[신규도입 컨설팅]
테스트기법과 방법론

한국소프트웨어진흥원
공개SW기술지원센터

<Revision 정보>

일자	VERSION	변경내역	작성자
2007. 5. 02	0.1	초기 작성	김용규
2007. 5. 14	0.2	목차 수정	김용규

목 차

1. 문서 개요

본 문서는 테스트 기법및 방법론에 관한 자료로 추후 솔루션 테스트시 다양한 테스트 케이스에 대한 기법과 표준화 방향을 제시하였으며 이에 대한 신규도입 컨설팅 참고자료로 활용하기 위해 제작되었다.

가. 문서의 목적

다음과 같은 세부적인 목적을 달성하기 위하여 작성되었다.

0 다양한 테스트 케이스의 도출

0 테스트 기법 및 실무사례

0 국내/외 테스트 표준화 기구 및 동향

2. SW 테스트 개념

가. 테스트의 정의

소프트웨어 테스트는 프로그램에 있는 오류를 찾기 위해 프로그램을 실행하여 실행결과와 예상결과를 비교하는것

나. 테스트의 목적

테스트를 통해 프로그램에 포함된 오류를 검출하는 것

다. 테스트와 확인(Validation, 검사)

테스트는 프로그램이나 시스템이 오류가 없음을 증명하지 않고 테스트 조건과 유사한 환경에서 오류가 없음을 간접적으로 <표현하는 것이며 확인(Validation, 검사)은 프로그램이 적절히 실행됨을 검사하는 과정이다.

라. 테스터의 9가지 역할

1) 테스트 관리자(Test Manager)
2) 테스트 리더(Test Leader)
3) 사용성 테스트 기사(Usablility Test Engineer)
4) 수동 테스트 기사(Manual Test Engineer)
5) 자동화 테스트 기사(Automated Test Engineer)
6) 네트워크 테스트 기사(Network Test Engineer)
7) 테스트 환경 전문가(Test Environment Specialist)
8) 보안 테스트 기사(Security Test Engineer)
9) 테스트 라이브러리와 형상관리 전문가(Test Library & Configuration Specialist)

마. 소프트웨어 오류

1) 오류의 공통적인 정의
   가) 프로그램과 명세 사이의 불일치는 프로그램 내부의 오류
2) 오류의 추가적인 정의
   가) 소프트웨어 오류는 최종 사용자가 프로그램이 수행할 것이라고 예상했던 것을 프로그램이 수행하지 않을 때 존재
   나) 버그에 대한 명백한 정의나 버그의 존재에 대한 확실한 판단은 어려움(프로그램 오류인지 아닌지는 사람의 판단에 의존함)
3) 오류의 13가지 종류
   가) 사용자 인터페이스 오류
   나) 오류처리 오류
   다) 경계 관련 오류

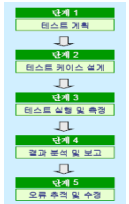
   라) 계산 오류
   마) 처음과 나중 상태 오류
   바) 제어흐름 오류
   사) 데이터의 처리나 해석 오류
   아) 레이스 조건 오류
   자) 부하 조건 오류
   차) 하드웨어 오류
   카) 소스와 버전 제어 오류
   타) 문서와 오류
   파) 테스트 오류

3. 테스트 프로세스

가. 테스트 프로세스의 중요성

1) 효율적인 테스트 인력관리
2) 체계적인 테스트 케이스 발굴
3) 시스템의 문제를 발견
4) 효율적인 보고 지원

나. 테스트 프로세스



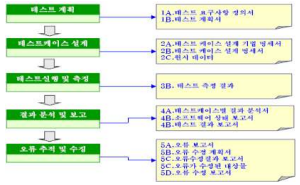
<그림 1 테스트 프로세스>

1) 테스트 계획(Test Planning)
   가) 테스트 요구사항 수집

   나) 테스트 계획 작성
   다) 테스트 계획 검토
2) 테스트 케이스 설계(Test Case Design)
   가) 테스트 설계 기법 정의
   나) 테스트 케이스 도출
   다) 원시 데이터 수집
3) 테스트 실행 및 측정(Test Execution and Measurement)
   가) 테스트 환경 구축
   나) 테스트 케이스 실행 및 측정
4) 결과 분석 및 보고(Result Analysis and Report)
   가) 측정 결과 분석
   나) 테스트 결과 보고
5) 오류 추적 및 수정(Error Tracking and Correction)
   가) Causal Effect 분석
   나) 오류 수정 계획
   다) 오류 수정
   라) 수정 후 검토

다. 산출물 검증

1) 단계별 산출물 및 흐름

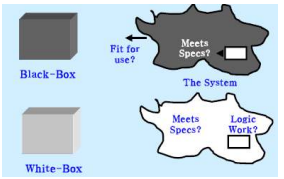


<그림 2 단계별 산출물 및 흐름>

4. 테스트기법

가. 개요 및 분류

1) 단계별 산출물 및 흐름테스트 기법의 필요성
2) 모든 테스트를 수행하는 것은 불가능
3) Error를 찾는 효과적이고 효율적인 test가 요구됨
   가) 효과적인 테스트 : 특정 error를 찾을 수 있는 테스트
   나) 효율적인 테스트 : Error를 찾을 기회가 가장 많은 테스트
4) 적절히 사용된다면 test case 설계는 기계적인 프로세스가 될수 있음
   즉, 동일한 기법을 사용하면 동일한 test case가 나옴
5) 관심의 분리가 가능
6) 테스트 기법의 분류
7) SWEBOK* Ch.5 SW Testing에 기반한 분류
   가) Based on tester's intuition and experience
   나) Specification-based
   다) Code-based
   라) Fault-based
   마) Usage-based
   바) Based on Nature of Application
   사) Selecting and combining Techniques
   * Guide to the SW Eng. Body of Knowledge. 2004 Version
8) Black Box & White Box 기법



<그림 3 Black Box & White Box기법>

*** Black-box testing = Specification-based Testing**
= Input/output testing = Functional testing

가) Black-Box Testing
(1) 프로그램이 어떻게 구현되어 있는지 관여하지 않고 의도된 대로 동작 확인
(2) 요구사항/명세에 기반하여 테스팅
(3) 요구사항으로부터 테스트 기준을 도출함
(4) 요구사항 문서가 없을 경우 시스템의 행위를 관찰함으로써 요구사항을 해석
(5) 주로 발견하는 error들
(가) 부정확한 기능 및 빠진 기능
(나) 인터페이스 오류(다른 기능과의 인터페이스, 데이터와의 인터페이스,
(다) 다른 시스템과의 인터페이스)
(라) 행위 및 성능 오류
(마) 초기화 및 종료 에러

*** White Box = Glass-Box testing**

나) White Box Testing
(1) 프로그램 코드의 내부 구조, 로직에 기반하여 테스트
(가) Structure를 통해 Control Flow를 알아내어 test case를 도출함
(예: if-then-else문)
(나) 테스터가 내부 구조를 볼 수 있고, 이해함
(다) 그렇지만 요구사항/명세가 필요함

다) 장점
(1) 특정 부분에 대한 집중적 테스트 가능
(2) 다양한 커버리지를 테스트함으로써 오류 발견율을 높일 수 있음
(3) 테스트 경로 흐름을 제어할 수 있음
(4) 테스트중 데이터의 값을 예상할 수 있음

나. 직관과 경험 기반의 테스트 기법

1) Ad hoc Test
가) 전문가의 스킬, 직관, 경험의존
나) 정형화된 기법으로 유도되지 않는 특별한 테스트를 식별할 수 있음
2) Exploratory Test
가) 미리 계획을 세우지 않음(dynamic)
(1) "동시 과학 → 테스트 설계 → 테스트 수행"
나) SW 엔지니어의 지식에 따라 효과성이 결정
(1) 테스팅 중 관찰된 product의 행위
(2) 지식의 예
(가) 어플리케이션, 플랫폼, 고장프로세스, 가능한 결함 및 고장들의 유형,
(나) 특정 product와 관련된 위험 등에 대해 익숙한 정도

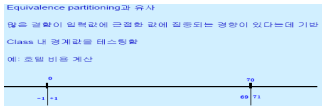
다. 명세 기반 테스트 기법

1) 동등분할(Equivalence partitioning)
가) 여러 테스트 케이스 중에 실제로 수행할 대<표적인 테스트 케이스를 선택하는 방법
나) 동등 클래스(Equivalence class) : 프로그램이 테스트 될 때 동일하게 취급되는 입력집합
다) 각 개별 partition은 동등함(equivalent) : Partition 내 하나의 입력 테스트는 나머지를 모두 테스트한 것과 같음
라) 예 : 호텔비용 계산(최소$0 초과, 최대 $70 이하)
(1) Test cases

Test Case ID	호텔 비용	테스트할 partition	예상 결과값
1	50	0<비용<=70	OK
2	-25	비용 <= 0	에러
3	89	비용 > 0	에러

<그림 4 동등분할>

2) 경계값 분석(Boundary value analysis)
가) Equivalence partioning과 유사
나) 많은 결함이 입력값에 근접한 값에 집중되는 경향이 있다는데 기반
다) Class 내 경계값을 테스팅 함
라) 예 : 호텔 비용 계산
(1) Test cases

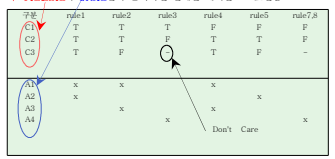


<그림 5 경계값 분석>

Test Case ID	호텔 비용	테스트할 boundary	예상 결과값
1	-1	0	에러
2	0		에러
3	1		OK
4	69	70	OK
5	70		OK
6	71		에러

