KIPS Trans. Softw. and Data Eng.
Vol.9, No.11 pp.357~366
ISSN: 2287-5905 (Print), ISSN: 2734-0503 (Online)

https://doi.org/10.3745/KTSDE.2020.9.11.357

Failure Analysis of Aircraft Software Test Cases from a Perspective of Requirements Traceability

Sung-Sub Kim† · Hee-Tae Cho†† · Seonah Lee†††

ABSTRACT

As the proportion and complexity of software embedded in aircraft increase, risk factors such as mission failure, function failure and performance failure due to software errors also increase. In the mission-critical software systems such as aircraft software, managing requirement traceability is essential to maintain the software systems with minimal period and cost. However, the development company is not accurately complying with the guideline for managing requirement traceability due to various reasons such as development cost and schedule. Therefore, it is not easy to systematically establish and maintain requirement traceability. In the paper, we analyze actual test cases of aviation software systems from the viewpoint of requirements traceability in order to learn if there are failure cases of test cases due to the absence of systematic traceability management activities. We also check the risks associated with the failure cases according to the type and severity of the cases. As a result of analyzing a total of 7 aircraft-mounted software, failure cases could be divided into three types: omission of requirements, lack of connection between requirements and test procedures, and omission of test procedures. There were a total of 18 failure cases, 6 for each type. The numbers of high, middle and low risks were 1, 13 and 4, respectively, where the number of middle risks is largest.

Keywords : Software Engineering, Aircraft Software, Requirement Traceability, Test Cases

요구사항 추적성 관점에서 항공기 탑재 소프트웨어 시험 사례 실패 분석

김 성 섭† · 조 희 태†† · 이 선 아†††

요 약

항공기 탑재 소프트웨어의 비중 및 복잡성 증가에 따라, 소프트웨어 고장으로 인한 임무수행이나 기능 실패 및 성능 미달 등의 위험요인 또한 증가하고 있다. 항공기 탑재 소프트웨어처럼 미션 크리티컬 도메인의 대규모 소프트웨어에서 최소한의 기간과 비용으로 유지보수를 진행하기 위해서는 요구사항 추적관리가 필수적이다. 하지만, 개발업체에서는 개발비용이나 개발일정 등의 여러 사유로 인해 요구사항 추적관리 지침을 정확히 준수하지 못하고 있으며, 체계적으로 추적성 수립 활동을 수행하기란 쉽지 않다. 논문에서는 체계적인 추적성 수립 활동의 부재에 따른 요구사항 추적성 실패사례가 실제로 존재하는지 항공분야 소프트웨어의 실제 시험사례를 요구사항 추적성 관점에서 분석하고, 추적성 실패사례의 유형과 사안 경중에 따른 위험을 확인한다. 총 7개의 항공기 탑재 소프트웨어를 대상으로 분석을 진행한 결과, 실패사례는 총 3가지 유형인 요구사항 누락, 요구사항-시험절차 연계부족, 시험절차 누락으로 나눌 수 있었다. 실패사례는 총 18건으로 각 유형별 6건씩 있었으며, 각 사안에 따른 Risk는 High, Middle, Low 순으로 각각 1건, 13건, 4건으로 중간 수준의 위험이 가장 많았다.

키워드 : 소프트웨어 공학, 항공기 소프트웨어, 요구사항 추적성, 시험 사례

1. 서 론

항공기에서 1960년대 F-4에서 8%였던 소프트웨어의 비중이 2000년대 F-22에서 80%, F-35에서 90% 점점 증가하고 있다[1]. 항공기 탑재 소프트웨어의 비중 및 복잡성 증가에 따라, 소프트웨어 고장으로 인한 임무수행이나 기능 실패 및 성능 미달 등의 위험요인 또한 증가하고 있다. 이러한 변화는 항공기 탑재 소프트웨어에서 시험을 통한 안전성 및 신뢰성 확보뿐만 아니라, 유지보수의 중요성을 증가시킨다. 항공기 탑재 소프트웨어처럼 미션 크리티컬 도메인의 대규모 소프트웨어에서 최소한의 기간과 비용으로 유지보수를 진행하기 위해서는 요구사항 추적관리가 필수적이다.

대한민국 방위사업청에서는 무기체계 소프트웨어 개발 및 관리 매뉴얼을 지속적으로 개정하여, 국내에서 개발되는 무기

※ 본 연구는 한국연구재단의 연구비(NRF-2018R1D1A1A02085551) 지원으로 수행하였습니다.
† 비 회 원 : 한국항공우주산업(주) KFX임무SW팀 선임연구원
†† 비 회 원 : 경상대학교 AI융합공학과 박사과정
††† 종신회원 : 경상대학교 항공우주 및 소프트웨어공학전공/AI융합공학과 부교수
Manuscript Received : August 26, 2020
Accepted : September 20, 2020
* Corresponding Author : Seonah Lee(saleese@gnu.ac.kr)

체계 소프트웨어의 체계적인 개발 및 관리를 통해 신뢰성을 확보하고자 노력하고 있다. 방위사업청의 매뉴얼이나 DO-178(항공용 소프트웨어 감항 인증)에서는 소프트웨어 개발 및 문서 작성 가이드라인을 제시하여, 비가시성의 소프트웨어에 대한 문제를 관리할 수 있도록 하며, 요구사항 추적관리에 대한 철저한 지침을 제공한다. 요구사항 추적관리는 식별자 체계에 따라 작성된 요구사항 식별자, 설계 식별자, 시험 식별자 등의 식별자 번호로 요구사항-설계-시험을 서로 연결해 누락 없이 관리하는 것이다. 요구사항 추적성은 상위요구사항-하위요구사항-소스코드-시험절차까지 모든 요구사항이 추적되어야 한다.

만약 이러한 추적성을 수립하지 않는다면 검증에 더욱 많은 시간과 노력이 들 수 있다. 예를 들어, 요구사항이 누락된 채로 하위수준의 시험을 진행하면, 상위수준의 시험에서 해당 문제를 발견하고 보완할 수 있다. 하지만 보다 많은 시간과 비용이 소모된다. 이러한 상황에서 요구사항 추적성이 있었다면, 하위수준의 시험에서 요구사항의 누락을 찾을 수 있어 적은 시간과 비용으로 보완할 수 있다. 이와 같은 상황을 사전에 예방하고 개발일정 및 비용을 단축하기 위해 가급적 개발 초기에 추적성을 수립 및 추적관리가 필요하다. 이러한 추적성의 이점에도 불구하고, 개발업체에서는 개발비용이나 개발일정 등의 여러 사유로 인해 요구사항 추적관리 지침을 정확히 준수하지 못하고 있으며, 체계적으로 추적성 수립 활동을 수행하기란 쉽지 않다.

본 논문에서는 체계적인 추적성 수립 활동의 부재에 따른 요구사항 추적성 실패사례가 실제로 존재하는지 항공분야 소프트웨어의 실제 시험사례를 요구사항 추적성 관점에서 분석하고, 추적성 실패사례의 유형과 사안 경중에 따른 Risk를 확인한다. 분석 대상은 개발업체의 규모, 소프트웨어의 크기, 소프트웨어의 기능 등을 고려해 분석의 결과가 치중되지 않도록 하며, 총 7개의 항공기 탑재 소프트웨어를 선정한다. 분석 결과, 실패사례는 총 3가지 유형인 요구사항 누락, 요구사항-시험절차 연계부족, 시험절차 누락으로 나눌 수 있었다. 실패사례는 총 18건으로 각 유형별 6건씩 있었으며, 각 사안에 따른 Risk는 High, Middle, Low 순으로 각각 1건, 13건, 4건으로 Middle Risk가 가장 많았다.

