7 Principles of software testing

Prepared by
Subash B

1. Exhaustive testing is not possible
2. Defect Clustering
3. Pesticide Paradox
4. Testing shows presence of defects
5. Absence of Error
6. Early Testing
7. Testing is context dependent

Exhaustive testing is not possible

- Testing everything including all combinations of inputs and preconditions is not possible.
- So, instead of doing the exhaustive testing we can use risk analysis and priorities to focus testing efforts
- **Example:** To Enter the Age in Age field. It should accept 18 to 85 in-between it should not accept 60 to 65
- Due to timelines we wont give all combinations of inputs
- We follow Equivalence Partitioning or boundary value analysis

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.

Pesticide Paradox

- Good testing requires continual development of new testing ideas.
- You need to updates your tests after repeating over and over again, because eventually the same set of test cases will no longer find any new defects.

Testing shows presence of defects

- Testing talks about the presence of defects and don’t talk about the absence of defects.
- 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.
- At max we can give 99% bug-free.

Absence of Error

- Even the software is 99% Bug free; If the system built is unusable and does not fulfil the user’s needs and expectations then finding and fixing defects does not help.
- Software testing is not mere finding defects, but also to check that software addresses the business needs.
- To solve this problem , the next principle of testing states that 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 early stages of testing

Testing is context dependent

- Testing is basically context dependent.
- Different kinds of sites are tested differently.
- For example, a software application in a medical device needs more testing than a games software.
- By the same token, a very popular website (IRCTC), needs to go through rigorous performance testing as well as functionality testing then other site (Smartprix) to make sure the performance is not affected by the load on the servers.