<그림 6 경계값 분석 Test Case>

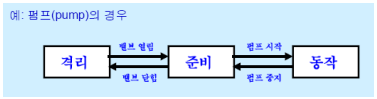
마) 경계 조건의 유형
(1) 첫 번째 / 마지막, 시작 / 끝, 비어 있는 / 가득 찬, 가장 느린 / 가장 빠른, 가장 큰 / 가장 작은, 처음 항목 / 마지막 항목, 최소 / 최대, 위 / 아래, 최단 / 최장, 가장 높은 / 가장 낮은
(2) 동등분할을 통해 데이터를 선택한 다음 경계 부분의 데이터를 선택하는 것이 효과적
3) 의사결정 테이블(Decision table)
*** Condition**과 **action**간의 논리적인 관계를 테이블로 <표현함



<표 1 의사결정 테이블>

가) Limited entry decision table
나) 모든 condition이 binary value만 가짐
다) Extended entry decision table
라) condition이 2개 이상의 값을 가질 수 있음

4) 유한 상태 기계 기반(Finite State Machine based testing)



<그림 7 유한 상태 기계 기반>

가) 상태 및 상태 사이의 전환 과정을 테스트
(1) Positive testing
(가) 밸브를 열었을때 "격리"->"준비"
(나) 밸브를 닫았을때 "준비"->"격리"
(다) 시작 버튼을 눌렀을 때 "준비"->"동작"
(라) 중지 버튼을 눌렀을 때 "동작"->"준비"
(2) Negative testing
(가) "동작"일 때 밸브를 닫으면 ?
(나) "격리"일 때 시작 버튼을 누르면 ?
(다) "동작"일 때 시작 버튼을 누르면 ?
(라) "동작"일 때 시작과 중지 버튼을 동시에 누르면 ?

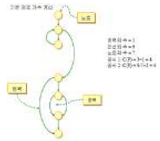
5) 정형명세 테스트(Formal specification testing)
가) 일반 테스트의 한계
오류가 없음을 증명하지는 못함
나) 정형명세(formal specification)
오류가 없음을 증명, 자료 중심 시스템보다는 제어 중심 시스템에 적합
다) Model-based specification
(1) 정형 명세 언어(Z, VDM) 활용
(2) 시스템의 모든 도달 가능한 State space를 탐색하여 deadlock의 가능성을 검사
라) Algebraic specification
SW 연산을 만족하는 등식의 집합으로 구성하여 점검

6) 무작위 테스트(Random testing)
가) 입력 도메인 내에서 무작위로 테스트 케이스 생성
SW 신뢰성 모델과 예측을 사전에 할 수 있음
나) 취약점
(1) 중요하지 않은 테스트 케이스만 생성할 수 있음

(2) 커버리지에 대한 보증이 없음
(3) 테스트 케이스가 정상적인 경우로 치우칠 수 있음
(4) 사용자의 통계적 특징을 예측하는 이론적 기반이 없음
(5) 임의 테스트(arbitrary testing)
(6) 출력 예상이 어려움
다) 다른 기법과 보완하여 사용

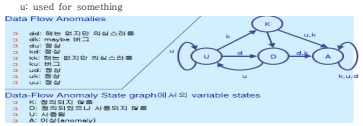
라. 코드 기반 테스트 기법

1) 제어 흐름 기반 범주(Control flow based criteria)
가) 테스트 케이스를 원시코드의 논리 흐름을 근거로 선택
나) 기본 테스트 경로의 수(C) = McCabe의 complexity
→공식 1: C = 영역(region)의수+1
→공식 2: C = 간선의수-노드의수+2



<그림 8 기본 경로의 수>

2) 데이터 흐름 기반 범주(Data flow-based criteria)
가) Motivation(Rapps and Weyuker)
"One would not feel confident about a program without having seen the effect of using value produced by each and every computation"
나) 프로그램 내 변수들 정보로서 flowgraph 작성
d: defined, created, initialized, etc
k: killed, undefined, released
u: used for something



<그림 9 데이터 흐름기반 범주>

마. 결함 기반 테스트 기법

1) 테스터의 직관 또는 경험에 의해 error 또는 defect를 발견하는 능력
가) 시스템의 설계 방법 및 구현 기술에 관한 지식
나) 이전 테스팅 단계 수행 결과에 대한 지식
다) 유사한 시스템 테스팅 경험
라) 전형적인 구현 에러에 관한 지식
예: divide by
2) 효과적이고 효율적인 테스팅이 될 수 있도록 잘 개발할 필요가 있음

3) 뮤테이션 테스트
가) 기본 가정
(1) Competent programmer hypothesis
(2) 유능한 프로그래머는 거의 오류가 없다.
(3) Coupling effect
간단한 결함을 검출할 수 있는 테스트 케이스 집합은 복잡한 결함도 검출할 수 있다.
나) Mutant 생성으로 시작함
테스트 대상 프로그램의 수정된(결함을 포함한) 버전
예) if X<Y if X<= Y
다) Mutant testing
(1) 테스트 케이스를 원본과 mutant에 대해 실행
(2) 원본과 mutant 실행 결과가 차이가 날 경우 mutant를 삭제
라) 좋은 테스트 케이스는 mutant를 가능한 많이 구분할 수 있는 것을 의미함

바. 사용 기반 테스트 기법

1) 신뢰성 평가를 위한 테스트에서의 테스트 환경은 SW 운용환경과 가능한 한 유사하도록 운용 환경을 제생하여야 함
2) 기본 idea는 테스트 결과로부터 SW가 실제 사용되었을 때 향후 SW의 신뢰성을 유추할수 있다는 것
3) 테스트를 향후 사용 패턴에 근거하여 확률분포에 의해 수행

VCR의 예	evStop	evRewind	evPlay	evFastForward	evRecord
Standby	0	0.2	0.5	0.2	0.1
Rewind	0.3	0	0.6	0.1	0
Play	0.5	0.3	0	0.2	0
FF	0.5	0.1	0.4	0	0
Record	1	0	0	0	0

event
state
확률

<그림 10 사용기반 테스트 기법>

사. 어플리케이션 특성 기반 테스트 기법

1) SW Reliability Engineered Testing(SRET)
모든 개발 프로세스를 포함,신뢰성 목<표와 서로 다른 기능들의 상대적 예상 사용도 및 중요도에 의해 테스트가 설계되고 진행됨
2) 참여자
테스터 외 시스템전문가, 시스템 아키텍트, 품질/신뢰성 전문가, 프로젝트 관리자 등
3) 기본 활동
신뢰성 정의 Operational Profile 개발 테스트 준비 테스트 실행 실패 데이터 해석

아. 테스트 기법의 선택과 조합

1) 두가지 기법은 베타적이 아니라 상호보완적으로 활용이 되어야 함
가) 서로 다른 정보를 사용함
나) 서로 다른 문제를 부각시켜 줌
2) 기능 테스트
가) 사용자 관점
나) 기능성과 특정에 관심이 있을 경우
3) 구조적 테스트
가) 구현자 관점
나) 프로그램의 상세한 구현에 관심이 있을 경우

자. 테스트 전략

1) 프로젝트 과정 전반에 걸쳐 테스트 설계를 이끌어가는 아이디어(선택사항) 집합
2) 가능한 테스트 전략은 매우 많음(간략 vs. 엄격, 깊이 vs. 넓이)
3) 자신의 상황에 적절한 테스트 계획 수립 : 상황만족 모델

차. 기타 테스트 기법

1) 테스트 기법 구분하는 5가지 요인
가) Tester : 누가 테스트를 수행하는가?(사용자/알파/베타/전문가/팩 테스트)
나) 범위 : 어떤 측면을 테스트 하는가?(함수,기능 or 함수통합, 메뉴, 도메인, 클래스, 경계값, 입력필드, 논리, 상태, 정로, 요구사항, ...)
다) 잠재적 문제(risk based):찾고 있는 문제의 형식은?
라) 행위 : 어떻게 테스트를 수행하는가? (회귀, 스크립트 기반, 계획라 식, 시나리오, 설치, 부하, 성능, ...)
마) 평가 : 테스트를 통과했는지 어떻게 알 수 있을까?(자가 검증 데이터, 저장된 결과와의 비교, 다른 공인 문서화 비교, 일관성, ...)
2) 기타 기법들
가) 호환성 및 상호운영성 테스팅
나) 문서 및 도움말 테스팅

다) Fault 회복[recovery] 테스팅
라) 성능 테스팅
마) 자동화된 테스팅
바) Intrusive 테스팅
사) Thread 테스팅
아) 보안(security) 테스팅
자) 스트레스 테스팅
차) 사용성(usability) 테스팅
카) 볼륨(volume) 테스팅
타) 준수성(compliance) 테스팅

5. 공식 기술 리뷰(Software Peer Review)

가. 소프트웨어 품질(Software Quality)

1) 어떤 시스템, 시스템 구성요소 또는 과정에 대한 명세요구 사항들
(사용자의 요구나 기대)정도
가) 목적에 대한 적합성
나) 명세에 대한 일치
다) 사용자가 SW에 대해 기꺼이 지불하게되는 관련비용
라) 소프트웨어 품질에 대한 두가지 견해
(1) 제품 품질
(2) 프로세스 품질

나. 동료검토란 무엇인가 ?(What is Peer Review ?)

