리스크 기반 임베디드 소프트웨어 테스트 전략

권원일 (STA테스팅컨설팅)

Ⅰ. 서 론

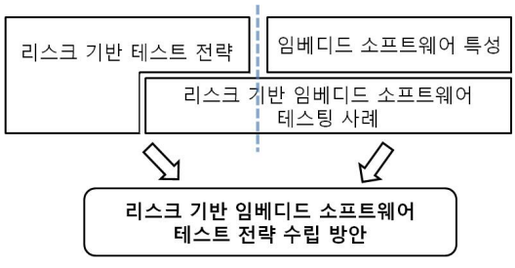
리스크 기반 테스트 전략은 리스크 분석 결과를 바탕으로 리스크 우선순위를 결정하고 이를 근거로 테스트 전략을 수립하는 것이다. 소프트웨어 테스팅을 전략적으로 수행하는데 필수적인 요소로 임베디드 소프트웨어 테스팅은 물론 테스팅 분야 전반에 걸쳐 사용되는 테스팅 접근법이다. 이는 ISO 소프트웨어 테스팅 프로세스 국제표준(ISO/IEC 29119), IEEE 테스트 문서화 표준(IEEE829), 사실상(de facto) 표준 테스트 프로세스 심사 모델인 TMMi(Test Maturity Model integration) 등의 핵심을 차지하고 있다. 리스크 기반 테스팅의 효과는 업계 사례를 통해 발표되고 있으며, 최근에는 리스크 기반 테스팅 전략에 따른 테스팅을 3년간 수행한 결과 출시 후 고객으로부터 피드백 받는 결함의 수가 1/10로 감소된 사례가 발표됐다.[7]

본 연구에서는 리스크 기반 테스팅 전략을 수립하는 방법 및 임베디드 소프트웨어 테스팅에서의 활용 사례를 통해 해당 전략을 임베디드 소프트웨어 테스팅에 활용하는 방안을 제시한다. 임베디드 소프트웨어가 하드웨어와의 연계성, 실시간성, 활용 가능한 연산처리 및 메모리 자원의 제약성 등 다양한 특징을 가지고 있어 이를 리스크 기반 테스트 전략 수립 시 리스크 요소 결정 및 전략 수립 시 테스트 설계 기법 선택 등에 반영해야 한다. 이와 같은 리스크 기반 테스팅을 임베디드 소프트웨어 테스팅 분야에 적용하는 방법을 소개하고자 한다.

Ⅱ. 관련 연구

1. 연구 방법

본 연구에서는 〈그림 1〉에서와 같이 리스크 기반 테스트 전략에 대해 살펴보고 임베디드 소프트웨어의 특성과 해당 전략의 활용 사례를 반영하여 리스크 기반 임베디드 소프트웨어 테스트 전략 수립 방안을 제안한다.



〈그림 1〉 리스크 기반 임베디드 소프트웨어 테스트 전략 연구 방법

2. 전략과 테스팅

전략의 정의는 경영 및 국방 분야를 포함하는 거의 모든 분야에서 여러 형태로 정의하고 있다. 이 중 전략에 대한 몇 가지 정의를 정리해 보면 아래와 같다.

- 경쟁 환경의 제약 하에서 목표 달성을 위한 리소스 사용 접근법 (The Handbook of Strategic Expertise)
- 목표를 향해 계획을 수립하는 기술 (Webster 사전)
- 조직이 의도하는 성취(Achievements)의 정의 (PMI Standard for Portfolio Management)
- 특정 산업에서 방어 가능한 위치를 선점하기 위한 공격적 또는 방어적인 액션 (Michael Porter)

테스트 전략에서 위의 전략 정의 중 어떤 것을 수용해도 되지만 여기서는 테스팅 프로젝트에 적합한 “경쟁 환경의 제약 하에서 목표 달성을 위해 리소스를 사용하는 접근법”

