KU KONKUK UNIVERSITY

2010/10/13

A Brief Essay on Software Testing

Antonia Bertolino and Eda Marchetti

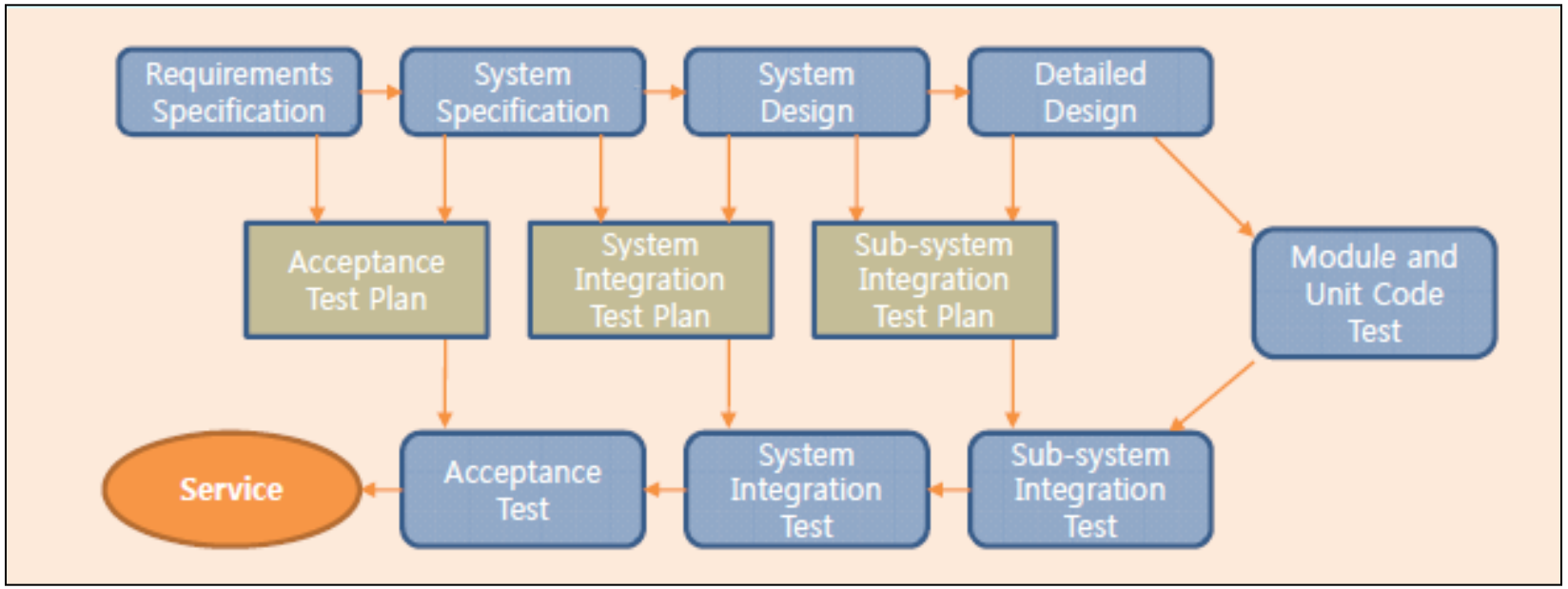
[B Class] T7
200911388 박미관
200911397 송찬우
200911398 신우철

0. CONTENTS

1. INTRODUCTION
2. TERMINOLOGY AND BASIC CONCEPTS
3. TYPES OF TESTS
4. TEST LEVELS
5. STRATEGIES FOR TEST CASE SELECTION
6. TEST DESIGN
7. TEST EXECUTION
8. TEST DOCUMENTATION
9. TEST MANAGEMENT
10. TEST MEASUREMENTS
11. Q & A

1. INTRODUCTION

✓ 소프트웨어 개발 과정



1. INTRODUCTION

- ✓ Testing :
  - 소프트웨어 개발 과정에서 중요한 부분.
  - 버그를 찾고, 제품의 신뢰도와 질을 향상 시킴.
  - 몇몇의 까다로운 활동들로 이루어져 있음.
    - 적절한 테스트 사례를 찾는 것.
    - 선별된 테스트를 수행하는 것.
    - 테스트 결과를 받아들일지 말지 결정하는 것.
    - 실패의 영향을 평가하는 것.
    - 실패의 직접적, 간접적 원인을 찾는 것.
    - 테스팅이 충분이 되었는지 판단하는 것.