1) 한명 또는 그이상의 동료가 기술적인 컨텐츠와 품질을 평가하기 위한 검토
요구사항, 디자인, 코드, 테스트절차들, 테스트 항목들,
2) 동료가 제품 개발중 제거해야할 결함들을 찾아내는것
3) 명세에 대한 일치
4) 사용자가 SW에 대해 기꺼이 지불하게되는 관련비용
5) 소프트웨어 품질에 대한 두가지 견해
가) 제품 품질
나) 프로세스 품질

다. 동료검토의 잇점들(Benefits of Peer Reviews)

1) 또다른 견해를 얻게됨 : 선입관을 갖지 않고 제품에 대해 결함을 더 쉽게 찾을수 있다.
2) 다음과 같은 것을 전단함 : 소프트웨어 제품과 결함에 대한 발견
3) 신속한 오류 발견과 제거 : 비용감소

4) 반복작업과 테스트 수고의 감소 : 전체 개발 수고 감소

라. 소프트웨어 정밀검사(Software Inspection)

1) 1976년 IBM의 Michael Fagan이 제시
2) 7가지 운영안으로 구성
3) 목적
가) 제품의 모든 결함을 찾음
나) 제품에 있는 결함을 만들어낸 프로세스의 시스템 결함을 모두 찾음
4) 또한 아래와 같은 사항을 할 수 있다.
가) 결함측정
나) 반복작업 결함관리
다) 개발 과정의 시스템 결함 제거

6. 테스트 종류(Types of Testing)

가. 테스트 종류

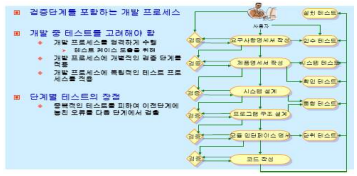
테스트의 종류는 여러 관점에 따라 분류된다.

관점		블랙박스 테스트			화이트박스 테스트
시점	목적	기능 테스트	성능 테스트	병행 테스트	구조 테스트
검증 (개발자)	단위 테스트				
	통합 테스트				
	유닛 테스트				
	시스템 테스트				
승인 (사용자)	인수 테스트				
	설치 테스트				

<그림 11 테스트의 종류>

나. 테스트 단계

1) 검증단계를 포함하는 개발 프로세스



<그림 12 테스트 단계>

2) 단계별 테스트의 장점
중복 테스트를 피해 이전단계에 놓친 오류를 다음 단계에서 검출

다. 단위 테스트

1) 단위 프로그램이나 모듈이 프로그램 명세서에 정의된 바와 같이 기능의 오류없이 수행되는지 검증하는 테스트로 개발자 중심의 테스트이다.
2) 모듈 테스트 또는 컴포넌트 테스트라고도 함
3) 프로그램내의 각 서브 프로그램, 서브루틴, 또는 프로시저, 클래스, 오브젝트 등 프로그램의 최소구성 단위를 테스트
4) 화이트박스 테스트와 블랙박스 테스트 기법 모두 적용 가능하나, 일반적으로 화이트 박스 테스트 기법을 사용함(ex. Branch Coverage)
5) 기능 테스트와 리소스 활용(ex. Memory leaks)과 같은 비 기능 테스트 케이스 포함
6) 장점
가) 디버깅 작업이 쉽다.
나) 병렬 처리가 가능하기 때문에 여러 개의 모듈들을 동시에 테스트 함.

라. 통합 테스트

1) 점진적인 테스트
가) 이미 테스트된 모듈들과 다음에 테스트될 모듈을 결합시킨 후 테스트
2) 단점
가) 오류 원인을 찾기 어려움
나) 디버깅에 많은 시간을 소비
3) 구분(효과적인 방법의 선택, 혼용)
가) 하향식 통합 테스트

나) 상향식 통합 테스트
다) 샌드위치 통합 테스트
라) 빅뱅 통합 테스트
마) 회귀 통합 테스트

마. 확인 테스트

1) 정의
가) 사용자 요구사항이 충족되었는지를 확인
나) 사용자가 테스트에 참여함
다) 프로그램과 프로그램외부 명세서와의 불일치를 찾아냄

2) 수행 방법
가) 기능성 테스트
(1) 블랙박스 테스트 기법 사용
(2) 동등분할, 경계값 분석, 인과 그래프, 오류 추측 기법 주로 사용
나) 비 기능성 테스트
(1) 형상 검토를 통해 소프트웨어 개발과 산출물의 적절성 확인
(2) 문서화, 사용자 편의성, 호환성, 이식성, 오류메시지, 회복, 유지보수성

3) 장점
가) 사용자 중심의 소프트웨어를 구현할 수 있는 정보를 획득함
나) 난해하거나 잘못 해석될 수 있는 문제점을 식별

바. 인수 테스트

1) 정의
가) 사용자에게 소프트웨어 개발이 완료되어 사용할 준비가 되었다는 확신을 주고,
고객의 모든 요구사항을 만족하는지 확인함
나) 개발자가 아닌 사용자가 많이 찾을 수 있는 오류가 존재함

2) 수행 방법
프로그램의 초기 요구사항과 프로그램 최종 사용자의 현재 요구사항을 비교함

3) 종류
가) 사용자 인수 테스트 : 사용자의 요구사항 충족 여부 테스트
나) 운영자 인수 테스트 : 백업/리스토어, 사고복구, 사용자관리, 유지보수 업무, 정기 점검
다) 규정 인수 테스트 : 계약 항목 충족 여부 테스트
라) 알파 & 베타 테스트 : COTS(Selecting a Commercial OfftheShelf)의 경우 사용자

의 피드백을 얻기 위해 베타 테스트를 실시한다.
(1) 알파 테스트 : 개발자가 참여하는 사용자 테스트
(2) 베타 테스트 : 개발자가 참여하지 않는 사용자 테스트

사. 설치 테스트

1) 정의
프로그램 설치 과정에서 오류들이 발생하는지를 확인

2) 이벤트들의 목록
가) 사용자는 다양한 옵션들을 반드시 선택해야함
나) 파일들과 라이브러리들은 반드시 저장되어야 함
다) 적절한 하드웨어 구성이 반드시 존재해야 함
라) 프로그램들간의 연결을 위하여 네트워크 연결이 요구됨

7. TMM(Testing Maturity Model)

가. 프로세스 성숙도모델 : CMM개요

1) 고품질의 제품을 개발하기 위해서는 소프트웨어를 생산하는 프로세스의 품질이 보다 중요함을 인식
2) 미국 국방부는 1984년 카네기멜론 대학에 소프트웨어공학 연구소(SEI:Software Engineering Institute)를 설립하고 1991년에 CMM이라는 프로세스 성숙도 모델 발<표
3) CMM은 소프트웨어 조직에게 소프트웨어를 개발하고 유지보수하기 위하여 조직의 프로세스를 어떻게 관리해야 하는가와 소프트웨어 개발측면과 관리측면의 문화를 어떻게 개선해 나가야 하는가를 제시하는 프레임워크

나. Testing Maturity Model(TMM)

1) 일리노이 공대의 Ilene Burnstein 교수팀에 의해 개발
2) SW-CMM에 호환될 수 있도록 설계된 테스팅 프로세스 성숙도 모델
CMM에서 테스트 활동에 대한 프로세스 개선 보조
3) 기존의 테스팅 프로세스 성숙도 모델들 중 가장 완성도가 높은 모델
가장 일관성 있고 편견한 성숙도 모델 구조를 갖고 있으며, 심사모델 및 절차, 심사도구 및 질문서, 팀 교육 등에 관한 기준을 제시

다. Testing Capability Maturity Model(TCMM)

1) Susan Burgess와 Rodger Drabick이 1996년에 개발한 모델로 다른모델들 보다 많은 부분에서 CMM과 유사하게 구성되어 있음
2) TCMM의 핵심영역은 CMM의 핵심영역의 많은 부분을 그대로 도입하면서, 여기에 테스

트 관련 활동들이 추가된 형태로 구성
3) TCMM은 미 연방 항공 운영 위원회(FAA)의 ACT-200 프로그램의 테스트 성숙도를 심사하는데 성공적으로 적용된 바 있음

라. Test Improvement Model(TIM)

1) Ericson, Subotic, Ursing에 의해 개발된 모델
2) CMM에 근간을 두고 있으며, 테스트 세부 활동을 세부적으로 파악하여 각 활동의 장점은 살리고 약점은 제거하는 기법들을 제시하고 있는 것이 특징임
3) 성숙도 모델구조
   가) 성숙도 수준(5수준) : 초기 / 베이스 라인 / 비용효과 / 위험감소 / 최적화 수준
   나) 핵심 영역 : 조직, 계획 및 추적, 테스트웨어, 테스트케이스, 검토
   CMM의 핵심영역과 유사, CMM의 핵심영역이 하나의 수준에 한정된 것에 비해 5개의 성숙도 수준에 걸쳐 확장됨
4) 심사모델 : 설정심사, 개선계획

마. Test Process Improvement Model(TPI)

1) Kooman과 Pol에 의해 1997년에 개발된 모델로 테스트 프로세스 개선을 더욱 용이하게 할 수 있도록 고안된 모델
2) 조직의 테스트 프로세스의 장점과 약점 영역 결정, 프로세스 성숙도 심사, 개선 사항을 제안
3) 성숙도 모델, 테스트 성숙도 매트릭스, 체크리스트, 개선 제안 사항 등을 지원
4) 성숙도 모델구조
   가) 3가지 성숙도 수준과 14개의 등급, 각각의 수준은 여러 개의 등급으로 구성됨
   (1) 제어수준 : 1~5등급, 테스트 프로세스가 제어되고 있음을 의미.
   (2) 수준별 특성을 가져야 하고 정의된 테스트 전략과 조화를 이루어야 함.
   나) 테스트 명세, 결점보고, 의사 소통이 일어나고, 테스트웨어 및 테스트 운영이 관리됨
   효율수준 : 6~10등급, 효율적 테스트 프로세스의 정립을 의미.
   다) 테스트 프로세스가 자동화되고 통합되며 개발조직에 정착됨
   최적화 수준 : 11~14등급, 최적화 상태를 의미. 계속적인 개선이 조직의 일상적인 활동으로 인식됨

