Unit- V
Software Test Automation

- Dr. Shivani Budhkar

Software Test Automation

- Developing software to test the software is called Test Automation
- Test automation saves time as software can execute test cases faster than human do.
- Test automation can free the Test engineers from mundane tasks and make them more focus on more creative task.
- Automated test can be more reliable
- Automation helps in immediate testing
- Automation can protect an organization against attrition of test engineers
- Test automation opens up opportunities for better utilization of global resources
- Certain types of tests can not be executed without automation
- Automation means end-to-end, not test execution alone

Skills needed for automation

- There are different “Generations of Automation”.
- The skills required for the automation depends on what generation of automation the company is in or desires to be in the near future
- Automation testing is broadly classified into generations
- 1st Generation- Record and Playback
- Skills for Test case Automation
  - Scripting languages
  - Record-Playback tool usage
- 2nd Generation- Modular and Data Driven
- Skills for Test case Automation
  - Programming Languages
  - Knowledge of data generation techniques
  - Usage of product under test
- 3rd Generation – Action Driven (Focus on Test case automation & framework design)
- Skills for Test case Automation
  - Scripting languages
  - Programming Languages
  - Design and Architecture of the product under test
  - Usage of the framework
- Skills for Framework
  - Programming Languages
  - Design and Architecture of the product under test
  - Generic Test requirements for multiple products

Skills needed for automation

- 4th Generation- BDD(Behavior Driven Development) and Hybrid
  - Programming Languages
  - **Designing the Frameworks**
  - **Creating the Test Scripts**
  - Understanding of Frameworks
- 4.5th Generation- Selenium Based Frameworks
  - Programming Languages
  - **Designing the Frameworks**
  - **Creating the Test Scripts**
  - Expertise With Automation Tools
  - Understanding The Business Requirements
- 5th Generation- Rise of AI, Full-Stack Frameworks
  - Programming Languages
  - **Designing the Frameworks**
  - **Creating the Test Scripts**
  - Expertise With Automation Tools
  - Understanding The Business Requirements
  - Understanding of Frameworks
  - Troubleshooting The Automation Tools
  - Experience With Test Management Tools
  - Knowledge Of Different Development Methodologies

Skills needed for Automation

- Automation Testing is the key to the business success of the software industry.
- With automation, you can expand your business to a larger audience saving both time and effort.
- As an automation tester, it is very essential to have certain skills which help in testing the application better.
- *Focus on analytical thinking*
- *Understanding of programming languages*
- *Good functional testing skills*
- *Expertise in creation of test scripts*
- *Possess good knowledge on Automated testing tools*
- *Clear understanding of business requirements*
- *Well versed with agile, DevOps and continuous delivery*
- *Maintain good communication and interaction with stakeholders*
- *Curious to learn new technologies and trends*
- *Able to assess and mitigate the risk*
- *Good problem-solving skills*
- *Good reporting skills*
- *Excellent at time management*
- *Passion for automation*
- *Online Automation Testing Courses and Certifications*

Scope of Automation

- Automation requirements define what needs to be automated looking into various aspects
- Identify the Types of Testing Amenable to Automation
  - Stress, reliability, scalability, performance testing
  - Regression Tests
  - Functional Tests
- Automating Areas Less prone to change
- Automate Tests that pertain to standards
- Management Aspects in Automation

Scope of Automation

- The scope of automation is the area of your Application Under Test which will be automated.
- Following points help determine scope:
- The features that are important for the business
- Scenarios which have **a large amount of data**
- **Common functionalities** across applications
- Technical feasibility
- The extent to which business components are reused
- **The complexity** of test cases
- Ability to use the same test cases for cross-browser testing

