9주 1강

테스트

이번 주차에는…

실전 적용 시뮬레이션

- 테스트의 이해
- 테스트의 분류
- 정적 테스트
- 동적 테스트
- 소프트웨어 개발 단계에 따른 테스트
- 기본 자바 소개 2

1. 테스트와 소프트웨어

死.後.藥.方.文

사람의 죽은 뒤에 약을 짓는다

그림 8-1 **테스트가 주는 교훈** : 절벽 밑에 응급차를 마련해놓는 것보다 절벽 위에 울타리를 쳐놓는 것이 안전하다.

2. 소프트웨어 오류로 인한 사고 사례

- 걸프전 당시 페트리어트 미사일 실패
  - 미사일이 1 km를 날아갈 때마다 0.5~1초 정도의 아주 미세한 오차 발생 때문

- 방사선 치료기 테락-25의 사고
  - 테락-20을 테락-25로 버전 업할 때, 사고가 없던 테락-20만 믿고 테락-25의 전체 코드에 대한 테스트를 생략했기 때문

- 상업용 우주선 아리안 5호의 공중 폭발
  - 64비트 값을 16비트 정수로 변환하는 과정에서 overflow → 자동 폭발 장치 작동
  - 아리안 4호 프로젝트에서 내버려둔 불필요한 dead code를 간과한 결과

SW의 사소한 결함 → 대형 사고

3. 전문가들의 소프트웨어 정의

- IEEE
  - 테스트는 시스템이 명시된 요구를 잘 만족하는지, 즉 예상된 결과와 실제 결과가 어떤 차이를 보이는지 수동이나 자동으로 검사하고 평가하는 작업

- Zoha Manna
  - 테스트는 시스템의 명세까지 완벽하게 옳다고 확신할 수 없고, 테스트 시스템 그 자체가 맞다고 증명할 수 없기 때문에 프로그램을 완전히 테스트할 수 없다.

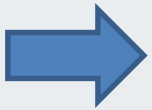
- Dahl, Dijkstra, Hoare
  - 테스트는 결함이 있음을 보여줄 뿐, 결함이 없음을 증명할 수는 없다.

3. 소프트웨어 테스트 정의

- 소프트웨어에 내에 존재하지만 드러나지 않고 숨어 있는 오류를 발견할 목적으로, 개발 과정에서 생성되는 문서나 프로그램에 있는 오류를 여러 기술을 이용해 검출하는 작업

BUT

- 오류를 찾아내 정상적으로 실행될 수 있도록 하는 정도이지, 소프트웨어에 오류가 없음을 확인시켜주지는 못한다.

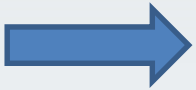


테스트는 오류를 찾고 올바르게 수정하여 프로그램을 작동시킬 수는 있지만, 그 프로그램이 완전하고 정확하다고 증명할 수는 없다.

4. 소프트웨어 테스트의 목표

- 작은 의미
  - 원시 코드 속에 남아 있는 오류를 발견하는 것
  - 결함이 생기지 않도록 예방하는 것

- 큰 의미
  - 개발된 소프트웨어가 고객의 요구를 만족시키는지 확인시켜주는 것
  - 개발자와 고객에게 사용하기에 충분한 소프트웨어임을 보여주는 것



개발된 소프트웨어에 신뢰성을 높여주기 위한 작업

5. 소프트웨어 테스트의 어려움과 특징

- 테스트 케이스가 적어 효과에 한계가 있다
- 완벽한 테스트 케이스를 도출하기 어렵다
- 테스트를 위한 실제 사용 환경을 구축하기 어렵다
- 작은 실수를 발견하기 어렵다
- 테스트 중요성에 대한 인식이 부족하다
- 고객의 요구 사항을 충족시켜야 한다
- 테스트 단계에서만 수행되는 단순한 활동이 아니라 개발 단계와 함께한다
- 파레토 원리를 적용할 수 있다
- 모듈 단위를 점점 확대해나가며 진행한다
- 완벽한 테스트는 불가능하다
- 개발자와 다른 별도의 팀에서 수행한다
- 살충제 패러독스(테스트 내성) 문제 해결을 위해 테스트 케이스 업데이트가 필요하다

6. 테스트에서 결함 관련 용어

- 오류error
  - 소프트웨어 개발자에 의해 만들어지는 실수로 결함의 원인이 된다.
- 결함defect, bug, fault
  - 오류에 의해 프로그램이 완전치 못한 것으로, 고장의 원인이 된다.
- 고장, 실패failure, 문제problem, 장애
  - 시스템이 요구 사항대로 작동하지 않는 것을 말한다.

