TPI 기반 국방 임베디드 SW 테스트 전략

국방대학교 윤희병

1. 서 론

임베디드 시스템이란 어떤 제품이나 솔루션 등에 탑재되어 특정 기능을 수행하는 시스템으로 일반적으로 사람의 간섭없이 독자적 기능을 수행하는 하드웨어와 소프트웨어의 결합으로 구성되며, 여기에 내장된 소프트웨어가 바로 임베디드 소프트웨어이다. 이중 무기체계 임베디드 소프트웨어는, 특히 무기체계에 탑재되어 하드웨어를 구동 및 제어, 또는 하드웨어 기능을 대체하는 역할을 수행하기 위하여 개발된 특정한 소프트웨어를 지칭한다[1]. 이러한 무기체계 임베디드 소프트웨어는 항공/함정무기체계, 정밀타격무기체계, 유도무기체계, 사격통제무기체계, 미사일 등으로 그 영역을 넓혀가고 있다.

임베디드 소프트웨어 테스트는 소프트웨어의 결함을 발견하기 위한 프로세스로서 숨어있는 결함을 찾기 위해 소프트웨어를 작동하는 일련의 행위와 절차를 일컫는다. 이러한 테스팅의 궁극적인 목적은 소프트웨어의 요구사항과 관련된 결함의 감시를 바탕으로 조직이 어떻게 시스템을 발전시킬 수 있는지에 대한 정보를 제공하는데 있지만, 테스팅 자체로는 소프트웨어의 품질을 직접적으로 개선할 수 없다. 그러나 조직과 관련된 위험과 시스템의 약점을 관찰하는 명확한 방법을 제공함으로써 간접적으로 시스템의 품질개선에 관여한다. 이처럼 테스팅이 시스템의 품질개선에 중요하지만 많은 조직들이 비전문가에 의해 거의 제품화되는 시기에 테스팅이 이루어짐으로써 결과적으로 나중에 수정하려면 비용이 많이 드는 결함이 그대로 남아있는 상태로 시스템을 인수받게 된다.

또한 테스트를 효율적으로 수행하기 위해서는 시스템의 명세에 기초하여 실행되기 전에 계획되고 준비되어야 하나, 실제 테스팅을 수행하는 부서에서는 테스팅에 필요한 시간, 인력, 자원, 전문가가 부족한 실정이고 대부분 개발 단계에 늦게 합류함으로써 다시 작업하거나 다시 테스트해야 하는 상황을 초래하기도 한다. 이를 극복하기 위해서는 테스팅에 대한 전략적 투자 및 효율적인 테스팅 전략을 수립하는 것이 필요한데 여기에는 테스트 프로세스의 개선, 테스트 자동화 및 테스트 관련 산출물들의 재활용 등이 고려되어야 한다.

테스트 프로세스 개선을 위해 설계된 참조모델로는 TMM(Testability Maturity Model), TIM(Test Improvement Model), SW-TMM(software Testing Maturity Model), TAP(Testing Assessment Program) 등[2,3]이 있으나 테스팅에 대한 개선방안과 실무적인 세부지침 등이 포함되어 있지 않다. 이러한 점 때문에 요구사항을 만족하기 위한 실무적인 지식과 기술에 바탕을 두고 설계된 것이 바로 테스트 프로세스 개선(TPI: Test Process Improvement) 모델[4,5]이며 이 모델은 전세계적으로 널리 사용되고 있고 현재도 모델의 개선 및 지원이 지속적으로 이루어지고 있다.

따라서 본 논문에서는 무기체계 임베디드 소프트웨어의 각 핵심영역별 효율적인 테스트 전략을 제안하기 위하여 먼저 테스트 프로세스 개선 모델에 대한 일반적인 개념과 프로세스 성숙도 모델을 설명한다. 이러한 성숙도 모델을 바탕으로 무기체계 임베디드 소프트웨어를 테스팅하기 위해 필요한 핵심영역을 도출하며 체크포인트를 이용하여 각 핵심영역별 테스트 프로세스를 평가한다. 테스트 프로세스의 평가 결과에 따라 핵심영역별 테스트 프로세스의 장·단점을 분석하며 이를 통해 각 핵심영역별로 효율적인 임베디드 소프트웨어 테스트 전략을 제안한다.

2. 테스트 프로세스 개선 모델

2.1 테스트 프로세스 개선 모델의 개념

테스트 프로세스 개선(TPI) 모델은 테스팅 업계의 요구사항을 반영하여 개발된 테스트 프로세스를 개선하기 위해 사용하는 모델이며 소프트웨어의 품질, 비용, 테스트 프로세스의 소요 시간 등 모든 정보서비스

의 최적화를 위한 모델이다[6].

테스트 프로세스를 개선하고자 하는 목적은 결함의 수정비용을 최소화할 수 있도록 신속한 결함의 발견과 시스템의 품질과 관련된 정보의 신속한 제공에 있으며, 이러한 TPI 모델은 조직 내의 테스트 프로세스의 성숙도를 보는 시각을 제공하며 순차적이고 통제 가능한 개선방안을 제시한다. TPI 모델에서는 각 핵심영역의 수준이 정의되고 상위수준의 도달을 위한 개선방안이 제시됨으로써 개선단계의 제어가 가능하며 실무에서 증명된 방법론을 기초로 개발되었으며 세부적인 설명을 포함하고 있다.

