K.T.S.P. MANDAL'S

HUTUTMA RAJGURU MAHAVIDYALAYA,

RAJGURUNAGAR TAL-KHED, DIST-PUNE 410 505

DEPARTMENT OF COMPUTER SCIENCE

F.Y.Bsc (Computer Science)

Semester- II

Software Testing

Subject – CS-362 -Software Testing

According to New Syllabus w.e.f. 2021-2022

Prof. Pallavi G. Darakhe

DEPARTMENT OF COMPUTER SCIENCE

Chapter 1st

Introduction to Software Testing

Software Testing **is a method to assess the functionality of the software program.** The process checks whether the actual software matches the expected requirements and ensures the software is bug-free. The purpose of software testing is to identify the errors, faults, or missing requirements in contrast to actual requirements. It mainly aims at measuring the specification, functionality, and performance of a software program or application

In software testing, we commonly use the term defect, bug, error, and failure to represent various scenarios in the testing process.

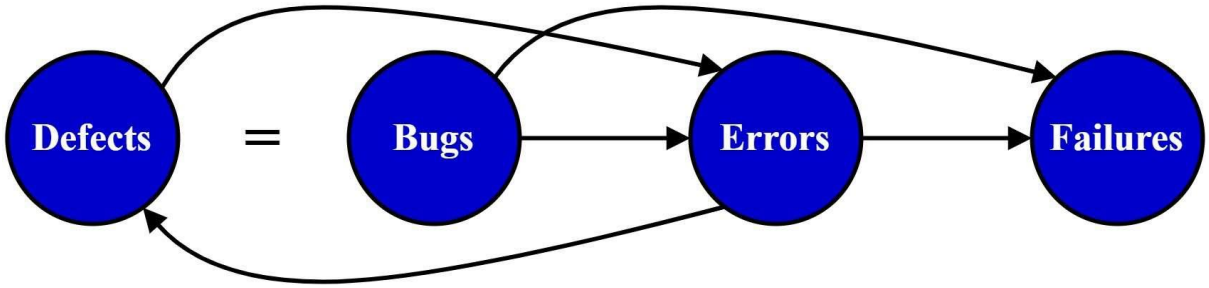
Basics of Software Testing- Faults, Errors and Failures

A Defect is a deviation from the expected behaviour in a software application, and it arises due to flaws in the code or design. In other words, a defect is a potential issue identified during testing.

We often use the term **"bug"** interchangeably with the defect. **While a defect is a more formal term used in the testing domain, a bug has become a usual way of referring to defects.**

An Error is a mistake a coder makes during the software development process. It occurs when a programmer or developer introduces a flaw in the code or logic, leading to unintended results when the software is executed. **Errors are human-made** and are a natural part of the software development process. These mistakes can be Syntax errors, logical errors, or other issues that affect the functionality of the software.

A Failure is when the software doesn't have an intended functionality or deliver the expected results during execution. Defects and errors have the potential to cause failures during software execution. When the software behaves unexpectedly or incorrectly while used by end-users, we consider it a failure.



We can see that defects (i.e., bugs) and errors can lead to failures, and errors can contribute to defects in the software development and testing process.

Benefits of Software Testing

1.Cost Effective

If an application works without any fault and with low maintenance will save a big amount for the owner.

- Software testing helps in early defect detection and fixing them to make a more successful and good return application.
- Every application demands maintenance and the application owner spends a good amount to maintain the working and functionality of the application.
- By testing an application, the maintenance area reduces too many folds and hence saves money.
- Software testing helps in saving the money of the developing organization by detecting defects in the early stages of the software which becomes easy and economical for the developer to redesign the module instead of detecting bugs after the complete development of the software.

2.Security

The main concern of the digital world is Security. A secure system always remains on the priority list of customers. Owners pay multiple dollars to make their systems secure from hackers, malicious attacks, and other thefts. Security testing technique is used to identify the security level of the application and testers try to find loopholes to break the security of an application.

- In the case of bank applications, security is the foremost requirement as it deals with customer money, and bank applications always remain on top of the thefts.
- The testing team uses security testing to identify defects and the development team tries to cover the application with multiple security layers.
- Thus, software testing is a must to maintain the security of an application.

