Object Oriented Software Engineering OOSE

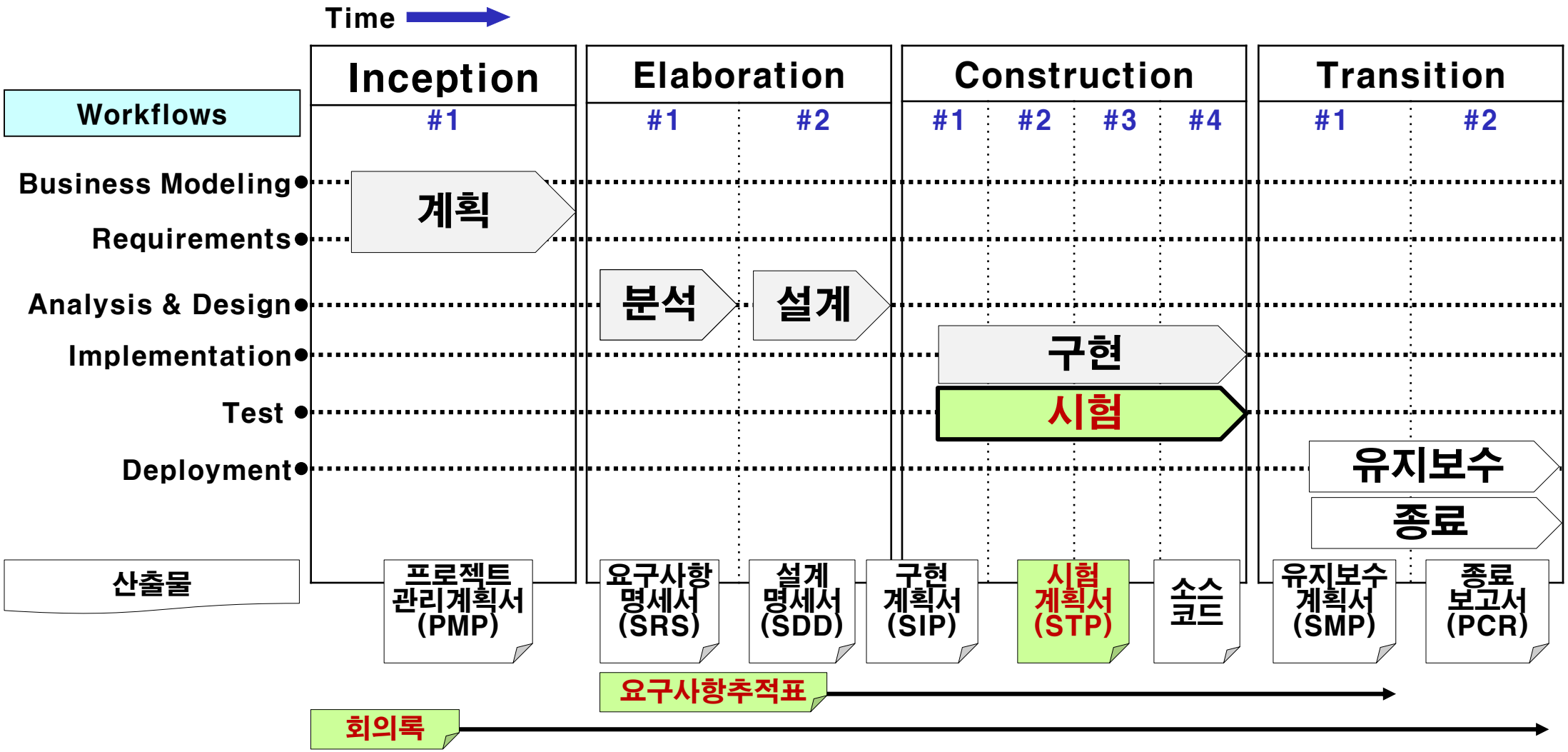
시험(Test)

- 테스트?
- 테스트 유형 및 절차
- 테스트 작업의 산출물(Artifacts)
- 테스트 작업의 수행자(Workers)
- 테스트 작업의 작업흐름(Workflows)

Computer & Software Engineering
Lee Jong Yeol
soyeum@kumoh.ac.kr

어디까지 왔나?

- 소프트웨어 개발 프로세스(SDLC, OO, UP, PMBOK)



What is Test?

- 테스트란?
  - s/w 의 품질을 평가하기 위한 모든 활동
    - s/w에 대한 결함(defects)을 발견하여 기록
    - s/w의 품질에 관하여 경영자에게 보고하는 활동
    - 요구기술서 및 설계 문서에 포함된 가정(assumption)에 대한 평가
    - s/w가 설계 문서에 부합되도록 구현되었는지 검증
    - 소프트웨어 요구가 시스템에 잘 반영되었는가를 확인

  - s/w의 품질
    ① 고객의 요구에 부합하는가?
    ② 본래의 사용 목적을 충족시키고 있는가?
    ③ 결함(defect)은 없는가?

