Journal of KOSSE. (2018. 12) Vol. 14, No. 2 pp. 83-88
DOI: https://doi.org/10.14248/JKOSSE.2018.14.2.083
www.kosse.or.kr

ISSN (print) : 1738-480X
ISSN (online) : 2288-3592

시스템엔지니어링 방법론을 적용한 소프트웨어 테스트 케이스 개발에 관한 연구

살림셀리* 신중욱 김진일

고등기술연구원 플랜트엔지니어링 본부

A Study on the Software Test Case Development using Systems Engineering Methodology

Shelly Salim*, Junguk Shin, Jinil Kim

Institute for Advanced Engineering, Plant Engineering Department

Abstract : Software has become an integral part of almost any system, triggered by the ever-growing demand for automation and artificial intelligent throughout engineering domains. The complexities of software-centric systems are also increasing, which make software test efforts become essential in software development projects. In this study, we applied systems engineering methodology in generating software test cases. We found out the similarities between requirements analysis and traceability concept of systems engineering and test specification contents of software test. In terms of acceptance test, software test cases could be considered as validation requirements. We also suggested a method to determine test order using a SysML modeling tool.

Key Words : Systems engineering methodology, software system, software test, software test case

Received: November 15, 2018 / **Revised**: November 30, 2018 / **Accepted**: December 28, 2018

* 교신저자 : Shelly Salim, shellysalim22@gmail.com

1. 서 론

최근 대부분 시스템 개발 과정에 있어서 소프트웨어는 빠질 수 없는 중요한 요소가 된다. 그 이유는 시스템에 자동화 또는 인공지능 기능 요구에 많아지고 이를 구현하기 위하여 통합적인 소프트웨어가 필요하기 때문이다. 이와 함께 소프트웨어의 복잡도 높아지고 있으므로 소프트웨어 개발 노력뿐만 아니라 소프트웨어 테스트(Software Test)의 중요성도 높아지고 있다. 소프트웨어 테스트는 이해관계자들이 소프트웨어에서 요구하는 기능들을 확인 또는 검증을 하는 활동이고, 주요 활동으로는 테스트 케이스를 작성하여 그에 맞게 테스트 실행하고 결과들을 기록하는 활동이다. 본 연구에서는 시스템엔지니어링 방법론을 활용하여 보다 더 효과적인 소프트웨어 테스트를 진행할 수 있도록 가이드라인을 제시하였으며 테스트 케이스 개발 사례도 소개하였다. 본 논문에 제안하는 방법론의 기대효과 다음과 같다.

1) 테스트케이스와 소프트웨어 요구사항 추적성 유지된다.
2) 테스트케이스 개발 시간 축소된다.
3) 테스트 순서 최적화하여 테스트케이스 수행 기간 축소된다.

2. 소프트웨어 테스트

소프트웨어 테스트는 소프트웨어 개발에 있어 빠질 수 없는 매우 중요한 활동이다. 특히 개발하려는 소프트웨어의 기능이 많거나 개발자가 많을 경우 소프트웨어 테스트를 하지 않으면 오류들이 생길 확률이 높아진다. 소프트웨어 테스트 활동은 소프트웨어 생명주기 모델(life cycle model)에 대한 개념이 결정해지기 전에 먼저 정의된 만큼 중요한 활동으로 1954년부터 인식되었으며 [1] 소프트웨어 테스트 관련 표준들을 꾸준히 개발해 왔다. 최근에 소프트웨어 테스트 관련 표준으로는 ISO/IEC/IEEE

[Figure 1] Test Plan Content Comparison between IEEE 829 and 29119-3

29119 (이하 29119) 시리즈가 있으며 총 5 파트(Part)들로 Table 1과 같이 제공된다[2, 3, 4, 5, 6].

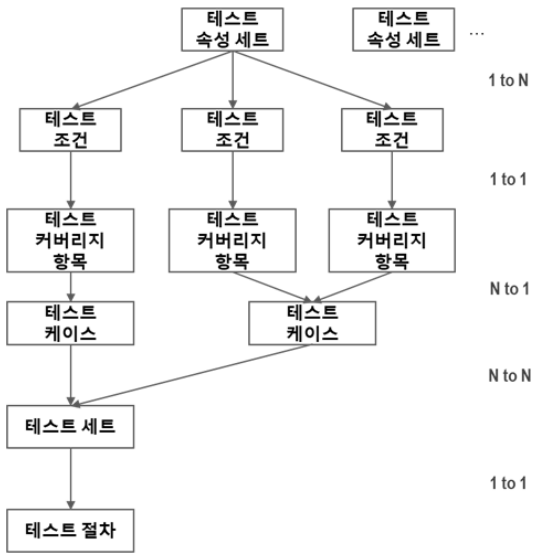
효과적인 소프트웨어 테스트 수행하기 위하여 테스트 계획서(Test Plan, TP)를 준비해야 한다. TP와 관련 있는 표준들은 29119-3, IEEE 829[7] 및 IEEE 1012[8]이다. IEEE 829는 Master Plan과 Level Test Plan으로 나뉘어 있었으나 29119-3으로 대체된다. IEEE 1012의 경우는 전반적인 시스템, 소프트웨어 및 하드웨어의 검증 및 확인에 대한 표준이므로 이번 연구에서는 29119-3이 TP의 주요 참고 표준이 된다. IEEE 829와 29119-3의 기술한 TP의 내용은 Figure 1과 같다.

