Part I. 소프트웨어 테스팅의 기초

I. 테스팅의 필요성

- 소프트웨어가 올바르게 동작하지 않는 경우 다양한 문제가 발생.
  -> 금전적, 시간적 손실, 시간 낭비, 비즈니스의 이미지 손상, 부상이나 사망
  -> 테스팅은 이러한소프트웨어 시스템의 문제를 최소화하기 위해 필요
- 개발자가 작성할때 결함(Defects)를 만드는 오류를 범할 수 있고 이러한 결함이 장애(Failure)의 원인이 된다.
- 결함의 발생 이유는 시간적 압박, 복잡한 코드, 기반환경의 복잡성, 기술이나 시스템의 변경, 시스템 상호간의 연동 등의 기술적인 이유 + 환경적인 조건

- 단위 테스팅과 통합 테스팅은 개발조직이 중심이 되어 수행, 시스템이 갖춰진 이후의 테스팅은 독립성을 가진 테스트 조직이 수행하는 것이 바람직하다
- 테스트 레벨에 따른 테스팅은 품질을 높이고 결함이 발생할 가능성을 최소화한다.
- 운영 테스팅 = 기존에 운영하던 소프트웨어 시스템이 변경 및 단종되거나 환경이 변했을 경우 필요

- 테스팅으로 품질에 대한 확신과 전반적인 리스크 수준을 감소시킬수있다
- 품질을 높이기 위해서는 이전 프로젝트를 통해 많은 테스트 경험과 정보를 확보하여야 한다

- 적절한 테스팅의 정도를 파악하기 위한 제약 사항
  -> 리스크 수준, 기술적인 내용, 비즈니스, 제품, 프로젝트 리스크, 시간, 비용

II. 테스팅이란 무엇인가

- 테스팅이란 동작과 성능, 안정성이 사용자가 요구하는 수준을 만족하는지 확인하기 위해 결함을 발견하는 것
- 전통적인 테스트 개념 = 시스템의 정상 작동 여부 확인
- 현재의 테스팅 = 사용자의 기대 수준과 요구사항에 맞게 구현되고 동작하는지 확인하고 이를 통한 결함 발견하여 이 리스크를 의사결정권자에게 전달
- 테스팅은 동적 테스팅뿐 아니라 문서의 리뷰와 정적 분석과 같은 일련의 활동들을 포함한다.

- 테스팅의 목적:
  -> 남아있는 결함 발견, 명세 충족 확인, 사용자의 요구 충족 확인, 결함 예방
  -> 품질 수준에 대한 자신감 획득, 정보에 근거한 조언 제공, 개발 프로세스 점검과 이슈 제기, 논리적 설계의 구현 검증 (validate), 시스템의 동작 확인

- 관점에 따른 테스팅 목적
  -> 개발과정에서의 테스팅 = 결함을 찾아내고 가능한 많은 장애 상황을 만들어 내는 것
  -> 인수 테스팅 = 예상된 대로 시스템이 동작하는지 확인하고 요구사항에 맞는지 확신을 얻는 과정
  -> 소프트웨어의 품질을 평가하기 위한 테스팅 = 시스템을 출시하는 것의 리스크를 개발 프로젝트 관련자에게 전달
  -> 유지보수 테스팅 = 변경 작업이 일어나는 경우 새로운 결함이 유입되었는지 확인
  -> 운영 테스팅 = 신뢰성 또는 가용성과 같은 시스템의 특성을 평가
- 테스팅과 디버깅은 구분되는 개념.
  -> 테스팅 = 결함을 발견하기 위한 활동 <——> 디버깅 = 결함의 원인을 밝히고 코드를 수정하는 개발 활동

III. 테스팅의 일반적인 원리

- 원리 1. 테스팅은 결함이 존재함을 밝히는 활동이다.
  -> 테스팅은 잠재적으로 존재하는 결함을 줄일 수는 있지만, 결함이 발견되지 않은 경우라도 결함이 없다고 증명할 수는 없다.

- 원리 2. 완벽한 테스팅은 불가능하다.
  -> 모든 가능성을 테스팅하는 것은 가능하지 않다.
  -> 무수히 많은 내부조건 (무한 경로), 무수히 많은 입력 값 (무한 입력값), 무수히 많은 이벤트 발생 순서 (무한 타이밍)

- 원리 3. 테스팅을 개발 초기에 시작한다.
  -> 테스트 활동은 개발 수명주기에서 가능한 초기에 시작 되어야 한다
  -> 조기 테스트 설계를 하면 코딩이 끝나자마자 준비된 테스트 케이스를 레벨별로 시행할 수 있고
  -> 결과를 단기간에 알 수 있어 테스팅 기간을 단축 가능하고 후반에 발견될 결함을 예방할 수 있다.