본 논문의 나머지 구성은 다음과 같다. 2장에서는 본 연구의 관련 연구를 소개한다. 3장에서는 방위사업청 지침에 관련된 배경 지식을 서술한다. 4장에서는 요구사항 추적성 분석 방법을 제시한다. 5장에서는 요구사항 추적성 분석 결과를 제시한다. 마지막으로 6장에서는 결론을 지으며 논문을 마무리한다.

2. 관련 연구

본 연구는 실제 시험사례를 요구사항 추적성 관점에서 분석한다. 본 2절에서는 관련 연구를 소개하며 크게 3가지로 나눌 수 있다. 먼저 요구사항 추적성을 정의하는 연구이다. 다음으로 요구사항 추적성 설계 연구이며, 마지막으로 요구사항과 시험 사이의 추적성 연구가 있다.

2.1 요구사항 추적성 정의 연구

요구사항 추적성은 요구사항을 정방향 및 역방향으로 설명하고 준수할 수 있는 능력을 말한다[2]. 요구사항 추적은 V모델 개발프로세스를 기반을 둔 것으로 개발프로세스 생명주기 동안 진화하는 요구사항에 대해 각 단계별로 요구사항의 일관성을 검증하고 단위시험, 통합시험, 시스템통합시험 등을 거쳐 요구사항의 완전성을 확인하는 것이다[3]. 요구사항 추적성 분석은 소프트웨어 개발과정에서 생산되는 산출물들이 어떠한 요구사항부터 생산되는 것인지 추적관계를 통해 분석하는 것을 의미하고[4], 시스템 요소를 위한 요구사항과 계층관계에 있는 요구사항이나 설계, 구현 등과 같은 다양한 산출물과 연결하는 프로세스이다[5]. 소프트웨어사업 요구사항 명세화 표준지침 개발을 연구는 제품의 품질 불만과 수익성 악화의 원인을 소프트웨어 개발사업 초기의 불명확한 요구사항 명세서로 인식하여, 보다 명확한 요구사항 명세와 요구사항의 효과적인 추적 관리를 위해 요구사항 명세화 표준지침을 개발하였다[6]. 요구사항 명세서에 첨부하는 요구사항 추적표 작성 양식 제안하는 연구는 요구사항 추적표를 거치지 않아 개발 마지막 단계에서 납기 지연, 과도한 비용 증가와 함께 개발 시스템 품질에 막대한 영향을 미치는 문제를 인식했다. 이에 모든 요구사항이 나오게 된 근거에서부터 설계 및 시험에 이르기까지 모든 정보를 양방향으로 추적성을 확보를 위한 요구사항 추적표를 제안하였다[7]. 추적테이블을 제시하는 연구는 기존의 요구사항 추적연구는 구체적인 추적대상물을 정의하지 않거나 정보검색기법, 이벤트기반기법, 시나리오기반기법등과 같이 추적모형을 구현하고 유지 보수하는데 비용이 많이 든다는 문제를 제시하며 추적테이블을 제시했다. 또한 제시한 추적테이블을 이용해 실제 수행한 15개의 프로젝트에서 추적 테이블이 프로젝트 성공에 영향을 확인하였다[8].

2.2 요구사항 추적성 설계 연구

경전철 개발 사업에서 체계공학 전산지원도구를 활용한 요구사항 추적사례 연구는 규모 및 복잡도가 큰 프로젝트에서 문서 단위의 요구사항 추적관리 시 많은 비용과 시간 투입이 불가피하여 체계공학 전산지원도구 활용할 것을 제안하며 재작업을 최소화할 수 있다고 주장하였다[9]. 또한 산업계 체험보고서 중 임베디드 소프트웨어에 대한 요구사항 추적성 연구에서 추적 관계 생성 및 유지보수를 결정할 때 고려해야 할 많은 문제가 있다고 주장하였다. 소프트웨어 설계문서 내 또는 문서 간 설계에서 구현 및 시험에 이르기까지 많은 추적관계가 있고, 프로젝트를 수행함에 있어서 보다 적은 비용으로 더 많은 이점을 고려하여 여러 추적관계의 조합 중 최선의 선택이 필요하다고 주장하였다[10].

2.3 요구사항과 시험 사이의 추적성 연구

테스트-코드간 추적성은 소프트웨어 개발에서 일반적이지 않기 때문에 테스트 사례와 소스코드간 추적 관계를 설정하는 메커니즘을 통해 소프트웨어 유지 관리를 지원하는 연구를 진행했다[11]. DO-178C(항공용 소프트웨어 감항인증)에 적합한 정적분석을 활용한 소프트웨어 동적시험 프로세스 구축 사례 연구는 소스코드와 불일치하는 설계문서를 근거로 테스트 설계를 수행하고 힘든 수작업에 의존하여 테스트 케이스를 작성하는 등 테스트 수행에 많은 문제를 안고 있는 점에 착안하여 정적분석의 결과를 활용하여 테스트 케이스를 자동 생성하여 요구사항과 소스코드 간 추적되지 않는 문제를 해결할 수 있는 방안을 제안하였다[12]. 또한 DO-178C(항공용 소프트웨어 감항인증)에 적합한 요구사항기반 시험 절차 및 Test Coverage 달성 방안연구는 요구도 기반의 동적시험에 대한 구체적인 절차와 방법을 찾기 위해 DO-178C를 참조하여 연구를 진행했으며, 결과 DO-178C의 테스트 요건(테스트 절차, 테스트 케이스 생성 및 수행, 테스트 결과 분석, 테스트 커버리지 달성 등)에 적합하며 효율적으로 수행할 수 있는 소프트웨어 시험 프로세스 구축 및 Test Coverage 달성방안을 제시하였다[13].

3. 배경 지식

본 연구는 방위사업청의 지침에서 제공하는 기술문서 작성 가이드에 따라 표준 템플릿을 적용하여 추적성 실패사례를 찾아낸다. 본 3절에서는 본 논문에서 사용하는 요구사항 추적성과 분석에 사용하는 동적 시험을 서술한다.

3.1 방위사업청 지침에서의 요구사항 추적성

대한민국 방위사업청 지침[14]에서 제시하는 요구사항 추적성은 각 문서마다 고유의 요구사항 식별자를 만들어 상/하위의 문서들을 식별자로 연결하는 것이다. 방위사업청 지침에서 요구하는 추적성 매트릭스는 Fig. 1과 같다.

Fig. 1에서 SRS(소프트웨어 요구사항 명세서)는 상위요구사항으로 BK-SRS-001로 식별된 하나의 기능에 대해 기술한 문서이다. SDD(소프트웨어 설계 기술서)는 SRS를 기반으로 상세설계를 진행해 나온 산출물로 하위요구사항을 기술한 문서이다. 정방향 추적성은 상위요구사항에서 하위요구사항을 추적하는 방향으로 상위요구사항인 SRS를 기준으로 작성한다. 반대로 역방향 추적성은 하위요구사항에서 상위요구사항을 추적하는 방향으로 하위요구사항인 SDD를 기준으로 작성한다. 이러한 요구사항 매트릭스에는 SRS, SDD외에도 STD(소프트웨어 시험 절차서)가 추가로 들어갈 수 있다.

