애플리케이션 테스트 관리

chap1. 애플리케이션 테스트 케이스 설계

소프트웨어 테스트의 이해

- 소프트웨어 테스트 필요성 (**발, 예, 향**)
    1. 오류 발견 관점
    2. 오류 예방 관점
    3. 품질 향상 관점
- 소프트웨어 테스트 기본 원칙
    - 결완초집 살정오
        - 테스팅은 결함이 존재함을 밝히는 것
        - 완벽한 테스팅은 불가능함
        - 개발 초기에 테스팅 시작
        - 결함 집중
        - 살충제 패러독스
        - 테스팅은 정황에 의존적
        - 오류-부재의 궤변
- 소프트웨어 테스트 산출물
    - 테스트 계획서
    - 테스트 베이시스
    - 테스트 케이스
    - 테스트 슈트
    - 테스트 시나리오
    - 테스트 스크립트
    - 테스트 결과서

소프트웨어 테스트 유형

소프트웨어 테스트 유형은

1. 프로그램 실행 여부
2. 테스트 상세 기법
3. 테스트에 대한 시각

4. 테스트의 목적
5. 테스트의 종류

- 프로그램 실행 여부에 따른 분류 -> **정적 테스트와 동적테스트**로 나눔.
    - 정적 테스트 : 실행하지 않고 구조를 분석함. **리뷰, 정적 분석**
    - 동적 테스트 : 소프트웨어를 실행하는 방식, 결함을 검출함. **화이트박스, 블랙박스, 경험기반 테스트**
- 테스트 기법에 따른 분류 -> **화이트박스 테스트, 블랙박스 테스트**
    - 화이트박스 테스트 유형
        - 구결조 조변다 기제데
            1. 구문 커버리지
            2. 결정 커버리지
            3. 조건 커버리지
            4. 조건/결정 커버리지
            5. 변경 조건/결정 커버리지
            6. 다중 조건 커버리지
            7. 기본 경로 커버리지
            8. 제어 흐름 테스트
            9. 데이터 흐름 테스트
    - 블랙박스 테스트 유형
        - 동경결상 유분페원비
            1. 동등분할 테스트
            2. 경곗값 분석 테스트
            3. 결정 테이블 테스트
            4. 상태 전이 테스트
            5. 유스케이스 테스트
            6. 분류 트리 테스트
            7. 페어와이즈 테스트
            8. 원인-결과 그래프 테스트
            9. 비교 테스트
- 테스트 시각에 따른 분류 - **검증, 확인**
    - 검증
        - 소프트웨어 ***개발 과정을 테스트***
        - 올바른 제품 생산 여부 검증
        - ***개발자 혹은 시험자***의 시각
    - 확인
        - 소프트웨어 ***결과를 테스트***
        - 만들어진 제품이 제대로 동작하는지 테스트
        - ***사용자***의 시각으로 입증
- 테스트 목적에 따른 분류 - **회안성 구회병**
    - 회복 테스트

    - 안전 테스트
    - 성능 테스트
    - 구조 테스트
    - 회피 테스트
    - 병행 테스트

- 테스트 종류에 따른 분류 - **명구경**

    - 명세 기반 테스트
        - 동경결상 유분페원비
    - 구조 기반 테스트
        - 구결조조변다기제데
    - 경험 기반 테스트
        - 탐색적, 오류추정, 체크리스트, 특성 테스트

정적 테스트

정적 테스트는 리뷰와 정적 분석으로 분류.

