TA Lecture-1

Introduction to Software Testing

Dr. Javed Imran

Assistant Professor, TIET

What is Software Testing?

- Many people understand many definitions of testing :

1. Testing is the process of demonstrating that errors are not present.
2. The purpose of testing is to show that a program performs its intended functions correctly.
3. Testing is the process of establishing confidence that a program does what it is supposed to do.

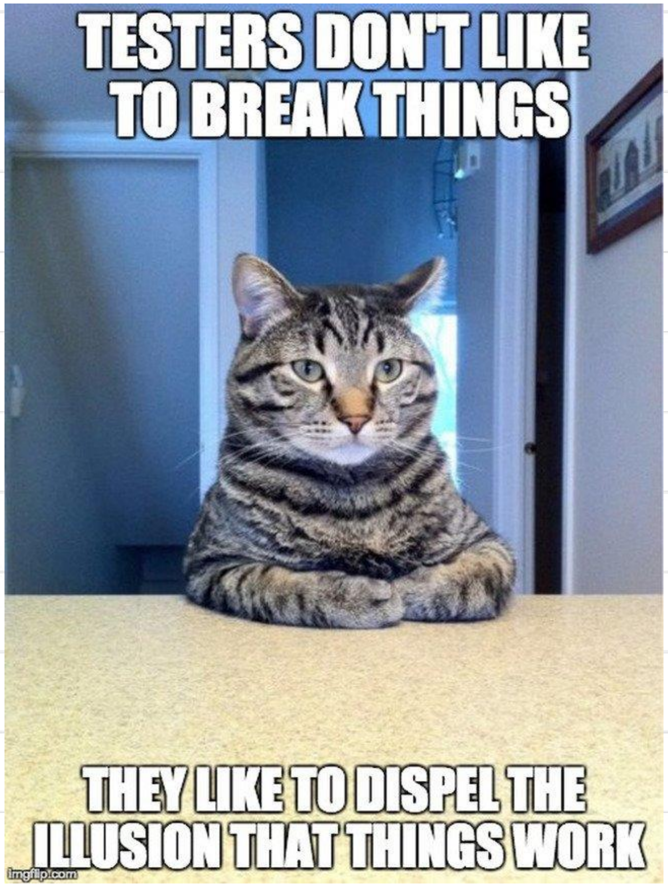
These definitions are incorrect!

Software Testing

- A more appropriate definition is:

  Software Testing is the execution of programs with the intent of finding defects.

- Testing is a the **process of exercising a software component**
  - using **a selected set of test cases**,
  - with the intent **identify defects (bugs)** before the software is delivered to end-users,
  - helps **improve overall quality**, enhance user **satisfaction**, and
  - **reduce the cost** of fixing issues later in the development lifecycle.


TESTERS DON'T LIKE TO BREAK THINGS
THEY LIKE TO DISPEL THE ILLUSION THAT THINGS WORK
imgflip.com

What Should We Test ?

- We should test the program's responses to every possible input.
- It means, we should test for all valid and invalid inputs.
- Suppose a program requires two 8 bit integers as inputs. Total possible combinations are $2^8 x 2^8$.
- If only one second it required to execute one set of inputs, it may take 18 hours to test all combinations.
- Practically, inputs are more than two and size is also more than 8 bits. We have also not considered invalid inputs where so many combinations are possible.
- Hence, complete testing is just not possible, although, we may wish to do so.

Why is Software Testing Important to a Business?

- Loss of money
- Loss of time
- Damage to business reputation
- Injury or death

Deployment without Software Testing

Software Testing Terms

- Error
- Mistake
- Bug
- Fault
- Failure
- Test
- Test Case
- Test Suite
- Verification
- Validation

Error, Mistake, Bug, Fault and Failure

- People make **errors**. A good synonym is **mistake**. This may be a syntax error or misunderstanding of specifications. Sometimes, there are logical errors.
- When developers make mistakes while coding, we call these mistakes "**bugs**".
- A **fault** is the representation of an error, where representation is the mode of expression, such as narrative text, data flow diagrams, ER diagrams, source code etc. Defect is a good synonym for fault.
- A **failure** occurs when a fault executes. A particular fault may cause different failures, depending on how it has been exercised.

Test, Test Case and Test Suite

- **Test** and **Test case** terms are used interchangeably. In practice, both are same and are treated as synonyms. Test case describes an input description and an expected output description.

Test Case ID	
Section-I (Before Execution)	Section-II (After Execution)
Purpose :	Execution History:
Pre condition: (If any)	Result:
Inputs:	If fails, any possible reason (Optional)
Expected Outputs:	Any other observation:
Post conditions:	Any suggestion:
Written by:	Run by:
Date:	Date:

- The set of test cases is called a **test suite**. Hence any combination of test cases may generate a test suite.

