Software Testing And Quality Control

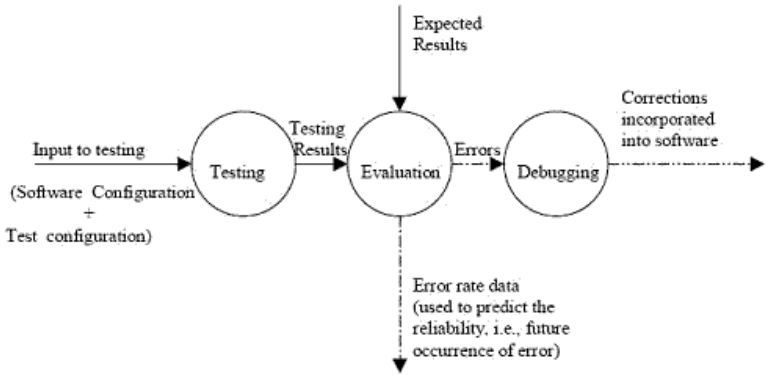
Mr. GD Makkar
HOD, Dept. of CA & IT
SGRRITS
Dehradun

- Testing is basically a process to detect errors in the software product.
- This process helps to identify the correctness, completeness and quality of developed computer software.
- Testing is the comparison between the actual values and expected one and the difference between actual and expected value is called error.
- These errors can due to wrong analysis, wrong design, or some fault on developer's part.
- All these errors need to be discovered before the system is implemented at the customer's site.
- Testing is a way to know about quality and reliability.

Test Information Flow

For testing two types of inputs are required.

1. Software configuration:
   - It includes software requirement specification, design specifications and source code of program.
   - Software configuration is required so that the testers know what is to be expected and tested
2. Test configuration:
   - It is basically test plan and procedure.
   - Test configuration is testing plan that is, the way how the testing will be conducted on the system. It specifies the test cases and their expected value.
   - Test cases are required to know what specific situations need to be tested. When tests are evaluated, test results are compared with actual results and if there is some error, then debugging is done to correct the error.



Testing Process

Test Case Design

Test cases are integral part of testing. The most desired expectation from a test case is that it should be able to find most errors with the least amount of time and effort. A software product can be tested in two ways.

- White Box testing
- Black Box testing

Black Box Testing

- **Testing of a function without knowing internal structure of the program.**
- In this test case overall functioning of the product is tested. Inputs are supplied to product and outputs are verified. If the outputs obtained are same as the expected ones then the product meets the functional requirements.
- It does not care about the internal functioning of the product. It treats the system as a “black – box”.
- Black – box test design is usually focused on testing functional requirements.
- Black box testing is also known as **behavioral, functional, and opaque – box** and **closed – box** testing.

Black box testing uncovers following types of errors:

- Incorrect or missing functions
- Interface errors
- External database access
- Performance errors
- Initialization and termination errors

White Box Testing

- **Testing of a function with knowing internal structure of the program.**
- In this test case, internal functioning of the product is tested. Each procedure is tested for its accuracy.
- It is more intensive than black box testing. Unlike black box testing, white box testing uses specific knowledge of programming code to examine outputs.
- The test is accurate only if the tester knows what program is supposed to do.
- White Box testing is also known as **glass box, Structural, clean box** and **open box** testing.

TYPES OF TEST DATA

There are two sources of test data are used for testing of the software:

- Live data
- Artificial / Dummy data

Live Data

- Live data are those that are actually extracted from organization files. This shows how the system will perform on typical data.
- All combinations and conditions of the system cannot be tested with this data. This may not contain values that may cause system failure.

Artificial / Dummy Data

- Artificial data are used to test all combinations of formats and values.
- They are generated using the utility programs of the information system.
- Using this type of data all logic and control paths through the program can be tested.
- For best results, the artificial test data should be generated by persons other than those who wrote the program.

LEVELS OF TESTING

Testing is done at 3 levels:

1. Unit Testing
2. Integrating Testing
3. System Testing

Unit Testing

- Smallest unit of software design is a module. Unit testing is performed to check the functionality of these units.
- It is done before modules are integrated together to build the overall system. Unit testing is always done by the programmer itself to test their respective module.
- Unit testing is basically white box oriented.

The following are the tests that are performed during the unit testing:

•**Module interface test:** This test is used to check if the information is properly flowing into the program unit and properly coming out of it.

•**Local data structures:** These are tested to see if the local data within unit (module) is stored properly by them.

•**Independent paths:** All independent paths are tested to see that they are properly executing their task and terminating at the end of the program.

•**Error handling paths:** These are tested to check if errors are handled properly by them.

