

대분류 / 20
정보통신

중분류 / 01
정보기술

소분류 / 02
정보기술개발

세분류 / 02
응용SW엔지니어링

학습모듈 / 07

07 개발자 테스트

LM2001020207_14v2

응용SW엔지니어링 학습모듈

01. 요구사항 확인



02. 애플리케이션 설계



03. 애플리케이션 구현



04. 화면 구현



05. 데이터 입출력 구현



06. 통합 구현



07. 개발자 테스트



08. 정보시스템 이행



09. 제품소프트웨어 패키징



10. 소프트웨어공학 활용

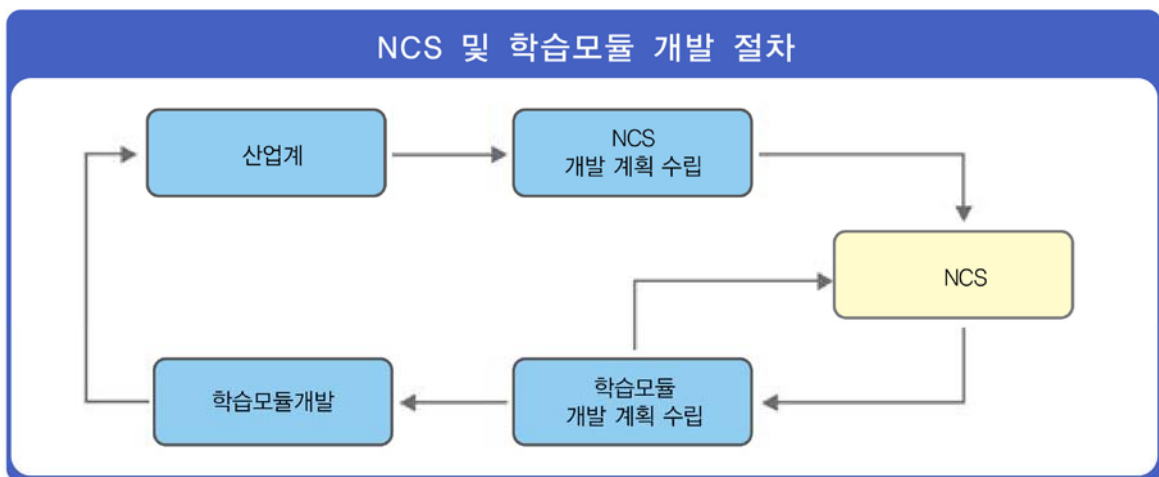


NCS 학습모듈의 이해

※ 본 학습모듈은 「NCS 국가직무능력표준」 사이트(<http://www.ncs.go.kr>) 에서 확인 및 다운로드 할 수 있습니다.

(1) NCS 학습모듈이란?

- 국가직무능력표준(NCS: National Competency Standards)이란 산업현장에서 직무를 수행하기 위해 요구되는 지식·기술·소양 등의 내용을 국가가 산업부문별·수준별로 체계화한 것으로 산업현장의 직무를 성공적으로 수행하기 위해 필요한 능력(지식, 기술, 태도)을 국가적 차원에서 표준화한 것을 의미합니다.
- 국가직무능력표준(이하 NCS)이 현장의 ‘직무 요구서’라고 한다면, NCS 학습모듈은 NCS의 능력단위를 교육훈련에서 학습할 수 있도록 구성한 ‘교수·학습 자료’입니다. NCS 학습모듈은 구체적 직무를 학습할 수 있도록 이론 및 실습과 관련된 내용을 상세하게 제시하고 있습니다.

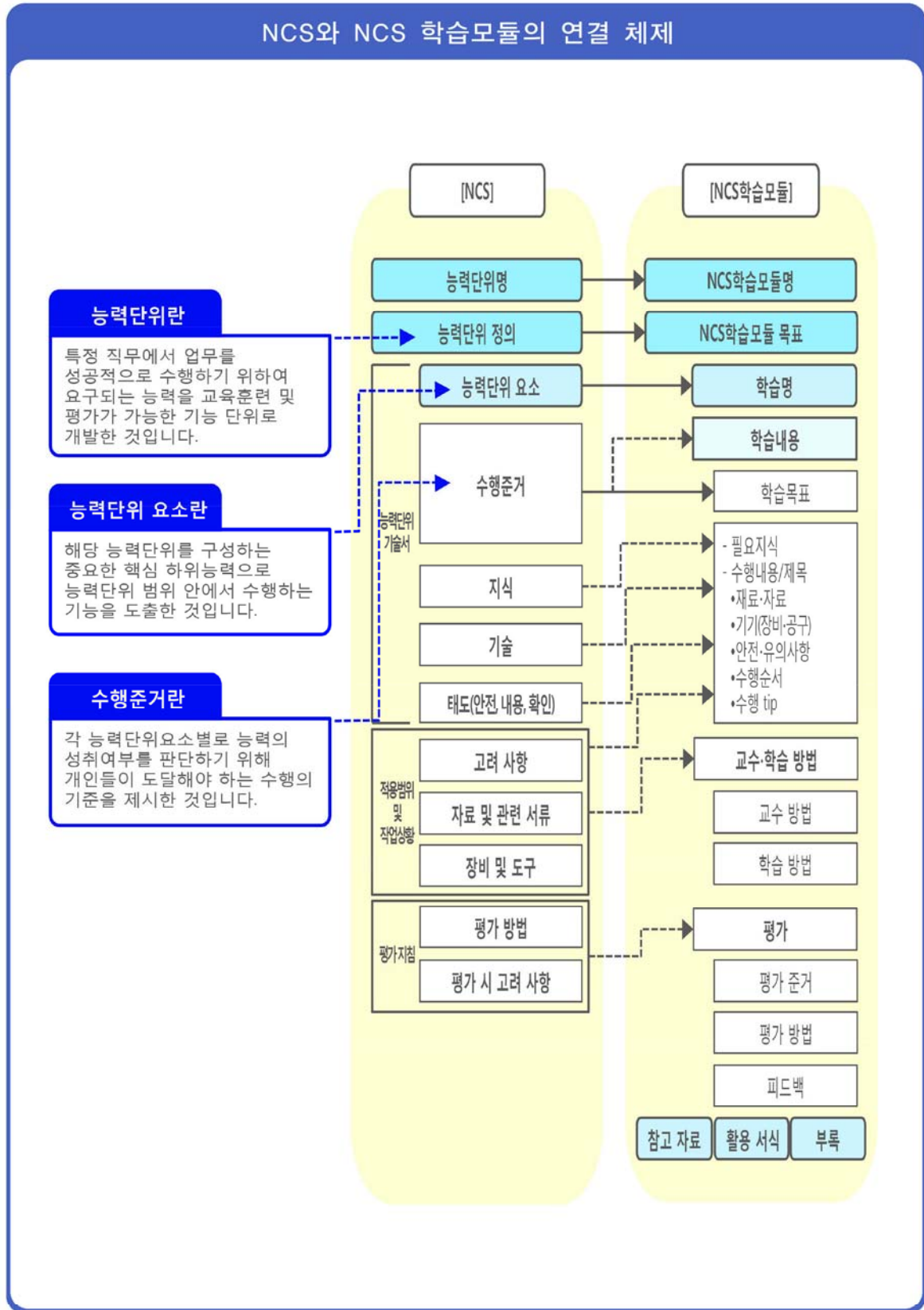


- NCS 학습모듈은 다음과 같은 특징을 가지고 있습니다.

첫째, NCS 학습모듈은 산업계에서 요구하는 직무능력을 교육훈련 현장에 활용할 수 있도록 성취목표와 학습의 방향을 명확히 제시하는 가이드라인의 역할을 합니다.

둘째, NCS 학습모듈은 특성화고, 마이스터고, 전문대학, 4년제 대학교의 교육기관 및 훈련기관, 직장교육기관 등에서 표준교재로 활용할 수 있으며 교육과정 개편 시에도 유용하게 참고할 수 있습니다.

- NCS와 NCS 학습모듈 간의 연결 체제를 살펴보면 아래 그림과 같습니다.



(2) NCS 학습모듈의 체계

- NCS 학습모듈은 1.학습모듈의 위치, 2.학습모듈의 개요, 3.학습모듈의 내용 체계, 4.참고 자료, 5.활용 서식/부록 으로 구성되어 있습니다.

1. NCS 학습모듈의 위치

- NCS 학습모듈의 위치는 NCS 분류 체계에서 해당 학습모듈이 어디에 위치하는지를 한 눈에 볼 수 있도록 그림으로 제시한 것입니다.

예시 : 이·미용 서비스 분야 중 네일미용 세분류

NCS-학습모듈의 위치

| | |
|-----|-----------------|
| 대분류 | 이용·숙박·여행·오락·스포츠 |
| 중분류 | 이·미용 |
| 소분류 | 아·미용 서비스 |

| 세분류 | 능력단위 | 학습모듈명 |
|------|-------------|-----------|
| 헤어미용 | 네일 샵 위생 서비스 | 네일샵 위생서비스 |
| 피부미용 | 네일 화장을 제거 | 네일 화장을 제거 |
| 메이크업 | 네일 기본 관리 | 네일 기본관리 |
| 네일미용 | 네일 랩 | 네일 랩 |
| 이용 | 네일 팁 | 네일 팁 |
| | 젤 네일 | 젤 네일 |
| | 아크릴릭 네일 | 아크릴 네일 |
| | 평면 네일아트 | 평면 네일아트 |
| | 융합 네일아트 | 융합 네일아트 |
| | 네일 샵 운영관리 | 네일샵 운영관리 |

학습모듈은

NCS 능력단위 1개당 1개의 학습모듈 개발을 원칙으로 합니다. 그러나 필요에 따라 고용 단위 및 교과단위를 고려하여 능력단위 몇 개를 묶어서 1개의 학습모듈로 개발할 수 있으며, NCS 능력단위 1개를 여러 개의 학습모듈로 나누어 개발할 수도 있습니다.

2. NCS 학습모듈의 개요

구 성

- NCS 학습모듈 개요는 학습모듈이 포함하고 있는 내용을 개략적으로 설명한 것으로서 **학습모듈의 목표**, **선수 학습**, **학습모듈의 내용 체계**, **핵심 용어**로 구성되어 있습니다.

| | |
|-------------|--|
| 학습모듈의 목표 | 해당 NCS 능력단위의 정의를 토대로 학습목표를 작성한 것입니다. |
| 선수 학습 | 해당 학습모듈에 대한 효과적인 교수·학습을 위하여 사전에 이수해야 하는 학습모듈, 학습 내용, 관련 교과목 등을 기술한 것입니다. |
| 학습모듈의 내용 체계 | 해당 NCS 능력단위요소가 학습모듈에서 구조화된 방식을 제시한 것입니다. |
| 핵심 용어 | 해당 학습모듈의 학습 내용, 수행 내용, 설비·기자재 등 가운데 핵심적인 용어를 제시한 것입니다. |

활 용 안 내

예시 : 네일미용 세분류의 ‘네일 기본관리’ 학습모듈

네일 기본관리 학습모듈의 개요

학습모듈의 목표

고객의 네일 보호와 미적 요구 충족을 위하여 효과적인 네일 관리로 프리에지 형태 만들기, 큐티클 정리하기, 컬러링하기, 보습제 도포하기, 마무리를 할 수 있다.

선수학습

네일숍 위생서비스(JM1201010401_14v2)

학습모듈의 내용체계

| 학습 | 학습내용 | NCS 능력단위요소 | | |
|------------------|--|-------------------|-------------|----|
| | | 코드번호 | 요소명칭 | 수준 |
| 1. 프리에지 형태 만들기 | 1-1. 네일 파일에 대한 이해와 활용 1-2. 프리에지 형태 파일링 | 1201010403_12v2.1 | 프리에지 모양 만들기 | 3 |
| 2. 큐티클 정리하기 | 2-1. 네일 기본관리 매뉴얼 이해 2-2. 큐티클 관리 | 1201010403_14v2.2 | 큐티클 정리하기 | 3 |
| 3. 컬러링하기 | 3-1. 컬러링 매뉴얼 이해 3-2. 컬러링 방법 선정과 작업 3-3. 컬러링 작업 | 1201010403_14v2.3 | 컬러링 | 3 |
| 4. 보습제 도포하기 | 4-1. 보습제 선정과 도포 4-2. 각질제거 | 1201010403_14v2.4 | 보습제 바르기 | 2 |
| 5. 네일 기본관리 마무리하기 | 5-1. 유분기 제거 5-2. 네일 기본관리 마무리와 정리 | 1201010403_14v2.5 | 마무리하기 | 3 |

핵심 용어

프리에지, 니퍼, 푸시, 플리시, 네일 파일, 스웨이형, 스웨이 오프형, 라운드형, 오발형, 포인트형

학습모듈의 목표는

학습자가 해당 학습모듈을 통해 성취해야 할 목표를 제시한 것으로, 교수자는 학습자가 학습모듈의 전체적인 내용흐름을 파악할 수 있도록 지도하는 것이 필요합니다.

선수학습은

교수자나 학습자가 해당 모듈을 교수 또는 학습하기 이전에 이수해야 할 학습내용, 교과목, 핵심 단어 등을 표기한 것입니다. 따라서 교수자는 학습자가 개별 학습, 자기 주도 학습, 방과 후 활동 등 다양한 방법을 통해 이수할 수 있도록 지도하는 것이 필요합니다.

핵심 용어는

학습모듈을 통해 학습되고 평가되어야 할 주요 용어입니다. 또한 당해 모듈 또는 타 모듈에서도 핵심 용어를 사용하여 학습내용을 구성할 수 있으며, 「NCS 국가 직무능력표준」 사이트(www.ncs.go.kr)에서 색인(찾아보기) 중 하나로 이용할 수 있습니다.

3. NCS 학습모듈의 내용 체계

구 성

- NCS 학습모듈의 내용은 크게 **학습**, **학습 내용**, **교수·학습 방법**, **평가** 로 구성되어 있습니다.

| | |
|-----------------|--|
| 학습 | 해당 NCS 능력단위요소 명칭을 사용하여 제시한 것입니다. 학습은 크게 학습 내용, 교수·학습 방법, 평가로 구성되며 해당 NCS 능력단위의 능력단위 요소별 지식, 기술, 태도 등을 토대로 학습 내용을 제시한 것입니다. |
| 학습 내용 | 학습 내용은 학습 목표, 필요 지식, 수행 내용으로 구성하였으며, 수행 내용은 재료·자료, 기기(장비·공구), 안전·유의 사항, 수행 순서, 수행 tip으로 구성한 것입니다. 학습모듈의 학습 내용은 업무의 표준화된 프로세스에 기반을 두고 실제 산업현장에서 이루어지는 업무활동을 다양한 방식으로 반영한 것입니다. |
| 교수·학습 방법 | 학습 목표를 성취하기 위한 교수자와 학습자 간, 학습자와 학습자 간의 상호 작용이 활발하게 일어날 수 있도록 교수자의 활동 및 교수 전략, 학습자의 활동을 제시한 것입니다. |
| 평가 | 평가는 해당 학습모듈의 학습 정도를 확인할 수 있는 평가 준거, 평가 방법, 평가 결과의 피드백 방법을 제시한 것입니다. |

활 용 안 내

예시 : 네일미용 세분류의 ‘네일 기본관리’ 학습모듈의 내용

| | |
|-------------|------------------------------------|
| 학습 1 | 프리에지 형태 만들기(LM1201010403_14v2.1) |
| 학습 2 | 큐티를 정리하기(LM1201010403_14v2.2) |
| 학습 3 | 컬러링하기(LM1201010403_14v2.3) |
| 학습 4 | 보습제 도포하기(LM1201010403_14v2.4) |
| 학습 5 | 네일 기본관리 마무리하기(LM1201010403_14v2.5) |

학습은

해당 NCS 능력단위요소 명칭을 사용하여 제시하였습니다. 학습은 일반교과의 '대단원'에 해당되며, 모듈을 구성하는 가장 큰 단위가 됩니다. 또한 완성된 직무를 수행하기 위한 가장 기본적인 단위로 사용할 수 있습니다.

학습내용은

요소 별 수행준거를 기준으로 제시하였습니다. 일반교과의 '중단원'에 해당합니다.

학습목표는

모듈 내의 학습내용을 이수했을 때 학습자가 보여줄 수 있는 행동수준을 의미합니다. 따라서 일반 수업시간의 과목목표로 활용할 수 있습니다.

필요지식은

해당 NCS의 지식을 토대로 해당 학습에 대한 이해와 성과를 높이기 위해 알아야 할 주요 지식을 제시하였습니다. 필요지식은 수행에 꼭 필요한 핵심 내용을 위주로 제시하여 교수자의 역할이 매우 중요하며, 이후 수행순서 내용과 연계하여 교수·학습으로 진행할 수 있습니다.

3-1. 컬러링 매뉴얼 이해

학습목표

- 고객의 요구에 따라 네일 폴리시 색상의 침착을 막기 위한 베이스코트를 아주 얇게 도포할 수 있다.
- 작업 매뉴얼에 따라 네일 폴리시를 열매 없이 균일하게 도포할 수 있다.
- 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 톱코트를 바를 수 있다.

필요 지식 /

□ 컬러링 매뉴얼

컬러링 작업 전, 이세론 또는 네일 폴리시 리무버를 사용하여 손톱표면과 큐티클 주변, 손톱 밑 부분까지 깨끗하게 유분기를 제거해야 한다. 컬러링의 순서는 Base coating 1회 → Polishing 2회 → 컬러수정 → Top coating 1회 → 최종수정의 순서로 한다. 베이스코트는 착색을 방지하고 발림성 향상을 위해 가장 먼저 도포하며 컬러링의 마지막에 컬러의 유지와 광택을 위해 톱코트를 도포한다. 네일 보강제(Nail Strengthner)를 바를 시에는 베이스코트를 도포하기 전에 사용한다.

수행 내용 / 컬러링 매뉴얼 실습하기

재료·자료

- 컬러링 관련 네일 미용 자료들
- 컬러바구니, 베이스코트, 네일 폴리시, 톱코트, 오렌지우드스티, 단자면, 폴리시리무버, 디스펜서 등

기기(장비·공구)

- 컴퓨터, 빔 프로젝터, 스크린 등

안전·유의사항

- 컬러링 재료들의 냄새를 직접적으로 맡지 않도록 유의한다.
- 컬러링 제품들이 대부분 유리병에 들어 있기 때문에 깨지지 않도록 각별히 조심한다.
- 컬러링 제품들은 상온에 마르기 때문에 개봉 후 뚜껑을 잘 닫도록 한다.

수행 순서

[1] 네일 폴리시를 바르게 짚는다.

1. 손바닥에 네일 폴리시를 놓고 약지 소지를 이용하여 네일 폴리시를 짚는다.
2. 폴리시를 왼 손의 엄지와 검지로 고객과 작업손가락을 짚는다.
3. 폴리시를 왼 손의 중지 손가락을 굳게 펴서 받침대가 되도록 한다.
4. 반대편 손으로 네일 폴리시의 뚜껑을 열고 소지 손가락을 펴서 네일 폴리시를 왼 중지 손가락 위에 받쳐놓는다.
5. 다양한 형태의 폴리시를 잡아본다.

수행 tip

- 흰색이 많이 섞인 네일 폴리시의 경우는 붓의 각도를 높이 세워 빠르게 브러시 작업을 해야 붓 자국이 나지 않는다.
- 컬러링은 기본 2회 정도이나 컬러에 따른 도포량과 컬러감에 따라 1~3회 사이로 증감할 수 있다.

수행 내용은

모듈에 제시한 것 중 기술(Skill)을 습득하기 위한 실습 과제로 활용할 수 있습니다.

재료·자료는

수행 내용을 수행하는데 필요한 재료 및 준비물로 실습 시 필요 준비물로 활용할 수 있습니다.

기기(장비·공구)는

수행 내용을 수행하는데 필요한 기본적인 장비 및 도구를 제시하였습니다. 제시된 기기 외에도 수행에 필요한 다양한 도구나 장비를 활용할 수 있습니다.

안전·유의사항은

수행 내용을 수행하는데 안전상 주의해야 할 점 및 유의사항을 제시하였습니다. 수행 시 유념해야 하며, NCS의 고려사항도 추가적으로 활용할 수 있습니다.

수행 순서는

실습과제의 진행 순서로 활용할 수 있습니다.

수행 tip은

수행 내용에서 수행의 수월성을 높일 수 있는 아이디어를 제시하였습니다. 따라서 수행tip은 지도상의 안전 및 유의사항 외에 전반적으로 적용되는 주의점 및 수행과제 목적에 대한 보충설명, 추가사항 등으로 활용할 수 있습니다.

학습3 교수·학습 방법

교수·학습 방법은

학습목표를 성취하는데 필요한 교수 방법과 학습 방법을 제시하였습니다.

교수 방법

- 컬러링 제품의 성분과 컬러별 점도의 차이, 베이스코트와 톱코트의 역할, 폴리시 짚는 방법, 큐어링 시간 등의 내용을 화면 자료와 함께 설명한다.
- 서식지를 활용하여 네일 컬러링 방법을 그림으로 그려 보게 한 뒤, 다양한 컬러링의 매뉴얼을 그려서 숙지하도록 한다.
- 겔 컬러링 시 주의사항을 계속 숙지시키도록 하며, 큐어링 시간에 대해 작성하도록 한다.

교수 방법은

해당 학습활동에 필요한 학습내용, 학습내용과 관련된 학습 자료명, 자료 형태, 수행내용의 진행 방식 등에 대하여 제시하였습니다. 또한 학습자의 수업참여도를 제고하기 위한 방법 및 수업진행상 유의사항 등도 제시하였습니다. 선수학습이 필요한 학습을 학습자가 숙지하였는지 교수자가 확인하는 과정으로 활용할 수도 있습니다.

학습 방법

- 컬러링을 위한 재료의 필요성과 사용방법을 숙지하고 컬러링 매뉴얼 과정에 맞추어 작업 내용을 이해한다.
- 컬러링의 다양성에 대한 용어를 숙지하고 진행과정에 맞추어 내용을 작업한다.
- 겔 컬러링 시 적합한 큐어링 시간을 선택해서 큐어링 해본다.

학습 방법은

해당 학습활동에 필요한 학습자의 자기주도적 학습 방법을 제시하였습니다. 또한 학습자가 숙달해야 할 실기능력과 학습과정에서 주의해야 할 사항 등으로 제시하였습니다. 학습자가 학습을 이수하기 전에 반드시 숙지해야 할 기본 지식을 학습하였는지 스스로 확인하는 과정으로 활용할 수 있습니다.

