Seven Principles of Software Testing | Software Testing Material

Last Updated on April 11, 2017 by Rajkumar May 7, 2016

There are seven principles of Software Testing.

1. Testing shows presence of defects
2. Exhaustive testing is impossible
3. Early testing
4. Defect clustering
5. Pesticide paradox
6. Testing is context dependent
7. Absence of error – fallacy

Here is a video tutorial to learn "**Principles of Software Testing**":

https://youtu.be/NC1aqG4tWl4

Please be patient. The video will load in some time.

If you liked this video, then please subscribe to our YouTube Channel for more video tutorials.
Let's see principles of Software Testing in detail.

Seven Principles of Software Testing:

1. Testing Shows Presence of Defects:

Testing shows the presence of defects in the software. The goal of testing is to make the software fail. Sufficient testing reduces the presence of defects. In case testers are unable to find defects after repeated regression testing doesn't mean that the software is bug-free.

Testing talks about the presence of defects and don't talk about the absence of defects.

2. Exhaustive Testing is Impossible:

What is Exhaustive Testing?

Testing all the functionalities using all valid and invalid inputs and preconditions is known as Exhaustive testing.

Why it's impossible to achieve Exhaustive Testing?

Assume we have to test an input field which accepts age between 18 to 20 so we do test the field using 18,19,20. In case the same input field accepts the range between 18 to 100 then we have to test using inputs such as 18, 19, 20, 21, …., 99, 100. It's a basic example, you may think that you could achieve it using automation tool. Imagine the same field accepts some billion values. It's impossible to test all possible values due to release time constraints.

If we keep on testing all possible test conditions then the software execution time and costs will rise. So instead of doing exhaustive testing, risks and priorities will be taken into consideration whilst doing testing and estimating testing efforts.

3. Early Testing:

Defects detected in early phases of SDLC are less expensive to fix. So conducting early testing reduces the cost of fixing defects.

Assume two scenarios, first one is you have identified an incorrect requirement in the requirement gathering phase and the second one is you have identified a bug in the fully developed functionality. It is cheaper to change the incorrect requirement compared to fixing the fully developed functionality which is not working as intended.

4. Defect Clustering:

Defect Clustering in software testing means that a small module or functionality contains most of the bugs or it has the most operational failures.

As per the Pareto Principle (80-20 Rule), 80% of issues comes from 20% of modules and remaining 20% of issues from remaining 80% of modules. So we do emphasize testing on the 20% of modules where we face 80% of bugs.

5. Pesticide Paradox:

Pesticide Paradox in software testing is the process of repeating the same test cases again and again, eventually, the same test cases will no longer find new bugs. So to overcome this Pesticide Paradox, it is necessary to review the test cases regularly and add or update them to find more defects.

6. Testing is Context Dependent:

Testing approach depends on the context of the software we develop. We do test the software differently in different contexts. For example, online banking application requires a different approach of testing compared to an e-commerce site.

7. Absence of Error – Fallacy:

99% of bug-free software may still be unusable, if wrong requirements were incorporated into the software and the software is not addressing the business needs.

The software which we built not only be a 99% bug-free software but also it must fulfill the business needs otherwise it will become an unusable software.

These are the seven principles of Software Testing every professional tester should know.