바. Test Organization Maturity Model(TOM)

1) System Evolution 사에서 개발, CMM에서 다루지 못하는 테스트 조직에 대한 이슈들을 강조하여 개발된 모델
2) 기존의 모델이 사후 문제 해결에만 관심을 두고 있기 때문에 문제의 원인, 목적, 계약 사항 등에 대한 요소를 충분히 다루지 못하고 있다고 판단하여 이를 보완하기 위한 방법들을 제시
3) TOM에서는 프로세스 개선을 실현하는 데에 가장 큰 장애 요소는 관리자의 의식변화 및

구성원의 저항감이라고 보고, 이러한 조직 차원의 문제들을 파악하고 우선 순위화 함으로써 해결책을 제시.
4) 기존의 모델과 같은 성숙도 모델 구조는 갖고 있지 않지만, 문제를 파악하고 우선 순위화 하기 위한 질문서와 개선 제한점 등을 포함
5) 지금까지 약 120개의 조직이 TOM을 사용하여 테스트 프로세스의 개선을 시도함

사. Maturity Model for Automated Software Testing(MMAST)

1) 1994년, Mitchel H. Krause에 의해 의료기기 제작에 필요한 테스트 소프트웨어를 자동화를 위한 목적으로 개발
2) 의료기기 제작회사에 적합한 자동화 수준을 결정하기 위하여 다양한 자동화 수준 기술
3) 특정 조직이 측정 수준에 맞추어 의료기기를 제작할 때 필요한 지침서 제공
4) 개별 수준에서 자동화를 달성하는데 소요되는 예상 비용 정보를 체크리스트와 함께 제공
5) 성숙도 모델 구조
   가) 4개의 성숙도 수준으로 구성, 각 수준은 자동화 정도의 수준을 나타냄
   우연적 자동화(측정적 시험), 초기 자동화(문서화, 도구이용), 의도적 자동화(자동화 정의, 스크립트 작성), 향상된 자동화(결점 추적)
   나) 각 수준에서의 주용 핵심 영역이나 실무 활동들은 별도로 정의 되어 있지 않음

아. 각 모델의 비교

	TMM	MMAST	TAP	TCMM	TIM	TOM	TPI	TSM
모델형태	성숙도	자동화 수준	성숙도	성숙도	성숙도	문제의 근원	성숙도	성숙도
개발년도	1996	1994	1995	1996	1996	?	1997	?
수준	5	4	5	5	5	없음	14	3
핵심영역의 수	13	없음	4	17	5	없음	20	6
핵심영역의 적용	하나의 수준에 적용	없음	?	하나의 수준에 적용	전체 수준에 적용	없음	전체 수준에 적용	전체 수준에 적용
테스트 활동	보통	불충분	?	?	보통	부족	충분	?
심사 형태	질문서	자기 심사	외부 심사	외부 심사	질문서	외부 심사	체크리스트	?
심사 요소	심사절차, 질문서 고득점권, 점검항 기준	질문서	질문서	질문서, 체크리스트, 템플릿	심사자동 및 질문서	질문서, 데이터베이스	심사 기준, 체크리스트, 성숙도 레퍼런스	?
심사기반	CMM, ISO, SPICE	질문서 및 자료분석	?	CMM	없음	없음	없음	?

<그림 13 각 모델별 비교>
TSM(Testability Support Model) : 조직의 테스트 활동지원에 대한 성숙도

8. 테스트 계획과 관리

가. 테스트 계획 문서

1) 검증 및 확인 계획

▌IEEE Std. 1059 – V & V Plan Outline

1. Purpose
2. Reference documents
3. Definitions
4. Verification & validation overview
   1. Organization
   2. Master schedule
   3. Resource summary
   4. Responsibilities
5. Tools, techniques & methodologies
6. Life cycle verification & validation
   1. Management of V&V
   2. Concept phase V&V
   3. Requirements phase V&V
   4. Design phase V&V
   5. Implementation phase V&V
   6. Test phase V&V
   7. Installation & checkout phase V&V
   8. Operation & maintenance V&V
6. Reporting
   1. Required reports
      1. Anomaly reports
      2. V&V task reporting
      3. V&V phase summary reports
      4. V&V final report
   2. Operational reports
7. Verification & validation administrative procedures
   1. Anomaly reporting & resolution
   2. Task iteration policy
   3. Deviation policy
   4. Control procedures
   5. Standards, practices & conventions

<표 2 IEEE Std. 1059-V&V Plan Outline>

2) 테스트 계획 문서

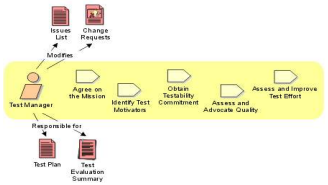
▌IEEE Std. 829 – Outline for a Test Plan

1. Test plan identifier
2. Introduction
3. Test items
4. Features to be tested
5. Features not to be tested
6. Approach
7. Item pass/fail criteria
8. Suspension criteria & resumption requirements
9. Test deliverables
10. Testing tasks
11. Environmental needs
12. Responsibilities
13. Staffing & training needs
14. Schedule
15. Risks & contingencies
16. Approvals

<표 3 IEEE Std. 829-Outline for a Test Plan>

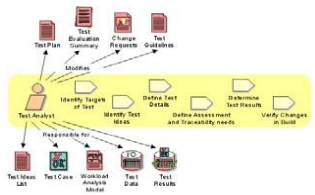
3) 테스트 노력 추정
   가) 전체 프로젝트 개발 일정에 테스트 관련 시간을 포함시키는 것이 중요
   나) 테스트 노력 추정 시 영향을 주는 요소
   (1) 조직의 문화 및 테스트 성숙도
   (2) 테스트 될 소프트웨어의 복잡도
   (3) 테스트 될 소프트웨어의 요구사항
   (4) 테스트를 수행하는 개발자의 기술력

4) 테스트 인력 및 팀 구성
   가) 테스트 팀의 능력은 테스트의 성공 또는 실패를 결정하는 데 중요한 영향을 미친다.
   나) 효과적인 테스트 팀은 기술적인 전문가 뿐만 아니라, 테스트 될 시스 템에 대한 도메인 지식을 가진 전문가도 포함되어야 한다.
   다) 효율적인 테스트를 위해서는 테스트 팀에 참여하는 각 역할에 대해서 수행할 작업과 책임에 대한 명확한 정의가 이루어져야 한다.
   라) 테스트 관련 역할
   (1) 테스트 관리자
   (가) 품질 및 테스트에 대한 책임
   (나) 자원에 대한 계획 및 관리
   (다) 테스트를 방해하는 이슈에 대한 해결



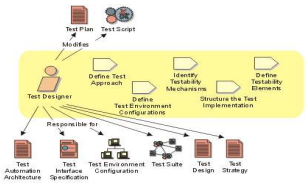
<그림 14 테스트 관리자>

   (2) 테스트 분석가
   (가) 필요한 테스트에 대한 파 악과 정의
   (나) 각 테스트 사이클 별 테 스트의 진행에 대한 자세 한 점검
   (다) 테스트 활동을 통한 품질 향상에 대한 평가



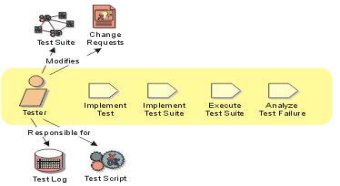
<그림 15 테스트 분석가>

   (3) 테스트 설계자
   (가) 테스트 기법을 결정하고 성 공적인 구현을 책임
   (나) 적절한 기술, 도구, 가이드라 인의 파악



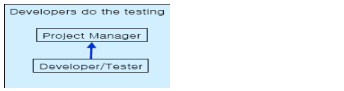
<그림 16 테스트 설계자>

   (4) 테스터
   (가) 실제로 테스트를 수행
   (나) 테스트 결과를 기록



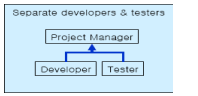
<그림 17 테스터>

5) 테스트 팀의 구조
   가) 개발자가 테스터의 역할도 함
   * 테스트될 소프트웨어에 대한 지식을 바탕으로 테스트를 효율적으로 수행할 수도 있음
   * 독립적이지 않으므로 테스트 효과에 대한 확신이 어려움



<그림 18 개발자가 테스터의 역할함>

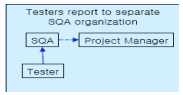
   나) 테스트 전담 인력
   * 전담 테스터가 프로젝트 관리자에게 보고
   * 개발자와 테스터 간의 의사소통 및 팀 워크가 좋음
   * 프로젝트 관리자가 테스트보다는 비용/일정에 초점을 두기 쉬움



<그림 19 테스트 전담 인력>

   다) 테스터가 별도의 SQA에게 보고
   * 비용/일정 뿐만 아니라 품질에도 초점이 주어짐
   * 테스터와 개발팀 간의 의사소통에 어 려움이 있을 수 있음
   * 개발 팀이 소프트웨어 품질에 대