- 리뷰
- 정적 분석 - 코딩 표준, 복잡도 측정, 자료 흐름 분석

동적 테스트

- 화이트박스 테스트 (구조 기반 테스트)

    - 각 응용 프로그램의 내부 구조, 동작 검사하는 소프트웨어 테스트.
    - aka) 구조 기반 테스트, 코드 기반 테스트, 로직 기반 테스트, 글래스 박스 테스트
    - 테스트 커버리지 개념
        - 프로그램의 테스트 수행 정도를 나타냄. 테스트 수행의 완벽성 측정 **기라코**
            - 기능 기반 커버리지
            - 라인 커버리지
            - 코드 커버리지
        - ***구결조 조변다 기제데***
        - 구문 커버리지 - 프로그램 내 모든 명령문을 적어도 한 번 수행함.
        - 결정 커버리지 - 결정 포인트 내의 전체 조건식이 적어도 한번은 T/F의 결과를 수행함.
        - 조건 커버리지 - 결정 포인트 내의 **개별** 조건식이 적어도 한번은 T/F의 결과를 수행함
        - 조건/결정 커버리지 - 조건 커버리지와 결정 커버리지를 최소한의 조합으로 달성함.
        - 변경 조건/결정 커버리지 - 각 개별 조건식이 다른 개별 조건식에 영향을 받지 않고, 전체 조건식의 결과에 독립적으로 영향을 주도록 함. 조건/결정 커버리지 향상.
        - 다중 조건 커버리지 - 결정 조건 내 모든 개별 조건식의 모든 가능한 조합으리 100% 보장함.
        - 기본 경로 커버리지 - **맥케이브 순환복잡도**를 기반으로 커버리지를 계산함.
            - 순환복잡도는 제어 흐름의 복잡한 정보를 정량적으로 표시하는 기법.
            - 복잡도 = 간선 수 - 노드 수 + 2
        - 제어 흐름 테스트 - 프로그램 제어 구조를 그래프 형태로 나타내 내부 로직을 테스트하는 기법.

- 블랙박스 테스트 (명세 기반 테스트)

    - 외부 사용자의 요구명세를 보면서 수행하는 기능 테스트.
    - 전체 소프트웨어 테스트 레벨에 적용가능한 테스트 기법이다.
    - ***동경결상 유분페원비***
    - 동등분할 테스트 - 입력 데이터의 영역을 유사한 도메인별로 그룹핑, 대표값 테스트 케이스를 도출해냄
    - 경곗값 분석 테스트 - 등가 분할 후 오류 발생 확률 높은 경곗값 부분에서 테스트
    - 결정 테이블 테스트 - 요구사항의 논리와 발생조건을 테이블 형태로 나열, 조합하는 테스트
    - 상태 전이 테스트 - 쌍태를 구분, 이벤트에 의해 어느 한 상태에서 다른 상태로 전이되는 경우의 수를 수행
    - 유스케이스 테스트 - 시스템이 실제 사용되는 유스케이스로 모델링 되어있을 때 수행하는 테스트
    - 분류 트리 테스트 - sw의 일부 ,전체를 트리 구조로 분석, 표현하여 설계하는 테스트 기법
    - 페어와이즈 테스트 - 데이터들 간 최소 한번씩 조합하는 방식. 상대적으로 적은 양의 테스트 세트 구성 위함

- 경험 기반 테스트

    - 유사 sw, 유사 기술 평가에서의 테스트 경험을 토대로 함. **탐색적 테스트, 오류추정, 체크리스트, 특성 테스트** 존재
        - ***탐오체특***
        - 탐색적 테스트 - 테케를 문서로 작성 않고 경험에 바탕 두고 탐색적으로 기능을 수행해 보는 테스트 기법
        - 오류 추정 - 실수를 추정하고 결함이 검출되도록 설계
        - 체크리스트 - 평가해야할 내용을 분류, 나열 후 하나씩 확인함
        - 특성테스트 - 국제 표준을 따라 이를 근간으로 테스트케이스 설계

테스트 케이스

특정 요구사항에 준수하는 지 확인하기 위해 개발된 입력값, 조건, 예상 결과의 집합.