또한 현 상태에 대한 상세하고 구체적이며 신속한 분석을 가능하게 하는 체크포인트를 활용하여 테스트 프로세스에 대한 핵심영영을 객관적으로 평가할 수 있도록 제시하고 있으며, 시스템 전체에 대한 적용뿐만 아니라 시스템의 중요한 부분에 대해서도 선택적으로 적용할 수 있고 또한 개선활동을 위한 팁으로서도 사용이 가능한 모델이다. 이러한 TPI 모델의 상위수준에서의 주요 개념[6,7]이 그림 1에 나타나 있다.

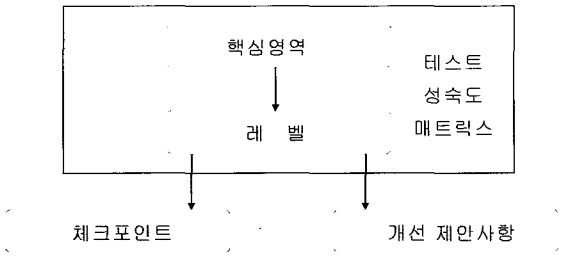

그림 1 테스트 테스트 프로세스 개선 모델의 개념

2.2 핵심영역 및 레벨

핵심영역이란 시스템을 테스트할 때 필요한 잘 정의된 테스트 프로세스를 만들기 위해 시스템에서 특별한 주의가 요구되는 특정 영역을 말한다. 이러한 핵심영역은 테스트 프로세스를 개선하고 구조화하는 기본요소이다. TPI 모델에는 20개의 핵심영역을 가지고 있으며 이러한 20개 핵심영역의 이름이 표 1에 기술되어 있다[6].

핵심영역은 프로세스의 성숙도를 결정하기 위해 구성되어 있다. 핵심영역의 상태를 관찰하기 위해 TPI 모델에서는 레벨을 적용하는데, 가장 낮은 등급은 레벨 A이며 가장 높은 등급은 레벨 D로서 상위레벨은 시간, 비용, 품질 면에서 더욱 우수한 것으로 평가된다. 각 레벨에는 핵심영역별로 요구사항(체크포인트)이 제시되어 있는데 특정 레벨을 받기 위해서는 해당 등급의 활동이 체크포인트를 만족해야 한다. 또한 보다 상위 등급을 받기 위해서는 하위 등급의 체크포인트를 모두 만족해야 한다. 테스트 프로세스가 레벨 A를 만족시키지 못하면 프로세스는 기초수준이라 한다.

20가지의 핵심영역별 레벨 A부터 레벨 D까지의 내용이 표 2에 나타나 있다[6,7]. 이러한 핵심영역별 레벨을 결정하기 위해, TPI 모델에서는 객관적인 측정도구인 체크포인트를 제공하고 있다. 이러한 핵심영역별 각 레벨별 체크포인트를 활용하여 테스트 프로세스를 평가하며 평가 결과를 통해 핵심영역에 적합한 레벨을 결정할 수 있다.

2.3 테스트 성숙도표(Test Maturity Matrix)

테스트 성숙도표는 현 테스트 프로세스의 강점과 약점을 제시하고 개선을 위한 우선적인 활동을 지원하는데 그 목적이 있다. 테스트 성숙도표는 레벨과 핵심영역 사이의 내부 우선순위와 의존성을 나타내는 데 유용하게 사용된다. 테스트 성숙도표의 세로축은 20개의 핵심영역을 나타내고, 가로축은 총 13단계의 성숙도 정도를 나타낸다. 각 레벨은 테스트 성숙도의 수준으로 나타낼 수 있다. 즉, 1~5의 세부레벨을 만족하면 테스트 프로세스가 제어되고 있음을 나타내며, 6~10의 세부레벨까지 만족하면 효율적 수준을 달성한 것이고, 마지막으로 최적화 수준은 11~13의 세부레벨을 모두

표 1 TPI 모델에서의 20개 핵심영역

테스트 전략	참여와 동기
생명주기 모델	테스트 기능과 훈련
수행시점	방법론의 범위
산정과 계획	의사소통
테스트명세 기법	보고
정적테스트 기법	결함관리
매트릭스	테스트웨어 관리
테스트 도구	테스트 프로세스 관리
시험환경	평가
사무실 환경	하위레벨 테스트