2. TERMINOLOGY AND BASIC CONCEPTS

- 2.1. On the Nature of the Testing Discipline
- 2.2. A General Definition
- 2.3. Fault Versus Failure
- 2.4. The Notion of Software Reliability

2.1. On the Nature of the Testing Discipline

- ✓ 테스팅의 목표:
  - 소프트웨어 엔지니어의 신뢰도를 향상 시키는 것.

- ✓ 테스트 기술은 두 가지로 분류됨:
  - Static analysis technique
    - 실행 중이 아닌 제품을 테스트 하는 것.
    - 부정확한 면이 있음.
  - Dynamic analysis technique
    - 실행 중인 제품을 테스트 하는 것.
    - 효율적이고, 위의 것보다 정확함.
    - 하지만 실행된 테스트에 대해서만 결과를 갖는다.

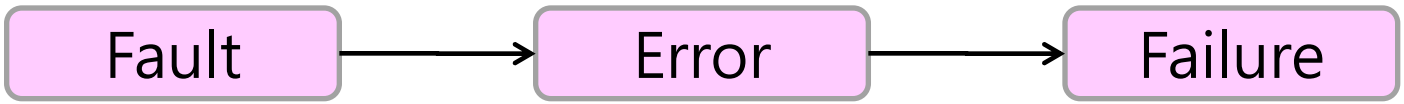
  ☞ 상호 보완적

2.2. A General Definition

- ✓ Software testing은 프로그램상에서 한정된(finite) test case을 통하여 dynamic verification으로 진행된다.
  - "Dynamic"
    - 프로그램이 실행 중에 있다는 의미.
  - "Finite"
    - 테스트 과정에서 실행할 수 있는 test cases의 숫자가 한정되어 있다는 의미.

2.3. Fault Versus Failure

- ✓ Fault
  - Failure의 원인
  - 예) 코드가 빠지거나 부정확한 것.
- ✓ Error
  - 프로그램의 불안정한 상태.
  - 오랫동안 fault가 발견되지 않고 남아있을 때 발생함.
- ✓ Failure
  - 요구된 기능을 수행하는데 있어, 프로그램의 무능력함.
  - 예) 잘못된 결과, 비정상적인 종료 등.



2.4. The Notion of Software Reliability

- ✓ 주어진 시간과 환경에서 실패 없이 프로그램이 실행될 확률.
- ✓ 즉, 프로그램 실행 중 최종사용자의 입력이 얼마나 failure를 발생하는지에 대한 정도
- ✓ 제품이 출시를 위한 준비가 되었는지를 결정하는데 중요한 역할을 함.

3. TYPES OF TESTS

- 3.1. Static Techniques
- 3.2. Dynamic Techniques
- 3.3. Objectives of Testing

3.1. Static Techniques

- ✓ Static techniques은 다음의 시험들에 기초한다
  - project documentation
  - software models and code
  - other related information about requirements and design.
- ✓ 다음 단계에서 사용된다.
  - Requirement phase:
    - 문법, 일치, 완벽함 등을 확인하기 위해서
  - Design phase:
    - 불일치성을 찾고, 요구사항의 수행을 평가하기 위해서
  - implementation phase :
    - 표준이나 협약에 준수 했는지, interfaces 와 data type이 올바른지와 같은 형태를 확인하기 위해서.