해서 완전한 책임을 지지 않음



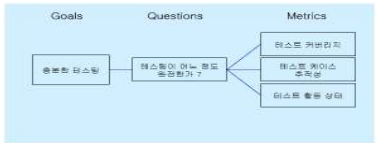
<그림 20 테스터가 별도의 SQA보고>

나. 테스트 관리

1) 테스트 관리 메트릭스

<그림 21 테스트 관리 메트릭스>

2) 테스트 완정성 메트릭스



<그림 22 테스트 완정성 메트릭스>

3) 테스트 커버리지

가) 구조적인 테스트 커버리지 메트릭

(1) 테스트된 명령어 비율

(2) 테스트된 프로그램 경로 비율

(3) 테스트된 관단 비율

(4) 테스트된 조건 비율

(5) 테스트된 단위(모듈)의 비율

(6) 테스트된 인터페이스의 비율

4) 기능적인 테스트 커버리지 메트릭

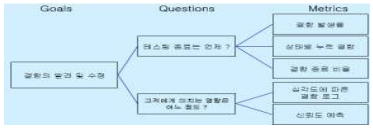
가) 순방향 추적성: 요구사항 → 테스트 케이스

* of functional reqs. traced to tests / Total number of functional reqs

나) 역방향 추적성: 테스트 케이스 → 요구사항

* of test cases traced / Total number of test cases

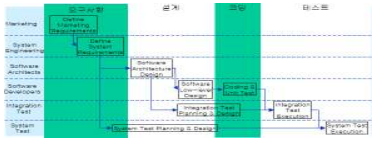
5) 결함 추적 메트릭스



<그림 23 결함 추적 메트릭스>

다. 테스트 프로세스

1) 프로세스 아키텍처



<그림 24 프로세스 아키텍처>

2) 테스트 프로세스 정의

가) 테스트 프로세스 목적

(1) 승인된 요구사항의 준수 여부 확인

(2) 제품이 고객/사용자에게 인도/배포되기 전에 이슈를 파악

나) 입력물

(1) 승인된 시스템 테스트 계획서

(2) 승인된 시스템 테스트 케이스 및 시스템 테스트 프로시저

(3) 승인된 사용자용 문서 및 설치 절차서

(4) 시스템 테스트될 소프트웨어

다) 시작 조건

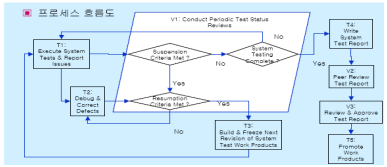
(1) 시스템 테스트를 위한 공간 확보

(2) 통합 테스트의 성공적인 완료

(3) 적당한 수준의 품질을 가진 소프트웨어

(4) 필요한 기술을 가진 테스트 인력의 확보

라) 프로세스 흐름도



<그림 25 프로세스 흐름도>

마) 태스크

바) 검증

사) 메트릭

(1) 수행/성공/실패된 테스트 케이스의 수

(2) 심각도 별 이슈 보고 비율

(3) 상태 별 누적 이슈 보고

아) 종료 조건

(1) 시스템 테스트 종료 조건 충족(시스템 테스트 계획서에 명시)

(2) 테스트 관리자/팀에 의해서 시스템 테스트 보고서가 승인

(3) 최종적인 시스템 테스트 후의 소프트웨어가 베타 테스트 상태로 전환

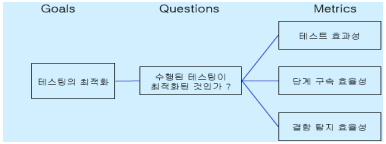
차) 산출물

(1) 시스템 테스트 수준의 소프트웨어 로드

(2) 시스템 테스트 보고서

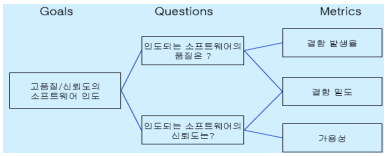
(3) 다음 테스트 사이클로 전환된 소프트웨어

3) 테스트 최적화 메트릭스



<그림 26 테스트 최적화 메트릭스>

4) 테스트 프로세스 효과 메트릭스



<그림 27 테스트 프로세스 효과 메트릭스>

9. 테스트 케이스 설계

가. 테스트 케이스의 정의

1) 특정한 프로그램 경로를 실행해 보거나 특정한 요구사항에 준수하는 지를 확인하기 위해 개발된 입력값, 실행 조건, 그리고 예상된 결과

2) 테스트 항목을 위한 입력값, 예상된 결과, 그리고 실행 조건을 명세 한 문서

3) 테스트하려는 시스템이 수행해야 하는 액션들로 구성되는 일련의 단 계를 의미 이러한 단계를 테스트 프로시저 또는 테스트 스크립트라고도 함

나. 테스트 케이스의 중요성

1) 디지니 라이온 킹(Animated Storybook), 1994-1995

2) 패트리어트 미사일 방어시스템, 1991

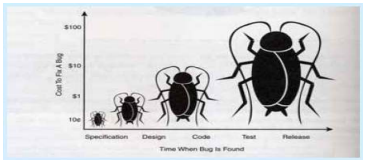
3) Killed 28 US soldiers in Dhahran, Saudi Arabia

4) Y2K(Year 2000) Bug

5) Intel Pentium Floating-Point Division Bug, 1994

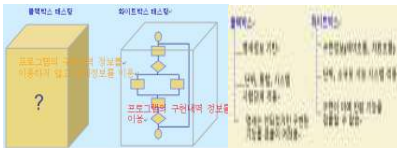
가) (4195835/3145727)*3145727-4195835

나) 자주 발생하지 않음



<그림 28 테스트 케이스의 중요성>

다. 테스트 케이스 설계 기법 분류



<그림 29 테스트 케이스 설계기법 분류>

라. 블랙박스 테스트의 테스트 케이스 설계 기법

1) 동등분할

2) 경계값 분석

3) 의사결정 테이블 기법

4) 인과 그래핑

5) 상태 전이도 기법

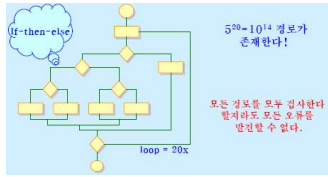
6) 카테고리 분할 기법

7) 에러예측

마. 화이트박스 테스트의 테스트 케이스 설계 기법

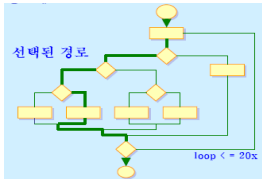
1) 경로 테스트

가) 모든 경로 테스트



<그림 30 모든 경로 테스트>

나) 선택적 경로 테스트



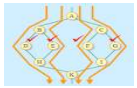
<그림 31 선택적 경로 테스트>

2) 문장 커버리지
가) 테스트하려는 프로그램 내의 모든 문장들을 적어도 한번 이상 실행하도록 요구하는 테스트 케이스 설계 방법
나) 100%의 문장 커버리지 : 프로그램 내의 모든 문장들을 적어도 한번씩 접근하여 테스트

3) 결정 커버리지
가) 정의
분기 커버리지(branch coverage)라고도 함
나) 충분한 테스트 케이스를 설계하여, 각 결정(분기)이 참(true)과 거짓(false)을 적어도 한번 이상 실행시키는 것을 기준으로 하는 테스트 방법
다) 프로그램의 모든 분기 조건들이 적어도 한번은 테스트되도록 테 스트 케이스를 선정하는 방법
라) 문장 커버리지보다 강한 형태의 기준을 가진 테스트 방법

4) 조건 커버리지
가) 정의
결정의 각 조건에 대해, 최소한 한번씩은 모든 가능한 결과를 실 행하도록 보장하기 위해 충분한 양의 테스트 케이스를 작성

5) 기본 경로 커버리지
가) 정의
테스트 케이스에 정의된 집합에 의해 실행 경로를 측정하여 결정



<그림 32 기본경로커버리지>

나) 논리 흐름도
4개의 경로 A, B, D, H, K의 경로는 하나의 경로에 해당하고,이 경로를 테스트 케이스로 정의하고 테스트를 수행하였다면,25% 적용 범위라 할 수 있다

6) 결정 조건 커버리지
가) 결정 내에 존재하는 각 조건들의 참과 거짓을 모두 커버하고, 각 결정의 참과 거짓에 해당하는 모든 분기들이 모두 커버될 수 있 도록 충분히 테스트 케이스를 만들어 나가는 방법
나) 결정 커버리지와 조건 커버리지의 문제점을 해결하기 위한 방법

7) MC/DC
가) 각 기본 조건이 다른 기본 조건들과는 무관하게 전체 조건의 평가에 영향을 미치는지를 알아보기 위한 테스트데이터를 생성하는 것이 목적이다.
나) 각 기본 조건 C에 대해 C를 참으로 만드는 테스트데이터와 거짓으로 만드는 테스트데이터 두 개를 선정한다. 이때, C를 제외한 모든 조건 들의 값은 동일하고 전체 조건의 값이 참이 되는 경우와 거짓이 되는 경우가 발생하도록 테스트 데이터들을 선정한다.