테스트의 중요성

❑ 최근의 보고 자료
- 투자 대비 효과(ROI: Return On Investment)가 높음
  - 미국 국방성이 개발하는 시스템의 평균 ROI: 800%
- 테스트 비용: 개발비의 30%~50% 차지

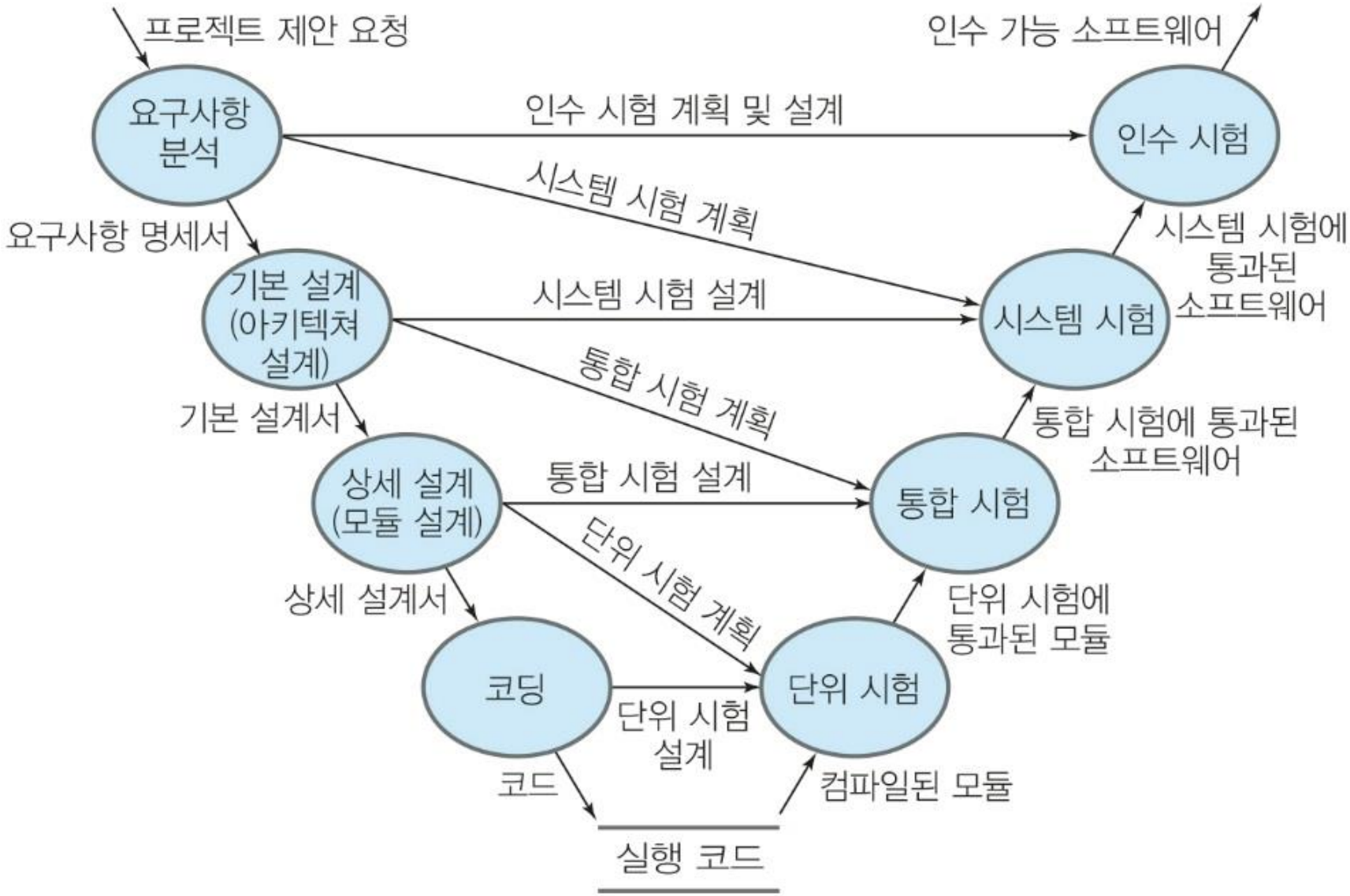
※ 테스트 = s/w의 품질 수준을 높이기 위한 가장 중요한 수단이자, 최후의 수단!!

❑ 테스트가 중요한 시스템
- 안전이 결정적인 시스템: 항공 관제 시스템, 미사일 유도 시스템, 의료 관련 시스템, …
- 경영정보시스템 -> 잘못 작동하는 경우에 고객에게 심각한 피해

테스트의 특성

- 테스트는 요구 추출, 분석, 설계, 구현과는 다름
  - 테스트: s/w 제품의 약점을 발견하는 것
    - 어떤 방법으로 시스템의 오작동을 유발시킬까?
    - 어떤 상황에서 시스템이 실패할 수 있을까?
  - 다른 작업: 완전성, 무결성, 정확성을 추구하는 작업
- 테스트를 충분히 하는 것은 지극히 어렵다!
  - 테스트 작업은 본질적으로 어려움
  - 명확한 방법론 부재 -> 개인의 역량에 의존
  - 자동화 도구 부족 -> 방대한 작업량을 감당하기 어려움
  - s/w의 복잡성 및 유연성 -> 완벽한 테스트는 거의 불가능

소프트웨어 개발 프로세스와 시험(V 모델)



테스트의 유형(1)