3.2 방위사업청 지침에서의 동적 시험

방위사업청 지침에서 2016년에 요구사항과 시험간의 추적성 확인을 위해 요구사항 기반의 동적 시험의 개념을 도입하였다. 동적 시험은 소프트웨어의 개발수준에 따라 Statement, Branch, MC/DC Coverage로 분류하여 시험을 진행하며, 개발수준이 중요할수록 더 까다로운 시험이 진행된다. 요구사항 기반의 동적 시험은 소프트웨어 요구사항 명세서에 정의된 요구사항들이 구현 및 통합된 소프트웨어로 실제 타깃 하드웨어에서 소프트웨어 통합시험 절차서에 따라 기능시험을 수행하며, 명세의 오류나 기능적 결함을 검출하고 code coverage를 측정 및 점검하는 시험이다. 하지만 요구사항 기반의 동적 시험만으로는 소스코드의 방어코드는 실행되지 않는다. 따라서 방어코드에 대한 소명자료를 작성하여 해당 코드는 문제가 없음을 증명해야 한다.

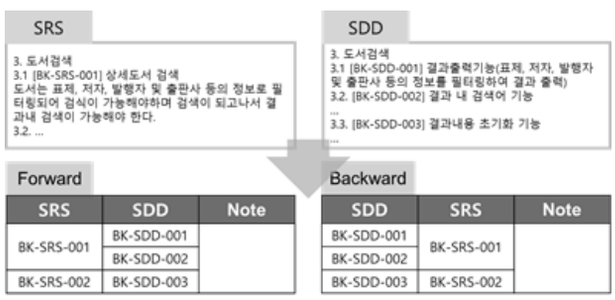


Fig. 1. Example of a Requirements Matrix from the Defense Acquisition Program Administration Guidelines

또한 동적 시험에서 코드 실행률 100%를 달성해야 되기 때문에, 시험절차를 최대한 상세하게 작성해야 한다. 동적 시험에서 결함이 식별되면 요구사항명세서-설계시술서-시험절차서-시험결과서 등 소프트웨어 기술문서를 분석해 결함의 위치를 찾고, 수정한 후 시험을 반복해야 하기 때문에 코드 실행률 100%를 충족하는 것은 많은 시간과 노력이 필요하다. 특히, 임베디드 환경에서는 코드 실행률을 확인하기 위해 탐침코드를 넣고 빌드해야 하기 때문에 메모리나 CPU의 사양도 고려해야 한다.

4. 분석 방법

본 4절에서는 분석 대상인 실제 시험사례를 소개하며, 이 사례들을 요구사항 추적성 관점에서 분석하기 위해 본 논문에서 사용한 방법을 제시한다.

4.1 실제 시험사례(분석 대상)

분석 대상은 개발업체의 규모, 소프트웨어의 크기, 소프트웨어 기능 등을 고려해 특정 분야에 치우치지 않고, 비슷한 분석 결과가 나오는 것을 방지하고자 하였으며 총 7종의 품목을 선정했다. Table 1은 선정한 7종의 소프트웨어를 각 항목별로 분류한 표이다.

Table 1에서 SRS(Software Requirements Specification)는 소프트웨어 요구사항 명세서로 고객의 요구사항을 소프트웨어 요구사항으로 세분화하여 작성한 문서이다. SDD(Software

Table 1. Information of Analysis Target

Index	Content	SRS	SDD	STD	SLOC	Company size
A	Electricity	90	250	160	11,000	S/M
B	Avionics	40	80	90	5,000	H
C	Avionics	160	620	1,000	46,000	H
D	Fuel	160	180	300	20,000	H
E	Cockpit	40	70	120	5,000	S/M
F	Secondary Power	160	850	1,000	80,000	H
G	Avionics	30	70	90	1,700	S/M
Sum		680	2,120	2,760	168,700	

Design Description)는 소프트웨어 설계 기술서로 SRS에서 식별한 요구사항을 바탕으로 상세설계를 진행한 산출물로 하위요구사항이 작성된 문서이다. STD(Software Test Description)는 소프트웨어 통합 시험 절차서로 SRS와 SDD에서 식별한 상위, 하위 요구사항을 바탕으로 구현된 소스코드를 하드웨어에 탑재하여 시험할 때의 절차를 작성한 문서이다. SLOC (Source Lines Of Code)는 SDD를 바탕으로 구현한 소스코드를 수치화한 데이터로 소스코드의 라인수이다.

각각의 분석 대상들을 간략히 설명하면 다음과 같다.

1) A: 배터리의 전압, 전류, 온도를 측정하고, 이를 기반으로 충전 상태 및 전류량을 추정한다. 배터리 상태가 한계 설정치를 상회하는 경우 정상적인 동작이 지속될 수 있도록 자가 보호기능을 수행하며, 통신을 통해 사용자와 연동된 제어기기에 그 상태를 표시하거나 전송한다.
2) B: 전파를 이용해 지상과 항공기 사이의 절대고도를 제공하기 위한 장비로, 소프트웨어는 항공기 고도측정을 위한 신호처리 및 고도변화율 측정 등의 기능을 수행한다.
3) C: 초기화 설정과 영상 시현에 필요한 각종 제어, 터치 입력처리, 화면 모드/밝기 제어 등 운용기능과 SW로딩, 결함탐지/식별/보고를 위한 BIT (Built In Test)관리 등의 기능을 제공한다.
4) D: 연료펌프와 밸브를 제어하고, 각 연료탱크별/그룹별 전체 연료량, 연료펌프와 연료밸브의 상태정보, 연료온도/압력 정보 및 저연료 주의 신호를 항공기 체계로 전달한다.
5) E: 시현기에서 입력되는 신호를 입력받아 조종실 조명 제어장치와 연동되는 패널, 신호등, 스위치 등의 조명 밝기를 조절하는 기능을 수행한다.
6) F: 이차동력계통 구성품을 지속적으로 제어하고 정상작동 여부를 감시하며, 구성품의 상태정보를 항공기 체계에 제공하는 기능을 수행한다.
7) G: 다수의 중복 주파수를 사용하는 송/수신기 간의 상호 간섭을 막기 위한 항공기의 송/수신기 운용을 조정하는 장치이다.

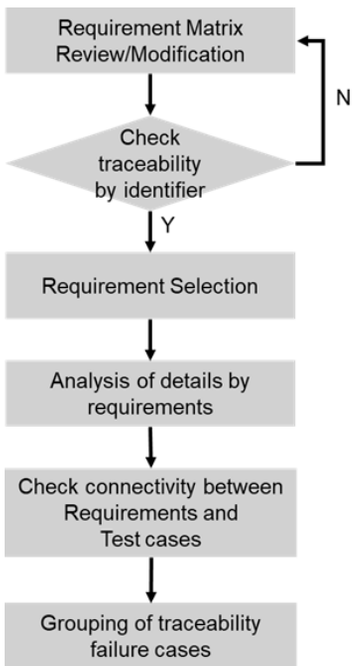


Fig. 2. Analysis Method in Terms of Traceability between Requirements and Test Cases

4.2 요구사항-시험 사이의 추적성 분석 방법

본 논문에서 소프트웨어 시험 사례를 요구사항 추적성 관점에서 분석하는 방법은 Fig. 2와 같다.

