# Software testing – Implementation

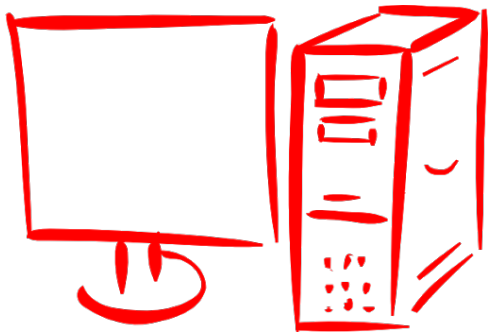
SCS3207/ IS3103

# Recap from previous session

- White box testing
  - data processing and calculation correctness tests
  - path coverage vs. line coverage
  - McCabe's cyclomatic complexity metrics
  - software qualification and reusability testing
  - advantages/disadvantages of white box testing
- Black box testing
  - Equivalence classes for output correctness tests
  - Operation / Revision / Transition factor testing classes
  - advantages/disadvantages of black box testing

# Overview

- The testing process
- Test case design
- Automated testing
- Alpha and beta site testing programs



# The Testing Process

The main issues to be raised with respect to testing implementation:

1. Test effectiveness : reduction of the percentage of undetected errors remaining in the software or system tested
2. Test efficiency : performance of those tests with fewer resources

# Test Effectiveness

Test Effectiveness can be defined as how effectively testing is done or goal is achieved that meets the customer requirement.

In SDLC (Software development Life Cycle), we have requirements gathering phase where SRS (Software Requirements Specification) and FRD (Functional Requirements Document) are prepared and based on that development team starts building the software application, at the same time test cases are carved out of SRS and FRD documents by the testing team.

# Test Effectiveness

- Test effectiveness starts right at the beginning of the development and execution of test cases and after development is completed to count the number of defects.

- Defects can be valid or invalid. Valid defects are required to be fixed in the application or product and invalid ones need to be closed or ignored.

- Thus, mathematically it is calculated as a percentage of a number of valid defects fixed in software application divided by the sum of a total number of defects injected and a total number of defects escaped.

# Test Efficiency

Test Efficiency can be defined as the most efficient way in which the resources associated with the software project are utilized to achieve an organization goal.

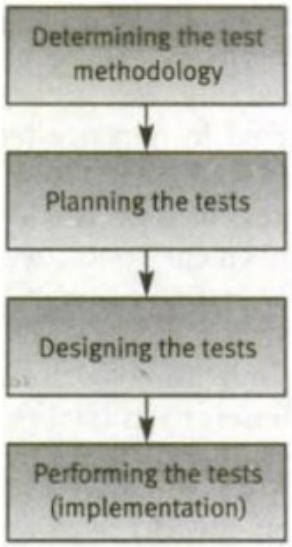
Test efficiency is an internal process for an organization that evaluates the utilization of resources to develop a software product.

Mathematically test efficiency is calculated as a percentage of the number of alpha testing (in-house or on-site) defects divided by sum of a number of alpha testing and a number of beta testing (off-site) defects.

# Testing Process

Test Planning and Designing activities include:

- Determining the test methodology
- Planning unit and integration tests
- Planning the system test
- Designing the tests
- Implementation



# Determining Test Methodology

Selecting the appropriate required software quality standard

Determining the test strategy :

- Big bang or incremental testing (bottom-up or top- down)?
- Which parts to test using white box?
- The extent of automated testing.

# Testing Planning

Planning the tests including,

- unit tests
- integration tests
- system tests (refer to the entire software package/system)

# Test Planning : Main issues

What to test?

- which modules to be unit tested?
- which integrations to be tested?
- for the system test : consider the already planned unit and integration tests, ignore low-priority modules?
- Prioritize based on the *Damage severity level* and *Software risk level*

- Which types of sources are to provide the test cases?
  - samples of real-life and/or synthetic test cases

# Test Planning : Main issues cont..

- Who performs the tests?
  - development team/ independent testing team(s)/ another development team/ outsourcing...

- Where will the tests be performed?
  - Developer's site/ customer's site/ third party site...

# Test Planning : Main issues cont..(2)

When to terminate the (system) tests?