- 테스트의 목적에 따른 분류
  - 확인 테스트(validation test or conformance test)
    - 고객의 요구에 부합하는가?
    - 완성된 프로그램이 프로토콜, 표준, 계약조건 등을 만족하는가?
    - 확인하고자 하는 품질 요소나 요구에 따라 성능 테스트, 사용 편의성 테스트, 안전성 테스트 등으로 구분
  - 결함 테스트(defect test)
    - 결함을 검출할 목적으로 수행하는 테스트
    - 개별 단위 또는 개별 단위를 통합하는 과정에서 수행

테스트의 유형(2)

- 테스트 기반에 따른 분류
  - 명세서 기반 테스트(black box 테스트, 함수적 테스트)
  - 코드 기반 테스트(white box 테스트, 구조적 테스트)
  - 결함 기반 테스트
  - Usage Based 테스트(Scenario Based 테스트)

- 테스트 설계 기법에 따른 분류
  - 체계적 테스트: 검출하고자 하는 결함 유형을 먼저 결정하고, 그에 따라 테스트 케이스를 고안
  - 무작위 테스트: 무작위로 생성된 테스트 케이스 적용

테스트의 유형(3)

❑ 테스트 수준에 따른 분류

▪ 테스트 수준(test level): 테스트 대상이 되는 소프트웨어가 어떤 모습을 갖춘 상태인가를 표시

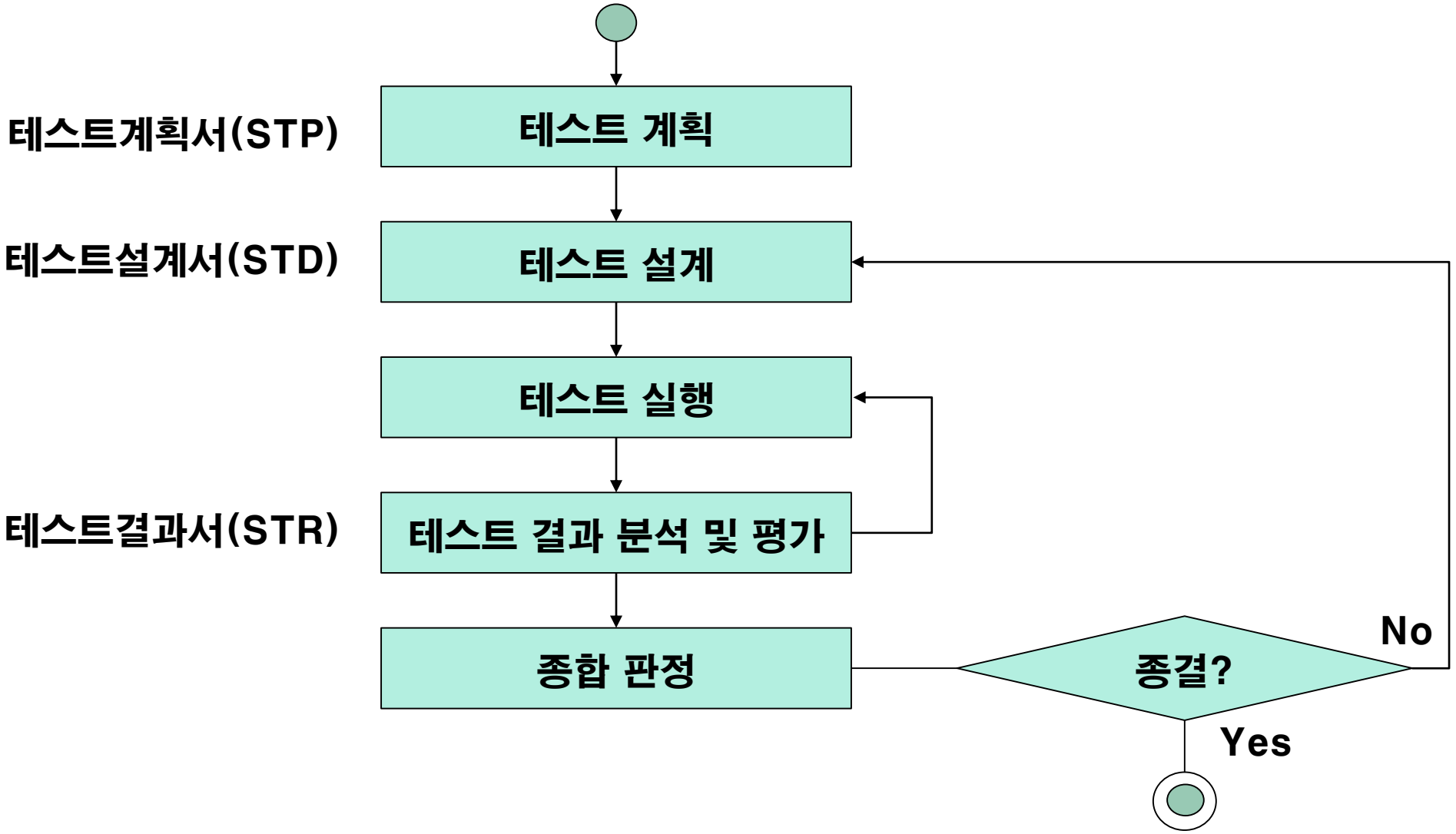
▪ 종류

- 단위 테스트(Unit Test, 모듈 테스트): 개별 모듈 차원
- 통합 테스트(Integration Test): 통합과정 자체를 테스트 대상으로 함
- 시스템 테스트(System Test): 전체 시스템을 대상으로 복구(Recovery), 보안(Security), 스트레스(Stress), 성능(Performance) 시험
- 인수 테스트(Acceptance Test) : 사용자 시험
- 설치 테스트(Installation Test)
- 회귀 테스트(Regression Test)

