UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

Experiment: 01 **Date: 13th Feb 2023**

Aim: 7 testing principles through illustrations with respect to different SDLC Models

The seven testing principles are:

1. Testing shows the presence of defects, not their absence.
2. Exhaustive testing is impossible.
3. Early testing saves time and money.
4. Defects found earlier are cheaper to fix.
5. Testing should be risk-based.
6. The Pareto principle applies to software testing.
7. Testing is context-dependent.

Here are the illustrations for these principles in four different SDLC models:

Waterfall Model:

The Waterfall model is a linear and sequential approach to software development that consists of five phases: Requirements gathering, Design, Implementation, Testing, and Maintenance.

Principle 1: Testing shows the presence of defects, not their absence.

In the Waterfall model, testing is performed after the implementation phase. If a defect is found during testing, it is usually costly and time-consuming to fix.

Principle 2: Exhaustive testing is impossible.

In the Waterfall model, testing is performed at the end of the development cycle, making it difficult to perform exhaustive testing due to time and budget constraints.

Principle 3: Early testing saves time and money.

In the Waterfall model, testing should be performed as early as possible to save time and money. For example, testing can be performed during the requirements gathering and design phases.

Principle 4: Defects found earlier are cheaper to fix.

In the Waterfall model, defects found during the requirements gathering and design phases are cheaper to fix than defects found during the testing phase.

Principle 5: Testing should be risk-based.

In the Waterfall model, testing should be focused on the high-risk areas of the software to ensure that they are working as intended.

Principle 6: The Pareto principle applies to software testing.

In the Waterfall model, 80% of defects are usually found in 20% of the code. Testing efforts should be focused on the critical areas of the software.

Principle 7: Testing is context-dependent.

In the Waterfall model, testing should be tailored to the specific context of the project, such as the type of software being developed and the requirements of the end-users.

Agile Model:

The Agile model is an iterative and incremental approach to software development that involves continuous testing and feedback.

Principle 1: Testing shows the presence of defects, not their absence.

In the Agile model, testing is a continuous process that helps to identify defects early in the development cycle.

Principle 2: Exhaustive testing is impossible.

In the Agile model, testing is performed continuously throughout the development cycle, making it easier to perform more comprehensive testing.

Principle 3: Early testing saves time and money.

In the Agile model, testing is performed as early as possible to provide quick feedback to the development team and reduce the overall cost of development.

Principle 4: Defects found earlier are cheaper to fix.

In the Agile model, defects found during the iterative cycles are easier and cheaper to fix than those found later in the development cycle.

Principle 5: Testing should be risk-based.

In the Agile model, testing should be focused on the highest risk areas of the software, such as new features or critical functionality.

Principle 6: The Pareto principle applies to software testing.

In the Agile model, testing efforts should be focused on the most critical areas of the software to ensure that the highest value is delivered to the end-users.

Principle 7: Testing is context-dependent.

In the Agile model, testing should be tailored to the specific context of the project, such as the requirements of the end-users, the complexity of the software, and the time and budget constraints.

Rapid Application Development:

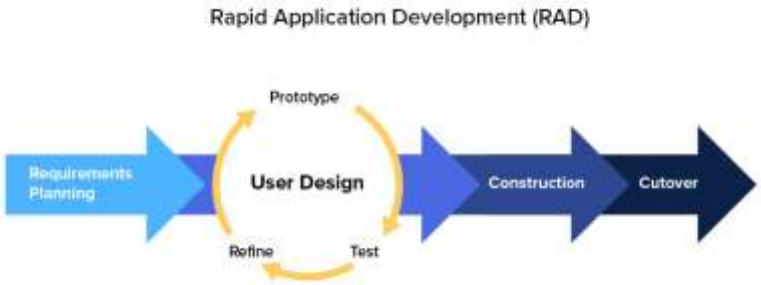
Testing shows presence of defects: This principle suggests that testing should be done with the objective of finding defects in the software. In RAD, testing is performed in short, iterative cycles, which makes it easier to identify and address defects quickly.

Exhaustive testing is impossible: It is impossible to test every possible scenario in software development. Therefore, in RAD, testing focuses on the most critical features of the software, which are prioritized based on user needs and feedback.

Early testing: This principle emphasizes the importance of testing early in the software development life cycle. In RAD, testing is integrated into the development process from the beginning, which allows for defects to be identified and resolved early on.

Defect clustering: This principle suggests that defects tend to cluster in specific areas or modules of the software. In RAD, testing is focused on these areas to ensure that defects are identified and resolved before they cause significant issues.

Pesticide paradox: This principle states that running the same tests repeatedly may not find new defects. In RAD, testing is done in short, iterative cycles with a focus on finding new defects in each cycle.



Testing is context-dependent: The effectiveness of testing depends on the specific context of the software being developed. In RAD, testing is tailored to the specific needs and requirements of the software, which ensures that the software meets the needs of its users.

Absence-of-errors fallacy: This principle suggests that the absence of errors in software does not necessarily mean that the software is of high quality. In RAD, testing is done with the objective of ensuring that the software meets the needs of its users and performs its intended functions, rather than just ensuring the absence of errors.