3.1. Static Techniques

- ✓ 전통적인 static techniques은 포함한다 :
  - Software inspection : 문서를 분석
  - Software reviews : 리뷰 분석
  - Code reading : 코드를 분석
  - Algorithm analysis and tracing : 알고리즘 분석

  ☞ 매우 수동적이고, error가 발생하기 쉽고, 시간이 오래 걸린다.

- ✓ 이러한 문제들을 해결하기 위해, formal methods의 사용에 기초한 static analysis technique 제안함.

- ✓ formal methods :
  - 수학적 기반의 technique.
  - 신뢰성과 견고한 설계를 돕는다.

3.2. Dynamic Techniques

- ✓ 프로그램의 실행을 관찰하여 프로그램의 정보를 얻는 것을 말한다.
- ✓ 입력 값을 받은 코드의 실행이 올바르게 실행되는지를 테스트 한다.
- ✓ 테스트 중 소요되는 전체시간과 노력 그리고 입력에 관해 반드시 균형 잡힌 테스트를 해야 한다.

3.3. Objectives of Testing

- ✓ Acceptance/qualification testing.
  - 최종 테스트.
  - 고객의 요구사항을 만족하는지 확인하기 위해서
- ✓ Installation testing.
  - 목적 환경에서 시스템이 설치되는지 확인하기 위해서
- ✓ Alpha testing.
  - 시스템 출시 이전, 사내 사람들이 수행하는 테스트
- ✓ Beta testing.
  - 외부 사용자에게 보급되어 수행되는 테스트
  - 각각의 테스터는 각자의 시스템 상태를 기준으로 테스트한다.

3.3. Objectives of Testing

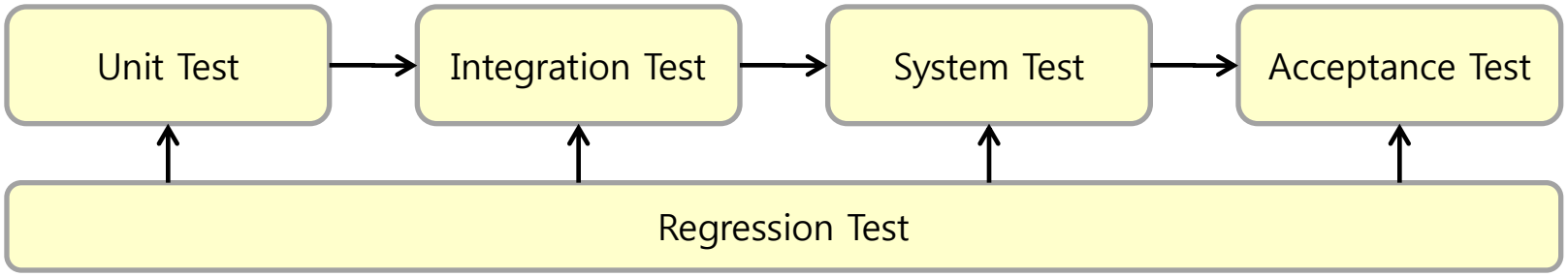
- ✓ Reliability achievement.
  - 신뢰도를 향상시키기 위해.
- ✓ Conformance testing/functional testing.
  - 기능이 의도대로 실행되는지, 그리고 필요한 service와 methods를 제공하는지 검사한다.
- ✓ Regression testing.
  - 이전 단계에서 통과한 시스템이 여전히 올바르게 작동하는지를 확인하기 위해서.
- ✓ Performance testing.
  - 명세 된 performance requirements를 충족시키는지 확인하기 위해서.