3. Quality Product

The main focus of software testing is to deliver a quality product to its clients which results in a satisfied client. The quality of a product can only be maintained when it is bug-free and meet all the user requirements.

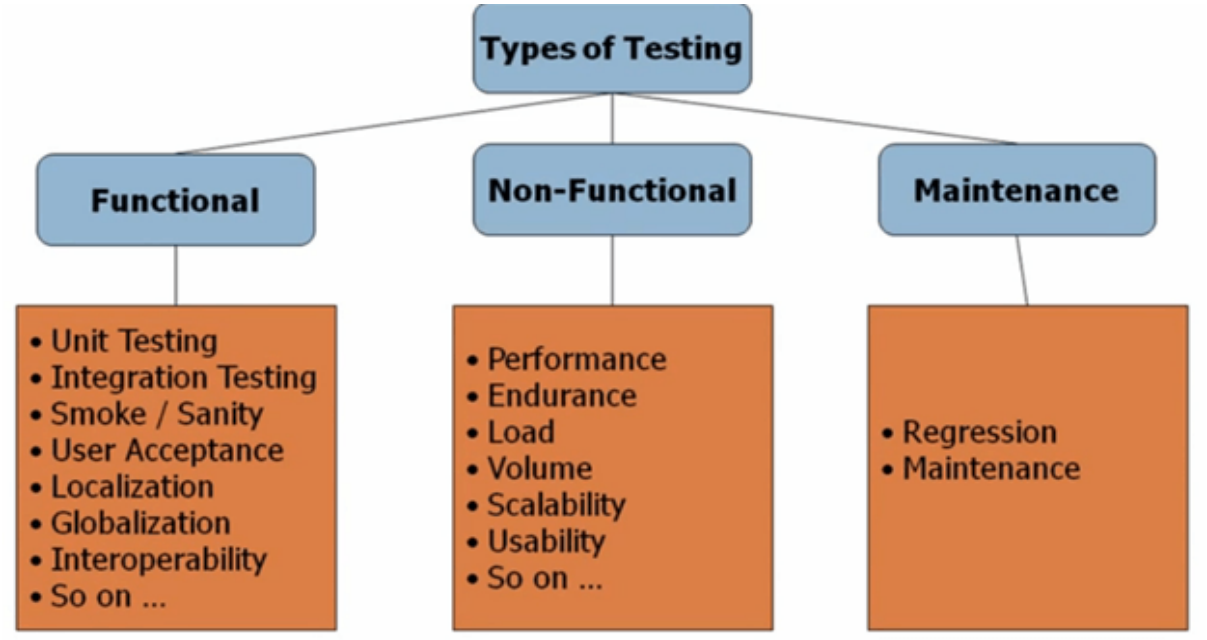
- To meet the product qualities, if an application works with other related applications then it must undergo compatibility testing.
- During software testing, an application undergoes several testing techniques to make a quality end product and the testing team tries their best by writing test cases and test scenarios to validate defect-free free applications.

4.Customer Satisfaction

The prime goal of any service-based organization is to serve the best features and experience to their customers, customer satisfaction is the only goal for the success and popularity of an application.

- Software testing helps in building the trust and satisfaction of the customer by assuring a defect-free application.
- UI testing enhances customer satisfaction. Software Testing tries to uncover all possible defects and test an application as per customer requirements.
- For example, eCommerce is dependent on the customer, and a satisfied customer will improve its market value and earnings.

👍Testing is classified into three categories



Functional Testing: This is a type of software testing that validates the software system against the functional requirements or specifications.The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation.

The prime objective of Functional testing is checking the functionalities of the software system. It mainly concentrates on –

- **Mainline functions**: Testing the main functions of an application
- **Basic Usability**: It involves basic usability testing of the system. It checks whether a user can freely navigate through the screens without any difficulties.
- **Accessibility**: Checks the accessibility of the system for the user
- **Error Conditions**: Usage of testing techniques to check for error conditions. It checks whether suitable error messages are displayed.

Non-Functional Testing or performance Testing : This is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.