테스트 오라클

테스트의 결과가 참인지 거짓인지를 판단하기 위해 사전에 정의된 참값을 입력하여 비교하는 기법

- 참 오라클 - 모든 입력값에 대해 기대하는 결과 생성, 발견된 오류를 모두 검출 가능
- 샘플링 오라클 - 특정 몇 개 입력에 대해서만 기대 결과를 제공
- 휴리스틱 오라클 - 샘플링 개선, 특정 입력에 대해 올바른 결과, 나머지에 대해서는 휴리스틱(추정)으로 처리
- 일관성 검사 오라클 - 앱 변경 시 수행 전, 후의 결과가 동일한지 확인함.

애플리케이션 테스트 시나리오 작성

테스트 레벨

함께 편성되고 관리되는 테스트 활동의 그룹, 프로젝트에서 책임과 연관이 있음.

- 테스트 레벨 종류
    - 개발 단계에 따라 분류 가능, ***단통시인***
    - **단**위 테스트
    - **통**합 테스트
    - **시**스템 테스트
    - **인**수 테스트

---

chap2. 애플리케이션 통합 테스트

애플리케이션 통합 테스트

단위 테스트 - 개별적인 모듈(컴포넌트)를 테스트. 구현 단계에서 각 모듈을 구현한 후 수행.

- 목(Mock) 객체 생성 프레임워크
  - 객체지향 프로그램에서 컴포넌트 테스트 수행 시 테스트 되는 메서드는 다른 클래스의 객체에 의존함.
  - 이럴 때 메서드를 고립화 해 테스트가 불가능하기에 스텁의 객체지향 버전인 목 객체 필요.
- 목 객체 유형 ***더스드 스가***
  - 더미 객체 > 테스트할 때 객체만 필요, 해당 객체의 기능까지는 필요없을 때
  - 테스트 스텁 > 제어 모듈이 호출하는 타 모듈의 기능을 단순히 수행만 함
  - 테스트 드라이버 > 테스트 대상 하위 모듈 호출, 파라미터 전달, 모듈 테스트 수행 후 결과 도출
  - 테스트 스파이 > 테스트 대상 클래스와 협력하는 클래스로 가는 출력을 검증
  - 가짜 객체 > 협력 클래스의 기능을 대체해야 할 경우 사용, 실제 기능의 일부를 훨씬 단순하게 구현.

통합테스트

애플리케이션 통합 테스트는 소프트웨어 각 모듈 간의 인터페이스 관련 오류 및 결함을 찾아내기 위한 체계적 테스트 기법.

- 하향식 통합 : 메인 제어 모듈로부터 아래 방향으로 이동하며 하향식으로 통합하며 테스트 진행.
  - 깊이-우선, 너비-우선 방식
- 상향식 통합 : 애플리케이션 구조에서 최하위 레벨의 모듈, 컴포넌트 로부터 위쪽 방향으로 제어의 경로를 따라 이동, 테스트
- 샌드위치 통합 : 상햐익 + 하향식 통합 테스트 방식 결합. 하위 프로젝트가 있는 큰 규모 통합 테스트에서 사용.

하스 상드

하향식 - 스텁, 상향식 - 드라이버

테스트 방안	빅뱅 테스트	상향식	하향식	샌드위치 테스트
테스트 수행 방법	모두 통합후 테스트	최하위 모듈부터 상위로	최상위 모듈부터 하위 모듈로	상위는 하향, 하위는 상향
드라이버/스텁	실제 모듈로 테스트	테스트 드라이버	스텁	드라이버, 스텁 모두
장점	단시간 테스트, 작은 시스템	장애 위치 파악 용이	장애 위치 파악 용이	병렬 테스트 가능, 시간 절약 가능
단점	장애 위치 파악 어려움.	중요 모듈들이 마지막 테스트 가능성 높음	많은 스텁 필요	많은 비용 소요