- 원리 4. 결함 집중
  -> 출시 전 대다수의 결함들은 소수의 특정 모듈에 집중되어 발생하는 경향
  -> 이러한 모듈의 특징 - 복잡한 구조, 다른 모듈과 복잡한 상호 작용, 개발 난이도가 높거나 최신기술 사용, 새롭게 개발, 크기가 큼, 경험이 미흡한 개발팀

- 원리 5. 살충제 패러독스
  -> 시스템에 잠재된 보다 많은 결함을 발견하기 위해서는 테스트 케이스를 정기적으로 리뷰하고 개선할 필요가 있음

- 원리 6. 테스팅은 정황에 의존적이다.
  -> 테스팅은 정황에 따라 다르게 진행된다. 정황과 도메인(분야)에 따라 다르게 테스팅을 진행해야 한다.
  -> 변하지 않는 사항) 주요활동에 대한 테스트 프로젝트 계획, 표준적인 기법 적용, 독립적 테스트 환경, 효율적 테스트 팀 조직, 정식 리포팅 등

- 원리 7. 오류-부재의 궤변(Absence of error fallacy)
  -> 사용자 또는 비즈니스의 요구를 충족시켜주지 못한다면 결함을 모두 발견하여 제거하였다고 하더라도 품질이 높다고 볼 수 없다.

- 결론
  -> 소프트웨어 테스팅은 결함이 존재함을 밝히고 사용자의 요구사항에 따라 대상 제품이 개발되었는지 확인.
  -> 리스크 기반으로 결함이 집중되어 있을 만한 곳을 예측하여 가능한 개발 초기부터 테스트를 수행
  -> 테스트 대상 제품의 특성과 요구사항 등 테스트 정황을 반영하여 설계하고 도출한 테스트 케이스는 보완과 추가를 통하여 더 많은 결함을 발견할 수 있도록 한다.

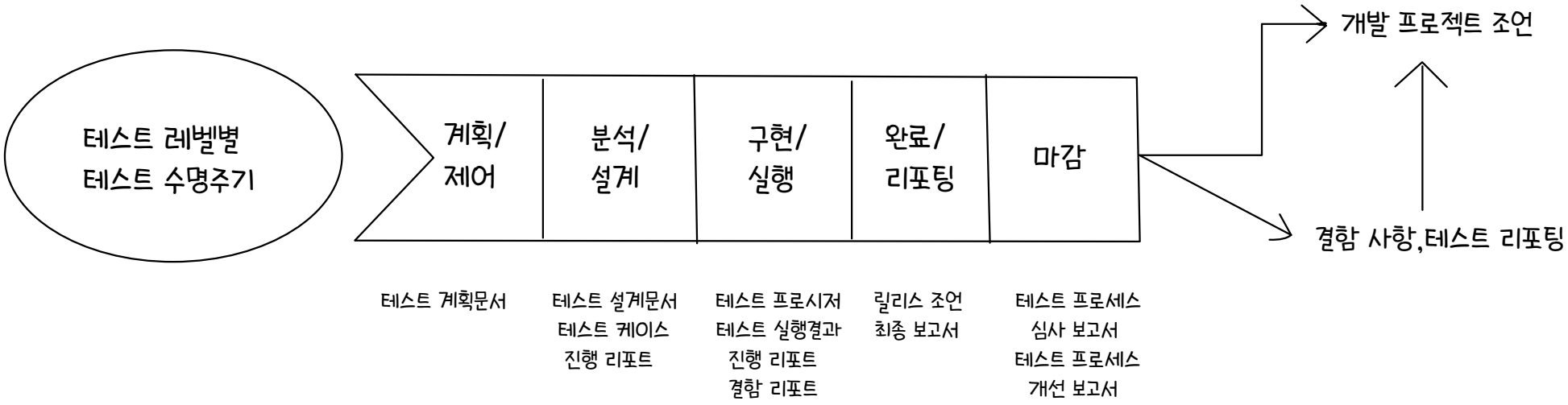
IV. 테스트 프로세스의 기초

- 기본적인 테스트 프로세스 주요 활동들
  -> 계획과 제어 (Planning and control)
  -> 분석과 설계 (Analysis and design)
  -> 구현과 실행 (Implementation and execution)
  -> 완료 조건의 평가와 리포팅 (Evaluating exit criteria and reporting)
  -> 테스트 마감 활동 (Test closure activities)

- 테스트 프로세스의 주요 활동은 논리적으로는 순차적이지만, 프로세스 내의 활동들은 중복되거나 동시에 발생할 수 있다.
- 테스트 프로세스는 다양한 활동이 체계적으로 진행되어 의도된 목적과 목표를 달성할 수 있도록 모든 구성 요소를 엮어주는 역할을 한다.