표 2 핵심영역별 레벨

핵심영역	레벨 A	레벨 B	레벨 C	레벨 D
테스트 전략	단독 상위레벨 테스트 전략	상위레벨 테스트 혼합전략	레벨 B 및 하위레벨 테스트와 평가	모든 테스트와 평가 레벨의 혼합전략
생명주기 모델	계획, 명세, 실행	계획, 준비, 명세, 실행, 완료		
수행시점	테스트 근거 완성	테스트 근거 시작	요구사항 정의단계시작	프로젝트 초기화
산정과 계획	산정과 계획 구체화	통계적 산정과 계획 구체화		
테스트명세기법	비정형 기술	정형 기술		
정적테스트기법	테스트 근거의 감독	체크리스트		
매트릭스	프로젝트 매트릭스 (제품)	프로젝트 매트릭스 (프로세스)	시스템 매트릭스	조직 매트릭스
테스트 자동화	계획 및 제어도구	실행 및 분석도구	광범위한 테스트 프로세스	
시험환경	시험환경 관리 및 제어	적합한 환경의 테스팅	즉각 테스팅 환경	
사무 환경	적합성, 적시성			
참여와 동기	예산과 시간 할당	프로젝트조직에 테스트 통합	테스트-공학	
테스트 기능 및 훈련	테스트 관리자와 테스터	정형적 방법, 기술, 기능적 지원, 관리	정형적 내부 품질보증	
방법론 범위	프로젝트 명세	조직의 일반	조직 최적화	
의사소통	내부 의사소통	프로젝트 의사소통 (결함, 변경관리)	품질/테스트 프로세스에 대한 조직내 의사소통	
보고	결함	진행상황, 활동, 우선순위가 있는 결함	위험 및 권고, 구현된 매트릭스	소프트웨어 프로세스 특성 개선의 권고
결함관리	내부 결함관리	유연한 보고도구 사용 광범위한 결함관리	프로젝트 결함관리	
테스트웨어 관리	내부 테스트웨어 관리	테스트 근거와 테스트 객체에 대한 외부 관리	재사용 가능한 테스트웨어	시스템 요구사항의 테스트케이스 추적성
테스트 프로세스 관리	계획과 실행	계획, 실행, 감시, 적용	조직 차원의 감시, 적응	
평가	평가 기술	평가 전략		
하위레벨 테스트	하위레벨 테스트 생명주기	화이트-박스 기술	하위레벨 테스트 전략	

만족할 때 달성된다. 이러한 테스트 성숙도표가 표 3에 나타나 있다[6].

3. 임베디드 소프트웨어 테스팅 메소드 및 핵심영역 도출

3.1 임베디드 소프트웨어 테스팅 메소드

구조화된 테스팅 방법으로서 임베디드 소프트웨어 테스팅(TEmb: Testing Embedded Software) 메소드가 사용된다. 이 메소드는 4가지의 기본요소로 이루어져 있으며 개발주기와 연관된 테스트 활동을 정의한 생명주기(L: Life cycle), 활동을 수행하기 위해 사용가능한 기법(T: Technique), 적합한 기반구조 및 도구(I: Infrastructure), 그리고 바람직한 조직(O: Organization) 등이 있다. 이러한 기본요소는 테스트 프로세스에서 반드시 고려되어야 하며 테스트 프로세스의 균형을 잡기 위해서는 4가지 기본요소 간 균형이 필요하다[8].

각 테스트 레벨마다, 구조적 테스팅의 4가지 기본요소는 공통적으로 적용되며 이러한 기본요소들은 "무엇을 언제," "어떻게," "무엇으로," "누구에 의해서"에 대한 답변이라 말할 수 있다. 모든 기본요소는 잘 정의된 테스트 프로세스라면 공평하게 사용되어야 하며 하나라도 무시된다면 그 테스트 프로세스는 문제가 생기게 된다. 4가지의 기본요소 중 생명주기는 중심적인 기본요소로서 기본요소 간 접착제와 같은 기능을 하고 생명주기의 각 단계마다 다른 세 가지 기본요소인 기술, 기반구조, 조직과 서로 상호작용을 해야 한다.

표 3 테스트 성숙도표(Test Maturity Matrix)

핵심영역	0	1	2	3	4	5	6	7	8	9	10	11	12	13
		제어					효율적					최적화		
테스트 전략		A					B				C		D	
생명주기 모델		A			B									
수행 시점			A				B				C		D	
산정과 계획				A							B			
테스트명세기법		A		B										
정적테스트기법					A		B							
매트릭스						A			B			C		D
테스트 자동화				A				B			C			
시험환경				A				B						C
사무실 환경				A										
참여와 동기		A				B						C		
테스트 기능 및 훈련				A			B			C				
방법론 범위					A						B			C
의사소통			A		B							C		
보고		A			B		C					D		
결함관리		A				B		C						
테스트웨어 관리			A			B				C				D
테스트 프로세스 관리		A		B								C		
평가							A			B				
하위레벨 테스트					A		B		C					

3.2 핵심영역 도출

현재 군에서 수행중인 무기체계 임베디드 소프트웨어의 군관리 업체주도라는 개발형태로 비추어 볼 때, 임베디드 소프트웨어 테스트는 개발자(업체)에 의해 주도적으로 실시되고 있다고 볼 수 있으며 발주자인 소요군은 업체가 작성한 기술자료 검토 및 평가라는 수락 위주의 시험평가를 하고 있다고 볼 수 있다.

이러한 환경하에서 TPI 모델의 20개 핵심영역 중 임베디드 소프트웨어 테스트에 필요한 핵심영역을 도출하기 위해서는 개발자가 자체적으로 실시하는 단위 및 단위통합시험, 형상품목수락시험, 소프트웨어 형상품목 및 하드웨어 형상통합시험, 소요군이 실시하는 수락시험, 기술시험, 운용시험, 그리고 무기체계 임베디드 소프트웨어의 여러 특성들을 고려하여야 한다.