소프트웨어 테스트 활동(Activity)들은 TP의 내용에 따라하여 테스트 케이스(Test Case)를 수행하는 것이다. 29119-3에서는 테스트 케이스가 테스트 규격서(Test Specification, TS)에 기술된다. TS의 내용 및 관계는 Figure 2에서 나타내었다.

TS에 나오는 내용은 소프트웨어 테스트에서 흔히 사용되는 용어들이다. 테스트 속성 세트(Test Feature Set)는 테스트 항목에 대한 요구 사항을

<Table 1> Content Summary of Software Test Standard ISO/IEC/IEEE 29119 Series

파트	파트 명	출판	내용 요약
1	Concepts and definitions (개념 및 정의)	2013	본 파트에서는 소프트웨어 테스트의 기본 개념을 설명한다. 조직 측면 및 프로젝트 측면에 소프트웨어 테스트의 필요성, 소프트웨어의 수명주기 관점으로 소프트웨어 테스트 활동, 위험기반 테스트(Risk-based testing)의 이해 및 방법 개요, 소프트웨어 테스트 관례 등을 설명한다.
2	Test processes (테스트 프로세스)	2013	본 파트에서는 소프트웨어 테스트의 프로세스를 정의하며 시스템 엔지니어링 분야에 알리는 프로세스의 개념과 비슷하나 소프트웨어 테스트는 Multi-layer로 정의된다. 총 3개의 프로세스를 제공하며, 이들은 조직적 테스트 프로세스 (Organizational Test Process), 테스트 관리 프로세스 (Test Management Process), 동적 테스트 프로세스(Dynamic Test Process)이다.
3	Test documentation (테스트 문서화)	2013	본 파트에서는 소프트웨어 테스트 수행하는 동안에 관련 있는 문서의 종류 및 그의 권장 내용을 설명한다. 파트 2에서 기술하는 3개의 프로세스를 기반으로 프로세스 별 문서화에 대한 설명한다. 몇 개의 주요 문서에 대한 자세한 가이드라인과 사례도 제공한다.
4	Test techniques (테스트 기법)	2015	본 파트에서는 총 3 종류의 테스트 설계 기법을 설명하며 이는 명세서/규격서 기반 테스트 설계 기법 (Specification-based test design techniques), 구조 기반 테스트 설계 기법 (Structure-based test design techniques), 경험 기반 테스트 설계 기법(Experience-based test design techniques)이다. 테스트 커버리지 관련 측정 방법과 기준도 제공한다.
5	Keyword-Driven testing (키워드 주도 테스팅)	2016	본 파트에서는 소프트웨어 테스트 방법론 중 하나인 키워드 주도 테스팅에 대한 개념, 응용, 프레임워크 및 데이터 교환 등을 설명한다. 키워드 주도 테스팅에 대한 별도의 표준을 만든 배경은 소프트웨어 테스팅 자동화에 일반적으로 적용하는 방법론이기 때문이다. 키워드 주도 테스팅은 자동 소프트웨어 테스트의 테스트 케이스 규격(Specification)또는 프레임워크 개발 시 활용된다.



[Figure 2] 29119-3 Test Specification Content

파악하기 위해 테스트 기준을 정의하는 것이며 테스트 조건(Test Condition)은 테스트 계획서에 정의된 종료 기준에 기초하여, 각 기능에 대한 테스트 조건을 결정하는 것이다. 테스트 커버리지 항목(Test Coverage Item)은 테스트 조건 중 테스트 수행을 통해 커버될 항목을 테스트 설계 기법을 적용하여 도출하는 것이며 테스트 케이스(Test Case)는 사전 조건, 입력, (필요 시) 선택된 테스트 커버리지 항목을 계산하기 위한 액션, 기대 결과를 결정하여 하나 이상의 테스트케이스를 도출하는 것이다. 테스트 세트(Test Set)는 실행 제약 사항을 기준으로 테스트케이스를 분류하여 하나 이상의 테스트케이스 세트로 구성하는 것이며 테스트 절차(Test Procedures)는 사전 조건, 사후 조건, 기타 테스트 요구사항이 기술된 종속물에 따라 테스트 세트 내의

