| 특집 03 |

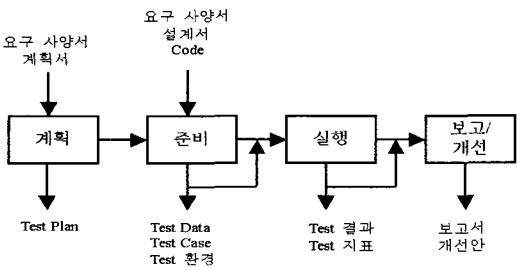
개발 전략에 따른 소프트웨어 시험 계획 수립

김종석
(삼성디지털미디어 연구소)

목 차

1. 서 론

일반적으로 시험 프로세스는 (그림 1)에서 보는 바와 같이 크게 네 단계로 구성되어 있다.



(그림 1) 시험 프로세스

계획 단계에서는 요구 사양서와 계획서를 바탕으로 하여 개발 전략에 따른 시험의 전략 수립과 일정 및 리소스에 대한 계획을 수립하고, 시험을 진행하는데 있어서의 발생 가능한 리스크를 파악하는 작업을 수행한다.

준비 단계에서는 요구 사양서, 설계서, 코드에 기초하여 시험에서 사용될 시험 데이터를 준비하고, 시험 케이스를 설계한다. 또한, 이 단계에서는 시험 수행을 위한 시험 환경을 구성한다. 시험 환경은 단계별 시험 전략에 따라 다르게 구성될 수도 있다.

실행 단계에서는 준비된 시험 케이스를 실행하여, 시험 케이스의 실행 결과를 검증하는 작업을 수행한다. 그리고 발생된 결함들을 Defect tracking 프로세스에 따라 처리하는 작업을 수행한다.

보고 및 개선 단계에서는 시험 수행 결과에 따른 보고서를 작성하고, 시험 수행 시 발생된 문제점을 바탕으로 시험 프로세스를 개선하는 작업을 수행한다.

위에 설명된 시험 프로세스의 단계 중 가장 중요한 단계가 계획 단계다. 나머지 부분들도 중요하지만, 첫 번째 단계에 기초하여 이루어지는 부분으로, 계획 수립이 잘 되지 못하면 후속 단계의 진행도 원활하게 이루어질 수가 없다. 또한, 계획 단계를 통해 개발자와 시험자 간의 업무를 명확히 하고, 시험 관련 정보를 공유함으로써, 효율적으로 시험을 진행할 수 있는 기

반을 확보할 수 있다[6][7].

현재, 국내의 많은 소프트웨어들이 Waterfall 모델에 기반으로 하여 개발이 되고 있는 관계로 Waterfall 모델을 기반으로 소프트웨어를 개발할 시에 많이 사용되는 시험 전략인 V-model [8]을 기반으로 하여 시험 계획을 수립하고 있다. 하지만, 많은 개발자들과 시험er들이 V-model을 적용하여 시험 계획을 수립할 때, 어떤 점들을 고려해야 되고, 어떻게 구체적으로 계획을 수립하고 추진해야 되는지 명확하게 잘 파악하지 못하고 있는 것이 사실이다.

그러므로, 본 논문에서는 그간 경험을 바탕으로, 개발 방법에 따라 소프트웨어 시험 전략을 어떻게 수립하는 것이 효율적인 것인지에 대해 논하고자 한다.

본 논문의 구성은 다음과 같다. 2절에서는 시험 계획 수립에 대한 일반적인 사항에 대해 논의하고, 3절에서는 각 단계별 시험 계획 수립 시 고려되어야 할 부분에 대해 이야기 하고자 한다. 4절에서는 Waterfall 이외의 다른 모델을 개발 방법으로 활용할 시, 시험 계획을 어떻게 수립해야 하는지에 대해 설명하고, 마지막 절에서는 앞에 설명된 내용에 대해 정리하였다.

2. 시험 계획

시험 프로세스의 첫 번째 계획 단계의 산출물은 시험 계획으로, 일반적으로 시험 계획은 아래 (그림 2)와 같은 내용을 포함한다[3][4].

(그림 2)의 항목을 간략하게 살펴보면, 개요 부분에서는 문서 목적을 포함한 일반적인 사항을 이야기한다.

시험 전략에서는 시험을 어떻게 구성하고 진행 할 것인지에 대한 내용을 기술하고, 이번 시험에서 시험이 수행될 대상과 제외할 대상을 식별해야 된다.

