# Module -4

- Testing automation
- Defect life cycle
- Regression testing
- Testing non-functional requirements.

## Testing Automation

- **Automation Testing or Test Automation** is a software testing technique that performs using special automated testing software tools to execute a test case suite.
- On the contrary, **Manual Testing** is performed by a human sitting in front of a computer carefully executing the test steps.
- Test automation is the process of performing software testing activities with little or no human interaction, in order to achieve greater speed and efficiency.

# Types of Automation Tests

- **Unit Testing**
- **Functional Testing**
- **Integration Testing**
- **Regression Testing**

Already covered with module -1 portions. Refer those slides.

- **Smoke Testing -** Smoke testing is performed to examine whether the deployed build is stable or not.

**Test Automation** is the best way to increase the effectiveness, test coverage, and execution speed in software testing.

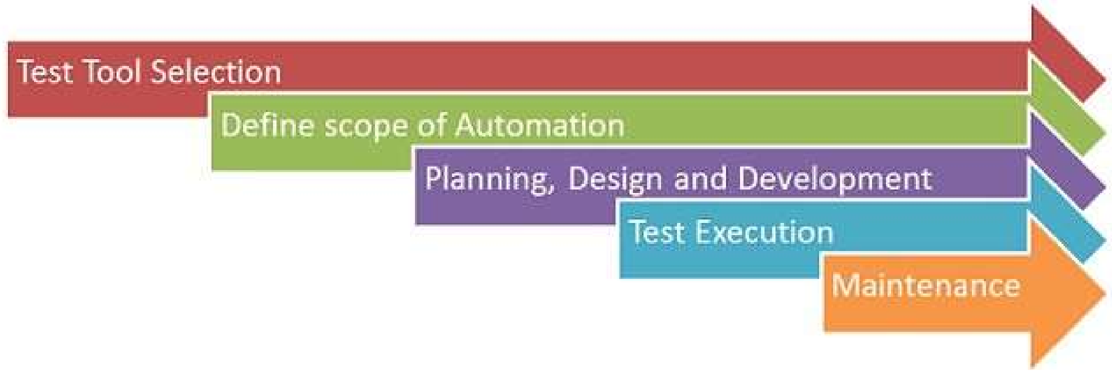
- Automated software testing is important due to the following reasons:
  1. Manual Testing of all workflows, all fields, all negative scenarios is time and money consuming
  2. It is difficult to test for multilingual sites manually
  3. Test Automation does not require Human intervention. You can run automated test unattended (overnight)
  4. Test Automation increases the speed of test execution
  5. Automation helps increase Test Coverage
  6. Manual Testing can become boring and hence error-prone.

## Which Test Cases to Automate?

- Test cases to be automated can be selected using the following criterion
  1. High Risk - Business Critical test cases
  2. Test cases that are repeatedly executed
  3. Test Cases that are very tedious or difficult to perform manually
  4. Test Cases which are time-consuming
- The following category of test cases are not suitable for automation:
  1. Test Cases that are newly designed and not executed manually at least once
  2. Test Cases for which the requirements are frequently changing
  3. Test cases which are executed on an ad-hoc basis.

## Automated Testing Process

- Following steps are followed in an Automation Process
- **Step 1)** Test Tool Selection
- **Step 2)** Define scope of Automation
- **Step 3)** Planning, Design and Development
- **Step 4)** Test Execution
- **Step 5)** Maintenance



