10주차 1차시 테스트

【학습목표】

1. 테스트의 주된 목적 및 객관성과 타당성을 설명할 수 있다.
2. 테스트의 종류별로 각각의 특징을 통해 구분할 수 있다.

학습내용1 : 시스템 구현과 테스트

1. 구현(implementation)

1) 설계된 내용을 바탕으로 코드화

2. 시스템 구현과 테스트

1) 테스트의 개요
2) 테스트의 종류
3) 테스트 공정
4) 명세서 변경관 관리

학습내용2 : 테스트의 개요

1. 테스트(test)

1) Myers가 정의한 바와 같이 에러의 발견을 목적으로 하여 프로그램을 실행하는 과정

2. 테스트의 주된 목적

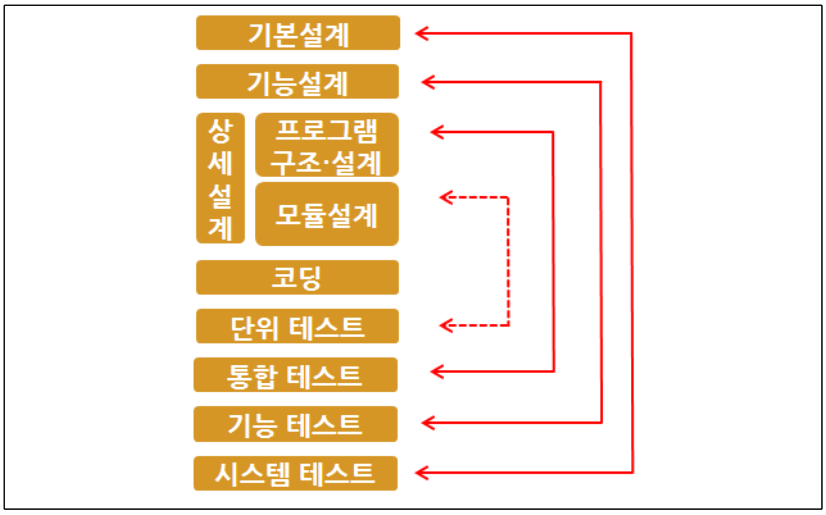
1) 소프트웨어(프로그램)에 포함되어 있는 에러(버그)를 찾아내기 위함임

3. 테스트의 객관성과 타당성

1) 테스트는 객관적이고 타당성을 확보하기 위하여 개발그룹이 아닌 독립한 부문인 테스트 팀(test team)에 의해서 이루어지는 게 바람직 함

학습내용3 : 테스트의 종류

1. 개발공정과 테스트 공정의 대응



2. 단위 테스트(unit test)

1) 시스템을 구성하는 최소 단위인 모듈에 착안
- 시스템을 수정 보완하거나 변경시키는 경우에 발생되는 비용

2) 목적
- 모듈의 논리(logic)나 인터페이스와 그 모듈의 외부명세(모듈의 기능, 입출력, 외부로의 영향)와 맞지 않는 사항을 발견하는 것
- 보편적으로 다른 모듈과 구분 격리된 상태에서 진행

3) 목적 달성을 위하여
- 모듈의 외부명세(기능)에 착안한 모듈 기능 테스트(module function test)
- 모듈 기능 테스트로 실행되지 않은 경로(path)를 모듈 로직 테스트(module logic test)로 보완하는 순서로 진행

4) 특별한 드라이버 모듈(driver module)과 스터브 모듈(stubmodule)이 필요한 경우가 있음

* 드라이버 모듈
① 피테스트 모듈
② 테스트 데이터를 보내거나 결과를 체크하는 경우에 필요

* 스터브 모듈
① 피테스트 모듈
② 다른 모듈을 호출(CALL)할 때 그 모듈을 시뮬레이트 함

5) 모듈 기능 테스트(module function test)
- 모듈의 외부명세를 바탕으로 모듈의 기능(모든 입출력 조건, 옵션, 에러 케이스 등)에 대하여 테스트