시험 방법에 대해서는, 수립된 전략에 대해

- ▶ 개요
  - – 목적
  - – 범위
  - – 용어 및 약어 정의
  - – 문서의 구조
- ▶ Test 전략
  - – Test 범위, 수행될 Test 종류, 모니터링 항목 선정, 관련 산출물
  - – Test 대상 항목
  - – Test 제외 항목
- ▶ Test 방법
- ▶ Test 기준
  - – Test Pass/Fail 기준
  - – Test 시작/완료 기준
  - – Test 중지/재개 기준
- ▶ Test 환경
  - – Test 환경 구성 및 자동화 도구 적용 방법
  - – Test data
- ▶ Test 일정 및 역할
  - – Task 및 일정
  - – 책임과 역할
  - – 교육 계획
- ▶ Test 관리
  - – Test 진척 관리(Test 결과 승인 절차 포함)
  - – 결함 관리
  - – 변경 관리
- ▶ 산출물
- ▶ Risks

(그림 2) 시험 계획

구체적인 시험 방법을 제시해야 된다. 예를 들어, White box/Black box 시험 기법 등을 어떻게 활용할 것인지 단계별 시험 전략에 따라 기획하고, 자동화 추진 시, 자동화는 어떻게 추진할 것인지를 방법적인 면에서 기술하여야 한다.

시험 기준 항목 기술에 있어서는, 시험 시작 및 완료 조건을 명확히 하여야 한다. 시험 시작 조건이 명확하지 못하면, 계획된 시험을 원활하게 수행하기 어렵다. 일반적으로 시험 시작 조건으로는 관련 코드의 완성도, 시험 대상에 관련 된 모듈의 유효성, 설계와 코드의 일관성 등을 사용할 수 있다. 또한 완료 조건을 명확히 하여, 시험 종료 시 시스템의 상태를 명확히 알 수 있도록 하여야 한다. 시험 종료 조건으로는 일반적으로 커버리지[1]나 신뢰성 모델 [5]을 이용할 수 있다.

시험 환경 구성은 시험 단계에 따라 시험을

어떤 환경에서 실시할 것인지 구체적으로 명시하여야 한다. 예를 들어 사용될 하드웨어에 대해서는 CPU 타입 및 속도, 메모리 관련 사항, 기타 I/O 디바이스에 대한 사항들을 기술하고, 사용되는 소프트웨어 툴에 대해서는 소프트웨어 이름 및 버전 등을 기술하여야 된다. 일반적으로 단위(unit) 시험, 통합(integration) 시험에 대해서는 개발 환경을 시험 환경으로 사용하고, 시스템 시험에 대해서는 사용자 환경을 시험 환경으로 사용한다. 이외에, 자동화 툴 및 측정 장비들을 포함할 수 있다.

시험 항목 중에서 고려되어야 할 부분이 시험 데이터에 대한 부분이다. 시험 데이터는 일반적인 경우뿐만 아니라 예외적인 경우도 고려하여 준비해야 되는데, 시험 계획에서 사용될 시험 데이터에 대한 사항과 확보 전략을 명시해야 된다. 특히, 단위 시험과 통합 시험에 사용되는 시험 데이터는 처리 중간 단계의 데이터인 경우가 많으므로, 확보 전략을 잘 수립하지 않으면 안 된다.

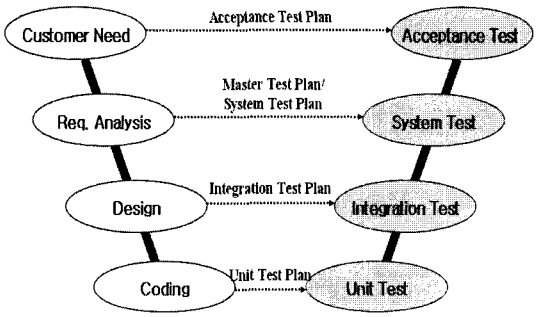
시험 수행 일정은 개발 일정과 연동하여 수립하되, 투입 리소스와 시험 종료조건을 달성할 수 있는 기간을 고려하여 수립하여야 한다. 국내에서 개발되는 많은 소프트웨어들이 개발 일정의 지연으로 충분한 시험을 실시하지 못하고 있다. 그러므로, 일정 계획을 수립할 시에는 개발과 충분한 협의를 통해 이행 가능한 일정을 수립하여야 하고, 필요 시에는 일정에 대한 리스크를 관리할 필요가 있다.

