Software Testing

Introduction to Software testing

- We all know the wonders that software technology has created for us.
- But for software companies creating the software is only half the battle.
- Devising a method to properly test the software is a challenge by itself.
- To create quality software a effective and efficient software testing process is required.
- So what is this software testing process which acts as the king maker in the success story of the software.

Software Testing id defined as follows,

According to **ANSI/IEEE 1059** standard – A process of analyzing a software item to detect the differences between existing and required conditions (i.e., defects) and to evaluate the features of the software item.

Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors.

It involves execution of a software component or system component to evaluate one or more properties of interest.

Software testing can also be stated as the process of validating and verifying that a software program/application/product.

The process consisting of all static and dynamic life cycle activities with respect to planning preparation and evaluation of software products to see if they satisfy the specified requirements and detect defects.

History :

Origins of software testing can be traced back to the fifties when the primary method of testing anything was debugging.

In seventies the approach turned to be one of destruction, meaning the code would be broken by the tester to find the gap in it.

This method was effective till a more robust preventive type of testing methodology came into existence.

In 1979 Glenford J Meyers came up with the theory and suggested that there should be a difference in debugging the process of finding bugs in the software to the process of checking the software in real world scenarios which can be termed testing.

Why testing code is important

Every component of the system must work in harmony.

Software bugs could be expensive or even dangerous & Bugs can potentially cause monetary and human loss.

Without testing any software system would eventually fail leaving the user base unhappy and holding the customer base is the success for any software product in this ever emerging market.

It is very important to ensure the Quality of the product.

Testing avoids delivering sub-standard code & increases code stability.

Testing is really required to point out the defects and errors that were made during the development phases.

Testing is required for an effective performance of software application or product.

Causes of software defects

Humans are prone to do mistakes.

Mistakes leads to defects in code and system specifications.

Defects are tend to be detected only when they become failures.

Failures in real time end up in becoming problems.

Goals of testing

Is to find defects, measure quality , and provide confidence.

The user gets a trustworthy product.

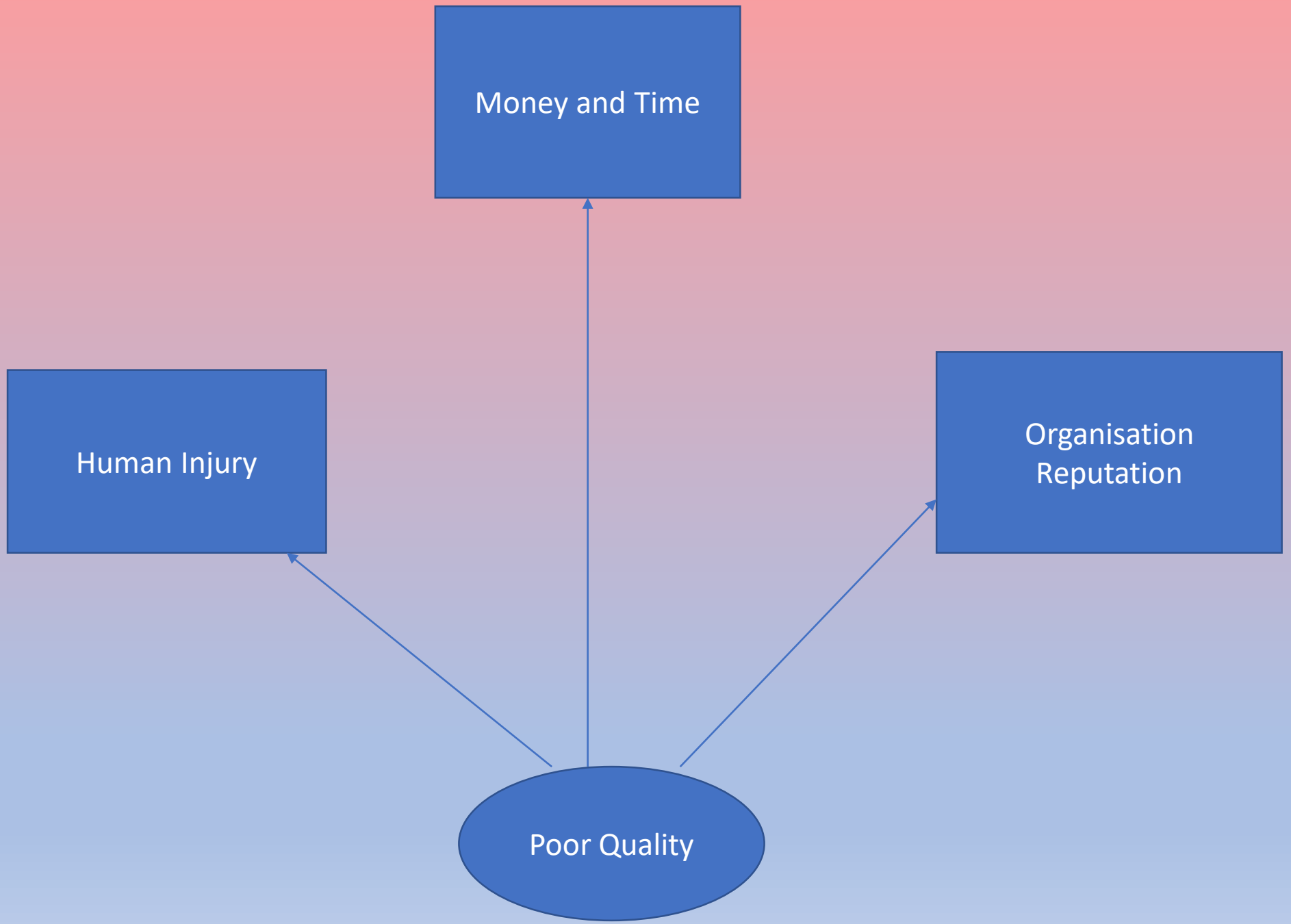
Keeps user’s personal information and data safe.