7. 테스트 절차

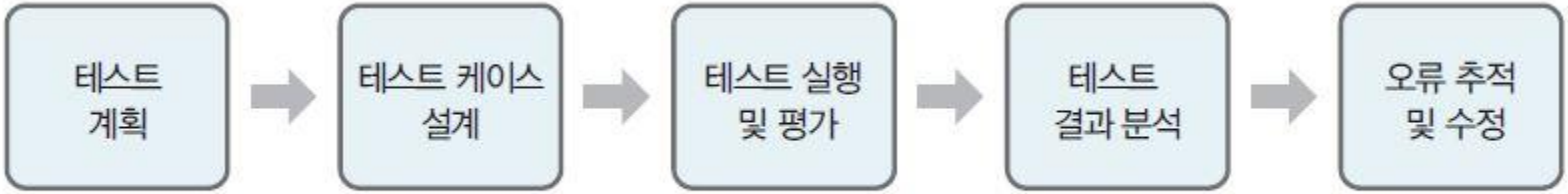


그림 8-2 **테스트 절차**

8. 테스트 계획

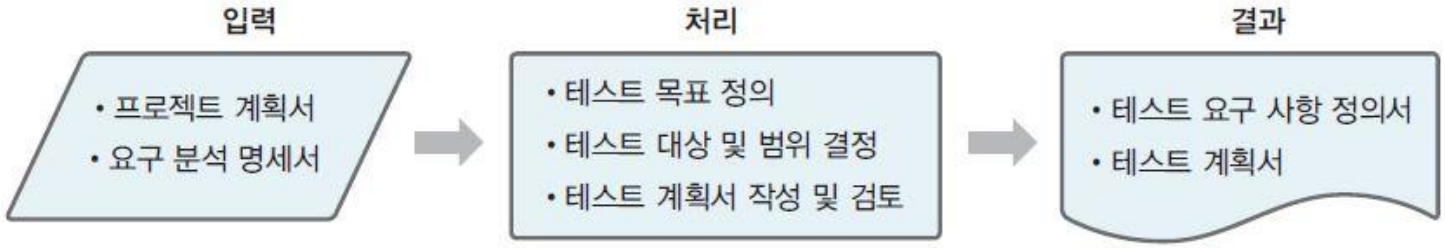


그림 8-3 소프트웨어 테스트 계획 단계

- 테스트 목표 정의
  - 테스트 항목 중에서 어떤 항목을 중점적으로 테스트할 것인지 명확히 나타낸다.
- 테스트 대상 및 범위 결정
- 테스트 계획서 작성 및 검토
  - 테스트의 목적, 담당 인원, 테스트 전략과 접근 방법 수립, 필요한 자원 및 자원 확보 일정, 실시할 테스트의 종류, 적용할 테스트 기법, 일정 등에 관한 정보를 기록