3.3. Objectives of Testing

- ✓ Usability testing.
  - 시스템의 기능적 효율성과 등을 평가하기 위해서
    - 시스템을 배우고, 사용 하는 것이 편리한지
    - The ability to recover form user errors.
- ✓ Test-driven development.
  - 요구사항의 정확한 수행에 관하여 따로 검사하는 대신 일부의 Test case specification으로 대체하여 사용하는 방법

4. TEST LEVELS

- 4.1. Unit Test
- 4.2. Integration Test
- 4.3. System Test
- 4.4 Regression Test



4.1. Unit Test

- ✓ Unit
  - 테스트 가능한 소프트웨어의 가장 작은 단위.
- ✓ 기능적 명세를 만족시키는지 보장하기 위한 과정.
- ✓ 다음의 목적으로 사용 됨:
  - Interfaces를 확인 하기 위해서
    - Ex) argument와 parameter의 수가 같은지
  - Data structure 또는 boundary conditions를 확인 하기 위해서
    - Ex) 부적절한 타이핑, 부정확한 변수 명

4.2. Integration Test

- ✓ 작은 소프트웨어 단위나 요소들이 합쳐진 큰 요소들에 대한 테스트.
- ✓ 위에서 말한 큰 요소에서 발생하는 문제들을 찾는 것이 목적
- ✓ 각각의 Unit에는 문제가 없더라도, Unit들이 합쳐진 복잡한 시스템에서는 문제가 발생할 수 있다.

4.3. System Test

- ✓ System testing
  - 실제 하드웨어 환경에 있는 모든 시스템/임베디드 시스템을 포함하여 진행한다.
  - 시스템이 사용자의 요구사항을 수행한다는 것을 입증한다.

- ✓ System testing의 주된 목표:
  - Unit test와 integration test에서 찾아내지 못한 failure을 발견해 내는 것.
  - 신뢰도 증가
  - 제품의 출고를 결정하는데 도움이 될만한 정보를 수집.

4.3. System Test

- ✓ Acceptance testing
  - System testing에 자주 추가되는 내용이다. 그러나 system testing의 새로운 단계라기보단 확장에 가깝다고 볼 수 있다.
  - 일부 명세와 다르게 실행보다는 요구사항의 usability에 초점을 맞춘다.
  - 최종사용자가 시스템의 기능을 최대한 활용하게끔 익히는데 어느 정도의 노력이 필요할 지 확인하고,
  - 최종 사용자만이 찾을 수 있는 오류 발견을 위해서

4.4. Regression Test

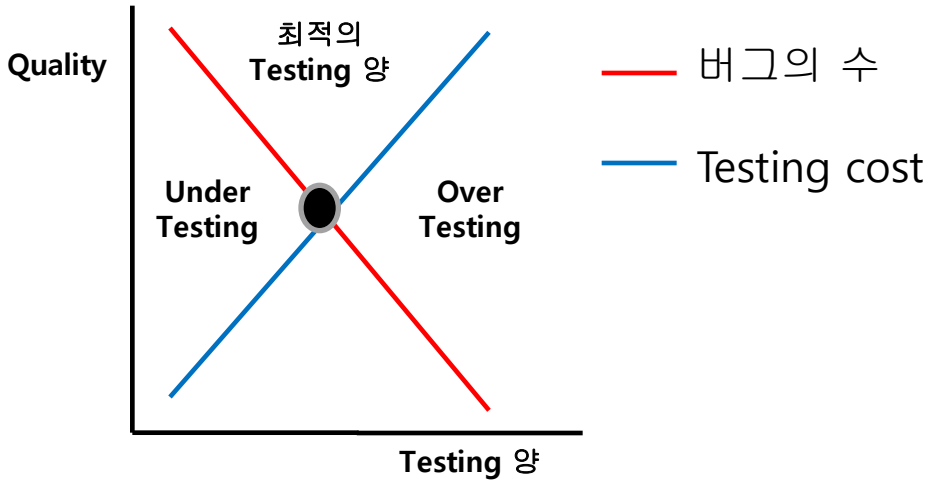
- ✓ 변경된 프로그램의 일부를 재검사(retesting)하여 소프트웨어의 에러를 발견하기 위한 software testing의 모든 유형을 말한다.
- ✓ 독립된 테스팅 단계가 아니다.
- ✓ A unit, a combination of components, or a whole system들의 재검사에도 관련이 있다.