## Benefits of Automation Testing

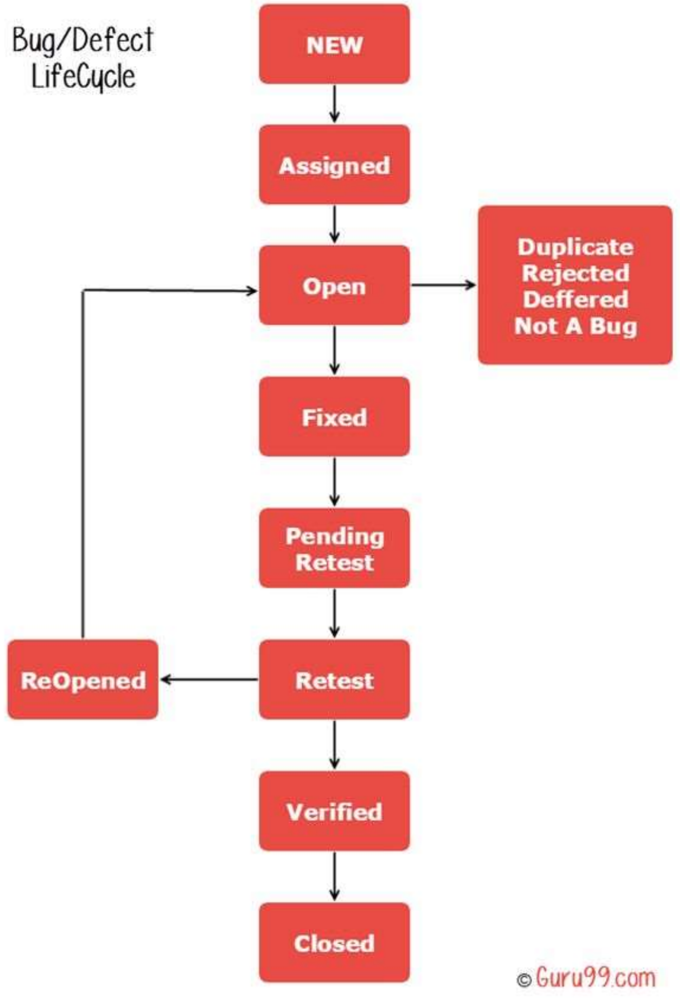
1. 70% faster than the manual testing
2. Wider test coverage of application features
3. Reliable in results
4. Ensure Consistency
5. Saves Time and Cost
6. Improves accuracy
7. Human Intervention is not required while execution
8. Increases Efficiency
9. Better speed in executing tests
10. Re-usable test scripts
11. Test Frequently and thoroughly
12. More cycle of execution can be achieved through automation
13. Early time to market

## Defect life cycle

- **Defect life cycle (**Bug **life cycle)** is a cycle which a defect goes through different stages during its entire lifetime. (during testing process)
- It starts when defect is found (when reported by the tester ) and ends when a tester ensures that a defect is closed, after ensuring that the issue is fixed and won't occur again.

- **Defect Status** (**Bug Status**) in defect life cycle is the present state from which the defect or a bug is currently undergoing.
- The goal of defect status is to precisely convey the current state or progress of a defect or bug in order to better track and understand the actual progress of the defect life cycle.