9. 테스트 케이스 설계

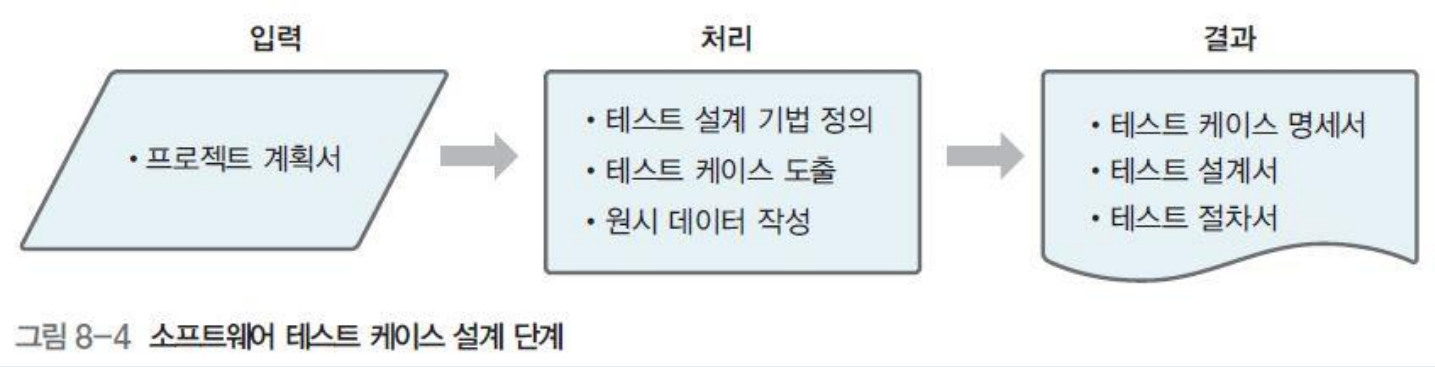


그림 8-4 소프트웨어 테스트 케이스 설계 단계

- 테스트 케이스 설계 기법 정의
  - 테스트할 프로젝트 문제의 성격을 파악하고 이 문제에 적합한 테스트 기법을 선정
- 테스트 케이스 도출
- 원시 데이터 작성

10. 테스트 실행 및 측정

그림 8-5 소프트웨어 테스트 실행 및 측정 단계

- **테스트 환경 구축**
  - 테스트 계획서에 정의된 환경 및 자원을 설정하여 테스트를 실행할 준비를 한다.

- **테스트 실행 및 측정**

11. 테스트 결과 분석

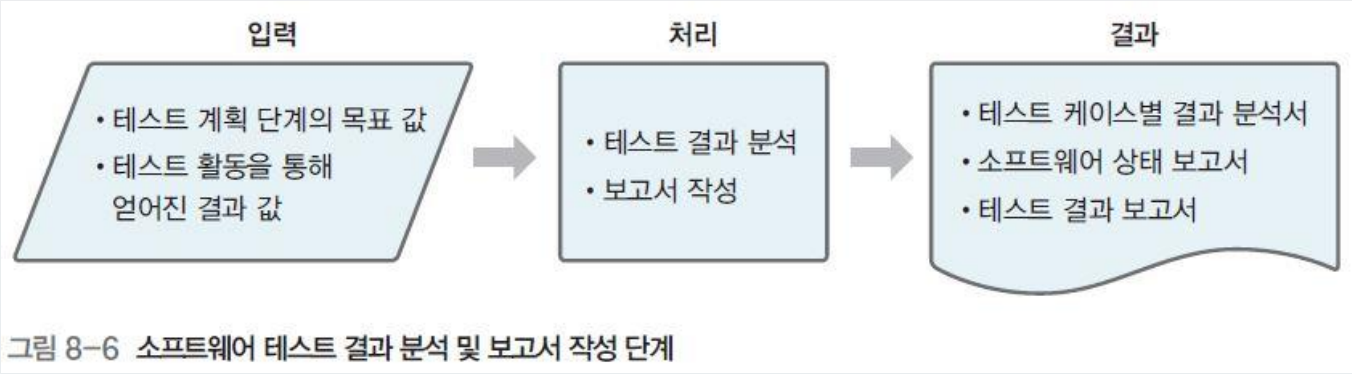


그림 8-6 소프트웨어 테스트 결과 분석 및 보고서 작성 단계

- **테스트 결과 분석**
  - 테스트 활동을 통해 얻어진 결과 값과 테스트 계획 단계에서 목표한 값을 비교하여 예정된 테스트 품질 목표가 달성되었는지를 비교·분석

- **보고서 작성**
  - 테스트를 수행한 결과와 테스트를 수행하는 데 사용된 방법 등을 기술
  - 결과에 따른 평가와 권고 사항도 기술

12. 오류 추적 및 수정

그림 8-7 오류 추적 및 수정 단계

- **오류 수정 계획**
  - 테스트 결과 보고서를 기반으로 오류가 발생된 위치를 찾아내고, 오류 수정 우선순위를 결정하여 오류 제거 계획을 세운다.
- **오류 수정**
- **수정된 내용 보고**

13. 시각에 따른 테스트(1)

[문제] 1부터 10까지 덧셈 프로그램

- 확인 테스트(verification rest)
  - 1부터 10까지 덧셈의 결과가 정확한지만 테스트, 결과가 55나오면 → 확인 테스트 합격!