도출된 9개의 핵심영역에는 ① 무기체계의 특성을 고려한 테스트 전략, ② 테스트 프로세스 중 어떤 한 부분이 아닌 전체 생명주기 동안 개선활동이 필요하다는 측면에서 생명주기, ③ 개발자가 자체적으로 수행한 테스트에 대한 적합성 및 수행여부를 판단하기 위해 필요한 테스트명세 기법, ④ 개발자가 소요군에 제출한 산출물을 바탕으로 시험가능성 및 적절성을 판단하는데 필요한 정적테스트 기법, ⑤ 무기체계 임베디드 소프트웨어의 제어시스템적인 특성을 고려하여 조성해야 하는 시험환경, ⑥ 테스트 인원의 전문성 확보로 테스트의 질 향상을 위한 테스트 조직 및 훈련, ⑦ 개발자와 소요군의 지속적인 의사소통을 통한 원만한 시험관리를 위한 의사소통, ⑧ 소요군이 개발자로부터 시스템 관련 보고를 받고 조치를 취해야 하는 보고, 마지막으로 ⑨ 테스트 수행 간 발생된 결함에 대한 조치 및 품질향상에 필요한 결함관리 등이 있다[9].

제시된 9개의 핵심영역은 임베디드 소프트웨어 테스팅(TEmb)의 기본요소인 수명주기(L), 기법(T), 인프라구조(I), 조직(O)에 의해 그림 2와 같이 분류될 수 있다. 생명주기(L)에는 테스트 전략과 생명주기 모델, 기법(T)에는 테스트명세 기법과 정적테스트 기법, 기반구조(I)에는 시험환경, 그리고 조직(O)에는 테스트

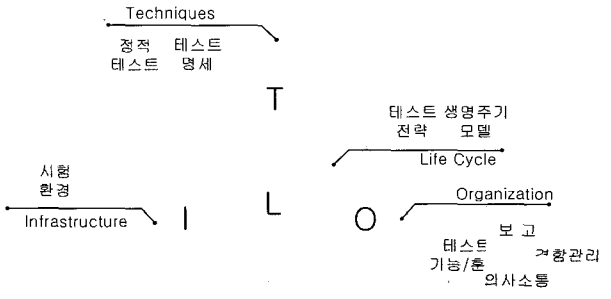

그림 2 도출된 9개 핵심영역 분류

기능 및 훈련, 의사소통, 보고, 결함관리가 있다.

4. 핵심영역별 테스트 프로세스 평가

4.1 핵심영역별 평가 결과

도출된 9개의 핵심영역에 대하여 TPI 기반의 체크포인트를 적용하여 평가를 수행한다. 평가는 그동안 군에서 성능개량 사업시 수행되었던 시스템들 중 1개 시스템의 테스트 활동사례를 그 대상으로 하였다. 여기서 제시된 평가 결과는 이러한 결과를 통해 각 핵심영역별로 어떻게 효율적인 테스트 전략을 도출하는지 그 방법을 보여주기 위해 활용한 사례일 뿐이다. 평가시 해당 레벨의 체크포인트가 모두 만족되었을때 해당 레벨을 만족한 것으로 간주한다.

4.1.1 테스트 전략(Test Strategy)

테스트 전략이란 가능한 신속하고 저렴하게 중요한 결함을 발견하는 것이 목적이며 어떤 테스트에 의해 제품의 위험과 요구사항이 만족되는가를 정의한다. 그리고 각각의 테스트 레벨에 맞춰 자체적인 전략을 수립하고 서로 다른 테스트 레벨이 혼합적으로 적용됨을 의미한다.

평가 결과, 레벨 A '단독 상위레벨 테스트 전략'은 4개 체크포인트 중 2개만을 만족하였고, 레벨 B '상위레벨 테스트를 위한 혼합 테스팅 전략'은 2개, 레벨 C '상위레벨 테스트를 위한 혼합전략 및 하위레벨 테스트 또는 평가'는 3개, 레벨 D '모든 테스트와 레벨에 대한 혼합전략'은 3개 체크포인트를 만족하지 못하여 기초수준으로 평가되었다.

도출된 결과 중 레벨 A에 대해 살펴보면, 먼저 체크포인트 '전략의 결정 및 관리 부분'이라는 항목은 현재 테스트 레벨별로 요구사항에 기초하여 테스트 전략을 결정하고 관리하고 있었기 때문에 만족된 상태로 볼 수 있었다. 예를 들면, 하위레벨 테스트는 단위시험 및 단위통합시험을, 상위레벨 테스트에서는 CSCI/HWCI 통합시험, 시스템 수락시험, 형상품목수락시험을, 평가단계에서는 기술시험 및 운용시험을 실시하고 있었다. 그러나 체크포인트 위험에 따른 테스트 레벨이라는 항목은 현재 위험분석 과정이 없었고 무기체계 관리지침에 의한 테스트 레벨만 정의되어 있으므로 만족하지 못한 상태이었다. 그리고 테스트명세는 소프트웨어 시험명세서(STD)에 기록하고 있으므로 만족스런 상태이었고 재시험 테스트 전략은 재시험 자체를 고려하지 않고 있으므로 불만족한 상태이었다.

4.1.2 생명주기 모델(Life Cycle Model)

생명주기 모델은 여러 활동을 계획하고 감시하며 상호 응집하게 하여 누가, 무엇을, 언제 임무를 수행해야 하는지를 정의함으로써 테스트 프로세스 관리를 가능하게 한다. 소프트웨어 시험계획서(STP) 및 품질계획서를 참조하여 평가한 결과, 레벨 A '계획, 명세, 실행 항목'은 4개 체크포인트 모두를 만족하였고, 레벨 B '계획, 준비, 명세, 실행, 완료 항목'은 3개 체크포인트 모두를 만족하지 못함으로써 레벨 A로 평가되었다.