Regression Test

- 회귀 테스트란?
  - 실행 시기: 운영 중인 소프트웨어를 변경한 때
  - 검증 항목
    - 변경이 제대로 되었는가?
    - 변경 전의 기능은 여전히 올바르게 작동하는가?
    - 시스템 기능 중에 후퇴(regress)한 부분은 없는가?
  - 종류
    - Corrective Regression Test: 결함을 제거하기 위한 수정
    - Progressive Regression Test: 기능 향상을 위한 수정
  - 중요성
    - 최근의 추세: 신규 개발 -> 기존 시스템(Legacy System)의 수정 보완
    - 시스템의 변경에 따른 회귀 테스트는 점점 중요해 짐

테스트 절차

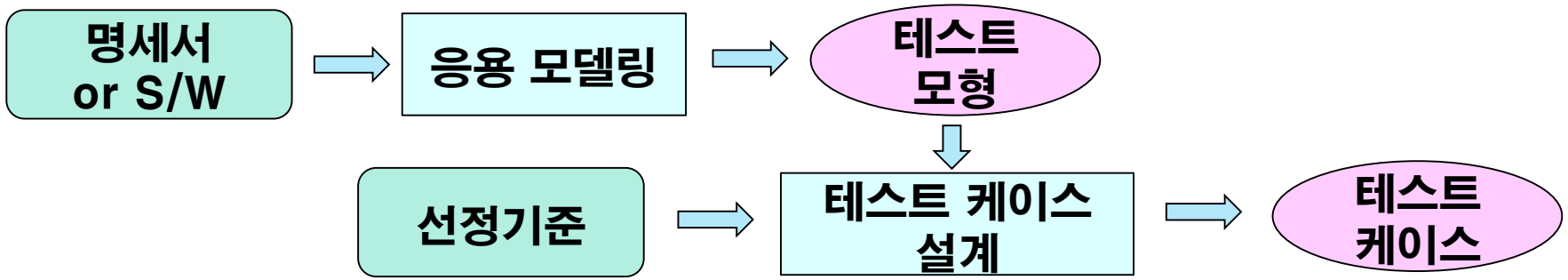

테스트계획서(STP)
테스트 계획
테스트설계서(STD)
테스트 설계
테스트 실행
테스트결과서(STR)
테스트 결과 분석 및 평가
종합 판정
종결?
No
Yes

테스트 설계

❑ 기본 개념

- 테스트 설계: 결함을 검출하기에 효과적인 테스트 케이스를 준비하는 작업
- 테스트 기반(Test Basis): 테스트 케이스를 구하기 위해서 참고하는 명세서나 코드 등의 자료
- 테스트 모형(Test Model): 테스트 대상을 이해하기 쉽게 표현한 것. 예)제어흐름 그래프

❑ 테스트 케이스 설계 절차

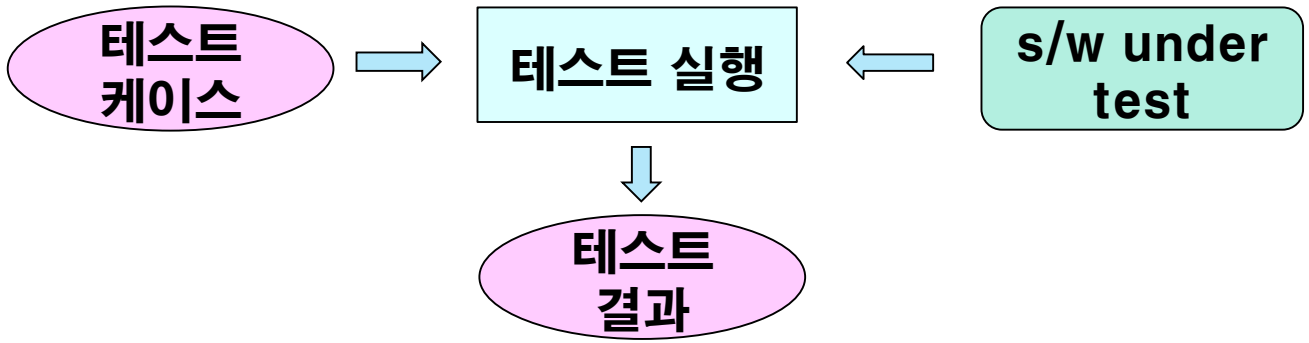


테스트 실행

❑ 기본 개념

- 테스트 실행: 준비한 테스트 케이스를 가지고, 컴퓨터에서 테스트 대상인 소프트웨어(SUT: s/w under test)를 실행시키는 작업
- 테스트 드라이버(Test Driver): 테스트 실행을 도와주는 자동화 도구
- 테스트 환경(Test Environment): 테스트 프로세스 전체를 자동화시킨 것