But

[사용자가 원하는 실제 문제] 1부터 10까지 곱셈 프로그램이었다면

→ 확인 테스트 체크 안됨

- 각 단계에서 개발자의 시각으로 테스트
- 설계도 대로 만들었는지 테스트
- 이전 단계에서 생성된 산출물이 현 단계의 산출물에 정확히 반영되었는지 테스트

14. 시각에 따른 테스트(2)

- 확인 테스트의 문제점
  - 사용자의 요구가 맞는지 틀리는지는 체크를 안 해보고, 오직 계산 과정이 맞는지 만 검증

- 검증 테스트(validation test)
  - 1부터 10까지 곱셈을 했는지 테스트, 결과는 덧셈을 했으므로 → 검증 테스트 **불합격!**
  - 사용자의 요구 사항대로 만들었는지를 테스트

사용자의 시각에서 테스트
- 사용자가 원하는 것을 만들었는지
- 완성된 제품이 사용자의 요구 사항을 모두 충족하는지

→ 소프트웨어가 사용자의 목적에 맞게 구현되었는지 확인 가능

15. 사용 목적에 따른 테스트 - 운영 목적 적합성 테스트

- 소프트웨어가 시스템의 운영 목적에 적합한지를 테스트

테스트	설명
성능	사용자의 요구 사항 중 성능과 관련된 요구 사항을 얼마나 준수하는지 테스트 예상된 부하에 대한 실행시간, 응답 시간, 처리 능력, 자원 사용량 등을 테스트
신뢰성	4장에서 설명
강건	비정상 상태(예상치 못한 입력, 예외적인 입력, 평상시보다 몇 배 많은 입력 데이터)에서도 소프트웨어가 올바르게 동작하는지 테스트
스트레스	평소보다 많은 비정상적인 값, 양, 빈도, 부피 등으로 부하를 발생시켜 부하가 최고치인 상황에서 시스템의 반응을 살피고 이때 발생하는 오류를 찾는 것
부하	소프트웨어가 과부하인 상태를 체크
보안	부당하고 불법적인 침입을 시도하여, 시스템을 보호하기 위해 구축된 보안 시스템이 불법적인 침투를 잘 막아내는지 테스트
사용성	4장에서 설명
안정성	며칠 동안 부하를 주면서 시스템이 안정적으로 돌아가는지를 테스트

16. 사용 목적에 따른 테스트 - 수정 용이성 테스트

- 소프트웨어 수정이 얼마나 쉬운지 테스트

테스트	설명
테스트 용이성	사용자의 요구 사항을 만족할 만큼 잘 수행하고 있는지를 얼마나 쉽고, 효율적이고, 철저하게 테스트할 수 있는가를 테스트
유지보수성	수정으로 인해 오류를 발생시키지 않고 변경시킬 수 있는지를 테스트

- 상호 운영성 테스트
  - 양립성compatibility, 일치성conformance, 이식성portability, 재사용성reusability 등을 체크

17. 사용 목적에 따른 테스트 - 운영지원 용이성 테스트

- 문서화documentation, 복원 가능성recovery, restart 등을 체크

테스트	설명
복원 가능성	소프트웨어를 고장 나게 해놓고(문제를 발생시켜놓고) 소프트웨어 복구가 잘 되는지(소프트웨어의 복구 능력) 확인해보는 테스트 회복이 소프트웨어에서 자동으로 이루어진다면: 재초기화가 제대로 수행되는지, 데이터는 완전하게 복구되었는지, 재시작이 정상적으로 이루어졌는지 등과 같은 소프트웨어 회복의 완벽성을 평가 만약 기술자에 의해 소프트웨어가 복구된다면: 고장 수리에 소요되는 평균 시간이 허용 범위 한계치 내인지 등을 테스트