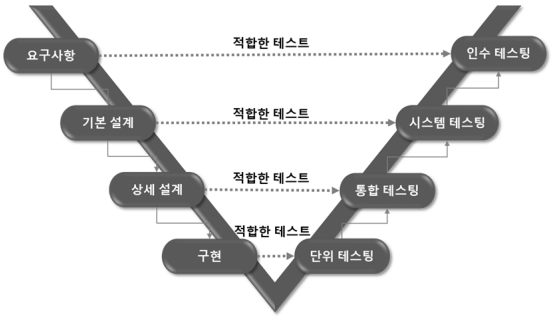
테스트케이스 순서에 따라 테스트 절차를 도출하는 것이다.

앞서 소프트웨어 테스트 관련 활동은, 소프트웨어 아키텍처와 마찬가지로, 성숙된 개념으로 판단할 수 있는 반면에, 소프트웨어 테스트의 아키텍처는 아직 개발 중인 개념이다 [9]. 따라서 널리 알리는 시스템엔지니어링 프로세스의 논리를 활용하여 소프트웨어 테스트 아키텍처를 구축할 필요가 있다.

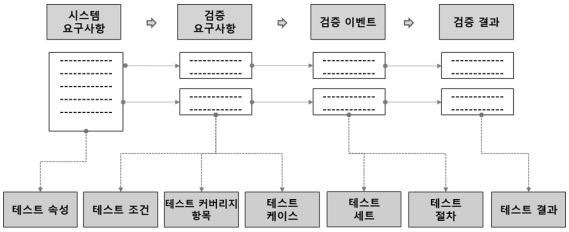
3. 소프트웨어 테스트 케이스 개발 결과

소프트웨어 테스트에서 중요한 활동은 테스트 케이스 개발하고 수행하는 것이다. 이에 시스템 엔지니어링 방법론을 적용하여 개발하였다. 먼저 소프트웨어 테스트는 Figure 3과 같이 V-model 에서의 소프트웨어 개발 단계별로 각각 테스트 레벨(Level)과 매핑 할 수 있다. 또한 소프트웨어 테스트는 시스템 검증(Verification)과 확인(Validation)으로 나눌 수 있는데 본 논문에서는 소프트웨어 테스트 케이스 개발 대상을 인수 테스트로 선정하여 검증(Verification) 보다 확인(Validation)의 개념으로 테스트 케이스를 개발 하였다. 이와 함께 시스템 엔지니어링에서의 요구사항 추적성을 소프트웨어 테스트에 적용하였다. 요구사항은 이해 관계자 니즈(needs)로부터 이해관계자 요구사항을 도출하고 시스템 및 하부시스템의 요구사항을 도출하게 되며 도출된 요구사항들은 서로 연결되어 추적성이 확보되어야 한다. 따라서 시스템 요구사항으로부터 검증 요구사항을 도출하였으며 검증 요구사항과 소프트웨어 테스트 규격서의 내용을 연결하여 추적성을 확보하였다. Figure 4는 요구사항 연결에 따른 추적성을 나타낸 것이다.

본 연구에서는 시스템 검증 요구사항을 소프트웨어 테스트 케이스로 구현하였다. 소프트웨어 테스트 케이스에서 기술하고 있는 내용들의 기준은 검증요구사항의 포맷이 되며 검증요구사항/테스트케이스와 시스템 요구사항을 서로 연결하여 추적성이 유지되도록 하였다. Figure 5와 같으며 다음과 같은 내용을 작성한다.



[Figure 3] Mapping of Software Test Level and V-model



[Figure 4] Software Test Specification's Content and Validation-related Outputs Traceability

관리번호 : TC.2.12.6					
시스템 요구사항	시스템은 사용자가 원하는 각도만큼 객체를 회전할 수 있도록 해야 한다.				
테스트 사전조건	3D CAD 모델을 화면에 띄운 상태로 회전시키고자 하는 객체는 선택되어 있음.				
테스트 방법	(1) '객체회전' 메뉴 선택 (2) '객체회전' 메뉴에서 회전 기준축(X축,Y축,Z축) 선택과 회전각도 입력				
테스트 수행자	홍길동	테스트 일자	2019년 00월 00일		
단계	입력	예상 출력	실행결과	조치사항	조치 후 시험결과
1	- 회전 기준축: X - 회전각도: +1°	X축 기준으로 +1° 회전			
2	- 회전 기준축: Y - 회전각도: -13°	Y축 기준으로 -13°회전			
3	- 회전 기준축: Z - 회전각도: +180°	Z축 기준으로 +180°회전			