1) 먼저 상/하위 요구사항 간의 관계를 검토한다. 소프트웨어에 할당된 각각의 상/하위 요구사항들을 추적할 수 있도록 모든 요구사항의 식별자를 Matrix에 기술하고, 해당 요구사항에 맞는 시험 식별자와 테스트 케이스를 동일한 Matrix에 기술한다.
2) 다음으로 식별자와 테스트 케이스가 기술된 Matrix에 누락사항이 없는지 확인하고 모든 요구사항이 추적가능한지 확인한다.
3) 식별자의 누락이 없고 추적이 가능하다면 분석을 진행할 기능 요구사항(SRS)을 선정한다.
4) 기능 요구사항을 선정하면, 다음과 같은 기준으로 세부내용을 분석한다.

a) 명세된 요구사항의 구현 가능성, 명세 표현의 정확성 및 완전성, 표준과의 일치성, 요구사항 간의 충돌, 기술적 결함을 확인한다.
- 고객의 요구를 에러없이 완전하게 반영했는가?
- 요구사항 간에 서로 모순되지 않는가?
- 요구 분석의 내용이 모호함 없이 모든 참여자들에 의해 명확하게 이해되는가?
- 요구사항 명세서가 “무엇을”에 관점을 두고 기술되었는가?
- 요구사항 명세서에 기술된 내용이 사용자의 요구를 만족하는가?

Table 2. Example of Writing Test Input on STD

<table>
<tr><th colspan="2">Contents</th><th>Conditions</th></tr>
<tr><td colspan="2">시험입력 명칭</td><td>요구도 제목 또는 유사하게 기술</td></tr>
<tr><td colspan="2">시험입력 목적, 설명</td><td>시험 목적 기술</td></tr>
<tr><td colspan="2">시험입력 사용방법</td><td>...</td></tr>
<tr><td colspan="2">시험 입력 값 실제 자료 여부</td><td>입력값이 실제 값인 경우 “실제값”, 모의 값인 경우 “모의값” 기술</td></tr>
<tr><td colspan="2">시험입력 순서</td><td>...</td></tr>
<tr><td rowspan="4">입력자료 통제방식</td><td>자료종류 및 최소/최대 적정자료 수</td><td>자료 종류: 초기화 값 <- 요구도 명과 관련하여 정의
<table><tr><th>항목</th><th>Input값/상태</th></tr><tr><td>ADC초기화</td><td>전원인가</td></tr><tr><td>PWM초기화</td><td>전원인가</td></tr><tr><td>제어 변수 초기화</td><td>전원인가</td></tr></table>최소/적정 자료 수: 3개의 Test Scenario로 구성</td></tr>
<tr><td>과부하, 최악의 자료 종류/값</td><td>정상적인 경우를 제외한 경우 발생 예상시 반영</td></tr>
<tr><td>비규칙적인 입력 자료 종류/값</td><td>정상적인 경우를 제외한 경우 발생 예상시 반영</td></tr>
<tr><td>재시험 기준</td><td></td></tr>
</table>

- 개발된 시스템이 요구사항 분석 내용과 일치하는지 검증할 수 있는가?

b) 시험 절차서에 기술된 Test Case는 다음 사항을 만족해야 한다.
- 요구사항에 대해 검증하려는 시험의 목적이 구체적인가?
- 요구사항에서 정의하는 조건에서 입력값에 따른 시험 항목을 식별할 수 있는가?
- 시험 항목이 결정된 후 발생할 수 있는 Test Case를 식별하고 예상값을 작성했는가?
- 작성된 예상값은 Pass/Fail을 명확히 구분할 수 있도록 가능한 수치로 작성되었는가?

Table 2는 시험절차서의 작성 예시이다. 시험절차서는 바로 아래의 요구도 예시를 기반으로 작성한 예시이다.

요구도 예시) “XX는 전원이 인가되면 ADC, PWM, 제어변수를 초기화 하여야 한다.”

Table 2의 “자료 종류 및 최소/적정 자료 수”에 항목별 Input값/상태는 요구도에서 명시된 “전원인가 시 ADC, PWM, 제어변수 초기화”를 3개의 Test Scenario로 구성하여 요구도의 모든 내용을 누락 없이 작성한다. 다음으로 Table 3은 시험절차서 내 시험예상결과의 작성 예시이다. Table 3은 요구도에서 정의하는 전원인가 시 ADC, PWM, 제어변수 초기화에 대한 Test Case를 구체화하는 단계로 각 Case별로 예상값을 작성한다. 그런 다음 항목마다 시험절차를 작성하고 시험절차와 동일한 번호를 표에 기입함으로써 요구사항에서 시험절차까지 추적할 수 있다.

Table 3. Example of Writing Expected Results on STD

<table>
<tr><th>Contents</th><th>Expected Results</th></tr>
<tr><td>정상적인 자료 입력 시</td><td><table>
<tr><th>구분</th><th>발생 Case</th><th>예상값 / 예상 상태</th><th>시험절차 번호</th></tr>
<tr><td rowspan="2">ADC 초기화 여부</td><td>초기화된 경우</td><td>a</td><td>2</td></tr>
<tr><td>초기화 안된 경우</td><td>a`</td><td>3</td></tr>
<tr><td rowspan="2">PWM 초기화 여부</td><td>초기화된 경우</td><td>b</td><td>4</td></tr>
<tr><td>초기화 안된 경우</td><td>b`</td><td>5</td></tr>
<tr><td rowspan="2">제어 변수 초기화 여부</td><td>초기화된 경우</td><td>c</td><td>6</td></tr>
<tr><td>초기화 안된 경우</td><td>c`</td><td>7</td></tr>
</table>요구도를 Test Scenario(3개)별 2개의 Test Case (6개)로 구분해 예상값 반영
- 유효한 경우 뿐만 아니라 유효하지 않은 경우도 추가로 필요하지만, 요구도 기능에 따라 유효한 경우가 없을 수 있으므로 검토/확인 필요</td></tr>
<tr><td>과부하, 최악의 자료 입력시</td><td>정상적인 경우를 제외한 경우 발생 예상시 반영</td></tr>
<tr><td>비규칙적인 자료 입력시</td><td>정상적인 경우를 제외한 경우 발생 예상시 반영</td></tr>
<tr><td>기타 예상결과</td><td>...</td></tr>
</table>

5) 요구사항-Test Case 사이의 연계성 및 요구사항 기반 Code Coverage를 확인한다.

a) 요구사항-Test Case 사이의 연계성

요구사항과 시험절차에 대해 분석한 내용을 바탕으로 추적성에 누락이 있는지 확인한다.

b) 요구사항 기반 Code Coverage 확인

요구사항 기반 Code Coverage 확인은 소프트웨어 신뢰성시험 도구를 활용해 결과를 확인할 수 있다. Fig. 3의 절차처럼, 요구사항대로 구현한 소스코드에 탐침 코드를 추가해 빌드 후 타깃에 업로드해 시험절차대로 수행하면 Fig. 4와 같이 신뢰성시험 도구에서 Code Coverage를 확인할 수 있다. 요구사항 기반의 Code Coverage는 요구사항 Coverage와 코드 Coverage를 동시에 만족해야하기 때문에 시험절차를 최대한 구체적으로 작성해야 모든 코드를 실행할 수 있으며,