세부적으로 살펴보면, 현재 무기체계 임베디드 소프트웨어 테스트는 시험계획, 시험명세, 시험실행의 3단계로 이루어지고 있었다. 계획단계에서는 프로젝트 소유자, 실행자가 공식적으로 지정되어 있었고 전체 계획이 결정되며 프로젝트 관리자 및 개발관리자의 임무 및 책임이 구분되어 있었다. 다음 명세단계에서는 테스트 스크립트가 준비되어 있었고 실제 테스트시 사용되는 테스트케이스는 소프트웨어 요구사항 명세서(SRS)의 요구사항에 1:1로 정의되어 있었다. 그러나 재시험은 정의되어 있지 않았고 이에 대한 스크립트나 테스트케이스도 없었다. 또한 생명주기에서 준비, 완료단계가 정의되어 있지 않았으며 준비, 완료단계에서 수행하는 활동 역시 정의되어 있지 않아 이들 단계는 생략되어 있었다.

4.1.3 테스트명세 기법 (Test Specification Technique)

테스트명세 기법이란 소스 정보로부터 테스트케이스를 유도하는 정형화된 방법이다. 테스트명세 기법을 개선함으로써 테스트 전략은 구체화될 수 있고 적합한 테스트를 할 수 있으며, 임의적으로 골라낸 테스트 케이스보다 결함을 효과적으로 탐지할 수 있다. 또한 주어진 테스트의 품질과 수준을 미리 살펴볼 수 있으므로 테스트의 재사용성이 증가한다. 평가 결과, 레벨 A는 '테스트케이스 구성,' 레벨 B는 '테스트케이스의 획득방법'을 만족한 것으로 나타났다. 반면 레벨A의 '문서화에 의한 테스트케이스 정의,' 레벨 B의 '테스트케이스의 적용 완전성' 및 '테스트웨어의 재사용성'은 만족하지 못한 것으로 나타남으로써 전체적으로 테스트 명세 기법은 기초수준이었다

4.1.4 정적테스트 기법(Static Test Technique)

정적테스트 기법은 소프트웨어 실행 없이 테스트를 수행하는 것을 말한다. 이러한 테스트는 산출물의 평가에 관심을 가지며 실제 품질보다 측정값의 품질에 초점을 맞춘다. 일반적으로 정적테스트 기법은 동적테스트 기법보다 저렴하고 빠르게 실행될 수 있으며 체크리스트가 유용하게 사용된다. 평가 결과, 레벨 A는 '테스트 산출물을 대상으로한 테스트 가능성 연구'와 '체크리스트 활용' 모두를 만족하였고, 레벨 B는 '고객의 승

인을 받은 체크리스트'가 만족하지 못하여 레벨 A로 평가되었다. 여기서 불만족된 것은 현재 테스트 산출물에 대한 검토 및 점검은 존재하지만 프로젝트 관리자 또는 군으로부터 승인받은 체크리스트를 활용한 기술평가 등이 미비하였기 때문이었다.

4.1.5 시험환경(Test Environment)

테스트의 실행은 시험환경에서 일어나며 이러한 시험환경의 주요 요소는 시험 장비, 개발된 하드웨어나 소프트웨어, 운용체제나 애플리케이션 프로그램, 데이터베이스와 테스트 파일, 사용되는 설비와 의사소통 수단까지 모두를 포함한다. 시험환경은 품질, 리드타임, 테스트 프로세스의 비용에 영향을 미치며 신뢰성, 관리성, 적시성, 유용성, 표현성, 유연성 등의 특성을 갖고 있다. 평가 결과, 레벨 A는 5개 체크포인트 모두 만족하였고, 레벨 B는 2개, 레벨 C는 1개의 체크포인트가 불만족되어 시험환경을 레벨 A로 평가하였다. 체크포인트인 '시험환경의 독립성'과 '빠르게 적용 가능한 준비된 환경'이 불만족된 것은 시험환경이 독립적으로 구성된 것이 아니라 개발실의 연장개념으로 개발실에서 테스트를 실행하는 경우가 많았기 때문이었다.

4.1.6 테스트 기능 및 훈련 (Test Function and Training)

테스트 기능 및 훈련에 관한 평가는 대부분 불만족으로 나타나 기초수준으로 평가되었다. 레벨 A의 '테스트 인원의 구성'이 불만족된 것은 현재 품질보증 조직은 구성되어 있으나 테스트 조직이 공식적으로 지정되어 있지 않았고 자체시험의 경우 테스터와 개발자가 동시에 지정되어 있기 때문이었다. 다만 기술시험과 운용시험 간에는 시험절차서에 따라 군이 승인한 시험관리자의 입회하에 테스터가 시험을 실시함으로써 시험을 위해 구성된 독립된 조직으로 볼 수 있었다. 그러나 테스트를 실행하기 전 테스트와 관련된 교육이 없었기 때문에 '테스트 관련 교육'과 '테스트, 테스트 주제에 관한 교육'이 불만족한 것으로 평가되었다.