Vulnerability free products.

Problems and risks are eliminated beforehand.

Saves a lot of troubles later on.

It makes software more reliable and user-friendly to operate.


Money and Time
Human Injury
Organisation Reputation
Poor Quality

When to Start Testing?

An early start to testing reduces the cost and time to rework and produce error-free software.

In Software Development Life Cycle (SDLC), testing can be started from the Requirements Gathering phase and continued till the deployment of the software.

It also depends on the development model that is being used. For example, in the Waterfall model, formal testing is conducted in the testing phase; but in the incremental model, testing is performed at the end of every increment/iteration and the whole application is tested at the end.

Testing is done in different forms at every phase of SDLC −

- During the requirement gathering phase, the analysis and verification of requirements are also considered as testing.
- Reviewing the design in the design phase with the intent to improve the design is also considered as testing.
- Testing performed by a developer on completion of the code is also categorized as testing.

When to Stop Testing?

It is difficult to determine when to stop testing, as testing is a never-ending process and no one can claim that a software is 100% tested.

The following aspects are to be considered for stopping the testing process −

- Testing Deadlines
- Completion of test case execution
- Completion of functional and code coverage to a certain point
- Bug rate falls below a certain level and no high-priority bugs are identified
- Management decision

Myth 1: Testing is Too Expensive

Myth 2: Testing is Time-Consuming

Myth 3: Only Fully Developed Products are Tested

Myth 4: Complete Testing is Possible

Myth 5: A Tested Software is Bug-Free

Myth 6: Missed Defects are due to Testers

Myth 7: Testers are Responsible for Quality of Product

Myth 8: Test Automation should be used wherever possible to Reduce Time

Myth 9: Anyone can Test a Software Application

Myth 10: A Tester's only Task is to Find Bugs

Verification and Validation

Testing = Verification + Validation

Verification :

- Process of evaluating the artefact's of software development in order to ensure that the product being developed will comply to the standards.
- Process oriented approach.
- Static process.
- Are we building the product right?
- Errors found during verification require lesser cost/resources to get fixed as compared to be found during validation phase.

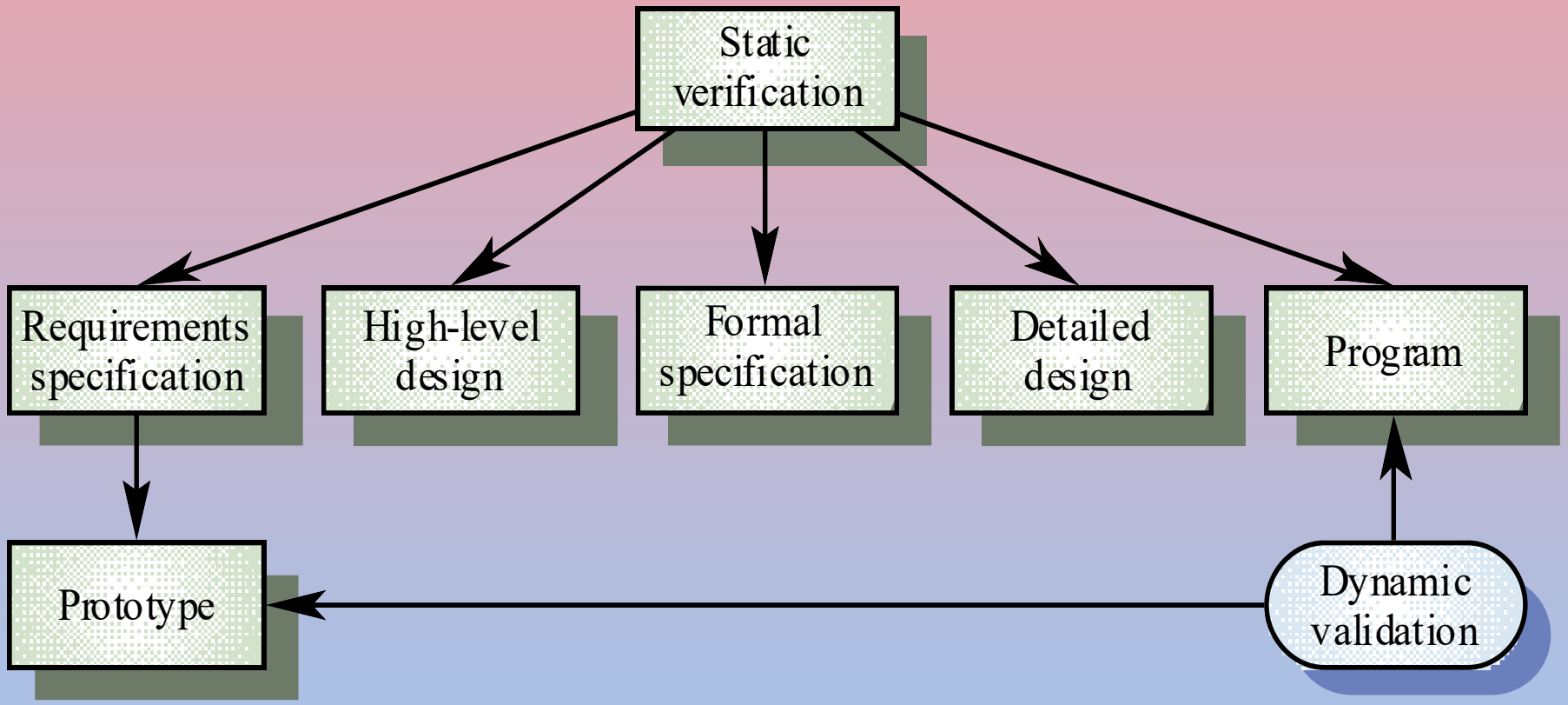
Validation :