- An excellent example of non-functional test would be to check how many people can simultaneously login into a software.
- Non-functional testing is equally important as functional testing and affects client satisfaction.

Maintenance Testing: It is also known as post-release **software testing**. This type of software testing occurs when the software has been released into production, and any changes have been made to fix bugs or add new features to the existing system.

Types Of Maintenance Testing

When the maintenance testers validate the application, they must consider two things. Based on the test types

Confirmation Testing: On this confirmation maintenance testing, the Testers or QA must mainly focus on the modified functionalities. They have to verify every aspect of the application is working as it should be.**Regression Testing:** Testing the existing functionality to ensure it is not broken or degraded by the new functionality.

👍Testing Strategies in Software Testing

1. **Unit testing** – Tests individual units or components of the software to ensure they are functioning as intended.
2. **Integration testing** – Tests the integration of different components of the software to ensure they work together as a system.
3. **System Testing-** is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is defined as a series of different tests whose sole purpose is to exercise the full computer-based system.
4. **Program Testing**- in software testing is a method of executing an actual software program with the aim of testing program behavior and finding errors. The software program is executed with test case data to analyse the program behavior or response to the test data. A good program testing is one which has high chances of finding bugs.

👍Testing Objectives

Verification: A brief explanation of what is verification in software testing would be: it is the process of assessing software to evaluate if the results of a certain development phase meet the requirements

established at the beginning of that phase. Software Verification Testing is the process of examining documentation, designs, code, and programs to determine whether or not the software was constructed in accordance with the requirements. Verification operations include reviews, walk-throughs, and inspections. Although verification can assist to identify whether the program is of good quality, it cannot guarantee that the system is functional. The purpose of verification is to determine if the system is well-engineered and error-free.

An example of verification in software testing

To answer the question of what verification activity is in software testing, we need to look into Static testing. Static testing techniques are a strong tool to increase software development quality and productivity by supporting engineers in identifying and correcting their own errors early in the **software development process**. This program is tested without running the code by doing review, walkthrough, inspection, or analysis, among other things.

Static testing may be done manually or using various software testing tools. It occurs early in the Software Development Life Cycle, during the Verification Process.

Static testing is not a substitute for dynamic testing, but all software companies should consider employing reviews in all key elements of their work, including requirements, design, implementation, testing, and maintenance.

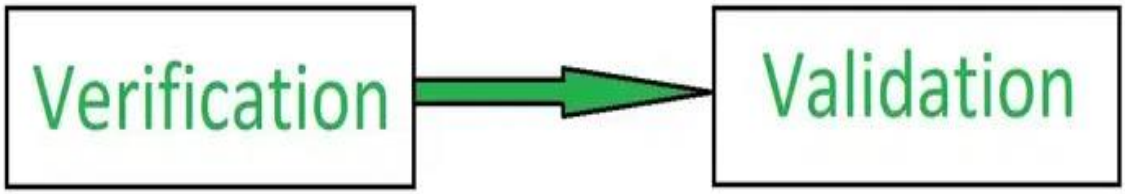
Defects found during static testing include departures from standards, missing requirements, design flaws, non-maintainable code, and conflicting interface definitions.

Validation

A short answer would be that, the method for determining if software meets stated criteria during or at the conclusion of the development process.
Validation is the process of determining if the completed product fulfills the customer's expectations and criteria. It is a dynamic validation and testing technique for the real product.

An example of validation in software testing

Software is performed using a set of input values in Dynamic Testing Technique, and its output is then reviewed and compared to what is expected. Dynamic execution is used to discover faults and assess the code's quality characteristics. Dynamic testing and static testing are complementary methodologies since they identify various sorts of faults in different ways. However, because it does not begin early in the Software Development Life Cycle, it significantly raises the cost of fault correction. It is completed during the Validation Process, which evaluates the finished product.



Verification	Validation
It includes checking documents, design, codes and programs.	It includes testing and validating the actual product.
Verification is the static testing.	Validation is the dynamic testing.