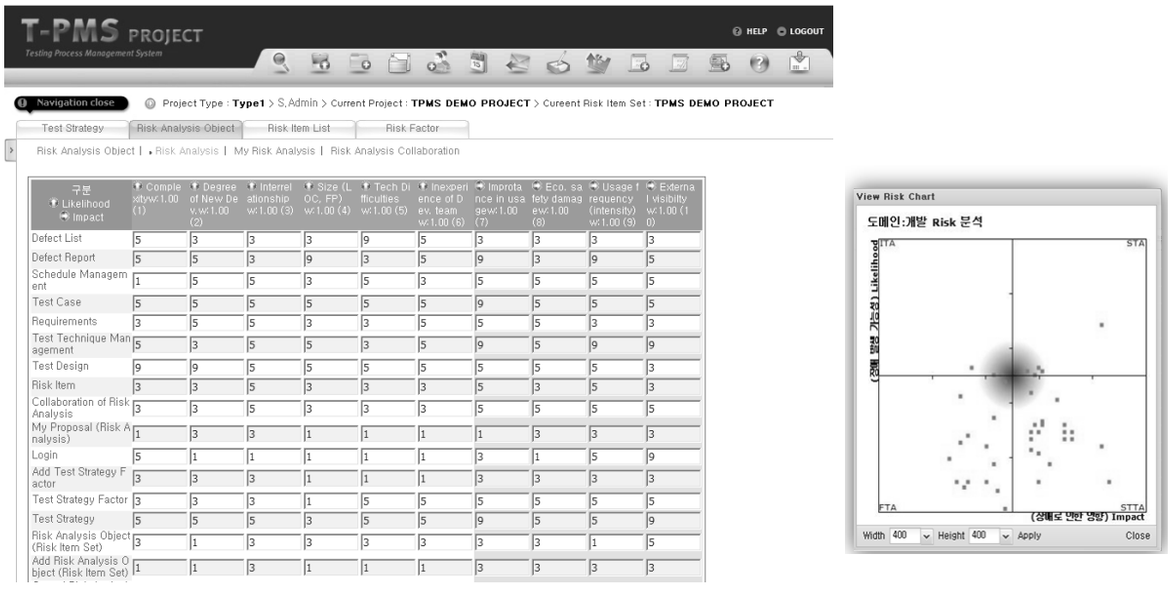
측면을 중점 고려한다.

테스팅 분야에서의 전략은 테스팅 분야의 특성상 리스크에 관심이 많아 테스팅 분야 종사자들이 리스크 기반 테스팅과 관련 전략에 대해 오랫동안 연구하고 업무에 활용해 왔다. 테스팅을 리스크를 줄이는 활동으로 보고 요구사항을 근간으로 리스크를 분석하여 리스크 우선순위에 따라 리스크 높은 부분은 보다 강도 높게 테스팅하는 것이 기본 개념이다. 이는 하드웨어 개발 중 안전분석 분야에서 사용해온 FMEA (Failure Mode and Effect Analysis)와 유사한 방식이다. 다른 시각에서 보면 리스크 관리 기법의 일부로 볼 수도 있다.

3. 리스크 기반 테스트 전략

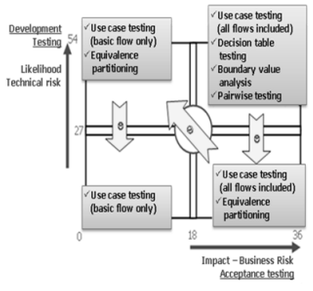
리스크 기반 테스팅 전략은 리스크 분석 결과를 바탕으로 리스크 우선순위를 결정하고 이를 근거로 전략을 수립하는 것이다. 〈그림 2〉에서 보는 바와 같이 요구사항 또는 관련 개발 산출물에서 리스크 아이템을 식별하고 이것을 장애 발생 가능성(Likelihood)과 장애가 발생했을 때의 비즈니스 영향도(Impact or Damage)라는 측면에서 리스크 요소를 세분화하여 리스크를 분석한다. 이를 수치화해 분석하고 나면 결과적으로 〈그림 2〉 하단의 리스크 분석 분포도를 그릴 수 있어 직관적으로 리스크의 높낮음을 알 수 있다.

이렇게 분석된 결과를 바탕으로 〈그림 3〉에서와 같이 전략 요소를 결정한 후 테스팅을 전체적으로 조율할 테스팅 전략을 수립한다. 위쪽 그림에서 테스팅 단계를 고려해 테스팅의 우선순위를 화살표로 표현했고, 리스크가 높은 경우 강도 높고 다양한 테스트 설계 기법을 사용해야 하고 리스크가 낮아짐에 따라 차등적으로 접근해 테스트 케이스를 도출함을 표현했다. 테스트 설계 기법 이외에도 일반적으로 테스팅을 수행하는데 리소스가 투입되는 대부분의 활동을 리스크 우선순