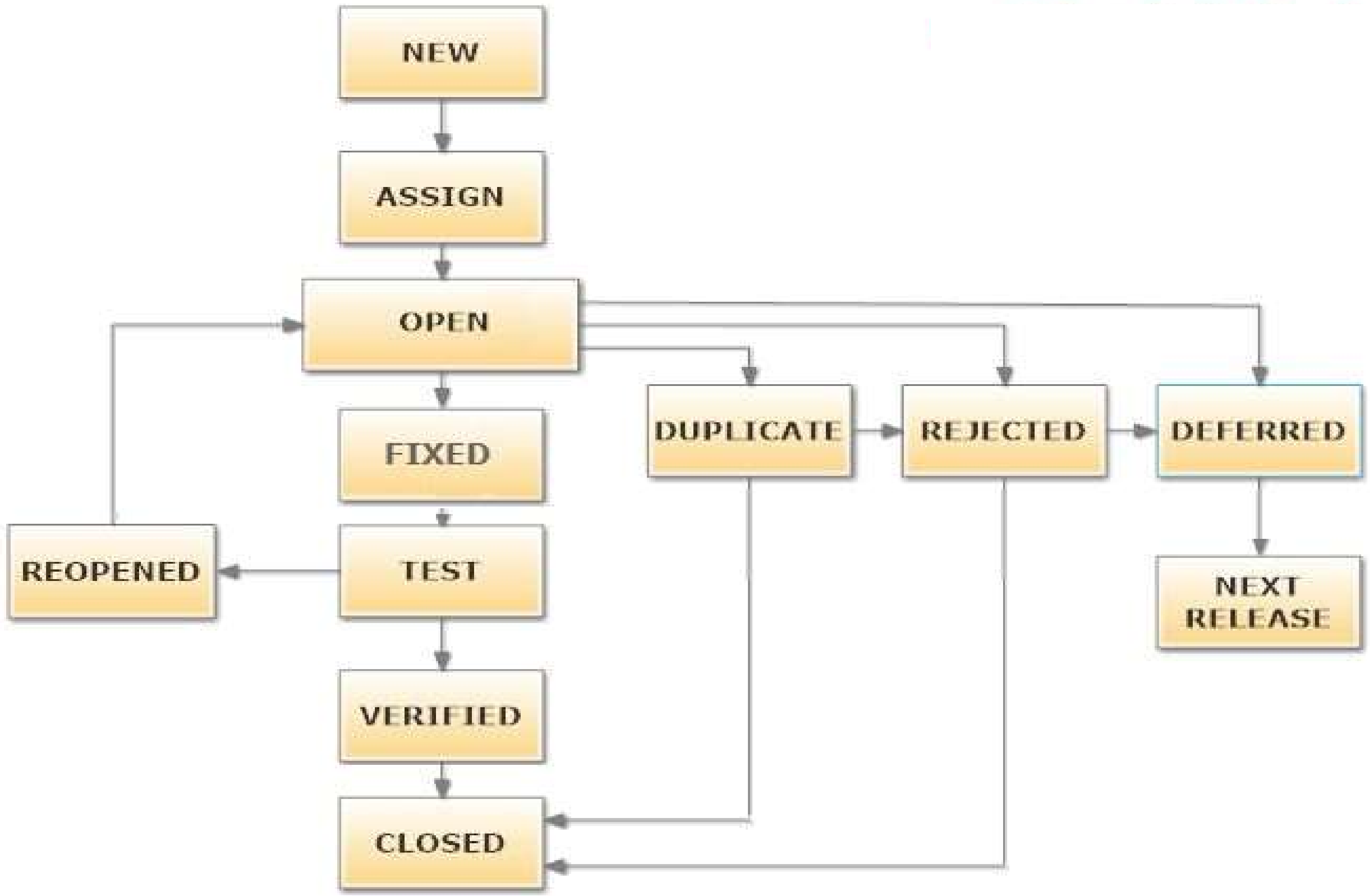
- The  number of states that a defect goes through varies from project to project.
- Lifecycle diagram given here covers all possible states.
- **New:** When a new defect is logged and posted for the first time. It is assigned a status as NEW.
- **Assigned:** Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team
- **Open**: The developer starts analyzing and works on the defect fix



- **Fixed**: When a developer makes a necessary code change and verifies the change, he or she can make bug status as "Fixed."
- **Pending retest**: Once the defect is fixed the developer gives a particular code for retesting the code to the tester. Since the software testing remains pending from the testers end, the status assigned is "pending retest."
- **Retest**: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to "Re-test.“

- **Reopen**: If the bug persists even after the developer has fixed the bug, the tester changes the status to "reopened". Once again the bug goes through the life cycle.
- **Closed**: If the bug is no longer exists then tester assigns the status "Closed."
- **Duplicate**: If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to "duplicate.“
- **Verified**: The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is "verified."

- **Rejected**: If the developer feels the defect is not a genuine defect then it changes the defect to "rejected."
- **Deferred**: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status "Deferred" is assigned to such bugs
- **Not a bug**: If it does not affect the functionality of the application then the status assigned to a bug is "Not a bug".

