**애플리케이션 테스트**

**애플리케이션에 잠재되어 있는 결함을 찾아내는 일련의 행위 또는 절차**

**1. 애플리케이션 테스트의 기본 원리**

* **완벽한 테스트 불가능**
  + 테스트를 통해 결함을 줄일 수 있지만 결함이 없다고 증명할 수 없다.
* **파레토 법칙**
  + 애플리케이션의 20%의 코드에서 전체 결함의 80%가 발견된다는 법칙
* **살충제 패러독스**
  + 동일한 테스트 케이스를 반복하면 더 이상 결함이 발견되지 않는 현상
* **테스팅은 정황에 의존**
  + 소프트웨어 특징, 테스트의 환경, 테스터의 역량등에 따라 결과가 달라질 수 있다.
  + 즉, 정황에 따라 테스트를 다르게 수행해야 된다.
* **오류-부재의 궤변**  
  + 결함이 없는 소프트웨어여도 사용자의 요구사항 만족도를 충족하지 못한다면 품질이 좋다고 할 수 없다.
* **테스트와 위험은 반비례**
  + 테스트를 많이 하면 할수록 미래의 발생할 위험을 줄일 수 있다.
* **테스트의 점진적 확대**
  + 테스트는 작은 부분에서 시작하여 점점 확대하며 진행해야 한다.
* **테스트의 별도 팀 수행**
  + 테스트는 개발자와 관계없는 별도의 팀에서 수행해야 한다.

**2. 애플리케이션 테스트의 분류**

**2.1 프로그램 실행 여부에 따른 뷴류**

* **정적 테스트**
  + 명세서나 소스 코드를 대상으로 하는 테스트
  + 워크스루, 인스펙션, 코드검사등
* **동적 테스트**
  + 프로그램을 실행하여 오류를 찾는 테스트
  + 소프트웨어 개발의 모든 단계에서 테스트를 수행한다.
  + 블랙박스 테스트, 화이트박스 테스트

**2.1.1 화이트박스 테스트 - White Box Test**

**원시 코드의 논리적인 모든 경로를 테스트하여 테스트 케이스를 설계하는 방법**

**2.1.1.1 종류**

* **기초 경로 검사 (Base Path Testing)**
  + 대표적인 화이트박스 테스트 기법
  + 절차적 설계의 논리적 복잡성을 측정할 수 있게 해주는 테스트 기법
* **제어 구조 검사 (Control Structure Testing)**  
  + **3가지의 검사 방법으로 구성됨**
    - 조건 검사 : 프로그램 모듈 내에 있는 논리적 조건을 설계
    - 루프 검사 : 프로그램의 반복 구조에 초점을 맞춰  설계
    - 데이터 흐름 검사 : 프로그램의 변수 정의와 변수 사용 위치에 초점을 맞춰 설계

**2.1.1.2 검증 기준**

* **문장 검증 기준**  
  + 소스코드의 모든 구문이 한번 이상 수행하도록 설계되었는가
* **분기 검증 기준**  
  + 소스코드의 모든 조건문의 전체조건식의 결과가 True와 False를 한번이상 수행하도록 설계되었는가
* **조건 검증 기준**  
  + 소스코드의 모든 조건문의 개별조건식의 결과가 True와 False를 한번이상 수행하도록 설계되었는가
* **분기/조건 기준**
  + 분기 및 조건 검증 기준을 모두 만족하는 설계

**2.1.2 블랙박스 테스트 - Black Box Test**

**각 기능이 완전히 작동되는 것을 입증하는 테스트 (기능 테스트라고도 한다.)**

**2.1.2.1 종류**

* **동치 분할 검사 / 동치 클래스 분해 (Equivalence Partitioning Testing)**  
  + 동등 분할 기법이라고도 한다.
  + 프로그램의 입력조건에 타당한 입력자료와 타당하지 않은 입력자료의 개수를 균등하게 하여 테스트
* **경계값 분석 (Boundary Value Analysis)**  
  + 입력 조건이 경계값에서 오류가 발생될 확률이 높다는 점을 이용한 테스트 케이스 기법
  + 경계값을 테스트
* **원인-효과 그래프 검사 (Cause-Effect Graphing Testing)**  
  + 입력과 출력 간의 영향도를 체계적으로 분석한 다음 알맞은 테스트 케이스를 선정하는 기법
* **오류 예측 검사 (Error Guessing)**
  + 과거의 경험이나 확인자의 감각으로 테스트하는 기법
* **비교 검사 (Comparison Testing)**
  + 여러 버전의 프로그램에 동일한 테스트 자료를 제공해 결과를 비교하는 테스트 기법

**2.2 테스트 기반에 따른 분류**

* **명세 기반 테스트**
  + 사용자의 요구사항에 대한 명세를 기반으로 테스트 케이스를 구현하고 확인하는 테스트
  + 동등 분할, 경계 값 분석 등
* **구조 기반 테스트**
  + 소프트웨어 내부 논리 흐름에 따라 테스트 케이스를 작성하고 확인하는 테스트
  + 구문 기반, 결정 기반, 조건 기반 등
* **경험 기반 테스트**
  + 유사 소프트웨어 또는 기술 등에 대한 테스터의 경험을 기반으로 수행하는 테스트
  + 요구사항 명세가 불충분하거나, 테스트 시간에 제약이 있는 경우 주로 사용
  + 에러 추정, 체크 리스트, 탐색적 테스팅등

**2.3 시각에 따른 분류**

* **검증(Verification) 테스트**
  + 개발자의 시각에서 제품의 생산 과정을 테스트
  + 명세서대로 완성이 되었는가를 확인하는 테스트
* **확인(Validation) 테스트**
  + 사용자의 시각에서 생산된 제품의 결과를 테스트
  + 요구대로 제품이 완성되었는지, 정상적으로 동작하는가를 확인하는 테스트

