Determining the test methodology

- Appropriate required software quality standard.
  e.g.: 1. software package for hospital patient bed monitor requires highest quality standard.
  2. A package developed for handling feedback information for an organization’s internal employee training program could make do with a medium-level software quality standard.

Software testing strategy

- Big bang
- Incremental
- Which part should be performed according to white box testing?
- Which part should be performed according to the automated testing model?
Planning the test

- **Unit tests** – deal with small units of software or modules, e.g: function, procedure, method. The goal of unit testing is to isolate each part of the program and show that the individual parts are correct.
- **Integration tests** – deal with several units that combine into a subsystem. To make sure that the interaction of two or more components produces results that satisfy functional requirement, e.g. receive data from different components and how to pass data to different components.
- **System tests** – deals with entire software system. Defects found during the system testing are either fixed after doing thorough impact analysis or are documented as known limitations.

Assessing Testing effort

- How to know when testing can stop?
- How to assess testing success at removing defects?
- Examples of approaches:
  - Independent test groups
  - Fault seeding technique
  - Mutant-based testing
  - Defect plotting

Estimation using Independent Test Groups

- Evaluate how many defects are in a software product and the efficiency of defect removal
  - Uses independent groups testing same program
  - Example: 2 groups
  - \( x \) = number of faults detected by Test Group 1
  - \( y \) = number of faults detected by Test Group 2
  - \( q \) = number of common faults detected by both Test Groups
  - \( n \) = total number of faults in the program
  - \( E1 \) effectiveness of Group 1: \( x/n \)
  - \( E2 \) effectiveness of Group 2: \( y/n \)
  - \( n = (x \times y)/q = q/(E1 \times E2) \)

Fault seeding technique

- To estimate the number of faults in a program
- Before testing, seed program with a number of typical faults
- After a period of testing, compare the number of seeded and non seeded faults detected
  - \( N \) = number of non seeded (unintentional) faults
  - \( S \) = number of seeded faults placed in the program
  - \( n \) = actual number of faults detected during testing
  - \( s \) = number of seeded faults detected during testing
  - \( N = S \times (n/s) \)
**Mutant-based testing**

- To assess the effectiveness of a test suite for defects detection
- A version of a program obtained by replicating the original program except for one small change (a mutation)
- Corresponds to a typical error
- Examples of mutations:
  - Value – change constants, subscripts, parameters by adding/subtracting one, etc.
  - Decision – modify the sense of decisions (e.g., `<` to `>`
  - Statement – delete/exchange statements
- Mutant killed by a test suite if it is revealed (fail)
- Kill ratio of a test suite = # mutant killed/# of mutants
- Higher the kill ratio, better the test suite is
- Needs tool support (e.g., Mothra)

**Defect Plotting**

- To help decide when to stop testing
- Plot number of defects found per period of testing
- Graph supposed to peak, then drop, and plateau

**Test Implementation Phase**

**Automated testing**

- Additional step in integration of computerized tools into the process of software development.
- Advantages – cost saving, shortened test duration, heightened thoroughness of the tests performed, improve test accuracy, improving of result.
Types of automated tests

- **Code auditing** – coding style follow coding style procedure? e.g. – naming conventions for variables, files.
- Auditor’s report includes a list of deviations from the standards and statistical summary of the findings.

- **Coverage monitoring** – produce reports about line coverage achieved when implementing a given test case file as well as listing of uncovered lines.

- **Functional tests** – replace manual black-box correctness tests.
  - **Load tests** – An automated testing system enables measurement of the expected performance of the software system under various load levels.
  - **Test management** – provide follow-up and reporting of the testing and correction of detected errors.

Advantages of automated tests

- Accuracy and completeness of performance
- Accuracy of results log and summary reports
- Obtain much more comprehensive information
- Performance requires few manpower resources
- Performance of complete regression tests
- Performance of test classes beyond the reach of manual testing.
Disadvantages of automated tests

- High investment in package purchasing and training
- High manpower tester resources for preparing the tests
- Considerable testing areas not covered by automated testing

Alpha testing and beta site testing programs

- Objective: to obtain comments about quality from the package’s potential users.
- Alpha testing – perform at developer’s site.
- Beta testing – advance version of software will be offered free of charge to potential users. Users will test at their site. Involve many participants.

Advantages of beta site testing

- Identification of unexpected errors
- A wider population in search of errors
- Low costs

Disadvantages of beta site tests

- Lack of systematic testing
- Low quality of error reports
- Difficult to produce the best environment
- Much effort is required to examine reports