❑ 테스트 실행 절차

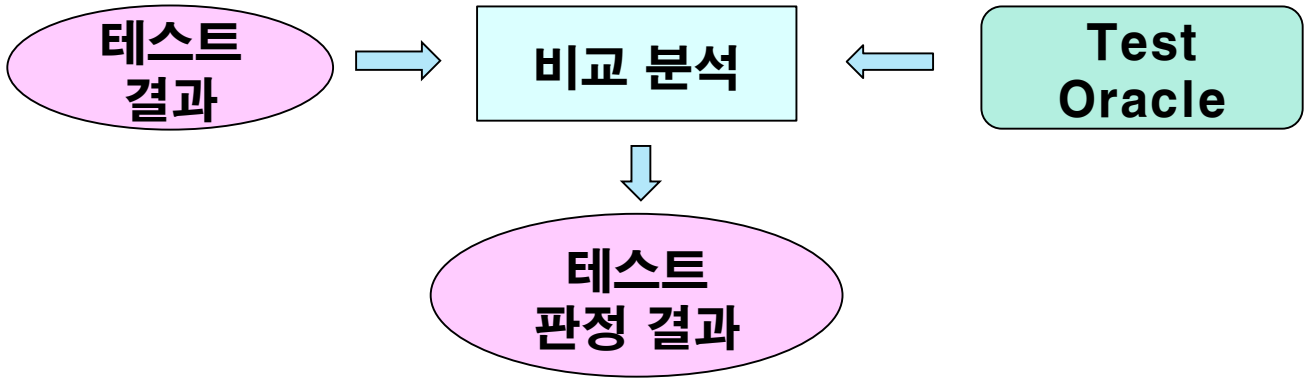


테스트 결과 평가

❑ 기본 개념

- 테스트 결과 평가: 테스트를 실행한 결과 값을 분석하여 테스트의 성공 여부를 판정하는 작업
- 테스트 오라클(Test Oracle): 테스트 실행 결과의 정확성을 판정하기 위한 수단이나 예상되는 결과

❑ 테스트 결과 평가 절차



객체지향 S/W 테스트

- 객체지향 시스템의 테스트는 어렵다!!
  - 정보 은폐와 자료 추상화 -> 클래스의 가시성이 낮음
  - 비절차적 -> 문장의 순서와 실행 순서는 무관
  - 통합할 때 참고할 수 있는 뚜렷한 구조가 없다
  - 상속, 동적 바인딩, 다형성 등의 특성
    -> 컴포넌트 사이의 종속 관계가 매우 복잡
  - 클래스의 속성은 동작에 대한 광역 변수
    -> 동작의 다양한 조합에 따른 변화는 예측이 거의 불가능

- 객체지향 시스템의 테스트 차원
  - Method 차원: 매우 단순
  - Class 차원: 동작 상호간, 동작과 속성의 연결 관계, …
  - Collaboration 차원: 객체 사이의 가능한 조합이 매우 복잡
  - 시스템 차원: 절차적 시스템과 유사

테스트 작업의 산출물

- Test Model(테스트 모델)
  - 테스트 모델은 1개의 테스트 시스템으로 표현
  - Test System
    - Test Case (테스트 케이스)
    - Test Procedure (테스트 절차)
    - Test Component (테스트 컴포넌트)

- Other Artifacts
  - Test Plan (테스트 계획)
  - Defect (결함)
  - Test Evaluation (테스트 결과 평가)

Test Model

- 기본 개념
  - 주로 구현 모델의 실행 가능 컴포넌트에 대하여,
    통합 테스트 및 시스템 테스트를 어떻게 할 것인가를 기술
  - 부가적으로 시스템의 특정 측면에 대한 테스트 방법을 기술
    - 사용자 인터페이스가 사용하기에 편한가? 일관성은 있는가?
    - 시스템에 대한 사용자 매뉴얼은 잘 작성되었는가?
    - 시스템의 성능, 메모리 사용량, …
  - 방대한 테스트 모델은 패키지화 -> 적절한 크기로 조정

- 테스트 시스템의 구성 요소
  - 테스트 케이스 (Test Case)
  - 테스트 절차 (Test Procedure)
  - 테스트 컴포넌트 (Test Component)

Test Case(1)

- 테스트 케이스란?
  - 정의: 시스템을 테스트하는 한가지 방법을 기술한 것
  - 구성 요소: 테스트 대상, 입력 자료, 예상 결과, 테스트 시행 조건
  - 무엇을 테스트 하는가?
    - 시스템의 기능적/비기능적 요구는 모두가 테스트의 대상
    - 과도한 비용이 소요되는 것은 제외
  - 테스트 케이스는 유즈 케이스(요구 추출) 또는 유즈 케이스 실현(설계)으로부터 도출 가능

- 테스트 케이스의 종류
  - 일반적인 테스트 케이스
  - 시스템 전체를 대상으로 하는 테스트 케이스

Test Case(2)

- 일반적인 테스트 케이스
  - 유즈 케이스 또는 이의 시나리오에 대한 테스트
    - Actor와 시스템 사이의 상호작용의 결과를 검증
    - 유즈 케이스에 기술된 사전/사후 조건을 만족하는가를 검증
    - 유즈 케이스에 기술된 일련의 동작이 제대로 이루어지는가를 검증
  - 유즈 케이스 실현(설계) 또는 이의 시나리오에 대한 테스트
    - 유즈 케이스를 실현한 컴포넌트 사이의 상호작용을 검증

- 시스템 전체를 대상으로 하는 테스트 케이스
  - Installation tests: 시스템을 운용 환경에 설치 가능한가? 설치 후에 제대로 작동하는가?
  - Configuration tests: 다양한 네트워크 환경에서 제대로 작동하는가?
  - Negative tests: 시스템의 약점을 발견하기 위한 테스트
  - Stress tests: 시스템이 과도한 부하에 대해서 잘 견디는가?