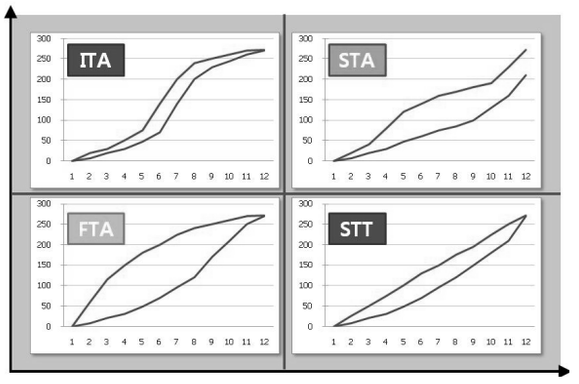
구분 Likelihood / Impact	Comple xity w:1.00 (1)	Degree of New Dev. w:1.00 (2)	Interrel ationship w:1.00 (3)	Size (LOC, FP) w:1.00 (4)	Tech Difficulties w:1.00 (5)	Inexperience of Dev. team w:1.00 (6)	Improtance in usage w:1.00 (7)	Eco. safety damage w:1.00 (8)	Usage frequency (intensity) w:1.00 (9)	External visibilty w:1.00 (10)
Defect List	5	3	3	3	9	5	3	3	3	3
Defect Report	5	5	3	9	3	5	9	3	9	5
Schedule Management	1	5	5	3	5	3	5	5	5	5
Test Case	5	5	5	5	5	5	9	5	5	5
Requirements	3	5	5	3	3	5	5	5	3	3
Test Technique Management	5	3	5	5	3	5	9	5	9	9
Test Design	9	9	5	5	5	5	5	5	5	3
Risk Item	3	3	5	3	3	3	5	3	5	3
Collaboration of Risk Analysis	3	3	5	3	3	3	5	5	5	5
My Proposal (Risk Analysis)	1	3	3	1	1	1	1	3	3	3
Login	5	1	1	1	1	1	3	1	5	9
Add Test Strategy Factor	3	3	3	1	1	1	3	3	3	3
Test Strategy Factor	3	3	3	1	5	5	5	5	5	5
Test Strategy	5	5	5	3	5	5	9	5	5	9
Risk Analysis Object (Risk Item Set)	3	1	3	3	3	3	3	3	1	5
Add Risk Analysis Object (Risk Item Set)	1	1	3	1	1	1	3	3	3	3

〈그림 2〉 테스팅 전략 수립을 위한 리스크 분석 예시



리스크 레벨	테스트 설계		완료 조건	리소스		테스트 실행	
	기법	설계 리뷰		인력 배치	역할	재테스팅	*리그레션 테스팅
STA	· *명세 분석 · 페어와이즈 + 경험적 조합 · 경계값 분석 · 탐색적 테스팅 · 비정상 TC 비율* : 정상 TC의 50%	· 테크니컬 리뷰 · 워크쓰루	· 가능한 TC 수행 · 심각도 중간 이상인 결함 없음 · Workaround 결함 3% 이하(실행된 TC수 대비) · 100% 요구사항 커버리지*	· PL급 개발자	· 테스트 계획서 작성 · TC 작성 · 테스트 수행 · 테스트 결과 검토 및 승인	2회 (1회는 교차 테스팅)	Full 리그레션 테스팅
				· 프로젝트 참여 현업 (초기 참여자)	· 테스트 수행		
ITA	· 명세 분석 · 페어와이즈 + 경험적 조합 · 경계값 분석 · 탐색적 테스팅 · 비정상 TC 비율* : 정상 TC의 30%	· 테크니컬 리뷰	· 가능한 TC 수행 · 심각도 중간 이상인 결함 없음 · Workaround 결함 10% 이하(실행된 TC수 대비) · 100% 요구사항 커버리지	· PL급 개발자	· 테스트 계획서 작성 · TC 작성 · 테스트 수행 · 테스트 결과 검토 및 승인	2회 (1회는 교차 테스팅)	Full 리그레션 테스팅
				· 프로젝트 참여 현업 (초기 참여자)	· 테스트 수행		
STTA	· 명세 분석 · 탐색적 테스팅(자유 넓은 범위 커버) · 페어와이즈 · 동등 분할 · 비정상 TC 비율* : 정상 TC의 10%	· 비공식 리뷰 (상대적으로 리스크 높은 아이템 대상)	· 80%이상의 가능한 TC 수행 · 심각도 중간 이상인 결함 없음 · Workaround 결함 20% 이하(실행된 TC수 대비)	· 개발자	· TC 작성 · 테스트 수행	1회	Partial 리그레션 테스팅
				· PL급 개발자	· 테스트 계획서 작성 · 테스트 결과 검토 및 승인		
				· 프로젝트 참여 현업	· 테스트 수행		
FTA	· 명세 분석 · 탐색적 테스팅(자유 넓은 범위 커버) · 비정상 TC 비율* : 정상 TC의 0%	· 비공식 리뷰 (상대적으로 리스크 높은 아이템 대상)	· 60%이상의 가능한 TC 수행 · 심각도 중간 이상인 결함 없음	· 개발자	· 설계&실행	1회	확인 테스팅

〈그림 3〉 리스크 기반 테스팅 전략 예시



〈그림 4〉 리스크 레벨별 누적결함 S-커브 예시

위에 따라 차등적으로 적용해야 하는데 이를 아래쪽 그림에서 테스팅 전략 테이블 형태로 표현했다. 테스트 설계 기법은 물론 테스트 완료 조건도 리스크 우선순위에 따라 차등적으로 결정해야 하고, 테스트 인력 배치 및 역할, 재테스팅 및 리그레션 테스팅의 강도 또한 리스크 우선순위를 중심으로 결정하는 전략을 수립할 수 있다.