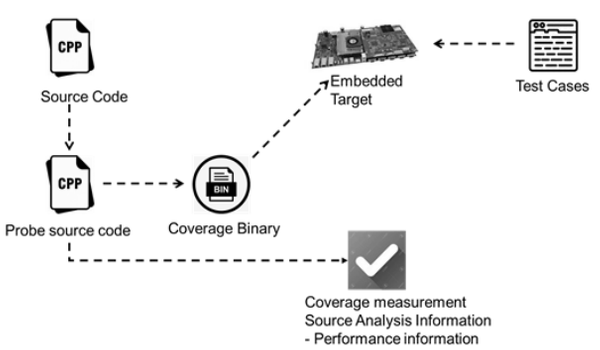
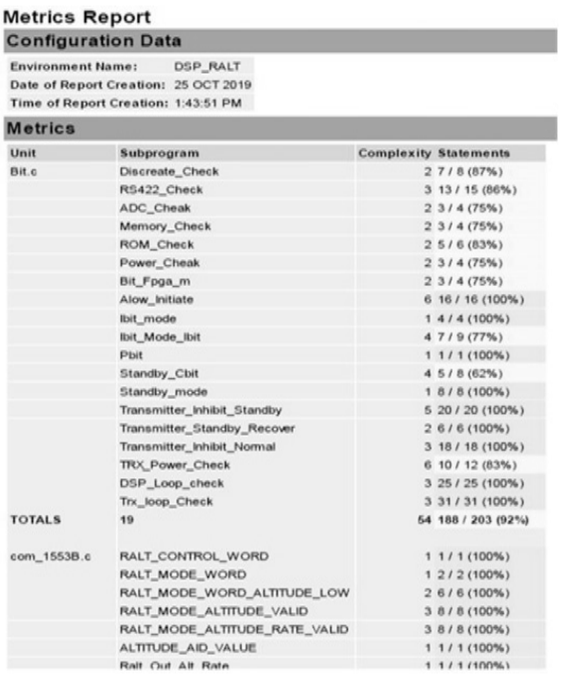


Fig. 3. Requirements-based Code Coverage Test Procedure



Metrics Report

Configuration Data

Environment Name: DSP_RALT
Date of Report Creation: 25 OCT 2019
Time of Report Creation: 1:43:51 PM

Metrics

Unit	Subprogram	Complexity	Statements
Bit.c	Discreate_Check	2	7 / 8 (87%)
	RS422_Check	3	13 / 15 (86%)
	ADC_Cheak	2	3 / 4 (75%)
	Memory_Check	2	3 / 4 (75%)
	ROM_Check	2	5 / 6 (83%)
	Power_Cheak	2	3 / 4 (75%)
	Bit_Fpga_m	2	3 / 4 (75%)
	Alow_Initiate	6	16 / 16 (100%)
	Ibit_mode	1	4 / 4 (100%)
	Ibit_Mode_Ibit	4	7 / 9 (77%)
	Pbit	1	1 / 1 (100%)
	Standby_Cbit	4	5 / 8 (62%)
	Standby_mode	1	8 / 8 (100%)
	Transmitter_Inhibit_Standby	5	20 / 20 (100%)
	Transmitter_Standby_Recover	2	6 / 6 (100%)
	Transmitter_Inhibit_Normal	3	18 / 18 (100%)
	TRX_Power_Check	6	10 / 12 (83%)
	DSP_Loop_check	3	25 / 25 (100%)
	Trx_loop_Check	3	31 / 31 (100%)
TOTALS	19	54	188 / 203 (92%)
com_1553B.c	RALT_CONTROL_WORD	1	1 / 1 (100%)
	RALT_MODE_WORD	1	2 / 2 (100%)
	RALT_MODE_WORD_ALTITUDE_LOW	2	6 / 6 (100%)
	RALT_MODE_ALTITUDE_VALID	3	8 / 8 (100%)
	RALT_MODE_ALTITUDE_RATE_VALID	3	8 / 8 (100%)
	ALTITUDE_AID_VALUE	1	1 / 1 (100%)
	Ralt_Out_Alt_Rate	1	1 / 1 (100%)

Fig. 4. Example Code Coverage Results Using Test Tools

100% 미달 시 시험 실패로 요구사항이나 코드 혹은 시험절차 등을 보완해야 한다.

5. 분석 결과

본 5절에서는 실제 시험사례를 분석한 결과를 제시한다. 먼저 분석 결과에서 우수 사례의 예시를 보인다. 다음으로 분석 결과의 유형화와 Risk 분석을 진행한 결과에 대한 간략한 요약을 진행한다. 그리고 각각의 세부적인 내용을 제시한다.

5.1 요구사항 추적성 우수 사례

아래 사례는 추적관리의 우수 사례로 요구사항과 시험절차에 누락이 없고 요구사항의 조건, 범위 등 시험절차에 잘 반영되어 있다.

Table 4. Battery Discharge Control Function Requirement

Identifier	R-LRU-SFR-004
Requirement	[배터리 방전 제어 기능] LRU는 전압이 10VDC 미만일 경우, 15A를 초과할 경우, 20초 이내에 방전을 중단할 수 있어야 한다.

Table 4는 배터리 방전 제어 기능 요구사항에서 요구도의 조건이 명시되어 있고, 해당 조건에서 어떤 기능의 구체적인 수치를 활용하여 제시했다.

Table 5는 배터리 방전 제어 기능에 대한 시험입력자료를 작성한 표로, 요구사항에 대한 모든 자료를 사용해 2개의 Test Scenario를 구현했다.

Table 5. Test Input on STD in the Battery Discharge Control

Contents		Conditions
시험입력 명칭		배터리 방전
시험입력 목적, 설명		방전조건에 따른 방전여부 확인
시험입력 사용방법		입력 전압
시험 입력 값 실제 자료 여부		모의값
시험입력 순서		DC Power on, 입력전압 10V 미만 설정
입력 자료 통제 방식	자료종류 및 최소/최대 적정자료 수	자료 종류: 방전값 항목: 10V 미만 방전 / Input값/상태: 10V 미만 값 방전전류 15A 초과시 20초 이내 방전 중단 / 15A 초과 값 최소/적정 자료 수: 2
	과부하, 최악의 자료 종류/값	해당사항 없음
	비규칙적인 입력 자료 종류/값	해당사항 없음
	재시험 기준	방전 조건 시 방전되지 않는 경우

Table 6. Expected Results on STD in the Battery Discharge Control

Contents	Expected Results			
정상적인 자료 입력 시	구분	발생 Case	예상값 / 예상 상태	시험절차 번호
	10V 미만 방전 여부	10V 미만	방전	5
		10V 이상	방전	6
	방전 중단 여부	15A 이하	방전	7
		15A 초과	20초 이내 방전 중단	8
과부하, 최악의 자료 입력시	해당사항 없음			
비규칙적인 자료 입력시	해당사항 없음			
기타 예상결과	해당사항 없음			

Table 6은 배터리 방전 제어 기능에 대한 시험 예상결과를 작성한 표이다. Table 5의 시험입력자료에서 구체화된 Test Scenario를 바탕으로 2개씩 총 4개의 Test Case로 식별했으면 각 Case별 시험 절차는 아래 〈배터리 방전 제어 기능 시험절차〉에서 누락 없이 식별했다.

〈배터리 방전 제어 기능 시험절차〉

1. 차단기1 ON한다.
2. LRU 시험장비에서 ON 스위치 켠다.
3. 차단기2를 ON하고 시험장비와 배터리가 연결되는 케이블에 전류측정기를 연결한다.
4. 배터리 충전기 시험장비에 충전기 스위치가 OFF 되었는지 확인한다. 전류 부하기로 부하를 인가한다.
5. 입력 전압이 10V 미만이 되었을 때 방전됨을 확인한다.

6. 입력 전압이 10V 이상이 되었을 때도 충전기 스위치가 OFF이면 방전되는지 확인한다.
7. 부하 15A 이하가 인가되었을 때 방전 조건여부를 확인하기 위해 LRU 방전 조건에 15A 이하의 부하를 인가하여 방전을 확인한다.
8. 부하 15A 초과가 인가되었을 때 방전 조건여부를 확인하기 위해 LRU 방전 조건에 15A 초과의 부하를 인가해 방전을 확인한다.