Test Case 예시(1)

[Pay Invoice 유즈 케이스를 위한 테스트 케이스]

테스트 엔지니어는 당해 유즈 케이스를 위하여 여러 개의 테스트 케이스를 설계하였으며, 각각의 테스트 케이스는 하나의 시나리오에 대하여 검증함.

▣ 테스트 케이스1 (케이스 명칭: Pay 300-Mountail Bike)

▷테스트 대상: 금액이 $300 인 산악 자전거의 송장에 대한 지불 시나리오

▷입력자료(Input)

① 구매자는 판매자(Crazy Mountaineer, Inc.)에게 산악 자전거를 주문
② 판매자는 주문을 접수(ID 12345)하고, 제 비용을 포함하여 $300에 주문을 승인
③ 구매자는 송장(ID 12345)을 접수. 모든 기재 사항이 이상이 없으며 송장이 Pending 상태임을 확인. 판매자의 계좌번호는 22-222-2222, 현재 잔고는 $963,456.00
④ 구매자의 계좌번호는 11-111-1111, 현재 잔고는 $350.

Test Case 예시(2)

▷예상 결과(Expected Result)
① 송장은 Closed 상태로 설정되어 있어야 함
② 구매자의 계좌 잔고는 $50 이어야 함
③ 판매자의 계좌 잔고는 $963,756.00 이어야 함

▷테스트 조건(Conditions)
① 당해 테스트 케이스를 실행하는 동안 다른 유즈 케이스가 구매자 및 판매자의 계좌에 접근해서는 안된다.

▣ 테스트 케이스2
…

▣ 테스트 케이스3
…

▣ 테스트 케이스4
…

Test Procedure

- 기본 개념
  - 정의: 테스트 케이스를 어떤 방법으로 실행시킬 것인가를 기술한 것
  - 테스트 절차는 유즈 케이스의 flow of events description과 유사
  - 테스트 절차에 포함되는 내용
    - 테스트 케이스를 수작업으로 실행할 때의 지침
    - 자동화 도구의 사용법
  - 테스트 절차와 테스트 케이스는 n:n 관계
    - 하나의 테스트 절차에서 여러 개의 테스트 케이스를 사용 가능
    - 하나의 테스트 케이스가 여러 개의 테스트 절차에서 사용 가능

Test Procedure 예시

▣ 테스트 케이스: Pay 300-Mountain Bike (대금 지불 유즈 케이스 관련)

▣ 수행 절차

1. 메인 윈도우에서 『Browse Invoices』메뉴를 선택
   -> Browse Invoice Query 대화상자가 열림
2. 상태 필드에서 Pending을 선택하고, 조회 버튼을 누른다. 나타난 조회 결과 창에서 테스트 케이스(ID12345)에서 기술한 송장이 있는가를 확인
3. 지불하려는 송장을 더블 클릭하여, 해당 송장에 대한 상세 내역이 나타나면, 다음 사항을 확인한다.
   - 상태: Pending
   - 지불일자: 공란
   - 주문승인번호: 테스트 케이스(ID 98765)에 기술한 ID
   - 금액: $300
   - 계좌번호: 테스트 케이스(ID 22222)에 기술한 계좌번호
4. 확인된 송장에 대해 지불 절차를 수행하려면, Pay체크박스에 체크하고, 확인 버튼을 누른다. -> 지불 대화상자가 나타남
5. 지불일자를 입력하고 승인 버튼을 누른다.

General Test Procedure 예시

▣ 적용 대상: 대부분의 유즈 케이스는 객체에 대한 검증 작업으로부터 시작한다. 예를 들어, Pay Invoice 유즈 케이스에서는 주문에 대한 승인 내역과 접수된 송장을 비교하여 이상 유무를 검증한다. 따라서 이 절차는 여러 개의 시나리오에 대한 테스트에서 공통으로 적용 가능한 범용 테스트 절차이다.

▣ 범용 테스트 절차: Validate Business Objects
1. 검증 대상 객체를 생성하여 초기화
2. 대상 객체를 검증하는 역할을 담당하는 능동 객체를 호출
3. 검증 작업의 결과 값을 예상 결과와 비교

▣ 범용 테스트 절차: Verify Scheduling
1. 송장의 지불 예정일자를 확인
2. 지불 일자가 도래한 송장에 대하여 계좌이체를 수행

▣ 범용 테스트 절차: Verify Account Transfer
1. 계좌이체가 올바르게 되었는가를 검증

Test Component

❑ 기본 개념

- 테스트 절차의 전부 또는 일부를 자동화하여 테스트에 사용하는 컴포넌트
  - 테스트 드라이버(Test Driver)
  - 테스트 장비(Test Harness)
  - 테스트 스크립트(Test Scripts)

- 스크립트 언어 또는 프로그래밍 언어로 개발하거나, 테스트 자동화 도구를 사용
- 구현 모델의 컴포넌트를 테스트 하는데 사용
- 테스트 컴포넌트와 테스트 절차는 n:n 관계