Verification and Validation

- **Verification**: The process of proving the programs correctness.
  - Verification answers the question: Am I building the product right?

- **Validation**: The process of evaluating software at the end of the software development to ensure compliance with respect to the customer needs and requirements. The process of finding errors by executing the program in a real environment
  - Validation answers the question: Am I building the right product?

I can Verify this is a house
I can Validate that this house was installed properly

Example

VERIFICATION

- 2 sleeves?
- Is it size L?
- Is it blue?
- Are any buttons missing?

VALIDATION

- Does it fit?
- Is it comfortable to drive in?
- Does the colour match my eyes?
- Can I afford it?
- Is it good quality?
- Will my date like it?

Verification vs. Validation

Verification	Validation
It means "Are we implementing the software right?"	It means "Are we implemented the right software?"
It comes before validation.	It comes after verification.
It is known as Static Testing.	It is known as Dynamic Testing.
It is executed by Quality assurance team or Developers.	It is executed by the Testing team.
It includes checking documents, design, code and program.	It includes testing and validating the actual product.
Whether the software conforms to specification is checked.	It checks whether the software meets the requirements and expectations of a customer.
It involves different methods like inspections, reviews and walkthroughs.	It includes testing like Functional, System, Integration and User Acceptance testing.
It does not involve executing the code.	It always involves executing the code.

The Cost of Software Development Phases

Software Development Phases	Cost Spent
Requirements Analysis	3%
Specification	3%
Design	5%
Coding	7%
Testing	15% (should be > 50%)
Operational and Maintenance	67%

Principles of Software Testing

Seven Principles of Software Testing

Testing shows the presence of defects

Exhaustive testing is impossible

Early testing saves time and money

Defects cluster together

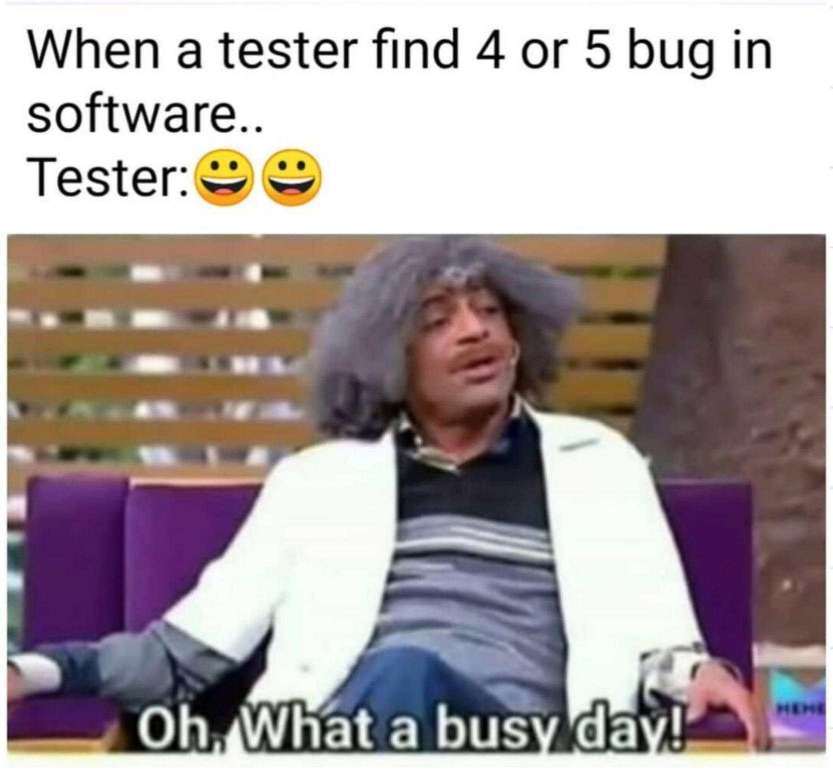
Beware of the pesticide paradox

Testing is context dependent

Absence-of-errors is a fallacy

1. Testing Shows Presence of Defects/Bugs

- Testing can show the defects/bugs are present, but cannot prove that there are no defects.
- After testing the application or product thoroughly we cannot say that the product is 100% defect free.
- Testing always reduces the number of undiscovered defects remaining in the software but even if no defects are found, it is not a proof of correctness.



2. Exhaustive Testing is Impossible

- Testing everything including all combinations of inputs and preconditions is not possible.
- For example: In an application in one screen there are 15 input fields, each having 5 possible values, then to test all the valid combinations you would need 5^{15} = 30,517,578,125 tests.
- This is very unlikely that the project timescales would allow for this number of tests.

3. Early Testing

- To find defects early, testing activities should be started as early as possible in the software development life cycle, and should be focused on defined objectives.

- When defects are found earlier in the lifecycle, they are much **easier** and **cheaper** to fix.