배터리 방전 제어 기능 사례에서 확인할 수 있듯이 요구사항에서 실험절차에 이르기까지 요구사항 및 시험절차 간 모두 추적가능 하도록 기술됨을 분석했다.

5.2 분석 결과의 유형화와 Risk 분석

총 7종 품목의 추적성을 유형과 Risk로 분석한 결과 Table 7과 같이 분류할 수 있었다.

Table 7. Classification of Software Traceability Analysis Results

Categories / Risk	Missing requirements	Lack of connection between Requirements -Test procedures	Missing test procedure	Sum
High	1	-	-	1
Middle	4	6	3	13
Low	1	-	3	4
합계	6	6	6	18

분석결과 Table 7과 같이 총 18건의 실패 사례를 찾을 수 있었으며, 요구사항 누락, 요구사항-시험절차 연계성 부족, 시험절차 누락과 같이 총 3개의 유형으로 분류할 수 있었다. 실패 사례는 각 유형별 6개씩 있었다. Risk를 기준으로 분석했을 때는 요구사항 누락으로 인한 요구사항 미검증에 해당하는 High가 1개, 요구사항에서 식별된 기능 중 일부 미검증에 해당하는 Middle이 13개, 실제 설계에는 반영되었지만 문서상 누락에 해당하는 Low가 4개였으며, Middle Risk에 해당하는 실패의 수가 가장 많았다.

5.3 요구사항 누락과 Risk 분석

요구사항 누락은 분류한 3가지 유형 중 가장 심각한 유형이다. 고객의 요구가 제대로 반영되지 않았을 뿐만 아니라 제품의 기능/성능에도 영향을 준다. 요구사항 누락의 경우는 아래와 같이 3가지로 다시 분류할 수 있었다.

1) 고객요구사항-시스템요구사항-상위요구사항-하위요구사항을 상세화하는 과정에서 누락되는 경우(1건)(High Risk)

분석 결과, Fig. 5와 같이 시스템 요구사항에서 소프트웨어 요구사항으로 구체화되는 과정에서 누락된 경우이다. 상위요구사항에서 요구사항이 누락되면 설계 및 구현이 이루어 지지않고 시험절차 또한 작성되지 않는다. 이는 최종 제품에 고객의 요구사항이 반영되지 않은 것을 의미하기 때문에 가장 위험한 경우라고 판단할 수 있다.

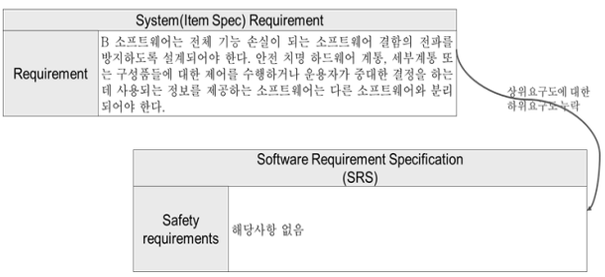


Fig. 5 Missing Lower Requirement for Higher Requirement

2) 특정기능에 대해 코드는 구현되었으나 소프트웨어 요구사항 명세서에 그 내용을 작성하지 않은 경우(1건) (Low Risk)

분석 결과, 소프트웨어 신뢰성시험 자동화도구에서 Fig. 6과 같은 요구사항 기반의 코드 실행률 결과를 보였다. 녹색은 실행된 코드를 나타내고, 빨간색은 실행되지 않은 코드를 나타낸다. 이는 문서상에 요구사항이 작성되어 있지 않아 실행되지 않은 코드였다. 해당 미실행 코드는 필요한 기능으로 소프트웨어 요구사항 명세서에 추가하여 STD를 보완해 목표값인 실행률 100%를 만족할 수 있었다.

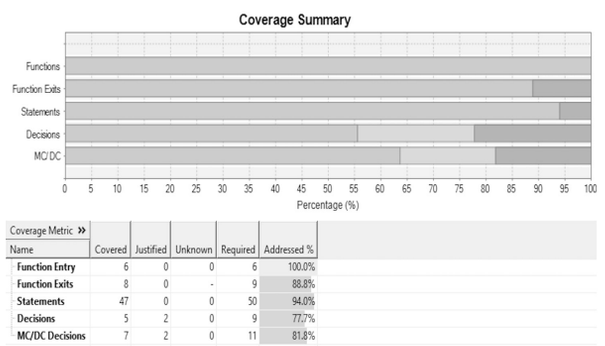


Fig. 6 Requirements-based Code Execution Rate Results

3) 요구사항에 특정 조건이나 범위가 전해지지 않아, 사용자 요구조건을 충족하지 못하는 경우(4건) (Middle Risk)

분석 결과, Fig. 7-Fig. 10과 같이 요구사항이 구체적으로 기술되지 않았었다. 이처럼 구체화되지 못한 요구사항은 100% 검증되지 않았기 때문에 최종 제품의 기능/성능에 대한 품질을 기대하기 어렵다.

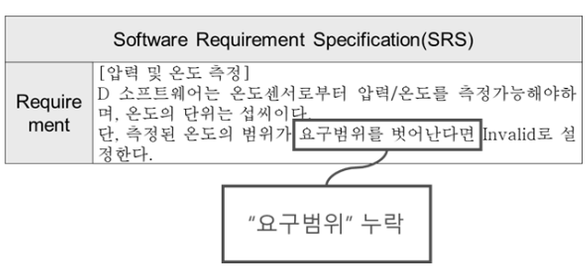


Fig. 7. Requirements Specification Error (1)

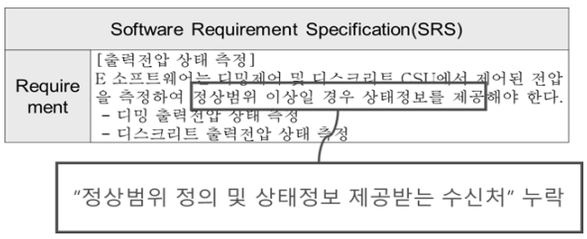


Fig. 8. Requirements Specification Error (2)

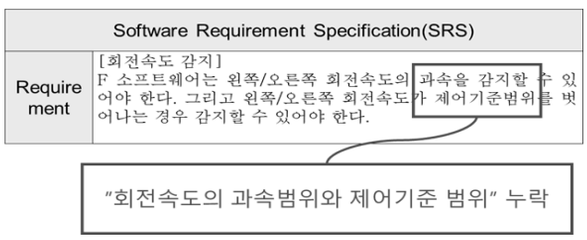


Fig. 9. Requirements Specification Error (3)

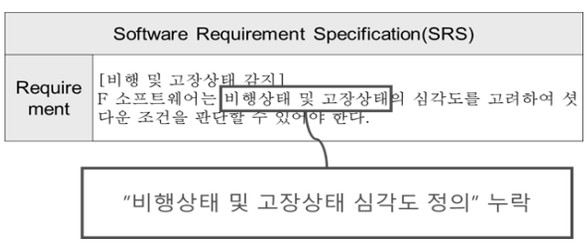


Fig. 10. Requirements Specification Error (4)

5.4 요구사항-시험절차 연계성 부족과 Risk 분석

요구사항-시험절차 연계성 부족은 하위 요구사항에서 구체화된 조건, 범위, 수치 등이 시험절차와 불일치하는 경우로, 요구사항에 대해 100% 검증되지 않는다.