Other Artifacts

- Test Plan(테스트 계획)
  - 테스트 정책, 가용 자원, 테스트 일정을 기술
  - 테스트 정책(Test Strategy)
    - 각 iteration에 대해서 실시할 테스트의 종류 및 목적을 정의
    - 요구되는 code coverage
    - 테스트 종결 기준

- Defect(결함)
  - 시스템 실패(failure)를 유발하는 원인이 되는 결점 또는 문제점
  - 개발자가 추적하여 해결해야 하는 대상

- Test Evaluation (테스트 결과 평가)
  - 테스트를 실시한 결과에 대한 평가
  - Test Case Coverage, Code Coverage, 결함 발생 상황

Workers for Test(1)

- Test Designer (테스트 설계자)
  - 다음 사항에 대한 책임을 담당
    - 테스트 모델의 무결성 확보
    - 테스트 모델이 테스트의 목적을 충족시키는가를 확인
    - 테스트 계획 수립
    - 테스트 케이스 작성 및 이에 따른 테스트 절차 수립
    - 통합 테스트 및 시스템 테스트에 대한 평가
  - 테스트 실행에 대해서는 책임이 없음

- Component Engineer(컴포넌트 엔지니어)
  - 테스트 컴포넌트 개발

Workers for Test(2)

- Integration Tester(통합 테스트 수행자)
  - Build에 대한 통합 테스트 실시
  - 발견한 결함을 컴포넌트 엔지니어에게 통보

- System Tester(시스템 테스트 수행자)
  - Build에 필요한 시스템 테스트 실시
    - 시스템 테스트는 actor와 시스템 사이의 상호작용을 검증
    - 경우에 따라 다른 프로젝트 구성원이나 고객(β-test)이 실시할 수도 있음
  - 발견한 결함을 컴포넌트 엔지니어에게 통보

Workflows in Test

1. Plan Test
2. Design Test
    - 2.1 Designing Integration Test Cases
    - 2.2 Designing System Test Cases
    - 2.3 Designing Regression Test Cases
    - 2.4 Identifying and Structuring Test Procedures
3. Implement Test
4. Perform Integration Test
5. Perform System test
6. Evaluate Test

Plan Test(1)

- 입출력 자료
  - 입력 자료: 유즈 케이스 모델, 보충 요구서, 분석 모델, 설계 모델, 구현 모델, 구조 기술서
  - 출력 자료: 테스트 계획

- 작업 개요
  - 테스트 정책(testing strategy)을 기술
    - 어떤 종류의 테스트를 실시할 것인가?
    - 테스트를 어떤 방법으로 수행할 것인가?
    - 언제 테스트를 실시할 것인가?
    - 테스트 종결 기준
  - 테스트에 필요한 인원 및 시스템 자원을 추정
  - 테스트를 위한 일정 계획 수립

Plan Test(2)

❑ Test Strategy 예시

[정제 단계의 마지막 iteration을 위한 시스템 테스트 정책]

▣ 테스트의 75% 이상은 자동화 하여야 하며, 나머지는 수작업 진행한다.

▣ 각각의 유즈 케이스에 대해서 아래의 제어흐름은 테스트 하여야 한다.
1. 정상적인 흐름 (normal flow)
2. 3개 이상의 대체 흐름(alternative flows)

▣ 테스트 종결 기준
1. 테스트 케이스의 90% 이상을 통과해야 한다.
2. 우선 순위가 중간 이상인 유즈 케이스에는 결함이 없어야 한다.

Design Test

- 입출력 자료
  - 입력 자료: 유즈 케이스 모델, 보충 요구서, 설계 모델,
    이전 builds에 대한 구현 모델,
    구조 기술서, 테스트 계획(테스트 정책 및 일정)
  - 출력 자료: 테스트 케이스, 테스트 절차

- 작업 목적
  - 각 build에 대한 테스트 케이스를 작성
  - 테스트를 어떤 절차에 따라 수행할 것인가를 기술

Designing Integration Test Cases

- 작업 개요
  - 통합 테스트를 위한 대부분의 테스트 케이스는 유즈 케이스 실현(설계)로부터 추출 가능
  - 테스트 계획에서 제시한 목표를 최소의 노력으로 달성할 수 있는 테스트 케이스를 작성해야 함
  - 테스트 설계자는 상호작용 다이어그램을 기반으로, 아래 사항을 조합하여 적절한 시나리오를 발견
    - Actor의 입력 자료
    - Actor를 위한 출력 자료
    - 시스템 시작 상태
  - 객체 사이에 발생 가능한 실제 상호작용을 포착하여, 상호작용 다이어그램과 비교

Designing System Test Cases

- 작업 개요
  - 유즈 케이스의 적절한 조합에 대해서 다양한 상황에서 올바르게 작동하는가를 테스트
    - 서로 다른 하드웨어 구성
    - 시스템에 대한 부하 수준 및 동시에 접속하는 Actors의 수
    - DB의 크기
  - 유즈 케이스의 조합에 대한 우선 순위를 고려하여 테스트 케이스 작성
    - 병렬로 수행되어야 하거나 수행될 수 있는 경우
    - 병렬로 수행되었을 때, 서로 영향을 미칠 수 있는 경우
    - 다수의 프로세스를 포함하는 경우
    - 복잡하고 예측하기 어려운 방법으로 시스템 자원을 사용하는 경우
  - 시스템 테스트를 위한 대부분의 테스트 케이스는 유즈 케이스로부터 추출 가능
    - flow of events 및 special requirements에 초점