6) 모듈 로직 테스트(module logic test)
- 모듈의 상세명세 또는 프로그램을 기초로 하여 모듈의 논리(logic)에 착안하여 테스트를 실시하는 방법

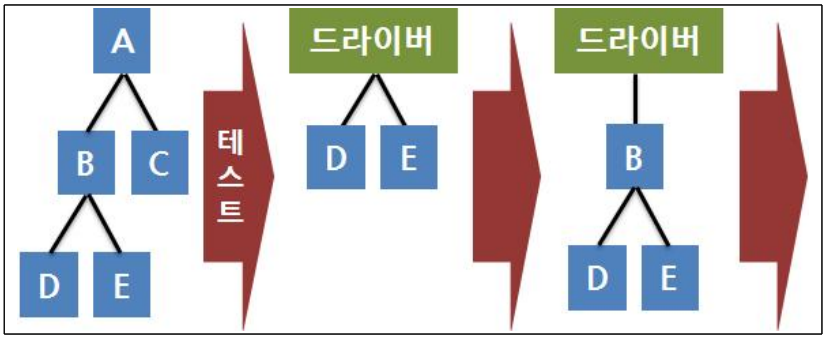
3. 통합 테스트 (integration test)

1) 단위 테스트 시에 드라이버 혹은 스터브를 이용하여 행한 모듈 사이의 인터페이스 테스트(interface test)를 실제로 통합된 모듈을 대상으로 하여 수행

2) 테스트 항목
- 단위 테스트에서 사용하던 테스트 항목 중에서 모듈 사이의 인터페이스에 관계되는 것을 선택하여 테스트하는 것이 일반적 관행

3) 모듈의 통합방법에 따라서 여러 가지 형태로 분류
- 상향식 테스트
- 하향식 테스트
- 혼합식 테스트
- 빅뱅 테스트

4) 상향식 테스트(bottom up test)
- 최하위에 위치한 모듈을 개발하여 테스트한 후에 그 모듈을 호출하는 모듈을 개발하여 이미 테스트를 마친 모듈과 결합하여 테스트를 하는 식으로 점차 상위(정상)에 있는 모듈을 개발하여 결합하면서 테스트하는 방식

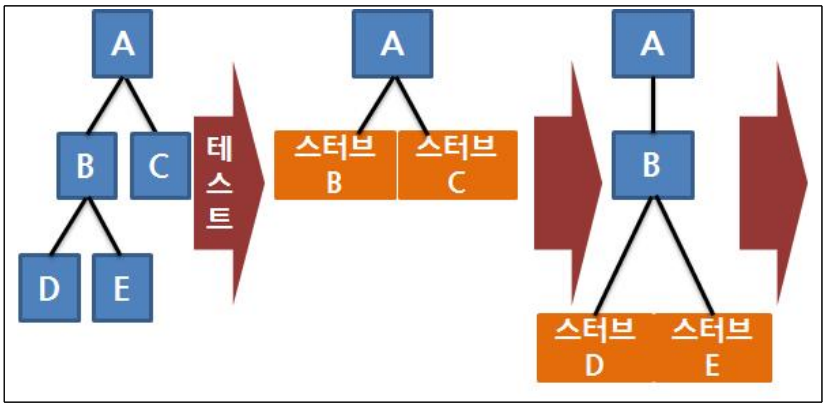


* 테스트 드라이버(test driver)
① 제어모듈의 모의 프로그램 또는 단위 테스트를 위한 프로그램
② 기능
  - 하위모듈의 호출
  - 하위모듈을 호출할 때 입력 데이터를 하위모듈에 전송 등과 같은 역할
③ 테스트 드라이버는 하위모듈을 호출하고 원하는 기능 수행에 필요한 데이터를 전달하기 위해서 작성된 것

* 상향식 테스트의 장점
① 다수의 요원들이 동시에 개발 및 테스트를 진행함으로써 시간단축과 효율을 높일 수 있음
② 하위단계의 프로그램 이미지 파악이 쉽고 정확하게 파악된 부분부터 취급할 수 있음
③ 개발요원이 많이 존재하는 현 체제에서 모두 동시 참여할 수 있음
④ 모듈에 대한 단위 테스트가 철저히 이루어짐