18. 프로그램 실행 여부에 따른 테스트(1)

- 정적 테스트
  - 프로그램을 실행하지 않고 코드를 검토하며 오류를 찾는 방법

그림 8-8 자동차 내부 상태를 체크하여 고장을 찾는 방법

19. 프로그램 실행 여부에 따른 테스트(2)

- 동적 테스트
  - 프로그램을 실행하면서 오류를 찾는 방법

그림 8-9 자동차 주행을 통해 고장을 찾는 방법

20. 정적 테스트

- 정적 테스트(static test)
  - 프로그램 코드를 실행하지 않고 여러 참여자가 모여 소프트웨어 개발 중에 생성되는 모든 명세나 코드를 검토해서 실패(failures)보다는 결함(defects)을 찾아내는 방법

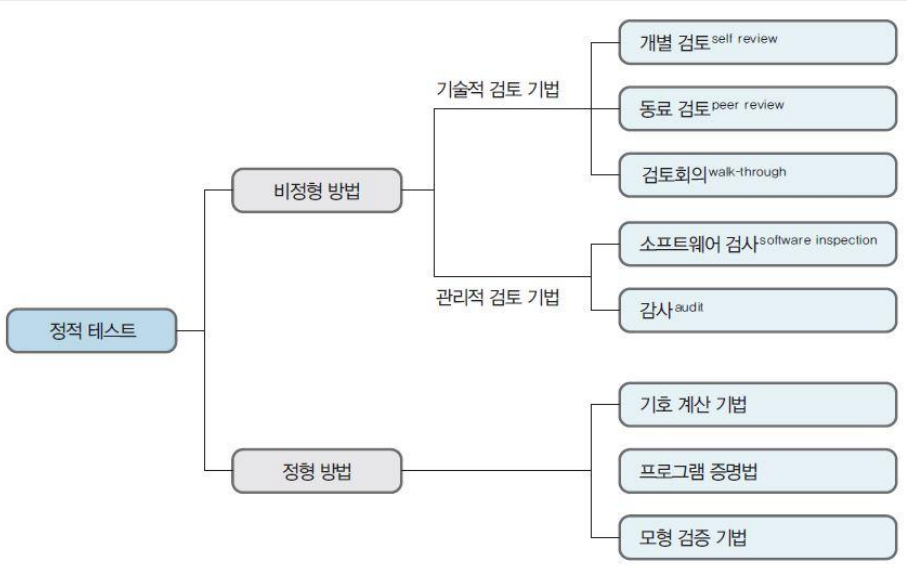


그림 8-10 정적 테스트 방법

21. 비공식/공식 검토

- 비공식 검토informal technical review
  - 산출물(문서, 프로그램)을 동료와 함께 책상에서 검사.
  - 제품을 검토할 목적으로 하는 간단한 만남 개별 검토, 동료 검토

- 공식 검토formal technical review
  - 동료와 전문가들이 수행.
  - 결함을 찾기 위해 정의된 절차에 따라 적절히 계획되고 통제된다.
  - 검토회의walk-through와 소프트웨어 검사software inspection

22. 공식 검토

- 공식 검토 내용
  - 원시 코드상에 존재하는 오류 검토
  - 소프트웨어가 사용자의 요구를 충분히 반영했는지 검토
  - 소프트웨어가 미리 정의된 표준을 지키는지 검토
  - 소프트웨어 개발 방식이 일관적인지 검토
  - 소프트웨어

- 공식 검토 수행 절차

그림 8-11 공식 검토 절차

23. 정적 테스트(1)

- 개별 검토self review
  - 체크리스트를 가지고 본인이 개발한 코드와 산출물 등을 검토
  - 본인 스스로 검토 → 가장 간단한 방법, 상대적으로 객관성이 떨어짐

- 동료 검토peer review
  - 동료에게 원시 코드나 여러 가지 산출물에 대한 검토를 의뢰하여 오류를 찾는 방법
  - 정해진 형식도 없고 별도의 격식을 차린 회의를 수행할 필요가 없어 비공식 검토