이러한 전략 수립 프로세스와 관련 요소 및 아이템은 조직(전사) 차원에서 정의돼 있을 수 있다. 이 경우, 이미 조직 내에서 입증된 절차와 방법 및 관련 요소 등이 미리 정리되어 있으므로 해당 조직 내에서 진행하는 모든 프로젝트는 전사 차원의 테스팅 전략을 상속 받아 일부 테일러링 후 효율적/효과적으로 사용할 수 있다. 이는 조직 내 모든 프로젝트의 테스팅이 기본적으로 일정 수준을 유지하고 일관성 있게 진행되며 재활용성이 높아 단시간에 진행되도록 지원한다.

리스크 기반으로 테스팅이 전략적으로 수행된 경우 테스팅 과정에서 개발 프로젝트에 리스크와 연계된 다양하고 유용한 정보를 제공할 수 있다. 예를 들어, 리스크가 높은 시스템 부분(〈그림 4〉에서 STA 영역)에 심각도 중간 이상인 결함 발견이 누적돼 증가하는 추이와 해결(Close)되는 추이를 보여줄 수 있다. 리스크 높은 부분에 심각도 높은 결함 발견이 계속 증가하고 있고 해결이 지연되고 있다면, 누가 봐도 해당 프로젝트는 문제의 소지가 많으므로 이 데이터를 바탕으로 적절한 조치를 취할 수 있다. 이는 출시가 임박했을 경우 출시 여부를 결정하는 데에도 결정적인 정보가 된다.

또한 리스크 자체 정보도 제공할 수 있는데 리스크 레벨별로 시간이 지남에 따라 잔존하는 리스크가 얼마나 되는지 〈그림 5〉와 같이 수치화하여 표현할 수 있다. 적절히 리스크 기반 테스팅이 이뤄지고 있다면 리스크 높은 영역의 리스크 수치는 다른 리스크 영역에 비해 상대적으로 〈그림 5〉의 STA 영역에서 보여지는 바와 같이 조속히 낮아져야 한다.

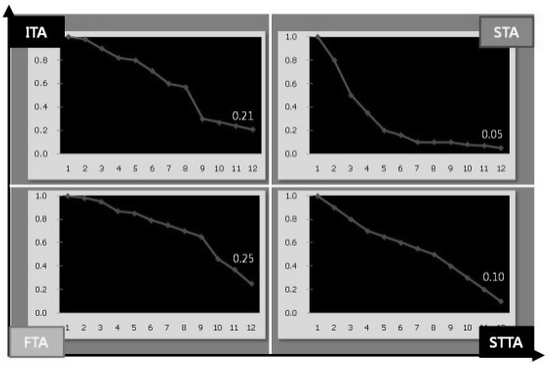
이처럼 리스크 기반 테스팅 전략은 테스팅을 수행하는 과정에서 단순한 결함 보고 이외의 가치 있는 정보를 프로젝트와 의사결정권자에게 제공한다.

4. 임베디드 소프트웨어 테스팅

임베디드 시스템은 한가지 또는 몇 가지의 기능만 수행되도록 설계된 컴퓨터 시스템으로 전자 하드웨어와 기계부분을 포함하는 전체 장치의 일부로 소프트웨어가 내장돼 있다.[16] 시스템 하드웨어를 구동시켜 목표하는 기능을 수행시키는 소프트웨어는 전용(Dedicated) 어플리케이션, OS, 미들웨어, 일반 어플리케이션 등으로 구분된다. 임베디드 시스템의 종류는 전자밥솥, 냉장고, MP3 플레이어, 카메라, DVDs, 게임기와 장난감에서부터 자동차, 의료장비, 자판기, 산업용 제어기, 로봇, 이동통신 및 네트워크 장비 등 다양하다.