* 상향식 테스트의 단점
① 최상위 모듈을 제외하고 모든 모듈에 대한 테스트 드라이버를 만들어야 하기 때문에 추가적인 노력과 시간이 필요함
- 생산성을 낮게 하는 요인이 됨
② 모듈 사이의 인터페이스가 가상적인 상태로 테스트가 이루어짐
- 일반적으로 이 방식에서 실제적인 상황과 같은 테스트는 개발의 종료단계에서 이루어짐
③ 이 방식에서 개발 및 테스트에 다수의 관계요원이 참여하는 것은 다음과 같은 문제점을 일으킬 가능성이 많음
- 경험이 적은 요원의 참여로 개발제품의 질적 저하 및 테스트가 불완전하게 이루어질 가능성이 있음
- 다수의 요원 사이에 불완전한 정보교환으로 사고가 발생할 가능성이 있음

5) 하향식 테스트(top-down test)
- 시스템을 구성하고 있는 최상위에 위치한 모듈을 개발하여 테스트하고, 다음에는 이 모듈이 호출할 하위모듈을 개발하여 이미 테스트 완료된 모듈과 결합하여 테스트하는 식으로 점차 하위에 위치한 모듈을 개발하여 결합하면서 테스트



* 스터브(stub)
① 상위모듈이 실행하는 과정에서 하위모듈을 호출할 필요성이 제기될 경우에 제어(control)를 넘겨받고 다시 되돌려 줌으로써 상위모듈에서 원하는 최적의 요건을 제공하는 것
② 스터브의 조건
- 실행성을 갖추어야 함
- 내용이 단순 명백해야 함

* 하향식 테스트의 장점
① 인터페이스에 대한 가정(假定)이 적고 실제와 유사한 환경에서 개발이 진행되므로 에러가 적어짐
② 초기의 테스트부터 인터페이스 검증이 가능함
③ 하위모듈을 테스트할 경우에 테스트가 완료된 상위모듈을 사용하기 때문에 테스트 자체가 시스템의 실제 가동과 유사한 환경에서 이루어짐
④ 상위모듈은 하위모듈의 호출로 인해서 반복적으로 테스트되므로 에러의 잠재 가능성이 감소되어 시스템의 신뢰성이 향상됨

⑤ 새로 구현될 모듈의 테스트에서 에러가 발생된다면 그 원인의 대부분은 상위모듈에 있는 것이 아니고 새로 구현된 모듈에 존재하므로 에러 발견이 쉬움
⑥ 스터브의 내용은 비교적 간단하고 단순한 기능을 요구하는 것이 대부분이기 때문에 시간과 노력이 적게 듬
⑦ 테스트를 위한 노력이나 시간이 소프트웨어 개발과정의 특정 단계에 집중되지 않고 모든 과정에 고르게 평준화 현상을 보임

* 하향식 테스트의 단점
① 개개의 모듈에 대한 완전한 테스트가 곤란한 경우가 있음
- 모듈에 대한 테스트 완료부분과 테스트 미실시 부분을 명확히 구분해 두지 않을 경우 테스트에 누락되는 부분이 발생할 가능성이 있음
- 테스트에 상위모듈을 사용하기 때문에 그 상위모듈의 제약으로 인하여 모든 논리경로에 대해 테스트하지
- 못하는 경우가 있음
- 스터브의 제약으로 모든 논리경로에 대해 테스트하지 못할 경우도 있음

② 초기의 테스트 과정에서는 프로그램의 구현활동(코딩)과 테스트를 동시에 수행하기 어려움

6) 빅뱅 테스트(big bang test)
- 하나하나의 모듈을 개발하고 테스트하여 각 모듈의 테스트가 종료된 시점에서 한 번에 결합하여 테스트하는 방식

