TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In the process of software development, errors can occur at any stage and during any phase. The errors encountered during the earlier phases of Software Development Life Cycle (SDLC) such as requirements, design phases are likely to be carried to the coding phase also, in addition to the errors generated at the coding phase. This is particularly true because, in earlier phases most of the verification techniques are manual and no executable code exists. Because code is the only product that can be executed and its actual behavior can be observed, testing is the phase where errors remaining from all the phases must be detected. Hence, testing performs a very critical role for quality assurance and for ensuring the reliability of software. In this chapter, we discuss software testing fundamentals, techniques for software test case design and different strategies for software testing.

Software testing is the process used to assess the quality of computer software. Software testing is an empirical technical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding software bugs. Quality is not an absolute; it is value to some person. With that in mind, testing can never completely establish the correctness of arbitrary computer software; testing furnishes a criticism or comparison that compares the state and behaviour of the product against a specification. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing.

Over its existence, computer software has continued to grow in complexity and size. Every software product has a target audience. For example, a video game software has its audience completely different from banking software. Therefore, when an organization develops or otherwise invests in a software product, it presumably must assess whether the software product will be acceptable to its end users, its target audience, its purchasers, and other stakeholders. Software testing is the process of attempting to make this assessment.

Software testing may be viewed as an important part of the software quality assurance (SQA) process. In SQA, software process specialists and auditors take a broader view on software and its development. They examine and change the software engineering process itself to reduce the amount of faults that end up in defect rate. What constitutes an acceptable defect rate depends on the nature of the software. An arcade video game designed to simulate flying an airoplane would presumably have a much higher tolerance for defects than software used to control an actual airliner. Although there are close links with SQA testing departments often exist independently, and there may be no SQA areas in some companies.

Software testing is used in association with verification and validation:

- **Verification :** Have we built the software right (*i.e.,* does it match the specification)?
- **Validation :** Have we built the right software (*i.e.,* is this what the customer wants)?

Software testing can be done by software testers. Until the 1950s the term software tester was used generally, but later it was also seen as a separate profession. Regarding the periods and the different goals in software testing, there have been established different roles: test lead/manager, test designer, tester, test automater/automation developer, and test administrator.

Simply stated, quality is very important. Many companies have not learned that quality is important and deliver more claimed functionality but at a lower quality level. It is much easier to explain to a customer why there is a missing feature than to explain to a customer why the product lacks quality. A customer satisfied with the quality of a product will remain loyal and wait for new functionality in the next version. Quality is a distinguishing attribute of a system indicating the degree of excellence.

Table 7.1. The Testing Phase: Improve Quality

Phase	Deliverable
Testing	▪ Regression Test ▪ Internal Testing ▪ Unit Testing ▪ Application Testing ▪ Stress Testing

The testing phase is summarized in Table 7.1.

In many software engineering methodologies, the testing phase is a separate phase which is performed by a different team after the implementation is completed. There is merit in this approach; it is hard to see one's own mistakes, and a fresh eye can discover obvious errors much faster than the person who has read and re-read the material many times. Unfortunately, delegating testing to another team leads to a slack attitude regarding quality by the implementation team.

Alternatively, another approach is to delegate testing to the the whole organization. If the teams are to be known as craftsmen, then the teams should be responsible for establishing high quality across all phases. Sometimes, an attitude change must take place to guarantee quality.

Regardless if testing is done after-the-fact or continuously, testing is usually based on a regression technique split into several major focuses, namely internal, unit, application, and stress.

7.1 TESTING PRINCIPLES

Software testing is an extremely creative and intellectually challenging task. When testing follows the principles given below, the creative element of test design and execution rivals any of the preceding software development steps.

- **Testing must be done by an independent party.**

  Testing should not be performed by the person or team that developed the software since they tend to defend the correctness of the program.
- **Assign best personnel to the task.**

  Because testing requires high creativity and responsibility only the best personnel must be assigned to design, implement, and analyze test cases, test data and test results.
- **Testing should not be planned under the tacit assumption that no errors will be found.**
- **Test for invalid and unexpected input conditions as well as valid conditions.**

  The program should generate correct messages when an invalid test is encountered and should generate correct results when the test is valid.
- **The probability of the existence of more errors in a module or group of modules is directly proportional to the number of errors already found.**
- **Testing is the process of executing software with the intent of finding errors.**
- **Keep software static during test.**

  The program must not be modified during the implementation of the set of designed test cases.
- **Document test cases and test results.**
- **Provide expected test results if possible.**

  A necessary part of test documentation is the specification of expected results, even if providing such results is impractical.