4.1.7 의사소통(Communication)

테스트 프로세스에서 의사소통은 군, 개발자 등 다양한 인원이 포함된 가운데 여러 가지 형태로 발생하며 테스트 프로세스의 원활한 운용을 위해 의사소통은 매우 중요하다. 평가 결과, 의사소통은 레벨 A이었다. 레벨 A '내부 의사소통' 항목에서 모두 만족된 것으로 평가된 것은 현재 테스트 팀의 전체 구성원이 참석하는 정기적인 회의가 있었으며 진행 상황 및 테스트 되는 객체의 품질에 대한 논의가 이루어지고 있었기 때문이었다. 그러나 테스트 관리자의 주기적인 결함 및 위험과 관련된 보고가 없었고 테스터와 관리자, 또는 개발자가 합의한 사항은 문서화되어 남아있지 않았다. 또한 테스트 관련 회의는 오로지 테스터들로만 이루어져 테스트를 통한 결함의 발견이나 다른 개발단계에 적용 가능한 개선방안의 활용이 제한적이었기 때문이었다.

4.1.8 보고(Reporting)

테스트에서 보고는 관리자와 사용자에게 구체적 권고를 주는 것에 초점을 맞춘다. 결함뿐만 아니라 추가 진행사항에 관한 보고는 테스트에 대한 추가 비용과 완료시점을 추정하게 해주며 계획에 대한 실행가능성까지 판단 가능하게 만든다. 평가 결과, 보고는 기초수준으로 평가되었다. 보고영역이 기초수준으로 평가된 것은 현재 정기적이며 주기적인 결함보고 없이 최종단계에서만 보고되고 있어 단위시험 및 통합시험의 수행여부 및 수행의 적절성이 확인되지 않았기 때문이었다. 또한 발견된 오류 및 결함은 그 자체로만 관리되며 이렇게 발견된 오류와 결함의 추세 분석 등도 없어 테스트가 단지 테스트로만 끝나고 테스트로 인한 개선(권고사항)이 개발단계 등에 영향을 주지 못하고 오직 시험시 결함을 발견하고 수정하는데 그쳤기 때문이었다.

4.1.9 결함관리(Defect Management)

결함관리가 프로젝트적 문제이지만 테스터에게는 특히 제한사항이 되는 문제이다. 좋은 결함관리는 전 생명주기 동안 결함이 감시될 수 있고 여러 가지 통계적 형태로 전체 내용을 보여줄 수 있다. 이러한 결함관리는 품질상태로 구체화되며 결함을 기록함으로서 결함을 관리하고 감시하며 조정할 수 있게 만든다. 평가 결과, 결함관리는 기초수준으로 평가되었다. 결함관리가 기초수준으로 평가된 것은 현재 수락시험에서 발견된 결함에 대한 보고만 이루어졌으며 결함보고를 위한 항목 중 고유번호, 결함 유발자, 날짜, 심각도 카테고리가 생략되어 구조적인 결함관리가 제한되었기 때문이다. 결함에 대한 적시적인 기록과 관리는 있었으나 전체적인 소프트웨어에 대한 결함 경향 관리 및 분석은 없었다. 단지 테스트에 대한 시험항목과 시험환경, 시험절차 및 일정만 검토하였으며 결함에 대한 분석은 미비하였다. 이는 결함에 대해 책임지는 품질보증관리자는 있으나 결함과 관련된 위원회가 존재하지 않음으로서 생기는 문제이었다.

4.2 테스트 성숙도표(TMM)를 활용한 평가 결과

테스트 프로세스의 평가 결과, 생명주기 모델, 정적 테스트 기법,시험환경, 의사소통 등의 핵심영역에서는 레벨 A를 달성하였으나, 테스트 전략, 테스트명세 기

법, 테스트 기능 및 훈련, 보고, 결함관리 등의 핵심영역에서는 기초수준으로 평가되었다. 이러한 핵심영역별 만족 수준을 테스트 성숙도표(TMM)를 활용하여 도표로 나타낼 수 있다.

TMM 수준을 달성하기 위하여 각 9개의 핵심영역은 일정 수준에 도달해야 하며 4.1절에서 기술한 각 핵심영역별 평가 결과 사례를 그림 3과 같이 나타낼 수 있다. 그림에서 적색으로 표시된 부분이 각 핵심영역이 도달한 현 상태의 레벨을 의미하며 이 부분을 표 상단에 있는 세 가지 수준과 맞추어 보면 핵심영역별 만족 수준을 얻을 수 있다. 예를 들면, 핵심영역 중 생명주기 모델, 정적테스트 기법, 시험환경, 의사소통은 제어 수준이며, 테스트 전략, 테스트명세 기법, 테스트 기능 및 훈련, 보고, 결함관리는 아직 제어되지 못한 기초수준임을 알 수 있다.

현 상태

핵심 영역		제어					효율적					최적화		
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
테스트 전략		A					B				C		D	
생명주기 모델		A			B									
테스트 명세 기법		A		B										
정적 테스트 기법					A		B							
시험환경				A				B						C
테스트 기능 및 훈련				A			B				C			
의사소통			A		B							C		
보고		A			B		C					D		
결함관리		A				B		C						

그림 3 테스트 성숙도표 활용 성숙도 평가사례

5. 효율적 테스트 전략

5.1 핵심영역별 만족수준

효율적 테스트 전략을 수립하기 위해서는 현 테스트 프로세스가 도달한 레벨을 분석하고 이를 테스트 성숙도표(TMM)에 제시하며, 제시된 각 핵심영역별 수준을 제어 수준, 효율적 수준, 최적화 수준으로 진화시켜 나갈 수 있도록 해당되는 수준별로 테스트 개선사항을 도출해야 한다.