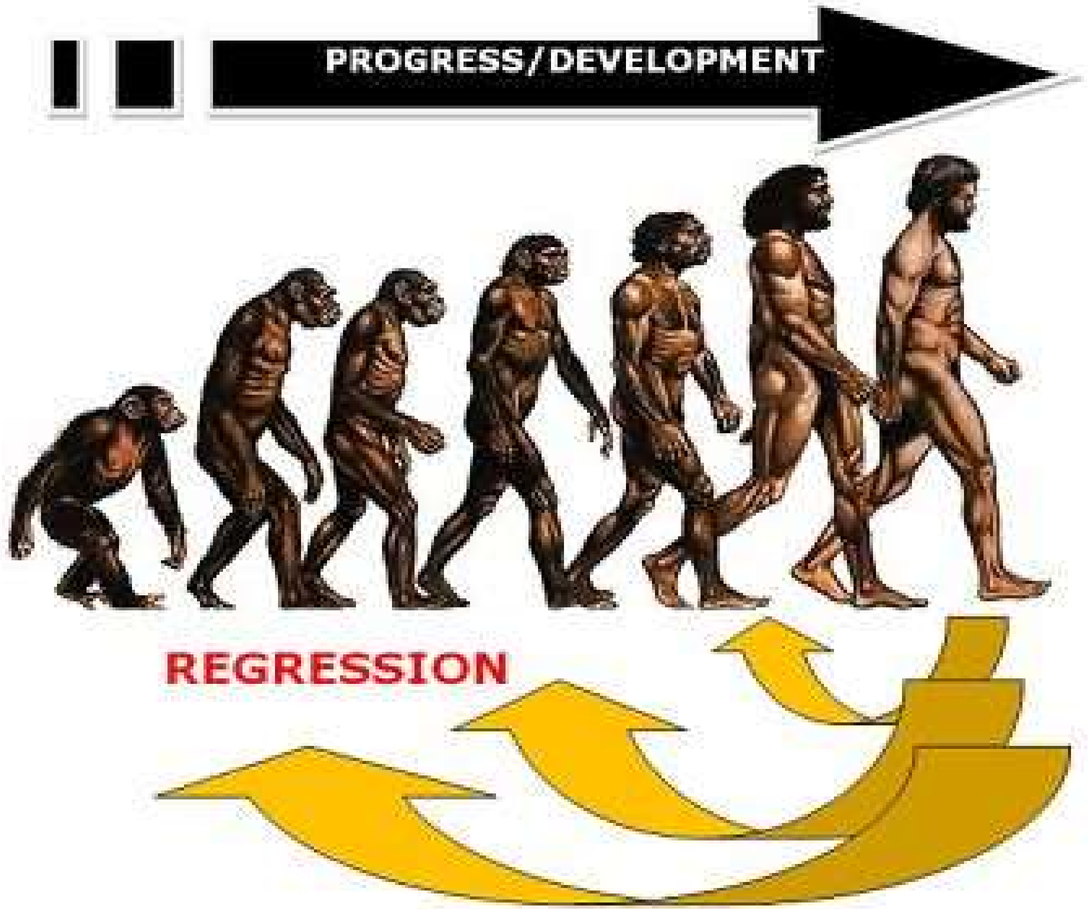

NEW
ASSIGN
OPEN
FIXED
TEST
REOPENED
VERIFIED
CLOSED
DUPLICATE
REJECTED
DEFERRED
NEXT RELEASE

- In **Agile development**, testing needs to happen early and often.
- So, instead of waiting for development to be finished before testing begins, testing happens continuously as features are added.
- Tests are prioritized just like user stories.
- Testers aim to get through as many tests as they can in an iteration.