임베디드 소프트웨어의 특성은 실시간성 및 상태 처리, 제한된 인터페이스 및 접근성(혼합 신호), 전용 프로세싱 기능(프로세싱 파워가 낮은 반면 빠른 응답과 속도 요구), 제한된 리소스(프로세서, 메모리), 이동성 지원 등이다. 임베디드 소프트웨어 테스팅은 이런 특성을 고려해 관련 테스팅 활동이 이뤄져야 한다. 이 밖에도 대규모 또는 고가의 장비와 같은 타 시스템 요소가 필요할 수 있어 운용환경을 시뮬레이션하기 어려우며 이 때문에 테스트 환경을 일반화하기 어렵고 테스트 자동화 도구를 도입하기 어렵다. 또한 많은 경우 모델 기반으로 개발이 이뤄지며 기능안전성 요구사항을 갖는 경우가 일반적이어서 이런 부분도 고려해 테스팅해야 한다.

임베디드 소프트웨어가 갖는 기술적인 특징에 따른 테스팅 고려사항 이외에도 임베디드 소프트웨어 개발 프로젝트가 갖는 특성도 고려해 테스팅돼야 한다. 즉, 소프트웨어 개발 인력이 소프트웨어 공학에 대한 전문 지식이 없는 경우가 대부분이고, 적은 개발자가 짧은 기간 투입돼 개발이 이뤄진다. 이런 이유로 명세서나 문서는 물론 설계의 완성도가 낮으며 소프트웨어를 테스트하는 전문 인력이나 자원을 확보하기 어렵다.[2]



〈그림 5〉 리스크 레벨 별 잔존 리스크 감소 추이예시

Ⅲ. 리스크 기반 임베디드 소프트웨어 테스팅 사례

리스크 분석 결과는 주로 테스트 우선순위나 강도를 결정하는데 사용하지만, 그 외에도 기존 테스트 케이스를 평가하고 개선하는 기준으로도 활용할 수 있다. 사용하는 용도는 다르지만 리스크 분석 결과를 토대로 테스트를 전략적으로 수행한다는 측면에서는 리스크 기반 테스트 전략의 핵심적인 일부라고 볼 수 있다. 본 사례에서는 임베디드 시스템인 불루레이를 테스팅하는 과정에서 리스크 분석 결과를 활용해 기존의 테스트 설계를 검토하고 보완하는 전략에 대해 중점적으로 다룬다.

테스트 전문 조직이나 품질조직이 있는 경우에는 대부분 테스트 케이스가 있으며, 테스트 케이스는 일반적으로 테스트 스윗(Test Suite) 형태로 관리된다. 많은 경우 이미 만들어진 기존의 테스트 스윗을 그대로 사용한 테스트 수행과 결과 중심의 테스팅 활동을 전개하며 테스트 케이스를 현행화(Update)하는 활동을 체계적으로 하는 경우는 보기 어렵다. 사용하기 어렵거나 불필요한 테스트 케이스는 테스터로부터 외면 받거나 테스트 결과를 왜곡시키게 되므로 이의 영향은 사소하지 않다. 더욱 심각한 것은 기존 테스트 케이스를 그대로 사용하면 살충제 효과(Pesticide Paradox)로 인해 시간이 갈수록 결함 발견율이 현저히 떨어진다. 이렇듯 테스트 케이스의 지속적인 유지보수는 중요하지만 관리하고 있는 테스트 케이스가 수 천에서 수 만개에 이른다면 이는 현실적으로 쉬운 일은 아니다.[11]

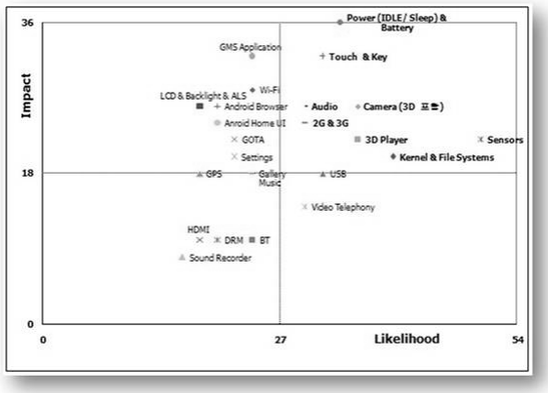
본 사례에서는 리스크 분석 결과를 유지보수가 필요한 테스트 케이스의 범위와 현행화(Update) 업무량 판단의 근거로 삼았다. 그리고 아래의 모든 과정에 개발자를 자연스럽게 참여시켜 테스트 케이스 유지보수에 대한 공감을 이끌어냈으며, 이후 개발자 새너티 테스트(Sanity test)를 프로세스화하는데 객관적인 동력이 됐다.[11]

리스크 분석 결과를 테스트 케이스 유지보수 전략에 활용한 절차는 다음과 같다.