- The completed implementation route : perfection approach
- The mathematical models application route :
  - estimate the % of undetected errors, based on the rate of error detection
- The error seeding route :
  - errors of various types are seeded (hidden) in the tested software prior to testing
  - assumes that the % of discovered seeded errors will correspond to the % of real errors detected

# Test Planning : Main issues cont..(3)

- The dual independent testing teams route
  - By comparing the lists of detected errors provided by two teams, the number of errors left undetected is estimated
- Termination after resources have petered out : undesirable!!

# Test Planning : Documentation

The planning stage of the software system tests is commonly documented in a "Software Test Plan" (STP).

The main sections of the STP include:

1. Scope of the tests
2. Testing environment
3. Test details (for each test)
4. Test schedule (for each test or test group) including time estimates

**Frame 10.2** **The software test plan (STP) – template**

**1 Scope of the tests**

1.1 The software package to be tested (name, version and revision)
1.2 The documents that provide the basis for the planned tests (name and version for each document)

**2 Testing environment**

2.1 Testing sites
2.2 Required hardware and firmware configuration
2.3 Participating organizations
2.4 Manpower requirements
2.5 Preparation and training required of the test team

**3 Test details (for each test)**

3.1 Test identification
3.2 Test objective
3.3 Cross-reference to the relevant design document and the requirement document
3.4 Test class
3.5 Test level (unit, integration or system tests)
3.6 Test case requirements
3.7 Special requirements (e.g., measurements of response times, security requirements)
3.8 Data to be recorded

**4 Test schedule (for each test or test group) including time estimates for the following:**

4.1 Preparation
4.2 Testing
4.3 Error correction
4.4 Regression tests

# Test Designing

The test design is carried out based on the Software Test Plan (STP).

The products of the test design stage are:

- Detailed design and procedure for each test → “software test procedure” document
- Test case database/file → “test case file” document

Or the products of the test design stage may be documented in a single document called the “Software Test description” (STD)

**Frame 10.3** **Software test descriptions (STD) – template**

**1 Scope of the tests**

1.1 The software package to be tested (name, version and revision)
1.2 The documents providing the basis for the designed tests (name and version for each document)

**2 Test environment (for each test)**

2.1 Test identification (the test details are documented in the STP)
2.2 Detailed description of the operating system and hardware configuration and the required switch settings for the tests
2.3 Instructions for software loading

**3 Testing process**

3.1 Instructions for input, detailing every step of the input process
3.2 Data to be recorded during the tests

**4 Test cases (for each case)**

4.1 Test case identification details
4.2 Input data and system settings
4.3 Expected intermediate results (if applicable)
4.4 Expected results (numerical, message, activation of equipment, etc.)

**5 Actions to be taken in case of program failure/cessation**

**6 Procedures to be applied according to the test results summary**

# Test Implementation

Commonly, the testing implementation phase activities consist of :

- a series of tests,
- corrections of detected errors
- and re-tests (regression tests).

# Test Implementation cont..

The tests are carried out by running the test cases according to the test procedures given in the “Software Test description” (STD).

Re-testing (also termed “regression testing”) is conducted to verify that the errors detected in the previous test runs have been properly corrected, and that no new errors have entered as a result of faulty corrections.

# Test Implementation: Documentation

The results of the individual tests and re-tests are documented in a “Software Test Report” (STR).

The summary of the set of planned tests is documented in the “Test Summary Report” (TSR).

**Frame 10.4** **Software test report (STR) – template**

**1 Test identification, site, schedule and participation**

1.1 The tested software identification (name, version and revision)
1.2 The documents providing the basis for the tests (name and version for each document)
1.3 Test site
1.4 Initiation and concluding times for each testing session
1.5 Test team members
1.6 Other participants
1.7 Hours invested in performing the tests

**2 Test environment**

2.1 Hardware and firmware configurations
2.2 Preparations and training prior to testing

**3 Test results**

3.1 Test identification
3.2 Test case results (for each test case individually)
  3.2.1 Test case identification
  3.2.2 Tester identification
  3.2.3 Results: OK / failed
  3.2.4 If failed: detailed description of the results/problems