테스트 자동화 도구 테스트 시간 단축, 인력 비용 최소화, 쉽고 효율적인 테스트

- 테스트 자동화 도구 유형
  - **정실성통** 정적, 실행, 성능, 통제
- 정적 분석 도구 - 애플리케이션을 실행하지 않고 분석함

- 테스트 실행 도구 - 작성된 스크립트 실행,
    - 데이터 주도 접근 방식
    - 키워드 주도 접근 방식
- 성능 테스트 도구
- 테스트 통제 도구
- 테스트 하네스 -> 애플리케이션 컴포넌트 및 모듈을 테스트하는 환경의 일부분, 테스트를 지원하기 위한 코드와 데이터를 말함. 단위, 모듈 테스트에 사용하기 위해 작성.
- 테스트 하네스 구성요소 - ***드 스슈케 스목***
    - 테스트 드라이버
    - 테스트 스텁
    - 테스트 슈트
    - 테스트 케이스
    - 테스트 스크립트
    - 목 오브젝트

애플리케이션 테스트 결과 분석

- 테스트 결과 분석
    - 소프트웨어 결함 - **에러, 결함, 결점, 버그, 실패**
    - 테스트 리포팅 **정요품 결실**
        - 테스트 결과 정리
        - 테스트 요약 문서
        - 품질 상태
        - 테스트 결과서
        - 테스트 실행 절차 리뷰 및 평가
- 결함 관리
    - 결함 관리 프로세스 ***계기검수 재추최***
        - 결함 관리 계획
        - 결함 기록
        - 결함 검토
        - 결함 수정
        - 결함 재확인
        - 결함 상태 추적 및 모니터링
        - 최종 결함 분석 및 보고서
    - 결함 분석 방법 ***구고일***
        - 구체화
        - 고립화
        - 일반화
- 애플리케이션 개선 조치사항 작성

        - 테스트 커버리지 - 주어진 테스트 케이스에 의해 수행되는 소프트웨어의 테스트 범위를 측정하는 테스트 품질 측정 기준. 테스트의 정확성과 신뢰성 향상
        - 테스트 커버리지 유형 ***기라코***
            - 기능 기반 커버리지
            - 라인 커버리지
            - 코드 커버리지
        - 결함의 유형 ***시기지문***
            - 시스템 결함
            - 기능 결함
            - GUI 결함
            - 문서 결함
        - 결함의 심각도별 분류 ***치주 보경단***
            - 치명적 결함
            - 주요 결함
            - 보통 결함
            - 경미한 결함
            - 단순 결함
        - 결함 우선순위 ***결높보낮***
            - 결정적
            - 높음
            - 보통
            - 낮음

---

chap3. 애플리케이션 성능 개선

애플리케이션 성능 분석

- 애플리케이션 성능 측정 지표 ***처응경자***
    - 처리량
    - 응답시간
    - 경과 시간
    - 자원 사용률

애플리케이션 성능 개선

소스코드 최적화

- 배드 코드 (Bad Code)
    - 외계인 코드
    - 스파게티 코드
    - 알 수 없는 변수명
    - 로직 중복
- 배드 코드 유형 ***오문이 결침***
    - 오염
    - 문서부족
    - 의미없는 이름

    - 높은 결합도
    - 아키텍처 침식
- 클린 코드 작성 원칙 ***가단의 중추***
    - 가독성
    - 단순성
    - 의존성 최소
    - 중복성 제거
    - 추상화
- 소스 코드 품질 분석 - 코드 내 존재하는 메모리 누수, 스레드 결함 등을 발견하기 위한 활동.
    - 정적, 동적 분석 도구 존재.
    - 정적 분석 도구 > 작성된 소스 코드를 직접 실행시키지 않고,코드 자체만으로 코딩 표준 준수 여부 등 확인
        - pmd, cppcheck, SonarQube, checkstyle, ccm, cobertura
    - 동적 분석 도구 > 애플리케이션 실행, 코드에 존재하는 메모리 누수 현황 발건함.
        - Avalanche, Valgrind
- 리팩토링을 통한 성능 개선
    - 리팩토링 : 유지보수 생산성 향상을 위해 기능을 변경하지 않고, 복잡한 소스코드 수정, 보완하여 가용성, 가독성을 높이는 기법.
    - 목적 : 유지보수성 향상, 유연한 시스템, 생산성 향상, 품질 향상