단계 1) 유지보수 대상 시스템에 대해 리스크 분석을 진행한다. (리스크 아이템 식별, 리스크 요소 선정, 리스트 분석 과정은 앞서 설명한 일반적인 리스크 분석 과정과 동일하다)

단계 2) 기존 테스트 케이스 중 사용하지 않거나 수정된 테스트 케이스가 있다면 이를 리뷰한 후, 리스크 아이템과 연결된 테스트 케이스 수를 산출한다.

단계 3) 리스크 분석 결과에 가중치를 부여하여 각 리스크 아이템의 필요 최소 테스트 케이스 수를 산출한다.



〈그림 6〉 임베디드 소프트웨어 리스크 분포도

이 때, 리스크 수준을 반영해 (리스크 높은 부분을 중심으로) 기존의 테스트 케이스를 현행화한다.

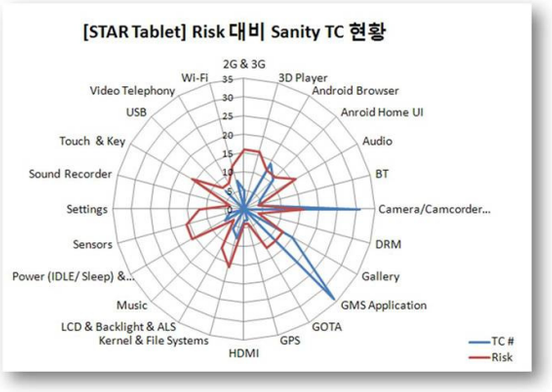
단계 4) 기존 테스트 케이스 수와 3단계에서 산출한 최소한으로 필요한 테스트 케이스 수를 비교할 수 있도록 리스크 수준과 테스트 케이스 개수를 하나의 방추형(레이더) 그래프에 표현한다.

단계 5) 리스크 레벨이 높지만 기존 테스트 케이스가 부족한 리스크 아이템이 있는지, 리스크 레벨은 낮지만 기존 테스트 케이스가 많은 리스크 아이템이 있는지 분석한다.

단계 6) 추가해야 할 테스트 케이스의 수량과 작업 규모가 결정되면 이를 토대로 인력과 일정을 산정한다.[11]

〈표 1〉 리스크 기반 테스트 케이스 수 차이 제시 (임베디드 시스템 테스팅 사례)

Risk Item	가중치	TC #	Risk	Gap
2G & 3G	4	5	16	11
3D Player	4	0	16	16
Android Browser	3	14	12	-2
Anroid Home UI	3	11	12	1
Audio	4	5	16	11
BT	1	4	4	0
Camera/Camcorder (3D 포함)	4	31	16	0
DRM	1	0	4	4
Gallery	3	15	12	-3
GMS Application	3	34	12	-22
GOTA	3	0	12	12
GPS	1	3	4	1
HDMI	1	3	4	1
Kernel & File Systems	4	8	16	8
LCD & Backlight & ALS	3	6	12	6
Music	1	0	4	4
Power (IDLE/ Sleep) & Battery	4	6	16	0
Sensors	4	4	16	12
Settings	3	0	12	12
Sound Recorder	1	1	4	3
Touch & Key	4	0	16	16
USB	2	7	8	1
Video Telephony	2	0	8	8
Wi-Fi	3	8	12	4
총 보완 필요 TC 수				104



〈그림 7〉 리스크 대비 테스트 케이스 현황 (임베디드 시스템 테스팅 사례)

Ⅳ. 리스크 기반 임베디드 소프트웨어 테스팅 전략

1. 임베디드 소프트웨어의 특성을 반영한 리스크 기반 테스트 전략 수립

임베디드 소프트웨어를 테스팅할 때 리스크 기반 테스팅 전략을 수립하고 이를 기반으로 S테스팅을 수행하는 프로세스는 위의 사례에서와 같이 대부분의 경우 일반 프로세스를 그대로 따른다. 즉, 리스크 아이템을 식별하고 리스크 요소를 선정한 후 리스크를 분석하고 이를 기반으로 테스트 전략을 수립하고 리스크의 변화 추이를 추적한다.

리스크 분석을 위한 리스크 요소 선정 시 앞서 설명한 임베디드 소프트웨어 테스팅의 특성을 반영한다. 즉, 일반 소프트웨어의 경우 리스크 아이템이 얼마나 복잡한 인터페이스를 갖는지를 보는 리스크 요소가 필요한 반면 임베디드 소프트웨어는 인터페이스의 제약성이 얼마나 되는지 파악하는 리스크 요소가 필요하다. 이 밖에도 리소스의 제약 정도, 인터페이싱하는 장비와 테스트 환경의 복잡성, 기능안전성 규정 준수 수준 등이 리스크 요소에 반영될 수 있다. 이러한 리스크 요소는 임베디드 소프트웨어의 종류와 상황에 따라 달라질 수 있어 리스크 분석 시 유의가 필요하다.

