮬레이터에 제어 명령을 기록하는 부분을 설계하였다. Fig. 12의 자동화시험장비(ATE) 소프트웨어와 시뮬레이터 제어 시, 입력된 정보들이 자동으로 텍스트(*.txt) 파일 형태로 기록된다. 그리고 자동화시험장비(ATE)와 시험대상장비 사이에서 발생하는 통신 데이터도 텍스트(*.txt) 파일 형태로 기록된다. 기록된 패턴 정보를 활용하여 소프트웨어 사용 패턴 기반 테스트 케이스를 효과적으로 만들 수 있다.

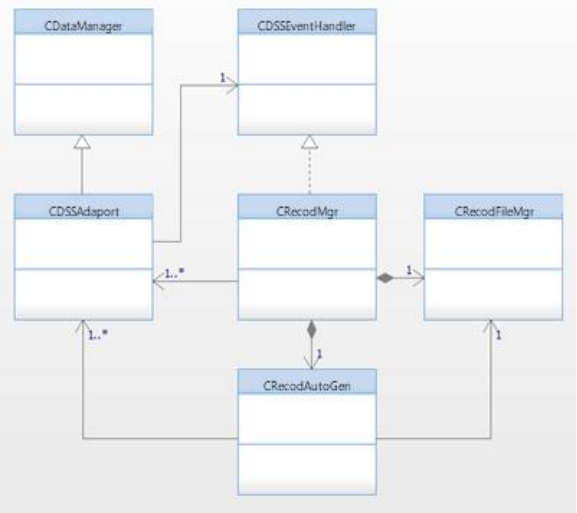

Fig. 11. User Pattern Record Class Diagram

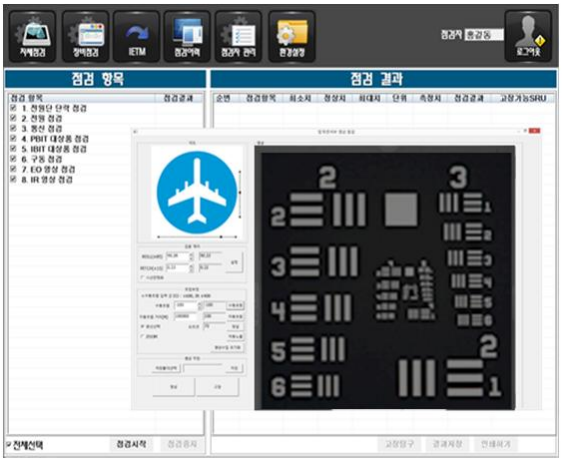

Fig. 12. EO/IR Automatic Test Software and Simulator

1.4 Verification and Test

실험은 임의의 무기체계에 대한 정비도 및 운용가용도를 산출한다. 또한, 산출된 정비도, 운용가용도를 기존 자동화시험장비(ATE)를 활용한 무기체계의 평균수리시간(MTTR), 운용가용도(OA)와 비교하여 제안하는 자동화시험장비(ATE)의 효과를 기술한다.

Table 3는 무기체계 평균수리시간(MTTR) 산출결과이다. 평균수리시간(MTTR)은 무기체계 전체의 총 고장정비시간(TCM)과 관계있는 수치로써 의미가 있다. 업무빈도는 자동화시험장비(ATE) 설계에 독립적인 수치이다. 자동화시험장비(ATE) 적용여부에 따라 변하지 않는다. 그러나 제안하는 자동화시험장비(ATE) 시험대상장비별 고장정비소요시간을 감소시켜 총 고장정비시간(TCM)을 감소시킨다. 이로 인해 제안하는 자동화시험장비(ATE)의 적용 시, 식 (3)을 통해 산출된 무기체계의 평균수리시간(MTTR)은 기존 대비 약 32.6%가 향상된다.

Table 3. Comparison of Maintainability by Automatic Test Equipment

	$\sum$ Task Freq.	TCM ($\sum$ (Task Freq. x Time))	MTTR
Existing	94.842	135.34	1.427
Proposed	94.842	91.23	0.962

Table 4는 식 (1)을 통해 산출된 무기체계 운용가용도(OA) 산출결과이다. 무기체계 설계 목표에 따라 연간 운용시간(OT), 총 행정/군수지연시간(TALDT) 값은 고정된다. 총 예방정비시간(TPM)은 자동화시험장비(ATE)와 독립적인 요소로 자동화시험장비(ATE) 없이 예방 정비하는 시간으로 기존과 동일하다. 대기시간(Standby Time, ST)는 총 시간(Total Time, TT)에서 연간 운용시간(OT), 총 예방정비시간(TPM),총 고장정비시간(TCM), 총 행정/군수지연시간(TALDT)을 뺀 값이다. 여기서, 총 시간(TT)은 연간 시간(8,760시간)에서 시간기준정비(TBM) 12주(2,016시간)을 제외한 6,744시간이다. 기존대비 운용가용도가 0.50%가 향상된다.