[Figure 5] Validation Requirements/Test Case Format

- 관리번호: 고유 ID
- 시스템 요구사항: 해당 검증요구사항/테스트케이스 추적하는 시스템 요구사항
- 테스트 사전조건: 테스트 실행할 수 있도록 존재해야 하는 조건
- 테스트 방법: 테스트 케이스 수행하는 방법
- 테스트 수행자 : 테스트를 수행하는 자
- 테스트 일자 : 테스트를 수행한 일자
- 테스트 단계: 테스트를 단계 별 수행하는 경우, 단계 별 입력값 및 예상 출력값을 제공
- 시행결과: 테스트 수행 결과 예상 출력값을 만족하는 경우는 합격으로 작성하고 아닌 경우는 불합격으로 작성
- 조치사항 및 조치 후 시험결과: 필요 시 작성

4. SysML 모델 이용한 테스트 순서 결정

테스트 케이스에는 테스트 사전조건이 존재하며 사전조건을 만족해야만 테스트를 수행할 수 있다. 다수의 테스트 케이스가 비슷하거나 동일한 사전조건들을 가지고 있을 수도 있다. 이러한 경우, 테스트 시간을 절약할 수 있게끔 테스트 순서를 별도 정하는 것이 효과적이다. 테스트 수행자가 수동으로 동일한 사전조건 찾고, 순서를 정하는 것보다 SysML을 이용한 모델링으로 간단한 시뮬레이션 결과를 통해 테스트 순서를 결정하면 큰 도움이 될 수 있다. SysML은 모델 기반 시스템 엔지니어링(Model Based Systems Engineering (MBSE)) 방법론 중 하나이며 시스템엔지니어링 프로세스 수행에 있어서 문서 기반 방법보다 더 효과적이다 [10].

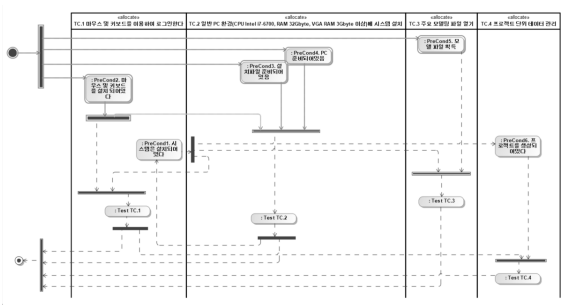
이를 위하여 MBSE 도구를 이용하여 테스트 케이스 수행의 우선순위를 정하는 시뮬레이션을 수행하였다. 먼저, 모든 시스템 요구사항이 테스트 케이스와 연결이 되었는지 확인한다. 각 시스템 요구사항들은 모두 확인(Validation)에 필요한 테스트 케이스를 수행해야 하며 하나의 요구사항이 여러 개의 테스트 케이스를 통해 확인(Validation)하는 경우도 있다. 각 요구사항과 개발된 테스트 케이스는 누락되는 경우가 없도록 Figure 6과 같이 매트릭스로 요구사항과 테스트 케이스와의 추적성을 확인하였다. 본 논문에서 4개의 간단한 요구사항 및 테스트 케이스 예시들을 다음과 같이 준비하였다.

- TC.1 마우스 및 키보드를 이용하여 로그인한다
- TC.2 일반 PC 환경(CPU Intel i7-6700, RAM 32Gbyte, VGA RAM 3Gbyte 이상)에 시스템 설치
- TC.3 주요 모델링 파일 열기
- TC.4 프로젝트 단위 데이터 관리