- 테스트 베이시스 (Test basis)는 테스트 설계 및 구현의 기반이 되어 기획 단계에서부터 필요하고 설계 시에는 반드시 요구된다.
- 테스트 조직은 계획 단계에서부터 필요하며 테스트 실행 단계에서는 상대적으로 다수의 인력이 필요

- 테스팅은 단순히 결함을 발견하기 위해 테스트를 진행하는 것만이 아닌, 체계적으로 발견한 결함과 관련 정보를 바탕으로 정량적으로 개발 프로젝트에 조언을 제공
- 이때 조언은 테스트 전략을 수립하기 위해 분석한 리스크를 반영하여 제공하는 것이 바람직



IV - i. 테스트 계획과 제어(통제)

- 테스트 계획 수립은 테스트 목표와 임무를 달성하기 위해 이를 확인하고 필요한 활동을 뜻 함
- 테스트 계획 수립 관련 주요 작업의 내용:
  -> 테스트 범위와 테스트를 위한 리스크에 대한 결정 + 테스트 목적에 대한 식별
  * 테스트 범위: 다른 시스템과의 인터페이스 정도, 품질 특성, 호환성(compatibility), 커버하고자 하는 테스트 레벨
  * 리스크 기반 테스트 전략: 테스트 대상이 충족해야할 품질 수준 및 특성과 기술적 어려움, 비즈니스 리스크를 고려한 테스트 전략 수립
  * 테스팅 목적: 품질 요구 수준, 테스트 레벨 별 목적, 커버하고자 하는 품질 특성
  -> 테스트 정책의 실현과 테스트 전략의 구현
  -> 테스트 접근 방법에 대한 결정 (테스트 기법 및 대상 테스트 베이시스 포함여부, 테스트 아이템, 팀간 의사소통, 테스트웨어 등)
  -> 테스트에 필요한 리소스의 결정 (조직, 인원, 테스트 환경 등)
  -> 테스트 분석과 설계 작업의 일정관리
  -> 테스트 구현, 실행 및 평가의 일정관리
  -> 테스트 완료 조건의 결정

- 테스트 제어는 계획 대비 실제 진행 상황을 비교하는 지속적인 활동
- 테스트 프로젝트의 목표 및 임무를 달성하기 위해 계획과의 차이에 대해 보고하고 조치를 취하는것
- 테스팅을 제어하기 위해서는 프로젝트 내내 테스팅 활동을 모니터링 해야 함 ==> 피드백은 테스트 계획에 반영
- 테스트 제어의 주요한 작업의 내용:
  -> 테스트 결과에 대한 측정과 분석
  -> 테스트 진척 상황, 테스트 커버리지와 완료 조건의 모니터링과 문서화
  -> 계획과의 차이를 교정하는 활동
  -> 테스팅의 진행과 변경에 대한 의사 결정 활동

IV - ii. 테스트 분석과 설계

- 일반적이고 추상적인 테스팅 목적을 실제적이고 구체적인 테스트 상황과 테스트 케이스로 변환하는 활동
- 주요한 작업:
  -> 테스트 베이시스 리뷰
  -> 테스트 대상의 분석을 통해 테스트 상황을 식별하고 우선 순위 결정
  -> 테스트 케이스 설계와 우선순위 선정
    * 공식적인 테스트 기법을 활용하여 테스트 케이스 도출
    * 비공식적인 테스트 기법으로 테스트 케이스 추가 도출 및 보안
  -> 테스트 케이스에 필요한 테스트 데이터 식별
  -> 테스트 환경 구축에 대한 디자인과 기반 시설 및 도구의 식별