7.2 TEST CASE DESIGN

7.2.1 Functional Testing

This is also called as Black box testing. It takes an external perspective of the test object to derive test cases. These tests can be functional or non-functional, though usually functional. The test designer selects valid and invalid input and determines the correct output. There is no knowledge of the test object's internal structure.

This method of test design is applicable to all levels of software testing: unit, integration, functional testing, system and acceptance. The higher the level, and hence the bigger and more complex the box, the more one is forced to use black box testing to simplify. While this method can uncover unimplemented parts of the specification, one cannot be sure that all existent paths are tested.

7.2.2 Structural Testing

A complementary approach to functional or black box testing is called structural or white box testing. In this approach, test group must have complete knowledge about the internal structure of the software. We can say structural testing is an approach to testing where the tests are derived from knowledge of the software structures and implementation. Structural testing is usually applied to relatively small program units such as subroutines the operations associated with an object. As the name implies, the tester can analyze the code and use knowledge about the structure of a component to derive test data.

Structural testing may find these errors which have been missed by functional testing. Structural testing has received the most attention in terms of testing research and there are numbers of methods associated with it.

- Path testing
- Data flow testing
- Mutation testing
- Control Structure testing

***Path testing:** Path testing has been one of the first test methods, and even though it is a typical white box test, it is nowadays also used in black box tests. The procedure itself is similar to the walk-through. First, a certain path through the program is chosen. Possible inputs and the correct result are written down. Then the program is executed by hand, and its result is compared to the predefined. Possible faults have to be written down at once.

There are different strategies about how to choose the path. The optimum would certainly be to follow each path through the module once, but this is not possible:

Assume there is a little program which consists of a loop that is executed twenty times. Inside this loop are a case or some if instructions which allow ten different routes. The total number of paths through this little program would be 1020. Definitely nobody is able to trace this huge amount by hand.

Since the best solution is not possible, it is necessary to reduce the number of possibilities. There is no perfect solution to this problem, and those strategies should only be understood as hints for the "scout" — a human tester:

7.4 VERIFICATION AND VALIDATION

Verification is a quality process that is used to evaluate whether or not a product, service, or system complies with a regulation, specification, or conditions imposed at the start of a development phase. Verification can be in development, scale-up, or production. This is often an internal process.

Validation is the process of establishing documented evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements. This often involves acceptance and suitability with external customers.

It is sometimes said that validation ensures that 'we built the right thing' and verification ensures that 'we built it right'. 'Building the right thing' refers back to the user's needs; while 'building it right' checks that the documented development process was followed. In some contexts, it is required to have written requirements for both as well as formal procedures or protocols for determining compliance.

V&V is intended to be a systematic and technical evaluation of software and associated products of the development and maintenance processes. Reviews and tests are done at the end of each phase of the development process to ensure software requirements are complete and testable and that design, code, documentation, and data satisfy those requirements.

7.4.1 Verification and Validation Planning

V&V is an expensive process. For some large systems, such as real-time systems with complex non-functional constraints, half the system development budget may be spent on V & V. Careful planning is needed to get the most out of inspections and testing and to control the costs of the verification and validation process. The planning of validation and verification of a software system should start early in the development process. Test planning is concerned with setting out standards for the testing process rather than describing product tests. Test plans also provide the information to staff to get an overall picture of the system tests and to place their own work in this context. Test plans also provide information who is responsible for ensuring that appropriate hardware and software resources are available to the testing team.

The structure of software test plan:

- The testing process
- Requirements traceability
- Tested items
- Testing schedule
- Test recording procedures
- Hardware and software requirements
- Constraints

Like other plans, the test plan is not a static document. It should be revised regularly as testing is an activity that is dependent on implementation being complete. If a part of the system is incomplete, it can not be delivered for integration testing.