학습3 평가

평가 준거

- 평가자는 학습자가 학습 목표 및 평가 항목에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

| 학습내용 | 평가항목 | 성취수준 | | |
|------------|---|------|---|---|
| | | 상 | 중 | 하 |
| 컬러링 매뉴얼 이해 | <ul style="list-style-type: none"> 고객의 요구에 따라 네일 폴리시 색상의 질감을 만들기 위한 베이스코트를 아주 얇게 도포할 수 있다. 작업 매뉴얼에 따라 네일 폴리시를 일찍 얹어 균일하게 도포할 수 있다. 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 톱코트를 바를 수 있다. | | | |

평가 방법

- 작업장 평가

| 학습내용 | 평가항목 | 성취수준 | | |
|------------|---|------|---|---|
| | | 상 | 중 | 하 |
| 컬러링 매뉴얼 이해 | <ul style="list-style-type: none"> 고객의 요구에 따라 네일 폴리시 색상의 질감을 만들기 위한 베이스코트를 아주 얇게 도포할 수 있다. 작업 매뉴얼에 따라 네일 폴리시를 일찍 얹어 균일하게 도포할 수 있다. 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 톱코트를 바를 수 있다. | | | |

피드백

- 작업장 평가
 - 작업 결과물을 확인하여 수정사항을 제시하고 수정 부분을 인지하도록 한다.

평가는

해당 NCS 능력단위 평가방법과 평가 시 고려 사항을 준용하여 작성하였습니다. 교수자 및 학습자가 평가항목 별 성취수준을 확인하는데 활용할 수 있습니다.

평가 준거는

학습자가 해당 학습을 어느 정도 성취하였는지를 평가하기 위한 기준을 제시하고 있습니다. 학습목표와 연계하여 단위수업 시간에 평가항목 별 성취수준을 평가하는데 활용할 수 있습니다.

평가 방법은

NCS 능력단위의 평가방법을 준용하였으며, 평가 준거에 따른 평가방법을 2개 이상 제시하였습니다. 평가방법으로는 포트폴리오, 문제해결 시나리오, 서술형 시험, 논술형 시험, 사례연구, 평가자 체크리스트, 작업장 평가 등이 있으며, NCS의 능력단위 요소 별 수행 수준을 평가하는데 가장 적절한 방법을 선정하여 활용할 수 있습니다.

피드백은

평가 후에 학습자들에게 평가 결과를 피드백하여 부족한 부분을 알려주고, 학습 결과가 미진한 경우, 해당 부분을 다시 학습하여 학습목표를 달성하는 데 활용할 수 있습니다.

4. 참고 자료

참고자료

- 김미원(2011). 『Nail Study』. 서울: 사)한국네일저서서비스협회.
- 민광경(2015). 『미용사(네일)평가』. 서울: 예문사.
- 박은주(2014). 『네일미용』. 서울: 정담미디어.

참고자료는


해당 학습모듈의 필요지식에 대한 출처와 인용한 참고자료 및 사이트를 제시하였습니다.

5. 활용 서식/부록

활용서식

프리에지 형태 실습지

1. 프리에지 형태의 이해

| 모양 | 이름 | 특징 |
|---|-----------------------|--|
|  | { Square nail } | -강한 느낌의 사각형태 -네일의 양끝 모서리 부분이 90° 사각의 형태이다. { } -발톱이 형태 활용 -내인성 발톱의 보정시에 적용 |

활용서식은

평가 서식, 실습시트 등 교수학습 시 활용 가능한 다양한 서식들로 구성하였습니다. 과제 진행에서 평가에 이르기까지 필요한 서식을 해당 학습모듈의 특성에 맞춰 개발하거나 기존의 양식을 활용하여 제시하였습니다.

부록

네일 기본관리 도구와 재료 목록

| 목록 | 비고 | 준비 |
|--------|---------------------|--------|
| 위생가운 | 흰색 | 작업자 착용 |
| 위생 마스크 | 흰색 | 작업자 착용 |
| 보호안경 | 투명한 렌즈 (안경으로 대체 가능) | 작업자 착용 |
| 재용접반함 | 재용, 색상 무관 | 작업대 |

부록은

활용서식 이외에 교수학습과정에서 참고할 수 있는 자료가 있는 경우 제시하였습니다.

[NCS-학습모듈의 위치]

| | | |
|-----|--------|--|
| 대분류 | 정보통신 | |
| 중분류 | 정보기술 | |
| 소분류 | 정보기술개발 | |

| 세분류 | 능력단위 | 학습모듈명 |
|----------------|-------------|-------------|
| SW아키텍처 | | |
| 응용SW 엔지니어링 | 요구사항확인 | 요구사항확인 |
| 시스템 엔지니어링 | 애플리케이션 설계 | 애플리케이션 설계 |
| DB엔지니어링 | 애플리케이션 구현 | 애플리케이션 구현 |
| NW엔지니어링 | 화면 구현 | 화면 구현 |
| 보안 엔지니어링 | 데이터 입출력 구현 | 데이터 입출력 구현 |
| UI/UX 엔지니어링 | 통합 구현 | 통합 구현 |
| 시스템SW 엔지니어링 | 개발자 테스트 | 개발자 테스트 |
| | 정보시스템 이행 | 정보시스템 이행 |
| | 제품소프트웨어 패키징 | 제품소프트웨어 패키징 |
| | 소프트웨어공학 활용 | 소프트웨어공학 활용 |

차 례

| | |
|-------------------------------|----|
| 학습모듈의 개요 | 1 |
| 학습 1. 개발자 테스트 케이스 설계하기 | |
| 1-1. 개발자 테스트 케이스 작성 | 3 |
| 1-2. 개발자 테스트 시나리오 작성 | 20 |
| • 교수·학습 방법 | 32 |
| • 평가 | 33 |
| 학습 2. 개발자 통합테스트하기 | |
| 2-1. 개발자 통합테스트 수행 | 35 |
| 2-2. 개발자 테스트 결과 분석 | 49 |
| • 교수·학습 방법 | 59 |
| • 평가 | 60 |
| 학습 3. 개발자 결함조치하기 | |
| 3-1. 개발자 테스트 결함 관리 | 62 |
| • 교수·학습 방법 | 71 |
| • 평가 | 72 |
| 참고 자료 | 74 |
| 활용 서식 | 75 |

개발자 테스트 학습מוד의 개요

학습מוד의 목표

요구사항대로 응용소프트웨어가 구현되었는지를 검증하기 위해서 테스트 케이스를 작성하고, 통합테스트를 수행하여 결함을 발견하고 결함을 조치할 수 있다.

선수학습

소프트웨어 아키텍처, 테스트 지식체계, 테스트 레벨(단위/통합/시스템/인수)별 테스트 접근 방법, 결함관리

학습מוד의 내용체계

| 학습 | 학습 내용 | NCS 능력단위요소 | | |
|---------------------|---------------------|-------------------|------------------|----|
| | | 코드번호 | 요소명칭 | 수준 |
| 1. 개발자 테스트 케이스 설계하기 | 1.1 개발자 테스트 케이스 작성 | 2001020207_14v2.1 | 개발자 테스트 케이스 설계하기 | 4 |
| | 1.2 개발자 테스트 시나리오 작성 | | | |
| 2. 개발자 통합 테스트하기 | 2.1 개발자 통합테스트 수행 | 2001020207_14v2.2 | 개발자 통합테스트하기 | 3 |
| | 2.2 개발자 테스트 결과 분석 | | | |
| 3. 개발자 결함 조치하기 | 3.1 개발자 테스트 결함 관리 | 2001020207_14v2.3 | 개발자 결함조치하기 | 4 |

핵심 용어

테스트 케이스, 테스트 시나리오, 테스트 환경 구축, TMMi Model, 통합테스트, 테스트 커버리지, 테스트 자동화, 결함 관리, 형상 관리

학습 1

개발자 테스트 케이스 설계하기 (LM2001020207_14v2.1)

학습 2

개발자 통합테스트하기(LM2001020207_14v2.2)

학습 3

개발자 결함조치하기(LM2001020207_14v2.3)

1-1. 개발자 테스트 케이스 작성

학습 목표

- 개발하고자 하는 응용소프트웨어의 특성을 반영한 테스트 방식, 대상과 범위를 결정하여 테스트 케이스를 작성할 수 있다.

필요 지식 /

① 소프트웨어 테스트의 개요

1. 테스트(Testing)의 개념

(1) 일반 개념

(가) 테스트는 일반적으로 작업이 끝난 후, 처음에 요구된 것과 현재 상태의 차이를 발견하기 위해 수행하는 활동이다.

(나) 즉, 제품이 만들어지고 난 뒤 요구된 기능들이 제대로 작동하는지를 확인하는 활동도 테스트의 일종이라고 할 수 있다.

(2) 전통적인 테스트 개념과 현재의 테스트 개념

(가) 소프트웨어 테스트

소프트웨어 테스트는 응용 프로그램 또는 구성요소를 포함한 시스템의 동작과 성능, 안정성이 사용자가 요구하는 수준을 만족하는지 확인하기 위해 결함을 발견하는 메커니즘이다.

(나) 전통적인 테스트 개념

전통적인 테스트 개념은 응용 프로그램 또는 시스템의 정상 작동 여부 확인이다.

(다) 현대적인 테스트 개념

현재의 테스트는 사용자의 기대수준과 요구사항에 맞게 구현되고 동작하는지를 확인하고, 이를 통해 결함을 발견하여, 최종적으로는 결함 데이터를 근간으로 개발 프로젝트의 리스크 정보를 정량적 수치로 의사결정권자(프로젝트 관리자 등)에게

전달하는 것이다.

(라) 결함 예방 활동

개발 프로젝트 초기에 개발 산출물을 테스트 관점에서 리뷰하고, 테스트 케이스를 만드는 과정에서 결함을 발견하는 작업(결함 예방 활동)도 테스트 활동의 중요한 부분으로 인식되고 있다.

2. 테스트의 필요성

(1) 소프트웨어 시스템 관점

(가) 소프트웨어 시스템은 비즈니스 애플리케이션에서 소비자 제품에 이르기까지 생활의 많은 부분에서 사용되고 있으며, 그 비중은 계속해서 증가하고 있다.

(나) 소프트웨어가 올바르게 동작하지 않는 경우 다양한 문제가 발생하며, 이로 인한 피해는 금전적인 손실, 시간 낭비, 비즈니스의 이미지 손상, 그리고 부상이나 사망에 이르기까지 다양하고 심각하다.

(다) 테스트는 이러한 소프트웨어 시스템의 문제를 최소화하기 위해 반드시 필요하다.

(2) 소프트웨어 결함의 원인

(가) 개발자는 소프트웨어나 시스템 또는 문서를 작성할 때 결함을 만드는 오류를 범할 수 있다.

(나) 코드에 존재하는 결함은 장애의 원인이 된다. 이때 시스템은 의도된 대로 동작하지 않거나, 동작하지 말아야 함에도 동작한다.

(다) 소프트웨어, 시스템, 문서의 결함은 장애의 원인이 되지만, 모든 결함이 장애를 일으키는 것은 아니다.

(라) 결함은 인간이 오류를 범하기 쉽기 때문에 발생하며, 시간적인 압박, 복잡한 코드, 기반환경(Infrastructure)의 복잡성, 기술이나 시스템의 변경, 그리고 수많은 시스템 상호 간의 연동 등의 이유로 발생한다.

(마) 장애는 이와 같은 결함에 의해서 뿐만 아니라 환경적인 조건에 의해서도 발생한다.

(바) 즉, 방사, 자기, 전자기장, 물리적 오염 또한 소프트웨어의 결함을 유발시킬 수 있으며, 이러한 환경적인 조건이 하드웨어 조건을 변경시켜 소프트웨어의 실행에 영향을 미칠 수 있다.

(3) 소프트웨어 테스트의 역할

(가) 발견하지 못했던 시스템 또는 문서의 결함들을 체계적인 테스트를 통해 배포 이전에 발견하고 수정한다면, 운영 환경 내에서 발생하는 결함들의 리스크를 줄이는 데 기여할 수 있으며, 소프트웨어 품질 향상에도 도움을 준다.

(나) 소프트웨어 개발 과정에서는 테스트가 개발 초기의 요구사항 분석 단계부터 리뷰와 정적분석을 통해 정적으로 시작될 수 있으며, 각각의 개발 단계에 대응하는 테스트 레벨(Test level)에 따른 테스트가 이루어진다.

(다) **컴포넌트(단위) 테스트와 통합 테스트**는 개발조직이 중심이 되어 수행되고, 시스템이 갖추어진 이후의 테스트는 개발조직의 지원을 받아 독립성을 가진 테스트 조직을 중심으로 수행하는 것이 바람직하다.

(라) 최종 소프트웨어 사용자가 인수하는 과정의 테스트도 전체 개발과정의 테스트 중 하나이다.

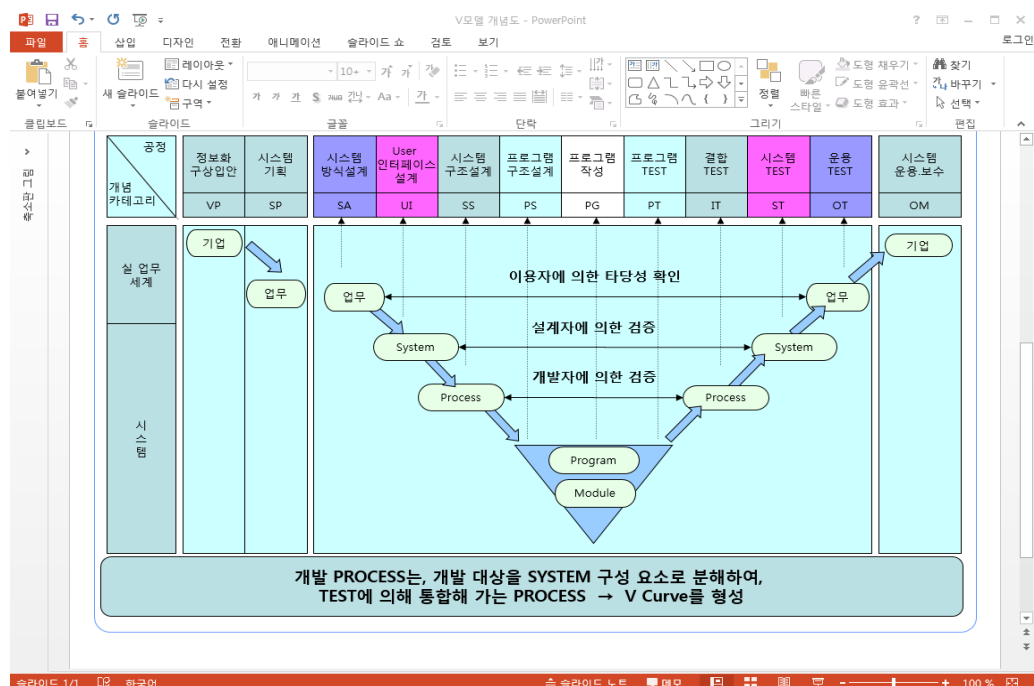
(마) 이와 같이 테스트 레벨에 따른 테스트는 개발과정에서 소프트웨어의 품질을 높이고, 소프트웨어가 고객에게 전달된 이후에 결함이 발생할 가능성을 최소화한다.

(4) V-모델과 단계별 테스트 레벨

(가) V-모델은 요구사항 정의 및 분석, 시스템 설계, 구현, 테스트이라는 일련의 단계를 통해 소프트웨어(시스템)를 개발하는 폭포수 개발 모델(Waterfall Model)에 근간을 두고 있다.

(나) 여기서 테스트는 한 번에 이루어지는 것이 아니라, 각각의 개발 단계에 대응하는 테스트 레벨이 별도로 존재하여 [그림 1-1]과 같이 V모양을 이룬다.

(‘[그림 1-1] V모델 개념도’ 참조)



[그림 1-1] V-모델 개념도 (실무 작성 사례)

※ MS로부터 이용 허락을 받았습니다.

- (다) 여러 가지 변형된 형태의 V-모델이 존재하지만 일반적인 유형의 V-모델에서는 4단계의 테스트 레벨을 제시하고 있고, 이들은 개발 단계의 요구사항 분석, 논리 설계, 물리 설계, 프로그램 코딩 등 4단계의 개발 활동으로 이루어졌으며, 대부분의 경우 일대일 대응된다.

3. 소프트웨어 테스트의 기본 원칙

(1) 검증과 확인

- (가) 소프트웨어 테스트는 흔히 검증(Verification)과 확인(Validation)이라고 하는 광범위한 주제의 한 요소이다.
- (나) 검증(Verification)은 소프트웨어가 특정 기능을 올바르게 구현하였는지를 보장하는 일련의 작업을 의미한다.
- (다) 확인(Validation)은 개발된 소프트웨어가 고객의 요구사항에 맞는지를 보장하는 또 다른 일련의 작업을 의미한다.
- (라) 검증과 확인은 다양한 SQA(Software Quality Assurance, 소프트웨어 품질보증) 활동을 포함한다.
- (마) 기술적 검토, 품질과 형상 감사, 성능 모니터링, 시뮬레이션, 타당성 조사, 문서 검토, 데이터베이스 검토, 알고리즘 분석, 개발 테스트, 사용성 테스트, 적격성 테스트, 인수 테스트, 그리고 설치 테스트 등 테스트가 V&V에서 매우 중요한 역할을 하고 있지만, 다양한 다른 활동들 역시 필요하다.

(2) 테스트 원리

- (가) 테스트는 결함이 존재함을 밝히는 활동이다.
결함이 없다는 것은 증명할 수 없다.
- (나) 완벽한 테스트는 불가능하다.
리스크 분석과 결정된 우선순위에 테스트를 집중해야 한다.
- (다) 테스트는 개발 초기에 시작한다.
개발 시작과 동시에 테스트를 계획, 전략적으로 접근해야 한다.
- (라) 결함 집중(Defect Clustering)
적은 수의 모듈에서 대다수의 결함이 발견된다.
- (마) 살충제 패러독스(Pesticide Paradox)
동일한 테스트를 반복적으로 수행하면 버그를 찾기 힘들다.
- (바) 테스트는 정황(Context)에 의존적이다.
효율적, 효과적 테스트 팀 조직과 독립적 테스트 환경이 필요하다.
- (사) 오류-부재의 궤변(Absence of Errors Fallacy)
사용자 요구사항에 맞지 않는다면 결함을 찾고 수정하는 것은 무의미하다. 즉, 결함을 모두 발견했다고 해도 품질이 높다고 할 수 없다.

(3) 테스트 용이성

테스팅의 목표는 오류를 발견하는 것이며, 훌륭한 테스트는 오류를 발견할 확률이 높은 것이다. 따라서 개발자는 “테스트 용이성”을 염두에 두고 컴퓨터 기반 시스템이나 제품을 설계하고 개발하여야 한다. 동시에 테스트 자체도 최소한의 노력으로 최대의 오류를 찾는 목표를 달성하는 일련의 특성을 보여야 한다.

테스트 가능한 소프트웨어의 특징은 다음과 같다.

(가) 운용성(Operability)

시스템이 품질을 염두에 두고 설계되고 구현된다면, 상대적으로 버그가 적어 테스트 실행은 방해받지 않으며, 따라서 테스트는 계속해서 진행될 수 있다.

(나) 관찰가능성(Observability)

테스팅의 일부분으로 제공된 입력은 뚜렷하게 구별되는 출력을 생산한다. 시스템 상태와 변수들은 실행 중에 보거나 질의할 수 있다. 부정확한 출력은 쉽게 알 수 있다. 내부 오류는 자동으로 발견되고 보고된다. 소스 코드는 접근 가능하다.

(다) 제어가능성(Controlability)

모든 가능한 출력은 입력의 조합에 따라 만들어질 수 있고, 입출력 형식은 일관적이고 잘 구성되어 있다. 모든 코드는 입력의 조합에 따라 실행 가능하다. 소프트웨어와 하드웨어의 상태 및 변수는 테스트 공학자에 의해 직접 제어될 수 있다. 테스트는 쉽게 명세화되고 자동화되며 재생산이 가능해진다.

(라) 분해가능성(Decomposability)

소프트웨어 시스템은 독립적으로 테스트 가능한 독립된 모듈로 구성되어 있다.

(마) 단순성(Simplicity)

프로그램은 기능적 단순성(Functional Simplicity, 예; 요구사항을 만족시키기 위한 특성들의 집합을 최소한으로 한다.), 구조적 단순성(Structural Simplicity, 예; 결함이 전파되지 않도록 아키텍처를 모듈화한다.), 코드 단순성(Code Simplicity, 예; 코딩 표준이 채택되어 조사와 유지보수를 쉽게 할 수 있다.)을 보여야 한다.

(바) 안정성(Stability)

소프트웨어에 대한 변경은 자주 일어나지 않으며, 변경이 일어나더라도 관리가 잘 되어서 기존의 테스트를 무효로 만들지 않는다. 소프트웨어는 장애로부터 쉽게 복구된다.

(사) 이해가능성(Understandability)

내부, 외부 및 공유 컴포넌트들의 구조적 설계와 의존 관계가 잘 파악되어 있다. 기술적 문서는 계속 접근 가능하고, 잘 구성되어 있으며, 상세하고 정확하다. 설계에 대한 변경은 테스터에게 알려진다.

② 소프트웨어 테스트 기법의 분류

1. 소프트웨어 내부 구조 참조 여부에 따른 분류

(1) 블랙박스 기법

테스트 대상의 내부구조(코드)를 참조하지 않고 테스트 베이스(Test Basis), 그리고 개발자와 테스터, 사용자들의 경험을 바탕으로 기능적 혹은 비기능적 테스트 케이스를 도출하고 선택하는 방법이다.

(2) 화이트박스 기법

화이트박스 기법은 컴포넌트(단위) 또는 소프트웨어(시스템)의 구조(코드)를 중심으로 테스트 케이스를 도출하는 방법이다.

<표 1-1> 블랙박스/화이트박스 기법 비교

| 구분 | 블랙박스 기법 | 화이트박스 기법 |
|----------|----------------------------------|----------------------------------|
| 검사 종류 | 동등 분할 | 기초 경로 검사 |
| | 경계값 분할 | 조건 검사 |
| | 원인-결과 그래프 기법 | 데이터 흐름 검사 |
| | 비교 검사 | 루프 검사 |
| 처리 | 프로그램의 외부 명세에 근거 (프로그램 기능 명세서) | 프로그램의 내부 명세에 근거 (원시 프로그램 리스트) |
| 기준 | What (무엇을 수행하는가?) | How (어떻게 처리되는가?) |
| 검사 | 기능 검사 | 구조 검사 |
| 시점 | 검사단계 후반부에서 수행 | 검사단계 전반부에서 수행 |
| 구조 | 제어구조 무시 (정보 영역에 초점) | 정보영역보다 제어구조 중시 |

2. 테스트 설계의 근원에 따른 분류

(1) 명세 기반 기법

(가) 개요

- 1) 해결할 문제를 명세하기 위해 공식적이거나 비공식적인 모델을 사용하며, 이러한 모델에서 테스트 케이스를 시스템적으로 도출하는 것이 가능하다.
- 2) 커버리지를 측정할 수 있으나 그 의미가 구조 기반 기법의 커버리지에 비해 제한적이다.(상태 전이 커버리지, 결정 테이블 커버리지, 요구사항 커버리지)

(나) 등가 분할 (Equivalence Partitioning)

- 1) 입력값/출력값 영역(Input/Output Space)을 유한 개의 상호 독립적인 집합 (Mutual Disjoint Subset)으로 나누어 수학적인 등가 집합을 만든 후, 각 등가 집합의 원소 중 대표값을 선택하여 테스트 케이스를 도출하는 방법이다.
- 2) 동등 분할 클래스는 유효한 입력 데이터뿐 아니라 유효하지 않은 입력 데이터 (입력되지 말아야 할 값)도 포함할 수 있다.

