공개SW 테스트 가이드

Open Source Software Test Guide

지식경제부 Ministry of Knowledge Economy
nipa 정보통신산업진흥원 National IT Industry Promotion Agency

공개SW 테스트 가이드

Open Source Software Test Guide

정보통신산업진흥원 귀하

본 자료를 “공개SW 테스트 가이드” 최종 연구 결과보고서로 제출합니다.

2012년 8월 10일

- **주관연구기관** : 정보통신산업진흥원
- **과제 책임자** : 김태열 팀장
- **과제 관리자** : 박성호 책임
  최재원 선임

- **연구수행기관** : 공개SW역량프라자
- **연 구 수 행 자** : 이영근 수석
  조동형 책임
  박혜영 선임

요 약 문

공개SW 테스트 가이드는 소프트웨어 테스트에 대한 인식을 제고하고, 소프트웨어 테스트에 대한 지식과 테스트 프로세스 도입 시 표준적이고 활용 가능한 가이드를 제공하고자 개발되었다.

공개SW 테스트 가이드는 개요, 소프트웨어 테스트, 공개SW 테스트로 구분하여 개발되었다.

개요에서는 소프트웨어 테스트의 정의 및 필요성을 통하여 소프트웨어 테스트에 대한 인식을 제고할 수 있도록 하였고, 소프트웨어 테스트에서는 테스트의 분류 및 테스트 프로세스를 소개하고, 테스트 수행 시 필요한 테스트 프로세스와 테스트 기법을 제시하였다.

공개SW 테스트에서는 공개SW의 소개 및 개발 방법론을 통하여 공개SW에 대한 이해를 돕고, 공개SW 역량프라자에서 수행하는 테스트 프로세스를 소개하여 테스트 프로세스에 대한 이해를 돕고자 하였다.

마지막으로 테스트 프로세스의 성공적인 적용을 위해서는 체계적인 소프트웨어의 분석, 설계, 구현이 선행되어야 하며, 철저한 테스트 전략과 계획이 수립되어야 한다. 공개SW 테스트 가이드는 테스트 프로세스에 대한 전반적인 이해와 효율적인 테스트 진행을 위한 가이드로 활용되기를 기대한다.

공개SW 역량프라자 소개

신뢰성 있는 공개SW 발굴 및 활용체계를 마련하고 정보화 예산 심의 지원을 위해 5개의 전문 조직으로 구성되어 있다.

주 요 역 할

공개SW 모델 수립
- 공개SW 수요 창출을 위한 공개SW 참조모델 수립 및 도입 컨설팅

신뢰성 있는 공개SW 발굴
- 공개SW 테스트 지원, 공개SW 기술참조모델 개발

안전한 공개SW 활용체계 마련
- 공개SW 라이선스 검증 서비스 (교육/검증/자문)

공개SW 지식 정보 제공
- 공개SW 포털 사이트(http://www.oss.kr), 기술세미나(Open Technet), 성공사례발굴 및 홍보, 공개SW 커뮤니티 지원

공공부문 공개SW 적용지원센터 운영
- 정보화 예산 심의 지원 (공개SW 도입 계획서 및 적용검토서 작성지원)

TEL : 02)2132-1400 E-mail : oss@nipa.kr

Open Source Software
Test Guide

CONTENTS

CONTENTS

표 목차

그림 목차

I. 개요

1. 소프트웨어 테스트란

소프트웨어 테스트는 프로젝트 단계에서 개발된 제품을 검증 및 확인하여, 사용자에게 높은 품질의 신뢰성 있는 소프트웨어를 제공하는데 그 목적이 있다. 테스트 활동은 개발된 제품의 정확성, 완성도, 그리고 품질을 식별하는데 필요한 지표를 제공하기 위한 작업으로, 제품이 최종 사용자에게 제공되기 전에 제품의 결함을 찾는 목적으로 실시하는 모든 활동의 집합을 말한다.

테스트 활동을 소프트웨어를 실행하면서 수행하는 것으로만 인식하는 경우가 많다. 하지만 그것은 테스트 활동의 일부이며, 테스트 활동은 테스트를 수행하기 전과 후에도 존재한다. 테스트 활동은 테스트 계획, 테스트 설계 같은 활동이나 우선순위의 선정 및 데이터의 생성, 테스트 케이스의 명세화, 테스트 수행 결과 점검 등 일련의 모든 작업을 포함한다.

테스트의 정의

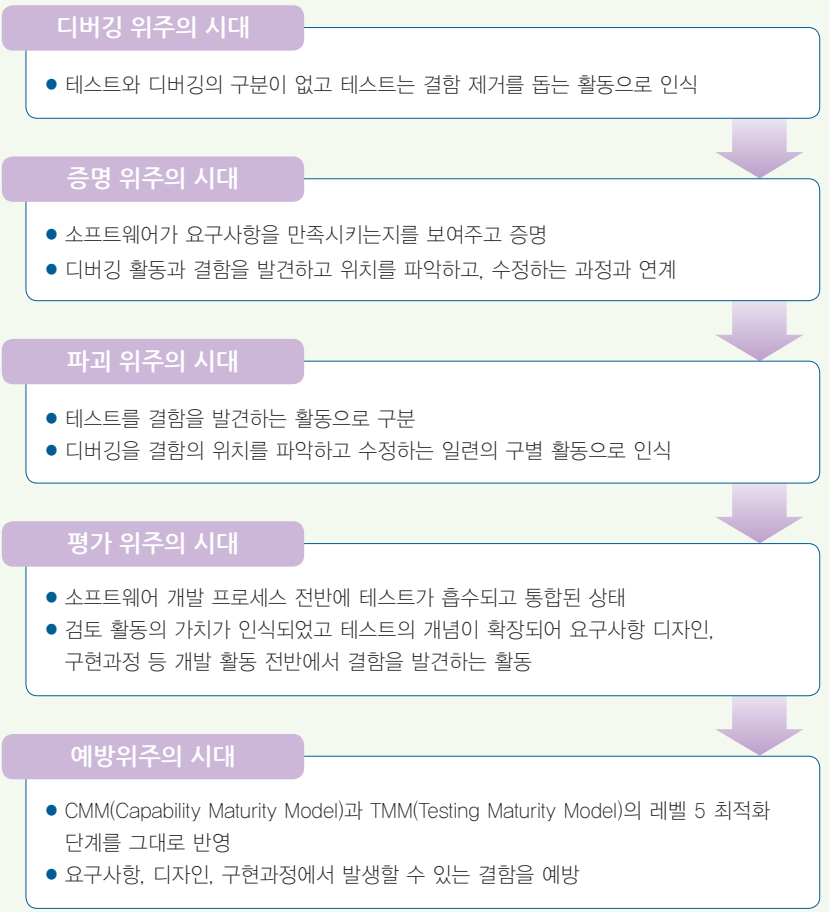
시스템이나 시스템의 구성 요소 또는 소프트웨어 프로그램을 실행하고 평가하는 과정으로, IEEE(Institute of Electrical and Electronics Engineers)[1] 정의에 따르면 수작업 또는 자동화된 방법으로 규정된 요구사항을 만족시키고 있는지 검증하고, 기대되는 결과와 실제 결과의 차이를 식별하는 작업을 말한다. G.J Myers[2]의 정의에 따르면 일반적으로 결함이 없음을 증명하는 것이 아니라 결함이 있음을 발견하기 위하여 체계적으로 수행하는 일련의 작업을 통칭한다.

*출처 : TTA(한국정보통신기술협회)

1) 1884년에 설립된 미국 전기 학회와 1912년에 설립된 무선 학회가 1963년에 합병하여 설립된 미국 최대의 학회
2) Clarkson University의 전기공학 학사, Syracuse University의 컴퓨터 과학 학사, Polytechnic Institute of New York University에서 컴퓨터 과학 박사를 과정을 수료한 컴퓨터 과학자로 사업가이자 작가로도 활동

테스트 기술의 발전 단계

소프트웨어 테스트의 역사는 학자마다 약간씩 상의하게 구분을 하지만, 크게 5단계 정도로 구분한다. 대표적으로 Gelperin[3]과 Hetzel[4]에 의해 정의된 진화적 테스트 모델의 개념을 소개한다.

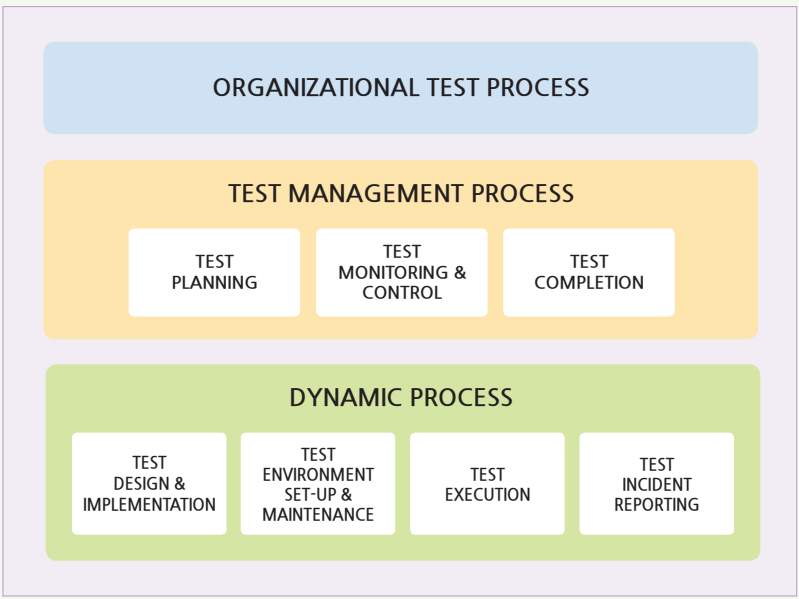


【그림 I-1. Gelperin과 Hetzel의 소프트웨어 기술 발전 5단계】

3) IEEE 829-1989 소프트웨어 테스트 문서 표준을 개발하는 작업 그룹의 전 위원장
4) 1991년에 소프트웨어 테스트 문화의 중추가 될 가이드를 작성한 가이드 문서의 저자

소프트웨어 테스트 프로세스

ISO/IEC[5] 29119에서는 소프트웨어 개발 및 테스트 생명주기 내에서 사용될 수 있는 일반적인 테스트 프로세스 모델을 정의한다. 아래 다이어그램은 테스트 프로세스의 다양한 레이어를 보여준다.



【그림 I-2. ISO/IEC 29119 Test Process Model】

Organizational Test Process에서는 테스트 정책과 테스트 전략 수립 및 관리 활동을 수행하고, Test Management Process에서는 테스트 계획과 모니터링 및 제어, 테스트 완료 보고 관련 활동을 수행한다. Dynamic Test Process에서는 테스트 설계 및 구현, 환경 설정, 유지 보수, 테스트 실행, 결함보고 활동을 수행한다.

5) 표준화와 국제 전기 기술위원회에 대한 국제기구의 합동 기술 위원회

2. 소프트웨어 테스트의 필요성

비즈니스 애플리케이션에서 소비자 제품까지 생활의 많은 부분에서 다양한 제품들이 사용되고 있으며, 그 비중은 계속해서 증가하고 있다. 대다수의 사람들은 이러한 시스템을 사용하면서, 제품이 기대한 대로 동작하지 않는 경우를 많이 접해 보았을 것이다. 제품이 올바르게 동작하지 않는 경우, 다양한 문제가 발생한다. 이로 인한 피해는 금전적인 손실, 시간 낭비, 비즈니스의 이미지 손상 그리고 부상이나 사망에 이르기까지 다양하고 심각하다. 테스트는 이러한 제품 시스템의 문제를 최소화하기 위해 반드시 필요하다.

소프트웨어 테스트의 중요성은 점점 증가되고 있으나, 일반적으로 개발자들은 빠른 시일 내에 제품을 개발해야 하기 때문에, 소프트웨어 테스트에는 시간과 관심을 크게 두지 못하는 것이 국내의 소프트웨어 개발의 현실이다. 프로젝트 관리자가 소프트웨어 테스트에 대한 관심이 없고 전문가도 부족한 상황에서 소프트웨어 제품의 신뢰성 향상을 위해서는 개발 전반에 걸친 소프트웨어 테스트에 대한 증대와 전문가의 양성이 필요하다.

테스트의 목적

다양한 목적의 테스트가 있을 수 있으나, 기본적인 것은 아래와 같다.

- 품질 수준에 대한 자신감의 획득과 정보의 제공
- 비즈니스 리스크를 감소시키는 정보에 근거한 결과 제공
- 개발 프로세스 점검 및 이슈 제기
- 논리적 설계의 구현의 검증
- 시스템과 소프트웨어가 명세를 충족하는지 확인

소프트웨어 테스트의 중요성: 사례

• T-money 무료 개방

2004년 7월 1일 서울특별시의 새로운 대중교통 시스템이 도입되었다. 시스템이 도입되기 전 시스템의 통합을 담당한 L사에서는 제대로 테스트를 거치지 못한 상황에서 서비스를 개시하는 것은 무리이므로 도입 일정을 연기하는 것을 제안하였으나, 서울시는 일정을 연기하지 않았다. 이로 인해 새 대중교통 시스템 도입 첫날 시스템 오류로 인해 전체 대중교통 수단이 무료로 개방되었다.

• 아마존 클라우드 서비스 중단

2011년 4월 세계 최대 퍼블릭 클라우드 서비스인 아마존 EC2[6]가 미러링 과정에서 나타난 용량부족으로 장애를 일으켰다. 이로 인해 전 세계에서 아마존 서비스를 이용하는 훗스위트, 포스퀘어, 퀘라, 넷플릭스, 레딧 등 유명 사이트마저 서비스가 동반 중단되었다.

• 교육행정정보시스템(나이스) 오류

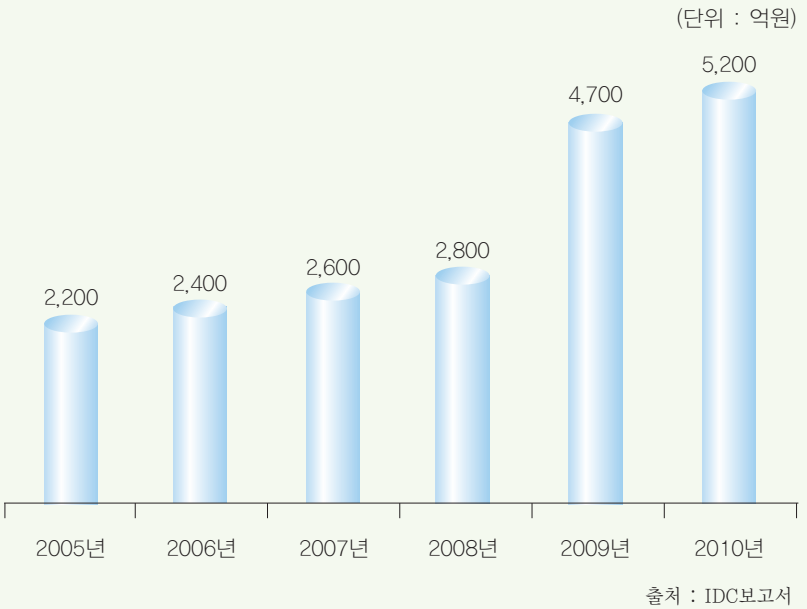
2011년 7월 국내에서는 S사가 정부기관에 납품한 전산시스템에서 오류가 발생, 고등학생 2만 9007명의 내신 석차와 등급이 잘못 산정되어 사회적으로 큰 파장을 일으켰다. 컴퓨터는 소수점 32번째 자리까지만 인식을 하고 마지막 자리는 임의의 숫자를 적용하므로, 통상 프로그램 개발자는 소수점 16번째 자리까지만 값을 인식하도록 인위적으로 계산 방식을 보정해왔다. 이 문제는 소수점의 보정과정이 이루어지지 않아 발생하였다.

6) 아마존에서 제공하는 클라우드 컴퓨팅 플랫폼

소프트웨어 테스트에 대한 업계 동향

• 국내 산업 및 시장 동향

소프트웨어 테스트 시장은 '05년도 2,200억원에서 '10년 5.200억원으로 꾸준한 성장세를 이루고 있다.



【그림 I-3. 국내 소프트웨어 테스트 시장 규모】

• 프로세스 영역별 수준점수 동향

프로세스 영역별로 2009년 대비 2010년 영역별 수준점수를 비교해 보면 PMC, PPQA, IPM 영역이 상대적으로 큰 향상을 보여주고 있다.

【표 I-1. 프로세스 영역별 현황】

구 분	2008년	2009년	2010년
REQM(요구사항 관리)	76.4	78.5	79.7 (1.2증가)
PP(프로젝트 계획 수립)	79.1	80.1	81.6 (1.5증가)
PMC (프로젝트 관리)	78.9	73.4	84.2 (10.8증가)
PPQA(프로세스 및 제품 품질보증)	66.8	64.8	74.6 **(9.8증가)**
MA(측정 및 분석)	57.5	55.6	60.5 **(4.9증가)**
CM(형상관리)	69.1	70.7	69.5 (1.2감소)
SAM(공급업체 계약 관리)	82.3	68.4	66.7 (1.7감소)
RD(요구사항 개발)	76.2	79.3	81.3 (2증가)
TS(기술 솔루션)	79.1	75.5	74.9 (0.6감소)
VER(검증)	68.2	69.1	71.0 (1.9증가)
VAL(확인)	82.2	84.6	81.6 (3감소)
PI(제품 통합)	73.7	81.1	76.7 (4.4감소)
OPF(조직 프로세스 중점관리)	66.1	67.3	73.7 (6.4증가)
OPD(조직 프로세스 정립)	54.2	56.3	62.9 (6.6증가)
OT(조직 교육 훈련)	68.6	72.5	68.5 (4감소)
IPM(통합 제품 팀 관리)	69.3	70.9	79.6 (8.7증가)
DAR(의사결정)	56.7	52.2	58.3 (6.1증가)
RSKM(위험관리)	54.8	62.1	69.3 (7.2증가)

출처 : 2011 공학백서

• 프로세스 보유비율 동향

조직 프로세스 보유 비율을 살펴보면 대부분의 프로세스 보유비율이 2009년 대비 2010년 증가하였다. 특히 품질보증, 측정 및 분석에 대한 보유율은 크게 증가한 것으로 나타났다.

【표 I-2. 프로세스 보유비율 비교】

프로세스 영역	2009년 보유비율	2010년 보유비율
프로젝트 계획	73%	75% (2%증가)
프로젝트 견적	61%	64% (3%증가)
프로젝트 관리	73%	75% (2%증가)
형상관리	67%	70% (3%증가)
품질 보증	66%	72% **(6%증가)**
측정 및 분석	50%	65% **(15%증가)**
공급업체관리	53%	55% (2%증가)
요구사항관리	64%	67% (3%증가)
검토(동료검토)	54%	59% (5%증가)
의사결정	50%	54% (4%증가)
위험관리	60%	69% (9%증가)
프로세스 개선 및 관리	59%	59% (0%증가)
교육훈련	55%	52% (3%감소)

출처 : 2011 공학백서

Ⅱ. 소프트웨어 테스트

1. 소프트웨어 프로세스

소프트웨어 제품을 개발하기 위해 필요한 과정 또는 구조이다. 비슷한 말로 소프트웨어 생명 주기가 있다.

소프트웨어 프로세스의 발전단계

Barry Boehm의 'A View of 20th and 21st Century Software Engineering'을 재구성하여 소프트웨어 프로세스의 발전단계를 소개한다.

1950년대에는 소프트웨어 개발 프로젝트를 위해 하드웨어 프로세스와 유사한 소프트웨어 프로세스 개념이 도입되기 시작하였다. 이후 1960년대에 소프트웨어에 대한 수요가 급증하면서 본격적으로 소프트웨어 프로세스가 도입되었다.

1970년대에는 소프트웨어에 대한 수요가 급증하면서 인력부족사태가 발생하였고, 이에 비전공자들이 투입되면서 선코딩–후수정하는 접근방식을 택하게 되었다. 이에 대한 부작용으로 많은 결함들이 발견되면서 구조적 또는 정형적 기법들이 발생하였으며, 분석, 설계, 구현 등을 순차적으로 진행하는 폭포수 모델을 개발하여 사용하기 시작하였다.

1980년대에는 폭포수 모델이 많은 비용이 소요되고 진척도가 떨어진다는 점을 인식하였고, 이에 재사용성을 높여 효율적으로 소프트웨어를 개발하기 위해 생산성을 높이기 위한 방법들이 연구되었다.

1990년대에는 시장에서 경쟁 우위를 점하기 위해 제품의 시장출시 시간을 단축하기 위한 생산성에 대한 연구가 활성화 되었으며, 폭포수 모델에서 요구사항, 설계 및 구현 등을 동시에 진행할 수 있는 동시프로세스에 집중한 모델을 활용하였다.

2000년대에는 소프트웨어에 대한 기술이나 시장 환경이 급속하게 변화하기 시작하면서 변화에 효과적으로 대응하기 위해 애자일 방법론[7]이 본격적으로 도입되었다.

7) 프로젝트의 생명주기동안 반복적인 개발을 촉진하는 프로세스

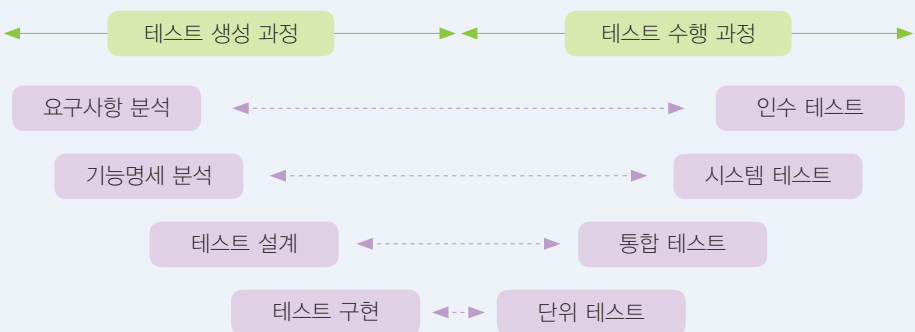
2. 소프트웨어 프로세스와 테스트

테스트는 소프트웨어 개발 활동과 독립적으로 존재하지 않고 밀접하게 연계되어 있으므로, 서로 다른 개발 생명주기 모델에 따라 적용할 수 있는 테스트 방법은 서로 다른 접근법을 필요로 한다.

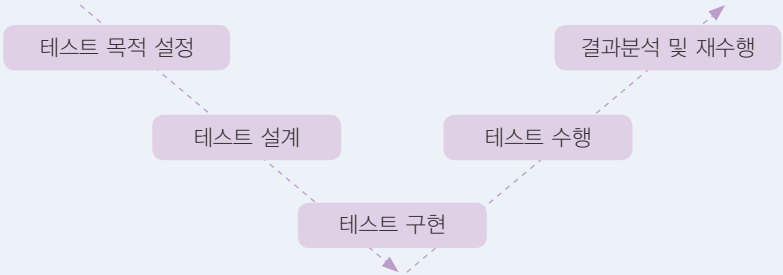
【그림 II-1. V-Model】

여러 가지 변형된 형태의 V-Model이 존재하지만, 일반적인 유형의 V-Model은 4단계의 테스트 레벨로 구성되어 있고, 4단계의 개발 레벨과 대응된다.

실제 V-Model에서는 소프트웨어 제품과 프로젝트에 따라서 개발과 테스트 레벨에 변화를 주거나, 특정 레벨을 추가 또는 삭제 할 수도 있다. 예를 들어, 컴포넌트 테스트 후에 컴포넌트 통합 테스트가 있을 수 있고, 시스템 테스트 후에 시스템 통합 테스트가 있을 수 있다.

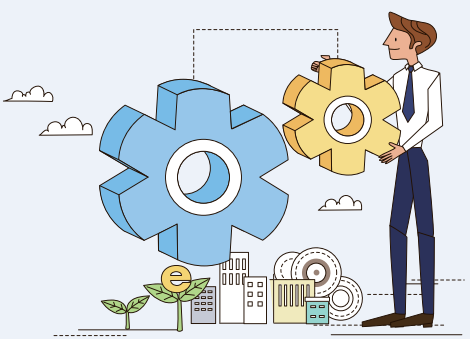


【그림 II-2. V-Model과 테스트】



【그림 II-3. 테스트 단계별 프로세스】

V-Model에서의 테스트의 역할은 각각의 개발 단계에서 접근할 수 있는 테스트 방법을 모델화하여 보여주는 것으로, 실제 현업에서 적용할 때에는 개발하는 소프트웨어 또는 시스템의 특성과 개발 및 테스트 조직의 성격에 맞게 변형시켜 사용해야 한다.



소프트웨어 테스트의 단계별 분류

현재 현업에서 시행되고 있는 소프트웨어 테스트의 분류를 단계별 기준에 따라 분류하면 다음과 같다.

【표 II-1. 단계별 분류】

테 스 트	내 용
단위테스트 (Unit Test)	• 분리된 기능에 대한 검증으로 단위 테스트 프레임워크를 이용하여 개발자가 테스트
통합 테스트 (Integration Test)	• 컴포넌트간의 상호 작용에 대한 검증으로 테스트 입력 값을 만들어 실행한 후 결과를 확인
시스템 테스트 (System Test)	• 전체 시스템 동작에 대한 검증으로 암복호화 데이터의 처리결과 확인, 데이터의 처리시간을 통해 시스템 속도 측정, 정확한 데이터 처리 확인, 성공률과 실패율 확인
인수 테스트 (Acceptance Test)	• 사용자 요구사항 처리에 대한 검증으로 사용자가 요구기능을 입력하고 기능이 정확하게 수행하는지 확인

• 단위 테스트(Unit Test)

개별적으로 테스트할 수 있는 소프트웨어 기능만을 분리하여 검증하며, 일반적으로 코드 접근을 허용하고, 디버깅 도구의 지원 하에 실행한다.

【표 II-2. 단위 테스트 유형】

테 스 트	내 용
인터페이스 테스트	다른 모듈과의 데이터 인터페이스에 대한 테스트
자료구조 테스트	모듈 내의 자료 구조상 오류가 있는가를 테스트
수행 경로 테스트	구조 및 루프 테스트 등에 의한 논리 경로 테스트
오류 처리 테스트	각종 오류들이 모듈에 의해 적절하게 처리되는가를 테스트
경계 테스트	오류가 발생하기 쉬운 경계 값들을 테스트 케이스를 만들어 테스트

• 통합 테스트(Integration Test)

소프트웨어 컴포넌트 간의 상호 작용을 검증하는 프로세스로 엔지니어가 하위 수준 관점을 배제하고, 컴포넌트 간, 서브시스템 간의 통합에 중점을 둔 테스트이다.

【표 II-3. 통합 테스트 유형】

구 분	내 용
하향식 통합 테스트 (Top-Down)	• 주요 제어 모듈은 테스트 드라이버로 사용되고, 스텁은 주요 제어 모듈에서 직접 종속되는 모듈로 교체 • 깊이 우선 또는 넓이 우선의 선택적 통합 접근법을 정하고, 하위 스텁은 실제 컴포넌트들로 한 번에 하나씩 대체 • 테스트들은 각 컴포넌트가 통합됨으로써 수행
상향식 통합 테스트 (Bottom-Up)	• 하위 수준의 컴포넌트들을 특별한 소프트웨어 보조 기능을 수행하는 클러스터로 결합 • 입/출력 테스트 케이스를 통합하기 위해 테스트 드라이버 사용 • 클러스터가 테스트 됨
혼합식 통합 테스트 (Sandwich)	• 하향식 통합 전략과 상향식 통합 전략을 절충한 방식 • 우선적으로 통합을 시도할 중요 모듈들을 선정한 후, 그 모듈을 중심으로 통합
비점진적 테스트 (Big Bang)	• 모든 모듈을 한꺼번에 통합하여 테스트 • 단위 테스트에 많은 시간이 필요 • 시스템의 중요 부분과 부수적인 부분을 구별하지 않음

• **시스템 테스트(System Test)**

소프트웨어 시스템의 특정 요구 사항을 완벽하게 통합된 시스템에서 시스템의 준수 여부를 평가하는 테스트이다. 시스템의 기능 측면뿐만 아니라 비 기능적 요구사항을 시스템이 만족하는지 여부를 검증한다.