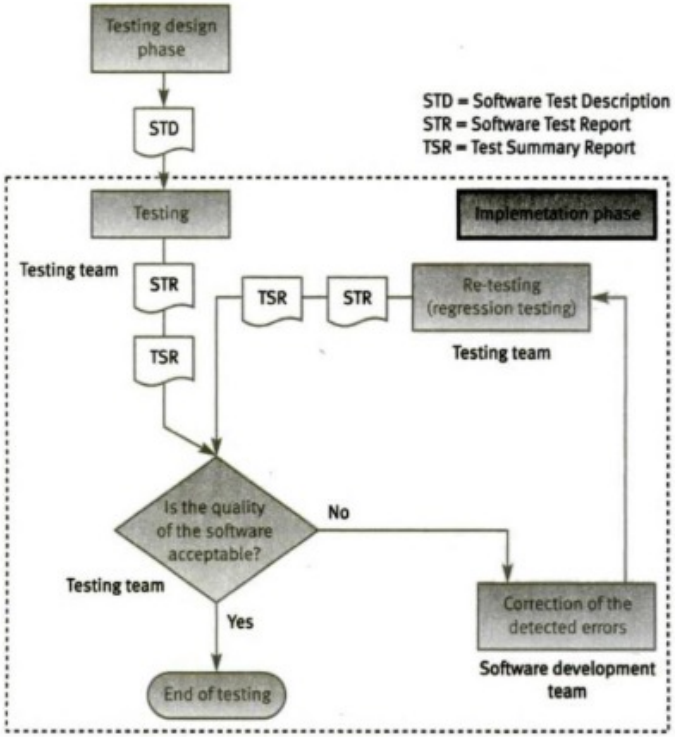
**4 Summary tables for total number of errors, their distribution and types**

4.1 Summary of current tests
4.2 Comparison with previous results (for regression test summaries)

**5 Special events and testers' proposals**

5.1 Special events and unpredicted responses of the software during testing
5.2 Problems encountered during testing
5.3 Proposals for changes in the test environment, including test preparations
5.4 Proposals for changes or corrections in test procedures and test case files

# Test Implementation Activities



# Test Case Design

A test case is a,

- documented set of the data inputs
- operating conditions required to run a test item,
- the expected results of the run.

The tester runs the program for the test item according to the test case documentation and compare the actual results with the expected results in the documents.

- If the obtained results completely agree with the expected results, no error is present or at least has been identified.
- When some or all of the results do not agree with the expected results, a potential error is identified.

# Test Case Data Components

Application of the test case will produce one or more of the following types of expected results:

- Numerical
- Alphabetic (name, address, etc.)
- Error message. Standard output informing user about missing data, errorneous data, unmet conditions etc.

With real-time software and firmware, there can be additional types of expected results.

# Test Case Sources

1. Random samples of real-life cases:

e.g.

- A sample of urban households (to test a new municipal tax information system)
- A sample of shipping bills (to test new billing software)
- A sample of control records (to test new software for control of manufacturing plant production)
- A recorded sample of events that will be “run” as a test case (to test online applications for an internet site, and for real-time applications)

# Test Case Sources cont..

2. Synthetic test cases (“simulated test cases”) prepared by the test designers.

- This type of test case does not refer to an existing customer, shipment or product but to combinations of the system’s operating conditions and parameters (defined by a set of input data).
- These combinations are designed to cover all known software operating situations or at least all situations that are expected to be in frequent use or that belong to a high error probability class.

| Implication | Type of test case source | |
|---|---|---|
| | Random sample of cases | Synthetic test cases |
| Effort required to prepare a test case file | Low effort, especially where expected results are available and need not be calculated | High effort; the parameters of each test case must be determined and expected results calculated |
| Required size of test case file | Relatively high as most cases refer to simple situations that repeat themselves frequently. In order to obtain a sufficient number of non-standard situations, a relatively large test case file needs to be compiled | Relatively small as it may be possible to avoid repetitions of any given combination of parameters |
| Efforts required to perform the software tests | High efforts (low efficiency) as tests must be carried out for large test case files. The low efficiency stems from the repetitiveness of case conditions, especially for the simple situations typical to most real-life case files | Low efforts (high efficiency) due to the relatively small test case file compiled so as to avoid repetitions |
| Effectiveness – probability of error detection | ■ Relatively low – unless the test case files are very large – due to the low percentage of uncommon combinations of parameters<br>■ No coverage of erroneous situations<br>■ Some ability to identify unexpected errors for unlisted situations | ■ Relatively high due to good coverage by design<br>■ Good coverage of erroneous situations by test case file design<br>■ Little possibility of identifying unexpected errors as all test cases are designed according to predefined parameters |

# Stratified Sampling

Using a stratified sampling procedure we can achieve substantial improvement in the efficiency of random sampling of test cases, rather than using standard random sampling of the entire population.