(a && b && c)

Test case	a	b	c	outcome
1	True	True	True	True
2	True	True	False	False
3	True	False	True	False
4	True	False	False	False
5	False	True	True	False
6	False	True	False	False
7	False	False	True	False
8	False	False	False	False

<그림 33 MC/DC>

10. 웹기반 SW 테스트

가. 네비게이션 테스트

1) 정의
사용자가 얼마나 쉽고 빠르게 원하는 정보를 찾아 낼 수 있나 고려한 웹페이지의 구성
2) 중요성
가) 사이트의 성공과 실패의 중요한 열쇠 : 사용자를 오래 머물수 있게
나) 기능과 성능에 대한 문제점도 발견
3) 종류
링크와 URL
가) 내부 링크 : 같은 웹 사이트의 웹 페이지의 링크
나) 외부 링크 : 다른 웹 사이트의 웹 페이지를 참조하는 링크

다) 앵커 : 방문자가 페이지 안의 특정한 위치로 바로 이동할 수 있게 해주는 것
라) 깨진 링크를 방지하기 위한 지침

항 목	설 명
외부링크(External Links)	제어하기 어려운 링크로 가능한 적게 사용해야 한다. 홈페이지로 연결되지 않는 링크가 더 깨지기 쉽다.
내부링크(Internal Links)	상대적인 링크를 더 많이 사용한다.
동적링크(Dynamic Links)	실시간 오브젝트들을 URL에 최소한으로 포함한다. 테스트는 많은 시간이 요구된다. 동적 링크를 사용하면 많은 permutation이 발생하고 이것은 모든 가능한 테스트를 불가능하게 한다.
링크 파라미터(Link Parameters)	가능하다면 파라미터를 요구하는 링크를 최소화한다. - 예: stock quote 페이지의 ticker symbol

<표 4 링크와 URL>

4) 간접 링크
가) 링크가 존재하지 않는 페이지 대신 나타내는 페이지
나) 간접 링크의 종류
* 공손한 <표현
* 요청한 웹 페이지가 존재 하지 않을때 검색 엔진을 연결하여 페이지를 검색
* 요청 페이지는 존재하지 않지만 같은 웹 사이트 안에서 근접한 것을 검색해서 제공
다) 단점
* 꾸준한 관리가 필요하며, 기대 하지 못한 이벤트에 대해서 제어하기 어렵다.
5) 북마크
가) 웹을 사용해서 특정 웹 페이지로 쉽게 찾아갈 수 있도록 페이지에 해당되는 링크를 목록 형태로 저장해 둔 것
나) 해당 페이지의 주소를 입력하지 않고도 그 사이트로 빠르게 이동하기 위해
다) 북마크 테스트를 위한 체크 리스트

북마크 / 즐겨 찾기 검증		
성공	실패	항 목
		북마크로 웹페이지의 컨텐츠를 정확하게 반영할 수 있다.
		32 characters 이상의 북마크가 없다.
		모든 북마크들이 "회사나 페이지의 특정 이름"의 형식으로 시작 한다

<표 5 북마크>

6) 프레임과 프레임셋
가) 프레임
하나의 화면에 여러 개의 문 서가 나타날 때 각 문서가 나타나는 영역
나) 프레임 셋

(1) 한 페이지에 한 개 이상의 프레임이 존재하는 것
(2) 로딩에 따르는 대역폭을 줄인다
(3) 사용자 인터페이스에 유연성을 제공하지만 잠재적인 사용성 문제점
프레임 사용은 방문객에게 그들이 가진 행동이나 기술적 습관을 바꿀 것을 요구
다) 프레임 테스트를 위한 체크 리스트

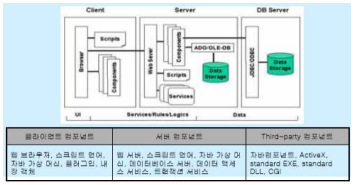
고객에게 보이는 브라우저에 대하여		
성공	실패	항 목
		프레임 셋이 정확하게 디스플레이 된다.
		프레임의 사이즈가 가변적이지 않다.
		프레임 셋 안의 프레임이 북마크 가능하다.
		BACK 버튼으로 바로 전 프레임 디스플레이로 돌아 간다.
		프레임 셋의 로딩 시간이 적절하다.
		프레임 셋을 사용하는 페이지가 올바르게 인쇄된다.
		모든 외부 링크는 새 창으로 시작된다.
		내포된 프레임 셋이 <표현 될 만한 충분한 공간이 있다.
		검색엔진이 프레임 셋 안의 모든 내용을 검색한다.

<표 6 프레임과 프레임셋>

나. 사용자 인터액션

1) 사용자 인터페이스 테스트
가) 단위 사이의 상호 작용을 테스트하는 통합테스트
나) 고려 사항
(1) 설계 방향 : 사용자의 이해가 적절히 이루어질 수 있는지, 사용자가 예상하지 못 한 사용자 인터페이스의 선택 또는 반응이 없는지 테스트
(2) 사용자 상호작용 : 사용자 입력과 관련된 요소와 동적 사용자 인터페이스 컨트롤 등을 테스트
(3) 자료 <표현 : 자료 출력 오류와 관련하여 데이터 오류, 데이터 베이스 질의어 오류,데이터 <표현 오류에 따라 해당되는 부분을 테스트

2) 웹 어플리케이션의 구성 요소
가) 클라이언트 컴포넌트
나) 서버 컴포넌트
다) Third-party 컴포넌트



클라이언트 컴포넌트	서버 컴포넌트	Third-party 컴포넌트
웹 브라우저, 스크립트 언어, 자바 가상 머신, 플러그인, 내장 객체	웹 서버, 스크립트 언어, 자바 가상 머신, 데이터베이스 서버, 데이터 액세스 서비스, 트랜잭션 서비스	자바컴포넌트, ActiveX, standard EXE, standard DLL, CGI

<그림 34 웹 어플리케이션의 구성요소>

3) 폼
가) 사용자가 채워 넣는 공백을 포함한 형식화된 문서
예) 회원 가입 폼
나) 웹 사이트와 사용자의 인터액션 메소드를 제공
다) 예상치 못하거나 원치 않은 결과
예) 뒤로 가기, 앞으로 가기 버튼, history 사이즈 변경 버튼 사용

4) 동적 HTML
가) HTML보다 사용자 상호작용에 좀 더 민감한 웹 페이지를 만들 수 있게 해주는, 새로운 HTML 태그, 옵션, 스타일 시트 및 프로그래밍 등을 의미하는 집합적인 용어
예) 자바 스크립트, VB스크립트
나) 두 유형의 브라우저 모두에 적합한 DHTML 문서 개발이 어렵다
오직 한 가지 플랫폼에서 동작하는 웹 페이지를 개발하거나 또는 각 각의 플랫폼을 위해 두 번 코딩 한다

동적 HTML 페이지		
성 공	실 패	항 목
		동적 HTML이 현재 웹 사이트에 적당하다 (MS- IE 4.0 이상과 Netscape 4.0 이 상을 대부분의 사용자가 사용하기 때문).
		모든 동적HTML 코드가 W3C 기준을 준수한다.
		MS- IE 4.0 이상의 브라우저에서 동적HTML을 포함하는 페이지가 올바르게 보인다.
		Netscape 4.0 이상의 브라우저에서 동적HTML을 포함하는 페이 지가 올바르게 보인다

<표 7 동적 HTML 페이지>

5) CGI
웹 서버를 이용해서 사용자와 어플리케이션 사이에 정보를 전송하는 표준 방법

다. 사용성, 접근성 테스트

1) 사용성(usability) 테스트
가) 잠재적인 웹 사이트의 사용자가 적당한 노력과 시간 안에 그들이 원하는 것을 찾을 수 있는 가를 측정하는 것
나) 처음 또는 자주 사용하지 않는 사용자가 얼마나 쉽게 웹사이트 사용법을 습득하는가의 몇 번의 시도를 통해서 방문자가 웹사이트의 특정 페이지를 찾는데 얼마나 시간이 걸리는지를 측정

2) 접근성(accessibility) 테스트
가) 얼마나 쉽게 웹 사이트의 본문 내용을 읽고 이해할 수 있는가를 측정
나) 개인능력 또는 클라이언트 컴퓨터의 하드웨어나 소프트웨어 구성 (configuration)
다) 문맹이나 종지 않은 시력으로 접근성이 떨어지는 사람들을 위해서는 문자를 소리로 바꿔주는 장치들이 요구

3) 필요성
가) 판매 감소로 인한 손실 vs. 사용성, 접근성에 대한 문제 해결 비용
판매 감소 손실 > 사용성, 접근성 해결 비용
나) 사용자들이 그들이 원하는 것을 제대로 찾지 못하면, 기업은 약50%의 잠재적 판매 손실

라. 성능 테스트

1) 필요성
가) 시스템은 비즈니스가 요구하는 로드를 견뎌 낼만한 능력을 가져야 한다.
나) 오랜 시간 기다리게 된다면 사용자는 불편을 느낄 것이다
다) 로드 테스트
이미 정해진 로드 레벨을 견뎌내는가를 테스트
라) 성능테스트
다양한 로드 레벨에서 성능의 변화를 체크하는 것
마) 스트레스 테스트
시스템이 견딜만한 브레이킹 포인트까지 스트레스를 준다

2) 성능평가 주요3요소
가) 시스템 환경과 사용가능 한 자원
나) 부하(workload)
다) 응답시간 (response time)

3) 수행방법
가) 특정 웹 사이트가 제공하는 서비스 레벨에 대한 측정을 시도
나) 하드웨어 용량(capacity) 계획과 확장 계획에(scalability) 도움
다) 사전 준비
(1) 테스트 담당
(2) 테스트 엔지니어, 시스템 관리자, 네트워크 관리자, 사용자 등
(3) 측정을 위한 요구
(4) 안정적인 시스템
(5) 생산 환경, 도구, 프로세스 등

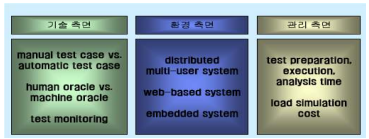
라) 성능 테스트 절차
(1) 명세 : 무엇을 어떻게 테스트 할지 테스트의 요구사항을 확인
(2) 준비 : 디자인, 개발 스크립트, 데이터 준비, 환경 설정
(3) 실행 : 테스트를 실행
(4) 분석 : 시스템 자원, 툴, 테스트로부터 얻은 자료를 분석하여 제출