3. 단계별 시험 계획 시 고려 사항

앞 절에서 시험 계획에 대해 일반적인 사항에 대해 설명하였는데, 이 절에서는 각 단계별 시험 계획 수립 시 고려되어야 될 점에 대해서 이야기 하고자 한다. Waterfall을 개발 모델로 활용할 경우, V-model을 이용하여 시험을 수행하게 되는데, 시험 계획은 (그림 3)에서 보는 바와 같이 V-model에서는 아래와 같이 단계별 시험 계획을 수립하도록 된다.

- Acceptance 시험 계획
- Master 시험 계획
- System 시험 계획
- Integration 시험 계획
- Unit 시험 계획

이장에서는. 위의 다섯 가지 계획 중, 고객이 작성하는 Acceptance 시험 계획을 제외한 네 가지의 시험 계획에 대해 논하기로 한다.



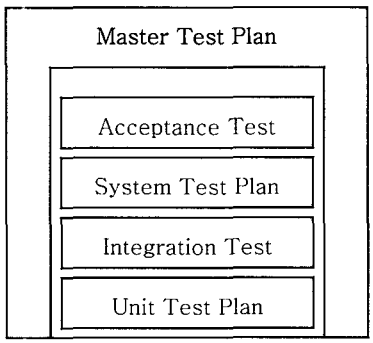
(그림 3) V-모델

3.1 마스터 시험 계획(Master Test Plan)

마스터 시험 계획은 시험 전략 수립의 근간이 되는 계획으로 개발 전 단계 시험의 기초가 된다. MTP는 일반 적으로 요구 분석 단계에서 작성하며, 아래 그림에서 보이는 것과 같이 단계별 시험 계획의 상위 문서로 단계별 시험 계획의 기초가 된다[2].

마스터 시험 계획 작성의 목적은 개발 초기에 전반적인 시험 계획을 수립하여, 개발자와 공유 함으로써, 개발이 시험 전략과 일치되게 수행 될 수 있도록 함과 동시에, 시험자들이 개발 전 과정에 걸쳐서 무엇을 해야될지를 알려주는 지침서가 된다. 즉 마스터 시험 계획은 개발과 시험 간의 협약서와 같은 역할을 한다. 개발은 마스터 시험 계획에서 제시된 시험 전략 및

일정 등을 고려하여 개발을 진행하고, 시험은 마스터 시험 계획에 기초하여 개발된 시스템에 대해 시험을 실시한다.



(그림 4) 마스터 시험 계획

마스터 시험 계획 작성 시 가장 고려되야 할 부분은, 시스템 개발 과정에서 수행되는 시험에 대한 기본적인 시험 전략 및 방법 수립과 자동화 전략 수립 부분이다. 시험 전략 및 방법 수립 시에는 개발자와 협의를 통해 과제의 성격 및 개발이 어떤 형태로 진행되는지를 파악하고 그에 맞는 계획을 수립하도록 한다.

마스터 시험 계획에서 기본 전략과 방법 수립은 구체적일 필요는 없으나, 수행될 시험에 대한 방향이 제시되어야 한다. 예를 들어,

- 단계별 시험은 어떻게 가져 갈 것인가?
- 각 단계별 시험의 중점 목표는 어떻게 할 거인가?
- 단계별 시험의 일정은 어떻게 할 것인가?
- 단계별로 수집할 지표는 어떤 것이 있는가?

각 부분의 구체적인 내용들은 마스터 시험 계획에 기초하여, 각 단계별 시험 계획에 기술하도록 한다.

시험 자동화 부분도 마스터 시험 계획 작성 시 고려해야 사항 중에 하나로, 자동화 도입 시 어떤 부분에 어떻게 활용할 것인지를 명확히 하여야 한다.

예를 들어 자동화를 추진할 경우, 단계별 자동화 방법 및 적용 범위, 수행 주체가 누구인지를 명시하고, 준비 계획을 제시하여야 한다. 또한 툴을 도입할 것인지, 아니면, 시험 툴을 개발할 것인지를 명시하고, 그 실행 계획도 제시하여야 한다. 특히, 툴을 개발을 하는 경우에는 시스템 개발과 같이 추진되어야 하므로, 개발자와 협업에 대해서도 명시하여야 한다.

마스터 시험 계획이 작성되면 개발자들과 공유하여 시험이 계획에 따라 실행이 될 수 있도록 해야 한다. 또한 개발자 측에서도 시험 실행 계획의 방법 및 일정 등이 타당한지에 대한 검토를 반드시 실시하고, 두 팀 간의 상호 협력하에 시험이 수행될 수 있도록 하여야 한다.