(다) 경계값 분석 (Boundary Value Analysis)

- 1) 동등 분할의 경계에서 결함이 발견될 확률이 높기 때문에 결함을 예방하기 위해 경계값까지 포함하여 테스트하는 기법이다.
- 2) 분할 영역의 최댓값과 최솟값은 그 영역의 경계값이 된다.

(라) 결정 테이블 테스트(Decision Table Testing)

- 1) 결정 테이블은 논리적인 조건이나 상황을 구현하는 시스템에서 요구사항을 도출하거나 내부 시스템 설계를 문서화하는 매우 유용한 도구이다.
- 2) 이것은 시스템이 구현해야 할 비즈니스 규칙을 문서화하는 데 사용된다.
- 3) 명세를 분석하고 시스템의 조건과 동작을 식별한다.
- 4) 결정 테이블에서 입력조건과 기대결과는 참과 거짓으로 표현된다.
- 5) 결정 테이블은 동작을 유발시키는 조건 또는 상황, 그리고 각 해당 조합의 예상 결과까지 포함한다.

(마) 상태 전이 테스트(State Transition Testing)

- 1) 시스템은 현재 상황과 이전의 이력을 반영하는 상태 및 그 변화에 따라 다르게 동작할 수 있다. 시스템의 이러한 측면을 상태 전이 다이어그램으로 표현할 수 있다.
- 2) 상태 전이 다이어그램을 통해 테스트 엔지니어는 소프트웨어 또는 시스템을 상태 사이의 관계 즉, 상태 간의 전이, 상태를 변화시키는 이벤트와 입력값, 상태의 변화로 유발되는 동작 등으로 파악한다.

(바) 유스케이스 테스트(Use Case Testing)

- 1) 유스케이스나 비즈니스 시나리오를 기반으로 테스트를 명세화할 수 있다.
- 2) 하나의 유스케이스는 액터와 액터 사이의 상호작용을 표현하고, 해당 상호작용은 시스템 유저에게 결과값을 제공한다.
- 3) 각각의 유스케이스는 그 유스케이스가 성공적으로 수행되기 위한 전제조건을 가지고 있다. 또 각각의 유스케이스는 임무를 완수한 후 후속조건을 가지면서 종료된다.

(사) 분류 트리 기법(CTM: Classification Tree Method)

- 1) 소프트웨어 일부 또는 전체를 트리(Tree) 구조로 분석 및 표현하고 이를 바탕으로 테스트 케이스를 도출하는 기법이다.

(아) 페어와이즈 조합 테스트(Pairwise Testing)

- 1) 커버해야 할 기능적 범위에 비해 상대적으로 적은 양의 테스트 세트를 구성하여 소프트웨어의 결함을 찾고 테스트에 대한 자신감을 얻을 수 있는 방법 중의 하나이다.
- 2) 대부분의 결함이 2개 요소의 상호작용에 기인한다는 것에 착안하여 2개 요소의 모든 조합을 다룬다.

(2) 구조 기반 기법

(가) 개요

- 1) 코드와 개발 설계 등의 소프트웨어 구현 정보로 테스트 케이스를 도출한다.
- 2) 수행된 테스트 케이스를 바탕으로 테스트 커버리지를 측정할 수 있으며, 커버리지를 높이기 위해 테스트 케이스를 시스템적으로 도출해 추가할 수 있다.

(나) 구문 테스트와 커버리지

- 1) 구문 커버리지는 테스트 스위트(Test Suite, 테스트 케이스 묶음)에 의해 실행된 구문이 몇 퍼센트인지를 측정하는 것이다.
- 2) 구문 테스트는 구문 커버리지를 늘리기 위해 특정 구문을 테스트하는 테스트 케이스를 도출하는 것이다.

(다) 결정 테스트와 커버리지

- 1) 결정 커버리지는 테스트 케이스 스위트(Suite, 묶음)에 의해 실행된 조건문 분기(if 구문의 참 혹은 거짓)가 전체 가능한 분기의 몇 퍼센트인지를 측정하고 평가하는 것이다.
- 2) 결정 커버리지는 결정 포인트(Decision Points) 내의 전체 조건식이 “참”과 “거짓”의 모든 값을 갖게 되어 모든 분기로 흐르게 되면 달성된다.
- 3) 결정 테스트는 결정 커버리지를 늘리기 위해 특정 조건문의 분기를 테스트하는 테스트 케이스를 도출하는 것이다.

(라) 조건 테스트와 커버리지

- 1) 조건 커버리지는 결정 포인트 내에 있는 개개의 개별 조건식이 “참”과 “거짓”의 모든 값을 갖게 되면 달성된다.
- 2) 다중 조건 커버리지는 결정 포인트 내에 있는 모든 개별 조건식의 모든 가능한 논리적인 조합을 고려한 강력한 커버리지를 의미한다.

(마) 변형 조건/결정 커버리지

- 1) 변형 조건/결정 커버리지(MC/DC)는 각 개별 조건식이 다른 개별 조건식에 영향을 받지 않고 전체 조건식의 결과에 독립적으로 영향을 주도록 함으로써 조건/결정 커버리지를 향상시킨 것으로 결정 커버리지, 조건 커버리지보다 강력하다.

(3) 경험 기반 기법

(가) 개요

- 1) 테스트 관련 인력의 지식이나 경험으로 테스트 케이스를 도출한다.
- 2) 테스터, 개발자, 사용자의 소프트웨어에 대한 지식이나, 소프트웨어에서 자주 발생하는 결함이나 결함 분포 등의 지식이 요구된다.

- 3) 테스트에 참고할 명세가 거의 없거나 불충분할 경우, 시간적인 압박이 심한 경우 유용하다.

(나) 탐색적 테스트 접근법

- 1) 테스트 케이스 기반의 공식적 테스트와 반대되는 개념의 테스트 접근법이다.
- 2) 테스트 케이스를 먼저 작성하지 않고, 테스트 대상 제품을 실행하면서 익숙해지는 것과 동시에 테스트를 설계하고 계획한다.
- 3) 테스트 케이스 작성 시간을 최소화하고 테스트 엔지니어의 발견적인(Heuristic) 지적 능력을 최대한 활용하여 테스트를 수행하는 방법이다.

(다) 오류 추정

- 1) 테스터가 테스트 대상 시스템을 완전히 이해하고 있다는 전제 하에 식별된 취약점에 기반하여 실시하는 테스트이다. 따라서 테스트의 마지막 단계에서 사용하는 것이 적절하다.
- 2) 오류 추정 기법을 구조적으로 적용하려면 가능한 오류를 모두 나열하고 이런 유형의 결함 또는 오류를 공격할 수 있도록 테스트를 설계해야 한다.
- 3) 이러한 결함이나 장애 리스트는 경험, 형식에 맞지 않거나 결함을 발생시키는 데이터와 장애 데이터, 소프트웨어가 왜 장애를 일으켰는지에 대한 일반적인 지식 등을 근거로 만든다.

(라) 체크리스트

- 1) 테스트하거나 평가해야 할 내용과 경험을 나열해 놓은 것을 의미한다.
- 2) 일반적으로 체크리스트는 체계적으로 도출되기보다는 테스트 경험과 노하우를 정리하고 목록화하여 다음 테스트에서 해당 내용을 누락 없이 검증하는 것을 목적으로 작성한다.

③ 테스트 케이스 작성 기법

1. 테스트 케이스의 개요

- (1) 테스트에서는 입력값을 주고, 그 실제 출력값이 예상 출력값과 같은지를 비교하게 된다.
- (2) 그래서 입력값은 무엇이고 예상되는 출력값은 무엇인지를 미리 정하게 되는데, 이것을 테스트 케이스라고 한다.
- (3) 일반적으로 테스트 계획 단계에서 테스트 케이스들을 만들고, 이 테스트 케이스들을 기반으로 테스트를 수행하는 것이다.

2. 테스트 오라클(Test Oracle)

(1) 개념

- (가) 테스트 대상 소프트웨어의 실제 결과와 비교할 목적으로 예상 결과를 결정하는 근거로 작성된다.
- (나) 벤치마크를 위한 기존 시스템, 사용자 매뉴얼 또는 개인의 전문 지식일 수 있으나, 코드(Code)가 될 수는 없다.

(2) 특징

- (가) 제한된 검증
모든 테스트 케이스별로 테스트 오라클을 작성하기가 현실적으로 어려움
- (나) 수학적 기법 적용
수학적인 기법을 적용하여 오라클의 값을 구할 수 있음

(3) 유형

- (가) 실제 오라클(True Oracle)
모든 입력값에 대해 결과가 맞는지 확인하는 것
- (나) 샘플링 오라클(Sampling Oracle)
특정값에 대해서만 결과가 맞는지 확인하는 것
- (다) 휴리스틱 오라클(Heuristic Oracle)
일정 단위로 결과값을 확인하는 것
- (라) 일관성 검사 오라클(Consistent Oracle)
변경이 있는 경우 전후의 결과값이 같은지 확인하는 것

수행 내용 / 테스트 개념 이해 및 테스트 케이스 작성하기

재료 · 자료

- 테스트 계획서, 설계서, 요구사항 명세서

기기(장비 · 공구)

- 컴퓨터, 프린터

안전 · 유의사항

- 테스트 케이스 작성시 중요한 요구사항의 테스트가 누락되지 않도록 유의한다.

수행 순서

① 테스트 요소를 식별한다.

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

1. 테스트 요소 식별

- (1) 테스트 대상 시스템 및 관련 비즈니스 위험을 식별
- (2) 테스트 수행 중에 평가되어야 하는 위험을 기반으로 테스트 요소를 식별
- (3) 심각도 및 위험도를 고려하여 테스트 요소를 분류

2. 식별된 테스트 요소에 대해 테스트 단계/유형별로 테스트 전략을 수립

- (1) 개발 프로젝트의 수명주기를 파악
- (2) 테스트 요소와 관련된 위험을 최소화하기 위해, 개발 프로젝트 수명주기의 각 단계에 적합한 테스트 유형을 결정
- (3) 테스트의 유형별로 테스트 방법을 결정

② 테스트 범위를 설정한다.

1. 개발 프로젝트의 범위 및 상위 요구사항을 분석하여 테스트 범위를 설정한다.
2. 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, 테스트하지 않을 항목의 특성을 명세화하는 기반 자료로 활용한다.

3. 테스트 체크리스트 작성 사례

| CRM IT Infra Checklist | | | | | | |
|---|--|-----|----|----|------|------|
| No. | IT Infra Checklist | Yes | No | NP | 진행상황 | |
| | | | | | 설계 | 개발 |
| Yes = 적용됨 No = 해당 범위 외 NP = 추후 검토 | | | | | | |
| 1. Member registration | | | | | | |
| 1. | 신규 회원 가입을 New membership launching 전에(11월 이후) 가능한지 여부(가입에 한함) | ★ | | | 설계완료 | 개발진행 |
| 1. | Web 회원이나 멤버십 회원이 신규 멤버십 프로그램에 재가입(Conversion) 할 때, 기존 고객정보의 이관여부 | ★ | | | 설계완료 | |
| 1. | 현행 멤버십 약관 변경(법률적 검토 포함) 및 고객 Permission 획득 방안프로세스 수립 여부 | ★ | | | 설계완료 | |
| 2. Point Reserve | | | | | | |
| 1. | 멤버 카드 미소지 시 포인트 적립 가능여부 | ★ | | | 설계완료 | |
| | Issue Log참고(사후 적립) | | | | 설계중 | |
| 1. | 매장 POS Off-line시 포인트 적립 가능성 여부(포인트 강제적립) | | ★ | | | |
| 1. | 특정일에 일괄 추가 포인트 적립 가능여부 | ★ | | | 설계완료 | |
| 1. | 포인트 적립(당일 AND /OR 익일 사용) 가능성 여부 | ★ | | | 설계완료 | |
| 1. | 고객 개인별 차등화 포인트 적립 여부 | | | ★ | | |
| 1. | 특정 고객 대상의 선 포인트 지급 가능성 여부 | | | ★ | | |
| 1. | 적립 포인트 불일치 시 보정기능(Point adjustment) | ★ | | | 설계중 | |
| 1. | 영수증에 포인트 적립 및 사용가능 포인트 등의 정보출력이 가능한지 | ★ | | | 설계중 | |
| 3. Point Redemption | | | | | | |
| 1. | 포인트를 현금처럼 사용가능 | ★ | | | 설계완료 | |
| 1. | 포인트를 무료메뉴제공으로 가능한지 | ★ | | | 설계완료 | |

[그림 1-2] 테스트 체크리스트 작성 사례

출처: 자체 제작

4. 테스트 케이스 작성 항목

(1) 테스트 케이스 명세서 공통 부분

(가) 작성 일자 및 문서 버전

문서 작성 일자, 수정 일자 및 현재 문서의 버전과 상태(초안, 리뷰 완료, 최종본 등) 기술

(나) 변경 기록

문서 변경 기록 및 변경 사유, 수정자, 역할 등 기술

(다) 승인

명세서의 승인자, 승인 관련자

(라) 범위

테스트 대상 소프트웨어 제품이나 시스템 아이템 및 기능 요약

(마) 담당 조직

테스트 케이스 명세서의 작성 및 배포 담당 조직

(바) 기타

참고 문서, 기타 컨텍스트, 명명 규칙 등

(2) 테스트 케이스 명세서 개요

(가) 테스트 케이스 고유 번호

테스트 케이스 각각의 고유 ID

(나) 목적

테스트 케이스의 목적이나 주요 중점 사항

(다) 입력 데이터

테스트 케이스 실행에 필요한 입력 데이터(입력값, 또는 트랜잭션 파일이나 상수 테이블 이름 등)

(라) 기대 결과

모든 출력 데이터와 기대 동작 및 기대 결과값

(마) 테스트 환경 요구사항

테스트 준비, 실행, 결과 기록 등을 위한 테스트 환경 요구사항

(바) 기타

테스트 프로시저 요구사항, 테스트 케이스 간 종속성, 품질 보증 요구사항 등 테스트 케이스와 관계있는 중요한 조건, 제약, 특성 등을 기술

③ 테스트 단계별 접근 방법을 설정한다.

프로젝트에서 적용할 테스트 단계와 단계별 테스트에 대한 수행방안을 설정한다.

1. 개발 수명주기별 테스트 단계 설정

(1) 개발 프로젝트에서 수행할 테스트 단계를 설정

(2) 시스템의 크기와 복잡도, 중요도, 개발 성숙도, 고객 요청사항, 테스트 일정, 테스트 환경 등을 고려

2. 각 단계별 테스트 수행방안 결정

(1) 개발 프로젝트의 수명주기를 파악하고 단계별로 필요한 테스트 유형 및 수행 방안을 결정한다.

(2) 단위 테스트

(가) 단위테스트는 V 모델 방식의 소프트웨어 개발에서 테스트 프로세스의 첫 단계로 에러를 줄이기 위한 의도로 작성된 코드에 대한 분석을 진행한다.

(나) 또 코드가 효율적으로 작성되었는지, 프로젝트 내에 합의된 코딩 표준을 준수하고 있는지도 검증한다.

(다) 모듈 설계 단계에서 준비된 테스트 케이스를 이용하며, 코드를 개발한 개발자가 직접 수행한다.

(3) 통합 테스트

(가) 통합테스트는 통합된 컴포넌트 간의 인터페이스와 상호 작용 상의 오류를 발견하는 작업을 수행한다.

(나) 아키텍처 설계 단계에서 준비된 테스트 케이스를 사용하여 테스트가 진행되며, 일반적으로 개발자가 진행한다.

(4) 시스템 테스트

(가) 시스템 테스트는 실제 구현된 시스템과 계획된 사양을 서로 비교하는 작업이다.

(나) 이 단계에서 시스템 테스트에 대한 설계가 시스템 설계 문서로부터 도출되어 사용된다.

(다) 때로 시스템 테스트는 자동화 도구를 이용하여 실시된다.

(라) 모든 모듈을 통합한 후에 시스템 레벨의 에러들이 이 테스트 단계를 통해 발견될 수 있다.

(마) 개발자와 다른 별도의 테스트 팀에 의해 수행된다.

(5) 인수 테스트

(가) 인수 테스트는 시스템이나 시스템의 일부 또는 특정한 비기능적인 특성에 대해 “확신(Confidence)”을 얻는 작업이다.

(나) 이 단계에서 결함을 찾는 것은 인수 테스트의 주목적이 아니다. 이 테스트는 시스템을 배포하거나 실제 사용할 만한 준비가 되었는 지에 대해 평가한다.

(다) 하지만 대규모의 통합 테스트를 시스템에 대한 인수 테스트 이후에 실행할 수 있기 때문에 인수 테스트가 반드시 최종 단계 테스트이라고는 할 수 없다.

(라) 인수 테스트의 전형적인 형태는 사용자 인수 테스트, 운영 상의 인수 테스트 등이 있다.

④ 테스트 계획서를 작성한다.

1. 품질 목표 수준 설정

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

2. 테스트 추진 체제 수립

상위 단계에서 설정한 테스트 단계 및 접근 방안을 기반으로, 테스트를 수행할 추진 조직 및 역할을 결정한다.

3. 테스트 산출물 책임자 선정

테스트 종류별 산출물을 선정하고, 작성 시점 및 책임자를 선정한다.

4. 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.

5. 총괄 테스트 계획서 작성

상위 단계를 기반으로 하여, 총괄 테스트 계획서를 작성한다. 총괄 테스트 계획서에는 식별한 테스트의 범위, 단계, 수행 방안, 추진 체제, 산출물, 책임자, 일정과 같은 테스트 전략에 대한 내용을 기술한다. 테스트 계획서의 작성 사례(목차)는 다음과 같다.

(1) 개요

(가) 범위

테스트 대상 시스템, 테스트 항목

(나) 목적

기대 결과 및 실제 결과 간 차이점 보고

(다) 전제조건

테스트 시작 조건의 충족, 테스트 기반 구조, 테스트 중단 조건, 테스트 재개 조건

(라) 문서 승인

작성, 리뷰, 승인

(2) 테스트 베이스

TPMS 요구명세서, 관련 기준 또는 표준, 설계 명세서, 프로젝트 계획서

(3) 테스트 전략

(가) 기능 테스트 전략

리스크 분석, 설계기 법 및 완료 조건 결정

(나) 비기능 테스트 전략

리스크 분석(전략 매트릭스 생성), 비기능 테스트 전략 수립, 테스트 규모 산정

(4) 일정 계획

활동에 대한 시작 및 종료 일자, 각 활동에 참여할 역할(인력)

(5) 프로젝트 위험 및 리스크

테스트를 위한 인도 날짜의 지연, 테스트 실행 기간 동안 개발자 가용, 도메인(현업) 전문가의 테스트 지원, 기능설계서와 사용자지침의 적시적인 제공 등

(6) 테스트 인프라

(가) 테스트 환경

(나) 테스트 툴 및 자동화 접근법

테스트 자동화 지원 도구 등

(다) 사무 환경

테스트 사무실 공간, 테스트 장비 등

(7) 테스트 조직

(가) 테스트 역할

테스트 매니저, 테스트 리더, 테스트 엔지니어, 도메인 전문가, 테스트 컨설턴트 등

(나) 조직 구조

테스트의 독립성과 전문성을 고려한 테스트 수행 조직의 구조

(다) 테스트 참여자

(라) 교육 훈련 요구사항 및 계획

교육 훈련 방법

(8) 테스트 산출물

(가) 테스트 보고 문서

테스트 계획서, 결함 보고서, 인시던트 보고서, 진행 보고서, 최종 보고서, 테스트 프로세스 리뷰 보고서

(나) 테스트 실행 관련 산출물

테스트 케이스, 테스트 프로시저(스크립트), 테스트 시나리오, 초기 테스트 데이터 집합

(다) 테스트 리포팅

충족된 테스트 완료 조건과 해당 시기, 잔존 결함의 평가, 테스트 수행의 경제적 이득, 부각된 리스크, 테스트된 소프트웨어에 대한 확신의 정도

(라) 테스트 프로세스 제어

테스트 진척도, 예산, 사용 시간 등

(9) 형상관리

테스트 수명주기에서 형상관리를 필요로 하는 테스트 관련 문서는 테스트 계획서, 테스트 케이스, 결함 리포트 등이다.

(10) 결함관리

(가) 결함 리포트

결함 내용, 테스트 케이스 식별번호, 결함 유형, 발견일, 심각도, 우선순위, 시정조치 예정일, 수정 담당자, 재테스트 결과, 종료일

(나) 결함관리 측정 지표

심각도 범주당 미해결 결함수, 심각도 범주당 기간 내 수정된 결함 수, 발견된 결함 전체수, 리스크 레벨별 잔존 결함의 심각도 수준 및 결함 수, 누적 결함 수

6. 총괄 테스트 계획서 검토 및 승인

(1) 총괄 테스트 계획서 검토

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

(2) 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.

수행 tip

- 각 단계별 테스트에 대한 계획은 개발 단계에 따라 작성되는 시점과 산출물이 다르다.
- 테스트는 결함이 없음을 보이려는 것이 아니며, 프로그램 내의 내부 조건 및 이벤트 발생 순서에 대한 조합이 무수히 많음으로 인하여 완벽한 테스트는 불가능하다는 것을 인지한다.
- 실무에서의 테스트는 의사 결정권자에게 소프트웨어 및 필요 일정, 비용에 대한 충분한 정보를 제공하여, 테스트 규모를 산정할 수 있도록 해야 한다.

1-2. 개발자 테스트 시나리오 작성

학습 목표

- 개발하고자 하는 응용소프트웨어의 특성을 반영한 테스트 방식, 대상과 범위가 적용된 시나리오를 정의할 수 있다.
- 개발자 테스트 수행에 필요한 테스트 데이터, 테스트 시작 및 종료 조건 등을 준비할 수 있다.

필요 지식 /

① 테스트 환경 구축

1. 테스트 환경 구축의 필요성

- (1) 소프트웨어의 성능을 평가한다. 요청된 부하(사용자 수)를 수용할 수 있는지를 테스트하고 요구된 응답 시간 내에 부하를 처리할 수 있는 지를 테스트한다.
- (2) 수정된 부분이 실제로 동작하는지, 특히 시스템의 나머지 부분과 연동하여 동작하는지와 이들이 다른 문제점을 일으키지는 않는지를 확인한다.
- (3) 개발 환경에서 작업한 업그레이드 스크립트가 올바르게 동작하는지를 확인한다.
- (4) 패치 업그레이드, 주요 버전 릴리스, 운영체제 패치 등과 같은 주요 작업이 시스템을 망가뜨리지 않도록 확인한다.