- Process of validating that the developed software product conforms to the specified business requirements.
- Dynamic.
- Product oriented approach.
- Are we building the right product?
- Errors found during validation require more cost/resources. Later the error is discovered higher is the cost to fix it.

Two principal objectives :

- Discovery of defects in a system.
- Assessment of whether the system is usable in an operational situation.
- Verification and Validation should establish confidence that the software is fit for purpose.
- Which does not mean completely free of defects,rather, it must be good enough for its intended use.

Static and Dynamic V&V



QA, QC and Testing in software development process

Quality Assurance	Quality Control	Testing
It is a procedure that focuses on providing assurance that quality requested will be achieved	It is a procedure that focuses on fulfilling the quality requested.	Process of detecting and solving software errors and flaws
aims to prevent the defect	aims to identify and fix defects	aims to identify and fix defects
Preventive technique	Corrective technique	Detection technique
Proactive measure	Reactive measure	Proactive measure
full software development life cycle	full software testing life cycle	full software testing life cycle
In order to meet the customer requirements, QA defines standards and methodologies	QC confirms that the standards are followed while working on the product	It ensures the identification and detection of the bugs/ defects(if any) in a software product.
procedure to create the deliverables	procedure to verify that deliverables	It concerns with the improvement of the quality.

Principles of Software Testing

Formulated over the past 40 years, the seven principles of software testing represent the ground rules for the process. These are:

- Testing shows presence of mistakes.
- Exhaustive testing is impossible.
- Early testing.
- Defect clustering.
- Pesticide paradox.
- Testing is context dependent.
- Absence-of-errors fallacy.

Testing shows presence of mistakes. Testing is aimed at detecting the defects within a piece of software. But no matter how thoroughly the product is tested, we can never be 100 percent sure that there are no defects. We can only use testing to reduce the number of unfound issues.

Exhaustive testing is impossible. There is no way to test all combinations of data inputs, scenarios, and preconditions within an application. For example, if a single app screen contains 10 input fields with 3 possible value options each, this means to cover all possible combinations, test engineers would need to create 59,049 (310) test scenarios. And what if the app contains 50+ of such screens? In order not to spend weeks creating millions of such less possible scenarios, it is better to focus on potentially more significant ones.

Early testing. As mentioned above, the cost of an error grows exponentially throughout the stages of the SDLC. Therefore it is important to start testing the software as soon as possible so that the detected issues are resolved and do not snowball.

Defect clustering. This principle is often referred to as an application of the Pareto principle to software testing. This means that approximately 80 percent of all errors are usually found in only 20 percent of the system modules. Therefore, if a defect is found in a particular module of a software program, the chances are there might be other defects. That is why it makes sense to test that area of the product thoroughly.

Pesticide paradox. Running the same set of tests again and again won't help you find more issues. As soon as the detected errors are fixed, these test scenarios become useless. Therefore, it is important to review and update the tests regularly in order to adapt and potentially find more errors.

Testing is context dependent. Depending on their purpose or industry, different applications should be tested differently. While safety could be of primary importance for a fintech product, it is less important for a corporate website. The latter, in its turn, puts an emphasis on usability and speed.

Absence-of-errors fallacy. The complete absence of errors in the product does not necessarily mean its success. No matter how much time is spent in testing the functionality if the product is not useful or does not meet the user expectations it won't be adopted by the target audience.

Apart from the above basic principles,

• Testing must be an independent process handled by unbiased professionals.

• Test for invalid and unexpected input values as well as valid and expected ones.

• Testing should be performed only on a static piece of software (no changes should be made in the process of testing).

• Use exhaustive and comprehensive documentation to define the expected test results.