3.2 시스템 시험 계획(System Test Plan)

시스템 시험 계획은 요구 분석 단계에서 마스터 시험 계획에 기초하여 작성 된다. 시스템 시험 계획을 작성 할 때 고려해야 될 사항 중에 하나는 Build management이다. 시스템 시험의 경우에는 전체 코드에 대한 시험을 실시하는 것으로, 전체 개발자가 영향을 받기 때문에 다른 단계 시험에 비해 Build management가 더욱 중요하다. Build management가 잘 되지 않는 상태에서 시험이 진행될 경우, 시험자와 개발자 간의 오해 소지가 생기고, 불필요한 작업이 발생하는 경우가 많다.

예를 들어, 시험자와 개발자가 보고 있는 소스 코드 버전이 다름으로 인해 결함 발생 및 수정에 대한 오해가 발생하기도 하고, 개발 단계에서도 개발자 간에 다른 버전을 가지고 개발함에 따라 결함이 발생하기도 한다.

Build management는 일반적으로 SQA나 시험 파트에서 개발자와 협의하여 진행하도록 하고, 하나의 시험 시작 조건으로 활용하도록 한다. 시험 계획에 코드 Build 프로세스에 대해 정의하여 프로세스에 따라 코드 Build가 이루어지고, Build가 성공적으로 되었을 때 시험을 실행하도록 한다.

또한, 시스템 시험 계획을 수립할 때 고려할 중요 사항 중에 하나는 시험 환경 구축이다. 일반적으로 단위 시험과 통합 시험은 개발 환경에서 실시되므로 큰 문제가 없지만, 시스템 시험의 경우 사용자 환경에서 실시되는 관계로 시험 환경 설정에 어려움이 많다. 특히, 비기능 시험의 환경 구성은 실제 실행을 위한 환경뿐만 아니라, 비기능 요구사양 측정을 위한 환경도 같이 구성해야 된다. 측정 장비를 활용할 시에는 측정장비가 실제 시스템에 영향을 주지 않도록 해야 한다. 그리고, 호환성 시험을 위한 환경을 구성할 때에는, 모든 구성을 갖추기 힘든 관계로 주요 타겟 중심으로 다양한 조합으로 시험을 실시할 수 있게 계획을 세우고 그에 따른 리스크가 어떤 것이 있는지를 식별하여야 한다. 그리고 시험 자동화 추진 시, 자동화에 대한 환경도 식별하여야 한다.

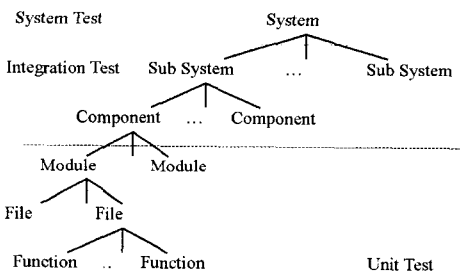
3.3. 통합 시험 계획(Integration Test Plan)

통합 시험 계획은 디자인 단계에서 마스터 시험 계획에 기초하여 작성하되, 시스템 시험과 연관성도 고려하여 작성한다. 통합 시험 계획 작성에 있어서 가장 중요한 부분은 통합 전략을 세우는 일이다. 통합 전략은 크게 다음과 같이 네 가지로 분류된다.

- Top-Down
- Bottom-Up
- Big Bang
- Sandwich

통합 전략은 개발 전략과 밀접한 관계를 가지는데, Window application과 같이 사용자 인터페이스 중심의 소프트웨어를 개발할 경우에는 인터페이스로부터 실제 실행부까지 통합해 가는 top-down 방식의 전략이 일반적으로 효율적이고, 많은 연산이 필요한 소프트웨어 개발에서는 실행부부터 검증하여 상위로 올라가는 Bottom-up 방식이 효율적일 수 있다. 임베디드 시스템의 경우에는 위와 아래에서 동시에 통합해가는 샌드위치 방식이나, 인터페이스가 단순하고, 데이터 처리가 많은 경우에는 Bottom-up 방식도 사용할 수 있다. 일반적으로 Big bang 방식은 Defect tracking에 어려움이 많은 관계로 지양해야 할 방법이지만, 기존의 레거시 코드를 활용하여 개발하여, 서브시스템의 경계를 결정하기 힘든 경우에는 활용할 수 있다. 단, 일정 관계로 인해 Big bang을 시도하는 것은 많은 위험을 내포하고 있는 관계로 지양하도록 하여야 할 것이다.