2. 테스트 환경 구축 시 유의점

- (1) 실제 시스템의 데이터를 대표할 수 있는 데이터를 사용한다.
 - (가) 테스트 데이터는 해당 도메인에 유사한 데이터라야 한다.
 - (나) 가장 좋은 테스트 데이터는 기존에 유사 서비스를 했던 시스템이 있을 때의 자료를 기반으로 작성하는 게 좋다.
 - (다) 기존 유사 서비스가 없는 경우에는 시스템 요구명세서의 용량 계획서를 참고하도록 한다. “전 세계 1억 명이 동시에 사용할 수 있도록 하는 시스템”과 같은 용량 목표는 테스트 데이터 준비에 유용하게 사용될 수 있다.

(2) 테스트 환경 구축

(가) 테스트 환경의 분리

- 1) 보통 테스트는 개발 환경에서 진행하거나 또는 오픈 전에 운영 환경에서 진행하는 것이 일반적이었다.

- 2) 그러나 Iteration 기반의 Agile 방법론을 사용한다면, 테스트를 진행하는 중에도 개발팀은 계속해서 다음 Iteration(Sprint) 개발을 하고 있기 때문에, 개발과 테스트가 상호 간섭을 하여 개발과 테스트가 원활하게 이루어지지 못하게 된다.
- 3) 이는 개발자와 테스트 엔지니어의 리소스 낭비로 이어지는 데, 이를 해결하기 위해서는 테스트 환경을 분리해서 운영하는 것이 좋다.

(나) 가상 머신 기반의 서버나 클라우드 환경의 이용

- 1) 비용적인 문제 때문에 테스트 환경을 별도로 운영하는 것을 꺼린다면, 가상 머신(Virtual Machine) 기반의 서버나 클라우드 환경을 이용하여 테스트 시에만 일시적으로 테스트 환경을 운용하는 것도 좋은 방법이다.
- 2) 단, 이 경우에는 다음 테스트에 대한 연속성을 위해서 테스트가 끝난 후에는 Virtual Machine Image를 저장해 놓고, 다음 테스트에 다시 로딩해서 사용해야 한다.

(다) 개발 환경, 테스트 환경, 실 운영 환경의 분리

- 1) 개발을 위한 개발환경, 테스트를 위한 전용 테스트 환경과 실 운영 환경은 가능하면 물리적으로 분리되는 것이 좋다.
- 2) 네트워크 대역을 나누고 공유 디스크를 분리하는 등의 작업이 필요한데, 특히 부하 테스트 시에 네트워크의 대역폭이 급격하게 차거나 CPU 사용률이 급격하게 올라가는 경우, 이 환경들을 공유했다면 다른 환경에 영향을 줘서 뜻하지 않은 장애나 결과를 유발할 수 있다.

(라) 네트워크의 분할과 공유디스크 관리

- 1) 물리적인 환경 분리가 불가능하다면, 네트워크는 VLAN과 같이 소프트웨어를 이용하여 논리적으로 분할을 하고, 공유 디스크의 경우, IO Segregation 기법을 사용하는 것도 좋다.
- 2) IO Segregation이란, 공유 디스크로 서버에 Attach되는 볼륨에 대해서 별도의 물리적 디스크로 분리하여 서로 간의 IO 성능 간섭을 줄이는 방법 중의 하나이다.

(마) 연동 시스템의 테스트 환경

- 1) 다른 시스템과 연동을 하는 시스템의 경우에는 연동 시스템에 대한 테스트 환경이 별도로 구성되어 있어야 한다.
- 2) 만약에 연동 대상 시스템도 개발 중이거나 아직 배포가 불가능한 경우에는 Mock-up 환경이라는 것을 만드는데, Mock-up 환경은 연동 테스트를 위해서 사용하는 일종의 껍데기 환경으로, 특정 Input에 대해서 기계적인 Output 만을 내도록 하는 테스트 전용 환경이다.
- 3) Moke-up 환경을 먼저 구축하여 테스트를 진행한 후에, 점차적으로 실 연동 환경으로 전환하는 방법을 사용하도록 한다.

② 테스트 시나리오 작성

1. 테스트 시나리오의 개념

여러 개의 테스트 케이스들의 집합을 수행하기 위한 동작 순서를 기술한 문서를 말하며 테스트 절차 명세(Test Procedure Specification)라고 볼 수 있다.

2. 테스트 시나리오 작성 시 유의점

- (1) 테스트 그룹은 고객의 요구사항과 설계문서 사양 등을 토대로 테스트 시나리오를 작성한다.
- (2) 테스트 항목은 고객 요구사항 및 설계문서, 사양 등의 어떤 부분을 참고하였는지 알 수 있도록 작성한다.
- (3) 각 테스트 항목은 고유 ID, 분류 코드, 절차 번호, 기대 결과, 기능/성능/안정성/UI 구분, 필수 수행항목 등을 포함하여 작성한다.
- (4) 작성된 테스트 시나리오는 기본적으로 테스트 그룹 리더의 검토 작업을 통해 수정하며, 필요에 따라 프로젝트 관리자, 품질보증, 개발 그룹의 검토를 거쳐 수정하여 작성을 완료한다.

③ 소프트웨어 시험 문서화 표준(TTA, IEEE 829 기준)

1. 개요

(1) 표준 제정 배경

- (가) 소프트웨어의 품질을 평가하고 향상시키기 위한 기술 개발의 필요성이 대두되었다.
- (나) 소프트웨어 개발 업체의 품질 인식을 높이고, 그 결과로 생산되는 소프트웨어 제품의 품질을 제고시킬 수 있는 시험 문서화에 대한 표준의 개발과, 국내 업계로 시급한 보급의 필요성을 인식하였다.
- (다) 사실상 국제 표준인 IEEE 829(IEEE Standard for Software Test Documentation)를 국내 실정에 맞도록 한글화함으로써 시험 문서화 표준 초안을 마련하고 2002년 9월에 TTA 표준으로 채택되었다.

(2) 표준의 내용

- (가) 국제 표준인 IEEE 829를 대부분 수용한 본 표준은 일련의 기본적인 소프트웨어 테스트 문서들에 대한 형식과 내용을 명시하여, 이의 사용자들 간에 의사소통을 원활하게 해주며, 현재 테스트 문서들을 평가하기 위한 기준선을 제공하는 데 있다.

(나) 이 표준은 테스트 계획서, 테스트 명세서, 테스트 보고서로 구분하여 각각에 대하여 규정하고 있다.

2. 테스트 계획서

(1) 개요

테스트 계획서는 테스트 활동의 범위, 접근 방법, 자원, 일정 등을 규정하고, 테스트 항목, 테스트할 특성, 수행할 테스트 작업, 각 작업 담당자 및 계획서와 관련된 위험 요소를 식별하고 있다.

(2) 내용

테스트 계획서 식별자, 서론, 테스트 항목, 테스트 대상 특성, 테스트 대상이 아닌 특성, 접근 방법, 항목의 성공/실패 기준, 일시 중지 기준 및 재개 요구사항, 테스트 인도물, 테스트 작업, 환경 요건, 책임, 구성원 및 교육 요건, 일정, 위험 요소 및 비상 대처상황, 승인의 16개 항목으로 구성되어 있으며, 나열된 항목들을 순서대로 구성하도록 명시하고 있다. 그중 몇 가지 주요 항목의 내용은 다음과 같다.

(가) 서론

테스트할 소프트웨어 항목과 소프트웨어 특성을 요약하며, 프로젝트 승인서, 프로젝트 계획서, 품질보증 계획서, 형상관리 계획서, 관련 정책 및 관련 표준에 관한 문서들이 존재할 경우, 최상위 수준의 테스트 계획서에는 이들에 대한 참조를 포함하도록 하고 있다.

(나) 시험 항목

시험 항목의 버전/개정 수준을 포함하여 이들을 식별하도록 하며, 하드웨어 요구사항에 영향을 주는 전달 매체의 특성을 명시하거나, 시험을 시작하기 전의 논리적 또는 물리적 변환의 필요성을 표시하도록 하고 있다.

(다) 접근 방법

테스트에 대한 전반적인 접근 방법을 기술하고, 특성이나 특성 조합의 중요한 그룹 각각에 대하여, 이들 특성 그룹이 적절히 테스트된다는 것을 보증하는 접근 방법을 명시하도록 하고 있다. 또 특성의 지정한 그룹들을 테스트하는 데 사용할 도구, 기법 및 주요 활동을 명시하고 주요 테스트 작업을 식별할 수 있도록 하며, 이들 각각의 작업을 하는데 필요한 시간을 예측할 수 있도록 접근 방법을 충분하고 상세히 기술하도록 규정하고 있다.

(라) 테스트 인도물

인도할 문서를 식별하도록 하며, 테스트 계획서, 테스트 설계 명세서, 테스트 케이스 명세서, 테스트 절차 명세서, 테스트 항목 전달 보고서, 테스트 상황 기록, 테스트 사고 보고서 및 테스트 요약 보고서 문서를 포함하도록 규정하고 있다.

3. 테스트 명세서

(1) 테스트 설계 명세서

테스트 설계 명세서는 설계 명세서 식별자, 테스트 대상 특성, 세부 접근방법, 테스트 식별 및 특성의 성공/실패 기준 등 5개 항목으로 목차를 구성하고, 이 항목들을 순서대로 나열하도록 규정하고 있다.

(가) 테스트 대상 특성

테스트 항목을 식별하고, 이 설계 명세서의 대상인 특성과 그 조합을 기술한다.

(나) 세부 접근방법

테스트 계획서에 기술된 방법들을 상세히 명시하고 있다.

(다) 테스트 식별

테스트 설계와 관련된 테스트 케이스의 식별자를 열거하고, 이들 각각에 대하여 간략히 설명하도록 하고 있다.

(라) 특성의 성공/실패 기준

특성 또는 특성의 조합이 합격인지 불합격인지를 결정하는 데 사용할 기준을 명시하고 있다.

(2) 테스트 케이스 명세서

테스트 케이스 명세서는 식별한 테스트 케이스를 정의하는 문서로서, 테스트 케이스 명세서 식별자, 테스트 항목, 입력 명세서, 출력 명세서, 환경 요건, 특정한 절차 요구사항, 테스트 케이스 간 내부 의존성의 7개 항목으로 목차를 순서대로 구성하도록 규정하고 있다.

(가) 테스트 항목

테스트 케이스에 의해 테스트를 수행할 항목과 특성을 식별하고, 이를 간략하게 기술한다.

(나) 입력 명세서

테스트 케이스의 실행에 필요한 입력을 명시한다.

(다) 출력 명세서

테스트 항목에 요구되는 출력과 특성을 명시한다.

(라) 환경 요건

테스트 케이스를 실행하는 데 필요한 하드웨어, 소프트웨어 및 기타 필요한 시설 등을 명시한다.

(마) 특정한 절차 요구사항

테스트 절차에 대한 특정 제약 사항을 모두 기술하도록 한다.

(바) 테스트 케이스 간 내부 의존성

테스트 케이스 전에 반드시 실시되어야 하는 시험 케이스의 식별자를 나열하도록 규정하고 있다.

(3) 테스트 절차 명세서

테스트 절차 명세서는 테스트 케이스 실행을 위한 단계를 명시하고 있는 문서로서 테스트 절차 명세서 식별자, 목적, 특별 요구사항, 테스트 절차단계의 4개 항목 순서대로 구성하도록 규정하고 있다.

(가) 특별 요구사항

절차의 실행에 필요한 특정 요구사항을 모두 식별한다.

(나) 테스트 절차 단계

상황 기록, 준비, 시작, 진행, 측정, 중단, 재시작, 중지, 마감, 비상 대처 상황과 같은 10개의 단계를 포함하도록 규정하고 있다.

4. 테스트 보고서

(1) 테스트 항목 전달 보고서

테스트를 위해 전달할 테스트 항목들을 식별하는 문서로서, 각 항목에 대한 책임자, 그 위치 및 상태를 포함하여 현재 상태의 항목 요구사항과 설계에서 벗어난 사항들을 모두 이 보고서에 기록하도록 하고 있다.

(가) 목차 구성

테스트 항목 전달 보고서 식별자, 전달된 테스트 항목, 위치, 상태, 승인의 5개 항목으로 목차가 구성되어 있다.

(2) 테스트 상황 기록

테스트 상황 기록은 테스트 실행과 관련된 상세한 내용을 시간 순으로 기록하는 문서로서 테스트를 수행하는 중에 발생한 내용을 기록하기 위해 테스트 팀이 사용한다.

(가) 목차 구성

목차는 테스트 상황기록 식별자, 설명, 활동 및 사건(event) 기입의 3개 항목으로 구성되어 있다.

(나) 활동 및 사건 기입

각 사건에 대해 활동 시작과 종료를 포함하여, 발생 날짜 및 시간을 기록자의 이름, 직책 등과 함께 기록한다. 실행 설명, 절차의 결과, 환경 정보, 예외적 사건 및 사고 보고서 식별자를 고려하여 기록하도록 하고 있다.

(3) 테스트 사건 보고서

테스트 프로세스 중에 발생한 사건(Event)에서 조사 요구하는 사건을 모두 문서화한 것으로, 테스트 사고 보고 식별자, 요약, 사고 설명 및 영향과 같은 항목을 규정하고 있다.

(가) 사고 설명

사고에 대해 설명을 입력, 예상 결과, 실제 결과, 예외 사항, 날짜와 시간, 절차의 단계(Procedure Step), 환경, 반복 시도 횟수, 테스트 수행자, 참관자 항목을 포함하도록 하고, 사고 원인을 분리시켜, 정정하는 데 도움을 줄 수 있는 관련 활동과 관측 사항을 포함하도록 명시하고 있다.

(4) 테스트 요약 보고서

지정된 테스트 활동의 결과를 요약하고, 그 결과를 근거로 한 평가를 기록하는 문서이다.

(가) 목차 구성

목차를 테스트 요약 보고서 식별자, 요약, 변동 상황, 포괄성 심사, 결과 요약, 평가, 활동요약, 승인의 7개 항목으로 구성하여 각 항목을 세부적으로 규정하고 있다.

(나) 변동 상황

설계 명세서로부터 벗어난 테스트 항목의 변동 상황을 모두 기록한다.

(다) 활동 요약

주요 테스트 활동과 사건을 투입한 자원에 대한 데이터(예, 주요 테스트 활동 각각에 대하여 활용한 전체 요원들의 수준, 총 기계 사용시간, 총 소요시간 등)를 요약하여 기술하도록 규정하고 있다.

수행 내용 / 테스트 환경 구축 및 테스트 시나리오 작성하기

재료 · 자료

- 테스트 계획서, 설계서, 요구사항 명세서, 테스트 데이터

기기(장비 · 공구)

- 컴퓨터, 프린터

안전 · 유의사항

- 테스트 환경은 개발 환경 및 운영 환경과 분리된 별도의 환경을 구축하는 것이 바람직하다.
- 테스트 데이터는 해당 업무 도메인에 유사한 데이터라야 하며, 가장 좋은 데이터는 기존에 유사 서비스를 했던 시스템이 있을 때의 자료를 기반으로 테스트 데이터를 작성하는 게 좋다.

수행 순서

① 테스트 계획을 점검한다.

1. 테스트를 위한 일정 계획을 점검 및 보완한다.

(1) 테스트의 목적을 설정한다.

예를 들어 이번 테스트에서는 어떤 기능을 위주로 보겠다든지, 성능을 위주로 보겠다든지, 안정성 검증을 목적으로 하겠다든지와 같이 목적을 설정한다.

(2) 테스트를 수행할 시스템의 범위를 설정한다.

하나의 소프트웨어 시스템은 단 하나의 소프트웨어 컴포넌트로 구성되는 경우는 드물다, 비즈니스 로직, 사용자 인터페이스, 데이터 저장 로직 등으로 나눌 수도 있고, 인터넷 뱅킹, 타행 연동, 대출 시스템과 같이 각 업무 시스템으로 나누어서 범위를 설정할 수도 있다.

(3) 테스트를 수행할 시스템의 기능 범위를 설정한다.

테스트를 수행할 컴포넌트의 기능을 파악하고, 개발 계획에 따라 구현되어 있는 기능 리스트들을 테스트의 범위에 포함시킨다.

(4) 테스트를 수행할 방법을 결정한다.

단순하게 기능 테스트만 할 것인지, 성능 테스트 및 안정성 테스트 등을 할 것인지 테스트 타입을 결정한다.

(5) 테스트 대상 시스템에 대한 구조를 파악한다.

(가) 어떤 기능을 가지고 있으며, 어떤 컴포넌트들로 구성이 되어 있으며, 상호 연계가 어떻게 되어 있는지를 파악한다.

(나) 시스템의 구조를 파악하지 못하고 테스트를 진행할 경우에는 테스트의 성공 실패 여부만을 판단할 수 있고, 실패 시의 원인 파악이 어렵기 때문에 반쪽짜리 테스트가 될 수 있다.

(다) 또 구조 파악 없이는 결함의 발생 가능성이 높은 곳을 찾기가 어렵기 때문에 정교한 테스트가 어렵다.

(6) 테스트 대상 시스템 구조를 아키텍처 문서로 서술한다.

(가) 업무 컴포넌트 정의

시스템을 구성하기 위해서 어떤 업무 컴포넌트들이 구성되었는지를 다이어그램으로 서술한다.

(나) 소프트웨어 배포 구조

각 업무 시스템이 사용하는 소프트웨어 솔루션의 배포 구조를 서술한다. 예를 들어 데스크탑 가상화라면, 가상화 소프트웨어는 무엇을 사용하였고, 사용자 인증 정보는 어디에 저장하였는지와 같이 업무 컴포넌트가 실제 어떤 솔루션으로 구현되어 배포되었는지를 서술한다.

(다) 하드웨어 배포 구조

어떤 서버에 어떤 업무 컴포넌트가 배포되었는지 등을 서술한다. 특히 서버 뿐만 아니라, 서버 간을 연결하는 네트워크 구성과 스토리지(디스크 어레이 등)를 어떻게 구성하였는지를 서술해야 한다.

(7) 테스트 스케줄을 결정한다.

아주 상세한 테스트 케이스별 스케줄을 결정하는 것이 아니라, 테스트의 전체적인 절차에 필요한 일정을 결정한다.

2. 테스트를 위한 팀을 구성하고 각 테스트 시나리오별 담당자를 지정한다.

테스트를 기간 내에 수행할 인력들에 대한 조직을 구성하고, 이들에 대한 인건비와 제반 비용을 산정하여 테스트에 소요되는 예산을 산정한다.

3. 테스트 담당자는 테스트를 실행하기 전에 다음 사항을 숙지한다.

(1) 유스케이스 정의(Use Case Specification) 및 설계서 내용

유스케이스(Use Case)에 기술된 각 기능별 처리가 사용자의 요구 사항에 맞게 연결되어 처리되는지를 확인한다.

(2) 시스템 환경

시스템의 개발 환경과 분리되어 안정적인 테스트가 가능한지 확인한다.

(3) 화면 표준안

설계 시의 화면 구현 표준안에 맞게 구현되어 있는지 확인한다.

(4) 테스트를 위한 기초 조작 방법

테스트 케이스에 기록된 테스트 방법을 정상적으로 수행할 수 있도록 시스템의 조작 방법을 숙지한다.

(5) 테스트 결과 확인을 위한 명령어 및 시스템 로그 확인 방법

테스트의 결과가 화면 등 사용자가 확인할 수 있는 방법으로 표현되지 않을 경우 데이터의 변경 결과를 직접 스크립트 등으로 확인할 경우에 대비하여 명령어 및 시스템 로그 확인 방법을 숙지한다.

(6) 본인이 수행해야 할 테스트 케이스

테스트 담당자별 중복되거나 누락된 테스트 케이스는 없는지 확인한다.

(7) 테스트 결과서 작성 및 보고 절차

테스트가 완료된 이후 테스트 결과서를 작성하여 보고 및 승인할 담당자를 확인한다.

② 통합 테스트를 위한 테스트 환경을 구축한다.

1. 테스트를 효율적으로 수행하기 위해서 테스트를 위해 필요로 하는 하드웨어, 네트워크, 시스템 소프트웨어, 데이터베이스 시스템, 테스트 도구 등 테스트 환경을 사전에 구축한다.
2. 일반적으로 통합 테스트를 위한 환경은 개발 환경과 거의 유사하며, 시스템 기능을 테스트하는 데 특별히 필요한 장비가 없으면 개발 환경을 그대로 사용한다. 그러나 별도의 테스트를 위한 환경을 구성할 필요가 있는 경우 이를 별도로 구성한다.

③ 통합 테스트를 위한 테스트 시나리오를 작성한다.

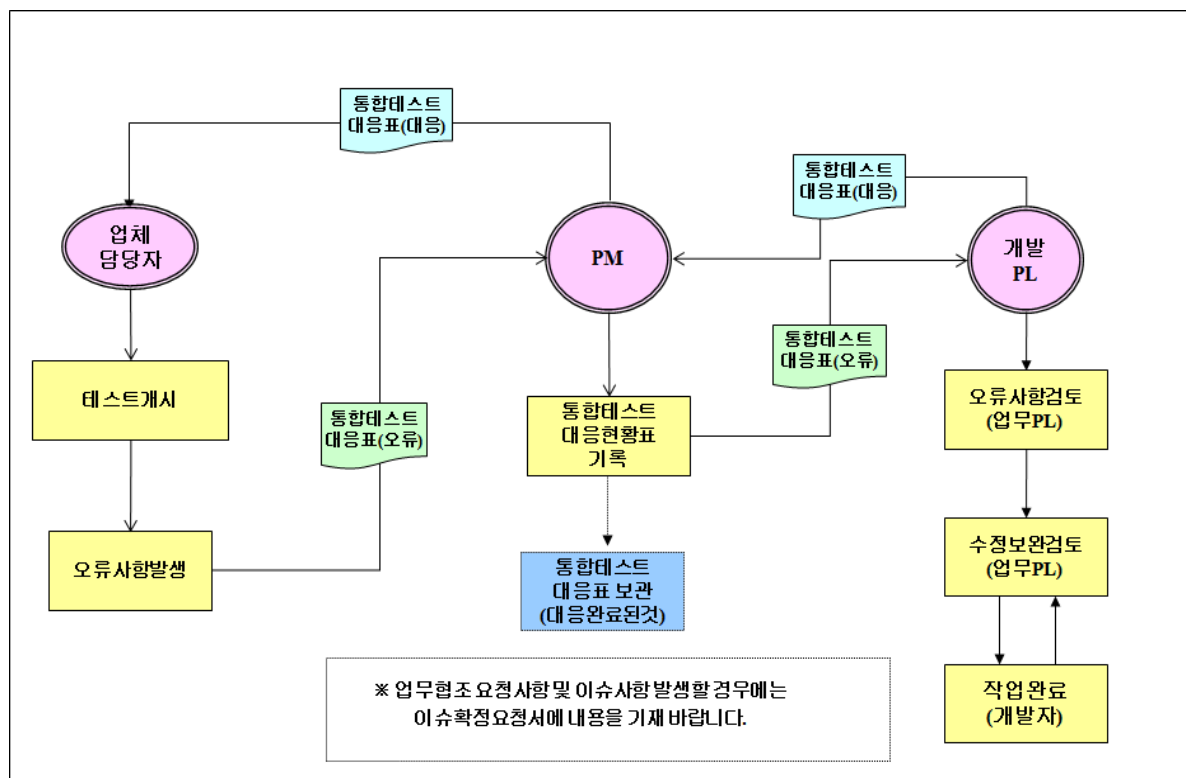
1. 업무 흐름에 따라 관련된 Use Case들이 서로 제대로 동작하는지 테스트하는 것이 가능하도록 작성한다.
2. 프로그램 간의 연계가 제대로 이루어지는지를 테스트하는 것이 가능하도록 작성한다.
3. 서브 시스템 간의 연계가 제대로 이루어지는지를 테스트하는 것이 가능하도록 작성한다.
4. 테스트 관련 산출물 실무 작성 사례

<표 1-2> 테스트시나리오 실무 작성 사례

| 테스트 시나리오 ID | 테스트 시나리오 | 테스트 시나리오 상세 설명 | 비 고 |
|----------------|-------------|--|-----|
| ITSC001 | DML 발주 | 주방별 DML 신청서 등록부터 DML PR, PR 생성 및 송신까지 일련의 DML 처리 연결을 확인한다. | |
| | | | |
| | | | |
| | | | |

<표 1-3> 테스트케이스 실무 작성 사례

| 테스트 시나리오 ID | 테스트 시나리오 | 테스트 케이스 ID | 테스트 케이스 설명 | 사전 조건 | 테스트 데이터 | 예상 결과 | 결과 확인 방법 | 실제 결과 | Pass / Fail |
|-------------------|-------------|------------------|------------------|--------------------------|-------------------|--------------------------|----------------|----------|-------------------|
| ITSC001 | DML 발주 | ITSC00 1-01 | DML PR 생성 | DML 신청서 등록 | 주방별 신청서 | PR 정보 생성 | PR 조회 | | |
| | | ITSC00 1-02 | 주방별 신청서 등록 | 주방별 사용 DML 코드등록 | 주방별 DML 신청서 | PR 생성 이후는 수정 불가 | 신 청 서 수정 | | |



[그림 1-3] 테스트 중 오류 발생 시 조치 절차도 작성 사례

④ 통합 테스트를 위한 별도의 테스트 데이터가 필요한 경우 데이터를 준비한다.