【표 II-4. ISO/IEC 9126 품질 특성 국제표준】

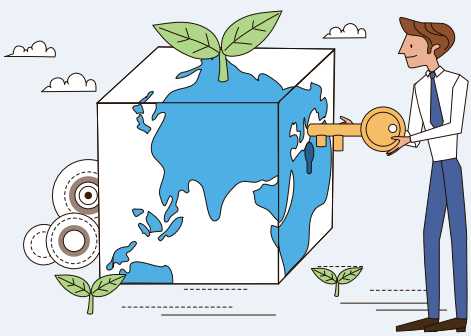
특 성	부 특 성	내 용
기능성 (Functionality)	적합성(Suitability)	적절한 기능을 제공함
	신뢰성(Reliability)	정확한 기능을 제공함
	사용성(Usability)	상호 작용이 가능한 기능을 제공함
	효율성(Efficiency)	정보에 대한 접근제한 기능을 제공함
	유지 보수성(Maintainability)	기능과 관련된 표준을 준수함
신뢰성 (Reliability)	성숙도(Maturity)	결함 회피 기능을 제공함
	장애 허용성(Fault Tolerance)	결함 발생 시 일정 성능수준을 유지함
	복구성(Recoverability)	결함 발생 시 회복이 가능함
	준수성(Compliance)	신뢰성과 관련된 표준을 준수함
사용성 (Usability)	이해성(Understandability)	사용자가 시스템의 기능을 이해할 수 있음
	학습성(Learnability)	사용자가 시스템을 학습할 수 있음
	운용성(Operability)	사용자가 시스템을 제어할 수 있음
	선호도(Attractiveness)	사용자가 시스템을 선호 함
	준수성(Compliance)	사용성과 관련된 표준을 준수함
효율성 (Efficiency)	시간 반응성 (Time Behaviour)	제품이 적절한 반응, 처리시간, 처리율을 제공함
	효율성(Resource Utilisation)	제품이 효율적으로 자원을 사용함
	준수성(Compliance)	효율성과 관련된 표준을 준수함
유지 보수성 (Maintainability)	분석성(Analyzability)	결함의 원인을 식별할 수 있음
	변경성(Changeability)	제품을 변경할 수 있음
	안전성(Stability)	제품 변경에 대한 위험을 회피할 수 있음
	시험용이성(Testability)	제품의 변경 사항을 확인할 수 있음
	준수성(Compliance)	유지보수에 관련된 표준을 준수함
이식성 (Portability)	적응성(Adaptability)	제품의 환경 변경이 가능함
	설치성(Installability)	제품이 설치가 가능함
	공존성(Co-Existence)	다른 제품과 호환이 가능함
	교환성(Replaceability)	제품이 대체가 가능함
	준수성(Compliance)	시스템 이식과 관련된 표준을 준수함

• **인수 테스트(Acceptance Test)**

개발된 시스템이 고객의 요구사항과 일치하는지 확인하기 위해 고객의 입장에서 수행하는 테스트이다. XP(eXtream Programming)[8] 에서는 애자일 개발 방법론의 XP 구현 단계에서 개발팀이 사용자 스토리 기반의 기능을 테스트하는 것으로 정의하고 있다.

【표 II-5. 인스 테스트 유형】

구 분	내 용
알파 테스트(Alpha Test)	사용자에 의해 테스트가 수행되지만 개발자 환경에서 통제된 상태로 수행
베타 테스트(Beta Test)	개발자가 참여하지 않는 테스트로 일정 수의 사용자들에 의해 수행



8) 애자일 개발 프로세스의 대표적인 방법론으로 고객과 함께 2주 정도의 반복개발을 하고, 테스트와 우선 개발을 특징으로 하는 명시적인 기술과 방법

3. 테스트 프로세스

개발 소프트웨어의 품질 기준을 만족하기 위한 테스트를 수행함에 있어 실행조직의 역할과 책임, 필요 작업과 절차 및 산출물을 정의한다. 테스트 프로세스의 제공으로 소프트웨어의 품질과 테스트 커버리지[9] 를 측정하고, 현 시스템의 특징을 파악하여 현재 구현되어 있는 소프트웨어의 모습과 기대치와의 차이점을 판별할 수 있도록 각 테스트 단계의 활동을 정의하는 것이 목적이다.

단, 논리적으로는 순차적이지만 프로세스 내의 활동들은 중첩되거나 동시에 진행될 수 있다.

【표 Ⅱ-6. 테스트 프로세스 단계】

단 계	단 계 별 작 업
테스트 계획	• 테스트 베이시스[10] 검토 • 테스트 상황/요구사항/데이터 식별 • 테스트 기법 할당 • 테스트 용이성 평가 • 테스트 환경 구축
테스트 설계	• 테스트 케이스[11] 명세화, 우선순위 선정, 데이터 생성 • 선행 테스트 • 기대 결과 비교
테스트 실행	• 테스트 실행 • 완료 조건의 달성 여부 확인 • 최종 보고서 작성
테스트 평가	• 산출물 확인 • 테스트 프로세스 평가

9) 시스템의 요구사항에 대한 테스트의 평가를 나타내는 지표
10) 시스템의 요구사항을 포함하고 있는 모든 문서
11) 테스트를 수행하기 위해 개발된 입력값, 실행 조건, 예상결과, 실제결과 등의 집합

테스트 계획

테스트의 목표를 달성하기 위해 필요한 활동 내역을 정의하는 단계이다.

• 테스트 전략

소프트웨어 개발 프로세스 요구사항 분석의 마지막 단계에서 이루어지며, 사용자의 요구사항이 파악되는 시점에 개발시스템의 리스크 범위, 담당자 지정, 확보된 예산 등을 고려하여 향후 수행될 필요가 있는 테스트에 대한 전체 설계를 구상하는 활동이다.

【표 II-7. 테스트 전략】

<table>
<tr><th colspan="3">테스트 전략 활동</th></tr>
<tr><th colspan="2">착수 기준</th><th>입력물</th></tr>
<tr><td colspan="2">• 사용자 요구사항 도출
• 프로젝트 계획서 작성</td><td>• 고객요구 정의서(명세서)
• 프로젝트 개발계획서
• 테스트 전략 가이드라인
• 총괄 테스트 계획서</td></tr>
<tr><th colspan="2">종료 조건</th><th>산출물</th></tr>
<tr><td colspan="2">• 검토되고 승인된 테스트 전략</td><td>• 총괄 테스트 계획서</td></tr>
<tr><td>작 업</td><td colspan="2">① 테스트 요소 식별
② 테스트 범위 정의
③ 테스트 단계별 접근방법 정의
④ 품질 목표 수준 정의
⑤ 테스트 추진 체제 정의
⑥ 테스트 산출물 및 책임자 정의
⑦ 테스트 일정 계획 수립
⑧ 총괄 테스트 계획서 작성
⑨ 총괄 테스트 계획서 검토
⑩ 총괄 테스트 계획서 승인</td></tr>
</table>

• **테스트 전략 활동의 상세 절차**

① 테스트 요소 식별

개발 시스템에 관련된 위험을 평가하고, 그 위험과 관련된 테스트 요소를 정의한다.

– 테스트 요소 식별
- 테스트 대상 시스템 및 관련 비즈니스 위험을 식별
- 테스트 수행 중에 평가되어야 하는 위험을 기반으로 테스트 요소를 식별
- 심각도 및 위험도를 고려하여 테스트 요소를 분류

– 식별된 테스트 요소에 대해 테스트 단계/유형별로 테스트 전략을 정의
- 개발 프로젝트의 생명주기를 파악
- 테스트 요소와 관련된 위험을 최소화하기 위해 개발프로젝트 생명주기의 각 단계에 적합한 테스트 유형을 정의
- 테스트의 유형별로 테스트 방법을 정의

② 테스트 범위 정의

개발 프로젝트의 범위 및 상위 요구사항을 분석하여 테스트 범위를 설정한다. 설정된 테스트 범위는 단계별 테스트에 대한 상세 계획 수립 시 요구분석 및 설계 단계에서 정의된 각 테스트 요구사항 등을 검토하여 테스트 항목, 테스트 항목의 특성, 테스트 하지 않을 항목의 특성을 명세화 하는 기반 자료로서 활용된다.

③ 테스트 단계별 접근방법 정의

프로젝트에서 적용할 테스트 단계와 단계별 테스트에 대한 수행방안을 정의한다.

– 개발생명주기별 테스트 단계 정의
- 개발 프로젝트에서 수행할 테스트 단계를 정의
- 시스템의 크기와 복잡도, 중요도, 개발 성숙도, 고객 요청사항, 테스트 일정, 테스트환경 등을 고려

– 각 단계별 테스트 수행방안 정의
- 단계별로 필요한 테스트 유형 및 수행 방안을 정의

④ 품질 목표 수준 정의

테스트 단계에서 수집 및 관리가 필요한 품질 지표와 목표 수준을 정의한다.

⑤ 테스트 추진 체제 정의

상위 단계에서 정의한 테스트 단계 및 접근 방안을 기반으로 테스트를 수행할 추진 조직 및 역할을 정의한다.

⑥ 테스트 산출물 책임자 정의

테스트 종류별 산출물을 선정하고, 작성 시점 및 책임자를 정의한다.

⑦ 테스트 일정 계획 수립

각 테스트 단계에 대한 전체적인 일정을 기술한다. 상세 일정 수립이 어려운 경우는 각 테스트 전체 일정을 기술하고, 상세 일정은 단계별 테스트 계획 활동에서 수립한다.

⑧ 총괄 테스트 계획서 작성

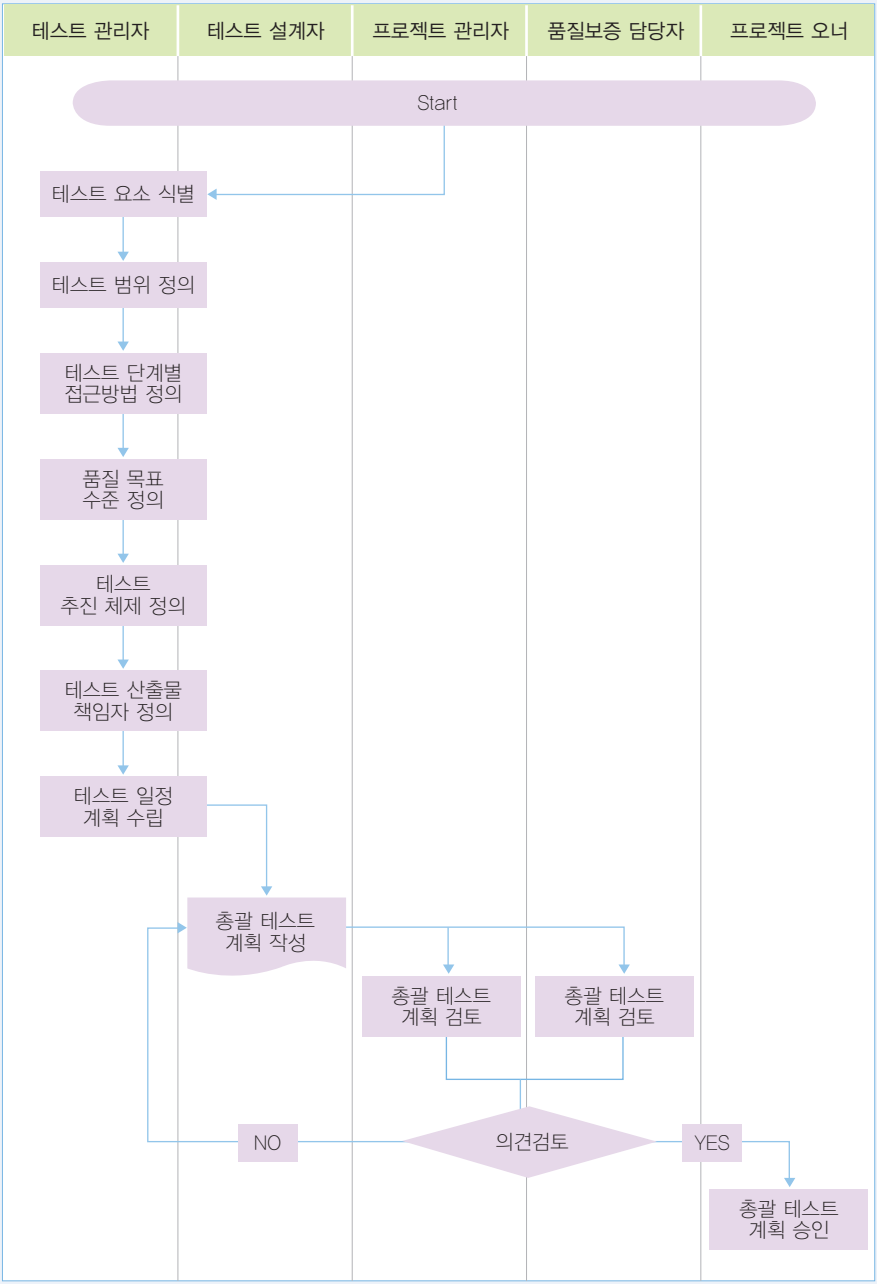
상위 단계를 기반으로 하여, 총괄 테스트 계획서를 작성한다. 총괄 테스트 계획서에는 식별한 테스트의 범위, 단계, 수행방안, 추진 체제, 산출물, 책임자, 일정과 같은 테스트 전략에 대한 내용을 기술한다.

⑨ 총괄 테스트 계획서 검토(테스트 전략 협의)

테스트 설계자는 총괄 테스트 계획서에 기술된 테스트 전략에 대하여, 프로젝트 관리자 및 품질보증 담당자와 검토 및 협의를 수행한다.

⑩ 총괄 테스트 계획서 승인

프로젝트 책임자는 총괄 테스트 계획서대로 테스트가 수행되면, 프로젝트 및 제품의 품질을 보증할 수 있는지 여부를 판단하고, 작성된 총괄 테스트 계획에 대하여 승인을 요청한다.



【그림 II-4. 테스트 전략 활동 작업 흐름도】

• 테스트 계획

각 단계별 상세 테스트 계획을 수립하는 활동으로 각 단계별 테스트에 대한 계획은 개발 단계에 따라 작성되는 시점과 산출물이 다르다.

【표 II-8. 테스트 계획】

<table>
<tr><th colspan="3">테스트 계획 활동</th></tr>
<tr><th colspan="2">착수기준</th><th>입력물</th></tr>
<tr><td colspan="2">• 인수 테스트 계획
– 고객 요구사항 도출</td><td>• 인수 테스트 계획
– 고객 요구사항 정의서(명세서)
– 프로젝트 개발계획서</td></tr>
<tr><td colspan="2">• 시스템 테스트 계획
– 업무 설계서 작성</td><td>• 시스템 테스트 계획
– 총괄 테스트 계획서
– 프로젝트 개발계획서
– 고객 요구사항 정의서(명세서)
– 설계서</td></tr>
<tr><td colspan="2">• 통합 테스트 계획
– 업무 설계서 작성</td><td>• 통합 테스트 계획
– 총괄 테스트 계획서
– 프로젝트 개발계획서
– 고객 요구사항 정의서
– 설계서
– 프로젝트 표준</td></tr>
<tr><td colspan="2">• 단위 테스트 계획
– 테크니컬 설계서</td><td>• 단위 테스트 계획
– 총괄 프로젝트 계획서
– 프로젝트 개발계획서
– 테크니컬 설계서
– 프로젝트 표준</td></tr>
<tr><th colspan="2">종료 조건</th><th>산출물</th></tr>
<tr><td colspan="2">• 테스트 계획서 검토 및 승인</td><td>• 각 단계별 테스트 계획서</td></tr>
<tr><td>작 업</td><td colspan="2">① 테스트 범위 정의
② 테스트 착수기준과 완료기준 정의
③ 테스트 환경 정의
④ 필요 자원 및 일정 정의
⑤ 테스트 리스크 정의
⑥ 결함 및 이슈 보고 절차 정의
⑦ 테스트 계획서 작성
⑧ 테스트 계획서 검토
⑨ 테스트 계획서 승인</td></tr>
</table>

- **테스트 계획 활동의 상세 절차**

① 테스트 범위 정의

테스트 전략을 기반으로 테스트 범위를 정의한다. 모든 대상 영역을 테스트 하지 않을 경우, 테스트 제외 범위와 제외 사유를 기술한다.

② 테스트 착수 기준과 완료 기준 정의

테스트를 시작하기 전에 완료되어야 할 활동들과, 테스트 활동이 종료되어야 할 시점을 정의한다.

③ 테스트 환경 정의

테스트를 수행할 환경을 정의한다.

– 테스트 환경에 대한 하드웨어 및 소프트웨어 구성을 기술한다.

– 테스트에 필요한 테스트 데이터 생성 및 유지 절차를 정의한다.

④ 필요 자원과 일정 정의

자원은 하드웨어, 소프트웨어, 인력으로 구분하여 정의하며 기존 보유 자원도 기술한다. 테스트 일정은 테스트 단계의 주요 활동 및 시작/완료 일자를 정의한다. 일반적으로 프로젝트 수행 일정과 함께 수립된다.

⑤ 테스트 리스크 정의

테스트 수행 시 고려되어야 할 사항과 위험 요소를 기술한다.

⑥ 결함 및 이슈 보고 절차 정의

테스트과정에서 발견된 결함 및 이슈 보고 절차를 정의한다. 보고 절차에는 발견된 결함 기록, 관련자의 정보 공유, 해결하기 위한 절차가 포함된다.

⑦ 테스트 계획서 작성

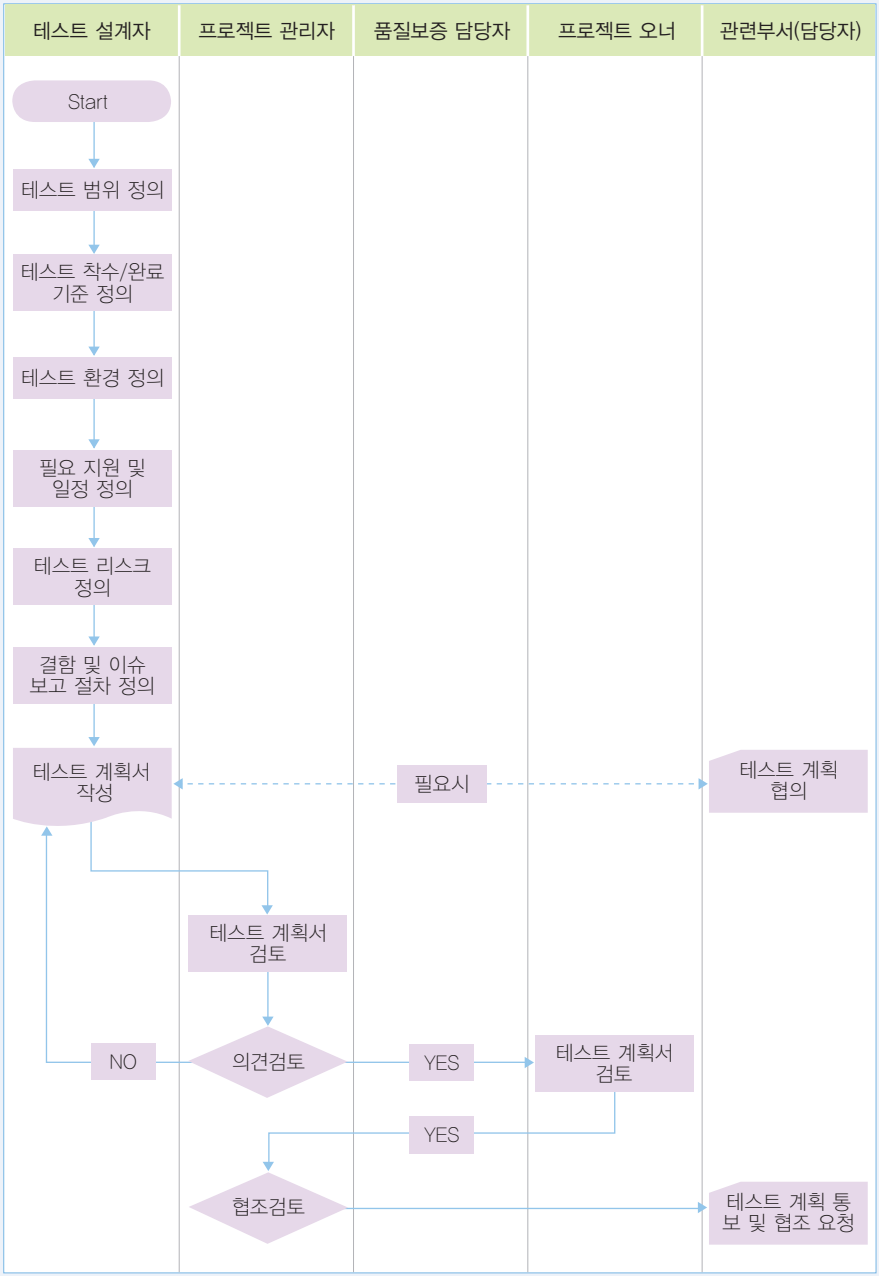
테스트 단계별로 상위 단계의 적업을 기반으로 하여, 상세한 테스트 계획서를 기술한다.

⑧ 테스트 계획서 검토

프로젝트 관리자 및 품질보증 담당자와 검토 및 협의를 수행한다.

⑨ 테스트 계획서 승인

테스트 계획서가 프로젝트 및 제품의 품질을 보증할 수 있도록 작성되었는지 여부를 판단하고 작성된 테스트 계획에 대하여 승인을 요청한다.



【그림 II-5. 테스트 계획 활동 작업 흐름도】

테스트 설계

일반적이고 추상적인 테스트의 목적을 구체적인 상황과 테스트 케이스로 변환하는 단계 이다.

• 테스트 설계

개발 산출물을 이용하여 테스트 케이스를 식별하고 정의하는 활동이다.

【표 II-9. 테스트 설계】

<table>
<tr><th colspan="3">테스트 설계 활동</th></tr>
<tr><th colspan="2">착수기준</th><th>입력물</th></tr>
<tr><td colspan="2">• 시스템 테스트 설계
– 고객요구 정의서 작성 완료
– 시스템 테스트 계획 수립 완료</td><td>• 시스템 테스트 설계
– 테스트 전략
– 고객요구 정의서
– 업무 설계서
– 시스템 테스트 계획서</td></tr>
<tr><td colspan="2">• 통합 테스트 설계
– 설계서 작성 완료
– 통합 테스트 계획 수립 완료</td><td>• 통합 테스트 설계
– 테스트 전략
– 통합 테스트 계획서
– 설계서
– 프로젝트 표준</td></tr>
<tr><td colspan="2">• 단위 테스트 설계
– 테크니컬 설계서
– 단위 테스트 계획 수립완료</td><td>• 단위 테스트 설계
– 테스트 전략
– 단위 테스트 계획서
– 테크니컬 설계서
– 프로그램 명세서
– 프로젝트 표준</td></tr>
<tr><th colspan="2">종료 조건</th><th>산출물</th></tr>
<tr><td colspan="2">• 테스트 케이스 승인</td><td>• 레벨별 갱신된 테스트 케이스 명세서,
테스트 데이터</td></tr>
<tr><td>작 업</td><td colspan="2">① 테스트 케이스 식별
② 테스트 시나리오 및 검증 포인트 정의
③ 테스트 케이스 속성 정의
④ 테스트 데이터 준비 및 검증
⑤ 테스트 케이스 명세서 작성
⑥ 테스트 케이스 명세서 검토
⑦ 테스트 케이스 명세서 승인</td></tr>
</table>

• 테스트 설계 활동의 상세 절차

① 테스트 케이스 식별

프로젝트 특성, 테스트 전략, 해당 테스트 계획서를 기반으로 테스트 케이스를 식별하고 작성한다.

– 단위 테스트

- 테스트 전략과 단위 테스트 계획서를 통해 테스트 범위를 정의
- 프로그램 명세서를 통해 상세 기능을 정의
- 시스템 개발 시 준수해야 하는 표준을 정의
- 각 모듈에 대한 단위 테스트 케이스를 식별하고 작성
- 식별된 테스트 케이스를 검토

– 통합 테스트

- 테스트 전략과 통합 테스트 계획서를 통해 테스트 범위를 정의
- 설계문서를 통하여 통합관점에서의 테스트 대상 기능을 정의
- 시스템 설계 표준을 정의
- 통합 테스트 케이스를 식별하고 작성
- 식별된 통합 테스트 케이스를 검토

– 시스템 테스트

- 테스트 전략과 시스템 테스트 계획서를 통해 테스트 범위를 정의
- 요구사항 명세서, 기능 설계서 등을 기반으로 시스템 기능을 정의
- 시스템 설계 표준을 정의
- 시스템 테스트 케이스를 식별하고 작성
- 식별된 시스템 테스트 케이스를 검토

② 테스트 시나리오 및 검증 포인트 정의

테스트 케이스를 기반으로 테스트 케이스 시나리오와 테스트 케이스 성공/실패의 기준을 정의한다.

③ 테스트 케이스 속성 정의

테스트 케이스 별로 작성자, 작성일, 테스트 케이스 ID, 설명, 기대결과 등의 속성을 템플릿에 정의한다.

④ 테스트 데이터(Test Data) 준비 및 검증

상위 레벨의 테스트 케이스들을 실행 가능한 형태의 실 데이터로 준비한다.

⑤ 테스트 케이스 명세서 작성

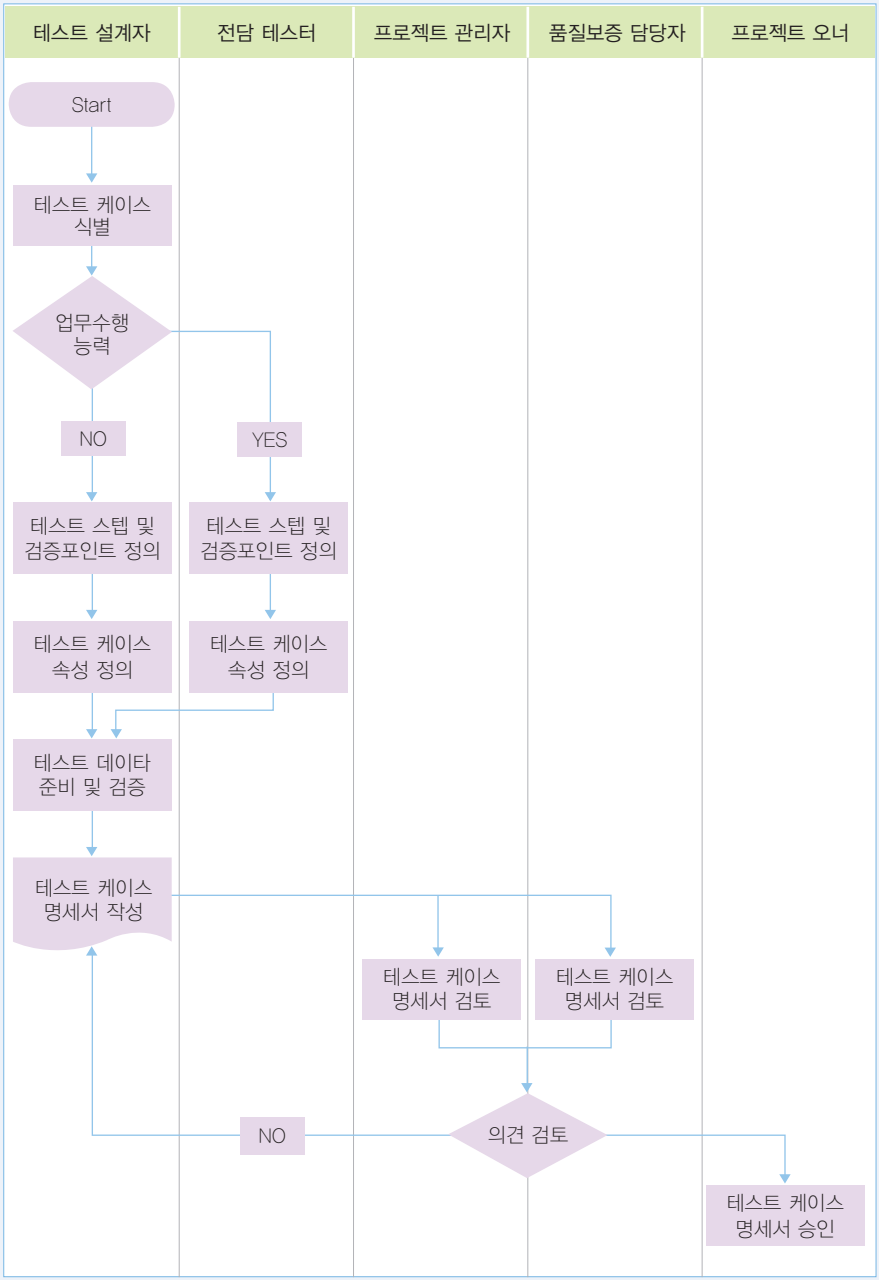
테스트의 효율적이고 효과적인 실행을 위하여 연관성 있는 테스트 케이스를 순서에 맞게 구성하고 테스트 케이스 명세서에 대한 속성을 정의한다.