기능안전성이 요구되는 임베디드 소프트웨어의 경우 개발 과정에서 FMEA를 수행하므로 해당 결과를 리스크 분석 시 참고할 수 있다.

리스크 분석 결과를 바탕으로 리스크 기반 테스팅 전략 수립 시 세부적인 내용이 달라질 수 있다. 예를 들어, 전략 수립 시 테스트 설계 기법을 선택할 경우 임베디드 소프트웨어의 특성으로 인해 다른 테스트 설계 기법에 비해 상태전이테스팅 기법이 보다 적절히 활용될 수 있다. 제어 기능 등 로직이 복잡한 코드를 테스트할 경우, 특히 MC/DC와 같은 특정 커버리지를 달성하는 테스트가 필요한 경우, 최소비교테스트(Elementary comparison test) 기법을 리스크 기반 테스트 전략에서 리스크 높은 부분의 테스트 설계 기법으로 선정하는 것이 합리적이다. 또한 앞서 정리한 임베디드 소프트웨어의 특성 중 현재 테스트 대상 소프트웨어에 해당하는 특성이 정의되면 이런 특성으로 인해 결함이나 장애가 발생하는지 확인하는 결함 공격(Fault Attack) 형태의 테스트 설계가 유용하다. 이는 앞서 언급한 임베디드 소프트웨어 개발 프로젝트 특성에도 맞아 적절하게 활용될 수 있는 테스트 설계 기법으로 판단된다.

이 밖에도 임베디드 소프트웨어 개발 프로젝트 특성을 반영하여 탐색적 테스팅 접근법이 리스크 기반 테스팅 접근법과 병행해 사용될 수 있다. 또한 임베디드 소프트웨어 개발 프로젝트가 소수의 개발 관련자가 테스트를 수행하는 경우가 많고 제품이 출시되면 현행화가 어려우므로 테스트 전략 수립 시 단위 테스트 전략이나 통합 테스트 전략을 다른 레벨의 테스트 전략에 비해 강도 높게 수립하는 것도 유용할 수 있다.

2. 리스크 기반 임베디드 소프트웨어 테스팅 효과

위의 임베디드 소프트웨어의 특성을 반영하여 리스크 기반 테스트 전략을 수립한 사례를 통한 효과는 테스팅이 체계화되고 테스팅 과정에서 리스크를 관리하고 최소화시키는데 기여한다는 것이다. 임베디드 분야에서 수치화된 효과 분석 연구가 현존하지는 않지만 적용 사례에서 다음과 같은 효과를 발표한다.[15]

- 수치적 접근이 테스팅 조직(팀)의 전문성을 높여줌
- 개발팀(PL)과 프로젝트 초반부터 원활한 의사소통
- 테스트 계획 작성에 개발팀(PL)의 자연스러운 참여

이 밖에도 임베디드 분야는 아니지만 최근 리스크 기반 테스팅 전략에 따른 테스팅을 3년간 수행한 결과 출시 후 고객으로부터 접수되는 결함의 수가 1/10로 감소된 사례 등이 업계 세미나를 통해 발표되고 있다.[7]

임베디드 분야에 리스크 기반 테스트 전략을 수립하기 위해 리스크를 분석하는 과정에서 얻은 교훈은 개발 계획 시에도 리스크 분석결과를 활용할 수 있고, 이해관계자가 누구로 선정하느냐가 중요하며, 리스크 분석은 이해관계자가 주관적으로 판단해 수치화하는데 이것이 모이면 대표성과 객관성을 갖게돼 이를 기반으로 한 테스팅 활동이 설득력을 갖는다는 것이다.[15]

V. 향후 연구 및 결론

리스크 기반 테스트 전략에 대해 살펴봤고, 임베디드 소프트웨어의 특성을 정리하고 실무에 사용된 리스크 기반 테스트 활용 사례를 살펴본 후 이를 반영한 테스트 전략 수립 방안을 제시했다.

적용 사례에서와 같이 리스크 기반 테스트 전략이 임베디드 분야라고 전략 수립 프로세스와 주요 내용이 바뀌지는 않는다. 단지 임베디드 소프트웨어의 특성을 반영해 리스크 분석 과정에서 리스크 요소가 변경될 수 있고, 전략 수립 시 적용할 수 있는 설계 기법 등이 달라진다. 이런 유형의 변형은 다른 어떤 분야의 소프트웨어를 테스팅하더라도 발생하는 것이어서 임베디드 소프트웨어나 시스템을 대상으로 리스크 기반 테스트 전략을 활용하는 것이 프로세스나 방법론 측면에서는 큰 차이가 없다고 판단한다.