1. 테스트 데이터는 해당 업무 도메인에 유사한 데이터라야 한다.

가장 좋은 데이터는 기존에 유사 서비스를 했던 시스템이 있을 때의 자료를 기반으로 테스트 데이터를 작성하는 것이 좋다.

([그림 1-4] 테스트 데이터 사례, 참조)

2. 테스트용 데이터는 최대한 실 운영 환경과 유사하게 작성되어야 한다.

필요에 따라서는 테스트 데이터를 생성하는 툴을 개발해서 데이터를 생성할 수 있다.

09. 통합테스트 CASE(20091222시험) [호환 모드] - Excel

| | E | F | G |
|----|---|---|---|
| | OPERATION | 입력데이터 | 출력데이터 |
| 2 | 1. 월회원 로그인 처리후 "4700501219573220" | "4700501219573220" 카드 등록 | |
| 4 | 및 주민번호를 입력하여 멤버 회원 등록 | | |
| 6 | 1. POS 카드 리더기에 회원카드를 읽고 "4700501219573220" 번호가 POS화면에 나타나는 지 확인 | "4700501219573220" 카드 읽힘 | "4700501219573220" 화면표시 |
| 8 | 1. 4가지 메뉴 선택 | [Appetize to Share] - BLOOM 선택 (₩9,500) [Appetize to Share] - RANGE 선택 (₩11,900) [Premium Combo] - P-Filet 선택 (₩35,800) [Ribeye] - G-Ribeye 선택 (₩28,500) | 총 주문 금액 ₩85,700(부가세 미포함) 총 주문 금액 ₩94,270(부가세 포함) |
| 9 | 1. POS 카드 리더기에 회원카드를 읽고 "4700501219573220" 번호가 POS화면에 나타나는 지 확인 | "4700501219573220" 카드 읽힘 | "4700501219573220" 화면표시 |
| 19 | 1. POS 카드 리더기에 회원카드를 읽고 "4700501219573220" 번호가 POS화면에 나타나는 지 확인 | "4700501219573220" 카드 읽힘 | "4700501219573220" 화면표시 |
| 21 | 1. 3가지 메뉴 선택 | [Appetize to Share] - BLOOM 선택 (₩9,500) [Premium Combo] - P-Filet 선택 (₩35,800) [ADE] - KA 2잔 선택 (₩9,000) [ADE] - SA 선택 (₩4,500) | 총 주문 금액 ₩58,800(부가세 미포함) 총 주문 금액 ₩64,680(부가세 포함) |
| 22 | 1. POS 카드 리더기에 회원카드를 읽고 "4700501219573220" 번호가 POS화면에 나타나는 지 확인 | "4700501219573220" 카드 읽힘 | "4700501219573220" 화면표시 |
| 32 | 1. POS 카드 리더기에 회원카드를 읽고 "4700505718693209" 번호가 POS화면에 나타나는 지 확인 | "4700505718693209" 카드 읽힘 | "4700505718693209" 화면표시 |
| 34 | 1. 2가지 메뉴 선택 | [Premium Combo] - P-Filet 선택 (₩35,800) [Ribeye] - G-Ribeye 선택 (₩28,500) | 총 주문 금액 ₩64,300(부가세 미포함) 총 주문 금액 ₩70,730(부가세 포함) |
| 35 | 1. POS 카드 리더기에 회원카드를 읽고 "4700505718693209" 번호가 POS화면에 나타나는 지 확인 | "4700505718693209" 카드 읽힘 | "4700505718693209" 화면표시 |
| 41 | 1. POS 카드 리더기에 회원카드를 읽고 "4700505718693209" 번호가 POS화면에 나타나는 지 확인 | "4700505718693209" 카드 읽힘 | "4700505718693209" 화면표시 |
| 43 | 1. 2가지 메뉴 선택 | SET-OA 선택 (₩3,200) Crispy 선택 (₩10,700) | 총 주문 금액 ₩13,900(부가세 미포함) 총 주문 금액 ₩15,290(부가세 포함) |
| 44 | 1. POS 카드 리더기에 회원카드를 읽고 "4700505718693209" 번호가 POS화면에 나타나는 지 확인 | "4700505718693209" 카드 읽힘 | "4700505718693209" 화면표시 |
| | 2. 카드번호 "4700505718693209"에 해당하는 회원 생성, 보유 포인트 확인 | | 멤버십 조회 정보 - 회원명 : 황성진 - 보유포인트 : 43,847 |

[그림 1-4] 테스트 데이터 사례

※ MS로부터 이용 허락을 받았습니다.

수행 tip

- 테스트 환경은 개발 환경과 최대한 유사해야 하나 동일한 시스템으로 하는 것은 좋지 않다.

학습 1 교수 · 학습 방법

교수 방법

- 소프트웨어 테스팅의 개념, 테스트 계획서 및 테스트 관련 국제 표준 지식체계에 대한 이해 정도를 파악한 후 수업을 진행한다.
- 사전에 개인별 학습 자료를 준비하여 모든 학생이 참여할 수 있는 문제 해결식 수업, 협력 수업이 가능하도록 한다.
- 소프트웨어 개발 수명주기의 단계별 테스트 종류에 대한 이해 정도를 파악한 후 수업을 진행한다.
- 교수자의 주도로 테스트의 분류 및 기법, 소프트웨어 시험 문서화 표준 등에 대한 내용을 PPT 자료로 제시한 후 설명한다.
- 교수자의 주도로 경험 기반 테스트 설계 기법, 명세 기반 테스트 설계 기법, 구조 기반 테스트 설계 기법의 종류와 특징에 대한 내용을 PPT 자료로 제시한 후 설명한다.
- 테스트 케이스 및 테스트 시나리오의 설계 과정을 순서에 따라 단계적으로 실습이 이루어질 수 있도록 지도한다.
- 실습 시 테스트 수행계획서, 테스트 케이스 검토 체크리스트, 통합 테스트 케이스 관리대장 등의 작성 서식은 활용 서식을 참조하여 제공한다.

학습 방법

- 소프트웨어 테스팅에 대한 관련 용어를 숙지하고, 테스트 계획서, 테스트 관련 국제 표준 지식체에 대해 이해한다.
- 교수자가 제시하는 문제 해결 시나리오에 학습자 전원이 적극적으로 참여하여 주도적인 학습이 되도록 학습한다.
- 소프트웨어 테스팅의 수명주기 각 단계별 중점 사항을 체계적으로 학습한다.
- 테스트의 분류 및 기법, 소프트웨어 시험 문서화 표준에 대한 내용을 이해한다.
- 경험 기반 테스트 설계 기법, 명세 기반 테스트 설계 기법, 구조 기반 테스트 설계 기법의 종류와 특징에 대한 내용을 학습한다.
- 테스트 케이스, 테스트 시나리오 설계 과정에 대한 전 과정을 실습한다.
- 실습 시 테스트 수행 계획서, 테스트 케이스 검토 체크리스트, 통합 테스트 케이스 관리 대장 등의 서식은 활용 서식을 참조하여 작성한다.

학습 1 평가

평가 준거

- 평가자는 학습자가 수행 준거 및 평가 항목에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

| 학습 내용 | 평가 항목 | 성취수준 | | |
|-----------------|---|------|---|---|
| | | 상 | 중 | 하 |
| 개발자 테스트 케이스 작성 | - 개발하고자 하는 응용소프트웨어의 특성을 반영한 테스트 방식, 대상과 범위를 결정하여 테스트 케이스를 작성할 수 있다. | | | |
| 개발자 테스트 시나리오 작성 | - 개발하고자 하는 응용소프트웨어의 특성을 반영한 테스트 방식, 대상과 범위가 적용된 시나리오를 정의할 수 있다. | | | |
| | - 개발자 테스트 수행에 필요한 테스트 데이터, 테스트 시작 및 종료 조건 등을 준비할 수 있다. | | | |

평가 방법

- 문제해결 시나리오

| 학습 내용 | 평가 항목 | 성취수준 | | |
|-----------------|-----------------------|------|---|---|
| | | 상 | 중 | 하 |
| 개발자 테스트 케이스 작성 | - 소프트웨어 테스트 개념에 대한 이해 | | | |
| | - 통합 테스트 절차에 대한 이해 | | | |
| | - 테스트 케이스 작성 능력 | | | |
| 개발자 테스트 시나리오 작성 | - 테스트 환경 구축 방안에 대한 이해 | | | |
| | - 테스트 시나리오 작성 능력 | | | |

- 평가자 체크리스트

| 학습 내용 | 평가 항목 | 성취수준 | | |
|-----------------|-----------------------|------|---|---|
| | | 상 | 중 | 하 |
| 개발자 테스트 케이스 작성 | - 소프트웨어 테스트 개념에 대한 이해 | | | |
| | - 통합 테스트 절차에 대한 이해 | | | |
| 개발자 테스트 시나리오 작성 | - 테스트 환경 구축 방안에 대한 이해 | | | |

피드백

1. 문제해결 시나리오
 - 작성된 테스트 케이스 및 테스트 시나리오를 평가하여 주요 사항 및 개선해야 할 사항에 대해 표시하여 돌려준다.
2. 체크리스트를 통한 관찰
 - 소프트웨어 테스트 개념에 대한 이해도를 실습 과정에서 평가한 후에 개선해야 할 사항 등을 정리하여 돌려준다.

| | |
|-------------|--|
| 학습 1 | 개발자 테스트 케이스 설계하기(LM2001020207_14v2.1) |
| 학습 2 | 개발자 통합테스트하기 (LM2001020207_14v2.2) |
| 학습 3 | 개발자 결함조치하기(LM2001020207_14v2.3) |

2-1. 개발자 통합테스트 수행

학습 목표

- 개발자 통합테스트 계획에 따라 통합 모듈 및 인터페이스가 요구사항을 충족하는지에 대한 테스트를 수행할 수 있다.

필요 지식 /

① 정적 테스트(Static Testing) 기법

1. 정적 테스트의 개념

- (1) 동적 테스트(Dynamic Testing)이 실제 구현된 시스템을 실행하여 테스트하는 것에 반해, 정적 테스트는 실제 시스템이 구현되기 전에 요구사항 정의서, 설계서, 소스 코드 등의 개발 산출물을 테스트하는 것을 의미한다.
- (2) 정적 테스트는 개발 프로세스의 초기에 결함을 발견함으로써 전체 개발 수명주기의 효율을 높여 개발 비용을 낮추는 데 도움을 주는 테스트 활동이다.
- (3) 정적 기법은 리뷰와 같은 수동적 기법과 정적 분석 자동화 도구를 활용한 방법이 있다.

2. 리뷰(Review) 프로세스

(1) 리뷰의 개요

리뷰는 코드를 포함하여 소프트웨어 개발 및 테스트 산출물을 검토하고 테스트하는 방법이며, 동적 테스트를 실행하기 전에 적절하게 수행할 수 있다. 또 리뷰는 동적 테스트에서 발견하기 어려운 개발 산출물의 누락과 같은 결함을 발견할 수 있다. 즉, 정적 기법은 동적 테스트와는 달리 장애 자체보다는 장애의 원인(결함)을 발견한다.

(2) 리뷰를 통해 발견하기 쉬운 결함의 종류

표준 위반, 요구사항 결함, 개발 설계(Design) 결함, 불충분한 유지보수성, 부정확한 인터페이스 명세 등

(3) 리뷰의 유형

(가) 비공식적 리뷰(Informal Review)

공식적인 절차 없이 페어(Pair) 프로그래밍에 의한 리뷰이거나 기술 선임자가 설계와 코드를 리뷰하는 것일 수 있으며, 저렴한 방법으로 일정 수준의 성과를 달성할 수 있다.

(나) 기술적 리뷰(Technical Review)

동료 검토(Peer Review)라고도 하며, 동료와 기술 전문가가 참여하는, 결함 발견을 위한 문서화되고 정의된 프로세스가 존재하며, 기술적 문제 해결, 토론, 의사 결정, 대안 평가, 결함 발견, 명세서 또는 표준과의 적합성 검토 등을 목적으로 한다. 그 특징은 다음과 같다.

1) 다른 관점의 확보

소프트웨어를 사용해 본 적 없는 사람이나, 소프트웨어의 정상 동작 결과를 모르는 사람이 더 쉽게 결함을 발견한다.

2) 지식 전달

소프트웨어 산출물 및 결함 발견

3) 결함 조기 발견 및 제거

결함 수정 비용 감소

4) 재작업 및 테스트 노력 감소

전반적인 개발 노력 감소

(다) 워크스루(Walkthrough)

저자에 의한 진행 및 제어가 특징이고, 학습, 시스템에 대한 이해 향상, 결함 발견을 목적으로 하며, 그 특징은 다음과 같다.

1) 시스템의 형태나 프로그램 개발 사항에 대해 동료들로 하여금 조기에 오류를 확인할 수 있도록 하는 검토 회의

2) 주로 작성자의 요청에 의해 이루어지며, 완성된 결과물이 아닌 중간 산출물을 대상으로 함

3) 토론하고 결함을 찾아서 그에 대한 해결책을 서로 조언

4) 참가자의 역할이 명확하게 구분되지 않음

5) 산출물의 작성자 본인이 회의를 주재할 수 있음

6) 참가자들이 작업물을 숙지한 후 회의에 참석하지 않고, 개요 정도만 파악한 후 참가

7) 발견된 결함이 모두 수정되었는지 확인하는 작업은 생략될 수 있음

(라) 인스펙션(Inspection)

저자가 아닌 훈련된 중재자(Moderator)에 의한 진행 및 제어가 일반적이며, 결함 발

전을 목적으로 한다. 그 특징은 다음과 같다.

- 1) 동료 검토 방법 중 가장 공식적인 형태로 비슷한 수준의 직급이나 역할을 가진 사람들로 구성된다.
- 2) 코딩 전까지는 일반적인 개발보다 소요 인력 2배 이상 필요
- 3) 초기 프로젝트 비용이 증가하지만, 성공적인 인스펙션 결과 코딩 단계에서의 재작업과 투입 인력 감소
- 4) 결과적으로 품질 비용 감소
- 5) 개발 기간 단축

3. 도구에 의한 정적 분석(Static Analysis by Tools)

(1) 정적 분석의 개요

정적 분석은 조사 대상 소프트웨어를 실제로 실행하지 않는 상태에서 도구의 지원으로 수행하는 것이다.

(2) 정적 분석의 특징

- (가) 정적 분석은 동적 테스트로 찾기 힘든 결함을 발견함
- (나) 리뷰와 마찬가지로 정적 분석은 장애보다는 결함을 발견함
- (다) 도구의 도움을 받아 수행함
- (라) 정적 분석 도구는 프로그램 코드를 분석하는 것은 물론, HTML이나 XML과 같이 생성된 결과물도 분석함

(3) 정적 분석 도구를 통해 발견되는 전형적인 결함

- (가) 정의되지 않은 값으로 변수 참조
- (나) 모듈과 컴포넌트 간에 일관되지 않은 인터페이스
- (다) 사용되지 않는 변수
- (라) 사용되지 않는 코드(Dead Code)
- (마) 코딩 표준 위반
- (바) 보안 취약성
- (사) 코드와 소프트웨어 모델의 구문 규칙(Syntax) 위반

(4) 정적 분석 도구

정적 분석을 수행하는 오픈소스 툴로는 PMD, Find Bugs, Jdepends 등이 있으며, 정적 분석 툴을 적용하려면 반드시 패턴을 검토하고 개발하고자 하는 소프트웨어에 맞도록 패턴을 재정의해서 사용해야 한다.

② 통합테스트 수행 방법

1. 개요

- (1) 통합 테스트는 소프트웨어 아키텍처를 구축하는 동시에 인터페이스와 관련된 오류를 찾아내기 위한 테스트를 수행하는 체계적인 기법이다.
- (2) 목적은 단위테스트가 끝난 컴포넌트들을 가지고 설계에서 지시된 프로그램 구조를 만드는 것이다.
- (3) 수행 방법을 크게 나누면, 비점증적인 “빅뱅” 방식과 점증적인 방법으로 나눌 수 있는데, 빅뱅 방식은 모든 컴포넌트들을 사전에 결합하여 전체 프로그램을 한꺼번에 테스트하는 것을 말하며, 점증적인 방법은 아래와 같다.

2. 하향식 통합(Top Down)

(1) 방식

- (가) 메인 제어 모듈(메인 프로그램)부터 시작하여 제어 계층 구조를 따라 아래 방향으로 이동하면서 통합된다.
- (나) 메인 제어 모듈에 종속되는 모듈들과 최하위 모듈은 깊이-우선 또는 넓이-우선 중 한 가지 방식으로 구조에 통합된다.

(2) 수행 단계

- (가) 메인 제어 모듈은 테스트 드라이버로 사용되고, 스텝이 메인 제어 모듈의 바로 하위에 있는 모든 컴포넌트를 대신한다.
- (나) 선택된 통합 방식(깊이 또는 넓이 우선)에 따라, 하위 스텝이 한 번에 하나씩 실제 컴포넌트로 대체된다.
- (다) 각 컴포넌트가 통합될 때마다 테스트가 수행된다.
- (라) 각 테스트 세트가 완료되면 다른 스텝이 실제 컴포넌트로 대체된다.
- (마) 새로운 오류가 발생되지 않았음을 보증하기 위해 회귀 테스트가 수행될 수도 있다.

3. 상향식 통합(Bottom Up)

(1) 방식

- (가) 원자 모듈(Atomic Modules), 즉, 프로그램 구조의 최하위 레벨에 있는 컴포넌트부터 구축과 테스트를 시작한다.
- (나) 컴포넌트들이 아래에서 위로 통합되기 때문에, 해당 레벨의 하위 모듈에 의해 제공되는 기능을 항상 사용 가능하며 스텝의 필요성이 없어진다.

(2) 수행 단계

- (가) 하위 레벨 컴포넌트들이 특정 소프트웨어 서브 기능을 수행하는 클러스터로 결합된다.
- (나) 테스트 케이스의 입력과 출력을 관장하기 위해 드라이버(테스팅을 위한 제어 프로그램)가 작성된다.
- (다) 클러스터가 테스트된다.
- (라) 드라이버들이 제거되고 클러스터들은 프로그램 구조의 위쪽으로 이동하며 결합된다.

4. 회귀 테스트(Regression Testing)

(1) 내용

- (가) 통합 테스트의 과정에서 새로운 모듈이 추가될 때마다 소프트웨어는 변경되는데, 이러한 변경과 연관된 부작용이 이전에 결함 없이 동작했던 기능에 문제의 원인이 될 수도 있다.
- (나) 회귀 테스트는 변화로 인해 의도하지 않은 부작용이 전파되지 않았다는 것을 보증하기 위해 이미 수행된 테스트의 일부분을 다시 실행하는 것이다.

(2) 회귀 테스트 케이스 사례

- (가) 모든 소프트웨어 기능을 실행할 테스트의 대표적인 샘플
- (나) 변경에 의해 영향을 받을 가능성이 높은 소프트웨어 기능에 초점을 맞춘 추가적인 테스트
- (다) 변경이 일어난 소프트웨어 컴포넌트에 초점을 맞춘 테스트

5. 스모크 테스트(Smoke Testing)

(1) 내용

- (가) 제품 소프트웨어가 개발될 때 일반적으로 사용되는 통합테스팅 방법이다.
- (나) 시간이 중요한 프로젝트에서 시간을 조절하기 위한 메커니즘으로 설계되었으며, 소프트웨어 팀이 수시로 프로젝트를 평가할 수 있게 한다.

(2) 방법

- (가) 코드로 통합된 소프트웨어 컴포넌트는 빌드(Build)로 통합된다. 빌드는 하나 또는 그 이상의 제품 기능의 구현이 요구되는 모든 데이터 파일, 라이브러리, 재사용 가능한 모듈, 그리고 공학이 적용된 컴포넌트들을 포함한다.
- (나) 일련의 테스트는 빌드의 기능을 제대로 수행하지 못하게 하는 에러들을 찾아내기 위해 설계된다. 목적은 소프트웨어 프로젝트가 계획된 일정을 맞추지 못하게 할 가능성이 가장 큰 오류를 밝혀내는데 있다.

(다) 빌드는 다른 빌드와 통합되고, 전체 제품은 매일 스모크 테스트된다. 통합 방식은 하향식 또는 상향식이 될 수 있다.

(3) 장점

(가) 통합 위험의 최소화

스모크 테스트는 매일 수행되기 때문에, 비호환성과 다른 중대한 오류가 조기에 발견되고, 따라서 오류가 발견되었을 때 일정상의 심각한 영향을 줄일 수 있다.