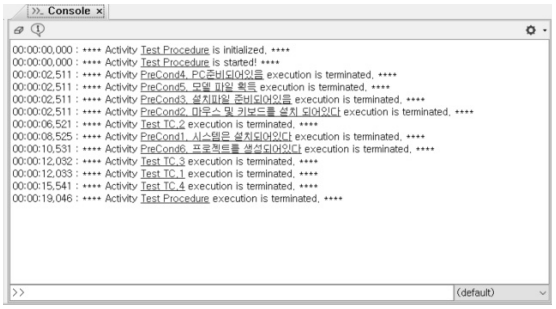
요구사항과 테스트 케이스간의 추적성이 확인되면 Activity Diagram을 이용하여 테스트 케이스 별로 테스트 활동(Activity)과 사전조건 활동을 만들며 활동들 간에 흐름을 모델링 한다 [11]. Figure 7은 4개의 테스트 케이스 예에 대한 테스트 활동 및 사전조건 활동을 Activity Diagram 으로 모델링하여 시뮬레이션 중인 모습을 나타낸 것이다. 모델링된 Activity Diagram을 시뮬레이션하면 테스트 케이스 중 어떤 케이스가 먼저 완료되었는지를 결과를 알 수 있다. 시뮬레이션 결과는 완료한 테스트 케이스 순서로 Figure 8에서 나타내었다. 시뮬레이션 결과 4개의 테스트 케이스를 고려하여 적합한 테스트 순서는: 'TC.2, TC.3, TC.1, TC.4'임을 알 수 있다. 미세한 수행 기간 차이로 TC들을 설계하였으니 모델링 및 시뮬레이션 결과에 영향을 낮췄다.

시뮬레이션 결과를 고려하여, 처음에 TC 설계와 달리 TC 순서를 알아냈다. 첫 TC 설계 시, 요구사

[Figure 6] System Requirements and Test Case Traceability



[Figure 7] Test Cases' Activity Diagram Simulation Snapshot



[Figure 8] Simulation Result Script

항 순서에 따라하였으며, TC.1, TC.2, TC.3, TC.4로 설계하였다. 그러나 TC 내용을 살펴보면, TC.2가 설치 관련 테스트이기 때문에, 논리적으로 TC.2는 보다 앞서 테스트해 하는 것을 파악하게 된다. 이러한 분석을 테스트 수행자가 TC 내용 일일이 보고 수동으로 순서를 정하는 것 보다 모델링하고 시뮬레이션하는 것이 더 효율적이다.

5. 결 론

시스템 엔지니어링 관점으로 소프트웨어 테스트 활동을 분석하였다. 특히 소프트웨어 테스트 케이스를 시스템엔지니어링 방법론을 적용하여 개발사례를 소개하였다. 시스템 엔지니어링 방법론을 활용하면 보다 요구사항 추적성을 유지하면서 테스트 케이스를 더욱 효과적으로 개발할 수 있다. 또한 모델기반 시스템 엔지니어링 도구를 사용하여 테스트 케이스 수행 순서를 효율적으로 결정하는 방법도 소개하였다. 향후 연구로 소프트웨어 테스트 아키텍처에 대한 보다 더 시스템적인 분석 진행할 예정이며 모델 기반 시스템 엔지니어링(Model Based Systems Engineering)방법론을 적용하여 테스트 케이스를 효율적으로 개발할 목표도 있다.

사 사

이 연구는 2018년도 산업통상자원부 및 산업기술평가관리원(KEIT) 연구비 지원에 의한 연구임(10080662, 효율적인 엔지니어링 프로젝트 협업 환경 구축을 위한 경량 3D 모델기반 디지털 협업 지원 시스템 개발)

References

1. Stuart Reid. The New Software Testing Standard. Achieving Systems Safety: Proceedings of the Twentieth Safety-Critical Systems Symposium, Bristol, UK, 7-9th February 2012.
2. ISO/IEC/IEEE 29119-1:2013, Software and systems engineering — Software testing — Part 1: Concepts and definitions.
3. ISO/IEC/IEEE 29119-2:2013, Software and systems engineering — Software testing — Part 2: Test processes.
4. ISO/IEC/IEEE 29119-3:2013, Software and systems engineering — Software testing — Part 3: Test documentation.
5. ISO/IEC/IEEE 29119-4:2015, Software and systems engineering — Software testing — Part 4: Test techniques.
6. ISO/IEC/IEEE 29119-5:2016, Software and systems engineering — Software testing — Part 5: Keyword-Driven Testing.
7. 829-2008 - IEEE Standard for Software and System Test Documentation.
8. IEEE 1012-2012 - IEEE Standard for System and Software Verification and Validation.
9. Yasuharu Nishi. Viewpoint-based Test Architecture Design. 2012 IEEE Sixth International Conference on Software Security and Reliability Companion.
10. 양환석, 장재덕, 정호, 최상욱, 이혜진, 이수용. 모델기반 시스템 엔지니어링(MBSE)을 적용한 요구사항개발 프로세스 연구. 시스템엔지니어링학술지, 제 14 권, 1호, pp. 51-56, 2017.
11. Tae Ho Yeo, Tae Ryong Kim, Chang Lak Kim. Development of reutilization system for Nuclear Power Plant Component using Object-Oriented Systems Engineering Method. 시스템엔지니어링학술지, 제 12 권, 2호, pp. 69-80, 2016.