Designing Regression Test Cases

- 작업 개요
  - 회귀 테스트에서 사용하는 테스트 케이스
    - 이전 builds에 적용했던 테스트 케이스
    - 신규로 개발한 테스트 케이스
  - 지속적인 재사용이 가능한 테스트 케이스의 개발이 중요
    - 대상 시스템의 변경에 절 견딜 수 있는 테스트 케이스
    - 노력을 투자한 만큼의 효과를 기대할 수 있는 테스트 케이스
  - 회귀 테스트를 위한 테스트 케이스는 세심한 주의를 기울여야 함

Identifying and Structuring Test Procedures

- 작업 개요
  - 테스트 설계자는 테스트 케이스 별로 테스트 절차를 제시할 수 있음
  - 가능한 한 재사용이 가능하고, 다수의 테스트 케이스에 적용할 수 있는 테스트 절차를 수립하는 것이 더욱 바람직함
  - 모든 테스트 절차는 서비스 서브시스템에 포함된 클래스를 테스트할 수 있도록 작성되어야 함
  - 테스트 절차를 서비스 서브시스템에 초점을 맞추어 작성
    -> 테스트 절차의 유지 관리가 용이

Test Procedure 작성 예시

▣ Accounts 서비스 서브시스템을 위한 테스트 절차 작성

▷ 서브시스템의 역할: 예금계좌 사이에 금액 이체를 담당하며, Pay Invoice를 포함한 다수의 유즈 케이스에서 사용

▷ 테스트 절차의 명칭: Verify Account Transfer

▷ 테스트를 위한 입력 자료
① 2개의 계좌 번호
② 이체할 금액

▷ 테스트 케이스: Pay Invoice 유즈 케이스를 위한 8개의 케이스와 Verify Account Transfer 유즈 케이스를 위한 14개의 케이스를 설계

▷ 수행 절차
1. 이체 대상 계좌의 이체 전 예금 잔액을 확인
2. 이체 대상 계좌의 이체 후 예금 잔액을 확인

Implement Test

- 입출력 자료
  - 입력 자료: 테스트 케이스, 테스트 절차, 테스트 대상 build의 구현 모델
  - 출력 자료: 테스트 컴포넌트

- 작업 개요
  - 테스트 컴포넌트의 용도: 테스트 절차의 전부 또는 일부를 자동화
  - 컴포넌트 확보 방안
    - Visual Basic 테스트 스크립트와 같은 자동화 도구를 사용
      -> 테스트 절차를 스크립트 언어로 기술
    - 프로그래밍 언어로 개발
      -> 테스트 절차를 요구 기술서로 사용

Perform Integration Test

❑ 입출력 자료

- 입력 자료: 테스트 케이스, 테스트 절차, 테스트 컴포넌트, 테스트 대상 build의 구현 모델
- 출력 자료: 결함

❑ 작업 개요

① 테스트 컴포넌트를 사용하거나 테스트 케이스별 수작업을 통하여 대상 build에 대한 통합 테스트 실시

② 테스트 결과를 예상 결과와 비교하여, 그 차이를 조사

③ 결함을 컴포넌트 엔지니어에게 통보

④ 테스트 결과를 테스트 설계자에게 보고

⑤ 테스트 설계자는 보고된 결함을 취합하여 적절한 조치를 취함

Perform System Test

- 입출력 자료
  - 입력 자료: 테스트 케이스, 테스트 절차, 테스트 컴포넌트, 테스트 대상 build의 구현 모델
  - 출력 자료: 결함

- 작업 개요
  - 통합 테스트의 결과가 목표 수준에 도달하면, 시스템 테스트를 시작
  - 테스트 과정은 통합 테스트와 비슷한 방법으로 실시

Evaluate Test(1)

❑ 입출력 자료
- 입력 자료: 테스트 계획, 테스트 모델, 결함
- 출력 자료: 대상 iteration에 대한 테스트 결과 평가

❑ 작업 개요

① 다음을 위한 Metrics(측정 도구) 준비
- 대상 시스템의 품질 수준 평가
- 추가로 필요한 테스트의 분량을 측정

② 테스트 결과를 테스트 계획에 기술된 목표 수준과 비교하여 결과에 대한 평가 실시

③ 테스트의 완전도, 신뢰도, 향후 조치 등을 평가서에 기록

④ 결함 발생 추세를 분석하여, 테스트 진행 방향을 결정
- 필요한 사항에 대한 추가 조치
- 테스트 종결

Evaluate Test(2)

- s/w 품질 수준 측정을 위한 Metrics
  - Testing completeness(테스트의 완전도)
    - 테스트 케이스의 coverage와 테스트 대상 컴포넌트의 coverage로부터 측정
  - Reliability(테스트의 신뢰도)
    - 발견된 결함의 추세 분석과 예상 결과와의 부합 정도를 바탕으로 측정

- 결함 추세 분석 결과에 따라 가능한 후속 조치
  - 시스템이 성숙되지 않았다면, 결함 발견을 위한 추가 테스트 실시
  - 당초 설정한 품질 수준이 너무 높다면, 종결 기준을 완화
  - 목표 수준에 도달한 부분과 그렇지 못한 부분을 구분
    - 도달한 부분 -> 고객에게 인도
    - 도달하지 못한 부분 -> 테스트를 계속