Stratified sampling allows us to break down the random sample into sub-populations of test cases,

- Thereby reducing the proportion of the majority “regular” population tested
- while increasing the sampling proportion of small populations and high potential error populations.

This method application minimizes the number of repetitions at the same time that it improves coverage of less frequent and rare conditions.

# Automated Testing

Automated testing represents an additional step in the integration of computerized tools into the process of software development.

These tools have joined computer aided software engineering (CASE) tools in performing a growing share of software analysis and design tasks.

Motivating Factors :

- anticipated cost savings
- shortened test duration
- heightened thoroughness of the tests performed, improvement of test accuracy
- improvement of result reporting
- statistical processing
- subsequent reporting

# The Process of Automated Testing

- Test planning
- Test design
- Test case preparation
- Test performance
- Test log and report preparation
- Re-testing after correction of detected errors (regression tests)
- Final test log and report preparation including comparison reports

# Types of Automated Testing

- Code auditing
- Coverage monitoring
- Functional tests
- Load tests
- Test management

# Code Auditing

Performs automated qualification testing

The computerized code auditors checks the compliance of code to specified standards and procedures of coding.

Auditor’s report includes a list of the deviations from the standards and a statistical summary of the findings.

# Coverage Monitoring

Coverage monitors produce reports about the line coverage achieved by implementing a given test case file.

It is a measurement of how many lines, statements, or blocks of your code are tested using your suite of automated tests.

It’s an essential metric to understand the quality of your QA efforts.

Code coverage shows you how much of your application is not covered by automated tests and is therefore vulnerable to defects.

Code coverage is typically measured in percentage values – the closer to 100%, the better.

# Functional Tests

Automated functional tests replace manual black-box correctness tests.

Before testing, the test cases are recorded in the test case database, and then we execute the tests.

The first test runs as well as the regression test runs, applied with the same test files and the same test cases, are performed by a computer program that replaces the “classic” tester.

# Load Tests

The load tests are based on simulated scenarios of maximal load situations the software system will confront.

An automated testing system enables measurement of the expected performance of the software system under various load levels.

# Test Management

The large workload involved in the testing phase (many participants detecting and correcting of errors, testing all items on test case files) makes timetable follow-ups important.

The main objectives of these automated tools are, to provide comprehensive follow-up and reporting of the testing and correction of detected errors

**Frame 10.6** **Automated test management packages – main features**

| Type of feature | Automated/manual testing |
|---|---|
| ***A. Test plans, test results and correction follow-up*** | |
| Preparation of lists, tables and visual presentations of test plans | A, M |
| List of test case | A, M |
| Listing of detected errors | A, M |
| Listing of correction schedule (performer, date of completion, etc.) | A, M |
| Listing of uncompleted corrections for follow-up | A, M |
| Error tracking: detection, correction and regression tests | A, M |
| Summary reports of testing and error correction follow-up | A, M |
| ***B. Test execution*** | |
| Execution of automated software tests | A |
| **Automated** listing of automated software test results | A |
| **Automated** listing of detected errors | A |
| ***C. Maintenance follow-up*** | |
| Correction of errors reported by users | A, M |
| Summary reports for maintenance correction services according to customer, software system applications, etc. | A, M |

# Alpha and Beta site testing programs

- Alpha site and beta site tests are done to obtain comments about quality from the package's potential users.

They are additional commonly used tools to identify software design and code errors in software packages in commercial over-the-counter sale (COTS).

Alpha and beta site tests could replace the customer's acceptance test, which is impractical under the conditions of commercial software package development.

However in no case should they replace the formal software tests performed by the developer.

# Alpha Site Testing

Alpha site testing is a method by which customers try out the new software package at the developer's site.

The customer, by applying the new software to the specific requirements of his organization, examines the package from angles not expected by the testing team.

The errors identified by alpha site tests are expected to include the errors that only use by a real user can reveal, and should be reported to the developer.

# Beta Site Testing

Beta site testing is where a selected group of users or customers receive an advanced version of the software to be installed in their sites, and report the errors they find.

Beta site tests are much more commonly applied than are alpha site tests.

Because beta site tests are considered to be a valuable tool, some developers involve hundreds or even thousands of participants (usually users of previously released packages, sophisticated software professionals etc.) in the process.

# Exercise

1. What are the advantages and disadvantages of automated testing?
2. What are the advantages and disadvantages of alpha testing?
3. What are the advantages and disadvantages of beta testing?
4. Compare and contrast alpha testing and beta testing.