(나) 최종 제품의 품질 향상

이 접근 방식은 구축(통합) 지향이기 때문에, 구조적인 오류와 컴포넌트 수준 설계 오류뿐만 아니라 기능 오류를 찾아낼 가능성이 높다. 이러한 오류들이 조기에 수정된다면 더 좋은 품질의 제품이 될 것이다.

(다) 오류 진단과 수정 간단

모든 통합 테스트의 접근 방식과 마찬가지로, 스모크 테스트 중에 발견되는 오류는 “소프트웨어의 새로운 증분”과 관련될 가능성이 크다. 즉, 빌드에 추가되는 소프트웨어는 새롭게 발견되는 오류의 원인일 가능성이 크다.

(라) 진행 상황 평가 용이

시간이 갈수록 더 많은 소프트웨어가 통합되고 동작하는 것을 입증하게 된다. 이것은 팀의 사기를 높여주고 관리자에게는 진행 상황이 진전되고 있다는 좋은 지표를 제공한다.

6. 통합 테스트 수행 방법 비교

대표적인 수행 방법의 내용 및 장단점을 비교하면 다음 <표 2-1>과 같다.

<표 2-1> 통합테스트 수행 방법 분류

| 구분 | 백본(Backbone) | 빅뱅(Big Bang) | 상향식(Bottom Up) | 하향식(Top Down) |
|---------|------------------------------|-----------------------|-------------------------------|-----------------------------|
| 수행 방법 | 가장 중요하고 리스크가 높은 모듈로 초기 통합 형성 | 모든 테스트 모듈을 동시에 통합 | 최하위 부분부터 점진적으로 모듈을 통합하며 테스트 | 최상위 모듈부터 통합해 가면서 테스트 |
| 드라이버/스텝 | 드라이버/스텝을 필요에 따라 만들어 사용 | 드라이버/스텝 없이 실제 모듈로 테스트 | 테스트 드라이버 필요 (점차 상부 모듈로 대체) | 테스트 스텝 필요 (점차 개발된 하부모듈로 대체) |
| 장점 | 결함 격리 쉬움 리스크가 높은 결함 초기 발견 | 단시간 테스트 | 결함 격리 쉬움 하위 모듈 테스트 | 결함 격리 쉬움 설계상 결함을 조기 발견 |
| 단점 | 테스트 시간이 과다 소요 | 결함 격리가 어려움 | 수정이 어려운 중요한 결함을 상부 구조에서 발견 가능 | 수정이 어려운 중요한 결함을 하부에서 발견 가능 |

③ 테스트 커버리지(Test Coverage)

1. 개요

- (1) 테스트 커버리지는 “테스트 대상의 전체 범위에서 테스트를 수행한 범위” 이다.
- (2) 즉, 테스트 대상을 얼마만큼 테스트했나를 정의하는 것으로, 테스트의 정확성을 판단하는 하나의 척도가 될 수 있다.

2. 기능 기반 커버리지

- (1) 테스트 커버리지에서 분수의 분모를 결정하는 것이 가장 중요한데, 그것이 ‘테스트의 범위를 무엇으로 측정할 것인가?’ 이다.
- (2) 가장 쉬운 방법은 테스트 대상 기능을 모수로 하는 방법이다.
- (3) UI가 많은 시스템의 경우 전체 화면 수를 모수로 사용할 수도 있다.
- (4) 기능 기반의 경우에는 테스트 커버리지를 100% 달성하는 것을 목표로 한다.

3. 라인 커버리지(Line Coverage)

- (1) 전체 소스코드 Line 수 대비 테스트 시나리오가 거쳐 가는 소스코드의 Line 수를 측정한 것을 ‘라인 커버리지’ 라고 하는데, 단위테스트(Unit Test)에서는 이 라인 커버리지를 척도로 삼는 경우가 많다.
- (2) 국내 SI 환경에서 적절한 라인 커버리지는 전체 시스템 중 40%를 난이도가 높거나 중요한 시스템을 기준으로 산정하여 이 시스템에 대해서 80%의 라인 커버리지를 달성하는 것을 목적으로 한다. 나머지 60% 시스템에 대해서는 60%의 라인 커버리지를 유지하는 것이 좋다.

4. 코드 커버리지(Code Coverage)

(1) 내용

- (가) 코드 커버리지는 소프트웨어의 테스트를 논할 때 얼마나 테스트가 충분한가를 나타내는 지표 중 하나다. 말 그대로 ‘코드가 얼마나 커버되었는가?’ 이다. 소프트웨어 테스트를 진행했을 때 코드 자체가 얼마나 실행되었냐는 것이다.
- (나) 코드의 구조를 이루는 것은 크게 구문(Statement), 조건(Condition), 결정(Decision) 이다. 이러한 구조를 얼마나 커버했느냐에 따라 코드 커버리지의 측정 기준은 나뉘게 된다.

(2) 구문(Statement) 커버리지

일반적으로 많이 사용되는 커버리지로, 실행 코드 라인이 한 번 이상 실행되면 충족된다.

(3) 조건(Condition) 커버리지

각 내부 조건이 참 혹은 거짓을 가지면 충족된다.

(4) 결정(Decision) 커버리지

각 분기의 내부 조건 자체가 아닌 이러한 조건으로 인해 전체 결과가 참 혹은 거짓이면 충족된다.

(5) MC/DC 커버리지

조건과 결정을 복합적으로 고려한다.

(6) 커버리지 측정 도구

커버리지를 측정하는 법은 사람이 로그를 찍어 가거나 디버거를 이용하여 볼 수는 있으나 매우 힘든 과정이다. 시중에는 많은 코드 커버리지 측정 도구가 나와 있으며, 도구의 특성 및 장단점 등을 감안하여 사용할 수 있다.

④ 테스트 자동화 도구

1. 배경

소프트웨어의 테스트는 소프트웨어 개발에 소요되는 총 노력의 40% 이상을 차지하기도 할 정도로 많은 자원이 투입되어 수행하는 프로세스이다. 따라서 테스트의 정확성을 유지하면서 테스트 시간을 줄일 수 있는 도구가 매우 중요하게 되었다.

2. 테스트 자동화

(1) 테스트 자동화의 개념

테스트 자동화란 사람에 의한 반복적 테스트 절차를 자동화 도구를 활용하여 스크립트 형태로 구현함으로써, 시간과 인력 투입의 부담을 최소화하면서 운영 중인 시스템의 모니터링 또는 UI가 없는 서비스의 경우에도 정밀한 테스트가 가능하도록 하는 것이다.

(2) 테스트 도구의 장점

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

(3) 테스트 도구의 단점

- (가) 도입후 프로세스 적용에 대한 시간, 비용, 노력에 대한 추가 투자
- (나) 도구의 사용에 필요한 노력과 시간에 대한 추가 투자

(다) 비공개 상용 소프트웨어의 경우 고가이며, 유지관리 비용이 높음

3. 자동 테스트 도구 유형

(1) 정적 분석 도구(Static Analysis Tools)

(가) 정적 프로그램 분석(Static Code Analysis)은 컴퓨터 소프트웨어를 분석하는 방법 가운데 하나로 그 소프트웨어로부터 만들어진 프로그램을 실제로 실행해 보지 않고 분석하는 방법이다.

(나) 대부분의 경우 원시 코드의 형태를 가지고 분석을 수행하지만 목적 코드의 형태를 가지고 분석하는 경우도 있다.

(다) 일반적으로 사람이 어느 정도 프로그램에 대한 이해를 바탕으로 자동화된 도구를 이용해서 분석하는 것을 정적 프로그램 분석이라고 부른다.

(2) 테스트 실행 도구(Test Execution Tools)

테스트 실행 도구는 테스트를 실행할 수 있도록 작성된 스크립트를 재생한다. 대규모의 테스트를 자동화하는 데는 적절치 않다. 저장된 스크립트는 각 스크립트마다 특정 데이터와 행위를 선형적으로 표현하고 있다. 따라서 이런 종류의 스크립트는 미처 예상하지 못한 이벤트에 취약할 수 있다.

(가) 데이터 주도 접근 방식

일반적으로 테스트 입력값(데이터)을 스프레드시트에 저장하고, 이 테스트 데이터를 읽을 수 있도록 한다. 이로써 다른 다양한 테스트 데이터로 동일 테스트를 수행할 수 있게 된다. 따라서 스크립트 언어에 익숙하지 않은 테스터라도 사전에 정의된 스크립트에 테스트 데이터를 입력하고 테스트를 수행할 수 있게 된다.

(나) 키워드 주도 접근 방식

스프레드시트에 수행할 동작을 나타내는 키워드(액션 워드)와 테스트 데이터가 포함되어 있다. 이 방식에서는 스크립트 언어에 익숙하지 않은 테스터라도 키워드를 이용하여 테스트를 정의할 수 있는데, 키워드는 테스트 대상 애플리케이션에 맞게 테일러링될 수 있다.

(3) 성능 테스트 도구(Performance Testing Tools)

성능 테스트 도구를 정상적으로 사용하기 위해서는 성능 테스트를 설계하고 테스트 결과를 해석할 수 있는 전문가가 필요하다.

(4) 테스트 관리 도구(Test Management Tools)

테스트 관리 도구는 조직의 요구에 맞는 최적화된 형태의 정보를 생성하기 위해 다른 도구나 스프레드시트 등과 연계하여 사용할 필요가 있다. 또 보고서는 효과적으로 설계되고 모니터링 되어 보고서의 가치를 전달할 수 있어야 한다.

(5) 테스트 장치(Test Harness)

- (가) 시스템 및 시스템 컴포넌트를 시험하는 환경의 일부분으로서 시험을 지원하는 목적 하에 생성된 코드와 데이터를 말한다.
- (나) 시험 드라이버(Test Driver)라고도 하며, 일반적으로 단위 시험이나 모듈 시험에 사용하기 위해 코드 개발자가 만든다.
- (다) 단순히 시험을 위한 사용자 인터페이스를 제공하거나 정교하게 제작된 경우, 코드가 변경되었을 때에도 항상 같은 결과를 제공하여 시험을 자동화시킬 수 있도록 디자인되어 있다.

(6) 테스트 활동에 따른 도구 분류

<표 2-2> 테스트 활동에 따른 자동화 도구

| 테스트 활동 | 테스트 도구 | 내 용 |
|--------------|--------------|---|
| 테스트 계획 | 요구사항 관리 | 고객 요구사항 정의 및 변경사항 관리 |
| 테스트 분석/설계 | 테스트케이스 생성 | 테스트 기법에 따른 테스트 데이터 및 케이스 작성 |
| | 커버리지 분석 | 대상시스템에 대한 테스트 완료 범위 척도 |
| | 테스트 자동화 | 기능 테스트 등 테스트 도구를 활용하여 자동화를 통한 테스트의 효율성을 높일 수 있음 |
| 테스트 수행 | 정적분석 | 코딩표준, 런타임 오류 등을 검증 |
| | 동적분석 | 대상시스템 시뮬레이션을 통한 오류 검출 |
| | 성능 테스트 | 가상사용자를 인위적으로 생성하여 시스템 처리능력 측정 |
| | 모니터링 | 시스템 자원(CPU, Memory 등) 상태 확인 및 분석 지원 도구 |
| 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 |

출처: 공개 SW 포털 (http://www.oss.kr/oss_repository6/69515)

재료 · 자료

- 설계서, 요구사항 명세서, 테스트 계획서, 테스트 데이터, CSTE(Certified Software Tester) CBOK (테스트 지식체계)

기기(장비 · 공구)

- 컴퓨터, 프린터

안전 · 유의사항

- 테스트 수행은 초기 데이터를 먼저 로딩한 다음에 테스트 케이스를 실행한 후 테스트 결과와 자원의 사용률을 기록하는 것이 중요하다.

수행 순서

① 테스트 계획을 점검한다.

작성된 통합테스트 계획서와 테스트 케이스 및 테스트 시나리오를 동료검토 등을 통하여 확인한다.

1. 통합테스트 계획서에는 대상 시스템의 범위 및 대상, 테스트 일정, 테스트 전략, 테스트 절차, 테스트 환경 및 테스트 결과에 대한 판정 기준, 완료 기준 등이 포함될 수 있다.
2. 테스트 전략에는 상향식(Bottom Up), 하향식(Top Down), 빅뱅(Big Bang) 방식 등이 있다.
3. 완료 기준의 예로는 “통합 대상물 간의 인터페이스는 최소 한 번 이상 테스트되어야 함” 등이 있을 수 있다.

② 통합테스트를 위한 테스트 환경을 구축한다.

통합 테스트 환경 구축은 다음 작업을 포함한다.

1. 통합 테스트를 위한 프로그램 작성
릴리즈된 시스템을 테스트 환경에 배포하는 것을 포함한다.

2. 통합 테스트를 위한 장비 및 도구 구축

부하 테스트 툴의 경우 컨트롤러 장비와 함께 부하 발생용 장비를 따로 요구하기 때문에 이에 대한 준비와 라이선스 설치 등을 병행해야 한다.

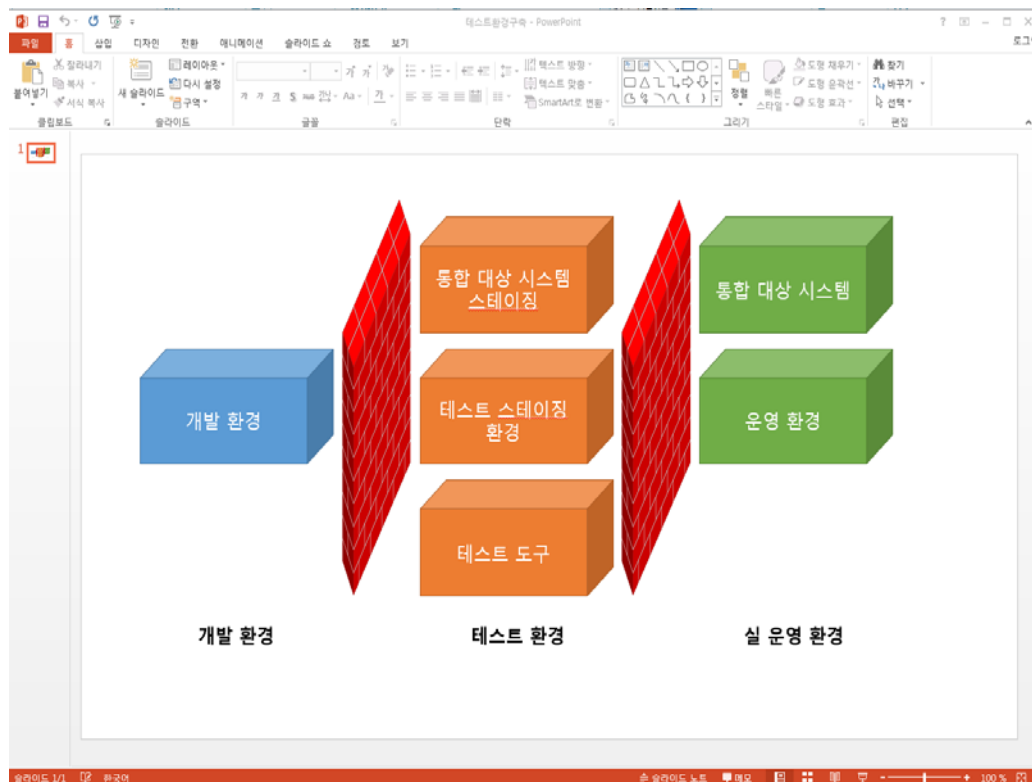
3. 통합 테스트에 필요한 담당자 교육

각 담당자별 테스트 케이스가 중복되거나 누락되지 않고 원활한 테스트가 이루어질 수 있도록 담당자 교육을 실시한다.

4. 통합 테스트를 위한 별도의 테스트 데이터가 필요한 경우 데이터 준비

테스트용 데이터는 실제 운영 환경과 유사하게 작성되어야 하며, 필요에 따라서는 테스트 데이터를 생성하는 툴을 개발해서 데이터를 생성해야 할 경우도 있다.

('[그림 2-1] 테스트 환경 구축' 참조)



[그림 2-1] 테스트 환경 구축

※ MS로부터 이용 허락을 받았습니다.

③ 각 기능별 시스템 간 연결 테스트를 수행한다.

1. 각 담당자별로 정해진 일정에 따라 테스트 케이스별 통합 테스트를 실행한다.

사전에 작성된 테스트 시나리오에 따라 테스트를 실행한다.

2. 테스트 케이스의 의존성을 고려하여 순차적으로 테스트 케이스별 통합테스트를 실행한다.

테스트 시나리오 작성 시 테스트 케이스의 의존성을 고려하여 순차적으로 실행될 수 있도록 작성하여야 한다.

3. 테스트 실행 시 기술적인 의문 사항은 테스트 지원 팀의 지원을 받는다.

테스트 종료 조건 등에 대한 문의에 대응할 수 있도록 테스트 지원 팀을 운영한다.

4. 테스트 결과를 통합테스트 케이스 결과서에 기록한다.

초기 데이터와 테스트 결과 및 자원의 사용률을 기록한다. 또 테스트 시 발견된 결함을 기록하고 재현 테스트를 할 수 있도록 결함의 발생 순서를 함께 기록한다.

④ 테스트 결과를 반영한다.

1. 테스트 수행 결과가 합격 기준에 부합되는지를 측정하여, 통과되지 못한 경우 오류를 수정하고 재테스트를 수행한다. 이 작업은 테스트가 성공적으로 끝날 때까지 반복된다.

2. 통합테스트 수행 보고서에 기록된 문제점에 대해 설계 문서, 프로그램, 사용자 문서에 변경이 있어야 한다면, 규정된 변경관리 절차에 따라 변경이 수행되어야 한다.

3. 프로그램 수정

(1) 통합테스트 수행자는 통합테스트 수행 중 파악된 프로그램에 대한 문제점을 기록한 통합테스트 결과서와 오류가 발생한 프로그램을 개발자에게 전달하여 오류를 바로잡을 수 있게 한다. 여러 클래스와 연관된 중대한 오류인 경우 설계 작업부터 다시 수행한다.

(2) 개발자는 프로그램의 오류를 수정한 후 그 프로그램에 단위테스트를 실시하고 테스트가 성공적이면 개발한 프로그램을 통합테스트 전문가에게 인도한다.

(3) 통합테스트 담당자는 수정된 프로그램이 제대로 동작하는지, 그리고 그 수정으로 인해 새로운 에러가 발생하는지를 확인하기 위해 관련된 테스트 케이스에 대해 테스트를 다시 수행한다.

⑤ 테스트 완료 결과를 보고한다.

1. 통합테스트의 결과가 통합테스트 계획서에 정해진 수준에 도달한 경우, 통합테스트 결과서를 작성하고, 동료 검토를 통하여 확인한다.

2. 통합테스트 결과서에는 테스트 항목에 대한 통과 여부, 완료 기준값과 충족 여부를 포함시키고, 결함 내용에 대한 분석을 통해 결함의 재발을 방지할 수 있도록 한다.

(활용 서식 “통합 테스트 결과서” 참조)

⑥ 사용자 매뉴얼을 작성한다.

사용자 매뉴얼을 작성하고, 동료 검토를 통하여 확인한다.

수행 tip

- 대부분의 SI 프로젝트에서 통합테스트 수행 방법은 수행 절차의 간편함과 효율적인 시간 사용을 위해서, Big Bang 방법을 사용한다.

2-2. 개발자 테스트 결과 분석

학습 목표

- 개발자 통합테스트 수행 결과로 발견된 결함에 대한 추이 분석을 통하여 잔존 결함을 추정할 수 있다.
- 개발자 통합테스트 결과에 대한 분석을 통해 테스트의 충분성 여부를 검증하고, 발견된 결함에 대한 개선 조치사항을 작성할 수 있다.

필요 지식 /

① 테스트 결과 분석

1. 소프트웨어 결함

일반적으로 소프트웨어의 결함을 지칭할 때 에러(Error), 결함(Defect), 결점(Fault), 버그(Bug), 그리고 실패(Failure)와 같은 용어들이 혼재되어 사용된다. 이러한 용어들의 차이를 정리해 보면 다음과 같다.

(1) 에러, 오류

결함(Defect)의 원인이 되는 것으로, 사람(소프트웨어 개발자, 분석가 등)에 의하여 생성된 실수가 대부분이다.

(2) 결함, 결점, 버그

에러가 원인이 되어 제품에 포함된 결함이다. 제거하지 않으면, 제품이 일으키게 되는 실패(Failure) 또는 문제(Problem)의 원인이 된다.

(3) 실패, 문제

제품의 결함이 있는 부분이 실행될 때 발생하는 현상을 말한다.

2. 테스트와 디버깅

(1) 테스트

(가) 시스템의 실행 중 실패가 발생하는 것은 프로그램 내에 있는 결함 때문이며, 그러한 결함들은 개발자의 에러에서 기인한다고 말할 수 있다. 테스트는 프로그램 내에 존재하는 결함을 제거하는 작업이다.

(나) 테스트는 제품을 인도하기 전에 가능한 한 많은 결함을 발견하여, 품질 높은 제품을 만들고자 하는 것이 목표이다. 테스트를 많이 하면 할수록, 더 품질이 높은 제품을 만들 가능성이 높아지는 것은 사실이지만, 시간, 비용 등의 제약 사항 때문에 적절한 양의 테스트를 실시할 수밖에 없다.

(다) 그러므로 어떻게 하면 프로그램의 특성과 프로젝트의 제약 조건에 맞는 효과적인 테스트를 실시하는가 하는 것이 중요한 이슈가 된다.

(2) 디버깅

(가) 개발자들이 테스트와 자주 혼동하는 용어로 디버깅(Debugging)이 있는데, 테스트가 발견의 작업이라면 디버깅은 수정의 작업이라고 말할 수 있다.

(나) 디버깅은 직접 소스코드를 수정해야 하기 때문에 시스템 내부를 아는 개발자에 의해 수행되지만, 테스트는 프로그램에 결함이 있다는 것을 가정하고 발견하는 작업이기 때문에, 프로그램을 직접 개발한 사람보다는 제3의 독립적인 사람에 의해 수행되는 것이 바람직하다.

3. 테스트 완료 조건

(1) 목적

테스트 완료 조건(Exit Criteria) 수립의 목적은 하나의 테스트 레벨 또는 특정 목적의 테스트가 언제 종료할지를 정의하는 것이다.

(2) 유의사항

테스팅은 많이 할수록 좋겠지만 프로젝트 일정, 비용, 조직의 성숙도 등에 따른 현실적인 제약이 있으므로, 적절한 규모의 테스트를 계획하여야 한다.

4. 테스트 리포팅

(1) 테스트 결과 정리

모든 테스트가 종료되었으면, 테스트 결과를 문서화하여 정리한다. 이 문서에는 테스트 계획, 설계 문서부터 개별 테스트 시나리오 및 테스트 결과가 모두 포함되어야 한다. 그래서, 다른 문서를 참조하지 않고 이 문서만 보더라도 테스트 대상 시스템을 이해하고, 테스트의 목적과 내용, 결과를 이해할 수 있도록 작성되어야 한다.