⑥ 테스트 케이스 명세서 검토

프로젝트 관리자 및 품질보증 담당자등과 검토 및 협의를 수행한다.

⑦ 테스트 케이스 명세서 승인

작성된 테스트 케이스에 대한 승인을 요청한다.



【그림 II-6. 테스트 설계 활동 작업 흐름도】

• 테스트 구현

기능 테스트 자동화 도구를 이용하여, 작성된 테스트 케이스 중 전체 또는 일부를 선정하여 테스트 자동화 실행을 적용할 경우에 해당되는 활동이다.

【표 II-10. 테스트 구현】

<table>
<tr><th colspan="3">테스트 구현 활동</th></tr>
<tr><th colspan="2">착수기준</th><th>입력물</th></tr>
<tr><td colspan="2">• 단위/통합/시스템/인수 테스트 구현
– 단위/통합/시스템/인수 테스트 케이스 설계 완료</td><td>• 단위/통합/시스템/인수 테스트 구현
– 단위/통합/시스템/인수 테스트 케이스 정의서
– 단위/통합/시스템/인수 테스트 계획서</td></tr>
<tr><th colspan="2">종료 조건</th><th>산출물</th></tr>
<tr><td colspan="2">• 선정된 자동화 테스트 케이스 승인
• 테스트 스크립트 승인
• 테스트 데이터 검증 완료</td><td>• 테스트 자동화 스크립트
• 테스트 스위트
• 테스트 데이터</td></tr>
<tr><td>작 업</td><td colspan="2">① 자동화 대상 테스트 케이스 식별
② 테스트 스크립트 레코딩
③ 테스트 스크립트 향상
④ 테스트 데이터 준비 및 검증
⑤ 테스트 스위트 작성
⑥ 테스트 자동화 스크립트 무결성 확인
⑦ 테스트 스위트 검토
⑧ 테스트 스위트 승인</td></tr>
</table>

- **테스트 구현 활동의 상세 절차**

① 자동화 대상 테스트 케이스 식별

비즈니스 리스크가 높은 테스트 케이스를 식별한다.

② 테스트 스크립트 레코딩

테스트 자동화 도구를 이용하여 선정된 테스트 케이스 스크립트를 레코딩한다.

③ 테스트 스크립트 향상

테스트 입력 데이터를 매개변수화(Parameterization)하고, 자동화 테스트 스크립트 실행 성공유무를 판단할 검증 포인트를 설정한다.

④ 테스트 데이터 준비 및 검증

테스트 스크립트 실행 시 테스트 입력 값으로 사용될 테스트 데이터를 준비한다.

⑤ 테스트 스위트 작성

연관성 있는 테스트 케이스를 순서에 맞게 구성하고 테스트 스위트에 대한 속성을 정의한다.

⑥ 테스트 자동화 스크립트의 무결성 확인

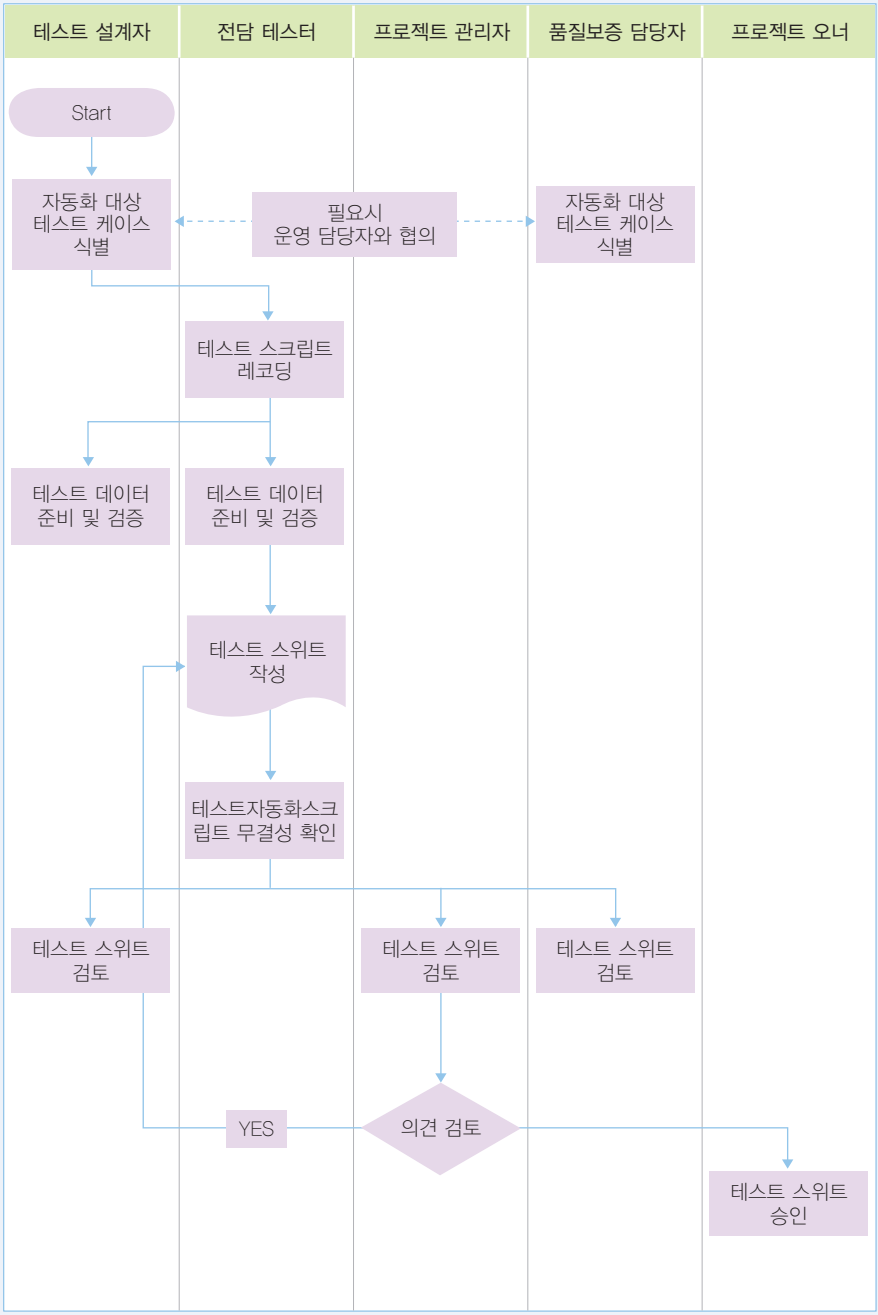
준비된 테스트 자동화 스크립트의 무결성을 검증한다.

⑦ 테스트 스위트 검토

작성된 테스트 스위트를 검토하고 검토 의견이 있을 경우, 개선을 요청한다.

⑧ 테스트 스위트 승인

작성된 테스트 스위트에 대한 승인을 요청한다.



【그림 II-7. 테스트 구현 활동 작업 흐름도】

테스트 실행

계획 및 구현된 테스트 활동을 실제로 실행하는 단계이다.

• 테스트 실행

준비되어 있는 테스트 케이스를 각 테스트 레벨별 테스트 계획에 따라 실행하는 활동이다.

【표 Ⅱ-11. 테스트 실행】

<table>
<tr><th colspan="2">테스트 실행 활동</th></tr>
<tr><th>착 수 기 준</th><th>입 력 물</th></tr>
<tr><td>• 단위 테스트 실행
– 코딩 작업 및 코드 검토 완료
– 단위 테스트 계획서, 테스트 케이스 작성 완료</td><td>• 단위 테스트 실행
– 단위 테스트 계획서
– 단위 테스트 케이스와 테스트 데이터
– 단위 테스트 결과 보고서 양식</td></tr>
<tr><td>• 통합 테스트 실행
– 단위 테스트 완료
– 통합 테스트 계획서, 테스트 케이스 작성 완료</td><td>• 통합 테스트 실행
– 통합 테스트 계획서
– 통합 테스트 케이스와 테스트 데이터
– 통합 테스트 결과 보고서 양식</td></tr>
<tr><td>• 시스템 테스트 실행
– 통합 테스트 완료
– 시스템 테스트 계획서, 테스트 케이스 작성 완료</td><td>• 시스템 테스트 실행
– 시스템 테스트 계획서
– 시스템 테스트 케이스
– 시스템 테스트 스위트 및 테스트 데이터
– 시스템 테스트 결과 보고서 양식</td></tr>
<tr><td>• 인수 테스트 실행
– 시스템 테스트 완료
– 인수 테스트 계획서, 테스트 케이스 작성 완료</td><td>• 인수 테스트 실행
– 인수 테스트 계획서
– 인수 테스트 케이스와 테스트 데이터
– 인수 테스트 결과 보고서 양식</td></tr>
<tr><th>종 료 조 건</th><th>산 출 물</th></tr>
<tr><td>• 설계 및 구현된 테스트 실행 완료</td><td>• 테스트 실행 로그
• 갱신된 테스트 결과 보고서</td></tr>
<tr><th>작 업</th><td>① 테스트 환경 설정
② 테스트 케이스 실행
③ 테스트 실행 결과분석
④ 테스트 자산 개선
⑤ 테스트 결과 작성
⑥ 테스트 실행 결과 검토
⑦ 테스트 실행 결과 승인</td></tr>
</table>

- **테스트 실행 활동의 상세 절차**

① 테스트 환경설정

테스트 계획서에 명시된 테스트 환경을 설정한다. 테스트 환경에는 하드웨어와 지원 소프트웨어를 포함한다.

② 테스트 케이스 실행

해당 테스트 케이스를 실행한다. 자동화 도구를 사용할 경우, 테스트 자동화 스크립트를 실행한다.

③ 테스트 케이스 결과분석

실패로 기록된 테스트 실행 결과를 분석하여 사용자의 단순한 에러인지 결함인지를 판단하여, 결함 관리대장에 기록하고, 발견된 결함을 결함관리대장 양식을 이용하여 기록한 후 결함 내역, 결함 분류, 결함 발생 단계, 결함 심각도 등의 속성을 고려하여 입력한다.

④ 테스트 자산 개선

테스트를 수행하면서 변경된 테스트 케이스, 테스트 요구사항, 테스트 데이터 등을 업데이트 한다.

⑤ 테스트 결과 작성

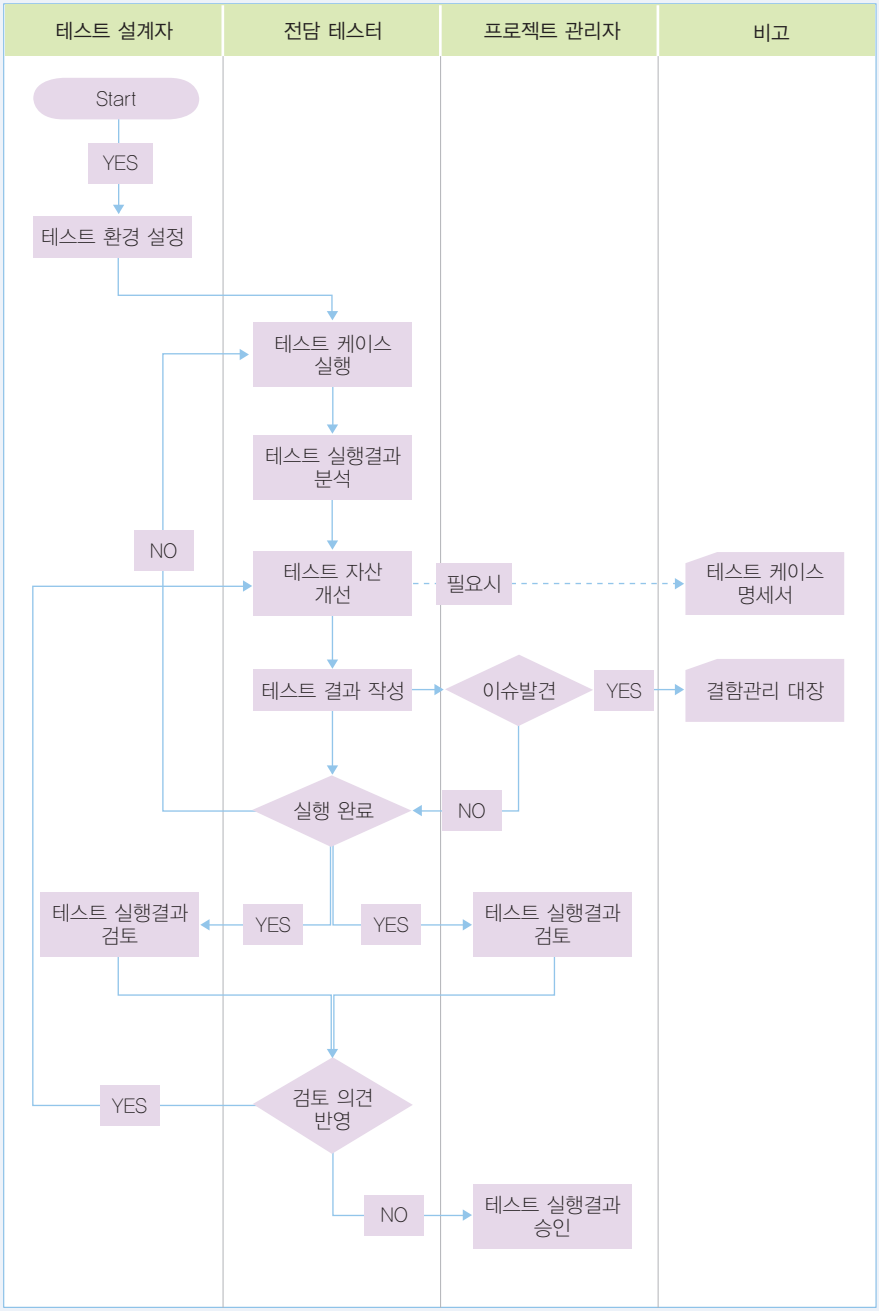
테스트 케이스 명세서에 해당 테스트 케이스 실행 결과를 작성한다.

⑥ 테스트 실행 결과 검토

작성된 테스트 결과 및 결함조치 내역을 취합하여 프로젝트 관리자와 검토 및 협의를 수행한다.

⑦ 테스트 실행 결과 승인

테스트 케이스에 대한 승인을 요청한다.



【그림 II-8. 테스트 실행 활동 작업 흐름도】

테스트 평가

완료된 테스트 활동에서 데이터를 수집하여, 테스트에서 발견된 수치적 데이터와 함께 테스트 경험을 축적하는 단계이다.

• 테스트 평가

각 테스트 레벨 및 유형 별 테스트 결과를 상세히 분석하여 해당 테스트 활동의 종료 기준에 적합한지를 판단하는 활동이다.

【표 II-12. 테스트 평가】

<table>
<tr><th colspan="2">테스트 평가 활동</th></tr>
<tr><th>착수기준</th><th>입력물</th></tr>
<tr><td>• 단위 테스트 평가
– 테스트 케이스 실행 완료
– 테스트 결과 보고 작성 완료</td><td>• 단위 테스트 평가
– 테스트 계획서
– 테스트 결과 보고
– 결함관리대장</td></tr>
<tr><td>• 통합 테스트 평가
– 통합 테스트 실행 완료
– 통합 테스트 결과 보고 작성 완료</td><td>• 통합 테스트 평가
– 테스트 계획서
– 테스트 결과 보고
– 결함관리대장</td></tr>
<tr><td>• 시스템 테스트 평가
– 시스템 테스트 실행 완료
– 시스템 테스트 결과 보고 작성 완료</td><td>• 시스템 테스트 평가
– 테스트 계획서
– 테스트 결과 보고
– 결함관리대장</td></tr>
<tr><td>• 운영 테스트 평가
– 운영 테스트 실행 완료
– 운영 테스트 결과 보고 작성 완료</td><td>• 운영 테스트 평가
– 테스트 계획서
– 테스트 결과 보고
– 결함관리대장</td></tr>
<tr><th>종료 조건</th><th>산출물</th></tr>
<tr><td>• 각 테스트 단계별 완료기준 만족</td><td>• 각 테스트 단계별 결과보고서
• 각 테스트 단계별 테스트 케이스 명세서
• 결함관리대장</td></tr>
<tr><td>작 업</td><td colspan="2">① 테스트 결과 상세 분석
② 테스트 커버리지 평가
③ 결함 분석 및 평가
④ 테스트 종료결정
⑤ 테스트 결과 보고서 작성
⑥ 테스트 결과 보고서 검토 및 승인</td></tr>
</table>

- **테스트 평가 활동의 상세 절차**

① 테스트 결과 상세분석

테스트 수행 결과로 작성된 모든 테스트 결과 보고와 결함관리대장을 검토한다.

② 테스트 커버리지 평가

실제로 제품의 모든 기능을 100% 테스트할 수 없으므로, 어느 정도 테스트가 수행 되었는지 파악하기 위해 커버리지를 측정한다.

③ 결함 분석 및 평가

– 테스트 수행 시 발견된 결함을 분석하여 의미 있는 정보를 도출한다.

  • 발견된 결함 평가는 소프트웨어 품질 및 신뢰성에 대한 정보를 제공한다.
  • 결함 분석 및 평가는 단순히 결함 수의 계산에서부터 통계 모델링 적용까지 다양한 방법을 적용하여 수행할 수 있다.

– 결함 분석을 위해 측정치를 작성하고, 결과에 대한 검토 및 분석을 수행한다.

  • 결함 분포: 특정 속성에 해당되는 결함 수
  • 결함 추세: 시간 흐름에 따른 결함 수에 대한 추이
  • 결함 에이징: 특정 결함 상태에 머물러 있는 시간

– 결함 분석에서는 하나 또는 그 이상의 결함 관련 파라미터와 연관하여, 결함 분포를 분석하는데, 가장 일반적으로 사용하는 결함 파라미터는 다음과 같다.

  • 우선순위(Priority): 결함 해결의 상대적인 중요성
  • 심각도(Severity): 결함의 상대적인 영향 정도
  • 근원(Source): 결함의 원인이 된 위치 및 근본적인 원인

④ 테스트 종료 결정

테스트가 수행되고 테스트 종료 조건을 만족할 경우 테스트 활동을 종료한다.

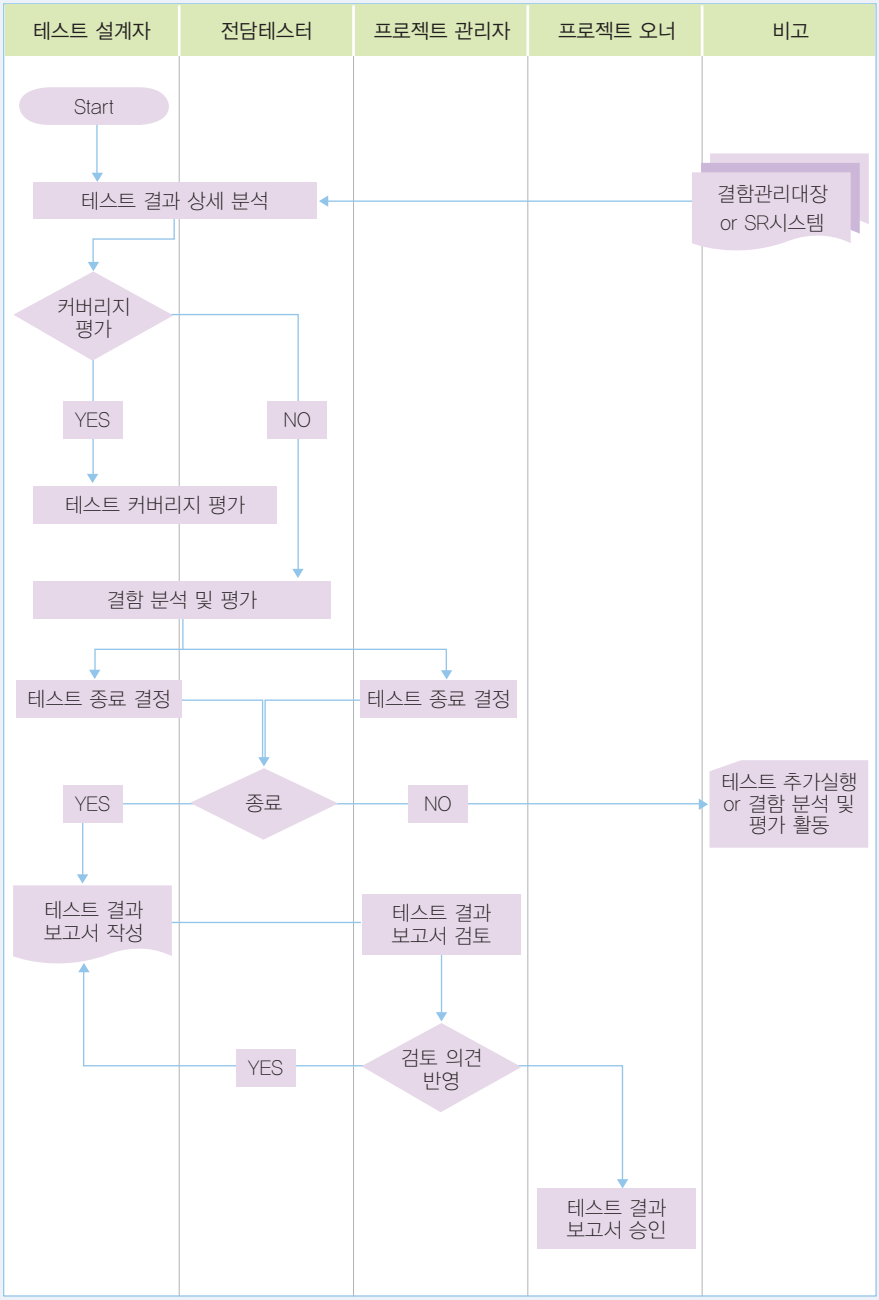
⑤ 테스트 결과 보고서 작성

테스트 결과 보고서는 테스트 대상 시스템의 품질과 테스트 상태에 대해 객관적인 평가를 내리고, 모든 이해관계자들과 의사소통 하는 수단으로 사용하기 위하여 작성된다. 경우에 따라 여러 관계자를 위한 문서를 만들 수 있는데, 대상이 누구냐에 따라 양식과 보고의 내용이 달라질 수 있다.

단, 테스트 결과는 다르게 왜곡해서는 안 되며, 독자의 수준에 따라 결과 보고서의 전문 용어의 사용 여부, 표현 방법 등이 달라질 수 있다.

⑥ 테스트 결과 보고서 검토 및 승인

작성된 테스트 결과 보고서를 프로젝트 관리자 및 품질보증 담당자등과 검토 및 협의를 수행한다.



【그림 II-9. 테스트 평가 활동 작업 흐름도】

결함관리

결함 관리 활동을 수행하여 발생한 결함의 재발생을 막고, 유사 결함 발견 시 처리 시간을 단축하기 위한 활동이다.

【표 II-13. 결함관리】

<table>
<tr><th colspan="3">결함 추적 관리 활동</th></tr>
<tr><th>착수기준</th><th colspan="2">입력물</th></tr>
<tr><td>• 단위 테스트 결함 추적관리
– 해당사항 없음</td><td colspan="2">• 단위 테스트 결함 추적관리
– 테스트 활동 로그
– 결함관리대장</td></tr>
<tr><td>• 통합 테스트 결함 추적관리
– 설계 문서 결함 발견
– 통합 테스트 평가 완료</td><td colspan="2">• 통합 테스트 결함 추적관리
– 테스트 활동 로그
– 결함관리대장</td></tr>
<tr><td>• 시스템 테스트 결함 추적관리
– 요구사항 명세서 결함 발견
– 시스템 테스트 평가 완료</td><td colspan="2">• 시스템 테스트 결함 추적관리
– 테스트 활동 로그
– 결함관리대장</td></tr>
<tr><td>• 운영 테스트 결함 추적관리
– 요구사항 명세서 결함발견
– 운영 테스트 평가 완료</td><td colspan="2">• 인수 테스트 결함 추적관리
– 테스트 활동 로그
– 결함관리대장</td></tr>
<tr><th>종료 조건</th><th colspan="2">산출물</th></tr>
<tr><td>• 심각도 상위 수준의 결함 해결여부 확인</td><td colspan="2">• 결함관리대장
• 테스트 활동로그
• 테스트 결과 보고서</td></tr>
<tr><th>작 업</th><td colspan="2">① 에러 발견
② 에러 등록
③ 에러 분석
④ 결함 확정
⑤ 결함 할당
⑥ 결함 조치
⑦ 검토 및 승인</td></tr>
</table>

- **결함관리 활동의 상세 절차**

① 에러 발견

요구사항 분석, 설계, 테스트 수행 중에 에러가 발견될 경우, 전문 테스터와 프로젝트 팀과 의논한다.

② 에러 등록

결함관리대장에 발견된 에러를 등록한다.

③ 에러 분석

등록된 에러를 분석하여 단순 에러인지 실제 결함인지를 분석한다.

④ 결함 확정

등록된 내용이 결함으로 확정 될 경우, 결함을 결함확정 상태로 설정한다.