제어 수준을 만족하기 위해서는 테스트 전략, 정적 테스트 기술, 시험환경, 테스트 기능 및 훈련 등의 핵심영역에서 레벨 A를 만족해야 하고, 생명주기 모델, 테스트명세 기법, 의사소통, 보고, 결함관리 등의 핵심영역에서는 레벨 A 또는 레벨 B를 만족해야 한다. 특히 상위레벨은 하위레벨의 모든 체크포인트를 만족시켜야 한다. 효율적 수준은 테스트 전략 및 테스트 기능 및 훈련 등의 핵심영역에서는 레벨 B 또는 레벨 C를, 정적테스트 기법, 시험환경 등의 핵심영역에서는 레벨 B를, 의사소통, 보고, 결함관리 등의 핵심영역에서는 레벨 C를 만족해야 한다. 또한 최적화 수준은 테스트 전략과 보고 등의 핵심영역에서 레벨 D를 시험환경에서 레벨 C를 만족해야 한다. 이러한 테스트 프로세스 관리수준별 핵심영역 만족수준 전략이 그림 4에 도시되어 있다.

현 상태 제어 효율 최적화

핵심 영역		제어					효율적					최적화		
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
테스트 전략		A					B				C			
생명주기 모델		A			B									
테스트 명세 기법		A		B										
정적 테스트 기법					A		B							
시험환경				A				B						
테스트 기능 및 훈련				A			B				C			
의사소통			A		B							C		
보고		A			B		C							
결함관리		A				B		C						

그림 4 테스트 프로세스 관리수준별 핵심영역 만족수준 전략

5.2 핵심영역별 테스트 전략

5.2.1 테스트 전략

테스트 전략의 현 상태가 기초수준이므로 제어수준을 위해서는 레벨 A를 만족해야 한다. 이를 위한 테스트 전략은 위험에 따른 테스트 레벨을 결정하고 품질특성을 고려한 상대적 중요도를 결정하며 재테스트도 고려해야 한다. 효율적 수준을 위해서는 레벨 B 또는 C를 만족해야 한다. 이를 위해 요구사항 및 위험평가를 바탕으로 테스트 전략을 결정하고 테스트 전략을 품질계획서상에 반영하며 전략이 수정시에는 즉각적인 반영이 되어야 한다. 또한 하위레벨 테스트를 위한 테스트 전략 결정에 개발자를 참여시키고 테스트 레벨에 품질특성을 반영하며 최초 전략에서 벗어난 것에 대해서는 보완해야 한다. 최적화 수준을 위해서는 레벨 D를 만족해야 하는데, 이를 위해서는 상위레벨 테스트와 하위레벨 테스트 및 평가를 혼합해서 실시하고 평가를 위한 테스트 전략을 결정하며 이를 적용해야 한다.

5.2.2 생명주기 모델

생명주기 모델을 제어수준으로 끌어 올리기 위해서는 레벨 A 또는 B를 만족해야 한다. 이를 위해 테스트 전략을 계획, 준비, 명세, 실행, 완료 단계로 구분하여, 준비단계에서는 연관문서를 결정하고 테스트 가능성을 분석하며, 완료단계에서는 미해결 결함을 명시하고 결함추세를 분석하는 등 테스트 객체를 평가하고

테스트웨어를 수집, 수정 및 보존해야 한다. 효율적 수준과 최적화 수준은 테스트 성숙도표에서 정의되지 않아 해당사항이 없다.

5.2.3 테스트명세 기법

테스트명세 기법은 현 상태가 기초수준으로서 제어수준을 위해서는 레벨 A 또는 B를 만족시켜야 한다. 이를 위해 테스트 케이스를 문서화하고 테스트 케이스 작성기준 및 작성 템플릿을 결정해야 한다. 그리고 생명주기 모델과 마찬가지로 테스트명세 기법도 효율적 수준과 최적화 수준에 대해서는 해당사항이 없다.

5.2.4 정적테스트 기법

정적테스트 기법은 현 상태가 레벨 A로서 제어수준을 만족한다. 그리고 효율적 수준을 위해서는 레벨 B를 만족해야 하는데, 이를 위해 산출물별 착안사항을 개발업체와 공유하고 정적테스트 기법에 관한 체크리스트를 작성하여 공유한다. 또한 테스트 실행상태에대한 체크리스트를 작성한다. 최적화 수준은 해당사항이 없다.

5.2.5 시험환경

시험환경도 정적테스트 기법과 마찬가지로 현 상태가 레벨 A로서 제어수준을 만족한다. 효율적 수준을 만족하기 위한 테스트 전략으로서 테스트 레벨에 따른 대표적인 시험환경이 필요하며 시험환경의 독립, 적시적 준비 그리고 타 활동과의 독립이 필요하다. 또한 위험 분석 및 측정값을 반영해야 한다. 최적화 수준은 해당사항이 없다.

5.2.6 테스트 기능 및 훈련

테스트 기능 및 훈련 항목은 현 상태가 기초수준으로 제어수준을 위해서는 레벨 A를 만족해야 한다. 이를 위해 개발자와 독립된 테스트팀을 구성하고 충분히 교육을 받은 테스터를 선정해야 한다. 또한 선정된 테스터는 훈련 및 교육을 시켜야 된다. 효율적 수준을 위해서는 레벨 B 또는 C를 만족해야 한다. 이를 위해 기반구조 관리, 업무할당, 진행사항 관리 그리고 테스트웨어 관리 등의 테스트 프로세스를 문서화해야 하며 이러한 문서화를 위해서는 문서화 활동을 독립하고 이를 위한 시간 확보를 추진해야 한다. 그리고 임무수행을 감독하고 테스트 계획과 동시에 품질보증계획을 수립해야 한다. 또한 품질보증 임무를 독립하고 품질보증 활동 결과를 테스트 프로세스 개선에 반영하며 이러한 일을 수행하는 품질보증 임무수행자는 충분한 지식을 갖추어야 한다. 최적화 수준은 해당사항이 없다.