Verification	Validation
It does *not* include the execution of the code.	It includes the execution of the code.
Methods used in verification are reviews, walkthroughs, inspections and desk-checking.	Methods used in validation are Black Box Testing, White Box Testing and non-functional testing.
It checks whether the software conforms to specifications or not.	It checks whether the software meets the requirements and expectations of a customer or not.
It can find the bugs in the early stage of the development.	It can only find the bugs that could not be found by the verification process.
The goal of verification is application and software architecture and specification.	The goal of validation is an actual product.
Quality assurance team does verification.	Validation is executed on software code with the help of testing team.
It comes before validation.	It comes after verification.
It consists of checking of documents/files and is performed by human.	It consists of execution of program and is performed by computer.
Verification refers to the set of activities that ensure software correctly implements the specific function.	Validation refers to the set of activities that ensure that the software that has been built is traceable to customer requirements.
After a valid and complete specification the verification starts.	Validation begins as soon as project starts.

Verification	Validation
Verification is for prevention of errors.	Validation is for detection of errors.
Verification is also termed as white box testing or static testing as work product goes through reviews.	Validation can be termed as black box testing or dynamic testing as work product is executed.
Verification finds about 50 to 60% of the defects.	Validation finds about 20 to 30% of the defects.
Verification is based on the opinion of reviewer and may change from person to person.	Validation is based on the fact and is often stable.
Verification is about process, standard and guideline.	Validation is about the product.

Defects

A software defect is an error, flaw, failure, or fault in a computer program that causes it to produce an incorrect or unexpected result, or to behave in unintended ways. A software bug occurs when the actual results don't match with the expected results. Developers and programmers sometimes make mistakes which create bugs called defects. Most bugs come from mistakes that developers or programmer make.

7 Principles of Software Testing

1. Testing shows presence of defects
2. Exhaustive testing is not possible
3. Early testing
4. Defect clustering
5. Pesticide paradox
6. Testing is context dependent

7. Absence of errors fallacy

1) Exhaustive testing is not possible

Exhaustive testing is not possible. Instead, we need the optimal amount of testing based on the risk assessment of the application.And the million dollar question is, how do you determine this risk?To answer this let's do an exercise

In your opinion, Which operation is most likely to cause your Operating system to fail?

I am sure most of you would have guessed, Opening 10 different application all at the same time.

So if you were testing this Operating system, you would realize that defects are likely to be found in multi-tasking activity and need to be tested thoroughly which brings us to our next principle Defect Clustering

2) Defect Clustering

Defect Clustering which states that a small number of modules contain most of the defects detected. This is the application of the Pareto Principle to software testing: approximately 80% of the problems are found in 20% of the modules.By experience, you can identify such risky modules. But this approach has its own problems

If the same tests are repeated over and over again, eventually the same test cases will no longer find new bugs.

3) Pesticide Paradox

Repetitive use of the same pesticide mix to eradicate insects during farming will over time lead to the insects developing resistance to the pesticide Thereby ineffective of pesticides on insects. The same applies to software testing. If the same set of repetitive tests are conducted, the method will be useless for discovering new defects.

To overcome this, the test cases need to be regularly reviewed & revised, adding new & different test cases to help find more defects.

Testers cannot simply depend on existing test techniques. He must look out continually to improve the existing methods to make testing

more effective. But even after all this sweat & hard work in testing, you can never claim your product is bug-free. To drive home this point, let's see this video of the public launch of Windows 98

You think a company like MICROSOFT would not have tested their OS thoroughly & would risk their reputation just to see their OS crashing during its public launch!

4) Testing shows a presence of defects

Hence, testing principle states that – Testing talks about the presence of defects and don't talk about the absence of defects. i.e. Software Testing reduces the probability of undiscovered defects remaining in the software but even if no defects are found, it is not a proof of correctness.