5. STRATEGIES FOR TEST CASE SELECTION

- 5.1. Selection Criteria Based on Code
- 5.2. Selection Criteria Based on Specifications
- 5.3. Other Criteria

5. STRATEGIES FOR TEST CASE SELECTION

✓ Effective testing requires strategies to trade off between
- 테스팅의 질을 높이는 것.
- 시간과 비용을 감소시키는 것.



5.1. Selection Criteria Based on Code

- ✓ “structrual test” 또는 “white-box testing”이라고 불림.
- ✓ Code의 역할과 작동방법을 관찰하여 얻은 정보를 사용하여 테스트 할 대상과 테스트에서 배제할 대상, 테스트 접근 방법 등을 결정함.
- ✓ 경로를 실행하여 test case를 설계하는 기법을 사용.
  - 경로 : 시작 지점부터 종료 지점까지 컴포넌트나 시스템 이벤트의 흐름.

5.1. Selection Criteria Based on Code

- ✓ Statement coverage
  - 모든 명령문을 적어도 한번 수행하여 test case 산출
- ✓ Branch coverage
  - 가장 간단한 경로를 실행하여 test case 산출
- ✓ Full-path coverage
  - 모든 경로를 적어도 한번 수행하여 test case 산출

5.2. Selection Criteria Based on Specifications

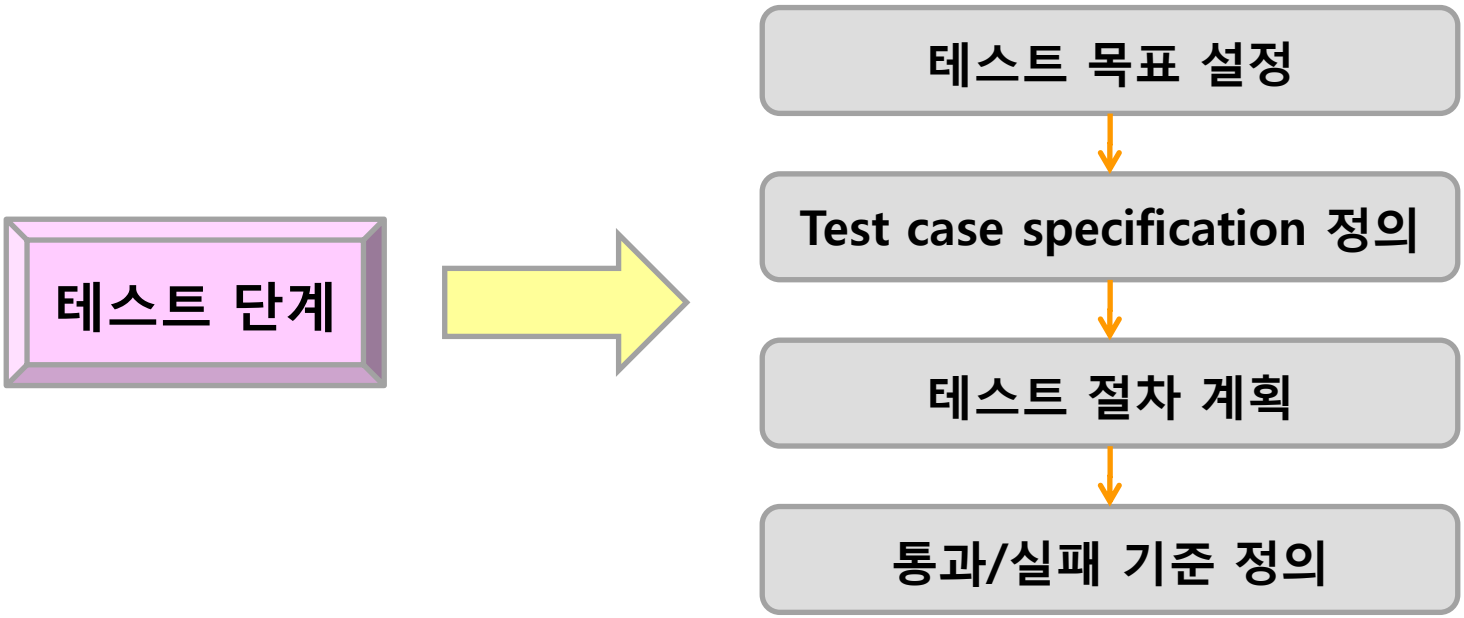
- ✓ Code의 세부사항에 대한 통찰 없이 입력과 출력을 확인함으로써 소프트웨어를 테스트 하는 것 (Black-box testing)
- ✓ 3가지 기법 :
  - Equivalence classes.
    - 즉, 가능한 test cases의 수를 테스트 하기에 충분하도록 쉽고 작은 숫자로 줄이는 것.
  - Boundary conditions.
    - 경계 부분에서 잘 작동한다면, 경계가 아닌 부분에서도 당연히 잘 작동할 것이라는 직관적 사실에 기반.
  - Cause-effect graphs
    - 입력과 원인을 연관된 출력과 함께 시각적으로 표현한 것. 테스트 케이스의 완벽성과 모호성을 체크하는데 도움이 된다.