마) 성능 테스트의 종류
(1) 스모크 / 초벌 테스트
(2) 부하와 확장 계획 테스트
(3) 스트레스와 핫 스팟 (Hot Spot) 테스트
(4) 스파이크 테스트
(5) 무결성(Integrity) 테스트

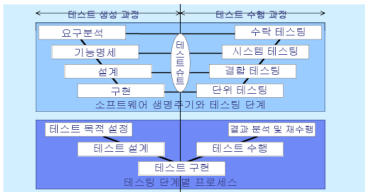
11. 테스트 자동화(Test Automation)

가. 테스트 자동화 개요

1) 개요 : 현재 사용하고 있는 수동 테스트 프로세스를 자동화 한 것
2) 정형화된 수동 테스트 프로세스가 가져야 할 두 가지 조건
가) 결과를 예상할 수 있는 상세한 테스트 케이스
나) 독립적인 테스트 환경

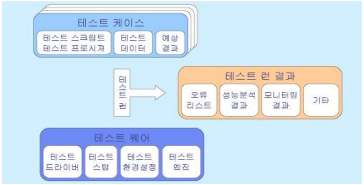


3) 테스팅 이론 - Dual V Model -



<그림 35 Dual V Model>

4) 테스팅의 실제



<그림 36 테스팅의 실제>

나. 테스트 자동화 필요성 및 요구사항

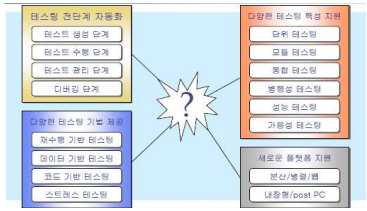
1) 필요성

필요 항목		설 명
기술적문제	수동 테스트 케이스 vs.	소프트웨어가 복잡해지면서 테스트 케이스 역시 복잡해지고 개수도 많아지기 때문에 테스트

환경적문제	자동 테스트 케이스	케이스를 수동으로 수행하는 경우에 시간과 비용이 지나치게 소모됨
	수동 오라클 vs. 자동 오라클	테스트 결과의 옳고 그름을 판정하기 위한 테스트 오라클의 경우 직접 테스트 엔지니어가 눈으로 확인함으로써 생산성이 떨어짐
	테스트 모니터링	분산/내장형 소프트웨어의 경우에 테스트 결과를 눈으로 모니터링 하기가 어려움
	분산 다중 사용자 시스템	다중 사용자를 지원하는 분산 소프트웨어의 테스트에서 수작업으로 다중 사용자의 동작을 제현하기가 어려움
	웹 기반 시스템	클라이언트/서버 구조를 가지므로 분산 소프트웨어의 특징을 가짐
	내장형 시스템	내장형 소프트웨어에서 필요로 하는 이벤트, 시간 문제를 수작업으로 테스트하기 어려움
관리문제	테스트 준비, 수행, 분석 시간	테스트 케이스가 많아지면 테스트 준비, 수행, 분석에 소요되는 시간이 소프트웨어 개발 시간보다 더 길어짐 각 단계별로 테스트 케이스가 달라짐으로 재활용이 불가능
	로드 시뮬레이션	성능 테스트를 위한 로드 생성은 수작업으로 불가능함

<표 8 테스트 자동화 필요성>

2) 요구사항



<그림 37 자동화의 요구사항>

다. 테스트 자동화 이슈(Issues of Test Automation)

1) 테스트 스크립트 : 스크립트를 어떻게 생성 할 것인가?
형식이 무엇인가? 스크립트 레코딩 방법은 어떤 것인가?
2) 테스트 오라클 : 프로그램 출력의 범위를 정함

3) 테스트 수행 엔진 : 시뮬레이션 기능이 어느 부분까지 지원 되는가?

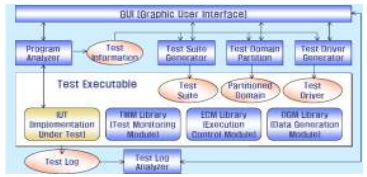
라. 테스트 자동화 도구 분류 및 소개

회사명	제품명	용 도
Mercury	WinRunner, QTP	Windows 어플리케이션의 기능테스트 자동화 도구
	LoadRunner	가상 유저를 생성하여 시스템 성능/부하 테스트
	Topaz	성능 모니터링 도구. 기능 테스트 및 성능 테스트를 거쳐 시스템이 Open된 이후, 실제 운영 환경에서의 시스템 성능 모니터링
	TestDirector	Web-based 테스트 케이스 관리 도구
Compuware	QARun	Windows 어플리케이션의 기능테스트 자동화 도구
	QALoad	가상 유저를 생성하여 시스템 성능/부하 테스트
	Application Expert	시스템 성능 모니터링
	QACenter	Capture and replay에 의한 기능 테스트
	QADirector	테스트 프로세스 관리
IBM Rational	Performance Studio	가상 유저를 생성하여 시스템 성능/부하 테스트
	Robot	Capture and replay에 의한 기능 테스트
Segue	Silk Test	Capture and replay에 의한 기능 테스트
슈어소프트테크	CodeScroll	소스코드 기반 테스트 도구로써, 별도의 명세가 없더라도 테스트를 수행할 수 있음
피슨넷	이지트랙	개발과정에서의 버그 리포트 및 관리, 출시전 베타 테스트 관리, 출시 후 사용 고객의 문의 사항 관리
soft4soft	RESORT for Java	Java 언어를 이용하여 소프트웨어를 개발하거나 또는 개발된 기존의 Java 소프트웨어의 소스 코드 분석
McCabe	McCabe IQ	소스 코드 분석. 사용자는 Source Code의 구조와 흐름, 복잡도를 살펴 볼 수 있음.
PRQA	QAC/QAC++/QAJ	구문분석을 통한 인스펙션 자동화 도구
Parasoft	Insure ++	소스 코드 분석
	Jtest	소스 코드 분석
	WebKing	웹 품질 및 트래픽 분석

<표 9 테스트 자동화 도구 및 소개>

마. 자동화 도구 개발 및 적용 사례

1) 개발 사례
가) API Tester

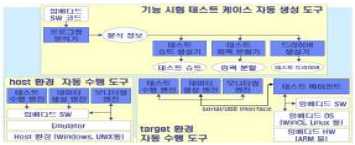


<그림 38 API Tester>

나) Concurrency Tester

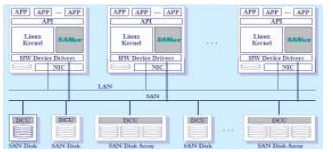
<그림 39 Concurrency Tester>

다) Embedded Tester



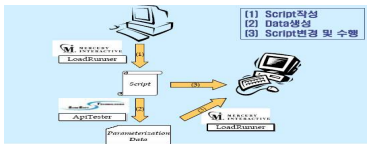
<그림 40 Embedded Tester>

2) 적용 사례
가) 분산 테스팅



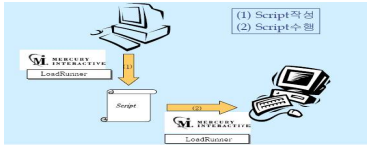
<그림 41 분산 테스팅>

나) 결함 허용성



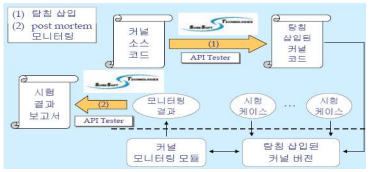
<그림 42 결함 허용성>

다) 동시 접속자 수



<그림 43 동시 접속자 수>

라) 커널 시험

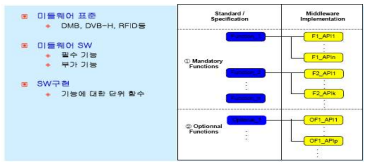


<그림 44 커널 시험>

12. 테스트 사례 - GS시험인증

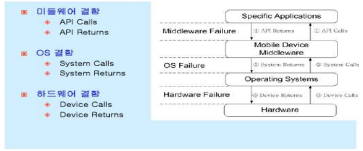
가. 이동단말용 미들웨어 SW 테스트 개념

1) 이동단말용 미들웨어 SW



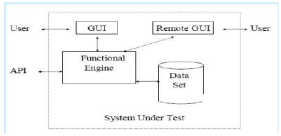
<그림 45 이동단말용 미들웨어 SW>

2) 결합 모델



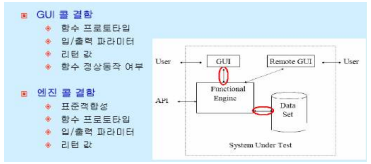
<그림 46 결합 모델>

3) 미들웨어 모델



<그림 47 미들웨어 모델>

4) 미들웨어 결합



<그림 48 미들웨어 결합>

5) OS 결합

<그림 49 OS 결합>

나. 테스트 방법

1) 테스트 방법
- 가) 정적 분석 활용
  - (1) 유닛 테스트
  - (2) API 단위 함수 시험
  - (3) 입/출력 Completeness 확인
  - (4) 매뉴얼 단위 테스트
- 나) 동적 분석 활용
  - (1) 실행시간 테스트
  - (2) 시나리오 테스트
  - (3) 기능 테스트
  - (4) 표준 적합성 테스트

2) 세부 테스트 방법
- 가) 명세서 기반(Specification Driven Testing)
- 나) 커버리지 기반(Coverage Based Testing)
  - (1) Functions
  - (2) Codes
  - (3) Branches
  - (4) Boundary
  - (5) Conditions
  - (6) States
  - (7) Inputs and Outputs
  - (8) Languages