통합 전략이 확정 되면, Big bang의 경우를 제외하고는, 몇 단계에 걸쳐 통합을 할 것인지를 계획을 하여야 한다. 예를 들어, (그림 5)과 같이 시스템이 구성되어 있는 경우,



(그림 5) 시스템 구성

일반적으로는 모듈 이하 부분에 대해서는 단위 시험에서 검증을 하고 컴포넌트로부터 서브 시스템까지는 통합 시험에서 검증을 한다. 시스템이 위와 같이 구성된 경우에는 주로 2단계의 통합을 실시하는 것이 바람직하지만, 서브 시스템의 규모와 서브 시스템 내부의 구조를 고려하여 여러 단계로 나누거나 또는 통합하여 실시할 수 있다.

통합 전략이 수립 되면, 개발자와 공유하여 개발의 우선순위를 조정하여 일정에 맞게 시험이 수행할 수 있도록 계획을 수립하여야 한다.

통합 시험을 잘 수행하기 위해서는 개발의 협조가 필요하다. V-model에서 보는 바와 같이 통합 시험은 디자인에 기초하여 실시된다. 그러므로, 효율적인 통합 시험을 위해서는 아래의

사항들이 고려되어야 한다.

- Code와 Design의 일관성 확보
- System의 hierarchy에 대한 정의
- Subsystem/component의 boundary에 대한 정의
- 공용 component에 대한 정의

위의 사항들이 잘 되어 있지 않을 경우, 체계적인 통합 시험 진행이 어렵고, 통합 시험을 Big bang 시험으로 실시할 수밖에 없게 된다.

3.4 단위 시험 계획(Unit Test Plan)

단위 시험 계획은 이론적으로는, 상세 설계 시 작성하는 것으로 되어있지만, 국내에서 많은 개발과제들이 상세설계에서 Function에 대한 Spec을 작성하지 않는 관계로 Function spec이 확보 되는 구현 초기에 작성하는 것이 효율적이다. 단위 시험 계획 작성 시 가장 먼저 고려되어야 할 사항은 Unit에 대한 정의이다. Unit의 단위는 보는 시각에 따라 (그림 5)에서 보는 바와 같이 작게는 Function 단위에서 크게는 파일 여러 개가 모인 모듈 단위까지 될 수가 있다.

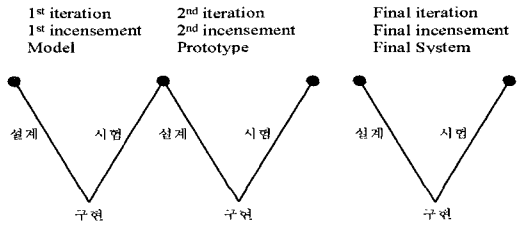
Unit의 단위를 결정하는 것은 과제의 성격에 따른다. 예를 들어 현재 추진되는 과제가 처음부터 모든 것을 만드는 것이라면, Unit의 단위를 Function으로 가져감으로서 Function level의 코드 검정을 실시하는 것이 바람직하다. 하지만, 현재의 과제가 이전 개발된 컴포넌트를 중심으로 추진되는 것이라면, Function level 보다는 조금 큰 사이즈로 Unit을 가져가는 것도 무방하다. Unit의 사이즈를 Function으로 가져갈 경우에는 초기 통합에 해당되는 Module level까지는 Unit 시험에서 검증하도록 하는 것이 바람직하다.

Unit 단위가 결정되면, 그에 따라 시험 방법에 대한 계획을 수립할 수 있다. 만약에 Unit을 Function level로 가져간다면, 가능한 White box 시험 방법을 추진하여 모든 코드를 검증하도록 하는 것이 좋다. 즉, 코드 커버리지를 100%로 목표를 설정하도록 한다. 실제 Function level로 시험을 실시할 시 하드웨어에 대한 종속성이 큰 경우 실제 Dynamic 시험을 실시하기에 힘든 경우도 있다. 이런 경우에도 가능한 Stub을 활용하여 시험을 진행 하도록 계획을 수립하고, 여의치 못한 경우 코드 Inspection 등의 Static 시험을 수행하여 코드를 검증하도록 한다.

4. 개발 모델 별 고려 사항

Spiral, Incremental, Prototype 등과 같이 Waterfall 모델이 아닌 다른 개발 모델을 바탕으로 시스템을 개발하는 경우에는 일반적인 V-model에 기초하여 시험 계획을 수립하기가 어렵다. 이 절에서는 Waterfall 이외의 모델을 개발 모델로 사용할 경우에 고려되어야 할 점에 대해 기술 하였다.