5.2.7 의사소통

의사소통은 제어수준을 위해 레벨 A와 B를 만족해야 한다. 이를 위해 테스트팀의 회의내용을 문서화하고 테스트 객체의 품질에 관한 변경사항 일정을 반영하며 테스트 진행과정을 보고한다. 또한 제안사항 변경으로 인한 변경에 대한 충격도 관리해야 한다. 효율적 수준을 위해서는 레벨 C를 만족해야 하는데 이를 위해 프로세스와 관련된 회의를 정례화하는 것이 필요하다. 최적화 수준은 해당사항이 없다.

5.2.8 보고

보고는 현 상태가 기초수준으로 레벨 A 또는 B를 만족해야 제어수준이 된다. 이를 위해 발견된 오류를 해결이나 미해결로 분류하고 모든 결함을 보고한다. 효율적 수준을 위해서는 레벨 C를 만족해야 하며, 이를 위해 테스트 객체의 품질을 판정하고 진행과정 및 품질에 관한 정기적인 보고를 실시한다. 또한 위험분석 및 고객의 권고사항도 보고한다. 최적화 수준을 위해서는 레벨 D를 만족해야 하는데 이것은 테스트 외 타 분야에 대한 개선사항을 보고하면 된다.

5.2.9 결함관리

결함관리도 보고와 마찬가지로 기초수준이며 제어수준을 만족하기 위해서는 레벨 A 또는 B를 만족해야 한다. 이를 위해 결함을 심각도 카테고리에 의해 분류하고 테스트 진행활동을 정기적으로 보고한다. 효율적 수준을 위해서는 레벨 C를 만족해야 하며 이것은 결함관리위원회를 조직하고 위원회 관리위원에게 임무를 할당하면 된다. 최적화 수준은 해당사항이 없다.

6. 결 론

본 논문은 TPI 기반 국방 임베디드 소프트웨어 대한 효율적 테스트 전략을 제안하기 위하여 무기체계 임베디드 소프트웨어가 갖고 있는 제반 특성과 테스트 관련 여러 지침들에 대한 분석, 그리고 TPI 모델 기반의 테스트 성숙 모델을 바탕으로 무기체계 임베디드 소프트웨어 테스팅 시 고려해야 될 9가지의 핵심영역을 도출하였다. 도출된 핵심영역에는 테스트 전략, 생명주기 모델, 테스트명세 기법, 정적테스트 기법, 시험환경, 테스트 기능 및 훈련, 의사소통, 보고, 결함관리 등이다. 도출된 9개의 핵심영역 각각에 대한 평가를 위해 그동안 군에서 수행했던 무기체계 시스템에 대한 테스트 활동 중 하나를 선정하여 테스트 프로세스의 성숙도를 평가하였다. 또한 이러한 평가 결과를 바탕으로 9개의 각 핵심영역별로 제어 수준, 효율적 수준, 최적화 수준에 도달하기 위한 효율적인 테스트 전략을 제안하였다.

참고문헌

[1] 국방부, 「무기/비무기체계 내장형 소프트웨어 개발 관리지침」, 2001.12.

[2] Thomas C. Staab, "Can a Testing Maturity Model Help Improve My Testing Process?," The Journal of the Quality Assurance Institute, pp.10-16, 2002.

[3] J.C. Jacobs, J.J.M. Trienkens, "Towards a Metrics Based Verification and Validation Maturity Model," 10th International Workshop on Software Technology and Engineering Practice, pp.1-6, 2002.

[4] Tim Koomen, "Stepwise Imporvement of the Test Process using TPI," ICS Test Conference, 2005.

[5] Wayne D. Woodruff, "Introduction of Test Process Improvement and the Impact on the Organization," SQP, Vol. 5, No. 4, pp.24-32, 2003.

[6] Tim Koomen, "Imptovement of the Test Process using TPI," STEN Journal, Vol. 3, pp.32-38, 2005.

[7] Jari Anderson, "TPI-a model for Test Process Improvement," Seminar on Quality Models for Software Engineering, 2004.

[8] Bart Broekman and Edwin Notenboom, Testing Embedded Software, Addison-Wesley, 2003.

[9] 윤수진, 윤희병, "무기체계 임베디드 소프트웨어 TPI 기반 테스트 프로세스 개선," 한국군사과학기술학회 종합학술대회, 2005.

윤 희 병

1983 해군사관학교(이학사)
1986 연세대학교 (공학사)
1991 미국 해군대학원 전산공학(석사)
1998 미국 조지아공대 전산공학(박사)
2002~현재 국방대 전산정보학과 교수
2004~현재 아주대학교 정보통신대학원 외래교수
2005~현재 국방소프트웨어산학연협회 기획이사
2006~현재 서강대학교 정보통신대학원 외래교수
관심분야 : 임베디드 소프트웨어, 소프트웨어 테스팅, 소프트웨어 아키텍처, 전술데이터링크
E-mail : hbyoon37@hanmail.net