1) Fig. 11은 요구사항에서 정의하고 있는 VS모드에서 1sec 이상 초과하는 경우를 시험절차에 변영하기 않았고, End 모드에서 3.2A미만을 3.2A이하로 시험절차에 작성해 요구사항과 시험절차 간 범위 불일치(Middle Risk)

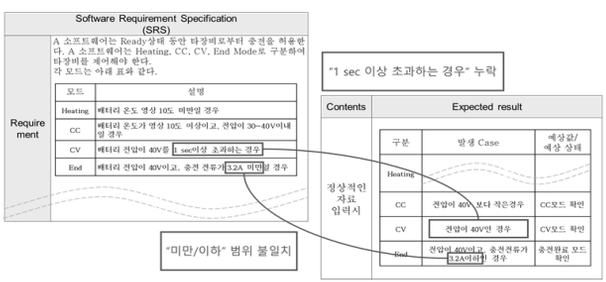


Fig. 11. Lack of Connection between Requirements and Test Procedures: Omission of Conditions and Inconsistent Scope

2) Fig. 12는 요구사항에서 정의하고 있는 CBIT 진단 빈도 5초 이내 조건이 시험절차와 불일치 (Middle Risk)

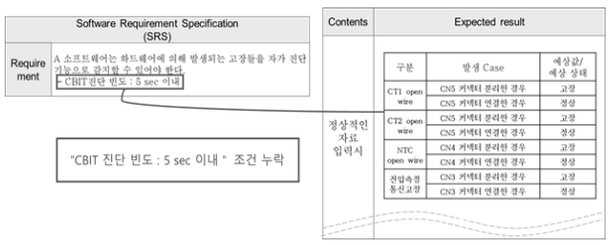


Fig. 12. Lack of Connection between Requirements and Test Procedures: Omission of Conditions

3) Fig. 13은 요구사항에서 기술된 내용 중 IBIT 20초 이내 수행 관련 시험절차가 기술되지 않은 경우로, IBIT 수행 여부만 확인하는 시험절차 (Middle Risk)

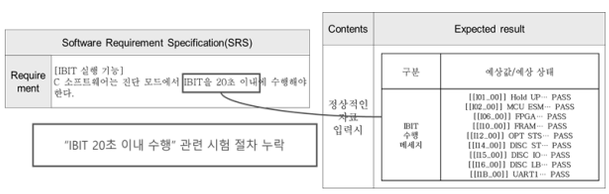


Fig. 13. Lack of Connection between Requirements and Test Procedures: Some of the Test Procedures are Omitted for Requirements (1)

4) Fig. 14는 요구사항에서 정의하고 있는 PBIT가 완료되는 시간 500ms에 대한 완료여부를 확인할 수 있는 시험절차가 작성되어 있지 않은 경우 (Middle Risk)

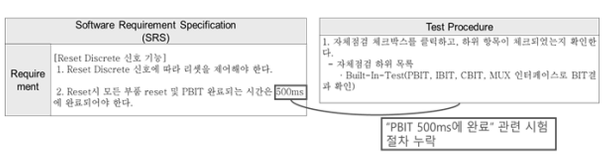


Fig. 14. Lack of Connection between Requirements and Test Procedures: Some of the Test Procedures are Omitted for Requirements (2)

5) Fig. 15는 요구사항에서 정의하고 있는 XX 전압을 공급받아 구성품이 정상 동작을 하는지 확인하는 시험절차가 작성되어 있지 않은 경우 (Middle Risk)

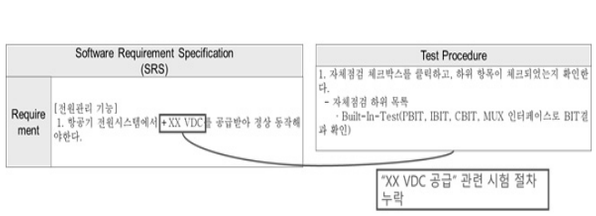


Fig. 15. Lack of Connection between Requirements and Test Procedures: Some of the Test Procedures are Omitted for Requirements (3)

6) Fig. 16은 요구사항에서 정의하고 있는 Switching 100 ms에 관한 시험절차가 작성되어 있지 않은 경우 (Middle Risk)

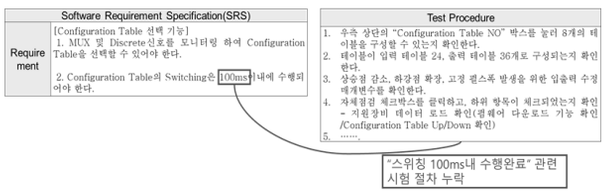


Fig. 16. Lack of Connection between Requirements and Test Procedures: Some of the test Procedures are Omitted for Requirements (4)

5.5 시험절차 누락과 Risk 분석

시험절차 누락은 요구사항에 기술된 내용 중 Test Case 일부가 시험절차에 기술되지 않은 경우로 요구사항의 모든 내용을 검증하기에 불충분하다.

1) Fig. 17은 요구사항에 기술된 내용 중 Fault가 발생하는 경우는 시험절차에 기술되었지만, Fault 미 발생의 경우 현재 상태를 유지하는지는 시험절차에서 누락한 경우 (Middle Risk)

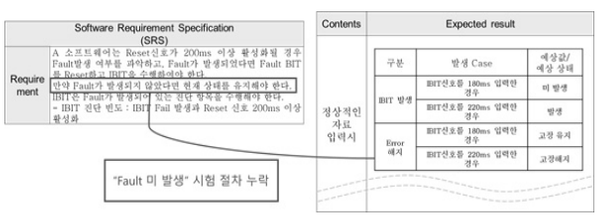


Fig. 17. Omission of Test Procedure for Requirements (1)

2) Fig. 18은 요구사항에 기술된 내용 중 대기모드에서 특정 신호를 송신하지 않는다는 것을 시험절차에서 누락되었으며, Test Case별 시험결과 예상값을 작성하지 않았지만, SW와 큰 관련이 없는 경우 (Low Risk)

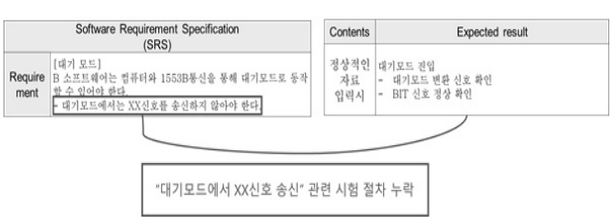


Fig. 18. Omission of Test Procedure for Requirements (2)

3) Fig. 19는 요구사항에 기술된 내용 중 고도측정 및 고도 변화율 측정관련 시험절차가 누락되었으나, SW와 큰 관련이 없는 경우 (Low Risk)

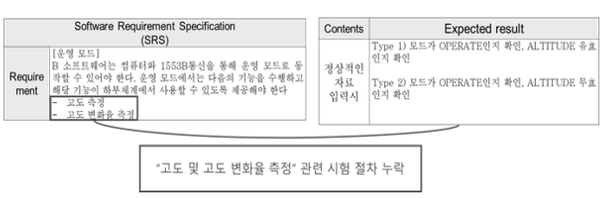


Fig. 19. Omission of Test Procedure for Requirements (3)

4) Fig. 20은 하나의 요구사항을 여러 가지 Test Case로 분류해 잘 작성했지만, WOW off시 PBIT수행여부 확인 시험절차가 누락된 경우 (Middle Risk)

