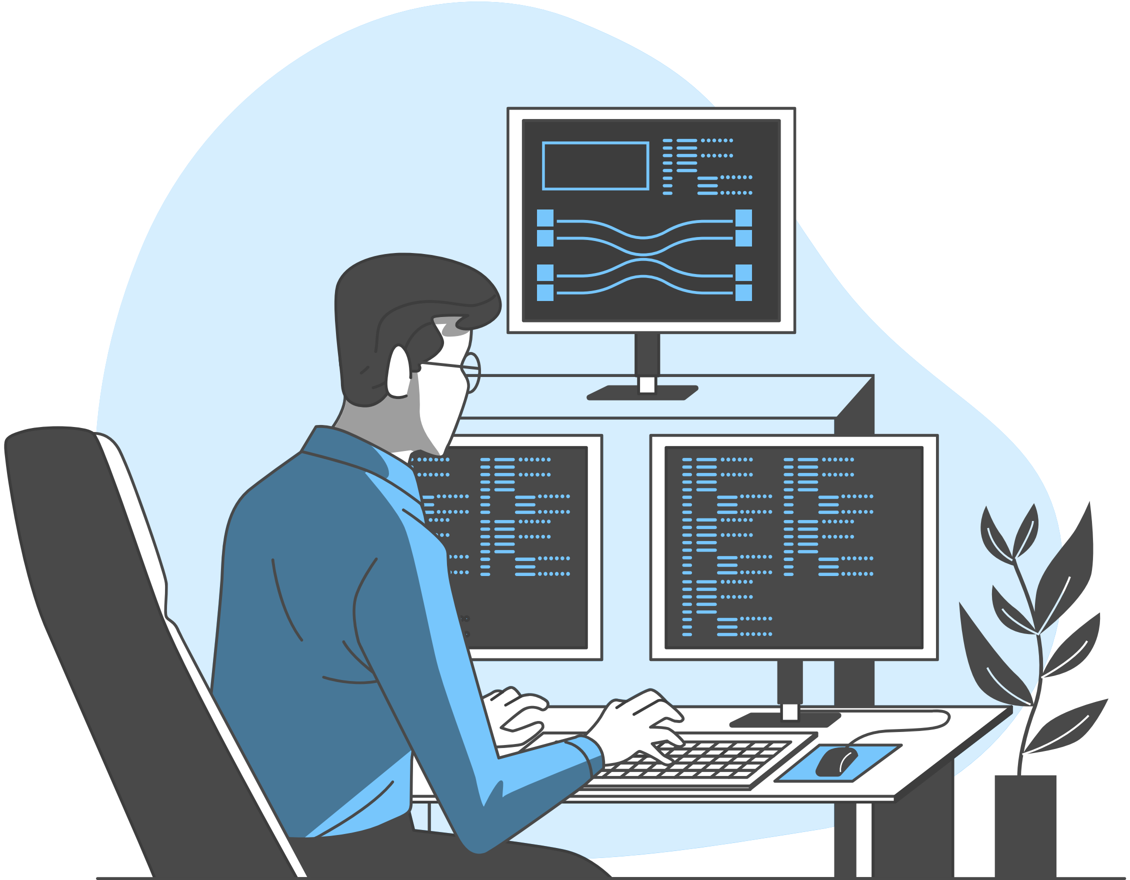
CDIP
Center for Development of IT Professionals

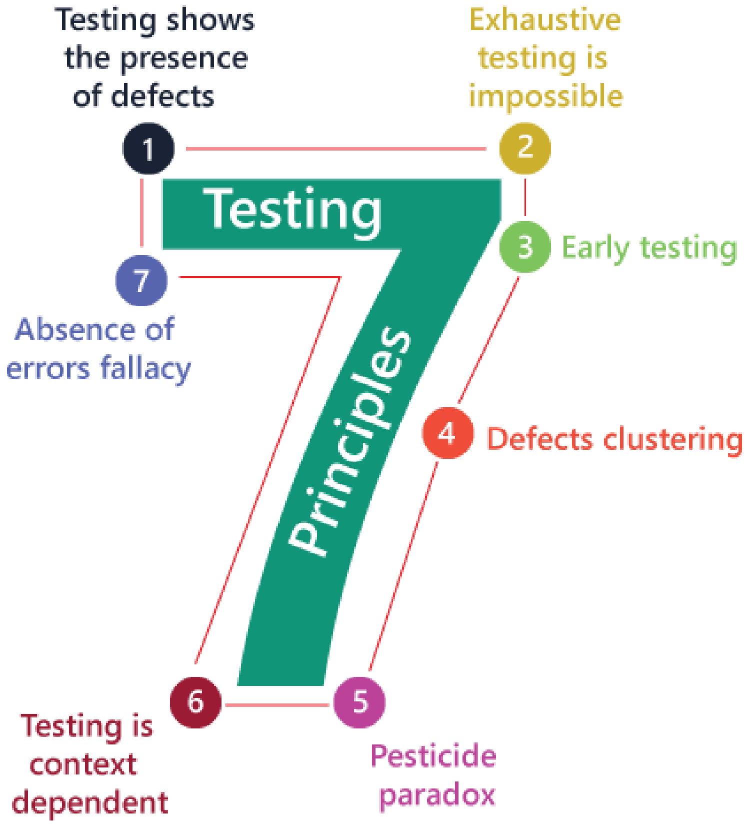


Professional Software Testing & Quality Assurance

Instructor

Parvez Hossain

Software Testing Principles


Testing shows the presence of defects
Exhaustive testing is impossible
1
2
Testing
3
Early testing
7
Absence of errors fallacy
Principles
4
Defects clustering
6
5
Testing is context dependent
Pesticide paradox