⑤ 결함 할당

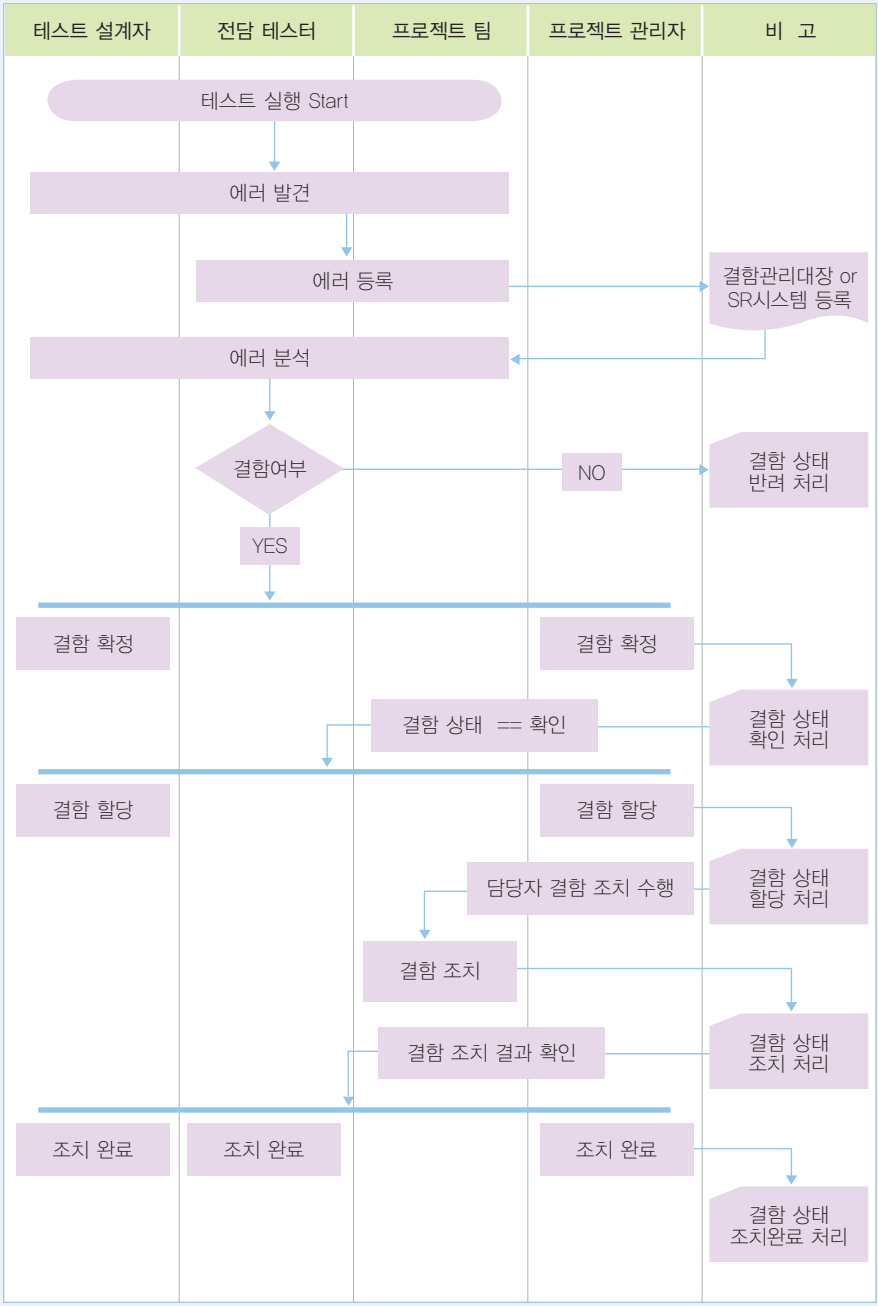
결함을 해결할 담당자를 지정하고, 해당 결함을 할당한다. 이때 결함은 할당 상태가 된다.

⑥ 결함 조치

결함에 대한 수정 활동을 수행하고, 수정활동이 완료된 경우 결함을 결함조치 상태로 설정한다.

⑦ 결함 조치 검토 및 승인

수정이 완료된 결함에 대한 확인 테스트를 수행하여, 정상적으로 결함이 수정되었는지를 확인한다. 정상적으로 조치 완료된 경우 결함을 조치완료 상태로 설정한다.



【그림 II-10. 결함관리 활동 작업 흐름도】

4. 테스트 기법

테스트는 보다 적은 테스트 케이스로 보다 많은 결함을 찾을 수 있도록 설계하기 위해, 테스트 기법을 활용할 필요가 있다. 현재 소프트웨어 테스트 관련 국제표준(ISO/IEC 29119)은 소프트웨어공학 기술위원회 표준화 분과가 운영되고 있으며, 작업그룹(WG26)에서 표준 제정을 담당하고 있다. 본 가이드에서는 국제표준(ISO/IEC 29119)에 따른 동적 소프트웨어 테스트 기법을 소개한다.

【표 II-14. 동적 테스트 기법의 분류】

명세기반 기법	구조기반 기법
• 등가분할(Equivalence Partitioning) • 분류 트리 기법(Classification Tree Method) • 경계값 분석(Boundary Value Analysis) • 상태 전이 테스팅(State Transition Testing) • 결정 테이블 테스팅(Decision Table Testing) • 원인-결과 분석(Cause-Effect Graphing) • 조합 테스트 기법(Combinatorial Test Techniques) • 시나리오 테스팅(Scenario Testing) • 오류 추정(Error Guessing)	• 구문 테스팅(Statement Testing) • 결정 테스팅(Decision Testing) • 조건 테스팅(Condition Testing) • 데이터 흐름 테스팅(Data Flow Testing)

명세기반 기법

시스템에서 제공하는 기능 및 메뉴 등 명세를 기반으로 테스트 케이스를 설계하는 기법이다.

• 등가분할(Equivalence Partitioning)

소프트웨어나 시스템의 입력값은, 입력의 결과로 나타날 결과값이 동일한 경우, 하나의 그룹으로 간주될 수 있다. 그리고 이러한 그룹 내의 입력값은 내부적으로 같은 방식으로 처리됨을 가정한다. 동등분할은 이러한 원리를 이용하여, 입력값/출력값 영역을 유한개의 상호 독립적인 집합으로 나누어 수학적인 등가 집합을 만든 후, 각 등가집합의 원소 중 대표값 하나를 선택하여 테스트 케이스를 선정하는 것이다. 이 기법은 가능한 모든 경우의 수에서 테스트 개수를 줄여준다.

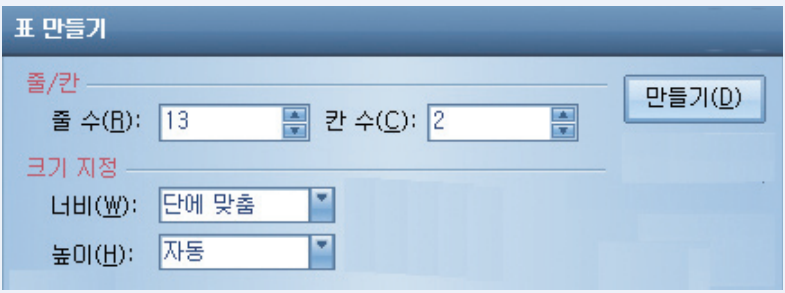
[예제]

A라는 학교에서 학생들의 성적에 따른 등급을 자동으로 계산해 주는 프로그램을 만들고자 한다. 100~90점은 A, 89~70점은 B, 69~50점은 C, 49점 이하는 D로 산정하고자 한다. 이때 등가분할에 의거한 데이터는 100〈=점수〈=90, 89〈=점수〈=70, 69〈=점수〈=50, 점수〈=49로 총 4개의 데이터가 산출될 수 있다. 이것은 최소한 결과의 값을 선택하여 테스트를 수행한다는 것까지 보장한다.

• **분류 트리 기법(Classification Tree Method)**

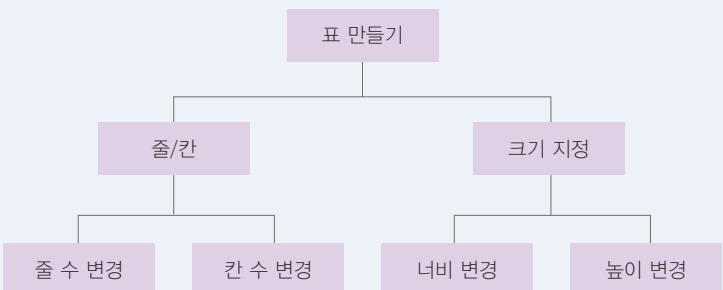
소프트웨어의 일부 또는 전체를 트리 구조로 분석 및 표현하여 테스트 케이스를 설계하는 기법이다. 트리구조를 시각화 하여 테스트 케이스를 설계하므로 불필요한 중복 및 누락을 회피할 수 있다.

[예제]



【그림 Ⅱ-11. 표 만들기 예제】

위의 표 만들기 구조를 분류 트리 기법을 적용하여 분석하면 아래와 같이 시각화 하여 나타낼 수 있다.



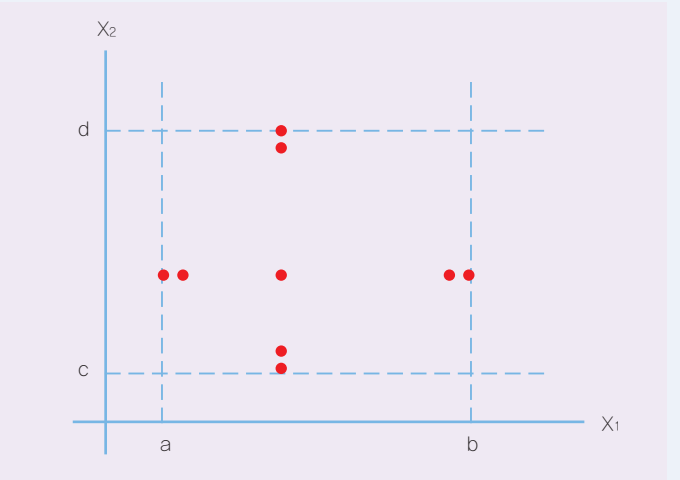
【그림 Ⅱ-12. 분류 트리 기법 예제】

• 경계값 분석(Boundary Value Analysis)

Partitioning을 이용하여 입출력 도메인을 equivalence class로 나누었을 때, 각 범위의 경계 값에서 결함이 발생하는 경우가 많다는 사실에 착안하여, 결함 검출 가능성을 높일 수 있도록 테스트 케이스를 설계하는 기법이다. 즉, equivalence class 안에서 테스트 케이스를 선정할 때, 임의의 데이터를 이용하는 것 대신에 class의 경계에 있는 데이터를 이용한다. 이 방법의 경우, 입출력의 수많은 변형이 존재할 수 있기 때문에 많은 수의 테스트 케이스를 요구 받게 되므로, 각 입력 변수를 위해 최소한 9개의 테스트 케이스를 선정해야 한다. 또 복잡한 계산을 요구하는 문제의 경우 equivalence class의 범위를 정의하는 것이 어려우므로 가능한 한 상세한 요구명세가 필요하다는 제한이 있다.

[예제]

조건이 a≤ X_1 ≤b 이고, c≤ X_2 ≤d 일 때, 각 범위의 경계값을 분석하여 아래와 같이 입력값을 산정할 수 있다.



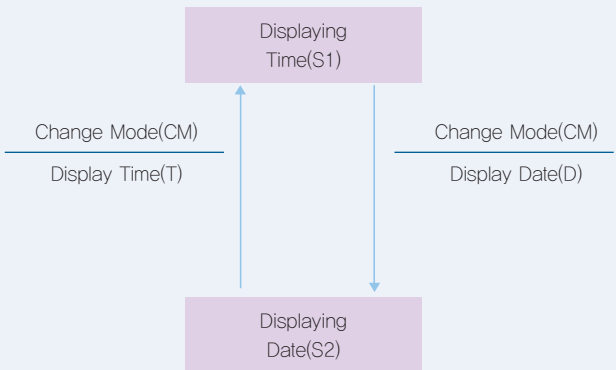
【그림 Ⅱ-13. 경계값 분석 예제】

• **상태 전이 테스팅(State Transition Testing)**

시스템의 모든 입출력 상태를 식별하고, 도달 가능한 모든 상태의 입력 조합을 포함하는 상태 전이 테이블을 정의한 후, 테스트 케이스를 설계한다.

[예제]

디스플레이의 TIME 모드와 DATE 모드의 상태 전이 테스트 케이스를 아래와 같이 설계할 수 있다.



Case	1	2
Start State	S1	S2
Input	CM	CM
Expected Output	D	T
Finish State	S2	S1

【그림 II-14. 상태 전이 테스팅 예제】

• **결정 테이블 테스팅(Decision Table Testing)**

요구사항의 논리와 발생 조건에 따라서 생성될 수 있는 결과를 테이블의 형태로 나열한 것으로, 조건과 행동의 모든 가능한 조합을 고려하여 테스트 케이스를 생성하는 기법이다.

[예제]

모든 사람은 350의 공제를 받는다. 단, 근무 이력이 있고, 나이가 40세를 초과하면 추가로 100의 공제를 받는다. 또한 앞의 조건과 상관없이 자녀가 4명이라면 50의 추가 공제를 받는다. 각 공제 조건에 따른 결과를 테이블 형태로 나열할 수 있다.

【그림 II-15. 결정 테이블 테스팅 예제】

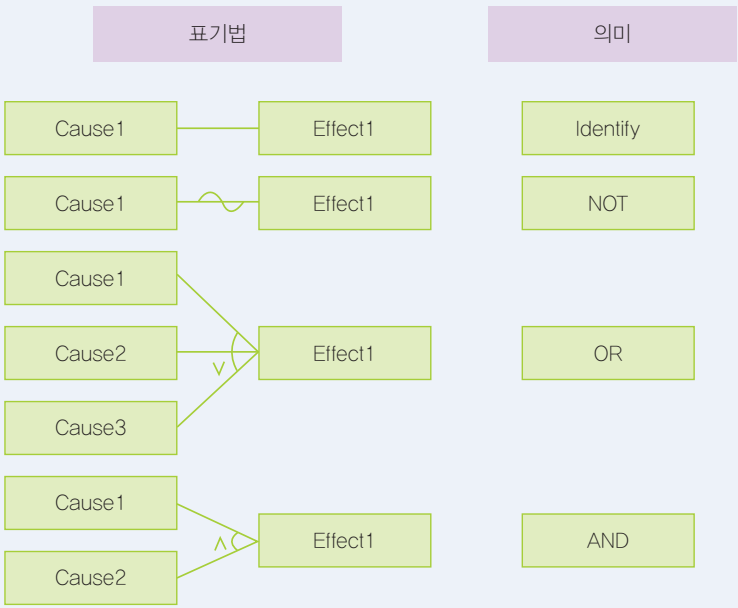
Condition	Case_1	Case_2	Case_3	Case_4	Case_5
근무이력	Y	Y	Y	Y	N
나이 〉 40	Y	Y	Y		Y
나이 〈 40				Y	
나이 = 40					
자녀 = 4	Y			Y	
자녀 〉 4		Y			
자녀 〈 4			Y		
공제금액					
350	X	X	X	X	X
100	X	X	X		
50	X			X	
총 공제금액	500	450	450	400	350

• **원인-결과 분석(Cause-Effect Graphing)**

논리적으로 테스트 케이스를 생성하기 위해, 원인과 결과에 근거하여 테스트 케이스를 생성하는 방법으로, 시스템의 외부 동작만을 고려한다. 원인은 시스템의 내부 변화의 입력 상태를 나타내고, 결과는 시스템의 변환 또는 원인의 조합으로 인한 출력 상태를 나타낸다.

[예제]

원인과 결과에 대한 표기법을 아래와 같이 나타낼 수 있다.



【그림 II-15. 원인-결과 분석 기호 예제】

• **조합 테스트 기법(Combinatorial Test Techniques)**

잠재적 조합 요소의 거대한 양을 처리하기 위한 테스트케이스를 선정하는데, 도움을 주는 통계적 테스트 기법이다. 요구되는 테스트 케이스의 이상적인 개수를 계산하고, 입력값과 조건의 차를 식별하여, 테스트 케이스 선정 프로세스에서 최대 커버리지를 가지는 최소 개수의 테스트 케이스를 제공하는데 도움을 준다.

[예제]

파랑, 노랑, 빨강을 가질 수 있는 3개의 입력 값이 있을 경우, 조합 테스트 기법을 이용하여 다음 표와 같이 3^3 즉, 27개의 테스트 조합을 만들 수 있다.

【표 II-16. 조합 테스트 기법 예제】

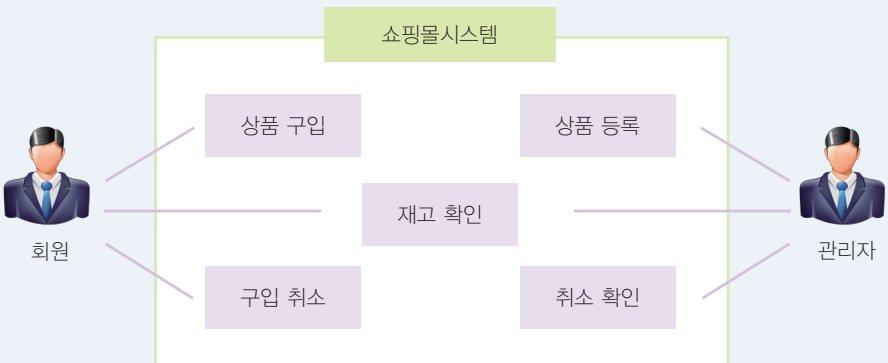
Case	입력값1	입력값2	입력값3	Case	입력값1	입력값2	입력값3
1	파랑	파랑	파랑	15	노랑	노랑	빨강
2	파랑	파랑	노랑	16	빨강	노랑	파랑
3	파랑	파랑	빨강	17	빨강	노랑	노랑
4	노랑	파랑	파랑	18	빨강	노랑	빨강
5	노랑	파랑	노랑	19	파랑	빨강	파랑
6	노랑	파랑	빨강	20	파랑	빨강	노랑
7	빨강	파랑	파랑	21	파랑	빨강	빨강
8	빨강	파랑	노랑	22	노랑	빨강	파랑
9	빨강	파랑	빨강	23	노랑	빨강	노랑
10	파랑	노랑	파랑	24	노랑	빨강	빨강
11	파랑	노랑	노랑	25	빨강	빨강	파랑
12	파랑	노랑	빨강	26	빨강	빨강	노랑
13	노랑	노랑	파랑	27	빨강	빨강	빨강
14	노랑	노랑	노랑	–	–	–	–

• **시나리오 테스팅(Scenario Testing)**

비즈니스 시나리오 또는 프로세스 흐름을 기반으로 테스트 케이스를 설계하는 방법이다. 시스템을 실제로 사용할 때 존재할 수 있는 결함을 발견하는데 유용하기 때문에, 인수 테스트를 설계하는데 유용하다.

[예제]

쇼핑몰 시스템의 경우 아래와 같은 다이어그램으로 나타낼 수 있다.



【그림 II-16. 시나리오 테스팅 예제】

다이어그램 안에 있는 상품 구입, 상품 등록, 재고 확인, 구입 취소, 취소 확인 시나리오에 대하여 Actor(회원/관리자)와 시스템의 교환을 스토리로 기술한다.

• **오류 추정(Error Guessing)**

Ad-hoc Testing이라고도 불리며, 특정한 소프트웨어가 주어졌을 때, 직관과 경험의 의하여 어떤 특정한 형태의 결함이 발생할 것이라고 예측하고, 이 결함을 드러내 주는 테스트 케이스를 설계하는 기법이다. 직관적인 프로세스이기 때문에 특정한 프로시저를 가지는 것이 어렵다. 결함이 발생할 수 있는 상황들을 리스트로 열거해보고, 이 리스트를 테스트 케이스로 활용한다.

[예제]

정렬 프로그램에서 에러가 발생하기 쉬운 경우는 아래와 같으며, 해당 상황을 고려하여 테스트 케이스를 선정할 수 있다.

(1) 입력 리스트가 공란 리스트인 경우

(2) 입력 리스트가 하나의 원소만을 갖는 경우

(3) 입력 리스트의 모든 원소가 같은 값을 가지는 경우

(4) 입력 리스트가 이미 정렬되어 있는 경우

구조기반 기법

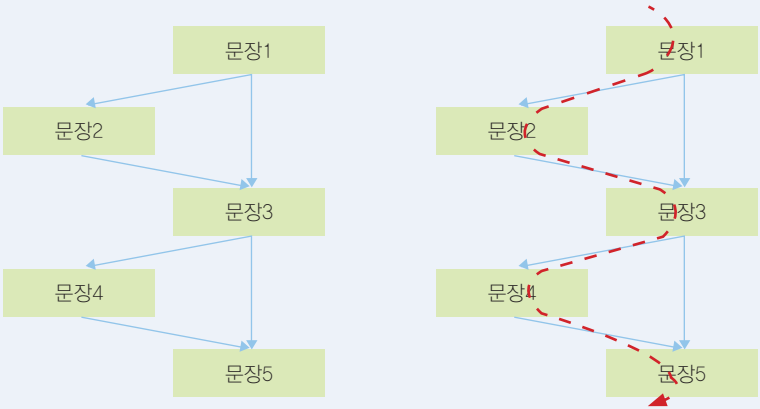
코드와 개발 설계 등의 소프트웨어 구현 정보를 기반으로 테스트 케이스를 설계하는 기법이다.

• **구문 테스팅(Statement Testing)**

프로그램 내의 모든 문장들을 한번 이상 수행하도록 테스트 케이스를 설계하는 기법이다.

[예제]

다음과 같은 제어 흐름도를 커버하는 구문 테스팅 테스트 케이스를 설계하기 위해서는 제어 흐름도의 모든 문장들을 통과하는 1개의 테스트 케이스가 필요하다.



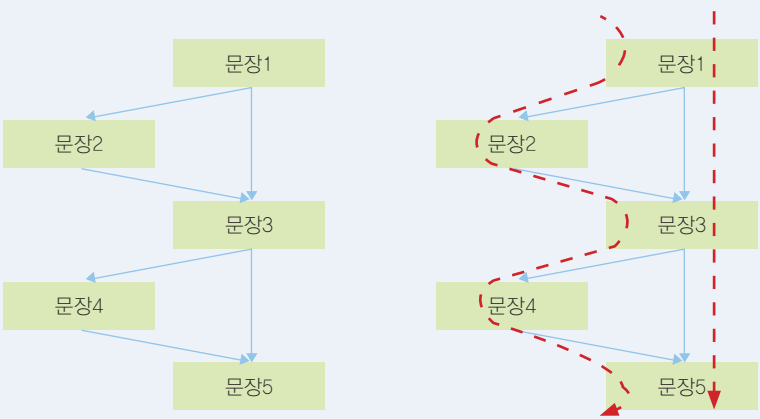
【그림 II-17. 구문 테스팅 예제】

• **결정 테스팅(Decision Testing)**

프로그램 내의 각 분기들을 한번 이상 수행하도록 테스트 케이스를 설계하는 기법이다.

[예제]

다음과 같은 제어 흐름도를 커버하는 결정 테스팅 테스트 케이스를 설계하기 위해서는 제어 흐름도의 모든 분기들을 통과하는 2개의 테스트 케이스가 필요하다.



【그림 II-18. 결정 테스팅 예제】

• **조건 테스팅(Condition Testing)**

프로그램 내의 각 조건들을 보장하기 위해 조건들이 참이 되는 경우와 거짓이 되는 경우를 모두 수행하도록 테스트 케이스를 설계하는 기법이다.

[예제]

다음과 같은 조건을 커버하는 조건 테스팅 테스트 케이스를 설계하기 위해서는 3개의 테스트 케이스가 필요하다.

```
if(A>1 AND B<=0){
   A=B+4;
 }
```

【표 II-17. 조건 테스팅 예제】

	A=2, B=-2	A=0, B=-2	A=0, B=2
A>1	TRUE	FALSE	FALSE
B<=0	TRUE	TRUE	FALSE
A>1 AND B<=0	TRUE	FALSE	FALSE

• **데이터 흐름 테스팅(Data Flow Testing)**

프로그램 내에서 변수들이 값을 할당 받은 지점이나 사용된 지점에 따라, 프로그램의 테스트 경로들을 선택하는 방법이다.

【표 Ⅱ-18. 데이터 흐름 테스팅 예제】

구 분	내 용
all-node	• 프로그램의 모든 node(statement)가 포함되도록 선정한다.
all-edge	• 프로그램의 모든 edge(statement의 흐름 또는 decision에 의한 branch등)가 포함되도록 선정한다.
all-defs	• 한 노드에서 정의된(assigned) 특정 변수가 그 값이 변하기 전에 적어도 한번은 활용될 수 있도록 path를 선정하고, 이에 따른 테스트 케이스를 추출한다. 노드는 프로그램 전체로 확장한다.
all-p-use (all-predicate-use)	• 한 노드에서 정의된 특정 변수가 그로부터 존재하는 모든 dpu의 원소(값이 바뀌지 않고 predicate use로 사용된 노드들)까지의 path가 존재하도록 선정한다. 노드는 프로그램 전체로 확장한다.
all-c-use (all-computation-use)	• 한 노드에서 정의된 특정 변수가 그로부터 존재하는 모든 dcu의 원소(값이 바뀌지 않고 computational use로 사용된 노드들)까지의 path가 존재하도록 선정한다. 노드는 프로그램 전체로 확장한다.
all-c-use/ some-p-use	• All-c-use를 적용시키되, c-use가 존재하지 않는 variable에 대해서는 p-use를 적용시키는데 이때는 all이 아닌 some으로 적용시킨다.
all-p-use/ some-c-use	• All-p-use를 적용시키되, p-use가 존재하지 않는 variable에 대해서는 c-use를 적용시키는데 이때는 all이 아닌 some으로 적용시킨다.
all-uses	• 한 노드에서의 variable에 대해 dcu, dpu를 만족시키는 path를 모두 고려하도록 테스트 케이스를 정한다.
app-paths	• 프로그램의 모든 path를 포함시켜서 테스트 케이스를 선정한다.

III. 공개SW 테스트

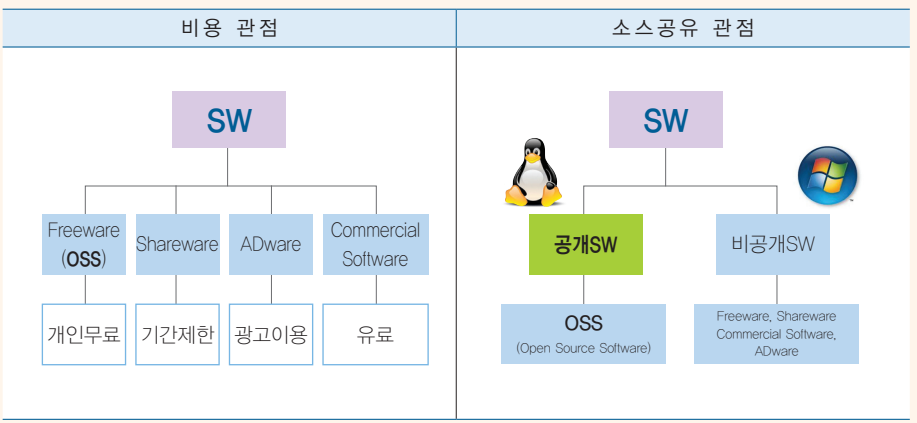
1. 공개SW 소개

공개SW 정의

공개SW는 저작권자가 소스코드를 공개하여 누구나 자유롭게 사용하고, 수정 및 재배포할 수 있는 자유로운 소프트웨어이다. 단, 재배포시에는 저작권자에 의한 라이선스 규정을 준수해야 한다.

공개SW 이해

공개SW는 "공개"라는 용어로 인해 프리웨어[12] 및 셰어웨어[13] 등과 혼돈하기 쉽지만, 공개SW는 비용의 관점이 아닌 소스의 공유 관점에서 이해해야 한다.



【그림 III-1. 공개SW 관점의 이해】

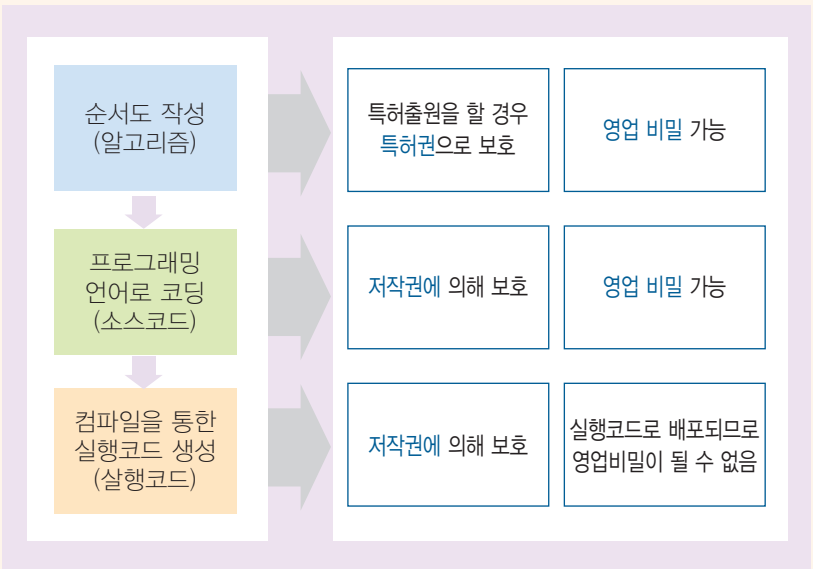
12) 저작권자가 대가없이 무료로 사용하도록 제작한 소프트웨어
13) 처음에는 누구나 사용할 수 있지만 일정한 기간 이후에 사용하려면 대가를 지불해야 하는 소프트웨어

공개SW 라이선스

공개SW 라이선스란 공개SW 개발자와 이용자 간에 사용 방법 및 조건의 범위를 명시한 계약이다.