But what if, you work extra hard, taking all precautions & make your software product 99% bug-free. And the software does not meet the needs & requirements of the clients.This leads us to our next principle, which states that- Absence of Error

5) Absence of Error – fallacy

It is possible that software which is 99% bug-free is still unusable. This can be the case if the system is tested thoroughly for the wrong requirement. Software testing is not mere finding defects, but also to check that software addresses the business needs. The absence of Error is a Fallacy i.e. Finding and fixing defects does not help if the system build is unusable and does not fulfill the user's needs & requirements.To solve this problem, the next principle of testing states that Early Testing

6) Early Testing

Early Testing – Testing should start as early as possible in the Software Development Life Cycle. So that any defects in the requirements or design phase are captured in early stages. It is much cheaper to fix a Defect in the early stages of testing. But how early one should start testing? It is recommended that you start finding the bug the moment the requirements are defined. More on this principle in a later training tutorial.

7) Testing is context dependent

Testing is context dependent which basically means that the way you test an e-commerce site will be different from the way you test a commercial off the shelf application. All the developed software's are not identical. You might use a different approach, methodologies, techniques, and types of testing depending upon the application type. For instance testing, any POS system at a retail store will be different than testing an ATM machine.

Difference between Testing and Debugging

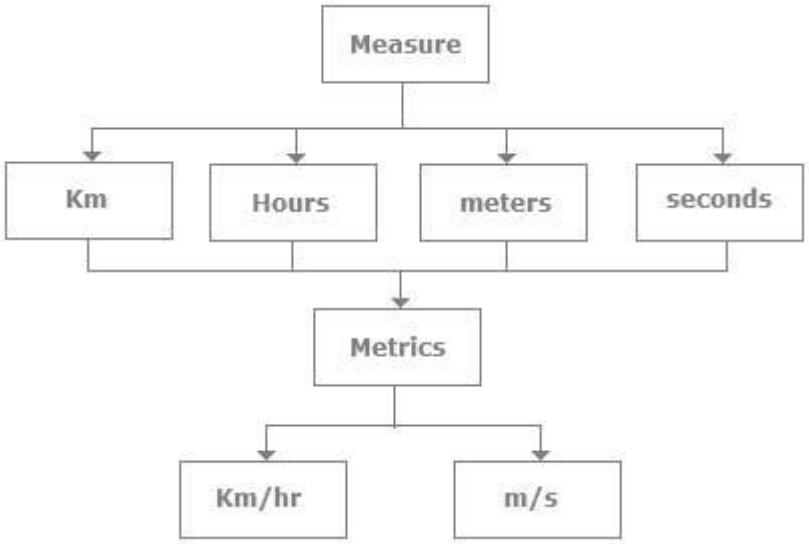
Testing	Debugging
Testing is the process to find bugs and errors.	Debugging is the process of correcting the bugs found during testing.
It is the process to identify the failure of implemented code.	It is the process to give absolution to code failure.
Testing is the display of errors.	Debugging is a deductive process.
Testing is done by the tester.	Debugging is done by either programmer or the developer.
There is no need of design knowledge in the testing process.	Debugging can't be done without proper design knowledge.

Testing	Debugging
Testing can be done by insiders as well as outsiders.	Debugging is done only by insiders. An outsider can’t do debugging.
Testing can be manual or automated.	Debugging is always manual. Debugging can’t be automated.
It is based on different testing levels i.e. unit testing, integration testing, system testing, etc.	Debugging is based on different types of bugs.
Testing is a stage of the software development life cycle (SDLC).	Debugging is not an aspect of the software development life cycle, it occurs as a consequence of testing.
Testing is composed of the validation and verification of software.	While debugging process seeks to match symptoms with cause, by that it leads to error correction.
Testing is initiated after the code is written.	Debugging commences with the execution of a test case.

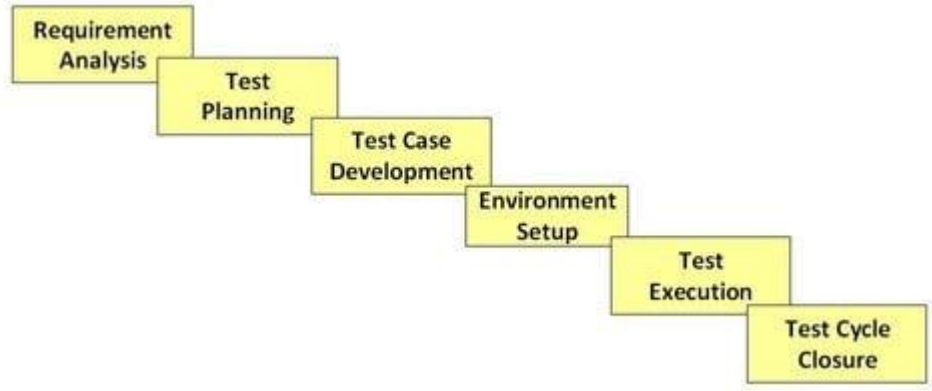
Testing	Debugging
Testing process based on various levels of testing-system testing, integration testing, unit testing, etc.	Debugging process based on various types of bugs is present in a system.