Testing Shows Presence of Defects

In software testing industry no one will say that there is no defect in the software, which is quite true as **testing can't prove that the software is error-free.**

You will never heard from any testing team that they have tested the software fully and there is no defect in the software. Instead of that, **every testing team confirms that the software meets all business requirements and it is functioning as per the needs of the end user.**

However, **the objective of testing is to find more and more hidden defects using different techniques and methods.** Testing can reveal undiscovered defects and if no defects are not found then it doesn't mean that the software is defect free.

Exhaustive Testing is Not Possible

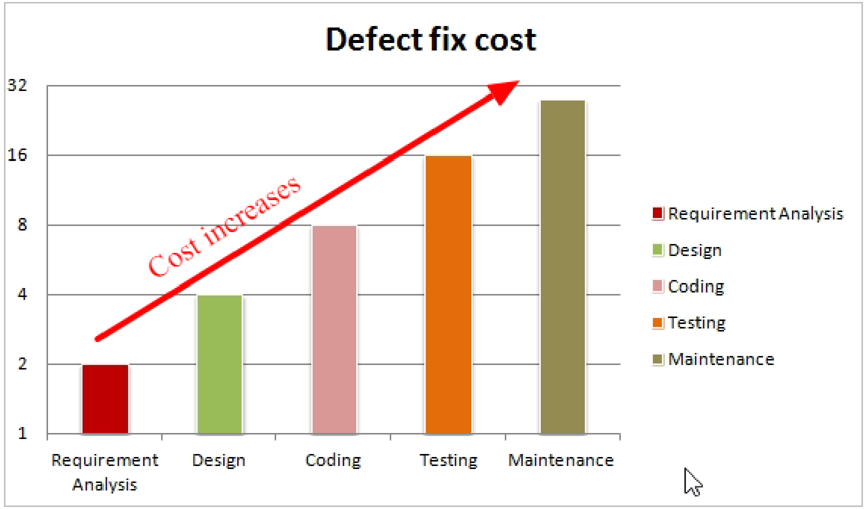
It is not possible to test all the functionalities with all valid and invalid combinations of input data during actual testing. Instead of this approach, testing of a few combinations is considered based on priority using different technique.

Exhaustive testing will take unlimited effort and most of those efforts are ineffective. Also the project timeline will not allow testing of so many number of combinations. **Hence it is recommended to test input data using different methods like Equivalence Partitioning and Boundary Value Analysis.**

Early Testing

Center for Development of IT Professionals

Testers need to get involved at an early stage of software development life cycle (SDLC). Thus the defects during the requirement analysis phase or any documentation defects can be identified. The cost involved in fixing such defects is very less when compared to those that are found during the later stage of testing.



Cost required for fixing a defect found during the requirement analysis phase is less and it goes on increasing as we move towards Testing and Maintenance.

Defect Clustering

Defect clustering means a small number of modules containing most of the defects. Basically the defects are not distributed uniformly across the entire application.

During testing it may happen that most of the defects found are related to a small number of modules. There might be multiple reasons for this, like the modules may be complex, coding related to such modules may be complicated etc.

Defect clustering is based on 'Pareto Principle' which is known as 80-20 rule. It means that 80% of the defects found are due to 20% of the modules in the application.

For an effective testing strategy, it is necessary to thoroughly examine these areas of the software. The defect clustering method relies on the teams' knowledge and experience to identify which modules to test. You can identify such risky modules from your experience. Therefore, the team only has to focus on those "sensitive" areas, saving both time and effort.

Pesticide Paradox

Pesticide paradox principle says that if the same set of test cases are executed again and again over the period of time then these set of test cases are not capable enough to identify new defects in the system.

In order to overcome this ‘Pesticide Paradox’ these set of test cases needs to be regularly reviewed and revised. If required a new set of test cases can be added and the existing test cases can be deleted if they are not able to find any more defects from system.

Testing is Context-Dependent

There are several domains available in the market like Banking, Insurance, Medical, Travel, Advertisement etc and each domain has a number of applications. Also for each domain, their applications have different requirements, functions, different testing purpose, risk, techniques etc.

Different domains are tested differently, thus testing is purely based on the context of the domain or application.

For Example, testing a banking application is different than testing any e-commerce or advertising application. The risk associated with each type of application is different, thus it is not effective to use the same method, technique, and testing type to test all types of application.

Absence of Error

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 only finding defects, but also to check that software addresses the business needs.

For Example, suppose the application is related to an e-commerce site and the requirements against "Shopping Cart or Shopping Basket" functionality which is wrongly interpreted and tested. Here, even finding more defects does not help to move the application into the next phase or in the production environment.

Thanks!

Do you have any questions?

Email: parvez9605@gmail.com
WhatsApp: +88 01772880239