• 공개SW 라이선스와 법적 관계

공개SW를 이용하기 위해서는 개발자가 규정한 라이선스를 지켜야 하며, 이를 위반할 경우에는 라이선스 위반 및 저작권 침해가 발생하고, 이에 대한 처벌을 받게 된다.



【그림 III-2. 공개SW 라이선스와 법적 관계】

2012년 현재 공개SW 라이선스의 인증을 관장하고 있는 OSI에 따르면, 69개의 라이선스가 공개SW 라이선스로 등록되어 있으며, 공개SW의 약 70% 이상이 GPL[14]와 LGPL[15] 라이선스를 사용하고 있다.

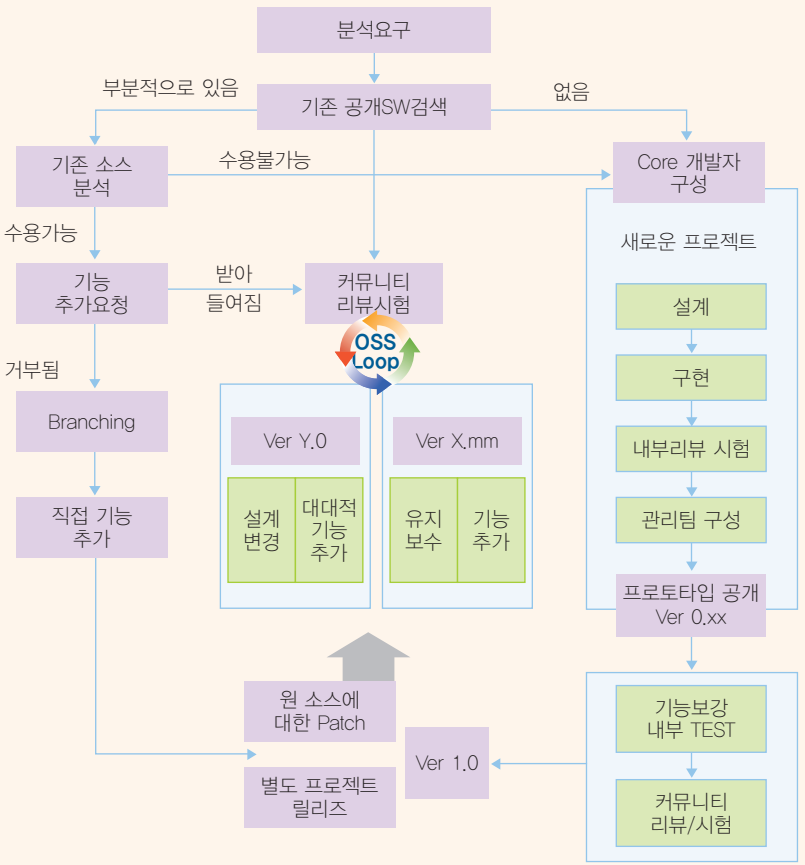
14) GPL(General Public License)은 일반 공중 사용 허가서로 이 라이선스를 가진 프로그램을 사용하여 새로운 프로그램을 만들게 되면 파생된 프로그램 역시 같은 카피레프트를 가져야 한다.
15) LGPL(Lesser General Public License)은 약소 일반 공중 사용 허가서로 카피레프트에 대한 규제를 프로그램 자체에 두나 이 프로그램을 사용하는 다른 프로그램에 대해서는 카피레프트를 두지 않는다.

2. 공개SW 프로세스

공개SW 프로젝트 생명주기

공개SW 프로젝트는 소프트웨어 개발이라는 관점에서는 기존의 많은 상용 소프트웨어와 유사한 생명주기를 갖게 된다. 하지만 공개SW 프로젝트에서는 '공개'라는 단어가 함축하듯이 여러 명의 개발자가 참여하는 분산 개발, 기존에 공개되어 있는 많은 소프트웨어 자원의 이용, 다양한 부류의 자원자들에 의한 소프트웨어 리뷰 및 시험 과정, 기술 지원 방법, 기능의 확장, 새로운 프로젝트로의 가지치기(Branching) 과정 등이 비공개 소프트웨어와 달리 매우 중요한 의미를 가지게 된다.

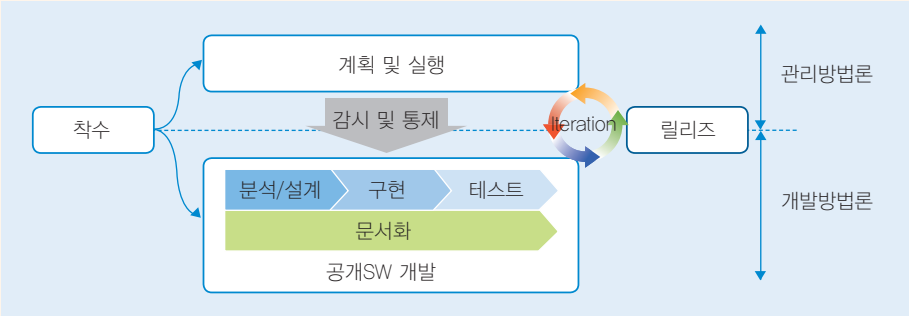
다음의 그림은 공개SW 프로젝트의 생명주기를 포괄하고 있다.



【그림 III-3. 공개SW 프로젝트 생명주기】

공개SW 프로세스

공개SW기반 개발에 있어서 일반적으로 적용되는 프로세스는 다음과 같다.



【그림 III-4. 일반적인 공개SW기반 방법론】

공개SW 프로세스는 소프트웨어공학 관점에서 비슷한 절차에 따라 수행되고 있으나, 공개SW에 많은 영향을 주고 있는 분산 네트워크에서 다양한 동기를 가진 자발적 참여자들의 복잡한 상호작용을 통해 이루어지는 커뮤니티의 특성에 따른 방법론을 적용하는 것을 고려하여야 한다.

• 공개SW 테스트 활동

공개SW 프로세스에서 수행할 수 있는 테스트 활동은 표준화 테스트, 소스 리뷰, 기능/비기능 테스트, 라이선스 검증으로 분류할 수 있으며 상세 내용은 다음과 같다.

【표 Ⅲ-1. 공개SW 테스트 활동 상세 내용】

분 류	테스트 종류	내 용
표준화 테스트	LSB[16] (Linux Standard Base)	• LSB 인증[17] 서비스를 받고자 하는 기관/기업에게 LSB 인증 프로세스 정의 및 수행방안 제시, 도구 및 활용정보 제공
	LTP[18] (Linux Test Project)	• 리눅스 운영체제에 대해 Tool을 활용한 리눅스의 신뢰성, 호환성, 안정성 테스트 수행
소스리뷰	피어리뷰	• 같은 분야의 전문가들이 저자의 결과물을 심사하는 과정
	소스 인스펙션	• 완성된 코드에 대한 검토를 통해서 코드 상에 존재하고 있는 잠재적인 문제를 발견하는 과정
기능/비기능 테스트	자동화 도구	• 테스트 도구를 활용한 기능/비기능 테스트 수행 – 기능 테스트(기능별 정확성 시험, 다양한 조합의 테스트 등) – 비기능 테스트(Stress Test/Endurance Test/Usability Test 등)
	시나리오 기반 테스트	• 시나리오 기법을 적용한 테스트 케이스를 이용하여 기능/비기능 테스트 수행
라이선스 검증	CodeEye	• 한국저작권 위원회에서 개발하고 서비스하는 공개SW 라이선스 검증 도구
	Protex	• 블랙덕소프트웨어에서 개발한 오픈소스저작권관리 소프트웨어
	FOSSology	• 공개SW 라이선스 관련 문자열을 검색하기 위한 공개SW

16) 수많은 리눅스 배포판들이 각각의 기능과 환경을 적용하여 개발됨에 따라 다른 많은 애플리케이션이 개발되어 활성화 되는데 저해요소가 됨으로 이러한 저해요소를 해결하기 위한 방안으로 설립

17) LSB 인증을 취득한 제품은 LSB 인증 Trademark를 사용할 수 있다. 이 마크는 개발자와 사용자가 LSB 인증된 배포판에서 LSB 인증된 애플리케이션을 사용할 경우 문제가 없다는 것을 보증한다.

18) LTP는 리눅스의 신뢰성, 호환성, 안정성을 입증하여 오픈소스 커뮤니티에 테스트 슈트를 제공하는 것이 주목적인 오픈소스 프로젝트이며, LTP 테스트 슈트는 리눅스 운영체제에 대해 다양한 관점에서 테스트를 수행할 수 있도록 개발된 자동, 반자동 테스트 슈트들의 집합체이다.

3. 공개SW 테스트

국내 공공기관이나 기업 등 공개SW를 도입하여 활용하는 사례가 크게 증가하고 있으며, 공개SW를 기반으로 새로운 SW를 만들어 이용하거나 상업적으로 제공하는 경우도 늘어나고 있어, 공개SW에 대한 기능 및 성능관련 품질에 대한 요구사항이 증대되었다.

이에 공개SW 역량프라자에서는 관련 체계 및 장비를 구비하여 공개SW 테스트에 대한 요청이 있을 경우 테스트 서비스를 신청 받아 지원하며, 신뢰성 있는 공개SW 발굴을 위한 3단계 로드맵을 수립하여 공개SW에 대한 테스트를 수행한다.

공개SW 테스트 서비스 지원

테스트 서비스 신청은 공개SW 포털(http://www.oss.kr) 웹사이트에 접속 후 '주요지원사업 ► 공개SW 역량프라자 운영 ► 공개SW 테스트'에서 신청할 수 있으며, 테스트 지원 절차는 아래와 같다.

요 청	검 토	테 스 트	분 석	보 고
· 공개SW포탈 사이트 (oss.kr) 테스트 신청서 작성	· 신청서 검토 · 사전 문제점 분석	· 공개SW 테스트툴 적용	· 테스트 결과 분석 · 오류 내용 별도 분류	· 테스트 결과서 작성 · 결과서 통보

【그림 III-5. 테스트 지원 절차】

공개SW Stack 통합 테스트

공개SW 역량프라자에서는 신뢰성 있는 공개SW 발굴을 위해 공개SW 관련 선정지표를 활용하여 공개SW를 선정하고, 체계적인 테스트를 수행하여 공개SW의 신뢰성을 제고한다.

• 공개SW 테스트 로드맵

공개SW 역량프라자 테스트 로드맵은 총 3단계로 구성된다. 1단계는 정보시스템 분야에서 즉시 도입 가능한 공개SW 발굴, 2단계는 클라우드 컴퓨팅 솔루션 분야 확대, 그리고 3단계는 모바일을 포함한 임베디드 분야 솔루션에 대한 신뢰성 및 통합 테스트를 진행하여 공개SW 기술참조모델(Technical Reference Model)[19]을 구축하는 것이 목표이다.

【표 III-2. 공개SW 테스트 로드맵】

단계	년도	목 표	주 요 활 동
1단계	2010년	• 공개SW Stack 미들웨어 제품 발굴	• 활용도 높은 공개SW 발굴을 위한 WEB, WAS, DB 등 대표적인 공개SW 20개 선정 및 테스트 수행
	2011년	• 공공기관 및 중소기업에서 활용 가능한 정보시스템 분야 공개SW 발굴	• CMS, LMS, EDM 등 정보시스템 분야 대상 발굴 및 테스트 수행 ※ 정부통합전산센터 클라우드 전환 정보시스템 테스트 수행
2단계	2012년	• 클라우드 컴퓨팅 분야 공개SW 발굴	• 클라우드 컴퓨팅 분야 신규 발굴 • 가상화(Virtualization)와 분산 파일 시스템(Distributed File System) 등 테스트 수행
3단계	2013년	• 클라우드 분야 확대 및 임베디드 분야 공개SW 발굴	• 클라우드 관리 및 관제 솔루션 분야 등 테스트 대상 확대 • 임베디드 솔루션 공개SW 발굴 및 테스트 수행

19) 공개SW 기술참조모델은 공개SW를 도입하고 활용하는데 필요한 기술적인 부분을 지원하는 도구로써, 공개SW 개발 및 공개SW 기반의 정보시스템 구축 또는 전환에 필요한 정보기술 및 표준들을 분류하고 관련 공개SW 기반의 솔루션 정보를 제공한다.

• 공개SW 테스트 현황

공개SW 역량프라자에서 1단계 로드맵 진행 결과 발굴된 미들웨어와 정보시스템분야에서 발굴된 공개SW는 아래와 같다. 공개SW 발굴은 다음 장에 소개하는 공개SW 테스트 프로세스의 절차에 따라 진행된다.

【표 III-3. 2010/2011년 테스트 현황】

년 도	그 룹 명	제품명	버 전
2010년	WAS (Web Application Server)	JBoss	5.1.0
		Resin	4.0.6
		Tomcat	6.0.26
		GlassFish	3.0.1
	WEB (Web Server)	Apache	2.2.15
		lighttpd	1.4.26
		Nginx	0.7.65
	DBMS (Database Management Server)	MySQL	5.1.48
		CUBRID	8.2.2
		PostgreSQL	8.4.4
	CRM (Customer Relationship Management)	SugarCRM	5.5.2
		ConcourseSuite	6.0
	ERP (Enterprise Resource Planning)	Openbravo	2.5
		SQL-Ledger	2.8
	Groupware	EGroupware	1.6
		Zimbra	6.0.7
	SMS (Server Management System)	Nagios	3.2.1
		Hyperic HQ	4.4.0
	Side Business	PeaZip	3.1
		WinDirStat	1.1.2
2011년	CMS (Content Management System)	XE	1.4.4.4
		KimsQ	1.0
	EMR (Electronic Medical Record)	OpenEMR	4.0
		OpenMRS	1.7.1
	EDM (Electronic Document Management)	Alfresco	3.4.d
		OpenKM	5.1.3
	LMS (Learning Management System)	Moodle	2.0.2
		Genie	1.2.

4. 공개SW 테스트 프로세스

공개SW 테스트 프로세스 소개

공개SW 프로젝트의 경우 비교적 폐쇄적인 초기 개발 단계를 거쳐 공개된 뒤에, 커뮤니티와 호흡하는 공개SW 순환구조에 들어간다. 이러한 공개SW의 프로젝트 특성에 따라 공개SW 역량프라자에서는 시스템이 사용자의 요구사항에 맞게 동작하는지 확인하는 기능 테스트와 동시 단말 사용자가 동시에 거래를 발생시키도록 하여 시스템의 상황을 점검하는 Spike 성능시험, 오랫동안 거래를 발생 시켰을 때의 시스템 상황을 점검하는 가용성 성능시험을 진행한다.

공개SW 테스트 수행절차는 다음과 같으며, 공개SW 역량프라자에서 수행하는 모든 테스트 활동에 기반이 된다.

【표 III-4. 공개SW 테스트 프로세스】

절 차	주 요 내 용
대상 선정	• 분야별 대체 가능한 공개SW 목록 조사 • 선정지표 기반 평가를 통한 대상 선정 ※ 공개SW 포털을 통한 테스트 신청은 대상이 정해져 있으므로 대상 선정 절차를 수행하지 않음
분석/설계	• 테스트 대상 매뉴얼 및 소프트웨어 구조 파악 • 테스트 케이스 및 시나리오 설계 • 테스트 도구 확보
테스트 실행	• 테스트 환경 구축 • 계획된 절차에 따라 테스트 수행 (기능/비기능) • 예상 결과와 실제 결과 비교 및 결함 도출
테스트 종료	• 테스트 결과 분석 및 보고서 작성 • 공개SW 역량프라자 포털에 보고서 게시

대상 선정

수많은 공개SW 중 활용도 높은 공개SW를 선정하기 위해서는 객관적인 기준이 필요하다. 이에 공개SW 역량프라자에서는 공개SW 커뮤니티 및 기업에서 제공하는 다양한 정보를 기반으로 선정지표(표 Ⅲ-5. 공개SW 선정지표 참조)를 개발하였다.

선정지표는 문서, 지원서비스, 제품, 커뮤니티 4개의 카테고리로 분류하였으며, 세부 항목으로 51개 평가항목(표 Ⅲ-6. 평가항목 참조)을 선정하여 공개SW 대상선정을 위한 평가 자료로 활용된다.

• 선정지표

【표 Ⅲ-5. 공개SW 선정지표】

구 분	항 목	평 가 항 목
문서 (Document)	기능성	접근통제
		시스템 로그
		상호운용성
		표준준수
	신뢰성	가용성
		오류허용성
		장애처리
		백업처리
	사용성	사용자 인터페이스
		에러 메시지
		한글지원
		문서식별(버전)
		문서포맷
		목차
		색인
	효율성	시간효율성
		처리효율성
		자원효율성
	유지보수성	시스템감시(모니터링)
		환경정보 설정
		유지보수 절차
	이식성	소프트웨어 지원환경
		하드웨어 지원환경
		호환시스템 정보
		설치/삭제 가이드

구 분	항 목	평 가 항 목
문서 (Document)	이식성	소프트웨어 지원환경
		하드웨어 지원환경
		호환시스템 정보
		설치/삭제 가이드
지원서비스 (Support)	기술지원	메일링리스트
		버그리스트
		포럼(Q&A, FAQ)
		24X7
		국내벤더
	교육	온라인교육
		전문서적
		국내 교육 기관
제품 (Product)	품질활동	QA조직 운영
		BTS(Bug Tracking System)
		릴리즈노트(Bug Fix 포함)
		소스형상관리
	소프트웨어	개요(소프트웨어 소개 및 주요특징)
		라이선스 OSI 인증
		상용 라이선스 정책
		소프트웨어 localization(한글) 지원
		로드맵
		버전 (1.0 이상 안정화 버전)
		개발상태
		지속성(릴리즈 주기)
커뮤니티 (Community)	사용자커뮤니티	포럼(사용자 간의 정보 공유 창구)
		국내 커뮤니티
		활동성
	개발자커뮤니티	포럼(개발자 간의 정보 공유 창구)
		국내 커뮤니티
		활동성

※ 본 선정기준에 의해 선정된 공개SW 대상이 품질 및 성능의 우수성을 뜻하는 것은 아님

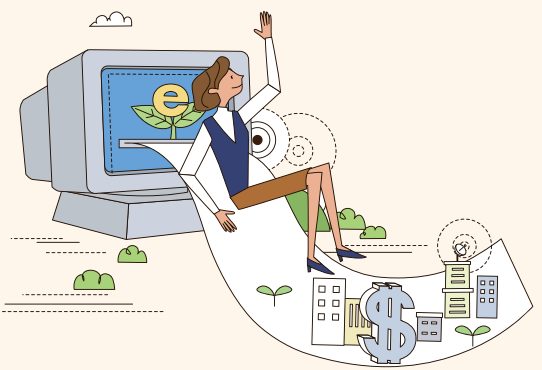
• 평가항목

【표 Ⅲ-6. 평가항목】

구 분	항 목	평 가 항 목	평 가
Document (25)	기능성	접근통제	S : 정보제공 P : N/A N : 없음
		시스템 로그	S : 정보제공 P : N/A N : 없음
		상호운용성	S : 정보제공 P : N/A N : 없음
		표준준수	S : 정보제공 P : N/A N : 없음
	신뢰성	가용성	S : 정보제공 P : N/A N : 없음
		오류허용성	S : 정보제공 P : N/A N : 없음
		장애처리	S : 정보제공 P : N/A N : 없음
		백업처리	S : 정보제공 P : N/A N : 없음
	사용성	사용자 인터페이스	S : 모두 제공 P : 부분적 제공(세부화면 누락) N : 해당사항 없음
		에러 메시지	S : 정보제공 P : N/A N : 없음
		한글지원	S : 정보제공 P : N/A N : 없음
		문서식별(버전)	S : 정보제공 P : N/A N : 없음
		문서포맷	S : 모두 제공 P : 부분적 제공(html or pdf) N : 해당사항 없음
		목차	S : 정보제공 P : N/A N : 없음
		색인	S : 정보제공 P : N/A N : 없음
	효율성	시간효율성	S : 정보제공 P : N/A N : 없음
		처리효율성	S : 정보제공 P : N/A N : 없음
		자원효율성	S : 정보제공 P : N/A N : 없음
	유지 보수성	시스템감시	S : 자체기능제공 P : 플러그인을 통한 타 시스템 연계 N : 없음
		환경정보 설정	S : GUI(관리자화면포함) P : CLI(콘솔기반) N : 없음
		유지보수 절차	S : 정보제공 P : N/A N : 없음
	이식성	소프트웨어 지원환경	S : 정보제공 P : N/A N : 없음
		하드웨어 지원환경	S : 최소/권장 사양 P : 일부(최소 or 권장) 사양 N : 없음
		호환시스템 정보	S : 정보제공 P : N/A N : 없음
		설치/삭제 가이드	S : 설치/삭제 가이드 P : 일부(설치 or 삭제) 가이드 N : 없음
Support (25)	기술 지원 (20)	메일링리스트	S : 정기 제공 P : 비정기 제공 N : 없음 (최근 6개월 or 릴리즈 기준)
		버그리스트	S : 정보제공 P : N/A N : 없음
		포럼(Q&A, FAQ)	S : 응답율 80% 이상 P : 50% ~ 80% N : 기타 (최근 3개월 기준)
		24X7	S : 무상제공 P : 유상제공 N : 없음
		국내벤더	S : 존재 P : N/A N : 없음

구 분	항 목	평 가 항 목	평 가
Support (25)	교육 (5)	온라인교육	S : 정기교육과정 P : 동영상(비정기적) N : 없음
		전문서적	S : 한글서적 P : 영문서적 N : 없음
		국내 교육 기관	S : 존재 P : N/A N : 없음
Product (30)	품질 활동 (10)	QA조직 운영	S : 존재 P : N/A N : 없음
		BTS	S : 사용 P : N/A N : 없음
		릴리즈노트	S : 신규기능 및 결함수정내역포함 P : 신규기능정보 N : 없음
		소스형상관리	S : 사용 P : N/A N : 없음
	소프트웨어 (20)	개요	S : 소개 및 주요특징 P : 소개 N : 없음
		라이선스 OSI 인증	S : OSI호환 P : 자체 라이선스(OSI 미호환) N : 없음
		상용 라이선스 정책	S : 존재 P : N/A N : 없음
		한글 지원	S : 모두 지원 P : 일부 지원(메뉴) N : 없음
		로드맵	S : 정보제공 P : N/A N : 없음
		버전	S : 메이저버전 P : 마이너버전 N : 버전 없음
		개발상태	S : Production, 신규 Beta P : Beta N : 기타
		지속성	S : 릴리즈 주기 6개월 이하 P : 6개월 이상 ~ 1년 이하 N : 1년 이상
Community (20)	사용자 커뮤니티 (10)	포럼	S : 존재 P : N/A N : 없음
		국내 커뮤니티	S : 존재 P : N/A N : 없음
		활동성	S : 정보갱신 1개월 이하 P : 1개월 이상 ~ 3개월 이하 N : 기타
	개발자 커뮤니티 (10)	포럼	S : 존재 P : N/A N : 없음
		국내 커뮤니티	S : 존재 P : N/A N : 없음
		활동성	S : 정보갱신 1개월 이하 P : 1개월 이상 ~ 3개월 이하 N : 기타

※ 각 항목은 S(Satisfied, 만족) 1점, P(Partially Satisfied, 부분만족) 0.5점, N(Not Satisfied, 불만족) 0점으로 평가하며 각 항목에 대한 가중치를 곱하여 총점을 계산

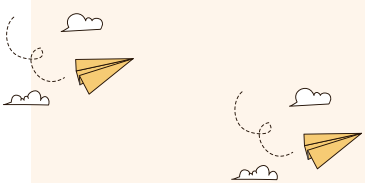


• **평가결과(예시)**

【표 Ⅲ-7. 시스템SW 분야 평가결과 예시】

구 분	대 상	항목(배점)				총점 (100)
		Document (25)	Support (25)	Product (30)	Community (20)	
WEB	Apache	21.5	21.7	22.5	13.3	79
	Lighttpd	16.5	12.8	22.5	13.3	65.2
	NginX	20	15.7	20	13.3	69
WAS	JBoss	20	22.2	27.5	16.7	86.3
	Resin	18	15.7	20	10	63.7
	Tomcat	21	19.7	25	13.3	79
DB	Cubrid	20.5	23	25	20	88.5
	MySQL	21	23	25	18.3	87.3
	PostgreSQL	20.5	19.7	22.5	6.7	69.3
OS	CentOS	15.5	21.3	20	6.7	63.5
	OpenSUSE	16	13.7	27.5	16.7	73.8
	SUlinux	14	15	17.5	8.3	54.8
	Ubuntu	18	17.3	25	18.3	78.7

※ 분야별 알파벳순으로 정렬



분석/설계

공개SW 시스템의 특성을 고려하여 적합한 테스트 목표 및 실행 전략, 테스트 범위, 접근 방법, 테스트 수행 시 요구되는 자원(도구 포함)등을 사전에 정의하여 프로젝트 수행 시 원활한 테스트 수행을 위한 사전 작업을 진행한다.

• 테스트 대상 매뉴얼 및 소프트웨어 기능 구조 파악

시스템과 관련된 기능을 분석하고 그 기능과 관련된 테스트 요소를 정의한다. 테스트 수행 시 기능 리스크 분석을 통해 테스트 케이스 설계 시 필요한 요소를 파악할 수 있다. 단, 소프트웨어가 사용하기 쉽지 않다면 처리 결과가 부정확할 수 있다.

【표 III-8. 기능 리스크 분석 상세 예시】

리스크 요소	개발사 리스크					사용자 리스크		
리스크 아이템	추가된 기능	상호관계	기능 중요도	결함 발생률	사용자 취급 중요도	안정적 피해	사용 빈도	외부적 가시성
회원가입	3	5	5	5	9	9	3	9
회원탈퇴	3	3	3	0	1	3	1	0
강의등록	5	9	5	1	0	9	9	5
강의수강	9	9	9	9	9	5	9	5
시험신청	5	9	5	5	3	5	3	5
점수산정	9	9	9	5	9	3	9	5
결과통보	5	5	5	3	3	9	5	5

• 테스트 케이스 및 시나리오 설계

프로젝트 범위 및 상위 요구사항을 분석하여 테스트 범위를 설정한다. 테스트를 통하여 적합성을 검증 받아야 할 테스트 범위는 대상 시스템 전체가 될 수도 있고, 경우에 따라 일부분이 될 수도 있다.

선정된 테스트 범위는 향후 테스트 단계별 상세계획 수립 시, 요구분석 및 설계 단계에서 정의된 각 테스트 요구사항 들을 검토하여 테스트 항목, 테스트할 항목의 특성과 테스트 하지 않을 항목의 특성을 명세화 하는 기반 자료로 활용된다.