그림 8-12 훈수 두는 모습

24. 정적 테스트(2)

- 검토 회의work-through
  - 개발자가 소집한 전문가들에 의해 개발자의 작업을 검토
  - 3~5명 정도의 전문가(프로젝트 팀장, 다른 개발자, 품질 보증단장)들이 절차에 따라 평가
  - 설계 문서들이 고객의 요구 사항을 정확히 명시하고 있는지 여부, 작업 진척 상황 등 확인
- 검토회의 준비 및 주의 사항

- 검토회의를 통해 검토 받고자 하는 개발자는 회의 자료(검토 받을 문서 또는 원시 코드)를 준비하고 회의 일정을 계획한다.
- 검토회의의 결과를 인사 평가 자료로 사용해서는 안 된다.
- 회의 자료는 회의가 있기 4~6일 전에 전달되어 미리 살펴보고 올 수 있도록 한다.
- 검토회의는 문제점을 찾는 데 주안점을 두고, 그 문제의 해결은 검토회의 이후로 미룬다.
- 검토회의 때 발견되고 작성된 오류 리스트는 회의가 끝난 후 개발자에게 전달한다.
- 검토회의 시간은 너무 길지 않아야 한다. 일반적으로 1~2시간 이내가 좋다.

25. 정적 테스트(3)

- **소프트웨어 검사**software inspection
  - 검토회의: 문제점을 찾는 데 초점을 두고, 검토회의 후 개발자가 해당 문제를 수정
  - 소프트웨어 검사: 문제점 수정 지침까지도 제시 및 수정을 잘하고 있는지 추후에 조사
  - 원시 코드뿐 아니라 각 단계 산출물의 문서 등을 포함하여 분석하고 품질을 평가
  - 소프트웨어 품질 보증 기법으로 유용
  - 공식 검토에 속함

- **소프트웨어 검사 절차**

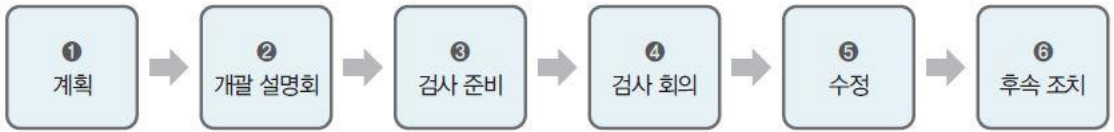


그림 8-13 소프트웨어 검사의 절차

26. 정적 테스트(4)

- **소프트웨어 검사 시 지켜야 할 원칙**

- 검사 회의는 2시간 이내로 하는 것이 적당하다.
- 검사 회의에 참가하는 총 인원은 5명 내외가 적당하다.
- 검토자가 사전에 읽어보고 올 수 있도록 최소한 2일 전에는 자료를 전달한다.
- 주제를 벗어나는 개별적인 질문은 삼가도록 한다.
- 발견된 오류에 대해서는 반드시 문서화하여 기록으로 남긴다.
- 검사 회의 목적은 오류를 발견하는 것이다. 따라서 오류 수정은 하지 않는다.
- 검사 회의가 끝나면 문서를 정리하여 관련된 사람들에게 전달한다.

27. 정적 테스트(5)

- 설계와 구현 시 소프트웨어 검사

표 8-2 설계와 구현 시 소프트웨어 검사

설계 검사	• '사용자가 요구한 기능들이 설계에 반영이 되었는가'와 같은 완전성 검사 • '사용되는 용어가 서로 다르게 해석되지 않았는가'와 같은 일관성 검사 • '모듈 사이에 인터페이스가 잘 정의되었는가'와 같은 정확성 검사
원시 코드 검사	• '프로그램에서 변수를 사용하기 전에 선언이 되었는가'와 같은 데이터 결함 검사 • '제어문에서 조건이 정확한가'와 같은 제어 결함 검사 • '모든 입력 변수가 사용되고 있는가'와 같은 입출력 결함 검사 • '인자의 데이터 타입, 개수, 순서가 일치하는가'와 같은 인터페이스 결함 검사

다음 시간

동적 테스트, 소프트웨어 개발 단계에 따른 테스트