(2) 테스트 요약 문서

상위 레벨 매니저를 위해서 테스트 결과 요약 문서를 작성한다. 이 문서에는 테스트 계획과 함께 가장 중요한 소요 비용과, 테스트 결과로 인해서 판단할 수 있는 현재 대상 시스템에 대한 품질 상태를 포함해야 한다.

(3) 품질 상태

품질 상태는 테스트 커버리지, 테스트 성공률, 발생한 결함의 수와 중요도 등을 포함한다. 결함이나 테스트 내용 자체는 상위 레벨 매니저의 경우 그 내용을 이해 못하는 경우가 많기 때문에 결함의 수나 커버리지 등의 정량화된 품질 지표를 이용하여 현재 품질에 대한 이해를 도와야 한다.

(4) 테스트 결과서

개발 팀을 위한 테스트 결과서에는 결함에 대해서 중점적으로 기록하여야 하며, 결함의 내용 및 결함 발생 시의 상황 (CPU 등의 자원의 사용률과 로그 정보 등), 그리고 결함의 재현 순서를 자세히 기록하여 개발팀이 결함을 재현 및 수정할 수 있도록 한다.

(5) 테스트 프로세스 및 결과 평가

매번 테스트가 끝날 때마다 테스트를 수행한 프로세스를 리뷰하고 결과에 대한 평가를 수행해야 한다. 특히 Agile 방법론을 사용할 경우 테스트는 연속해서 이루어지기 때문에 매번 테스트가 끝날 때마다 프로세스를 최적화하여 다음 테스트에 더 높은 효과를 얻을 수 있도록 한다.

② 결함 추이 분석

1. TMMi(Test Maturity Model integration) Model

(1) TMMi 개요

(가) TMMi 프레임워크는 TMMi 파운데이션에서 개발되었다. TMMi는 테스트 프로세스 개선을 위한 가이드 라인과 레퍼런스 프레임워크로서 개발되었고, 또한 CMMI의 보조적인 모델로 위치하고 있다.

(나) 또 테스트 매니저들과 테스트 엔지니어들, 소프트웨어 품질 전문가들에게 중요한 이슈들을 전달하고 있다.

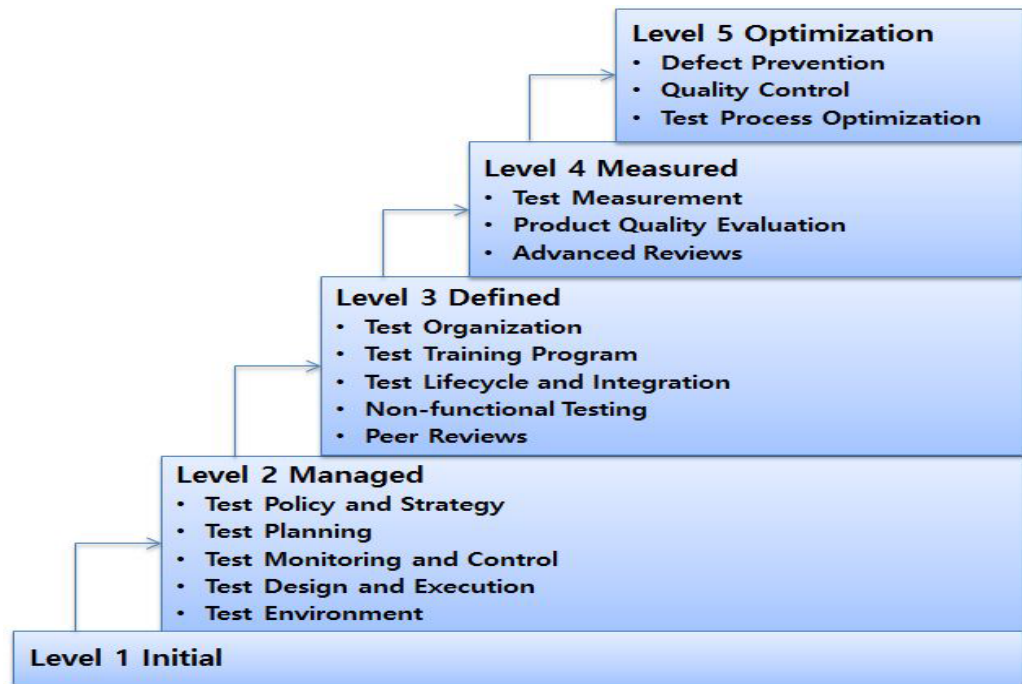
(다) 그러나 TMMi는 CMMI와 독립적으로도 사용될 수 있다. 실제적으로는 개발 프로세스를 개선하기 위해서 CMMI를 사용하는 대부분의 조직들에서 테스트 프로세스를 향상시키기 위해서 종종 TMMi를 사용한다.

(2) TMMi의 성숙도

(가) CMMI의 계층적 표현과 아주 비슷하게 TMMi도 성숙도 레벨 개념을 사용한다.([그림 2-1] 참조)

(나) 더욱이 프로세스 영역과 목적 그리고 실천 방법들도 동일하다.

(다) TMMi 성숙도 기준을 적용함으로써 테스트 프로세스를 개선시킬 수 있고, 제품 품질과 테스트 공학 생산성과 사이클 타임 노력에 긍정적인 영향을 끼칠 것이다.



출처: The TMMi Foundation, Assessment Certification, http://www.tmmi.org/?=assessment_cert

[그림 2-2] TMMi 성숙도 레벨과 프로세스 영역

(3) TMMi의 활용

- (가) TMMi는 테스트 프로세스를 평가하고 개선하려는 조직들을 지원하기 위해서 개발되어 왔다.
- (나) TMMi를 통하여 테스트 실행의 혼란과 부족한 자원, 도구, 전문 테스터들의 부족으로 야기되어 잘못 정의된 프로세스에서 체계적으로 잘 관리되어 성숙한 프로세스로 발전하게 된다.
- (다) 테스트 분야는 전문 분야로서, 개발 프로세스에 완전히 통합되어 한 부분으로 인식되고 있다.

(4) TMMi 레벨 4

(가) 레벨 단계

Measured

(나) 테스트 측정 (Test Measurement)

테스트 프로세스의 효과성과 효율성, 테스트 참여자의 생산성 및 제품 품질을 객관적으로 평가하는 측면에서 조직을 지원하는 데 필요한 측정을 식별, 수집하며 분석, 적용함. 또한 테스트 개선 결과 평가를 지원함

(다) 제품 품질 평가

제품 품질의 정량적 이해 확보를 토대로 해당 프로젝트의 제품 품질 목표 달성을 지원함

(라) 고급 동료 검토

제품 품질을 개발 수명주기 초기에 측정하고, 동료 검토 (정적 테스트)를 동적 테스트와 연결하여 테스트 전략과 접근법의 가치를 높임

(5) TMMi 레벨 5

(가) 레벨 단계

Optimizing

(나) 결함 예방

개발 주기 동안 발생하는 결함의 일반적인 원인을 식별하고 분석하여 미래에 발생할 수 있는 관련 결함을 예방하는 활동

(다) 품질 제어

테스트 프로세스와 각 프로젝트에서 수행된 테스트를 정량적으로 관리하고 제어하는 것으로, 테스트 프로세스 활동이 충분히 예측 가능하고 허용 한계 내에서 안정적으로 수행된다. 프로젝트 레벨에서 테스트는 제품의 품질을 예측하고 보다 효율적으로 하기 위해 대표 샘플을 통해 통계적인 방법으로 수행된다.

(라) 테스트 프로세스 최적화

현재 조직 내에서 수행되는 테스트 프로세스를 지속적으로 개선하며, 적절하고 새로운 테스트 기술(즉, 테스트 도구 혹은 테스트 방법)을 식별하고, 그 기술을 조직 내에 바르게 이행한다. 이러한 개선은 조직의 품질과 조직의 비즈니스 목적과 부합하는 프로세스 수행 목적을 지원한다.

2. 소프트웨어 신뢰도 및 예방 프로세스

(1) 소프트웨어 신뢰도

(가) 소프트웨어 신뢰도는 실제 운영 환경에서 소프트웨어가 일정 기간 동안 실패하지 않고 안정적으로 작동할 확률을 의미한다.

(나) 그래서 소프트웨어 신뢰도 측정은 시스템 인도 이후 결함을 측정함으로써 판단이 가능하다.

(다) 일반적으로 소프트웨어 신뢰도 측정의 대상은 시스템의 최종 인도 이후 소프트웨어 개발 생산성과 결함 발견의 정도로 판단할 수 있다.

(2) 결함 정보의 수집 및 분석

시스템의 최종 인도 이후 발생하는 결함 정보의 수집 및 분석은 현실적으로 매우 어려운 상황으로 다음과 같은 활동이 필요하다.

(가) 소프트웨어 신뢰도 예측

(나) 소프트웨어 개발 수명주기 결함 예측

- (다) 전체 프로젝트 수행 단계 중 요구사항 분석 단계에서 테스트 단계까지 개발 단계별 결함 예측 및 결과 분석
- (라) 프로젝트 수행 시 필연적으로 발생하는 결함 예측
- (마) 프로젝트 초기에 보다 정확히 예측하기 위한 모델

③ 테스트 결함 개선 조치사항 작성

1. SI 프로젝트에서 테스트 결함 및 결과 관리

테스트 결함 및 결과 관리는 테스트 진행 과정 상의 문제점을 개선하고, 계획했던 목표 품질을 달성했는지 판단하기 위한 중요한 기능이며, 다음과 같은 사항을 확인하여야 한다.

- (1) 개발자의 테스트 결과는 운영이 가능할 정도로 충분하고 깊이 있게 수행되었는가?
- (2) 테스트 수행 결과의 제3자 검증 과정 철저 필요(프로젝트 리더, TMO, 고객)
- (3) 결함의 조치와 확인 등 결함 관리 피드백 프로세스는 프로젝트 의사결정에 매우 중요한 역할을 차지함

2. 테스트 프로세스의 성과 지표 예시

(1) 관리 지표

프로세스 개선, 이슈/위험관리, 일정/자원 통제

(가) 테스트 진척률

- 1) 계획된 일정을 준수할 수 있는가?
- 2) Bottle Neck은 어디에 있는가?

(나) 결함 제거 효율성

- 1) 결함 제거가 얼마나 효율적으로 수행되는가?
- 2) 결함 제거에 투입되는 노력은 얼마나 되는가?

(다) 테스트 생산성

- 1) 테스트 수행 조직과 각 테스터의 테스트 수행 역량에 차이가 있는가?
- 2) 차이가 있다면 그 원인은 무엇인가?

(라) 테스트 커버리지

- 1) 요구기능에 대하여 테스트를 충분하게 수행하였는가?

(마) 검증 완료율

- 1) 계획된 모든 테스트 활동을 수행하였는가?

(바) 결함 유형 비율

- 1) 발견된 결함의 유형은 어떤 것들인가?
- 2) 발견된 결함의 원인은 어디에 있는가?

(2) 이행 지표

프로젝트 의사 결정, 다음 단계 공정 조정, 이슈/위험 관리, 일정/자원통 제

(가) 테스트 성공률

1) 테스트 결과, 성공적으로 기능이 동작하는 비율은 얼마나 되는가?

(나) 결함 조치율

1) 발견된 결함이 충분하게 조치되었는가?

2) 결함 조치가 지연되는 이유는 무엇인가?

(다) 테스트 커버리지

1) 요구기능에 대한 테스트를 충분하게 수행하였는가?

2) 중요 기능의 테스트를 누락하지는 않았는가?

(라) 테스트 진척률

1) 계획된 모든 테스트 활동을 수행하였는가?

수행 내용 / 개발자 테스트 결과 분석하기

재료 · 자료

- 테스트 계획서, 테스트 데이터, 요구사항 명세서, 결함 관리대장, TMMi(Test Maturity Model integration) Model

기기(장비 · 공구)

- 컴퓨터, 프린터

안전 · 유의사항

- 테스트 시 발견된 결함을 기록하고, 결함이 발생하였을 때는 가능하면 재현 테스트를 할 수 있도록 결함의 발생 순서를 함께 기록한다.

수행 순서

① 테스트 결과를 상세 분석한다.

테스트 수행 결과로 작성된 모든 테스트 결과 보고와 결함 관리 대장을 검토한다.

② 테스트 커버리지를 평가한다.

현실적으로 모든 기능을 100% 테스트할 수 없으므로, 어느 정도 테스트가 수행되었는지 파악하기 위해 커버리지를 측정한다.

③ 결함을 분석 및 평가한다.

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

- (1) 발견된 결함 평가는 소프트웨어 품질 및 신뢰성에 대한 정보를 제공한다.
- (2) 결함 분석 및 평가는 단순히 결함 수의 계산부터 통계 모델링 적용까지 다양한 방법을 적용하여 수행할 수 있다.

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

- (1) 결함 분포: 특정 속성에 해당되는 결함 수
- (2) 결함 추세: 시간 흐름에 따른 결함 수에 대한 추이
- (3) 결함 에이징: 특정 결함 상태에 머물러 있는 시간

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

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

④ 테스트 종료를 결정한다.

1. 테스트가 수행되고 테스트 종료 조건을 만족할 경우 테스트 활동을 종료한다.

2. 테스트 종료 조건의 지정 사례는 다음과 같다.

- (1) 코드 커버리지, 기능 커버리지, 리스크 커버리지와 같은 보장성 또는 충분함에 대한 측정치
테스트 전략 및 테스트 계획서에 명시된 커버리지 정도
- (2) 추정된 결함 밀도 또는 신뢰성 측정치
결함의 심각도를 고려한다.

(3) 잔존 리스크

수정되지 않은 결함, 또는 테스트 커버리지가 부족한 테스트 영역 등

(4) 테스트 비용과 예산

금융 기관 등 Mission Critical한 경우에는 테스트 비용과 관계없이 테스트 결과가 모두 성공인 것을 종료 목표로 설정한다.

(5) 시장 출시 시기에 따른 스케줄

SNS 서비스를 하는 벤처 기업 등의 경우 시장 출시 시간이 중요하기 때문에, 중 결함을 테스트 기간 중에 최대한 해결하되, 오픈 이후에 경결함을 해결하는 전략을 선택할 수도 있다.

(6) 수정되지 않은 결함 수

결함의 심각도를 고려한다.

(7) 시간당 결함 수(0에 근접할 때)

결함의 심각도를 고려한다.

(8) 재테스트(Re-Test) 횟수

회귀 테스트를 포함한 재테스트 횟수를 고려한다.

(9) 모든 테스트 케이스를 한번 이상 수행하고, 심각한 결함이 없거나 특정 수 이하 일 때

테스트 케이스가 리스크 기반 전략에 의해 설계된 경우, 기법 및 커버리지 내용을 포함한다.

(10) 예방된 피해와 소요되는 테스트 비용 비교

테스트 전략 및 업종에 따른 결함의 심각도와 영향도를 고려한다.

⑤ 테스트 결과보고서를 작성한다.

1. 테스트 결과 보고서는 테스트 대상 시스템의 품질과 테스트 상태에 대해 객관적인 평가를 내리고, 모든 이해관계자들과 의사소통하는 수단으로 사용하기 위하여 작성된다. (활용 서식 “통합 테스트 결과서” 참조)
2. 경우에 따라 여러 관계자를 위한 문서를 만들 수 있는데, 대상이 누구냐에 따라 양식과 보고의 내용이 달라질 수 있다.
3. 단, 테스트 결과는 다르게 왜곡해서는 안되며, 독자의 수준에 따라 결과 보고서에 전문 용어의 사용 여부, 표현 방법 등이 달라질 수 있다.

⑥ 테스트 결과보고서를 검토 및 승인한다.

작성된 테스트 결과보고서를 프로젝트 관리자 및 품질보증 담당자 등과 검토 및 협의를 수행한다.

수행 tip

- 테스트 종료 조건은 시스템의 목적과 성격, 그리고 프로젝트의 납기와 예산에 따라서 다른 수준으로 설정된다.

학습 2 교수 · 학습 방법

교수 방법

- 소프트웨어 테스팅의 개념, 테스트 계획서 및 테스트 관련 국제 표준 지식체계에 대한 이해 정도를 파악한 후 수업을 진행한다.
- 사전에 개인별 학습 자료를 준비하여 모든 학생이 참여할 수 있는 문제 해결식 수업, 협력 수업이 가능하도록 한다.
- 소프트웨어 개발 수명주기의 단계별 테스트 종류에 대한 이해 정도를 파악한 후 수업을 진행한다.
- 교수자의 주도로 TMMi의 단계별 성숙도에 대한 내용을 PPT 자료로 제시한 후 설명한다.
- 교수자의 주도로 통합테스트 수행방법별 특징과 활용에 대한 내용을 PPT 자료로 제시한 후 설명한다.
- 테스트 커버리지 및 테스트 자동화 도구의 종류 및 적용방법에 대하여 설명한다.
- 통합테스트 수행 및 테스트 결과 분석 과정을 순서에 따라 단계적으로 실습이 이루어질 수 있도록 지도한다.

학습 방법

- 소프트웨어 테스팅에 대한 관련 용어를 숙지하고, 테스트 계획서, 테스트 관련 국제 표준 지식체에 대해 이해한다.
- 교수자가 제시하는 문제 해결 시나리오에 학습자 전원이 적극적으로 참여하여 주도적인 학습이 되도록 학습한다.
- 소프트웨어 테스팅의 수명주기 각 단계별 중점 사항을 체계적으로 학습한다.
- TMMi의 단계별 성숙도에 대한 내용을 이해한다.
- 통합테스트 수행방법별 특징과 활용에 대한 내용을 이해한다.
- 테스트 커버리지 및 테스트 자동화 도구의 종류 및 적용방법에 대하여 이해한다.
- 통합테스트 수행 및 테스트 결과보고서 작성에 대한 전 과정을 실습한다.

학습 2 평가

평가 준거

- 평가자는 학습자가 수행 준거 및 평가 항목에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

| 학습 내용 | 평가 항목 | 성취수준 | | |
|---------------|--|------|---|---|
| | | 상 | 중 | 하 |
| 개발자 통합테스트 수행 | - 개발자 통합테스트 계획에 따라 통합 모듈 및 인터페이스가 요구사항을 충족하는지에 대한 테스트를 수행할 수 있다. | | | |
| 개발자 테스트 결과 분석 | - 개발자 통합테스트 수행 결과 발견된 결함에 대한 추이 분석을 통하여 잔존 결함을 추정할 수 있다. | | | |
| | - 개발자 통합테스트 결과에 대한 분석을 통해 테스트의 충분성 여부를 검증하고, 발견된 결함에 대한 개선 조치사항을 작성할 수 있다. | | | |

평가 방법

- 문제해결 시나리오

| 학습 내용 | 평가 항목 | 성취수준 | | |
|---------------|--------------------|------|---|---|
| | | 상 | 중 | 하 |
| 개발자 통합테스트 수행 | - 통합테스트 절차에 대한 이해 | | | |
| | - 테스트 계획서 작성 능력 | | | |
| 개발자 테스트 결과 분석 | - 통합테스트 결과 분석 능력 | | | |
| | - 결함 개선 조치사항 작성 능력 | | | |
| | - 결함 추이 분석 능력 | | | |

• 평가자 체크리스트

| 학습 내용 | 평가 항목 | 성취수준 | | |
|---------------|-------------------|------|---|---|
| | | 상 | 중 | 하 |
| 개발자 통합테스트 수행 | - 통합테스트 절차에 대한 이해 | | | |
| 개발자 테스트 결과 분석 | - 통합테스트 결과 분석 능력 | | | |
| | - 결함 추이 분석 능력 | | | |

피드백

- 문제해결 시나리오
 - 작성된 테스트 계획서 및 테스트 결과보고서를 평가하여 주요 사항 및 개선해야 할 사항에 대하여 표시하여 돌려준다.
- 체크리스트를 통한 관찰
 - 통합테스트 절차에 대한 이해도 및 통합테스트 결과 분석 능력, 결함 추이 분석 능력을 실습 과정에서 평가한 후에 개선해야 할 사항 등을 정리하여 돌려준다.

| | |
|-------------|--|
| 학습 1 | 개발자 테스트 케이스 설계하기(LM2001020207_14v2.1) |
| 학습 2 | 개발자 통합테스트하기(LM2001020207_14v2.2) |
| 학습 3 | 개발자 결함조치하기 (LM2001020207_14v2.3) |

3-1. 개발자 테스트 결함 관리

학습 목표

- 개발자 테스트 수행 결과에서 발견된 결함을 식별하고 조치에 대한 우선순위를 결정하고 적용할 수 있다.
- 결함이 발생한 소스를 분석하여 추가적인 코딩으로 결함을 제거하고, 결함조치시 기존에 구현된 로직과의 연관성을 고려하여 부작용을 최소화할 수 있다.
- 개발자 통합 테스트 결과 결함 조치로 변경되는 소스의 버전을 관리하고 결함 조치 결과에 대한 이력을 관리할 수 있다.

필요 지식 /

① 테스트 결함 식별

1. 결함의 분류

(1) 결함 유형 분류

(가) 기획시 유입되는 결함

요구사항의 표준 미준수, 테스트 불가능, 불명확, 불완전, 불일치, 기타 결함

(나) 설계시 유입되는 결함

설계 표준 미준수, 테스트 불가능, 불명확, 불완전, 불일치, 기타 결함

(다) 코딩시 유입되는 결함

코딩 표준 미준수, 불완전, 불일치, 인터페이스 결함, 데이터 결함, 기타 결함

(라) 테스트 부족으로 유입되는 결함

마무리 부족, 팀간 의사소통 부족, 코딩 실수

(2) 결함 심각도별 분류

(가) 결함 심각도 분류 사례

치명적, 매우 심각, 심각, 보통, 경미

치명적 결함, 주요 결함, 일반 결함, 사소한 결함, 개선 사항

(나) 부적절한 결함 심각도 분류 사례

Major, Minor, Trifle

A, B, C

(다) 결함 심각도 관리

프로젝트 및 조직 차원에서 합의된, 각 단계를 구분하기 위한 구체적이고 명확한 기준과 용어를 사용해야 결함 리포팅의 신뢰성 저하, 불필요한 논쟁 유발, 시간 낭비 등 낭비되는 요소를 막을 수 있다.

(2) 결함 우선순위

(가) 결함 심각도와 결함 우선순위

결함 심각도와 결함 우선순위는 구분되어야 한다. 모든 경우 결함 심각도가 가장 높다고 해서 가장 먼저 그 결함을 수정해야 하는 것은 아니다.

(나) 결함 우선순위 표현 사례

즉시 해결, 주의 요망, 대기, 낮은 순위

2. 결함 리포트

결함 관리를 위해 결함 발생 시 결함 관리시 스템에 결함을 등록하되 아래 항목들을 필수로 한다.

(1) 결함 내용

(2) 테스트 케이스 식별 번호

(3) 결함 유형

(4) 발견일

(5) 심각도

(6) 우선순위(결함 수정의 우선순위)

(7) 시정 조치 예정일

(8) 수정 담당자

(9) 재 테스트 결과

(10) 종료일

3. 결함 관리 측정 지표

결함 관리 시 다음 측정 지표를 추적해야 한다.