【표 III-9. 테스트 케이스 도출 기법 예시】

테스트 단계	테스트 유형	내 용
단위테스트	기능	• 개별 Component의 기능 테스트 • 단위 업무 기능 테스트
	표준준수	• 표준 준수여부 체크
	보안	• 단위 프로그램 내 사용자별 기능 제한 사항 체크
	성능	• 대상 단위업무에 대한 개별 성능 측정
통합테스트	Component통합	• Component 통합 하에서 기능 테스트 : 단위테스트에서 수행되기도 함
	데이터 연계	• 업무 처리 흐름에 따른 데이터 흐름 테스트
	시스템 간 통합	• 타 시스템과의 프로세스I/F 테스트 • 타 시스템간의 데이터 I/F 테스트
	소프트웨어 통합	• 가능한 소프트웨어 구성에 따른 시스템의 운영여부 테스트
	하드웨어 통합	• 시스템 구성 하드웨어 간의 I/F 테스트
	보안	• 사용자별 접근 제한(통합환경)
	시스템 기능	• 단위 테스트와 통합과정에서 이루어졌던 기능 중 시스템 전반에 걸친 주요 기능에 대해 테스트
시스템테스트	성능	• 사용자 수 대비 응답시간 측정
	볼륨	• 시간당 대상 업무에서 처리될 수 있는 최대 건수 측정
	스트레스	• 시스템의 한계상황 하에서의 부족한 리소스 및 시스템 상태 확인
	구성	• 소프트웨어들의 구성, 소프트웨어 환경설정, 하드웨어의 물리적 구성 등에서 최적의 운영환경 발견하기 위한 테스트
	보안	• 시스템 접근 제한에 대한 테스트
	설치	• 시스템의 설치 및 인스톨 프로그램의 설치 테스트

【표 Ⅲ-10. 사용자 시나리오 설계 예시】

순번	대분류	중분류	소분류	시나리오 명	시나리오 개요
1	콘텐츠 관리	파일 삭제	사용자	사용자의 파일 접근 권한	• 일반 사용자가 등록되어 있는 콘텐츠에 접근하여 삭제할 수 있는 권한이 있는지 확인한다.
2			관리자	관리자의 파일 접근 권한	• 관리자가 등록되어 있는 콘텐츠에 접근하여 삭제할 수 있는 권한이 있는지 확인한다.

【표 Ⅲ-11. 테스트 케이스 설계 예시】

순번	테스트 케이스	입력데이터	예상결과
1	• 문서목록보기 Page 목록에서 다른 사용자가 등록한 문서를 클릭	Add_001	Add_001이 등록한 문서로 이동
2	• 문서내용보기 Page에서 수정버튼을 클릭		정보입력 Page로 이동
3	• 등록한 문서의 내용을 변경한 후 등록버튼 클릭	Test_023.txt	내용보기 Page로 이동
4	• 변경한 내용이 적용되었는지 확인		Test_023 적용

【표 Ⅲ-12. 성능 테스트 케이스 설계 예시】

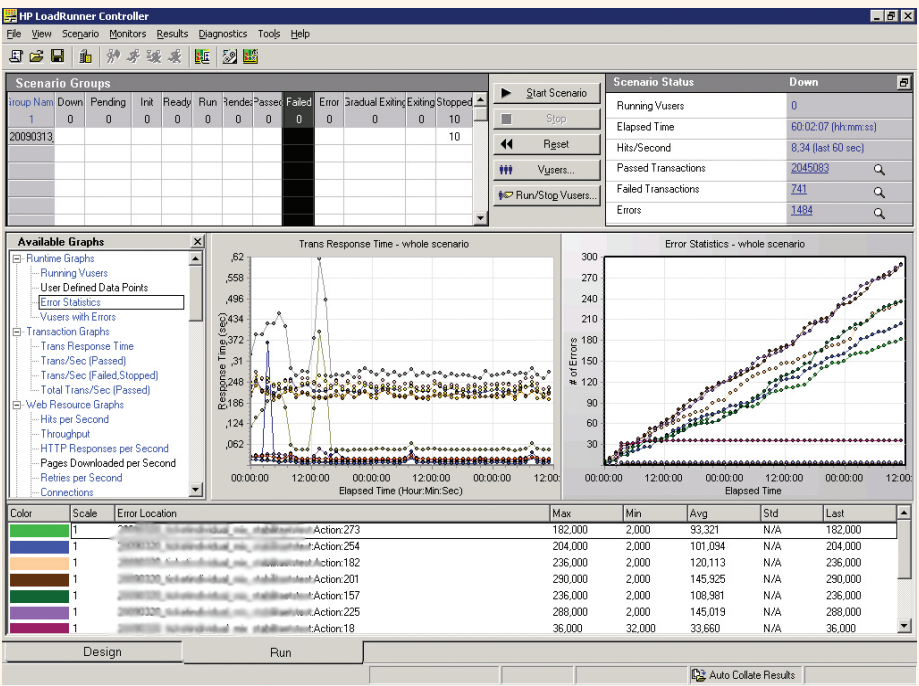
테스트 목표	• 상황을 고려한 Workload Model이 적용된 테스트로써 38.23 TPS를 목표로 함			
목표치	사용자수	10000	호출간격	44.4
	TPS	〉 38.23	응답시간	Read〈1Sec
측정항목	• 거래 성공 비율, TPS, 응답시간, 시스템 사용률			
시나리오	• Ramp-UP Model : (100 Users/3초) * 51초(51초 후에 1000 Users 유입, 측정 시작) • 1000 User 유입 완료 후 2분 동안 부하 발생, 성능 측정 • Ramp-Down Model : (300 Users/초) * 30초(30초 후에 시험 종료) • Think Time Model : 호출 간격(1.4초) – Real 응답 시간			
사전 확인 사항	• 테스트 시작 시간		□ 테스트 종료 시간	
	• 스크립트 수행(N번)		□ 부하 발생기 상태 확인	
	• 데이터베이스 상태 확인		□ 시나리오 확인	
	• Configuration 확인		□ 테스트 데이터 Import	

• 테스트 도구

공개SW 역량프라자에서는 효율적인 비기능 테스트 진행을 위해 TTA와 협력하여 성능 테스트를 수행하며, 다양한 테스트 도구 중 LoadRunner, Jmeter 등의 성능 테스트 도구를 활용한다.

– LoadRunner

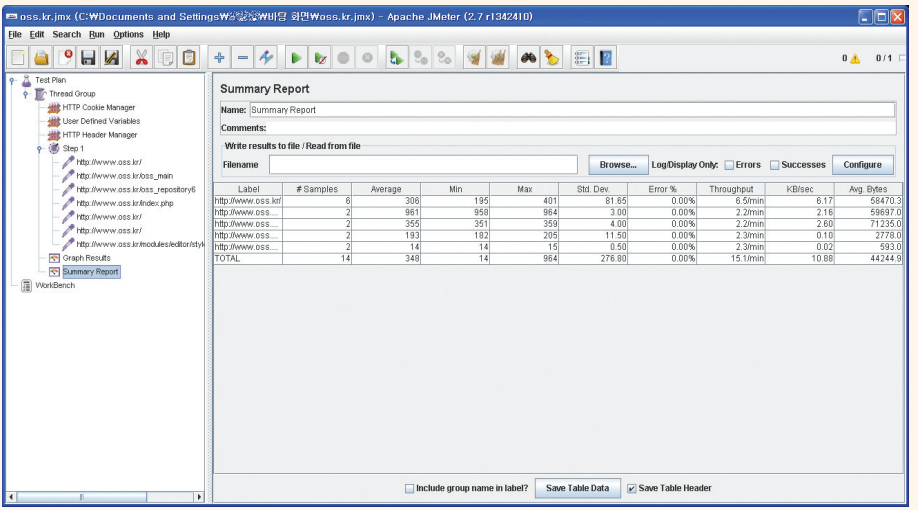
업계 표준의 애플리케이션 부하 테스트 도구로 Web, C/S, SAP, Oracle 등의 다양한 환경의 애플리케이션에 대하여 성능 시험과 부하 시험을 정확하고 효율적으로 진행할 수 있으며 부하 또는 성능 테스트를 진행하는 동안 해당 시스템의 성능과 기능성을 측정, 감시하고 분석하여 성능 개선을 위한 자료를 제공한다.



【그림 III–6. LoadRunner 수행 화면 예시】

– Jmeter

Apache Jakarta 프로젝트의 일환으로 만들어진, 테스트 기능과 퍼포먼스를 측정하는 공개SW로 100% 순수 자바로 작성되었으며, 본래 웹 애플리케이션을 위해 디자인되고 사용되었으나, 현재는 다른 기능의 테스트를 위해 확장되고 있다. static, dynamic 한 모든 자원(file, servlet, perl script, java object, database and query, ftp, soap 등)에 대해 성능을 테스트할 수 있다.



【그림 III-7. Jmeter 수행 화면 예시】

테스트 실행

● 테스트 환경설정

H/W와 지원 S/W(예, 운영 시스템, 데이터베이스, 미들웨어 등)를 포함한 테스트 환경의 접근 권한을 설정한 후 테스트 대상 모듈을 인스톨한다.

【표 Ⅲ-13. 테스트 환경구성 예시】

테스트 대상	Version
Moodle	2.0.2+

구성	OS	WEB	DB
A Stack	CentOS 5.3 (64bit)	Apache2.2.14(PHP:5.3.2)	MySQL 5.1.4
B Stack	CentOS 5.3 (64bit)	Apache2.2.14(PHP:5.3.2)	PostgreSQL 8.4
C Stack	Ubuntu 10.04 (64bit)	Apache2.2.17(PHP:5.3.5)	MySQL 5.0.77
D Stack	Ubuntu 10.04 (64bit)	Apache2.2.17(PHP:5.3.5)	PostgreSQL 8.4

제조사	모델명	CPU	MEM	Disk	NIC
IBM	X3550M2	Intel Xeon(R)CPU 2.40GHz * 4	8GB	320GB	Gigabit 1Port

• 테스트 케이스 실행

분석/설계 단계에서 설계된 테스트 케이스를 실행하며, 테스트를 수행하면서 발생하는 항목들의 로그를 기록한다. 테스트 케이스 명세서에 기술된 테스트 기준 및 예상결과를 실제 테스트 결과와 비교하고, 일치하지 않거나 문제가 발생한 경우 해당 사항을 기록한다.

【표 Ⅲ-14. 사용자 시나리오 테스트 케이스 수행 예시】

테스트 케이스	입력데이터	예상결과	실제결과
• 문서목록보기 Page 목록에서 다른 사용자가 등록한 문서를 클릭	Add_001	Add_001이 등록한 문서로 이동	PASS
• 문서내용보기 Page에서 수정버튼을 클릭		정보입력 Page로 이동	PASS
• 등록한 문서의 내용을 변경한 후 등록버튼 클릭	Test_023.txt	내용보기 Page로 이동	FAIL
• 변경한 내용이 적용되었는지 확인		Test_023 적용	N/A

【표 Ⅲ-15. 성능 테스트 케이스 수행 예시】

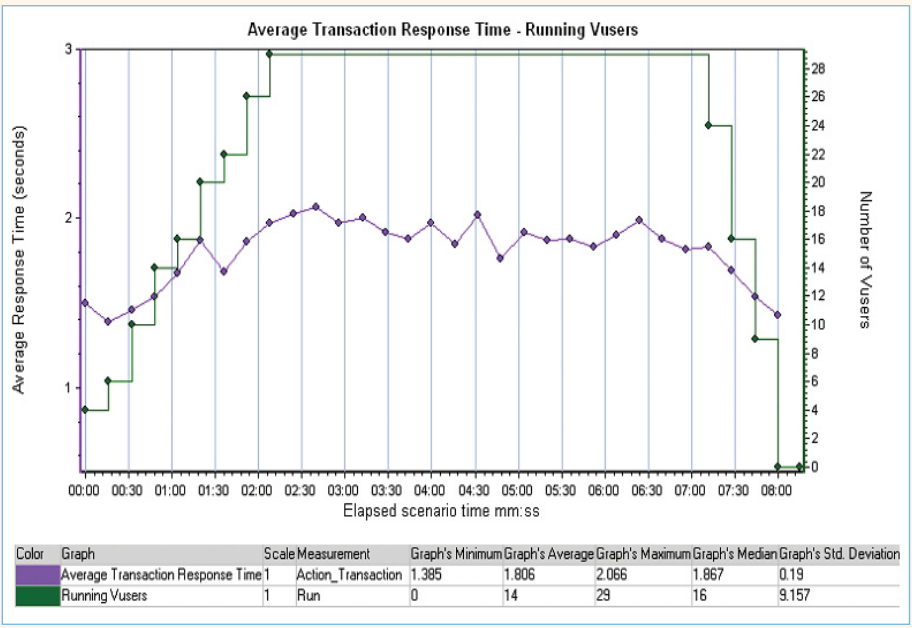
테스트 목표	• 상황을 고려한 Workload Model이 적용된 테스트로써 38.23 TPS를 목표로 함			
목표치	사용자수	10000	호출간격	44.4
	TPS	〉 38.23	응답시간	Read〈1Sec
측정항목	• 거래 성공 비율, TPS, 응답시간, 시스템 사용률			
시나리오	• Ramp-UP Model : (100 Users/3초) * 51초(51초 후에 1000 Users 유입, 측정 시작 • 1000 User 유입 완료 후 2분 동안 부하 발생, 성능 측정 • Ramp-Down Model : (300 Users/초) * 30초(30초 후에 시험 종료) • Think Time Model : 호출 간격(1.4초) – Real 응답 시간			
사전 확인 사항	• 테스트 시작 시간	10:00(AM)	• 테스트 종료 시간	14:00(PM)
	• 스크립트 수행(N번)	50	• 부하 발생기 상태 확인	
	• 데이터베이스 상태 확인		• 시나리오 확인	Load_1, Load_2
	• Configuration 확인		• 테스트 데이터 Import	Moodle_01

- **테스트 결과 작성**

테스트 케이스 명세서에 해당 테스트 케이스 실행 결과를 작성한다. 테스트 결과 보고는 최종 보고서가 아닌 단순한 테스트 케이스 실행에 대한 결과 보고이다. 본 테스트 결과를 기반으로 최종 결과 보고서를 작성한다.

【표 III-16. 사용자 시나리오 테스트 케이스 수행 결과 예시】

예상결과	실제결과	결함내역
Add_001이 등록한 문서로 이동	PASS	
정보입력 Page로 이동	PASS	
내용보기 Page로 이동	FAIL	• Add_001 문서의 내용보기 Page가 아닌 Add_003 문서의 내용보기 Page로 이동함
Test_023 적용	N/A	• 위 테스트 케이스 오류로 인해 Test_023이 적용되었는지 확인 불가



【그림 III-8. 성능 테스트 케이스 수행 결과 예시】

테스트 종료

• 테스트 결과 보고서 작성

공개SW 역량프라자 테스트 결과서는 시스템의 품질에 대한 객관적 평가 결과를 제공하기 위해 작성되며, 내용은 다음과 같다.

공개SW Stack 통합 테스트 결과 보고서 (예시)

1. Stack 통합 테스트 개요
공개SW Stack 통합 테스트의 배경 기술
가. 목적
공개SW Stack 통합 테스트의 목적 기술

2. 제품 소개
가. 제품개요
제품군에 대한 배경 기술
나. 주요SW
제품군에 속한 다양한 공개SW 기술

3. 테스트 대상 소개
가. 선정배경
제품군에 속한 다양한 공개소프트웨어 중 해당 제품이 선정된 이유 및 근거
나. 대상소개
선정된 제품의 주요 기능 및 지원환경 소개

4. Stack 통합 테스트
가. 테스트 환경
정의된 테스트 환경에 대한 H/W 및 S/W 구성을 기술
나. 주요 테스트 방법
적용된 테스트 기법 기술
다. 테스트 시나리오 현황
정의된 테스트 시나리오와 테스트 케이스 명시
라. 테스트 수행 결과
테스트 결과 및 결함 내역 보고

5. 종합
시스템 분석 및 평가

【그림 III-9. 공개SW 테스트 결과 보고서 예시】

• 결과 정보 제공

공개SW 역량프라자에서 운영하는 공개SW 포털(http://www.oss.kr) 웹사이트에 접속 후 '정보마당〉공개SW 테스트'에서 최신 테스트결과 보고서 및 이전에 등록 된 테스트 결과를 열람할 수 있다.

【그림 III-10. 공개SW 포털 테스트 결과 보고서】

[별첨1] 공개SW 테스트 도구

소프트웨어 테스트 자동화를 위한 공개SW 테스트 도구들이 많이 소개되어 있으며, 이러한 도구들은 쉽고 효율적인 소프트웨어 테스트 수행을 지원한다. 하지만 테스트 도구를 사용하는 것은 장점뿐 아니라 단점 역시 포함하고 있으므로 테스트 도구에 대한 이해를 통한 적절한 도구를 선택하는 것이 필요하다.

테스트 도구의 장점

- 테스트 데이터의 재입력 같은 반복 작업의 자동화
- 요구사항의 일관성과 반복의 가능성
- 정적인 측정값 등 객관적인 평가 기준 제공
- 성능에 대한 통계와 그래프 등 테스트 정보에 대한 쉬운 접근

테스트 도구의 단점

- 도입 후 프로세스 적용에 대한 시간, 비용, 노력에 대한 추가 투자
- 도구의 사용에 필요한 노력과 시간에 대한 추가 투자
- 비공개 상용SW의 경우 고가이며, 유지 관리 비용이 높음

테스트 활동에 따른 도구 분류

【표 별첨1-1. 테스트 도구 분류】

테스트 활동	테스트 도구	내 용
테스트 계획	요구사항 관리	고객 요구사항 정의 및 변경사항 관리
테스트 분석/설계	테스트 케이스 생성	테스트 기법에 따른 테스트 데이터 및 케이스 작성
	커버리지 분석	대상시스템에 대한 테스트 완료 범위 척도
테스트 수행	테스트 자동화	기능 테스트 등 테스트 도구를 활용하여 자동화를 통한
	정적분석	테스트의 효율성을 높일 수 있음
	동적분석	코딩표준, 런타임 오류 등을 검증
	성능 테스트	대상시스템 시뮬레이션을 통한 오류 검출
	모니터링	가상사용자를 인위적으로 생성하여 시스템 처리능력 측정
테스트 통제	형상관리	테스트 수행에 필요한 다양한 도구 및 데이터 관리
	테스트 관리	전반적인 테스트 계획 및 활동에 대한 관리
	결함 추적/관리	테스트에서 발생한 결함 관리 및 협업 지원

주요 공개SW 테스트 도구의 소개

테스트 도구는 프로세스화 되었을 때 큰 효과를 볼 수 있기 때문에, 테스트 도구에 대한 도입시 조직적이고 장기적인 측면에서 고려해야 하며, 활용 가능한 주요 공개SW 테스트 도구는 아래와 같다.

• 정적분석 도구

소스코드에 대한 코딩 표준/스타일, 복잡도 및 잔존 결함을 발견하기 위하여 사용하는 도구

【표 별첨1-2. 정적분석 도구】

분 류	제 품 명	세 부 정 보	
결함 예방/발견	pmd(cpd)	개요	자바 소스코드에 대한 잠재적인 문제에 대한 분석
		지원 환경	Linux, Windows
		개발도구 지원	Eclipse, NetBeans 등
		홈페이지	http://pmd.sourceforge.net/

분 류	제 품 명	세 부 정 보	
결함 예방/발견	pmd(cpd)	개요	자바 소스코드에 대한 잠재적인 문제에 대한 분석
		지원 환경	Linux, Windows
		개발도구 지원	Eclipse, NetBeans 등
		홈페이지	http://pmd.sourceforge.net/
	findbugs	개요	자바 소스코드에 대한 잠재적인 문제에 대한 분석
		지원 환경	Cross-Platform
		개발도구 지원	Eclipse, NetBeans 등
		홈페이지	http://findbugs.sourceforge.net/
	cppcheck	개요	C/C++ 소스코드에 대한 잠재적인 문제에 대한 분석
		지원 환경	Windows
		개발도구 지원	Eclipse, gedit 등
		홈페이지	http://cppcheck.sourceforge.net/
	sonar	개요	지속적인 소스 품질 검사를 수행하기 위한 통합 플랫폼으로 C/C++, Java 등 다양한 언어 지원 및 PMD, CheckStyle 등 플러그인을 통하여 확장 가능
		지원 환경	Cross-Platform
		개발도구 지원	eclipse
		홈페이지	http://www.sonarsource.org/
코딩 표준	checkstyle	개요	자바 프로그램에 대한 코딩 표준 준수 검사 도구로, 다양한 개발 도구에 통합하여 활용 가능
		지원 환경	Cross-Platform
		개발도구 지원	Ant, Eclipse, NetBeans 등
		홈페이지	http://checkstyle.sourceforge.net/
	N'SIQ CppStyle	개요	C/C++ 프로그램 언어에 대한 코딩 표준 준수 검사 도구
		지원 환경	Cross-Platform
		개발도구 지원	Visual Studio, eclipse
		홈페이지	http://dev.naver.com/projects/nsiqcppstyle/
	StyleCop	개요	C# 프로그램 언어에 대한 코딩 표준 준수 검사 도구
		지원 환경	Windows
		개발도구 지원	Visual Studio
		홈페이지	http://stylecop.codeplex.com/
	cpplint	개요	구글에서 사용하고 있는 C++ 스타일 가이드 준수 검사 도구로 CLI(파이썬) 형태로 지원됨
		지원 환경	Cross-Platform
		개발도구 지원	–
		홈페이지	http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml#cpplint

분 류	제 품 명	세 부 정 보	
코드 복잡도	ccm	개요	소스코드 복잡도 분석 도구이며, Linux, Mac환경에서는 CLI(Command Line Interface) 형태로 지원됨
		지원 환경	Cross-Platform
		개발도구 지원	Visual Studio
		홈페이지	http://www.blunck.info/ccm.html
	eclipsemetrics	개요	소스코드 복잡도 분석 소스 코드 통계 정보 제공 도구
		지원 환경	Cross-Platform
		개발도구 지원	Eclipse
		홈페이지	http://www.stateofflow.com/projects/16/eclipsemetrics
	sourcemonitor	개요	윈도우 기반 소스코드 복잡도 분석 도구
		지원 환경	Windows
		개발도구 지원	–
		홈페이지	http://www.campwoodsw.com/sourcemonitor.html
	cobertura	개요	자바언어에 대한 소스코드 복잡도 분석 및 커버리지 측정
		지원 환경	Cross-Platform
		개발도구 지원	Ant, Maven
		홈페이지	http://cobertura.sourceforge.net/
	javancss	개요	자바언어에 대한 소스코드 복잡도 분석 도구, CLI 형태로 지원됨
		지원 환경	Cross-Platform
		개발도구 지원	Ant, Jacob
		홈페이지	http://www.kclee.de/clemens/java/javancss/

- **동적분석 도구**

프로그램을 실행하여, 코드 내에 존재하는 메모리 누수, 쓰레드 결함 등을 분석하기 위하여 사용하는 도구

【표 별첨1-3. 동적분석 도구】

제품명	세 부 정 보	
Avalanche	개요	Valgrind 프레임워크 기반으로 구현되었으며, 프로그램에 대한 결함 및 취약점 동적 분석 도구
	지원 환경	Linux, Android
	개발도구 지원	–
	홈페이지	http://code.google.com/p/avalanche/
Valgrind	개요	프로그램 내에 존재하는 메모리 및 쓰레드의 결함을 발견하는 동적 분석 도구
	지원 환경	Cross-Platform
	개발도구 지원	Eclipse, NetBeans 등
	홈페이지	http://valgrind.org

• 테스트 자동화 프레임워크 도구

단위테스트, 통합테스트 등 테스트 단계별 자동화 도구를 활용한 기능 테스트 도구

【표 별첨1-4. 자동화 프레임워크 도구】

제품명	세 부 정 보	
xUnit	개요	java(Junit), C++(Cppunit), .Net(Nunit) 등 다양한 언어를 지원하는 단위테스트 프레임워크
	지원 환경	각각의 도구별 지원 환경 상이
	개발도구 지원	eclipse 등
	홈페이지	http://www.junit.org/ http://sourceforge.net/apps/mediawiki/cppunit/index.php?title=Main_Page http://www.nunit.org/
STAF	개요	서비스 호출, 컴포넌트 재사용 등 다양한 환경을 지원하는 테스트 프레임워크
	지원 환경	Cross-Platform
	개발도구 지원	eclipse
	홈페이지	http://staf.sourceforge.net/
FitNesse	개요	웹기반 테스트케이스 설계/실행/결과확인 등을 지원하는 테스트 프레임워크
	지원 환경	Cross-Platform
	개발도구 지원	eclipse
	홈페이지	http://fitnesse.org/
NTAF	개요	NHN 테스트 자동화 프레임워크이며, STAF와 FitNesse를 통합
	지원 환경	Cross-Platform
	개발도구 지원	eclipse, Maven 등
	홈페이지	http://dev.naver.com/projects/ntaf/
Selenium	개요	다양한 브라우저 지원 및 개발언어를 지원하는 웹 애플리케이션 테스트 프레임워크
	지원 환경	Cross-Platform
	개발도구 지원	-
	홈페이지	http://seleniumhq.org/
watir	개요	Ruby 기반 웹 애플리케이션 테스트 프레임워크
	지원 환경	Cross-Platform
	개발도구 지원	-
	홈페이지	http://watir.com/

• 성능 테스트 도구

가상사용자를 인위적으로 생성하여 시스템의 처리량, 응답시간 등을 테스트하기 위한 도구

【표 별첨1-5. 성능 테스트 도구】

제품명	세 부 정 보	
JMeter	개요	HTTP, FTP, LDAP 등 다양한 프로토콜을 지원하는 부하 테스트 도구
	지원 환경	Cross-Platform
	개발도구 지원	eclipse, Ant
	홈페이지	http://jmeter.apache.org/
AB	개요	아파치 웹서버 부하 테스트 도구이며, CLI기반으로 동작
	지원 환경	Cross-Platform
	개발도구 지원	-
	홈페이지	http://httpd.apache.org/docs/2.2/programs/ab.html
OpenSTA	개요	HTTP, HTTPS 프로토콜에 대한 부한 테스트 도구
	지원 환경	Windows
	개발도구 지원	-
	홈페이지	http://opensta.org/
LoadUI	개요	HTTP, JDBC 등 다양한 프로토콜을 지원하며, 서버 모니터링, Drag&Drop 등 사용자 편리성이 강화된 부하 테스트 도구
	지원 환경	Cross-Platform
	개발도구 지원	-
	홈페이지	http://www.loadui.org/

• 시스템 모니터링 도구

서버 자원(프로세스 상태, 네트워크 등)에 대한 모니터링 도구

【표 별첨1-6. 시스템 모니터링 도구】

제품명	세 부 정 보	
Nagios	개요	웹기반 서버, 서비스, 애플리케이션 등에 대한 모니터링 도구
	지원 환경	Cross-Platform
	개발도구 지원	-
	홈페이지	http://www.nagios.org/
Zenoss	개요	웹기반 서버, 서비스, 애플리케이션 등에 대한 모니터링 도구
	지원 환경	Cross-Platform
	개발도구 지원	-
	홈페이지	http://community.zenoss.org/index.jspa
Zabbix	개요	웹기반 서버, 서비스, 애플리케이션 등에 대한 모니터링 도구
	지원 환경	Cross-Platform
	개발도구 지원	-
	홈페이지	http://www.zabbix.com
Hyperic HQ	개요	웹기반 서버, 서비스, 애플리케이션 등에 대한 모니터링 도구
	지원 환경	Cross-Platform
	개발도구 지원	-
	홈페이지	http://sourceforge.net/projects/hyperic-hq/

[별첨 2] 테스트 결과 보고서 예시(XE)

I. CMS(Content Management System) 소개

1. CMS개요

CMS는 텍스트나 이미지 등 다양한 미디어 콘텐츠의 생성, 수집, 관리, 배포 등의 콘텐츠 관리의 전반적인 활동을 지원하는 애플리케이션으로, 초기에는 기업 내의 서로 다른 부서간의 콘텐츠를 공유하기 위한 목적으로 ECMS(Enterprise Content Management System)에 대한 비중이 높았지만, 최근에는 웹 콘텐츠 및 웹 사이트 관리에 대한 관심 증대로 인해 WCMS(Web Content Management System)에 대한 많은 솔루션이 개발 및 배포되고 있다.