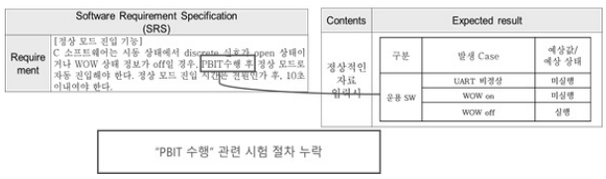


Fig. 20. Omission of Test Procedure for Requirements (4)

5) Fig. 21은 요구사항에 기술된 내용 중 IBIT 실패 관련 시험절차가 누락된 경우 (Middle Risk)

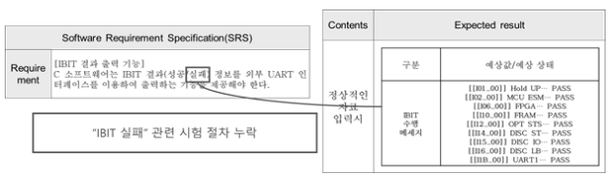


Fig. 21 Omission of Test Procedure for Requirements (5)

6) Fig. 22는 요구사항에 기술된 내용 중 Impedance 측정관련 시험절차가 누락되었지만, SW와 관련이 없는 경우 (Low Risk)

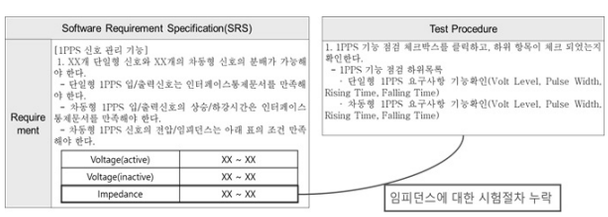


Fig. 22. Omission of Test Procedure for Requirements (6)

6. 결 론

본 연구는 항공기 탑재 소프트웨어의 실제 시험 사례를 대상으로 추적성 관점에서의 분석 절차를 제시하고, 추적성 실패 사례들을 유형화 하였다. 분석 결과, 요구사항 누락, 요구사항-시험절차 연계성 부족, 시험절차 누락 3가지 유형으로 분류할 수 있었다. 각 유형별 결과는 각 6건씩으로 어느 한곳에 치우치지 않았다. 즉, 요구사항 분석-설계-구현-시험 전체 개발프로세스 전반에 거쳐 추적관리 실패가 발생한 것을 확인하였다.

향후 연구과제로는 더 다양하고 많은 항공기 소프트웨어 개발사례 분석을 통해 오류분석 결과를 일반화할 수 있는 연구가 필요하다. 본 연구에서는 7가지 사례를 가지고 3가지 오류분석 유형을 제안하였다. 그러나 개발하는 소프트웨어의 특성이 다양하기 때문에 더 많은 유형이 존재할 수 있다. 시험사례가 다양하고 많을수록 Lessons-Learned(경험적 자산)이 쌓이고 또 그것을 활용하여 소프트웨어 개발시 참고한다면 소프트웨어 개발수준을 한층 더 올리는데 기여할 수 있다.

References

[1] S. N. Lee, "A Proposal for Creating Blue Ocean on Weapon System Software," *Defense & Technology*, Vol.433, pp.76-87, 2015.

[2] F. A. C. Pinheiro, "Requirements Traceability. In: do Prado Leite J.C.S., Doorn J.H. (eds) Perspectives on Software Requirements," *The Springer International Series in Engineering and Computer Science*, Vol.753, Springer, Boston, MA, 213004.

[3] M. Elizabeth and C. Hull, "Introduction to SW Verification and Validation, Requirements Engineering" 2005.

[4] J. K. Lee, H. K. Cho, and I. Y. Ko. "A Method for Requirements Traceability for Reuse of Artifacts using Requirements- Ontology-based Semantic Tagging," *Journal of KISS: Software and Applications*, Vol.35, No.6, pp.357-365, 2008.

[5] G. Antoniol, G. Canfora, G. Casazza, De Lucia, A. and E. Merlo, "Recovering Traceability Links Between Code and Documentation," *Software Engineering, IEEE Transactions on*, Vol.28, No.10, pp.970-983, 2002.

[6] B. G. Lee, M. S. Hwang, Y. B. Lee, H. J. Lee, J. M. Baik, and C. K. Lee, "Design and Development of a Standard Guidance for Software Requirement Specification," *Journal of KISS : Software and Applications*, Vol.36, No.7, pp.531-538, 2009.

[7] D. S. Kim, "A Suggestion on a Better Template for Requirements Traceability Matrix of a Requirements Specification," *Journal of the Korea Society of Systems Engineering*, Vol.12, No.1, pp.1-5, 2016.

[8] J. Y. Kim, S. Y. Rhew, and M. S. Hwang. "A Study of Requirement Change Management and Traceability Effect Using Traceability Table," *Journal of KIPS D*, Vol.17, No.4, pp.271-282, 2010.

[9] M. H. Yim and H. S. Kim, "Case of Requirement Traceability Management Using Computer-Aided Systems Engineering Tool In Light Rail Transit Project," *Proc. of the KSR Conference*, pp.768-773, 2016.

[10] L. Murray, A. Griths, P. Lindsay, and P. Strooper, "Requirements Traceability for Embedded Software - an Industry Experience Report," *Proceedings of the 6th IASTED International Conference on Software Engineering and Applications*, pp.374-068, SEA 2002.

[11] A. Qusef, "Test-to-code traceability: Why and how?," *IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, Amman, pp.1-8, doi: 10.1109/AEECT.2013.6716450, 2013.

[12] J. H. Jang, J. S. Jang, Y. S. Kang, and B. D. Shin. "The Case Study on Software Dynamic Testing Process Using Static Analysis in Compliance with DO-178C(Airborne Software Certification)," *Proc. of the KSASS Spring Conference*, pp.842-844, 2016.

[13] J. H. Jang, J. S. Jang, and Y. S. Kang, "The Case Study on Requirement-based Software Testing Process and achievement of Test Coverage in compliance with DO-178C(Airborne Software Certification)," *Proc. of the KSASS Fall Conference*, pp.1184-1185, 2016.

[14] Weapon System Software Development and Management Manual No.2017-8: Appendix 7, 5-7.

김 성 섭

https://orcid.org/0000-0001-8505-7687
e-mail : sungb429@naver.com
2013년 연세대학교 신소재공학부(학사)
2020년 경상대학교 항공우주공학(석사)
2015년 ~ 현 재 한국항공우주산업(주) KFX임무SW팀 선임연구원
관심분야 : 소프트웨어 공학, 요구도 추적

조 희 태

https://orcid.org/0000-0002-1178-7852
e-mail : cht3205@gmail.com
2017년 경상대학교 항공우주 및 소프트웨어공학전공(학사)
2020년 경상대학교 정보과학과(석사)
2020년 ~ 현 재 경상대학교 AI융합공학과 박사과정
관심분야 : 소프트웨어 공학, 기계학습

이 선 아

https://orcid.org/0000-0002-2004-2924
e-mail : saleese@gnu.ac.kr
1997년 이화여자대학교 전산학(학사)
1999년 이화여자대학교 전산학(석사)
2005년 카네기멜론대학교 소프트웨어공학(석사)
2013년 KAIST 전산학(박사)
1999년 ~ 2006년 삼성전자 선임/책임연구원
2013년 ~ 2015년 KAIST 연구조교수
2016년 ~ 현 재 경상대학교 항공우주 및 소프트웨어공학전공 / AI융합공학과 부교수, 항공기부품기술연구소 멤버
관심분야 : Software Architecture & Software Repository Mining & Recommendation System & Software Evolution