* 빅뱅 테스트 방식의 결점
① 모듈은 테스트 과정이 완전하게 완료되지도 않고 또한 모듈들도 통합되어 있지 않은 상태이기 때문에 인터페이스에 중대한 에러가 잠재되어 있더라도 오랫동안 방치될 가능성이 많음
② 모듈은 경우에 따라서는 드라이버와 스터브 양쪽을 모두 필요로 하기 때문에 문제가 발생될 수도 있음

7) 샌드위치 테스트(sandwich test)
- 상향식 및 하향식 테스트 각각의 단점을 배제하고 장점을 취하는 테스트 방식
- 일반적으로 하향식 및 상향식 테스트를 동시에 시작하여 상위 및 하위에서 프로그램을 통합하고 최종적으로 어느 한 지점에서 만나게 함

4. 기능 테스트(function test)

1) 목적
- 프로그램과 외부의 기능을 규정한 기능명세서 사이에 불일치 하는 사항을 찾아내기 위해서 실시됨

2) 기능명세서를 바탕으로 하여 “원인-결과 그래프(cause-effect graph)기법”이나 “실험계획법을 이용하는 기법” 등에 의해서 테스트 항목을 설정하여 실시함

3) 기능 테스트
- 개개의 프로그램에 대한 기능 테스트
- 프로그램 전체를 처음부터 끝까지 하는 기능 테스트
- 두 가지 모두 기능명세서를 바탕으로 실시할 필요가 있지만 특히 후자에 대해서는 기본명세서의 시스템 구성도를 기초로 하여관련하는 프로그램 사이의 인터페이스에도 유념할 필요

5. 시스템 테스트(system test)

1) 프로그램과 산출물 목표와의 불일치한 사항을 발견해 내기 위한 목적으로 실시

2) 시스템 테스트는 산출물의 목표를 규정한 “기본명세서”를 기준으로 하여 실제 구현된 프로그램에 대하여 다음과 같은 관점에서 실시

3) 부하 테스트(stress test)
- 짧은 기간에 극단적으로 무거운 부하를 시스템에 걸리게 하여 그 결과 시스템에 나타나는 반응의 정도와 상태를 테스트하는 방법

* 예 : 시분할 시스템(TSS : time sharing system)에서 모든 사용자가 로그 온(log-on)하고 그 결과 시스템에 나타나는 반응의 정도나 시스템의 상태를 테스트하는 것과 같은 원리를 의미

4) 볼륨 테스트(volume test)
- 시스템의 처리작업이 어느 정도 효율적인가를 테스트하는 방식

* 예 : 컴파일러에 대단히 길이가 긴 소스 프로그램(source program)을 컴파일 시킬 때 그 효율성을 점검하는 것과 같은 테스트

5) 기억력 테스트(storage test)
- 주기억장치와 보조기억장치의 기억용량에 대한 시스템의 목표가 달성되었는지 여부를 체크하는 방식

6) 구성 테스트(configuration test)
- 시스템을 지원하는 하드웨어 및 소프트웨어 구성에 적합성 정도를 테스트하는 방식

7) 적합성 테스트(compatibility/degrade test)
- 기존의 기능이 그대로 작동하는지 여부를 체크하는 방식으로서 호환성 테스트(degrade test)라고도 함

8) 시큐리티 테스트(security test)
- 시스템의 시큐리티 기능을 테스트하는 방식

* 예 : 다른 사용자의 데이터 또는 프로그램을 쉽게 참조할 수 있도록 되어있는지 여부를 테스트하는 것과 같은 유형의 테스트

9) 성능 테스트(performance test)
- 응답시간(response time)이나 생산성(throughput) 등의 시스템에 성능목표가 달성되고 있는지 여부를 체크하는 방식

10) 신뢰성 테스트(reliability test)
- MTBF 또는 잔존 버그 수와 같은 시스템의 신뢰성 목표가 달성되고 있는지 여부를 테스트하는 방법