IV - iii. 테스트 구현과 실행

- 효율적이고 효과적으로 테스트를 실행하기 위하여 테스트 케이스를 조합하고 필요한 다른 정보를 포함하는 테스트 프로시저 또는 테스트 스크립트를 명세화하는 활동
- 테스트 실행에 필요한 테스트 환경이 구축되어 있어야 한다
- 주요한 작업:
  -> 테스트 케이스의 개발, 구현과 우선순위 선정
    * 테스트 프로시저의 작성과 테스트 데이터의 생성
  -> 자동화 테스트 스크립트 작성
  -> 테스트 하네스 준비
  -> 효율적인 테스트 실행을 위해 테스트 수트 (테스트 케이스 묶음) 생성
  -> 테스트 환경의 올바른 구축 여부 확인
  -> 계획된 순서에 의거하여 준비된 테스트 프로시저를 수행
  -> 테스트 수행 결과를 기록
  -> 예상 결과와 실제 결과를 비교 => 결과간의 차이에서 오는 불일치를 인시던트 또는 결함으로 보고
  -> 불일치의 원인을 알아내기 위해 결과나 현상을 분석
    * 테스트 케이스 결함 == 테스트 케이스 자체의 결함 때문에 생긴 경우 (스크립트, 시나리오, 잘못된 데이터 또는 입력값)
    * 테스트 정황 결함 == 테스트 수행 상의 오류, 테스트 데이터 입력 오류
    * 어플리케이션 결함 == 소프트웨어/하드웨어 결함, 진행 절차상의 결함, 설치상의 결함, 코드상의 결함
    * 어플리케이션 정황 결함 == 사용상의 오류, 테스트 환경과 관련된 소프트웨어 결함
- 조치한 결과를 확인하기 위해 테스트 활동을 반복
    * 이전에 실패한 테스트를 진행 + 수정으로 인해 새로운 버그가 발생하지 않았는지 확인하는 테스트 진행
- 발견할 수 있는 결함은 품질 보증 절차를 개선하기 위해 유형별로 분류 가능
  -> 기획 시 유입된 결함
  -> 설계 시 유입된 결함
  -> 코딩 시 유입된 결함
  -> 테스트 부족으로 유입된 결함
  -> 마무리 부족
  -> 팀간 의사소통 부족
  -> 코딩 실수
- 결함 심각도 별로 분류하여 관리 필요
  -> 치명적 결함, 주요 결함, 일반 결함, 사소한 결함, 새겐 사항
- 결함 심각도 구분별로 해당하는 내용이 모호할 경우 결함 리포팅의 신뢰성이 떨어지고 사람마다 다르게 평가하여 논쟁에 불필요한 시간을 낭비할 수 있음

IV - iv. 테스트 완료 조건 (Exit criteria)과 리포팅

- 테스트 완료 조건은 초기에 정의된 테스트 목표에 비해 어느 정도 실제 테스트가 수행되었는지를 평가하는 활동
- 테스트 목표를 달성하였다면 테스트가 완료된다.
- 해당 활동은 각 테스트 레벨마다 수행되는 것이 원칙
- 주요한 작업:
  -> 테스트 실행 결과가 테스트 계획의 명시된 완료 조건을 만족하는지 확인
  -> 추가적인 테스트가 필요한지 아니면 명세된 테스트 완료 조건을 변경 해야 하는지에 대한 평가 수행
  -> 이해관계자에게 배포할 테스트 요약 보고서 작성

- 리포팅은 진행보고와 릴리즈 조언 및 최종보고 형태로 이루어지며 테스팅을 수행하면서 수집한 결함 및 테스트 진행 관련 데이터를 가공하여 수치화하고 관련 정보를 표현하는 작업

- 주요한 작업:
  -> 발견된 결함과 미해결 결함의 추이 및 우선순위
  -> 테스트 진척도
  -> 리스크 및 메트릭으로 실증된 조언
  -> 테스트 환경의 가용성
  -> 테스트 커버리지, 결함 발견 효율성, 품질 평가 결과, 결함 상태별 결함 수, 사이즈 대비 결함 수, 요구사항 별 테스트 일 수, 해결되지 않은 결함과 그 영향, 오랫동안 수정되지 않은 결함 분석 등등
- 진행보고 = 테스트 제어 활동 단계에서 테스트의 진행 상황을 모니터링하는 활동으로 분류
- 릴리즈 조언과 최종보고 = 테스트 완료 단계에서의 활동

IV - v. 테스트 마감 활동

- 완료된 테스트 활동에서 데이터를 수집하여 발견된 사실이나 수치적 데이터와 함께 테스트 경험과 테스트웨어를 종합하고 축적하는 활동
- 테스팅이 얼마나 체계적으로 수행되었는지 평가하고 테스팅을 개선하기 위해 모범 사례를 모델로 테스트 프로세스를 심사 하는 것
- 시스템이 출시되어 프로젝트가 완료되었을 때, 모든 마일스톤이 달성되었을 때, 유지보수 활동 중 추가 개발되거나 업데이트 된 부분이 출시 완료되었을 때 발생한다.
- 주요한 작업:
  -> 테스트 결과 마감 == 예정된 산출물 확인, 인시던트 리포트 종료, 해결되지 않은 요구 사항에 대한 처리, 시스템을 인수하는 것 문서화
- 테스트웨어, 테스트 환경, 테스트 기반설비를 차후에 사용할 것을 대비하여 마감하고 보관
- 테스트웨어를 유지보수 조직에 이관
- 테스트 프로세스 심사 및 개선 사항 제안
- 이후 개선의 지침이 될 수 있도록 테스트 프로젝트를 통해 얻은 교훈을 분석