Integration Testing

- Unit testing ensures that all modules have been tested and each of them works properly individually.
- Unit testing does not guarantee if these modules will work fine if these are integrated together as a whole system.
- It is observed that many errors crop up when the modules are joined together.
- Integration testing uncovers errors that arise when modules are integrated to build the overall system.
- The objective of integration testing is to take unit tested modules, integrate them, find errors, remove them and build the overall program structure as specified by design.

Following types of errors may arise in the integration testing:

- Data can be lost across an interface. That is data coming out of a module is not going into the desired module.
- Sub-functions, when combined, may not produce the desired major function.
- Global data structures can present problems: For example, in a system there is a global memory. Now these modules are combined. All are accessing the same global memory. Because so many functions are accessing that memory, low memory problem can arise.

- Integration Testing is also called **Verification testing**. Verification testing runs the system in a simulated environment using simulated data.
- This simulated test is sometimes called **alpha testing.**

System Testing

- System testing is conducted on a complete, integrated system to evaluate the system compliance with its specified user requirements.
- When system is tested in a live environment using real data, then it is called **Validation Testing.** This is sometimes called **Beta Testing.**

During the system / validation testing, following items are tested:

1. System Performance
2. Peak workload processing performance
3. Human engineering test
4. Methods and procedure test
5. Storage testing
6. Back up and Recovery testing
7. Security testing

- **System Performance:**

  This test refers to the throughput and response time of the system being installed, whether it meets a normal processing workload.

- **Peak Workload processing performance:**

  This is used to determine whether the system can handle the workload during peak processing period.

  This test applies mainly for on – line system. E.g.: Banking System, Railway Reservation System.

- **Human Engineering Test:**

  This is used to determine whether the system is easy to learn and use. If not, what enhancement can be made to make the system user friendly.

  This test includes the screen designing, and error messages or other messages or other messages to be displayed for the user help.

- **Methods and Procedure Test:**

  During conversion, the methods and procedures for the new system will be put to work for their first real test, methods and procedures may have to be modified if they are prove to be inefficient from the end – user's stand point.

- **Storage Testing:**
  This test is to be carried out to determine the capacity of the system to store transaction data on a disk or in other files. Capacities are measured in terms of the number of records that a disk will handle or a file can contain.

- **Backup and Recovery Testing:**
  In this testing, all backup and recovery procedures should be tested. This should be including simulating a data loss disaster and testing the time required to recover from that disaster. Also, a before – and – after comparisons of the data should be performed to ensure that data was properly recovered.

- **Security Testing**

  - Security testing verify the protection mechanism built into a system against the unauthorized access.
  - During security testing, the tester attacks the system for unauthorized access to break down any defenses that have been constructed; may overwhelm the system, thereby denying service to others; may purposely cause system errors, hoping to find the key to system entry; and so on.

QUALITY ASSURANCE

- Quality assurance is the review of software products and related documentation for completeness, reliability and maintainability so that errors are corrected early in the development process. Actions are taken in each phase to ensure that there are no errors in the final software.
- For quality, **Software Engineering Institute (SEI)** Carnegie Mellon University, Pennsylvania has devised a quality model known as **Capability Maturity Model (CMM)**. This model is based on the principles of **Total Quality Management (TQM)** for improving the entire software development process.

Levels of CMM

LEVEL 1: ***Initial*** - processes are not adequately defined and documented.

LEVEL 2: ***Repeatable*** - implementation of project management tools.

LEVEL 3: ***Defined*** - all projects use an organized, documented and standardized set of activities that are implemented throughout the project life cycle.

LEVEL 4: ***Managed*** - Specific process and quality outcome measures are implemented.

LEVEL 5: ***Optimized*** - Continuous process improvement is adopted in the entire organization. Sophisticated methods of defect prevention, technology changes management and process change management is implemented.

QUALTY SOFTWARE SPECIFICATIONS

Portability	The ease with which software can be transferred from one hardware configuration to another.
Reliability	The ability of a program to perform a required function under stated condition for a stated period of times.
Efficiency	The amount of computer resources required by a program to perform a function.
Accuracy	The required precision in input, computations and output.
Error	A discrepancy between computed values and the true value.

Robustness	The extent to which software can continue to operate correctly despite of the introduction of invalid input.
Correctness	• The extent to which software is free from design defects and from coding defects. • The extent to which software meets its specified requirements. • The extent to which software meets user expectation.
Usability	The effort required to learn and operate a system.
Maintainability	The ease with which program errors are located and corrected

Testability	The effort required to test a program to ensure its correct performance.
Expandability	Ease of adding or expanding the existing database, procedure or module.
Access control	Control of access to the system and the extent to which that access can be audited.
Communicativeness	How much input and output of the system are useful.