4. Defect Clustering

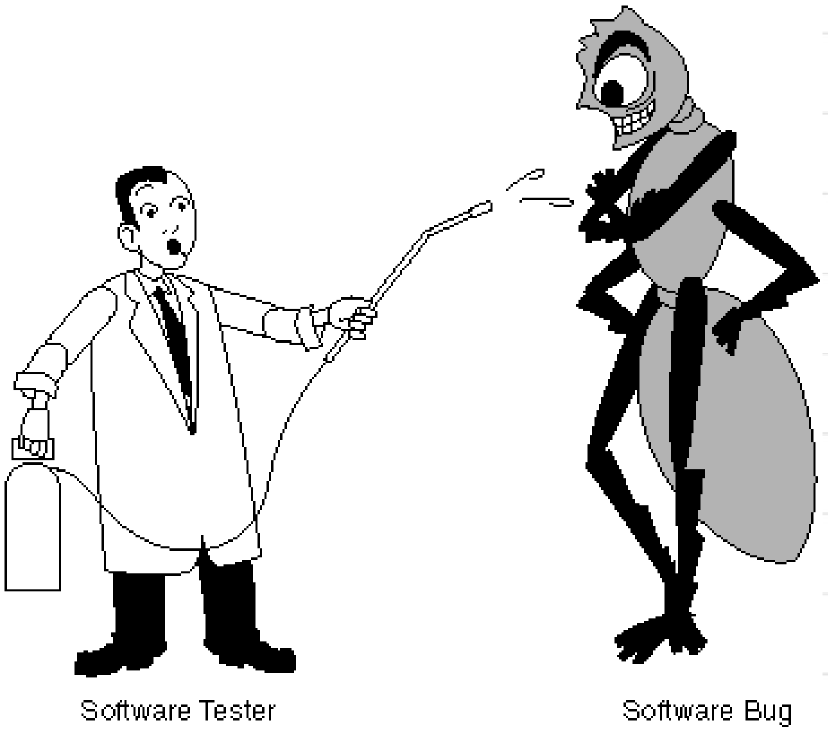
- A small number of modules usually contains most of the defects discovered during pre-release testing, or is responsible for most of the operational failures.

- There is NO equal distribution of defects within one test object. The place where defect occurs, it’s likely to find some more. The testing process must be flexible and respond to this behavior.

5. Pesticide Paradox

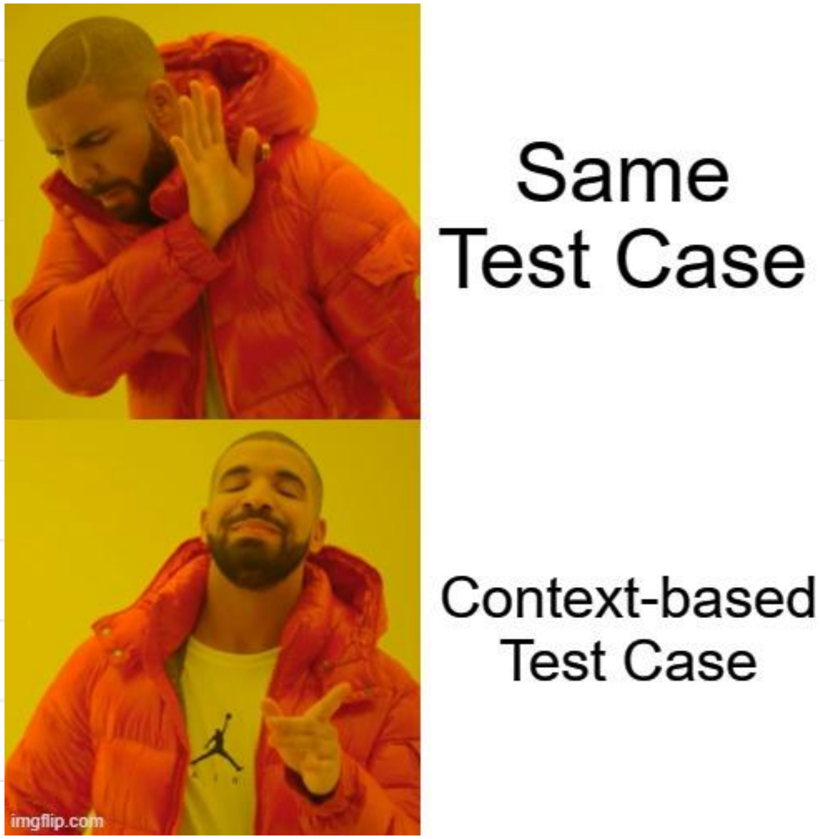
- If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects.

- To overcome this "pesticide paradox", test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to find potentially more defects.



6. Testing is Context Dependent

- Testing is done differently in different contexts.
- For example, an E-commerce application cannot be tested like a Flight Simulation application, or may be a Health Domain application cannot be treated like a Retail Market application.