Scope of Automation

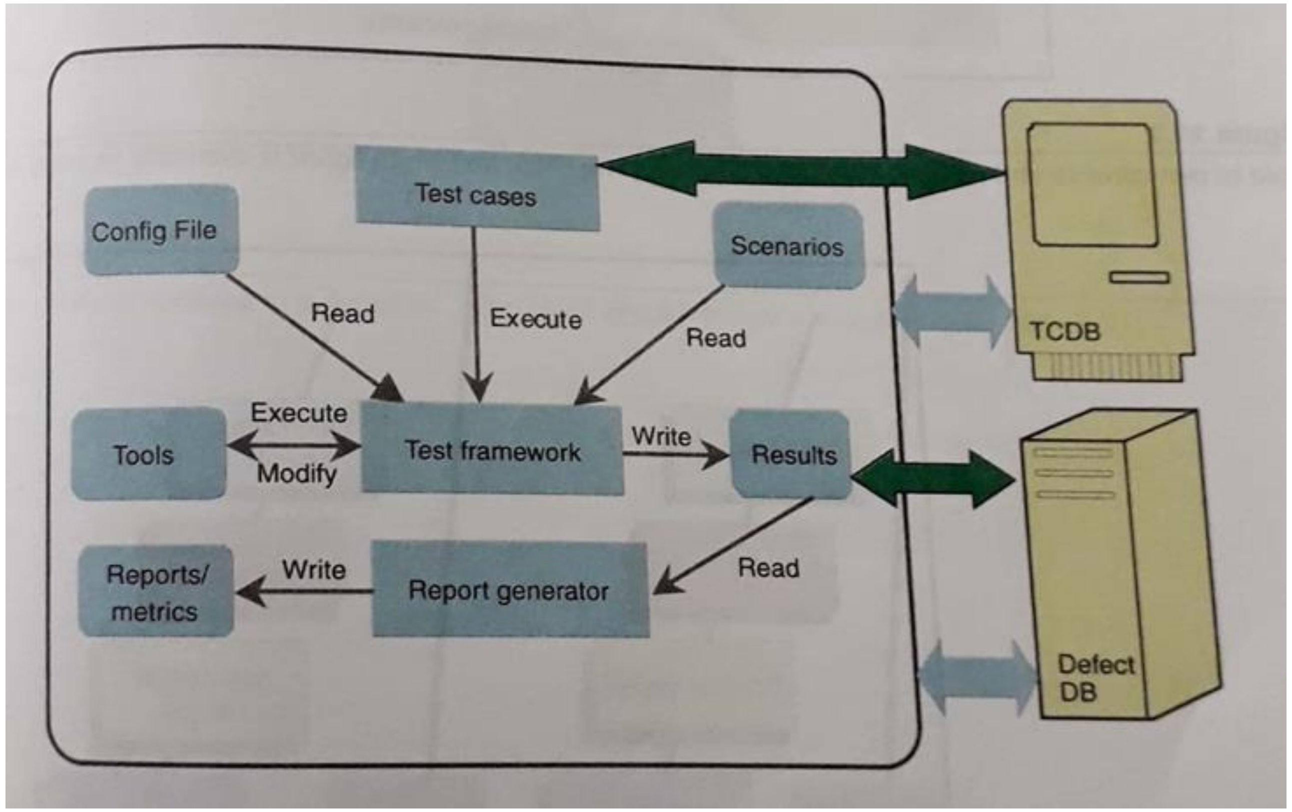
- Automation is the process of evaluating the AUT(Application under Test) against the specification with the help of a tool.
- Depending on the nature of testing there are two main branches under automation.
- **Functional testing with automation**
- It is advisable to keep at least 60 to 70 % of regression cases to be automated for being adherent to the timelines of test case execution. Automating test cases is not desirable if it is a onetime testing.
- **Performance testing with automation**
- Performance testing is almost impossible by manual means. There are different tools used across organizations for evaluating application performance. There are different divisions under performance testing based on the nature of testing.
    - Load testing
    - Stress testing

Requirements for a Test Tool

1. No hard coding in test suit
2. Test case/suit expandability
3. Reuse of code for different types of testing, test cases
4. Automatic setup and clean-up
5. Independent Test cases
6. Test case dependency
7. Insulating test cases during execution
8. Coding standards and directory structures
9. Selective execution of test cases
10. Random execution of test cases
11. Parallel execution of test cases
12. Looping the test cases
13. Grouping of test scenarios
14. Test case execution based on previous results
15. Remote execution of test cases
16. Automatic Archival of test data
17. Reporting Scheme
18. Independent of languages
19. Portability to different Platforms

Design and Architecture for Automation

- Components of Test Automation



Design and Architecture for Automation

- External Modules
- Scenario and Configuration File Modules
- Test case and Test Framework Module
- Tools and Results modules
- Report Generator and Reports/ Metrics Modules