임베디드 분야에 리스크 기반 테스팅을 확산하기위해서는 리스크 기반 테스팅이 임베디드 분야에서 적용된 효과를 수치화된 분석 자료로 보여주는 연구가 필요하다. 즉, 전사 차원의 리스크 기반 테스트 전략을 수립하고 이를 프로젝트 차원의 테스트 전략, 엔지니어링 차원의 테스트 전략으로 구체화하여 제어된 환경에서 적용하고 (Controlled experiments) 그 효과를 확인해 본 제안의 타당성을 검증하는 연구가 필요하다.

이 밖에도 임베디드 소프트웨어가 하드웨어에 의존하는 정도, 소프트웨어 스택에서의 위치(하드웨어를 제어하는 드라이버 레벨 ~ 어플리케이션 레벨)에 따른 리스크 요소와 테스트 전략의 차별적 적용에 대한 연구가 필요하다.

또한 본 연구에서 다룬 제품을 부분으로 나눠 부여한 제품 리스크 이외에 별도의 제품 리스크 및 프로젝트 리스크를 도출하고 이를 최소화하는 방법과 접근법을 연구하여 제안된 전략을 보완하고 완성도를 높이는 노력이 필요하다.

리스크 기반 테스트 전략은 임베디드 소프트웨어 또는 시스템 개발에서 제한된 테스트 리소스의 합리적이고 적절한 사용을 직접적으로 지원하다. 모든 테스트 관련 활동의 추적성을 확보하고 테스팅 전 과정을 리스크를 중심으로 모니터링 및 리포팅하고 제어하여 테스팅의 효율적/효과적 수행은 물론 산출되는 제품의 품질 향상에 직접적으로 기여한다.

참고문헌

[1] B. Broekman 외, (역자-윤희병, 권원일 외), 임베디드 소프트웨어 테스팅, 홍릉출판사, 2008.

[2] Ian F. Smith, Test automation for embedded products, 2004. 6.

[3] IEEE 829 “Standard for Software Testing Documentation”, 2008.

[4] ISO/IEC 29119-2, Software Testing, DIS, 2012.

[5] M. Hetrick, Mastering the project portfolio, MPP Stanford Online v2.0, 2010.

[6] TMMi Framework v3.1, TMMi Foundation, 2011.

[7] 강성훈, SW 전문 기업의 리스크 기반 테스팅 추진 사례, STEN 세미나(리스크 기반 테스팅 사례 - http://sten.kr/bbs/tb.php/news/2432), 2011. 12.

[8] 권원일, 권호열, TMMi 테스트 계획 프로세스 영역(PA2.2)의 진단과 분석, 대한전자공학회 추계학술대회, 2011. 11.

[9] 권원일 외, 개발자도 알아야 할 소프트웨어 테스팅 실무 3판, STA컨설팅, 2011.

[10] 권원일 외, 리스크 기반 개발 전략(리스크 분석 결과를 개발과정에서 활용), 제33회 한국정보처리학회 춘계학술발표대회 논문집 제17권 제1호, 2010. 4.

[11] 권원일, 윤진우, 최은희, 위험천만 테스팅, STA컨설팅, 2012.

[12] 권원일, 위험천만 테스팅, 리스크와 ISO표준 및 TMMi와의 관계, STEN 세미나(리스크 기반 테스팅 사례 - http://sten.kr/bbs/tb.php/news/2432), 2011. 12.

[13] 리스크 분석 및 전략 수립 지원 툴 (Radar) - www.sten.or.kr/radar

[14] 윤진우, 권원일, “소프트웨어 품질 리스크의 수치화에 관한 연구”, 한국정보처리학회 춘계학술대회, 2010. 4.

[15] 최은희, 임베디드 소프트웨어 리스크 기반 테스팅 (전략 수립) 사례, STEN 세미나(리스크 기반 테스팅 사례 - http://sten.kr/bbs/tb.php/news/2432), 2011. 12.

[16] 한무희, 소스코드 기반 임베디드 소프트웨어 테스팅 사례, 강원대학교 SW공학특론 강좌 테스팅 세미나, 2011. 11.

권 원 일

1998년 5월 Auckland University 학부 (학사).
2000년 2월 KAIST (석사).
2000년 1월~2003년 8월 ETRI, TTA SW시험인증센터.
2003년 8월~현재 (주)STA테스팅컨설팅 대표.
2008년 10월~현재 ISO/IEC 33063 주에디터 (표준화 위원).
〈관심분야〉 소프트웨어 테스팅, 소프트웨어 품질 관리, 요구공학