다) 시나리오 테스트(Senario Testing)

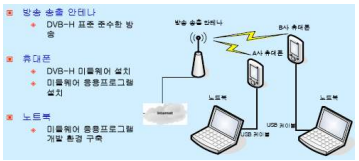
다. 테스트 사례

1) 이동 단말용 DMB

가) 시험 대상 및 절차

<그림 50 시험 대상 및 절차>

나) 시험 환경



<그림 51 시험환경>

다) 유닛 테스트 케이스

<그림 52 유닛 테스트 케이스>

라) 유닛 테스트 : 정적 분석

<그림 53 유닛 테스트 케이스>

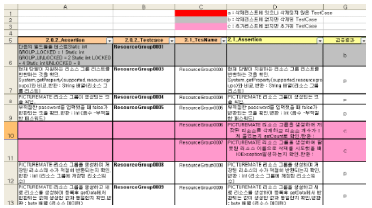
마) 정적 분석 툴
- (1) Open Source
  - (가) FindBugs(Stand alone, Eclipse Plug-In) : http://findbugs.sourceforge.net/
  - (나) Flawfinder(C, C++) : Security weakness http://www.dwheeler.com/flawfinder/
  - (다) PMD(Java) : http://pmd.sourceforge.net/
- (2) Commercial Products
  - (가) Axivion Bauhaus Suite(C, C++, Java, Ada) : Architecture checking, interface analyze, clone detection
  - (나) Coverity Prevent(C, C++, Java) : based on Stanford checker
  - (다) Klocwork(C, C++, Java)

(라) Fortify Software(ASP.NET, C/C++, C#, Java, JSP, PL/SQL, T-SQL, VB.NET, XML) : security vulnerability
(마) FxCop(Microsfot.NET) : stand alone, MS-VS integrated
(바) Understand(C, C++, Java, Ada, Fortran, Jovial, Delphi)

바) 기능 테스트 : 동적 분석

<그림 53 기능테스트 : 동적 분석>

사) 시험 결과



<그림 54 시험 결과>

13. 테스트 사례 - VeriTest

가. VeriTest Inc. 소개

1) 1987년 설립, 미국, 아시아, 유럽 등지에 13개 연구소, 450여명의 연구원 보유
2) SW, HW 및 네트워크 장비에 대한 시험 및 QA컨설팅, BMT수행
3) Microsoft, Unisys, Rational software(IBM) 과 기술 협력 관계 유지를 통해
4) 다양한 인증프로그램 수행

나. 국제 시험인증 프로그램 소개

1) 주요 내용
   가) VeriTest와 TTA의 기술협력 계약 체결(2001.12.01)
   나) VeriTest 현지 기술 연수
   다) VeriTest와 TTA가 공동으로 품질평가모델 개발
   라) 시험결과에 따라 VeriTest-TTA 로고 및 인증서 부여
   마) 국내외 홍보 및 마케팅 지원
   바) 시험.인증 서비스 개시 (2002.7.1)

2) 평가 방법
   가) 평가대상: 최종 SW제품
   나) 평가기준
      (1) VeriTest사의 평가모델 및 방법론을 근거, VeriTest사와 TTA가 공동개발
      (2) 5가지 평가항목
         (가) Core Functionality
         (나) OS Compatibility
         (다) Usability
         (라) Internationalized
         (마) Performance
      (3) 평가결과: Pass/Fail
      (4) 최종 인증확인: VeriTest사에서 수행

다. 국제 시험인증 프레임워크

<그림 55 국제시험인증 프레임워크>

1) Core Functionality
   가) Def: All of core functions in the application should work as expected regardless of operating platform
   나) 기능분류(suggested by James Bach)
      (1) Primary Function
      (2) Contributing function
   다) Sub-characteristic
      (1) Installation
      (2) Functional completeness
      (3) Boundaries (if supported)
      (4) Security (if supported)
      (5) Stability
2) OS Compatibility
   가) Def: These requirements are to ensure proper functionality per targeted platform.
   나) They are general by nature to cover the wide variety of operating system flavors, yet specific enough to ensure a high level of quality
   다) Sub-characteristic
      (1) Installation
      (2) Support OS Fundamentals
      (3) Display
      (4) Accessibility
      (5) Support switching between tasks
      (6) File I/O and printing

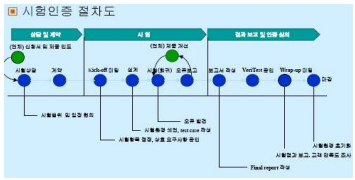
(7) Security
(8) ONNOW/ACPI
(9) Clustering (if supported)

3) Usability
   가) Def:User productivity can be enhanced by providing consistent and comprehensible displays, flexibility to change or structure a system, informative feedback, error tolerance, and reduced demands on short-term memory-all of which give the user a sense of competence, mastery,and control over the system.
   나) 사용성 분류
   다) User-Centered Design Principles and Guidelines
   라) Guidelines for Basic User-Interface Components
   마) Requirement vs. Recommendation

4) Performance
   가) Def: The performance requirements will be established with TTA and the client. TTA will base its standing on performance standards based on user acceptance, such as maximum page load time.
   나) Sub-characteristic
      (1) Performance requirements as per business objective
      (2) Time Endurance

5) I18n(Internationalization)
   가) Def: The internationalization requirements are to ensure proper functionality on foreign language operating systems and/or with localized input/output. These requirements are based on the Western European (Latin-1) languages, represented in the ISO 8859-1 table, as well as Double - Byte languages. IF OTHER LATIN LANGUAGES ARE TARGETED, ADDITIONAL TESTING WILL BE NEEDED.
   나) Sub-characteristic
      (1) Installation
      (2) Character input
      (3) Check for character splitting
      (4) Locales

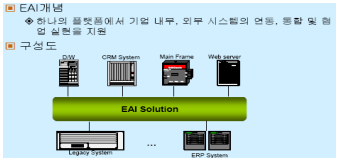
라. 국제 시험인증 절차



<그림 56 국제 시험인증 절차>

마. 국제 시험인증 사례

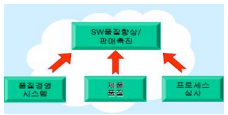
1) 시험개요
   가) 제품종류: EAI (Enterprise Application Integration)
   나) 제품버전: 1.0
   다) OS
      (1) Server: Windows NT계열, Linux, Unix (Solaris)
      (2) Client: Windows NT W/S, 2000 pro
   라) Dependencies
      (1) Web application server: WebLogic
      (2) DBMS: Oracle
      (3) Java VM: JDK 1.3.0 이상
   마) 소요일수: 32 Working day
   바) 투입인원: 4 test engineer, 1 test manager, 1 VeriTest consultant



<그림 57 국제 시험인증 사례>

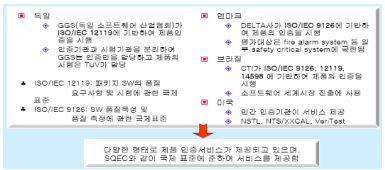
14. 테스트 표준

가. SW 품질 이란?



<그림 58 SW품질>

나. 외국의 품질 인증 사례



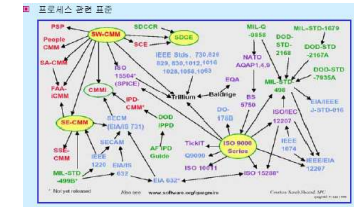
<그림 59 외국의 품질 인증 사례>

다. SW 품질인증 제도

<그림 60 SW 품질인증 제도>

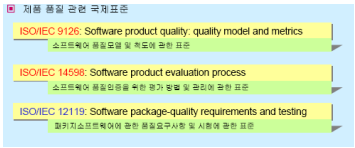
라. SW 품질 관련 국제 표준

1) 프로세스관련 표준



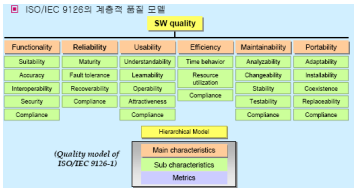
<그림 61 프로세스관련 표준>

2) 소프트웨어 시험 관련 표준



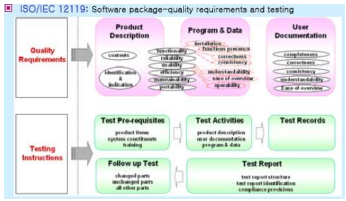
<그림 62 소프트웨어 시험 관련 표준>

3) ISO/IEC 9126



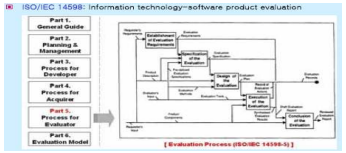
<그림 63 ISO/IEC 9126>

4) ISO/IEC 12119



<그림 64 ISO/IEC 12119>

5) ISO/IEC 14598



<그림 65 ISO/IEC 14598>

15. 관련 사이트, 참고문헌

가. 관련 사이트 : http://www.tta.or.kr(한국정보통신기술협회)

나. 참고문헌 : TTA 사이트 STEP 자료실

T1-SW 테스트 개념.pdf
T2-테스트프로세스.pdf
T3-테스트기법.pdf
T4-공식기술검토_2.pdf
T5-테스트종류.pdf
T6-TMM.pdf
T7-테스트계획과관리.pdf
T8-테스트케이스설계.pdf
A1-표준적합성테스트_1.pdf
A2-웹기반 SW 테스트.pdf
A3-임베디드 SW 테스트.pdf
A4-테스트 자동화.pdf
A5-테스트사례-GS시험인증.pdf
A5-테스트사례-VeriTest.pdf
A6-테스트표준.pdf