5.3. Other Criteria

- ✓ Based on Tester's Intuition and experience
  - 유사한 프로그램에 대한 테스터의 능력, 직관, 경험에 의한 것.
- ✓ Fault-based
  - 예상되는 fault를 찾아내는데 중점을 둔 test cases를 제안한다.
- ✓ Based on operational usage
  - 소프트웨어의 작동 환경을 가능한 가깝게 재연한다.

6. TEST DESIGN

✓ 테스트 계획을 수립하거나 갱신하는 활동



7. TEST EXECUTION

- 7.1. Launching the Tests
- 7.2. Test Oracles
- 7.3. Test Tools

7.1. Launching the Tests

- ✓ 테스트 대상 컴포넌트나 시스템에 대해 테스트를 수행하여 실제 결과를 생성.

- ✓ Code-based criterion :
  - 경로를 이용한 테스트 기법을 사용하여 취약점을 찾는다.

- ✓ Specification-based criterion :
  - 앞에서 말한 기법들을 이용하여 얻어진 test case를 가지고 테스트 수행.

7.2. Test Oracles

- ✓ 테스트 대상 소프트웨어의 실제 결과와 비교할 목적으로 예상 결과를 결정하는 근거.
  - Ex) 기존 시스템, 사용자 매뉴얼, 또는 개인의 전문 지식
- ✓ 주어진 테스트 결과를 받아들일 수 있는지 없는지 판단. 하지만 항상 올바른 판단을 하는 것은 아니다.

7.3. Test Tools

- ✓ Test harness(drivers, stubs)
  - 시스템의 부분을 실행시키기 위해, 시뮬레이터 제공.
- ✓ Test generators
  - Test case를 생성하는 것을 도움
- ✓ Capture/Replay
  - 자동적으로 이전에 수행된 테스트를 재실행한다.
- ✓ Oracle/file comparators/assertion checking
  - 테스트 결과가 성공적인지 실패적인지 결정하는데 도움
- ✓ Coverage analyzer/Instrumenter
  - 경로의 규모 측정

7.3. Test Tools

- ✓ Tracers
  - 프로그램 실행 과정을 추적.
- ✓ Reliability evaluation tools
  - 신뢰도를 평가.