Challenges in Automation

- Management Commitment
- Management should have patience and persist with automation
- Effective Communicating and Collaborating in Team
- Selecting a Right Tool
- Demanding Skilled Resources
- Selecting a Proper Testing Approach
- High Upfront Investment Cost

Tracking the Bug

- Bug tracking is the process of logging and monitoring bugs or errors during software testing.
- It is also referred to as defect tracking or issue tracking.
- Large systems may have hundreds or thousands of defects. Each needs to be evaluated, monitored and prioritized for debugging. In some cases, bugs may need to be tracked over a long period of time.
- Defect tracking is an important process in software engineering, as complex and business critical systems have hundreds of defects. One of the challenging factors is managing, evaluating and prioritizing these defects. The number of defects gets multiplied over a period of time and to effectively manage them, a defect tracking system is used to make the job easier.
- Defect tracking helps ensure that bugs found in the system actually get fixed

Tracking the Bug

- **For Bug tracking software, it is essential to have:**
- **Reporting facility** – complete with fields that will let you provide information about the bug, environment, module, severity, screenshots, etc.
- **Assigning** – What good is a bug when all you can do is find it and keep it to yourself, right?
- Progressing through life cycle stages – Workflow
- History/work logs/comments
- **Reports** – graphs or charts
- **Storage and Retrieval** – Every entity in a testing process needs to be uniquely identifiable. The same rule applies to bugs too. A bug tracking tool must provide a way to have an ID that can be used to store, retrieve (search) and organize bug information

Tracking the Bug

- Most Popular Bug Tracking Software:
- Backlog
- Katalon TestOps
- BugHerd
- Bird Eats Bug
- Bugzilla
- JIRA
- IBM Rational ClearQuest
- Lighthouse
- Zoho Bug Tracker
- The Bug Genie
- BugHost

Debugging

- Debugging is the process of fixing a bug in the software. **Debugging** is the process of locating the cause of a software error and correcting it.
- It can defined as the identifying, analysing and removing errors.
- This activity begins after the software fails to execute properly and concludes by solving the problem and successfully testing the software.
- It is considered to be an extremely complex and tedious task because errors need to be resolved at all stages of debugging.
- Once a test case has been executed and a bug located, debugging begins. Software engineers are only ever presented with the software's behaviour, and they do not directly see the error's cause. Debugging, then, relies on the software engineer to determine from the software's incorrect behaviour the causes of this behaviour.

Debugging

- Debugging begins with the software failing a test case. The software engineer debugging the software will then hypothesise a possible cause and implement the needed changes to correct this in the software. The failed test is then rerun. Further test cases may also be written to help narrow down the actual cause. This all occurs iteratively, with each step hopefully providing more information to the developer as to the root cause of the error.
- A number of debugging tactics have been proposed. They can be used alone, although they become far more effective when used in combination with each other.

Debugging

- **Brute force debugging -**This is conceptually the simplest of the methods, and often the least successful. This involves the developer manually searching through stack-traces, memory-dumps, log files, and so on, for traces of the error. Extra output statements, in addition to break points, are often added to the code in order to examine what the software is doing at every step.
- **Backtracking -**This method has the developer begin with the code that immediately produces the observable error. The developer than backtracks through the execution path, looking for the cause. Unfortunately, the number of execution paths which lead to any given point in the software can become quite large the further the cause of the bug is from where the error occurs, and so this method can become impractical.
- **Cause elimination -**In this method, the developer develops hypotheses as to why the bug has occurred. The code can either be directly examined for the bug, or data to test the hypothesis can be constructed. This method can often result in the shortest debug times, although it does rely on the developers understanding the software well.
- **Bisect -**Bisect is a useful method for locating bugs which are new to the software. Previous versions of the software are examined until a version which does not have the error is located. The difference between that version's source code and the next is then examined to find the bug.