Table 4. Comparison of Operational Availability by Automatic Test Equipment

	OT	ST	TPM	TCM	TALDT	AO (%)
Existing	3,150	3,274.32	112.34	135.34	72	96.35
Proposed	3,150	3,318.43	112.34	91.23	72	96.85

미 국방부(DoD)에서 분류한 유지관리 4가지 유형 중 제안하는 자동화시험장비(ATE)를 활용한다면, Table 5와 같이 유지관리 대상 소프트웨어의 오류 진단 및 오류를 사전에 감지할 수 있다. 소프트웨어 수정은 불가하지만, 소프트웨어에 대한 신규 기능 추가, 기능 변경, 기능 삭제에 대한 정보 식별은 가능하다. 또한, 수집된 정보를 제조사(개발자)에 제공하여, 데이터 분석을 통해 고장정비소요시간이 획기적으로 감소하며, 소프트웨어 진단, 소프트웨어 사용 패턴 기반 테스트, 회귀 테스트 케이스 집합의 확장 등에 필요한 정보를 획득할 수 있다.

Table 5. Comparison of Software Maintenance by ATE

	Existing	Proposed
Corrective Sustainment	disable	Enable
Preventive Sustainment	disable	Enable
Adaptive Sustainment	disable	Enable
Perfective Sustainment	disable	disable

IV. Conclusions

본 논문은 무기체계 소프트웨어 기능 점검 및 유지보수를 위한 방법으로 시뮬레이터를 활용한 자동화시험장비를 설계하였다. 2장에서 소프트웨어 신뢰성에 대한 이론적 배경, 사용 패턴 기반 테스트를 통해 발생빈도가 높은 장애율의 에러를 발견하고 소프트웨어를 수정하는 편이 시스템의 평균고장시간(MTTF)에 미치는 효과에 대한 내용을 서술하였다. 그리고 총 고장정비시간(TCM)과 정비도 및 운용가용도(OA)의 관계를 산출식을 통해 설명하였고, 자동화시험장비(ATE) 정의와 분류 및 소프트웨어 유지보수를 위한 요구사항들을 서술하였다. 3장에서는 EO/IR 자동화시험장비 구성, 시험 구성, 사용자 패턴 기록을 위한 설계, 마지막으로 시험을 통해 정비도 수치인 평균수리시간(MTTR) 32.6%, 운용가용도 0.50%가 향상과 소프트웨어 유지보수의 효과성을 검증하였다. 이렇게 제안된 자동화시험장비(ATE)는 기존 대비 정비도 및 운용가용도가 개선됨을 입증하였다. 그러므로 본 연구에서 제시한 시뮬레이터를 활용한 자동화시험장비(ATE)를 사용한다면 무기체계 소프트웨어 유지보수 시 뛰어난 정비도 및 운용가용도(OA)를 유지하여 국방력 강화에 적지 않게 기여될 것으로 판단된다.

추후 연구 과제에서는 본 논문의 연구결과를 기초로 자동화시험장비(ATE) 실무에 적용할 수 있는 더욱 구체화한 방안을 제시하고, 소프트웨어 유지보수 기능들을 실질적인 검증하는 작업이 필요하다. 무엇보다 현재 무기체계에 있어 소프트웨어 중요성을 인지하고 있지만, 소프트웨어 유지보수 실무에서 적용되지 못하고 있는 문제를 현실적으로 해결하여 이른 시일 내에 실무에 적용하는데 이바지할 수 있을 것으로 기대한다.

REFERENCES

[1] Il-Hoon Cho, Seong-Huk Hwang, Ik-Do Lee, Yeonk-Yeong Park, Jung-hoon Lee, Chang-Hoon Shin, “The Case Study on Application of Software Reliability Analysis Model by Utilizing Failure History Data of Weapon System”, Journal of Applied Reliability 17(4), pp. 296-304, 2017.12

[2] Seo, YoungHee, Kang, Dongsu, “Current Status of US Weapon System Software Management and Its Implications”, MONTHLY SOFTWARE ORIENTED SOCIETY, pp. 23-28, 2020.01

[3] Defense Acquisition Program Administration. “Defense Acquisition Management Regulation”, DAPA LOI 361, Article 36, Vol. 311, pp. 14-159, 2016

[4] Defense Acquisition Program Administration. “Manual of Softeare Development and Management in Weapon System Software Development”, DAPA Manual, pp. 100, 2016-4

[5] DOD Instruction(DODI) 4151.20, Depot Maintenance Core Capabilities Determination Process, GAO-19-173, May 4, 2018

[6] Zhonglin he, G. Staples, Margaret Ross, I. Court, "Software black box mechanism: a pragmatic method for software crash diagnosis and usage maintenance testing", 13th International Conference on Software Maintenance (ICSM'97), DOI:10.1109/ICSM 1997.624240

[7] Michael Dyer, "The Clean room Approach to Quality Software Development" John Wiley & Sons, 1992.3

[8] Defense Acquisition Program Administration, “Integrated Logistics Support Development guidelines", pp. 1-373, 2015

[9] Rim-Hwan Lee, “The design of a Portable Automatic Test Equipment for Operational Availability of Combat System” Journal of the Korea Academy Industrial Cooperation Society 21(3), pp. 453-459, 2020.3

[10] Seok-Min Lee, “A study on Automatic field Test Equipment with improved maintenance and environmental reliability”, Journal of the Korea Society of Computer and Information 23(3), pp. 9-16, 2018.3

Authors

Il-Kwon Chae received the B.S. degree in Electrical and Electronics Engineering from Korea Maritime & Ocean University, Korea, in 2014. He is currently a Engineer in the Naval R&D Center, Hanwha Systems Co., Ltd. He is interested in Software Maintenance and Combat Management System Software.