SOFTWARE TESTING BASICS

Software testing is a process, which is used to identify the correctness, completeness and quality of software. IEEE defines testing as "*the process of exercising or evaluating a system or system component by manual or automated means to verify that it satisfies specified requirements or to identify differences between expected and actual results*."

Software testing is often used in association with the terms verification and validation.

Verification refers to checking or testing of items, including software, for conformance and consistency with an associated specification. For verification, techniques like reviews, analysis, inspections and walkthroughs are commonly used. While **validation** refers to the process of checking that the developed software is according the requirements specified by the user. Verification and validation can be summarised as follows:

Verification: Are we developing the **software right**?

Validation: Are we developing the **right software**?

(*a*) *Objectives of Software Testing*: Software testing evaluates software by manual and automated means to ensure that it is functioning in accordance with user requirements.

The main objectives of software testing are listed below:

• To remove errors, which prevent software from producing outputs according to user require-
-ments?

• To remove errors that lead to software failure.

• To determines whether system meets business and user needs.

• To ensure that software is developed according to user requirements.

• To improve the quality of software by removing maximum possible errors from it.

(*b*) *Testing in Software Development Life Cycle (SDLC)*: Software testing comprises of a set of activities, which are planned before testing begins. These activities are carried out for detecting errors that occur during various phases of SDLC. The role of testing in software development life cycle is listed in Table.

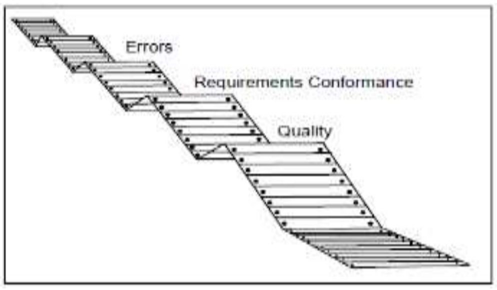
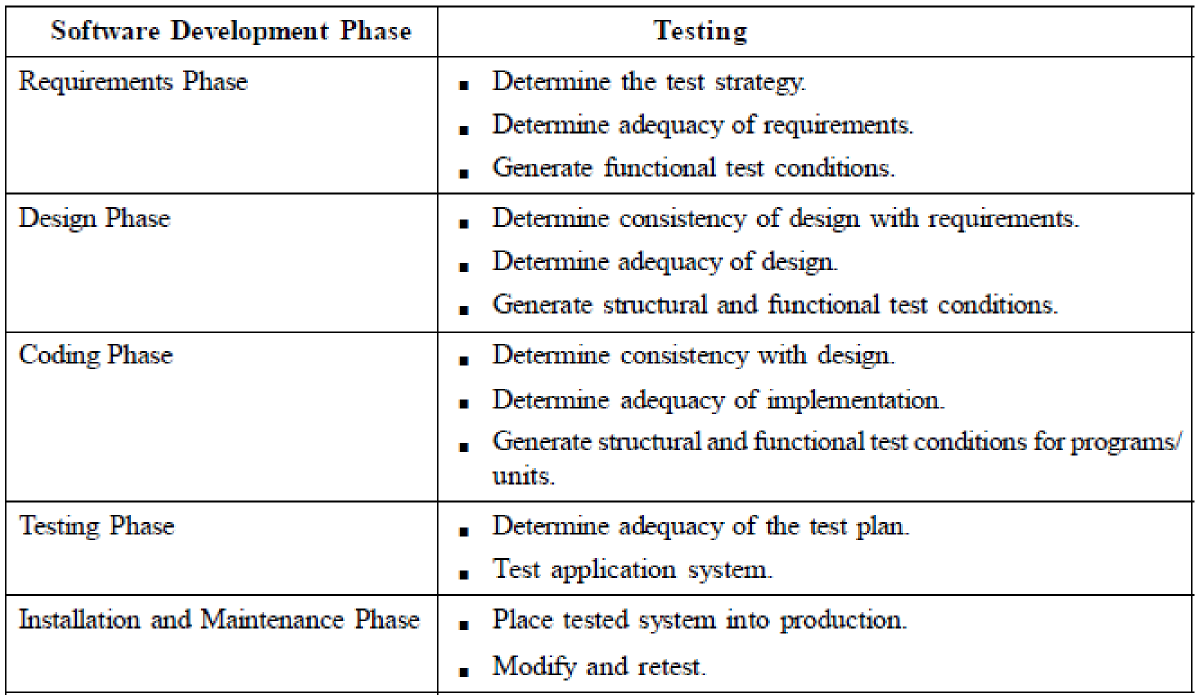


FIG: Objectives of Software Testing

Table: Role of Testing in Various Phases of SDLC



Software Development Phase	Testing
Requirements Phase	▪ Determine the test strategy. ▪ Determine adequacy of requirements. ▪ Generate functional test conditions.
Design Phase	▪ Determine consistency of design with requirements. ▪ Determine adequacy of design. ▪ Generate structural and functional test conditions.
Coding Phase	▪ Determine consistency with design. ▪ Determine adequacy of implementation. ▪ Generate structural and functional test conditions for programs/ units.
Testing Phase	▪ Determine adequacy of the test plan. ▪ Test application system.
Installation and Maintenance Phase	▪ Place tested system into production. ▪ Modify and retest.