8. TEST DOCUMENTATION

- ✓ 문서화는 테스트과정의 필수적인 부분이다.
- ✓ 문서의 여러 가지 종류.
  - Test plan
  - Test design specification
  - Test case specification
  - Test procedure specification
  - Test log
  - Test incident or problem report

8. TEST DOCUMENTATION

- ✓ Test Plan.
  - 테스트 도구나 통과/실패 기준 등을 정의한다.
- ✓ Test Design Specification.
  - 테스트 할 특징과 그에 연관된 테스트 방안을 정한다.
- ✓ Test Case Specification.
  - 특정 제약사항이나 test case의 실행에 필요한 입/출력을 정의한다.

Test Case	Specification
Test case ID	The unique identifier associated with the test case
Test items and purpose	The items and features exercised
Input data	The explicit list of the inputs required for executing the test case (values, files, database, etc.)
Test case behavior	Description of the expected test case behavior
Output data	The list of the outputs admitted for each feature involved in the test case, possibly associated with tolerance values
Environmental setup	The hardware/software configuration required
Specific procedural requirements	The constraints and the special procedures required
Test case dependencies	The Ids of the test cases that must be executed prior to this test case.

8. TEST DOCUMENTATION

- ✓ Test Procedure Specification.
  - 단계와 test case를 실행하는데 필요한 특별한 요구사항을 명세 한다.
- ✓ Test Log.
  - 테스트의 결과를 기록한다.
  - Ex) failure의 발생, 프로젝트의 검토에 필요한 정보들

Test log ID	The unique identifier associated with the test log
Items tested	Details of the items tested, including environmental attributes
Events	The list of the events that occurred including the start and end date and time of each event, ID of the test procedures executed, personnel who executed the procedures, description of text procedures results, environmental details, description of the anomalous events that occurred

- ✓ Test Incident or Problem Report.
  - 입력, 얻어지는 결과와 같은 여러 사건들을 묘사한다.

9. TEST MANAGEMENT

- ✓ 테스팅 단계에서 성공적인 테스팅을 위한 가장 중요한 요소는 협력적인 태도이다.
- ✓ Manager는 개발 기간 동안에 오류발견에 대해 호의적인 태도를 갖도록 하는데 중요한 역할을 한다.
- ✓ Main manager's activity :
  - 업무의 완료 시기를 정함.
  - 업무 수행에 필요한 자원과 노력을 추정.
  - 업무와 관련된 risk의 양을 정함.
  - 노력/비용을 추정.
  - 품질 관리 방안을 마련.

10. TEST MEASUREMENTS

- 10.1. Evaluation of the Program under Test
- 10.2. Evaluation of the Test Performed
- 10.3. Measures for Monitoring the Testing Process

10.1. Evaluation of the Program under Test

- ✓ Program measurement : 테스트 계획과 설계에 초점
  - Linguistic measures.
  - : 프로그램 또는 명세서의 특성에 기반
  - Structural measures.
  - : object들 사이의 구조적 관계에 기반
  - Hybrid measures.
  - : 언어적, 구조적 특성에 기반
- ✓ Fault density.
  - fault의 개수를 계산함으로써 측정
- ✓ Life testing, reliability evaluation.
  - 테스팅을 중단해도 되는지 결정하고, 설정한 목표에 신뢰도가 도달했는지 평가한다.

10.2. Evaluation of the Test Performed

✓ 다음과 같은 것을 평가한다 :

- Coverage/thoroughness measure
  - Testing 동안 실행되어지는 수많은 기능, 경로, 문장 등을 평가.
- Effectiveness
  - 프로그램에서 테스팅의 효과를 정량화 하여 효율성을 평가.

10.3. Measures for Monitoring the Testing Process

- ✓ 많은 수의 fault 또는 failure를 찾아낼 수 있는 테스트 기준이 가장 유용하다.
- ✓ 하나의 fault가 여러 개의 failure를 만들 수 있고, 또는 하나의 failure가 여러 개의 fault에서 발생 될 수 있다.
- ✓ 통계적인 measure이 가장 객관적임.

11. Q & A