Difference between testing and debugging

Testing	Debugging
Testing is the process to find bugs and errors.	Debugging is the process to correct the bugs found during testing.
It is the process to identify the failure of implemented code.	It is the process to give the absolution to code failure.
Testing is the display of errors.	Debugging is a deductive process.
Testing is done by the tester.	Debugging is done by either programmer or developer.
There is no need of design knowledge in the testing process.	Debugging can't be done without proper design knowledge.
Testing can be done by insider as well as outsider.	Debugging is done only by insider. Outsider can't do debugging.
Testing can be manual or automated.	Debugging is always manual. Debugging can't be automated.
It is based on different testing levels i.e. unit testing, integration testing, system testing etc.	Debugging is based on different types of bugs.
Testing is a stage of software development life cycle (SDLC).	Debugging is not an aspect of software development life cycle, it occurs as a consequence of testing.
Testing is composed of validation and verification of software.	While debugging process seeks to match symptom with cause, by that it leads to the error correction.
Testing is initiated after the code is written.	Debugging commences with the execution of a test case.

Automation Testing vs Manual Testing

Parameter	Automation Testing	Manual Testing
Definition	Automation Testing uses automation tools to execute test cases.	In manual testing, test cases are executed by a human tester and software.
Processing time	Automated testing is significantly faster than a manual approach.	Manual testing is time-consuming and takes up human resources.
Exploratory Testing	Automation does not allow random testing	Exploratory testing is possible in Manual Testing
Initial investment	The initial investment in the automated testing is higher. Though the ROI is better in the long run.	The initial investment in the Manual testing is comparatively lower. ROI is lower compared to Automation testing in the long run.
Reliability	Automated testing is a reliable method, as it is performed by tools and scripts. There is no testing Fatigue.	Manual testing is not as accurate because of the possibility of the human errors.
UI Change	For even a trivial change in the UI of the AUT, Automated Test Scripts need to be modified to work as expected	Small changes like change in id, class, etc. of a button wouldn't thwart execution of a manual tester.
Investment	Investment is required for testing tools as well as automation engineers	Investment is needed for human resources.
Cost-effective	Not cost effective for low volume regression	Not cost effective for high volume regression.
Test Report Visibility	With automation testing, all stakeholders can login into the automation system and check test execution results	Manual Tests are usually recorded in an Excel or Word, and test results are not readily/ readily available.
Human observation	Automated testing does not involve human consideration. So it can never give assurance of user-friendliness and positive customer experience.	The manual testing method allows human observation, which may be useful to offer user-friendly system.
Performance Testing	Performance Tests like Load Testing, Stress Testing, Spike Testing, etc. have to be tested by an automation tool compulsorily.	Performance Testing is not feasible manually
Parallel Execution	This testing can be executed on different operating platforms in parallel and reduce test execution time.	Manual tests can be executed in parallel but would need to increase your human resource which is expensive

Automation Testing vs Manual Testing

Parameter	Automation Testing	Manual Testing
Batch testing	You can Batch multiple Test Scripts for nightly execution.	Manual tests cannot be batched.
Programming knowledge	Programming knowledge is a must in automation testing.	No need for programming in Manual Testing.
Set up	Automation test requires less complex test execution set up.	Manual testing needs have a more straightforward test execution setup
Engagement	Done by tools. Its accurate and never gets bored!	Repetitive Manual Test Execution can get boring and error-prone.
Ideal approach	Automation testing is useful when frequently executing the same set of test cases	Manual testing proves useful when the test case only needs to run once or twice.
Build Verification Testing	Automation testing is useful for Build Verification Testing (BVT).	Executing the Build Verification Testing (BVT) is very difficult and time-consuming in manual testing.
Deadlines	Automated Tests have zero risks of missing out a pre-decided test.	Manual Testing has a higher risk of missing out the pre-decided test deadline.
Framework	Automation testing uses frameworks like Data Drive, Keyword, Hybrid to accelerate the automation process.	Manual Testing does not use frameworks but may use guidelines, checklists, stringent processes to draft certain test cases.
Documentation	Automated Tests acts as a document provides training value especially for automated unit test cases. A new developer can look into a unit test cases and understand the code base quickly.	Manual Test cases provide no training value
Test Design	Automated Unit Tests enforce/drive Test Driven Development Design.	Manual Unit Tests do not drive design into the coding process
Devops	Automated Tests help in Build Verification Testing and are an integral part of DevOps Cycle	Manual Testing defeats the automated build principle of DevOps
When to Use?	Automated Testing is suited for Regression Testing, Performance Testing, Load Testing or highly repeatable functional test cases	Manual Testing is suitable for Exploratory, Usability and Adhoc Testing. It should also be used where the AUT changes frequently