👍 Testing Metrics and measurements

Software testing metrics are quantifiable indicators of the software testing process progress, quality, productivity, and overall health. The purpose of software testing metrics is to increase the efficiency and effectiveness of the software testing process while also assisting in making better decisions for future testing by providing accurate data about the testing process. A metric expresses the degree to which a system, system component, or process possesses a certain attribute in numerical terms. A weekly mileage of an automobile compared to its ideal mileage specified by the manufacturer is an excellent illustration of metrics.



Testing Life Cycle



Software Testing Myths

Myth 1: Testing is Expensive

Reality: Testing is not cheap, but not testing is even more expensive. A study by IBM demonstrated that fixing a production defect is at least six times more costly than testing. Defects also lead to a bad customer experience and possible loss of customers.

Myth 2: Testing is Time-Consuming

Reality: Test can be time-consuming when it is considered an afterthought. Testing processes can happen parallelly during the SDLC phases. The QA team should start writing test cases and begin planning as soon as the software specification is made available. When the developers finish coding their modules, the testers can perform unit testing and integration testing. System testing indeed takes time because it is a manual process. But if you plan well, you can ensure that the release is not delayed unnecessarily.

Myth 3: Only Fully Developed Products are Tested

Reality: There is no doubt that testing relies on the source code, but testing processes and planning can begin as soon as the software specification document is available. Your QA team can start writing tests as soon as the software specification is available without waiting for the developers to complete coding.

Myth 4: Complete Testing is Possible

Reality: Software is inherently complex, and practically, no amount of testing can fully test a software application. Almost all software applications depend on other software libraries or components which means that bugs in those areas will be manifest to end-users.

Myth 5: A tested software is bug-free

Reality: This is a widespread misconception believed by clients, managers, and even novice developers. Even after an experienced tester has reviewed the application, no one can say with utter certainty that a software application is 100 percent bug-free.

Myth 6: Missed flaws are due to testers

Reality: This is not always true. Developers have a habit of changing the code without notifying the QA group because they believe it was "just a small change." Sometimes the time allocated to testing is not enough to do thorough testing. In some cases, the documentation is ambiguous and open to interpretation resulting incomplete or incorrect test cases.

Myth 7: Testers are in charge of the quality of the product

Reality: Software quality has multiple dimensions and is challenging to achieve. To be sincere about quality means dedicating enough time and resources to it. Unfortunately, in many instances, this is not the case. The testing team is often the scapegoat because it is easy to ask them: "How did you not test this basic thing?". The answer is not easy as it appears.

Myth 8: Test automation always decreases the time

Reality: Yes, automated testing can decrease time if used appropriately. In some cases, like malware and vulnerability scanning in security testing, automation can significantly reduce time without much human effort. But in most cases, automation requires a non-trivial setup, massive test data, and entering scenarios. If requirements are changing, automated testing can be slower and end up wasting even more time.

Myth 9: Anyone can test a software application

Reality: People outside the IT industry believe that anyone can test software and that testing is not a creative process. Testers, however, know very well that this is a myth. A good tester can write faster, fewer, and more effective test cases than a bad tester. During ad hoc testing, a good tester can easily outshine an average tester. Not everyone can be a detective!

Myth 10: A tester's only job is to find bugs

Reality: The job of the QA team is to reduce the overall cost of quality and secure customer satisfaction, which means ensuring high quality in all the applicable software quality dimensions, not just bugs. It starts with examining functional suitability to testing end-user usability. But if a team hires testers only for hunting defects, that is all they will get.