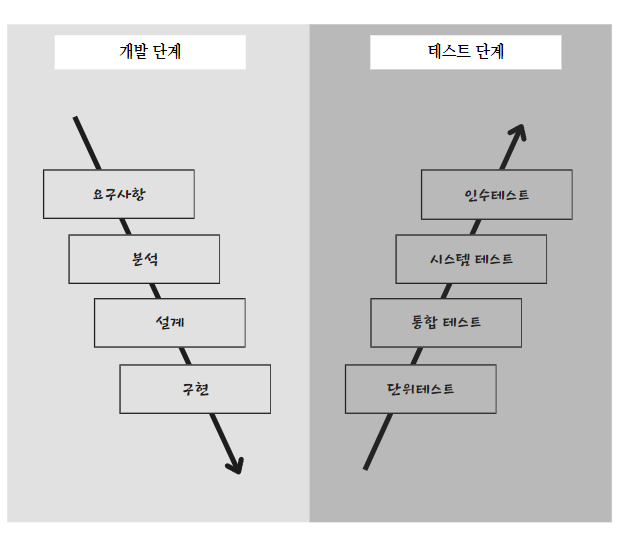
**2.4 목적에 따른 분류**

* **회복(Recovery) 테스트**  
  + 시스템에 결함이 발생 시 올바르게 복구되는지 확인하는 테스트
* **안전(Security) 테스트**
  + 불법적인 침입으로부터 보호할 수 있는지 확인하는 테스트
* **강도(Stress) 테스트**
  + 과부하 동작시에 소프트웨어가 정상적으로 동작하는지 확인하는 테스트
* **성능(Performance) 테스트**
  + 성능과 효율성을 진단하기 위해 응답시간, 처리량등을 확인하는 테스트
* **구조(Structure) 테스트**
  + 논리적인 경로, 소스 코드의 복잡도 등을 평가하는 테스트
* **회귀(Regression) 테스트**  
  + 소프트웨어의 변경 또는 수정 부분에 결함이 없음을 확인하는 테스트
* **병행(Parallel) 테스트**  
  + 변경된 소프트웨어와 기존 소프트웨어와 동일한 입력 시 결과를 비교하는 테스트

**개발 단계에 따른 애플리케이션 테스트**

**1. 소프트웨어 V-모델**

**애플리케이션 테스트와 소프트웨어 개발 단계를 연결하여 표현한 것을 V-모델**



**2. 단위 테스트 - Unit Test**

**모듈이나 컴포넌트에 초점을 맞춰 테스트하는 것**

**3. 통합 테스트 - Integration Test**

**단위 테스트 이후 모듈들을 결합하여 하나의 시스템으로 완성시키는 과정에서의 테스트**

**3.1 종류**

* **비점진적 통합 방식**  
  + 빅뱅 통합 테스트 방식
* **점진적 통합 방식**
  + **하향식 통합 테스트 ( Top Down Integration Test )**
    - **상위 모듈에서 하위 모듈 발향으로 통합하면서 테스트 (스텁 필요)**
  + **상향식 통합 테스트 ( Bottom Up Integration Test )**
    - **하위 모듈에서 상위 모듈 방향으로 통합하는 테스트 (드라이버 필요)**
  + 혼합식 통합 테스트
    - 하위 수준에서는 상향식 통합, 상위 수준에서는 하향식 통합을 하는 방식
    - 샌드위치식 통합 테스트라고도 합니다.

**4. 시스템 테스트 - System Test**

**개발된 소프트웨어가 컴퓨터에서 완벽하게 수행되는가를 점검하는 테스트**

**5. 인수 테스트 - Acceptance Test**

**사용자의 요구사항을 충족하는지 중점을 둔 테스트**

* 종류
  + 사용자 인수 테스트 - 사용자가 시스템의 적절성을 확인
  + 운영상의 인수 테스트 - 시스템 관리자가 인수 시 수행하는 테스트
  + 계약 인수 테스트 - 계약상의 인수/검수 조건 준수 확인 테스트
  + 규정 인수 테스트 - 소프트웨어가 법규, 규정 등에 맞게 개발되었는지 확인 테스트
  + 알파 테스트 - 통제된 환경에서 확인 테스트
  + 베타 테스트 - 선정된 사용자가 여려 명의 사용자 앞에서 행하는 테스트

**테스트 케이스 / 테스트 시나리오 /  테스트 오라클**

* **테스트 케이스 - 사용자의 요구사항을 정확하게 준수했는지를 확인하기 위해 설계된 테스트 항목 명세서**
* **테스트 시나리오 - 테스트 케이스를 묶은 집합**
* **테스트 오라클 - 테스트 결과가 올바른지 판단하기 위해 사전에 정의된 참 값을 대입하여 비교하는 기법**

**1. 테스트 오라클의 특징**

* 제한된 검증 - 모든 테스트 케이스에 테스트 오라클을 적용할 수 없다.
* 수학적 기법 - 테스트 오라클의 값을 수학적 기법을 이용하여 구할 수 있다.
* 자동화 기능 - 자동화할 수 있다.

**2. 테스트 오라클의 종류**

* 참 오라클 - 모든 테스트 케이스 입력값에 대해 기대하는 결과를 제공하는 오라클
* 샘플링 오라클 - 몇몇 테스트 케이스에 대해서만 결과를 제공하는 오라클
* 추정 오라클 - 몇몇 테스트 케이스에서만 결과를 제공하고 나머지는 추정하는 오라클
* 일관성 검사 오라클 - 애플리케이션 변경이 있을 때 테스트 케이스의 수행이 전후 결과 값이 동일한지 확인하는 오라클