주요기능

- 콘텐츠 생성
  - 콘텐츠를 제작 및 생성할 수 있는 다양한 템플릿 지원 및 작성

- 콘텐츠 관리
  - 사용자 권한 관리, 콘텐츠 버전 관리, 콘텐츠 분류 및 검색 기능

- 콘텐츠 배포
  - 사용자의 선호도 및 권한에 맞는 개인화된 콘텐츠를 제공

2. CMS 분야 주요 공개SW

[표 I-1. 주요 공개SW]

제품명	지원 환경		홈페이지	비 고
XE	OS	Windows, Linux	http://www.xpressengine.com/	- 추가지원DB Firebird, SQLite, postgresql,ms-sql
	WEB	Apache		
	WAS	-		
	DB	MySQL, Cubrid		
	Language	PHP		
KimsQRb	OS	Windows, Linux	http://www.kimsq.com	
	WEB	Apache, IIS		
	WAS	-		
	DB	MySQL		
	Language	PHP		
magnolia	OS	Cross-platform	http://www.magnolia-cms.com/	
	WEB	-		
	WAS	Tomcat		
	DB	-		
	Language	Java		
Zope	OS	Windows, Linux	http://www.zope.org	- 자체DB 사용 (ZODB)
	WEB	-		
	WAS	Zope		
	DB	-		
	Language	Python		
opencms	OS	Linux	http://www.opencms.org	
	WEB	-		
	WAS	Tomcat		
	DB	MySQL		
	Language	Java 5		
drupal	OS	Windows, Linux	http://drupal.org	
	WEB	Apache, IIS		
	WAS	-		
	DB	MySQL, SQLite, PostgreSQL		
	Language	PHP		
mambo	OS	Cross-platform	http://mambo-foundation.org/	
	WEB	Apache		
	WAS	-		
	DB	MySQL		
	Language	PHP		

II. 테스트 대상 소개

1. XE 대상 소개

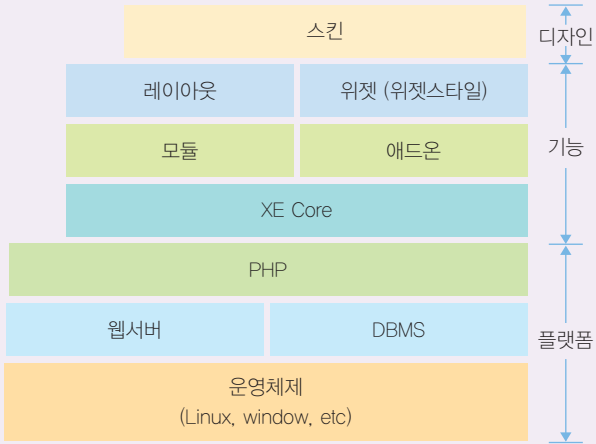
Xpress Engine(구 제로보드 XE)은 고영수님이 여러 자원 봉사자들과 함께 개발한 LGPL 기반 오픈 프로젝트로, 제로보드 4나 zb5와는 별개로 완전히 새로 개발한 웹 프레임워크이다. 제로보드 4와는 달리 BBS, 블로그, 쇼핑몰, 위키 등 웹 사이트에 필요한 모든 것을 모듈로 구현해, 종합적인 웹 빌더로 사용할 수 있는 프레임워크를 목표로 개발이 진행 중이다. 이전 명칭은 '제로보드 XE' 였으나, 정식으로 CMS 기능을 갖춘 1.1.0 버전 안내를 공지하면서 '보드'의 개념이 맞지 않는다고 판단하여 명칭을 변경하였다.

제로보드의 명칭을 유지한 1.0.6 버전까지는 다양한 사이트와 블로그를 운영할 수 있는 홈 빌더의 형태였으나, Xpress Engine으로 명칭을 변경한 1.1.0 버전부터는 마이크로 블로그(플래닛) 패키지, 가상 사이트를 이용한 분양 시스템 등 더 폭넓고 다양한 기능이 탑재되어 있다. 또한 1.4.4 버전부터는 오픈소스 검색 엔진인 Lucene을 기반으로 한 nLucene을 이용하여 게시물 검색 속도가 개선되었다.

1.4.0 버전을 발표하며 사용권 문서가 GPL v.2에서 LGPL v.2로 변경되었고, 현재 XE Core 2.0 개발이 진행 중에 있다.

(출처 : 위키백과)

XE 환경 및 모듈 구성



[그림 II-1. XE 환경 및 모듈 구성]

지원시스템 환경

[표 II-1. 지원 시스템 환경]

분 류	SW	비 고
OS	Windows, Linux	
WEB Server	Apache	
Database	Cubrid, PostgreSQL, Sqlite3 이상, MySQL 4.1 이상, Firebird, MS-SQL	
PHP	PHP 4.x ~ 5.x XML 라이브러리 필수 GD 라이브러리 필수 ICONV 라이브러리 선택	PHP 5.2.2 버전은 PHP 자체 오류로 사용 불가

III. Stack 통합 테스트

1. 테스트 환경

XE 환경

[표 III-1. XE 환경]

모듈	Core	Board	Resource
Version	1.4.4.4	1.2	1.1

Stack 환경

[표 III-2. Stack 환경]

구성	OS	WEB	DB
A Stack	CentOS 5.4 (64bit)	Apache2.2.3(PHP:5.1.6)	MySQL 5.0.77
B Stack	CentOS 5.4 (64bit)	Apache2.2.3(PHP:5.2.16)	Cubrid 8.3.1

HW 환경

[표 III-3. HW 환경]

제조사	모델명	CPU	MEM	Disk	NIC
IBM	X3550M2	Intel Xeon(R)CPU 2.40GHz * 4	8GB	320GB	Gigabit 1Port

2. 기능 테스트 수행 결과

테스트 수행 관련 세부 시나리오 및 테스트 결과는 별첨 「XE Scenario」 문서를 참고한다.

테스트 시나리오 현황

[표 III-4. 테스트 시나리오 환경]

기 능	테스트 시나리오	테스트 케이스
회원관리	5	23
콘텐츠 관리	16	59
시스템 관리	2	10
자료실	5	23
검색	3	14
합 계	31	129

테스트 결과

[표 III-5. 테스트 결과]

분 류		PASS		FAIL		N/A	
기 능	개 수	A Stack	B Stack	A Stack	B Stack	A Stack	B Stack
회원관리	23	23	23	0	0	0	0
콘텐츠 관리	59	59	59	0	0	0	0
검색	14	14	14	0	0	0	0
시스템관리	10	10	10	0	0	0	0
자료실	23	22	21	1	2	0	0

특이사항

- **자료실(resource) 모듈 설정 문제**

  – Cubrid 데이터베이스를 사용하는 B Stack 환경에서 발견된 문제로써, 자료실 모듈 설치 시 Cubrid 데이터베이스에 테이블을 생성하는 과정에서 잘못된 속성을 설정하여 발생하는 문제로 확인됨

  – 해결책

  아래의 방법은 임시방편이며, 정상적으로 문제를 해결하기 위해서는 자료실 모듈의 패치버전이 릴리즈 되어야 함(2011년 2월 현재 잔존 오류)

  1) Cubrid Manager Client를 구동하여 XE 데이터베이스 서버에 접속
  2) xe_resource_packages 테이블의 update_order 컬럼 속성정보 중 DEFAULT 값을 0으로 설정

참고사항

- Cubrid 8.3.1 버전을 사용 시 주의할 사항은 Cubrid 8.3.1 버전과 함께 배포되는 PHP 모듈(CUBRID-PHP-8.3.1.0173.src.tar.gz)에서 Cubrid를 인식하지 못 하는 문제가 발생하여, 현재(2011년 2월) 배포된 모듈을 사용할 경우 정상적으로 PHP 연동되지 않으며 이와 관련된 문제는 Cubrid 기술지원을 통해 해결해야 함

- 게시판 첨부파일의 용량제한 크기를 변경 할 경우 XE 설정정보를 변경하여도 정상적으로 파일첨부 크기가 늘어나지 않는 경우 php.ini 정보 중 upload_max_filesize 정보를 수정 후 웹서버를 재부팅 하면, 정상적인 첨부파일 제한 크기로 설정되어 있음을 확인할 수 있음

3. 성능테스트 수행결과

성능 테스트의 경우 하드웨어 사양뿐 아니라, OS 및 애플리케이션 환경 구성에 따라 성능 측정 결과가 상이하므로, 실제 운영 시스템 환경에 따라 테스트 결과가 다를 수 있다.

본 성능 테스트는 사용자별 응답시간 및 자원사용률을 측정하였음

테스트 시나리오

[표 III-6. 테스트 시나리오]

시나리오 ID	시 나 리 오
SC_XE_1	게시판 콘텐츠 등록
SC_XE_2	게시판 콘텐츠 내용 검색 (제목+내용)

서버 설정 정보

[표 III-7. 서버 설정 정보]

구 분	항 목	
WEB	PHP	memory_limit = 512M max_input_time = 60 max_execution_time = 30
	HTTP	MaxClients 150 KeepAliveTimeout 5 Timeout 300
DB	MySQL	max-connections 151
	Cubrid	max_clients=200 MAX_NUM_APPL_SERVER=210
Network Bandwidth	PC<->WEB	94Mbits/sec
	WEB<->DB	941 Mbits/sec

측정 항목

[표 III-8. 측정항목]

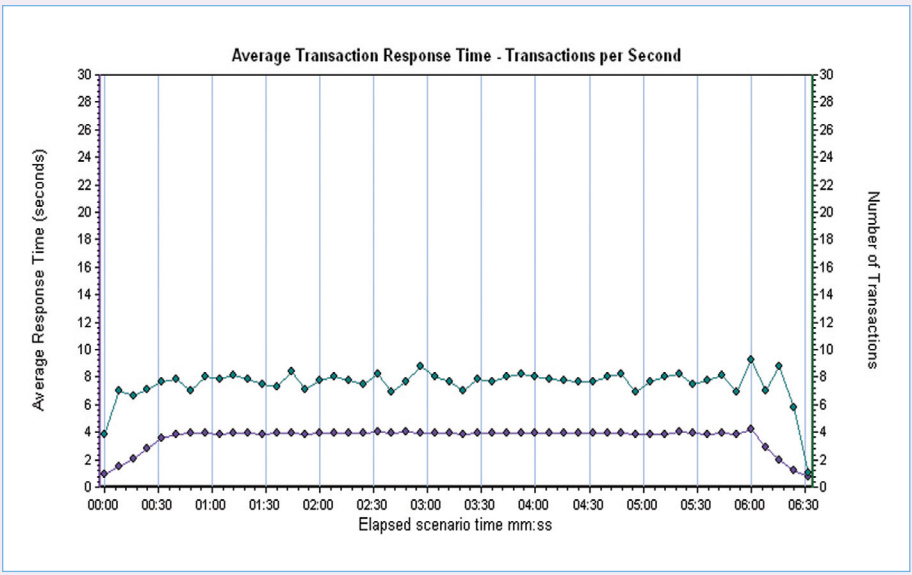
항 목	내 용
TPS	단위시간 당 트랜잭션 처리 수
응답시간	사용자 요청을 처리하기 위한 소요된 총시간
자원 사용률	CPU, Memory 등 서버 자원사용 현황

테스트 결과

- 게시판 콘텐츠 등록 (A Stack)

수행 조건	1. Lamp up : 2User / 1초 2. Running-Time : 5분 3. Lamp down : 30User / 5초
사용자 시나리오	1. 로그인 2. 게시판 목록 이동 3. 글쓰기 및 저장 (측정 대상) 4. 로그아웃

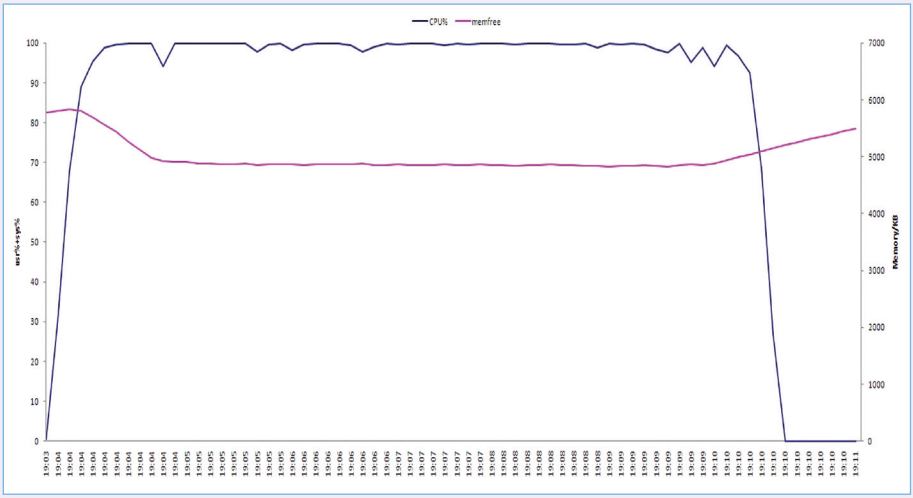
– 측정결과 (100User)



Color	Graph	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximum	Graph's Median	Graph's Std. Deviation
	Average Transaction Response Time	1	write	0.782	3.561	4.159	3.896	0.84
	Transactions per Second	1	write:Pass	1	7.48	9.25	7.75	1.221

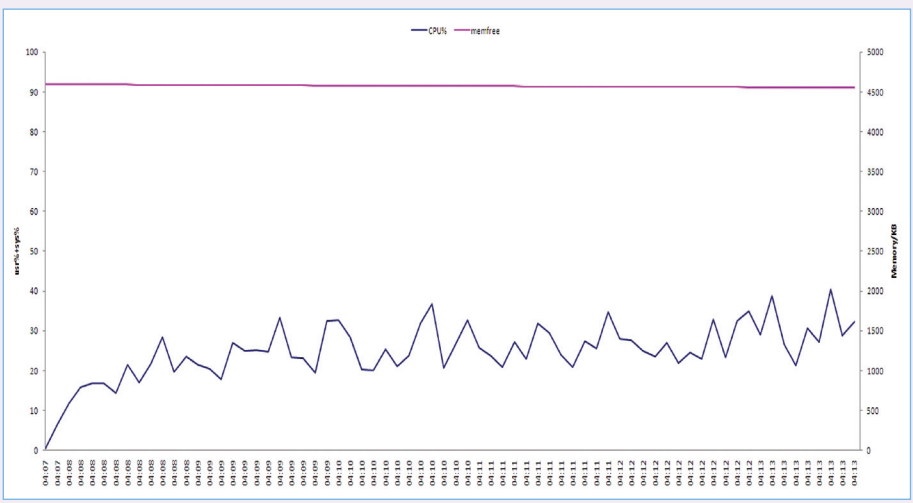
[그림 III-1. 100User 트랜잭션 결과]

응답시간은 평균 4초 이내의 안정적인 응답을 보이고 있음



[그림 III-2. WEB 자원 사용률]

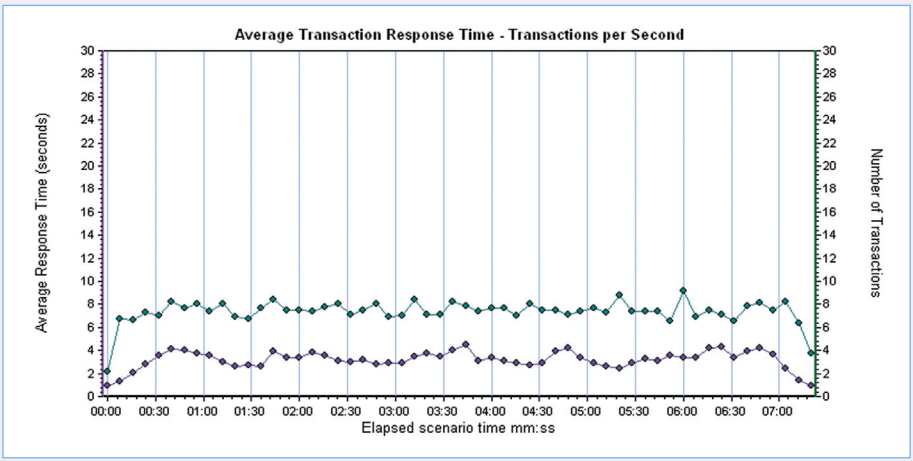
CPU 사용률이 다소 높게 나타나고 있어 해당 자원에 대한 부족 현상이 관찰되고 있고, 메모리 사용량은 일정 수준의 사용량이 보이고 있음



[그림 III-3. DB 자원 사용률]

CPU 사용률이 40% 이내에서 동작하며, 가용 메모리가 일정 수준으로 유지하는 등 서버의 자원사용률이 높지 않은 편임

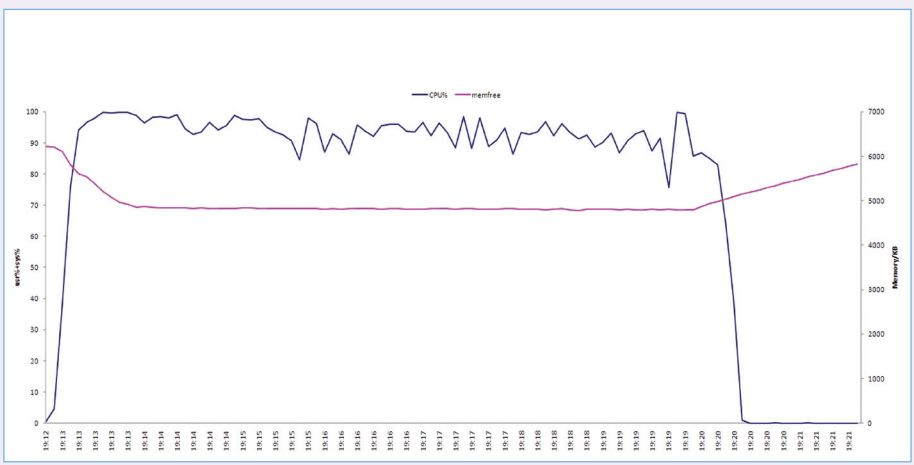
– 측정결과 (150User)



Color	Graph	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximum	Graph's Median	Graph's Std. Deviation
	Average Transaction Response Time	1	write	0.89	3.175	4.507	3.328	0.775
	Transactions per Second	1	write:Pass	2.125	7.328	9.125	7.5	1.025

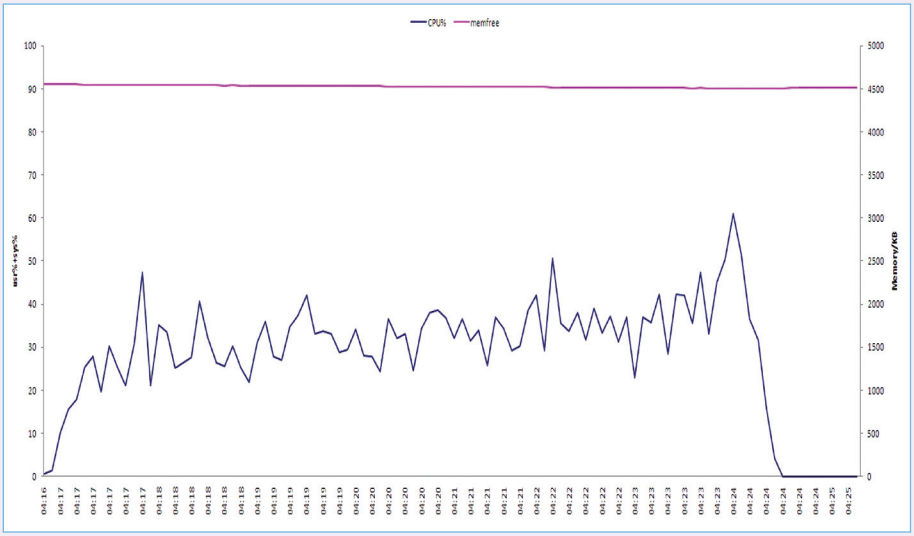
[그림 III–4. 150User 트랜잭션 결과]

응답시간은 평균 4초 이내의 안정적인 응답을 보이고 있음



[그림 III–5. WEB 자원 사용률]

CPU 사용률이 다소 높게 나타나고 있어 해당 자원에 대한 부족 현상이 관찰되고 있고, 메모리 사용량은 일정 수준의 사용량이 보이고 있음



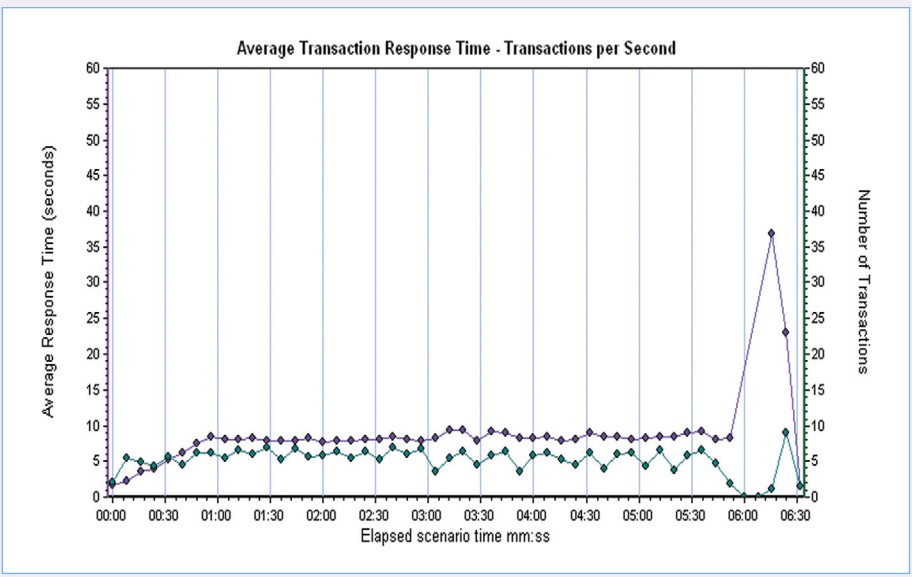
[그림 III-6. DB 자원 사용률]

CPU 사용률이 50% 이내에서 동작하며, 가용 메모리가 일정 수준으로 유지하는 등 서버의 자원사용률이 높지 않은 편임

• 게시판 콘텐츠 등록 (B Stack)

수행 조건	1. Lamp up : 2User / 1초 2. Running-Time : 5분 3. Lamp down : 30User / 5초
사용자 시나리오	1. 로그인 2. 게시판 목록 이동 3. 글쓰기 및 저장 (측정 대상) 4. 로그아웃

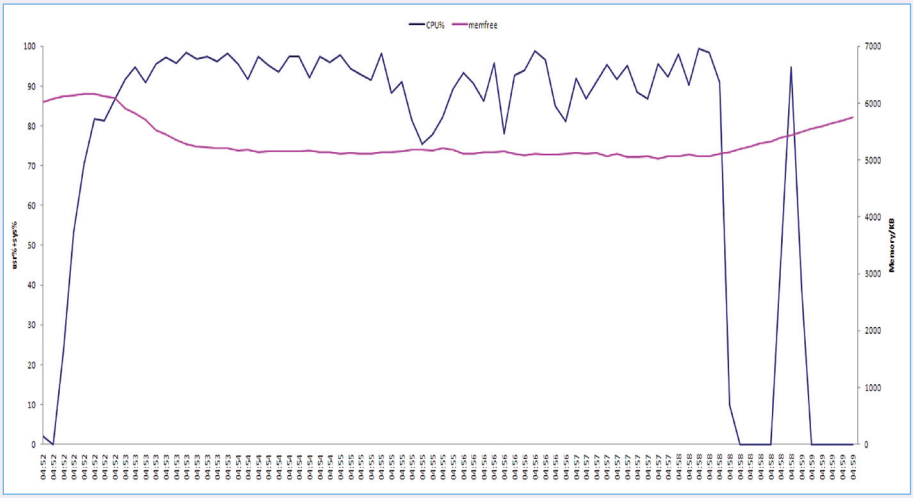
– 측정결과 (100User)



Color	Graph	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximum	Graph's Median	Graph's Std. Deviation
	Average Transaction Response Time	1	write	1.471	8.481	36.785	8.113	5.038
	Transactions per Second	1	write:Pass	0	5.115	9	5.625	1.808

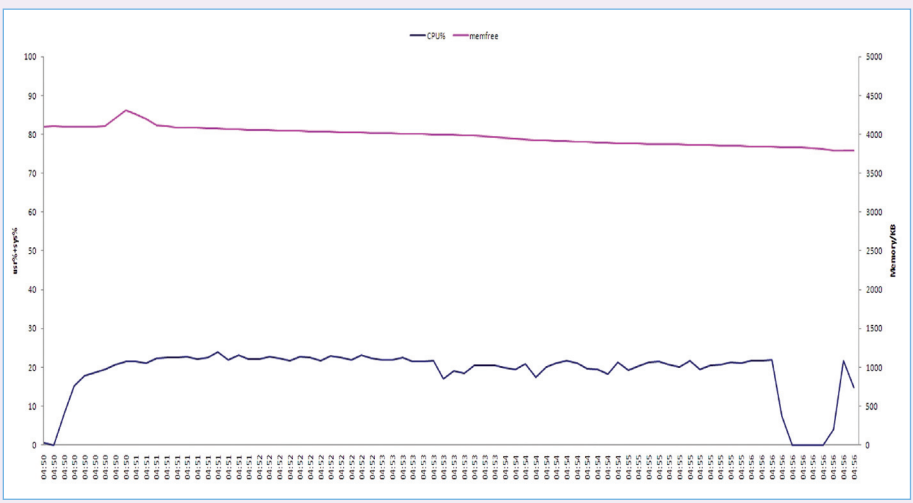
[그림 III-7. 100User 트랜잭션 결과]

응답시간은 10초 이내로 다소 응답시간이 오래 소요되나, 고른 응답을 나타냄



[그림 III-8. WEB 자원 사용률]

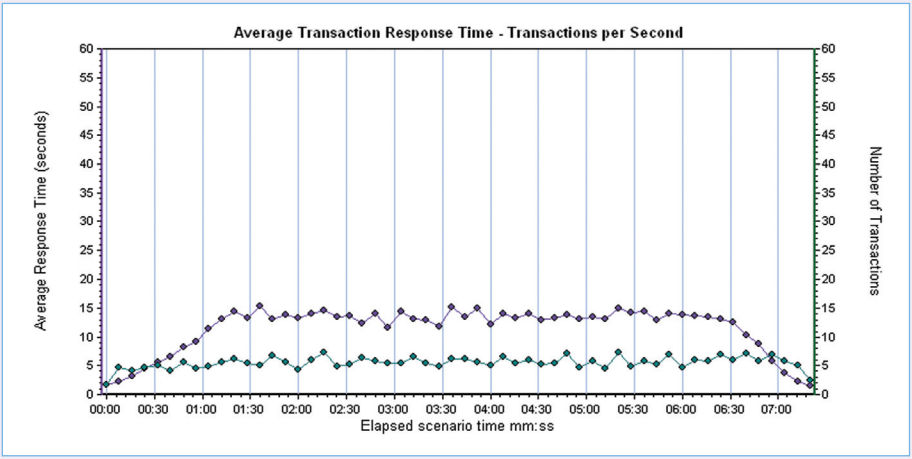
CPU 사용률이 다소 높게 나타나고 있어 해당 자원에 대한 부족 현상이 관찰되고 있고, 메모리 사용량은 일정 수준의 사용량이 보이고 있음



[그림 III-9. DB 자원 사용률]

CPU 사용률이 20% 내외에서 동작하며, 가용 메모리가 일정 수준으로 유지하는 등 서버의 자원사용률이 높지 않은 편임

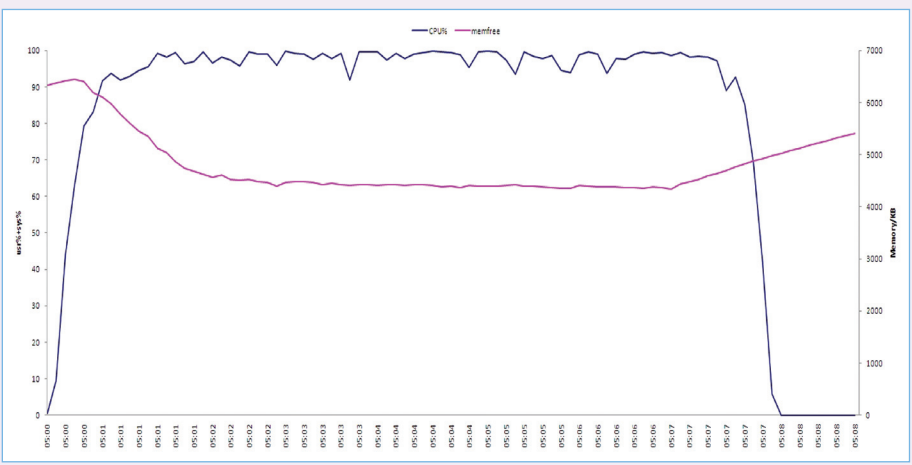
– 측정결과 (150User)