## Regression Testing

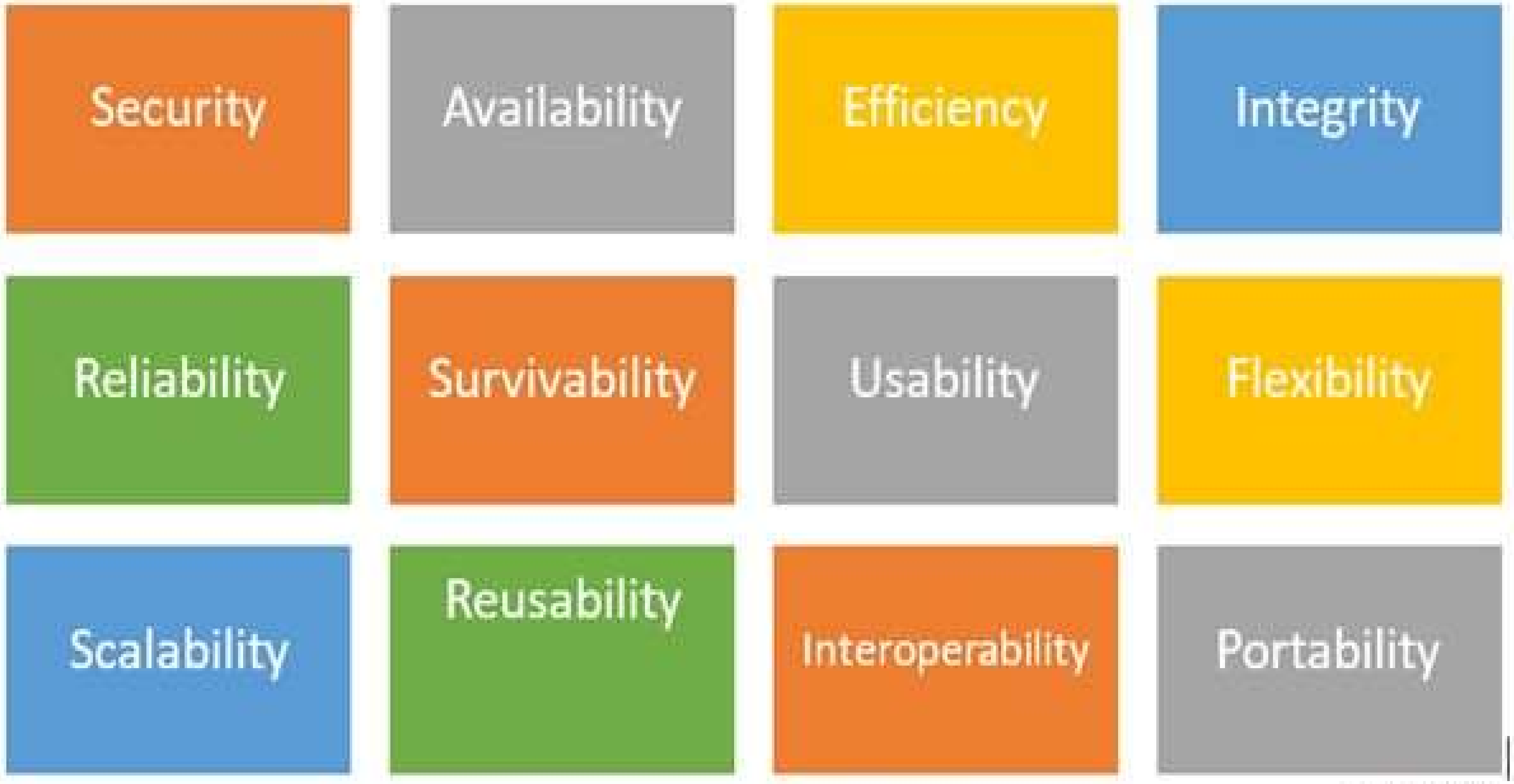
- The purpose of regression testing is to verify if code change introduces issues/defects into the existing functionality.
- There are so many kinds of possible changes that can impact the existing functionality in an application system.
- Even the simplest change to the code could impact previously tested functionality.
- Regression testing is performed when there is a code change in a software application.

- **Regression testing:-** Test the program's entire functionality to see if any thing changed when you added new code to the project.
  - Regression testing (which can also be performed at the unit level) is used to validate the updated software against the old set of test cases that have already been passed.
  - Regression testing helps to ensure that changes (due to testing or for other reasons) do not introduce unintended behavior or additional errors.


PROGRESS/DEVELOPMENT
REGRESSION

## Testing Non-Functional Requirements.

- **NON-FUNCTIONAL TESTING** is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application.
- It is designed to test the readiness of a system as per non-functional parameters which are never addressed by functional testing.



Non Functional Testing Parameters

# Non-functional testing includes:

- Baseline testing
- Compliance testing
- Documentation testing
- Endurance testing or reliability testing
- Load testing
- Localization testing and Internationalization testing
- **Performance testing**
- **[illegible]**

**Please note:-** Those given in bold means portions already covered in first module with testing part. check those slides….

- **Performance and scalability.** How fast does the system return results? How much will this performance change with higher workloads?
- **Portability and compatibility.** Which hardware, operating systems, browsers, and their versions does the software run on? Does it conflict with other applications and processes within these environments?
- **Reliability, availability, maintainability.** How often does the system experience critical failures? and how much time is it available to users against downtimes?
- **Security.** How are the system and its data protected against attacks?
- **Localization.** Does the system match local specifics?
- **Usability.** How easy is it for a customer to use the system?