(*c*) ***Bugs, Error, Fault and Failure*****:** The purpose of software testing is to find bugs, errors, faults, and failures present in the software. **Bug** is defined as a logical mistake, which is caused by a software developer while writing the software code. **Error** is defined as the difference between the outputs produced by the software and the output desired by the user (expected output). **Fault** is defined as the condition that leads to malfunctioning of the software. Malfunctioning of software is caused due to several reasons, such as change in the design, architecture, or software code. Defect that causes error in operation or negative impact is called failure. **Failure** is defined as the state in which software is unable to perform a function according to user requirements. Bugs, errors, faults, and failures prevent software from performing efficiently and hence, cause the software to produce unexpected outputs. Errors can be present in the software due to the reasons listed below:

• **Programming errors:** Programmers can make mistakes while developing the source code.

• **Unclear requirements:** The user is not clear about the desired requirements or the developers are unable to understand the user requirements in a clear and concise manner.

• **Software complexity:** The complexity of current software can be difficult to comprehend for someone who does not have prior experience in software development.

• **Changing requirements:** The user may not understand the effects of change. If there are minor changes or major changes, known and unknown dependencies among parts of the project are likely to interact and cause problems. This may lead to complexity of keeping track of changes and ultimately may result in errors.

• **Time pressures:** Maintaining schedule of software projects is difficult. When deadlines are not met, the attempt to speed up the work causes errors.

• **Poorly documented code:** It is difficult to maintain and modify code that is badly written or poorly documented. This causes errors to occur.

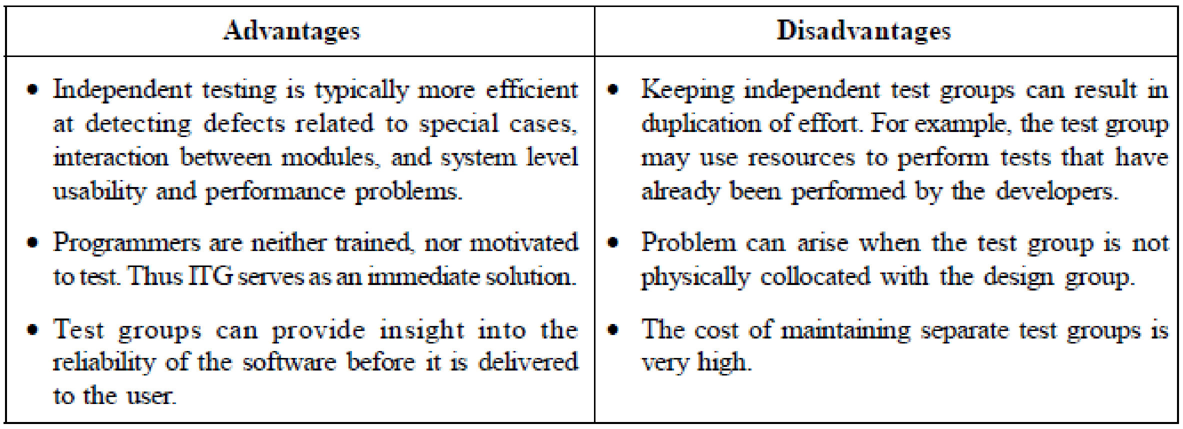
(*d*) *Who Performs Testing*?: Testing is an organisational issue, which is performed either by the software developers (who originally developed the software) or by an *independent test group* (ITG), which comprises of software testers. The software developers are considered to be the best persons to perform testing as they have the best knowledge about the software. However, since software developers are involved in the development process, they may have their own interest to show that the software is error free, meets user requirements, and is within schedule and budget. This vested interest hinders the process of testing.

To avoid this problem, the task of testing is assigned to an independent test group (ITG), which is responsible to detect errors that may have been neglected by the software developers. ITG tests the software without any discrimination since the group is not directly involved in the development process. However, the testing group does not completely take over the testing process, instead it works with the software developers in the software project to ensure that testing is performed in an efficient manner. During the testing process, developers are responsible for correcting the errors uncovered by the testing group.

Generally, an independent testing group forms a part of the software development project team. This is because the group becomes involved during the specification activity and stays involved (planning and specifying test procedures) throughout the development process.

• The various advantages and disadvantages associated with independent testing group
are listed in Table.

Table: Advantages and Disadvantages of Independent Test Group



Advantages	Disadvantages
• Independent testing is typically more efficient at detecting defects related to special cases, interaction between modules, and system level usability and performance problems.	• Keeping independent test groups can result in duplication of effort. For example, the test group may use resources to perform tests that have already been performed by the developers.
• Programmers are neither trained, nor motivated to test. Thus ITG serves as an immediate solution.	• Problem can arise when the test group is not physically collocated with the design group.
• Test groups can provide insight into the reliability of the software before it is delivered to the user.	• The cost of maintaining separate test groups is very high.

To plan and perform testing, software testers should have the knowledge about the function for which the software has been developed, the inputs and how they can be combined, and the environment in which the software will eventually operate. This process is time-consuming and requires technical sophistication and proper planning on the part of the testers.

To achieve technical know-how, testers must not only have good development skills but also possess knowledge about formal languages, graph theory, and algorithms.

Other factors that should be kept in mind while performing testing are:

- Time available to perform testing.
- Training required acquainting testers about the software.
- Attitude of testers.
- Relationship between testers and developers.