11) 복구 테스트(recovery test)
- 데이터베이스에서 상실한 데이터의 복구 등과 같은 시스템의 복구기능에 대한 테스트 방식

12) 유지보수성 테스트(maintainability test)
- 유지보수에 대한 툴(tool)이나 매뉴얼에 대한 테스트

* 예 : 에러의 징후가 보일 때 해당 매뉴얼에 의해서 어떻게 정확하게 그 에러의 원인을 찾을 수 있는지 여부를 체크하는 것과 같은 원리

13) 매뉴얼 테스트(manual test)
- 사용자를 위한 매뉴얼의 정확성에 대한 테스트
- 특히 기재된 내용의 에러, 애매모호성, 용어의 통일성 여부, 표현 부족 또는 부적절과 같은 사항을 중심으로 정당성을 확인

14) 유용성 테스트(usability test)
- 시스템 사용의 용이성 여부를 테스트하는 방식

15) “기본명세서”에서 규정되어 있는 항목에 대해서는 모두 테스트를 실시할 필요가 있음

16) 이들 항목의 일부는 단위 테스트나 기능 테스트의 항목과 중복되지만 끝까지 시스템 테스트의 관점에서 독자적으로 실시할 필요가 있음

6. 개발공정 이후의 테스트

1) 개발공정에서 4단계의 테스트가 완료되어 완성된 제품은 최종사용자에게 인도되기 전에 최종적으로 추가적인 테스트를 거침

2) 수락 테스트(acceptance test : 필드 테스트”field test” : 사용자 테스트”user test”)
- 사용자의 승인을 받기 위해서 실시
- 실제로 개발한 시스템으로 처리한 결과와 수작업에 의해 얻은 결과를 대조하고 비교 분석하여 사용자의 요구조건을 충족시키고 있는지 여부를 테스트
- 수락 테스트의 주된 내용 : 처리절차, 중요기능, 문서화 등이 사용자의 요구대로 되었는지를 체크
- 테스트의 주체 : 개발자가 아닌 사용자 자신이거나 별도의 테스트팀이 진행
- 일반적으로 테스트의 종료시기 : 사용자가 만족하여 수락하거나 또는 모든 테스트 경로마다 작성된 테스트 데이터를 모두 실행시킨 결과가 별다른 이상을 보이지 않을 경우

* 수락 테스트 과정
① 설계검증 테스트(design verification test)
    - 시스템의 실제 작동환경을 모방한 여건 하에서 모의 데이터에 따라 실시하는 테스트
    - 사용자의 실제 요구사항이 설계과정에서 누락되거나 잘못된 부분을 찾아낼 수 있음
② 작동확인 테스트
    - 시스템을 실제 상황에서 실제 데이터를 가지고 수행시켜 시스템이 업무처리에 문제가 없는지 여부를 체크하는 방법
    - 다음과 같은 사항들에 대해서 확인

* 작동확인 테스트 확인 사항
- 단위 시간당 작업량(throughput), 응답시간(response time), 피크타임(peak time) 시의 처리 등에 처리속도가 적합한지 여부가 확인된다.
- 시스템의 운영절차와 합리성, 비효율성, 편의성의 측면에서 문제가 있는지 여부 및 수준이 확인된다.
- 사용자 및 관련요원의 사용상 용이성 수준이 확인된다. 시스템의 백업 및 데이터 복구의 편리 및 용이성이 확인된다.

* 감사 테스트(EDP audit)
① 시스템의 기밀보호 및 범죄행위로부터 어느 정도 안전하게 구축되었는지 그 수준을 파악할 수 있음
② 문제가 있다면 보완개선을 지적 또는 지시 할 수 있음

3) 도입 테스트
- 개발을 완료하여 설치한 시스템을 도입하여 운용하는 과정에서 에러가 발생되지 않을 것인지를 체크하고 검증하는 테스트
- 시스템을 개발자로부터 인도받은 이후 일정 기간 시험운영과 종합적인 측면을 점검하면서 진행 됨

【학습정리】

1. 시스템 구현과 테스트를 익힌다.

2. 테스트의 개요를 알아본다.

3. 테스트의 종류를 이해한다.