Spiral, Incremental, Prototype 등을 개발 모델로 활용하는 경우에는 (그림 6)과 같이 Multiple V-model[2]을 바탕으로 하여 시험 계획을 수립하는 것이 효율적일 수 있다.



(그림 6) Multiple V-모델

Spiral, Incremental, Prototype 모델에서는 개발 모델의 특성상, 여러 번의 반복적 개발이 이루어지는데, 이 경우에 대응하여, 첫 번째 반복 시점에서 기초적인 계획을 세우고, 차수가 진행됨에 따라 차수 별로 초기 계획을 바탕으로 시험 계획을 구체화해가야 된다.

Multiple V-model을 활용할 경우, 몇 번의 반복이 필요할지는 개발 전략에 따라 결정된다.

이 경우 첫 번째 단계에서 계획을 세우는데 가장 신경을 써야 할 부분이 차수 별 시험 대상 및 제외 대상을 설정하는 것이다. 물론, 차수가 올라감에 따라 계획을 수정할 수 있지만, 초기에 시험 대상을 잘 설정해 두지 않으면, 불필요한 작업을 할 소지가 많다.

Spiral, Prototype 모델이 Incremental 모델과의 차이점은, 최종 시스템 시험이 Spiral, Prototype 모델의 경우에는 마지막 V-model에서 실시하고 이전의 V-model에 대해서는 Prototype에 대한 시스템 시험을 실시한다. 하지만, Incremental 모델에서는 각 단계마다 release를 할 수 있으므로, 각 V-model에서 모두 시스템 시험을 실시하는 점이 다르다. 그러므로 시험 계획을 수립할 시 이를 고려해서 하여야 한다.

5. 결론 및 향후 계획

본 글에서는 단계별 시험 계획을 수립하는데 있어서 고려해야 될 점과 개발 모델에 따라 고려해야 될 점에 대해 간략하게 기술하였다.

앞에서 여러 가지 사항을 제시하였지만, 시험 계획 수립에 있어서 가장 중요한 것은 개발자와 협의를 통해 시험 계획을 수립하는 것이다. 시험이라는 업무자체가 개발에서 이어지는 일이므로, 개발과 유기적으로 협업하지 않으면 성공적으로 진행될 수 없다. 그러므로 시험 계획을 수립할 시에 개발 전략에 맞게 계획을 세워야 되고, 그 내용들이 충분히 검토되고 공유되어야 한다. 그리고, 개발에서도 시험이 시스템의 품질을 확보하는 마지막 단계라는 인식하에 시험이 계획된 대로 수행될 수 있도록 개발을 추진하여야 한다.

추후로 시험 계획을 수립할 시, 성공적인 시험 수행을 위해서는 앞에서 제시된 내용뿐만 아니라, 과제 특성과 개발 전략을 고려하여 계획에 반영될 수 있도록 하여야 할 것이다.

참고문헌

[1] Marick, Brian, *The Craft of 소프트웨어 시험ing*, Prentice Hall, 1995.

[2] Broekman, Bart and Notenboom, Edwin, *시험ing Embedded 소프트웨어*, Addison-Wesly, 2003.

[3] DOD-STD-498, *Military Standard 소프트웨어 Development and Documentation.*

[4] IEEE Std. 829-1998, *IEEE Standard for 소프트웨어 시험 Documentation.*

[5] Lyu, Michael R., *Handbook of 소프트웨어 Reliability Engineering*, Mcgraw-Hill, 1996.

[6] Black, Rex, Managing the 시험 프로세스 2nd Ed., Wiley Publishing Inc., 2002.

[7] Black, Rex, *Critical 시험ing 프로세스, 계획, Prepare, Perform, Perfect*, Addison-Wesly, 2004.

[8] Watkins, John, *시험ing IT, An Off-the-Shelf 소프트웨어 시험ing 프로세스*, Cambridge University Press, 2001.

저자약력

김 종 석

1990년 숭실대학교 전산학 학사

1995년 Florida Institute of Technology 전산학 석사-소프트웨어 Engineering

2003년 Florida Institute of Technology 전산학 박사-소프트웨어 시험

2003년~현재 삼성전자 DM 연구소 책임 연구원

관심분야: S/W 시험 프로세스, S/W 시험 Automation, S/W Reliability