Color	Graph	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximum	Graph's Median	Graph's Std. Deviation
	Average Transaction Response Time	1	write	1.45	11.441	15.256	13.169	3.932
	Transactions per Second	1	write:Pass	1.625	5.494	7.25	5.625	1.046

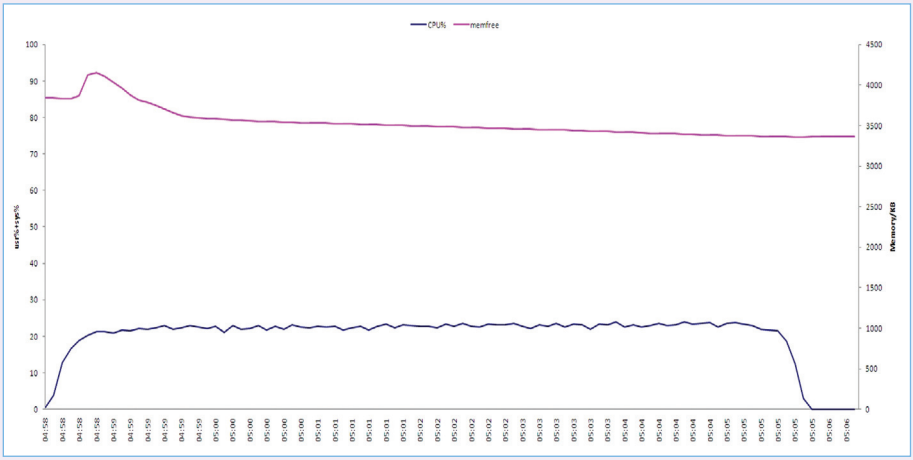
[그림 III–10. 150User 트랜잭션 결과]

응답시간은 12초 이내로 다소 응답시간이 오래 소요되나, 고른 응답을 나타냄



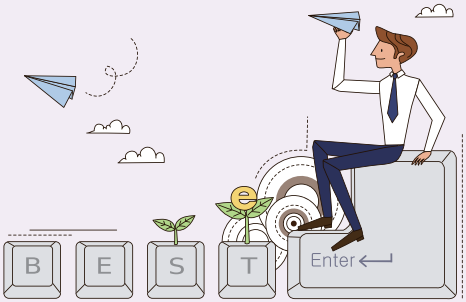
[그림 III–11. WEB 자원 사용률]

CPU 사용률이 다소 높게 나타나고 있어 해당 자원에 대한 부족 현상이 관찰되고 있고, 메모리 사용량은 초기에 사용자 증가에 따라 사용량이 급증하였으나, 이후 일정 수준의 사용량이 보이고 있음



[그림 III-12. DB 자원 사용률]

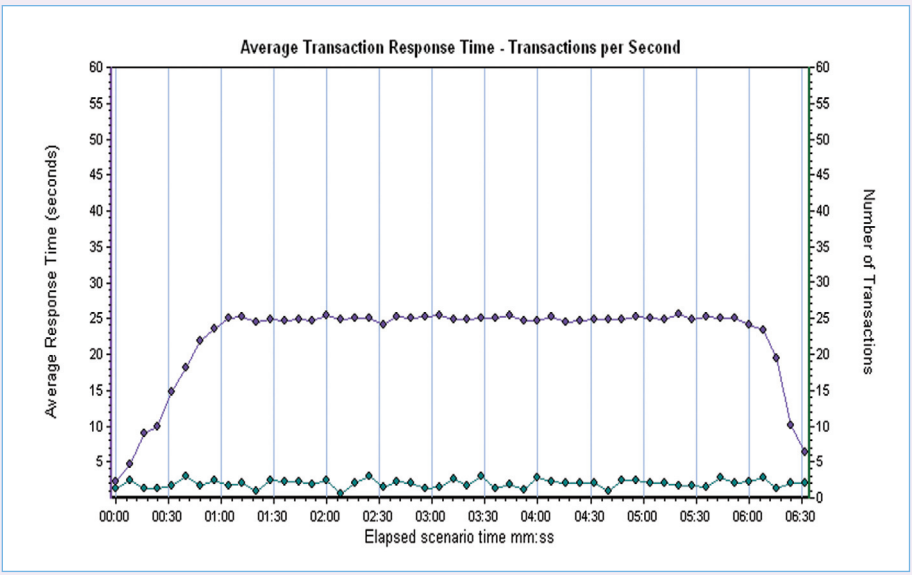
CPU 사용률이 20% 내외에서 동작하며, 사용자 증가에 따른 메모리 사용량 증가 후 일정한 수준을 유지하고 있음



• 게시판 콘텐츠 등록 (A Stack)

수행 조건	1. Lamp up : 2User / 1초 2. Running-Time : 5분 3. Lamp down : 30User / 5초
사용자 시나리오	1. 로그인 2. 게시판 목록 이동 3. 내용 검색 (측정 대상) 4. 로그아웃

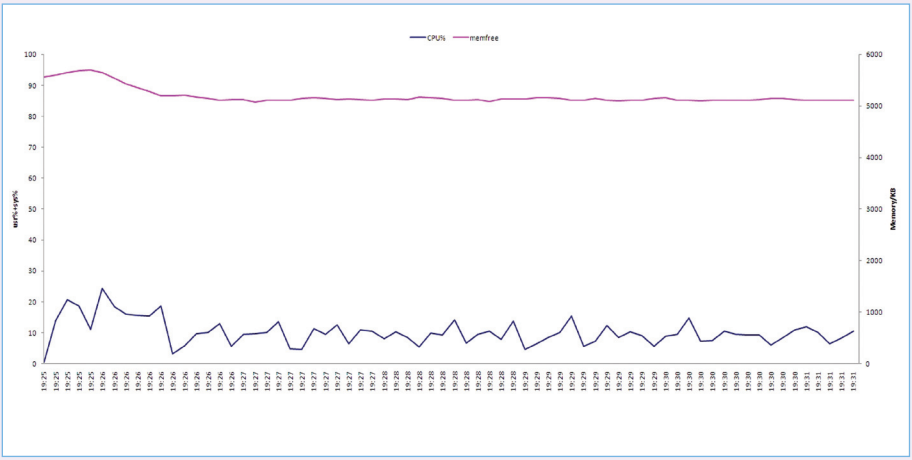
– 측정결과 (100User)



Color	Graph	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximum	Graph's Median	Graph's Std. Deviation
	Average Transaction Response Time	1	search	2.188	22.221	25.552	24.846	6.015
	Transactions per Second	1	search:Pass	0.625	1.968	3	2	0.568

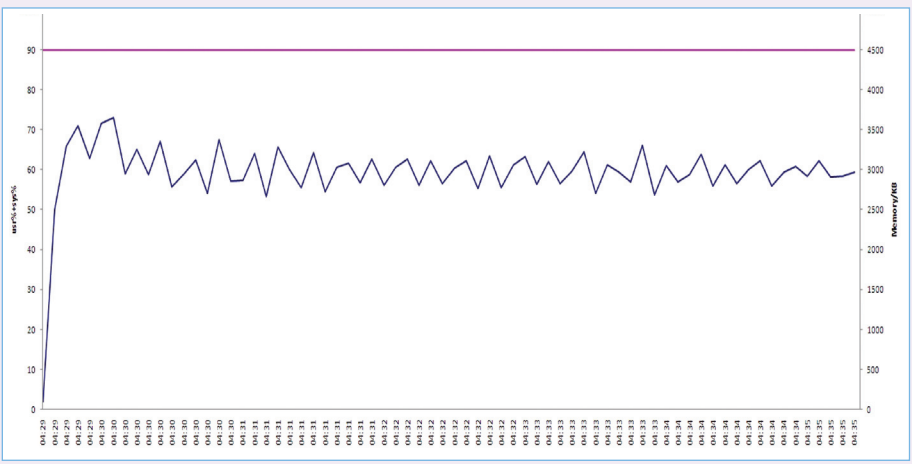
[그림 III-13. 100User 트랜잭션 결과]

평균 응답시간은 22초 내외로 다소 응답시간이 오래 소요되는 현상을 보임



[그림 III-14. WEB 자원 사용률]

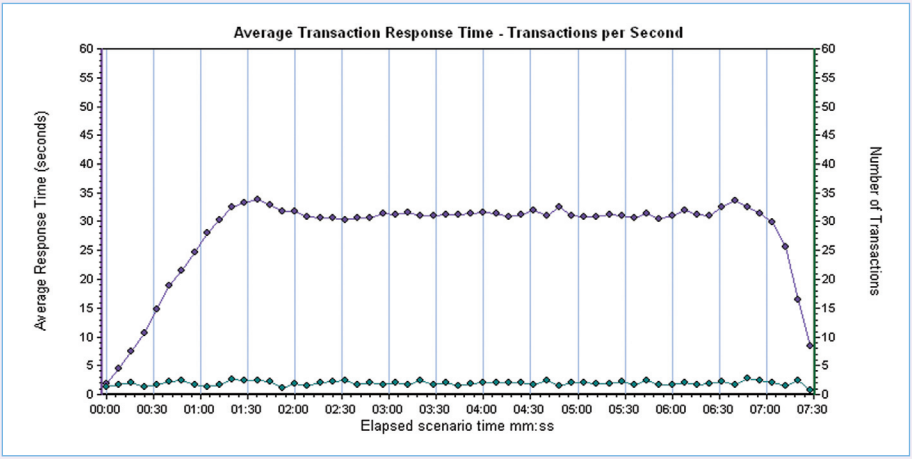
CPU 사용률이 20% 이내에서 동작하며, 가용 메모리가 일정 수준으로 유지하는 등 서버의 자원사용률이 높지 않은 편임



[그림 III-15. DB 자원 사용률]

CPU 사용률이 70% 내외에서 동작하며, 검색 기능에 대한 자원 사용률이 WEB서버보다 DB서버의 자원 사용량이 많은 것으로 파악되며, 가용 메모리는 일정 수준으로 유지됨

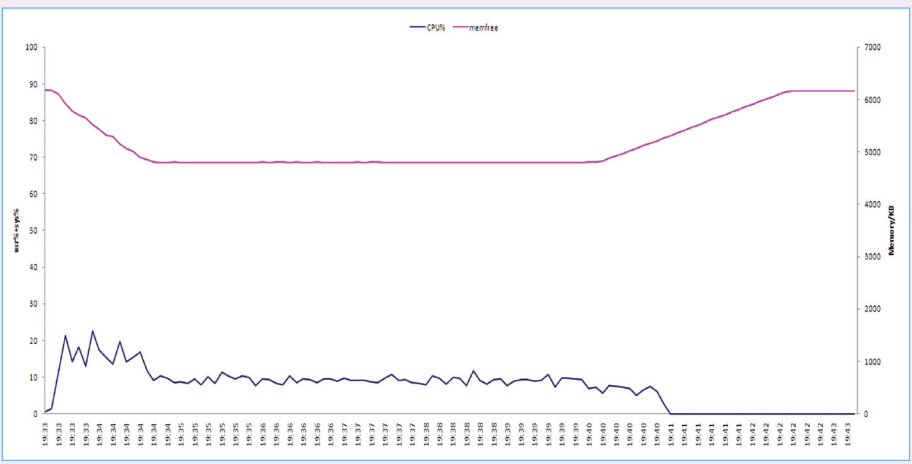
– 측정결과 (150User)



Color	Graph	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximum	Graph's Median	Graph's Std. Deviation
	Average Transaction Response Time	1	search	0.56	1.878	2.349	2.055	0.464
	Transactions per Second	1	search:Pass	1.5	8.868	11	9.375	1.849

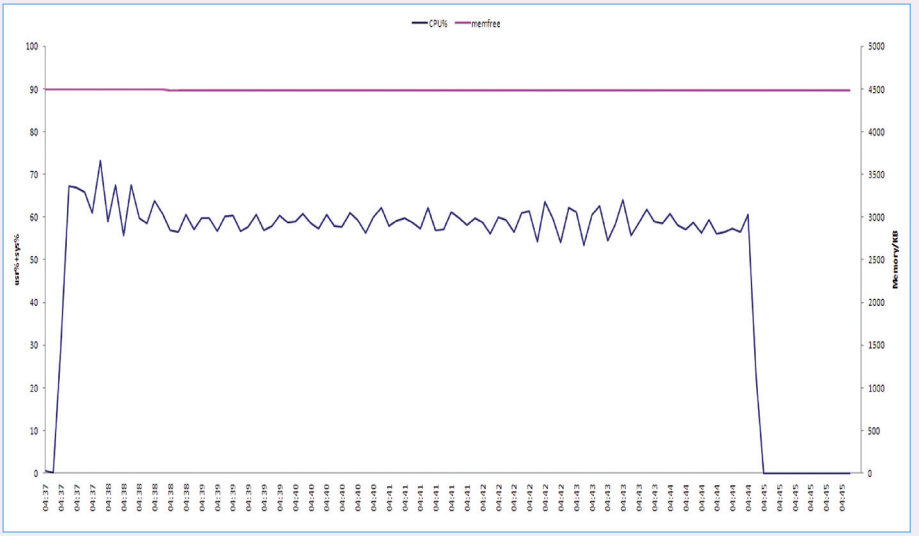
[그림 III–16. 150User 트랜잭션 결과]

사용자 증가에 따라 평균 응답시간은 28초 내외로 다소 응답시간이 오래 소요되는 현상을 보임



[그림 III–17. WEB 자원 사용률]

CPU 사용률이 20% 내외에서 동작하며, 메모리 사용량은 초기에 사용자 증가에 따라 사용량이 급증하였으나, 이후 일정 수준의 사용량이 보이고 있음



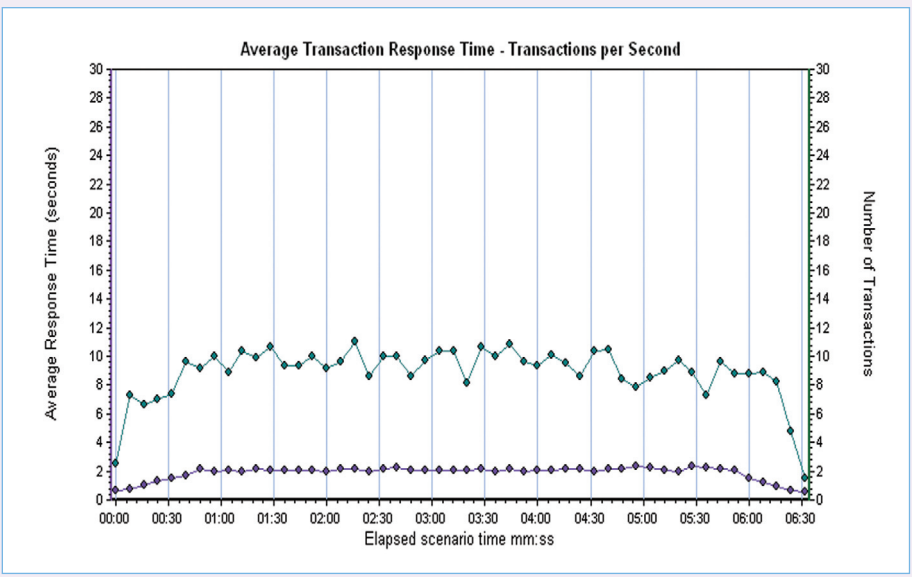
[그림 III-18. DB 자원 사용률]

CPU 사용률이 70% 내외에서 동작하며, 검색 기능에 대한 자원 사용률이 WEB 서버보다 DB서버의 자원 사용량이 많은 것으로 파악되며, 가용 메모리는 일정 수준으로 유지됨

● 게시판 콘텐츠 내용검색 (B Stack)

수행 조건	1. Lamp up : 2User / 1초 2. Running–Time : 5분 3. Lamp down : 30User / 5초
사용자 시나리오	1. 로그인 2. 게시판 목록 이동 3. 내용 검색 (측정 대상) 4. 로그아웃

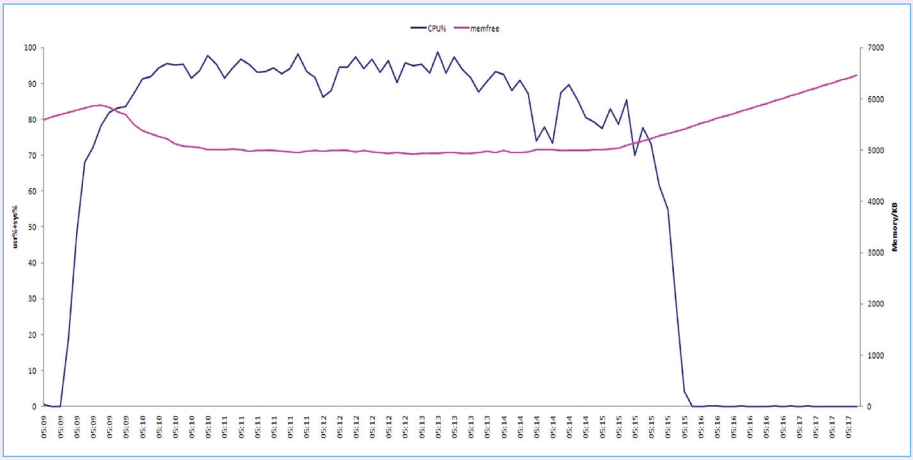
– 측정결과 (100User)



Color	Graph	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximum	Graph's Median	Graph's Std. Deviation
	Average Transaction Response Time	1	search	0.56	1.878	2.349	2.055	0.464
	Transactions per Second	1	search:Pass	1.5	8.868	11	9.375	1.849

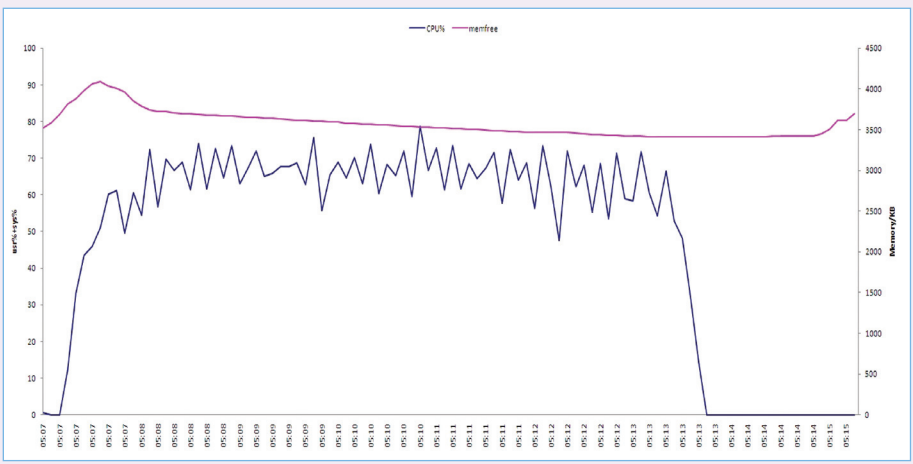
[그림 III–19. 100User 트랜잭션 결과]

응답시간은 평균 2초 내외에 안정적인 응답을 보이고 있음



[그림 III-20. WEB 자원 사용률]

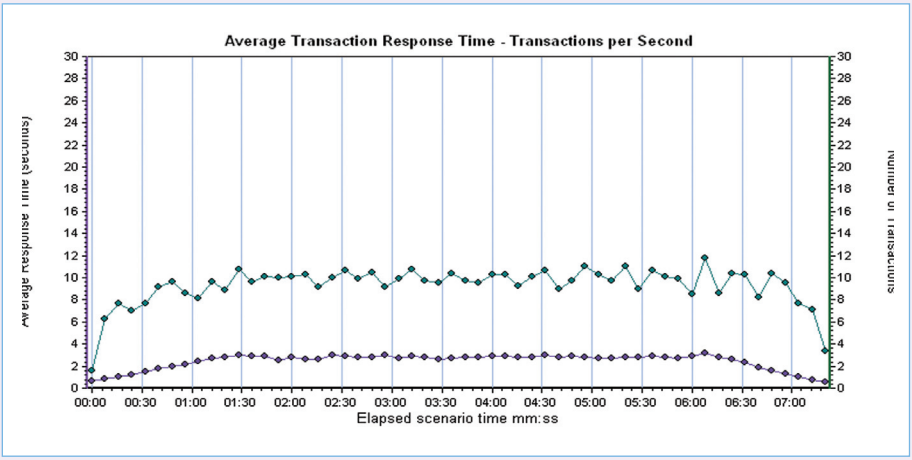
CPU 사용률이 다소 높게 나타나고 있어 해당 자원에 대한 부족 현상이 관찰되고 있으며, 메모리 사용량은 초기에 사용자 증가에 따라 사용량이 증가하였으나, 이후 일정 수준의 사용량이 보이고 있음



[그림 III-21. DB 자원 사용률]

CPU 사용률이 70% 내외에서 동작하며, 검색 기능에 대한 DB 의존도가 높아 자원 사용량이 많은 것으로 파악되고, 가용 메모리는 일정 수준으로 유지됨

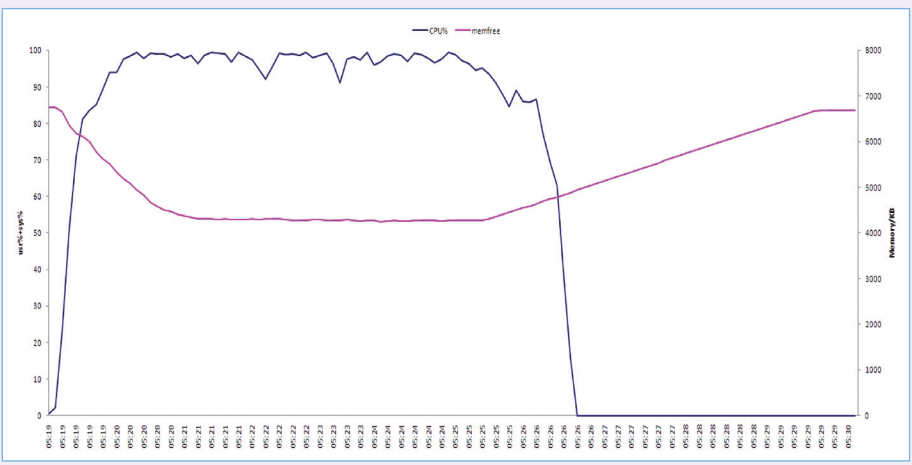
– 측정결과 (150User)



Color	Graph	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximum	Graph's Median	Graph's Std. Deviation
	Average Transaction Response Time	1	search	0.585	2.42	3.215	2.765	0.703
	Transactions per Second	1	search:Pass	1.625	9.292	11.75	9.75	1.708

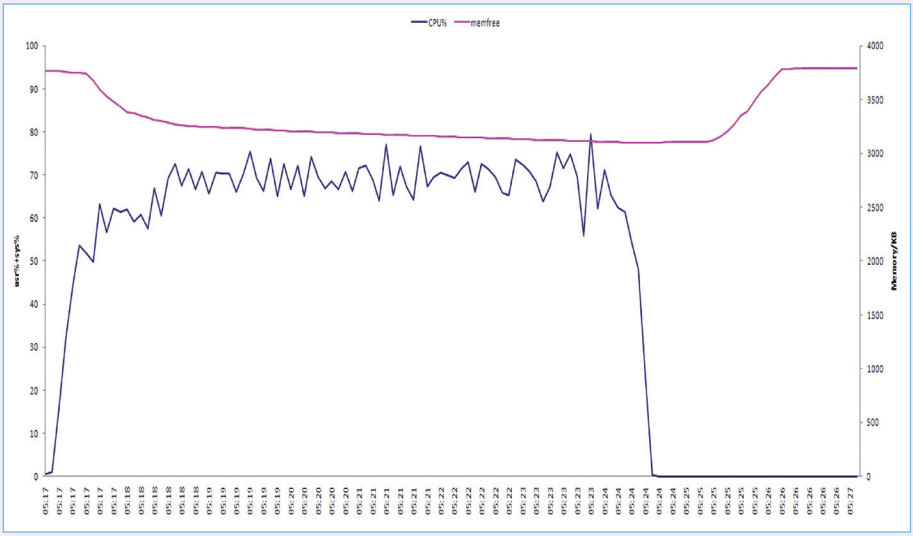
[그림 III-22. 150User 트랜잭션 결과]

응답시간은 평균 2초 내외에 안정적인 응답을 보이고 있음



[그림 III-23. WEB 자원 사용률]

CPU 사용률이 다소 높게 나타나고 있어 해당 자원에 대한 부족 현상이 관찰되고 있으며, 메모리 사용량은 초기에 사용자 증가에 따라 사용량이 급증하였으나, 이후 일정 수준의 사용량이 보이고 있음



[그림 III-24. DB 자원 사용률]

CPU 사용률이 70% 내외에서 동작하며, 검색 기능에 대한 DB 의존도가 높아 자원 사용량이 많은 것으로 파악되고, 메모리 사용량은 다소 높게 나타남

IV. 종합

- 국내 WCMS 분야에서 높은 인지도와 사용률을 보이고 있는 Xpress Engine 통합테스트 결과 공개SW로 구성된 A, B Stack 상에서 각 기능 시나리오 수행 시 치명적 오류 또는 심각한 장애가 발생하지 않았으며, Stack을 구성하는 각 공개SW가 유기적으로 동작함을 확인하였음

- 다만, 일부 환경에서 발생한 경미한 오류와 같은 상황의 방지를 위해 신규 버전 릴리즈 시 보다 다양한 Stack 조합의 테스트가 수반되어야 할 것으로 판단되며, 향후 XE를 도입하고자 하는 공공 및 민간에서 본 통합테스트의 Stack 환경 통합테스트 정보가 유용하게 활용될 수 있을 것으로 판단됨

[참고자료]

[1] Gelperin, D., and B. Hetzel, "The Growth of Software Testing", CACM, Vol. 31, No, 6, 1988, pp. 687–695.

[2] Ilene Burnstein, Taratip Suwannasart and C.R.Carlson, "Developing a Testing Maturity Model : Part I" Crosstalk, August 1996

[3] Tester's Insight 2008년 겨울호 STA(소프트웨어테스트연구소)

[4] 권원일 외, 개발자도 알아야 할 소프트웨어 테스트 실무 제2판, 2008

[5] 권원일 외, 개발자도 알아야 할 소프트웨어 테스트 용어, 2007

[6] 국가표준종합정보센터(http://www.standard.go.kr)

[7] 소프트웨어공학 국제표준화 기술위원회(http://www.jtcl-sc7.org)

[8] 지식경제부 기술 표준원 (http://www.kats.go.kr/)

[9] ISO/IEC 29119 Software Testing (http://softwaretestingstandard.org)

[10] ISO (http://www.iso.org)

[11] TTA 한국정보통신기술협회 (http://www.tta.or.kr/)

[12] NIPA 소프트웨어공학센터 (http://www.software.kr/)

공개SW 테스트 가이드

2012년 9월 1일 인쇄
2012년 9월 1일 발행

발행인 정 경 원
발행처 정보통신산업진흥원
121-795 서울특별시 마포구 상암동 1605
누리꿈스퀘어 연구개발타워 13층 공개SW역량프라자
TEL. 02-2132-1400 FAX. 02-2132-1428

디자인·인쇄 연화디앤피 TEL. 02-2266-8755

ISBN 978-89-6108-203-7 93000

9 788961 082037 93000
ISBN 978-89-6108-203-7