Software Testing Mentor

www.softwaretestingmentor.com

ISTQB Foundation Level and Software Testing Training

Module 1
Fundamentals of Software Testing

Session 3 – Seven Principles of Testing

Seven Testing Principles

There are seven testing principles which offer general guidelines common for all testing

- Testing shows the presence of defects
- Exhaustive testing is impossible
- Early testing
- Defect clustering
- Pesticide paradox
- Testing is context dependent
- Absence-of-error fallacy

Principle 1: Testing shows the presence of defects

Testing can show that defects are present, but cannot prove that there are no defects

Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not proof of correctness

For example: If you only see white swans, you can say that “All swans are white”

but as soon as you see one black swan you cannot say “All swans are white”

Similarly for software

If you are executing test cases and you did not find any defect in many test cases that you executed, you still cannot say that there are no bugs in software.

As soon as you find one bug you can say that “Software is not defect free”

Principle 2: Exhaustive testing is impossible

Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases.

Instead of exhaustive testing, we use risks and priorities to focus testing efforts.

Example 1: How many tests would you need to do to completely test a one-digit numeric field?

- There are 10 possible valid numeric values
- Invalid scenario: There are 26 uppercase alpha characters, 26 lower case,
- At least some special and punctuation characters as well as a blank value. So there would be approx. 68-70 tests for this example of a one-digit field.

Example 2: If we take an example where one screen has 15 input fields

- Each having 5 possible values, then to test all of the valid input value combinations you would need 30 517 578 125 (5^15) tests!
- It is very unlikely that the project timescales would allow for this number of tests.

Principle 3: Early testing

Testing activities should start as early as possible in the software development life cycle and should be focused on defined objectives.

Early testing - such as early test design and review activities because its cheap to find and fix defects in initial phases of SDLC.

After development objective should be to find as many integration, system defects as you can

UAT objective should be to find out if the software meets end user requirements

Maintenance testing objective should be to ensure that there are no new defects introduced in system by changing the software or fixing defects. This is known as regression testing

If we continue testing production use, main objective should be to asses system availability and reliability.

Principle 4: Defect clustering

A small number of modules contain most of the defects discovered during pre-release testing or show the most operational failures.

- Most of the testers have noticed that defects tend to cluster
- Defect clustering can happen because an area of the code is complex and tricky
- Changing software and other products tends to cause knock-on defects.
- Testers should use this information when making their risk assessment for planning the tests, and focus on known 'hot spots'.

Principle 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 bugs.

To overcome this 'pesticide paradox', the 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 potentially find more defects.

- As the 'hot spots' for bugs get cleaned up we need to move our focus elsewhere because same set of tests would not find any more defects
- Over time, focus should be changed from finding coding bugs, to looking at the requirements and design defects

Principle 6: Testing is context dependent

Testing is done differently in different contexts

For example, safety critical software is tested differently from an e-commerce site

Not all software systems carry the same level of risk. So, different approach is taken to test software's in different contexts

Few examples:
1. Family website – Adhoc testing is enough, no tool used, just manual testing is enough
2. eCommerce application – More testing done for websites functional, non-functional attributes, different tools are used and general company procedures and guidelines are followed.
3. Traffic control system – Testing is done by different set of tools and technologies, also much more strict testing guidelines need to be followed. Strict entry and exit criteria for testing.

Principle 7: Absence-of-error fallacy

Finding and fixing defects does not help if the system built is unusable and DOES NOT fulfill the users needs and expectations

- The customer who buys software do not bother about number of defects until software directly affects them and is unstable to use
- They are more interested in software fulfilling their requirements and does what they want it to do.

Conclusion

To conclude, In this session we learned 7 principles of testing

- Testing shows the presence of defects
- Exhaustive testing is impossible
- Early testing
- Defect clustering
- Pesticide paradox
- Testing is context dependent
- Absence-of-error fallacy

Thank You!!!