7. Absence-of-errors Fallacy

- Finding and fixing defects does not help if the system built is unusable and does not fulfill the users' needs and expectations.

- Just because testing didn't find any defects in the software, it does not mean that the software is ready to be shipped.

TESTERS WHEN THEY FIND NO ERRORS
CUSTOMERS WHEN THE ACTUALLY USE THE PRODUCT

SDLC vs. STLC

Software Development Life Cycle (SDLC)

- Software Development Life Cycle is a classification of individual activities executed throughout the software development process.

- The SDLC includes various phases:

  1. Requirement Phase
  2. Analysis Phase
  3. Design Phase
  4. Development Phase
  5. Testing Phase
  6. Deployment & Maintenance Phase

Software Testing Life Cycle (STLC)

- Software Testing Life Cycle is the order of different activities executed throughout the software testing process.

- Testing itself has many phases:
  1. Requirement Analysis
  2. Test Planning
  3. Test Design & Development
  4. Test Environment Setup
  5. Test Execution
  6. Test Closure

SDLC vs. STLC

Criterion	SDLC	STLC
Origin	Development Life Cycle	Testing Life Cycle
Stands for	SDLC stands for Software Development Life Cycle	STLC stands for Software Testing Life Cycle
Focus	On both development and testing process	On only testing process
Relationship	It is taken as the predecessor	It is taken as the successor
Phases	Requirement Gathering, Analysis, Design, Coding, Testing, Deployment & maintenance	Requirement Analysis, Test Planning, Test Design & Development, Environment Setup, Test Execution, Test Closure

SDLC vs. STLC

Criterion	SDLC	STLC
Requirement Gathering Phase	Business analyst gathers the requirements and create Development Plan	QA team analyses requirement documents and create System Test Plan
Design Phase	The development team develops the high and low-level design of the software based on the requirements	Test Architect or a Test Lead usually plan the test strategy
Coding Phase	The actual code is developed as per the designed document	The QA team prepares the test environment
Testing Phase	Actual testing is done in this phase. It includes Unit, Integration, System, Retesting & Regression testing etc., Also the development team involves in fixing the bugs reported	Actual testing is done in this phase. Defect reporting & retesting is done here

SDLC vs. STLC

Criterion	SDLC	STLC
Deployment or Maintenance Phase	The development team involves in support and release updates	The QA team executes regression suites to check maintenance code deployed
When it is performed	The SDLC phases are performed before the STLC phases	The STLC phases are performed after the SDLC phases
Outcome	A good quality software product	A bug free software

Software Testing Life Cycle

1. Requirement Analysis

- Reviewing the software requirements document (SRD) and other related documents.
- Interviewing stakeholders to gather additional information.
- Identifying any ambiguities or inconsistencies in the requirements.
- Identifying any missing or incomplete requirements.
- Identifying any potential risks or issues that may impact the testing process.

2. Test Planning

- Identifying the testing objectives and scope.
- Developing a test strategy: selecting the testing methods and techniques that will be used.
- Identifying the testing environment and resources needed.
- Identifying the test cases that will be executed and the test data that will be used.
- Estimating the time and cost required for testing.
- Identifying the test deliverables and milestones.
- Assigning roles and responsibilities to the testing team.
- Reviewing and approving the test plan.

3. Test Designing & Development

- Identifying the test cases that will be developed.
- Writing test cases that are clear, concise, and easy to understand.
- Creating test data and test scenarios that will be used in the test cases.
- Identifying the expected results for each test case.
- Reviewing and validating the test cases.
- Updating the requirement traceability matrix (RTM) to map requirements to test cases.
- Types:
  1. Specification Based Technique
  2. Structure Based Technique
  3. Experience Based Technique

4. Test Environment Setup

- Test environment setup is a vital part of the STLC.
- Basically, the test environment decides the conditions on which software is tested.
- This is independent activity and can be started along with test case development.
- In this process, the testing team is not involved.
- Either the developer or the customer creates the testing environment.

5. Test Execution

- The test cases and scripts created in the test design stage are run against the software application.
- Any defects or issues that are found during test execution are logged in a defect tracking system.
- The results of the test execution are analyzed to determine the software’s performance and identify any defects or issues.
- Any defects that are identified during test execution are retested to ensure that they have been fixed correctly.
- Test results are documented and reported to the relevant stakeholders.

6. Test Closure

- A report is created that summarizes the overall testing process, including the number of test cases executed, the number of defects found, and the overall pass/fail rate.
- All defects that were identified during testing are tracked and managed until they are resolved.
- The test environment is cleaned up, and all test data and test artifacts are archived.
- A report is created that documents all the testing-related activities that took place, including the testing objectives, scope, schedule, and resources used.
- Feedback from the testing process is collected and used to improve future testing processes.