(1) 심각도 범주당 미해결 결함 수

(2) 심각도 범주당 기간 내 수정된 결함 수

- (3) 발견된 결함 전체수
- (4) 리스크 레벨별 잔존 결함의 심각도 수준 및 결함 수(리스크가 높은 부분 중심)
- (5) 누적 결함 수

② 결함 제거 및 결함 조치 이력관리

1. 결함 분석(Defect Analysis) 관련 용어 설명

(1) 결함 분포(Defect Density)

정해진 운영 기간 중에 발견된 결함을 소프트웨어 크기로 나눈 수치이다.

(2) 결함 발견 효율성(Defect Detection Efficiency)

특정 단계에서 삽입된 결함 중에서 발견된 결함의 수치이다. 해당 단계에서 삽입된 결함 대비 퍼센트로 표시한다.

(3) 인도 전 결함 제거 효율성(Pre-Delivery Defect Removal Efficiency)

고객에게 소프트웨어를 인도하기 전에 제거되는 결함의 수치이다. 현재까지 발견된 결함들의 총계 대비 퍼센트로 표시한다.

(4) 결함 해결 효율성(Defect Correction Efficiency)

새로운 결함을 발생시키지 않고 해결된 결함의 수치이다. 해결된 결함의 총계 대비 퍼센트로 표시한다.

(5) 결함 가능성(Defect Potential)

소프트웨어의 운영 중에 발생할 수 있는 결함 유형별 가능 수치이다.

(6) 평균 수리 시간(Mean Time To Repair)

해결된 모든 결함 집합에 대해서 결함 보고 후 결함 해결까지의 경과 시간에 대한 산술 평균이다.

(7) 평균 결함 발생 시간(Mean Time Between Failures)

발견된 모든 결함에 대해서 현재까지의 총 운영 시간을 총 결함 수로 나눈 산술 평균이다.

(8) 대표적인 결함(Outstanding Defects)

발생한 모든 결함에 대해서, 월별로 가장 많이 발생한 결함을 표시하는 결함 유형이다.

(9) 평균 수리 비용(Mean Cost To Repair)

해결된 모든 결함에 대해서, 해결 작업에 투입된 평균 비용이다.

(10) 상대적인 재작업 비용(Relative Cost of Rework)

소프트웨어 총 개발비용 대비 총 재작업 비용의 퍼센트로 표시된다.

2. 형상 관리(Configuration Management)

(1) 목적

프로젝트나 제품의 전체 수명주기에 걸쳐 시스템이나 소프트웨어(컴포넌트, 데이터, 문서)의 상태를 그대로 보전, 보호하고 유지하기 위함이다. 정리하면 다음과 같다.

(가) 소프트웨어의 형상을 집합적으로 정의하는 항목을 모두 확인하는 일

(나) 이 항목들에 대한 변경을 관리하는 일

(다) 응용 소프트웨어의 다른 버전 개발을 촉진하는 일

(라) 시간이 흐름에 따라 형상이 진화할 때 소프트웨어의 품질을 유지하는 일

(2) 형상 관리 시스템의 요소

(가) 컴포넌트 요소

각 소프트웨어의 형상 항목을 접근하고 관리할 수 있게 하는 파일 관리 시스템(데이터베이스 등)과 결부된 일련의 툴

(나) 프로세스 요소

컴퓨터 소프트웨어를 관리하고 생성하며 사용하는 모든 이해관계자들을 위한 변경 관리 및 관계된 행위들의 효과적인 방법을 정의하는 일련의 절차 및 작업

(다) 구성 요소

확인된 일련의 컴포넌트들만으로 조립되었음을 확인함으로써 소프트웨어의 구축을 자동화하는 일련의 툴

(라) 인적 요소

효과적인 소프트웨어 형상관리의 구현을 위한 소프트웨어 팀에 의해 사용되는 일련의 툴 및 프로세스의 특징

(3) 소프트웨어 형상 관리의 특성

(가) 버전 관리

프로젝트가 진행되면서 작성되는 여러 가지 버전의 작업 산출물을 효과적으로 관리할 수 있게 하고, 테스트와 디버깅을 수행하는 동안 개발자들이 이전의 버전으로 돌아갈 수 있도록 모든 버전을 저장할 수 있어야 한다.

(나) 의존성 추적 및 변경 관리

형상관리 저장소는 저장되어 있는 데이터 간의 매우 다양한 관계를 관리하고 추적할 수 있어야 한다. 다양한 관계란 기업 전체의 실체와 프로세스들 간의 관계, 응용 소프트웨어의 설계 각 부분들 간의 관계, 설계 컴포넌트와 기업 전체의 정보구조, 설계 요소와 납품물 간의 관계 등을 말한다.

(다) 요구사항 추적

요구분석 명세서에 명시된 내용에 대한 작업의 결과인 설계와 구현, 납품물 모두를 추적할 수 있는 기능을 제공한다. 더불어 어떤 요구사항이 제품의 어느 부분을 만들게 했는지도 추적할 수 있게 한다.

(라) 형상 관리

특정한 프로젝트의 이정표나 납품물을 표현하는 일련의 형상을 추적한다.

(마) 감사 행적

- 1) 감사 행적은 언제, 누가, 무슨 이유로 변경을 가했는지에 대한 추가적인 정보를 확립한다.
- 2) 변경의 출처에 대한 정보는 저장소에 특정한 객체의 속성으로 저장된다.
- 3) 저장소의 동작 개시 장치는 개발자와 설계 요소가 변경될 때마다 변경 사유 등의 감사 정보를 추가하는 데 사용되는 틀을 지원한다.

(4) 테스트 측면에서의 형상 관리

(가) 테스트웨어(Testware) 식별, 버전 관리, 변경 추적, 상호 연관성, 개발 아이템(테스트 대상)과의 연계 관리 등을 통해 테스트 프로세스 전반에 걸친 추적성 확보

(나) 테스트 문서에서 참조하고 있는 모든 관련 문서 또는 소프트웨어 아이템이 모호하지 않게 관리됨.

(5) 활용

(가) 테스트는 형상 관리를 통해 테스트된 품목, 테스트 문서, 테스트 케이스와 테스트 하네스 등을 혼선 없이 관리하고 효율적으로 재사용할 수 있다.

(나) 형상 관리 절차와 인프라는 테스트 계획 단계에서 결정되고 문서화되어, 구현되어야 한다.

③ 테스트 마감 활동

1. 개요

완료된 테스트 활동에서 데이터를 수집, 테스트에서 발견된 사실, 수치적 데이터와 함께 테스트 경험, 테스트웨어를 종합하고 축적하는 활동이다. 테스트가 체계적으로 수행되었는지를 평가하고, 향후 테스트를 개선하기 위해 모범 사례를 모델로 테스트 프로세스를 심사(평가)한다.

2. 테스트 마감활동의 발생 시기

- (1) 소프트웨어 시스템이 출시되어 테스트 프로젝트가 완료되었을 때
- (2) 테스트 프로젝트가 취소되었을 때

(3) 계획된 모든 마일스톤이 달성되었을 때

(4) 유지보수 활동 중 추가 개발되거나 업데이트된 부분이 출시 완료되었을 때

3. 테스트 마감 활동의 주요 포함 사항

(1) 테스트 결과 마감

(가) 예정된 산출물 확인

(나) 인시던트 리포트(결함 리포트 포함) 종료

(다) 해결되지 않은 추가 및 변경 요구사항에 대한 처리

(라) 시스템을 인수하는 것을 문서화

(2) 테스트웨어, 테스트 환경, 테스트 기반 설비를 차후에 사용할 것에 대비하여 마감하고 보관

(3) 테스트웨어를 유지보수 조직에 이관

(4) 테스트 프로세스 심사(평가) 및 개선사항 제안

(5) 이후 릴리즈나 프로젝트, 테스트 성숙도의 개선에 지침이 될 수 있도록 테스트 프로젝트를 통해 얻은 교훈을 분석

수행 내용 / 개발자 테스트 결함 관리하기

재료 · 자료

- 테스트 계획서, 테스트 데이터, 테스트 결함 관리대장

기기(장비 · 공구)

- 컴퓨터, 프린터

안전 · 유의사항

- 테스트 결함의 관리는 요구분석 단계에서부터 추적 관리가 가능하도록 되어야 한다.

수행 순서

① 결함 관리 활동을 수행한다.

1. 결함 관리

동료 검토, 합동 검토, QA 리뷰, 결함 조치, 확인 등의 과정을 반복 수행한다.

2. 결함 발견

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

② 결함 관리 대장을 등록한다.

1. 결함 등록

주간 단위로 QA가 1차 검토하고, 발견된 결함은 결함 관리 대장에 등록한다.

2. 결함 분석

등록된 결함을 분석하여 단순 에러인지 실제 결함인지 분석한다.

3. 결함 확정

등록된 내용이 결함으로 확정될 경우, 결함을 결함 확정 상태로 설정한다.

4. 형상 관리

PL이 전체적인 내용의 흐름을 잘 숙지, 개발자에게 산출물 작성을 전달하고, 이를 취합 정리하여 완성된 산출물은 형상 관리 서버에 등록한다.

5. 결함 관리 및 형상 관리 산출물 작성 사례

(1) 통합테스트 대응표 작성 사례

| 통합 테스트 대응표 | | | | | 프로젝트명 | | 시스템명 | | 서비스시스템명 | |
|------------|--------------|--|------|------|---------|---------|----------------------|--------------------|--------------------|------------------|
| | | | | | FSP | | CRM관리 | | 기준정보 | |
| 관리번호 | | | | | 프로그램 ID | GCOD001 | | 사양변경 | 유 / 무 | |
| 프로그램명 | 행사 조회 | | | | 수신 | 담당자 | | 첨부자료 | 유 / 무 | |
| 발신 | 팀명 | | 발생일 | | 수정착수일 | | | 수정완료일 | | |
| | 담당자 | | 발생시각 | | | | | | | |
| 첨부자료 | 오류발생 화면 DUMP | | | 대응기간 | 긴급, 통상 | 원인분류 | 1. 프로그램문제 5. 사양변경 | 2. 기능보완 6. 환경문제 | 3. 사양누락 7. 이행문제 | 4. 조작미스 8. 기타 |
| 오류 내용 | | | | | 대응내용 | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |

[그림 3-1] 통합테스트 대응표 작성 사례

출처: 자체 제작

(2) 형상 관리 산출물 작성 사례

| 형상변경요청서 | | | |
|---------|--|--------|------------|
| 프로젝트명 | XXXXXX <u>신정보시스템</u> 구축 | | |
| 변경요청자 | 사업관리팀 천수홍 | 요청일자 | www. 8. 25 |
| 항목 ID | PN_10(사업수행계획)_01 | 항목명 | 사업수행계획서 |
| 처리담당자 | | 완료예정일자 | |
| 승인자 | 홍길동 (인) | 처리결과확인 | 이순신 (인) |
| 요청내용 | 1. 위험관리방안 보완 * 예상되는 위험 및 이의 대응방안 수립에 관한 내용 추가 * 발생한 위험의 평가 및 처리절차 구체화 2. 사업의 범위 구체화 * 과업지시서 및 제안서의 내용과 상이한 부분을 정리하여 보완 | | |
| 변경사유 | 1 차 <u>감리지적사항</u> 에 대한 보완 | | |
| 처리결과 | 변경요청사항의 내용이 <u>이상없이</u> 보완됨. | | |

[그림 3-2] 형상변경요청서 작성 사례

출처: 자체 제작

<표 3-1> 형상 관리 대장 실무 작성 사례

| 대분류 | 중분류 | 소분류 | 세분류 | 세부자료 | 버전 | 변경사유 | 담당자 |
|-----|-----|-----|-----|--------|-----|------|-----|
| 개발 | 모델 | 업무1 | 세무A | 업무분석서 | 1.0 | 중간감리 | |
| | | 업무2 | | 아키텍처정의 | 1.1 | 설계모델 | |
| | 분석 | 업무3 | | 메뉴구성도 | 1.5 | 범위변경 | |

③ 결함에 대한 시정 조치를 수행한다.

1. 결함 할당

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

2. 결함 조치

각 PL들은 결함 관리 대장에 대한 주기적 모니터링과 결함에 대한 시정 조치 즉 수정활동을 수행한다.

4 결함 조치 이력을 관리한다.

1. 결함 조치 검토

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

2. 결함 조치 승인

QA도 수시 모니터링을 통해 결함 조치가 완료된 건을 확인한 후, “확인 완료”로 상태 항목을 변경한다.

수행 tip

- 통합테스트 시 발견되는 소스 기능 테스트의 결함 관리는, 분석 설계 시 발견되는 결함과는 결함관리 산출물에서 약간 다르게 관리된다.

학습 3 교수 · 학습 방법

교수 방법

- 소프트웨어 테스팅의 개념, 테스트 계획서 및 테스트 관련 국제 표준 지식체계에 대한 이해 정도를 파악한 후 수업을 진행한다.
- 사전에 개인별 학습 자료를 준비하여 모든 학생이 참여할 수 있는 문제 해결식 수업, 협력 수업이 가능하도록 한다.
- 소프트웨어 결함의 종류에 대한 이해 정도를 파악한 후 수업을 진행한다.
- 결함 제거 및 결함 조치 이력관리에 대한 내용을 사례를 들어 설명한다.
- 교수자의 주도로 형상 관리에 대한 내용을 PPT 자료로 제시한 후 설명한다.
- 테스트 마감 활동에 대한 내용을 PPT 자료로 제시한 후 설명한다.
- 테스트 결함 식별 및 조치와 이력 관리 과정을 순서에 따라 단계적으로 실습이 이루어질 수 있도록 지도한다.

학습 방법

- 소프트웨어 테스팅에 대한 관련 용어를 숙지하고, 테스트 계획서, 테스트 관련 국제 표준 지식체에 대해 이해한다.
- 교수자가 제시하는 문제해결 시나리오에 학습자 전원이 적극적으로 참여하여 주도적인 학습이 되도록 학습한다.
- 소프트웨어 결함 관리에 대한 내용을 체계적으로 학습한다.
- 결함 제거 및 결함 조치 이력관리에 대한 내용을 사례와 함께 학습한다.
- 형상 관리에 대한 내용을 이해한다.
- 테스트 마감 활동에 대한 내용을 이해한다.
- 테스트 결함 식별 및 조치와 이력관리에 대한 전 과정을 실습한다.

학습 3 평가

평가 준거

- 평가자는 학습자가 수행 준거 및 평가 항목에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

| 학습 내용 | 평가 항목 | 성취수준 | | |
|---------------|---|------|---|---|
| | | 상 | 중 | 하 |
| 개발자 테스트 결함 관리 | - 개발자 테스트 수행 결과에서 발견된 결함을 식별하고 조치에 대한 우선순위를 결정하고 적용할 수 있다. | | | |
| | - 결함이 발생한 소스를 분석하여 추가적인 코딩으로 결함을 제거하고, 결함 조치 시 기존에 구현된 로직과의 연관성을 고려하여 부작용을 최소화할 수 있다. | | | |
| | - 개발자 통합 테스트 결과 결함 조치로 변경되는 소스의 버전을 관리하고, 결함 조치 결과에 대한 이력을 관리할 수 있다. | | | |

평가 방법

- 문제 해결 시나리오

| 학습내용 | 평가항목 | 성취수준 | | |
|---------------|-------------------------|------|---|---|
| | | 상 | 중 | 하 |
| 개발자 테스트 결함 관리 | - 결함 식별 및 조치 우선순위 결정 능력 | | | |
| | - 결함 조치 능력 | | | |
| | - 결함 조치 이력 관리 능력 | | | |

- 평가자 체크리스트

| 학습내용 | 평가항목 | 성취수준 | | |
|---------------|--------------------|------|---|---|
| | | 상 | 중 | 하 |
| 개발자 테스트 결함 관리 | - 결함 조치 우선순위 결정 능력 | | | |
| | - 결함 조치 결과 이력 관리 | | | |

피드백

1. 문제 해결 시나리오
 - 결함 조치 이력 관리 보고서를 평가하여 주요 사항 및 개선해야 할 사항에 대하여 표시하여 돌려준다.
2. 체크리스트를 통한 관찰
 - 결함조치 우선순위 결정 능력을 실습 과정에서 평가한 후에 개선해야 할 사항 등을 정리하여 돌려준다.



- 권원일 · 이현주 · 최승희 · 이승호 · 박은영 · 조현길(개정3판, 2014). 『개발자도 알아야 할 소프트웨어 테스트 실무』. STA테스팅컨설팅.
- 정보통신단체표준(2010). TTAK.KO-11.0103. 「소프트웨어 요구사항 품질 평가 항목」. 한국정보통신기술협회.
- 한혁수(개정 증보판, 2008). 『소프트웨어 공학의 소개』. 홍릉과학출판사.
- Rogers S.Pressman · Bruce R.Maxim(2015). 『소프트웨어공학 실무적 접근(Software Engineering: a Practitioner's Approach)』. 이은석 · 김성철 · 김영곤 · 김정아 · 송재승 · 이말레 · 이종근 · 조영석 · 진영택(공역). 홍릉과학출판사.
- 공개SW포털. <http://www.oss.kr>.
- 국가표준인증 종합정보센터(KSinfo). <http://standard.go.kr>.
- 네이버 지식백과. www.naver.com.
- 한국정보화진흥원. <http://www.nia.or.kr>.
- TMMI Foundation. www.tnmi.org.



문제해결 시나리오

| | |
|---|---------------------------------------|
| 평가 일자 : 사번: 이름: | |
| 학습 1) 개발자 테스트케이스 설계하기 | - 프로젝트의 특성을 반영한 테스트 케이스와 테스트 시나리오 작성 |
| 학습 2) 개발자 통합테스트하기 | - 통합테스트 계획에 따른 테스트 수행 순서 설명 |
| 학습 3) 개발자 결함조치하기 | - 테스트 수행 결과에서 발견된 결함을 식별하고 조치 우선순위 결정 |

테스트 수행계획서

| | | | |
|-----------------|-----------|-----------|---------|
| 프로젝트명 | | | |
| 시스템명 | | 작성자 | |
| 테스트 배경 | | | |
| 테스트별 테스트 항목 | | | |
| 테스트 종류 | 테스트 항목 | | |
| | | | |
| | | | |
| | | | |
| 테스트 일정 및 기법 담당자 | | | |
| 테스트 종류 | 테스트 일정 | 테스트 기법 | 테스트 담당자 |
| | | | |
| | | | |
| | | | |
| 테스트 환경 | | | |
| 테스트 종류 | SW 테스트 환경 | HW 테스트 환경 | 기 타 |
| | | | |
| | | | |
| | | | |

테스트 케이스 검토 체크리스트

| 구분 | 평가 점검 항목 | 평가 결과 | |
|-----|--|-------|-----|
| | | 적합 | 부적합 |
| 적정성 | 요구사항의 추적이 가능한가? | | |
| | 테스트계획서에 기술된 테스트 전략에 준하여 설계되었는가? | | |
| | 테스트 케이스의 수가 너무 많거나 적지 않은가? | | |
| | 체계적인 테스트 설계 기법을 적용하여 도출되었는가? | | |
| 다양성 | 비정상적인 조건을 테스트하는 케이스가 적절히 있는가? | | |
| | 부적절한 데이터도 적절히 포함되었는가? | | |
| | 예외 상황을 표현한 테스트 케이스가 적절히 있는가? | | |
| | 시스템이 제대로 작동하지 않도록 만드는 테스트 케이스가 적절히 있는가? | | |
| | 테스트 케이스 설계시 발생 가능한 결함 목록이 활용되었는가? | | |
| 완전성 | 테스트 케이스의 구성 항목이 적절하게 정의되었는가? | | |
| | 응답 시간의 적정성을 체크하는 것이 반영되었는가? | | |
| | 품질 특성별 각 항목을 검증하기 위한 내용이 반영되었는가? | | |
| 명확성 | 실행 결과와 비교하기 위한 사전 정의 예상 결과가 있는가? | | |
| | 선행 작업과 후속 작업이 필요시 결과 확인 절차가 식별되어 명확히 기술되었는가? | | |

통합 테스트 결과서

| | | | | | |
|--|--------------------------------------|------------|--|---------------|--|
| 시스템/ 서브시스템 | | 테스트 담당자 | | 테스트 일자 | |
| 테스트 케이스 ID | | 확인자 | | 완료 일자 | |
| 설명 | 테스트 케이스에 대한 간략한 설명 | | | | |
| 테스트 시나리오 | | | | | |
| 작업내용 | 작업구분/테스트데이터/관련 Use Case/테스트결과 및 오류내용 | | | | |
| 작업 절차 #1. 테스트 시나리오를 위해 수행해야 할 작업내용 기술 | 작업 구분 | | | | |
| | 테스트 데이터 | | | | |
| | 관련 Use Case ID/명 | | | | |
| | 테스트 결과 | | | 오류 발생 시 오류 내용 | |
| 작업 절차 #2. 테스트 시나리오를 위해 수행해야 할 작업내용 기술 | 작업 구분 | | | | |
| | 테스트 데이터 | | | | |
| | 관련 Use Case ID/명 | | | | |
| | 테스트 결과 | | | 오류 발생 시 오류 내용 | |
| 비 고 | 테스트 세부 내용 기재(필요시) | | | | |
| 최종 평가 | 합격 / 불합격 | | | | |
| | 사유: | | | | |

결함 관리 대장

| 발견 일자 | 발견자 | 결함 구분 | 영향 (심각도) | 우선 순위 | 상세 내용 | 결함 유입 단계 | 결함 유형 | 결론 제언 | 결함 상태 | 결함 조치 |
|----------|-----|-------|-----------------|-------------|-------|----------|-------|-------|----------|-------|
| 결함 발견 일자 | | | 결함이 시스템에 끼치는 영향 | 결함 조치 우선 순위 | | 결함 발견 단계 | | | 수명 주기 상태 | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |

NCS 학습모듈 개발진

(대표집필자)

강석진(이비스툼)

(집필진)

김보운(이화여자대학교)

김홍진(LG CNS)

유은희

장현섭((주)커리텍)

주선태(T3Q)

진권기(이비스툼)

최재준

(검토진)

김승현(경희대학교)

엄기영(우리에프아이이에스)

장운순(한국IT컨설팅)

조상욱(세종대학교)

조성호(삼성카드)

(개발기관)

최기원(한국소프트웨어기술진흥협회)

이두현(한국소프트웨어기술진흥협회)

(연구기관)

옥준필(한국직업능력개발원)

김상진(한국직업능력개발원)

김성남(한국직업능력개발원)

김지영(한국직업능력개발원)

문한나(한국직업능력개발원)

홍서희(한국직업능력개발원)

*표시는 NCS 개발진임

※ 본 학습모듈은 자격기본법 시행령 제8조 국가직무능력표준의 활용에 의거하여 개발하였으며
저작권법 25조에 따라 관리됩니다.

※ 본 학습모듈은 <http://www.ncs.go.kr>에서 확인 및 다운로드할 수 있습니다.



www.ncs.go.kr