UNIT 5

AUTOMATION AND TESTING TOOLS

The testing techniques, either static or dynamic, can be automated with the help of software tools which can greatly enhance the software testing process. Testing today is not a manual operation but is assisted with many efficient tools that help the testers. Testing automation is effective such that any kind of repetitive and tedious activities can be done by machines, and testers can utilize their time in more creative and critical work.

In the last decade, a lot of testing tools have been developed for use throughout the various SDLC phases. But the major part is the selection of tools from a pool of various categories of tools. Apart from the high cost of these tools, a single tool may not cover the whole testing automation. Thus, tools must be selected according to their application and needs of the organization. Automation is bound to fail if chosen for wrong reasons at the wrong places in SDLC.

However, automated testing should not be viewed as a replacement for manual testing. There is a misconception among professionals that software testing is easy as you only run the test cases with automated tools. This is the reason why newcomers prefer testing jobs. But the truth is that testers have many duties in the development and it is not only about running automated tools. There are many activities in the testing life cycle which cannot be automated and manual effort is required. Thus, automated tools are merely a part of the solution; they are not a magical answer to the testing problem. Automated testing tools will never replace the analytical skills required to conduct the test, nor will they replace manual testing. It must be seen as an enhancement to the manual testing process.

NEED FOR AUTOMATION: -

If an organization needs to choose a testing tool, the following benefits of automation must be considered

1. **Reduction of testing effort: -** In verification and validation strategies, numerous test case design methods have been studied. Test cases for complete software may be hundreds of thousands or more in number. Executing all of them manually takes a lot of testing effort and time. Thus, execution of test suits through software tools greatly reduces the amount of time required.
2. **Reduces the testers' involvement in executing tests: -** Sometimes executing the test cases takes a long time. Automating this process of executing the test suit will relieve the testers to do some other work, thereby increasing the parallelism in testing efforts.
3. **Facilitates regression testing: -** As we know, regression testing is the most timeconsuming process. If we automate the process of regression testing, then testing effort as well as the time taken will reduce as compared to manual testing.
4. **Avoids human mistakes: -** Manually executing the test cases may incorporate errors in the process or sometimes, we may be biased towards limited test cases while checking the software. Testing tools will not cause these problems which are introduced due to manual testing
5. **Reduces overall cost of the software: -** As we have seen that if testing time increases, cost of the software also increases. But due to testing tools, time and therefore cost can be reduced to a greater level as testing tools ease the burden of the test case production and execution.
6. **Simulated testing: -** Load performance testing is an example of testing where the real-life situation needs to be simulated in the test environment. Sometimes, it may not be possible to create the load of a number of concurrent users or large amount of data in a project. Automated tools, on the other hand, can create millions of concurrent virtual users/data and effectively test the project in the test environment before releasing the product.
7. **Internal testing: -** Testing may require testing for memory leakage or checking the coverage of testing. Automation tools can help in these tasks quickly and accurately, whereas doing this manually would be cumbersome, inaccurate, and time-consuming.
8. **Test enablers: -** While development is not complete, some modules for testing are not ready. At that time, stubs or drivers are needed to prepare data, simulate environment, make calls, and then verify results. Automation reduces the effort required in this case and becomes essential.

9. **Test case design: -** Automated tools can be used to design test cases also. Through automation, better coverage can be guaranteed, than if done manually.

CATEGORIZATION OF TESTING TOOLS: -

A single tool may not cover the whole testing process, therefore, a variety of testing tools are available according to different needs and users. The different categories of testing tools are

STATIC AND DYNAMIC TESTING TOOLS: -

These tools are based on the type of execution of test cases, namely static and dynamic, as discussed in software testing techniques:

Static testing tools: -

For static testing, there are static program analysers which scan the source program and detect possible faults and anomalies. These static tools parse the program text, recognize the various sentences, and detect the following:

- Statements are well-formed.
- Inferences about the control flow of the program.
- Compute the set of all possible values for program data.

Static tools perform the following types of static analysis:

1. **Control flow analysis: -** This analysis detects loops with multiple exits and entry points and unreachable code.
2. **Data use analysis: -** It detects all types of data faults.
3. **Interface analysis: -** It detects all interface faults. It also detects functions which are never declared and never called or function results that are never used.
4. **Path analysis: -** It identifies all possible paths through the program and unravels the program's control.

Dynamic testing tools: -

These tools support the following:

- Dynamic testing activities.
- Many a times, systems are difficult to test because several operations are being performed concurrently. In such cases, it is difficult to anticipate conditions and generate representative test cases. Automated test tools enable the test team to capture the state of events during the execution of a program by preserving a snapshot of the conditions. These tools are sometimes called program monitors. The monitors perform the following functions:
  - List the number of times a component is called or line of code is executed. This information about the statement or path coverage of their test cases is used by testers.
  - Report on whether a decision point has branched in all directions, thereby providing information about branch coverage.
  - Report summary statistics providing a high-level view of the percentage of statements, paths, and branches that have been covered by the collective set of test cases run. This information is important when test objectives are stated in terms of coverage.

TESTING ACTIVITY TOOLS

These tools are based on the testing activities or tasks in a particular phase of the SDLC. Testing activities can be categorized as

- Reviews and inspections
- Test planning
- Test design and development
- Test execution and evaluation

Tools for review and inspections: -
Since these tools are for static analysis on many items, some tools are designed to work with specifications but there are far too many tools available that work exclusively with code. In this category, the following types of tools are required:

- **Complexity analysis tools: -** It is important for testers that complexity is analysed so that testing time and resources can be estimated. The complexity analysis tools analyse the areas of complexity and provide indication to testers.
-
- **Code comprehension: -** These tools help in understanding dependencies, tracing program logic, viewing graphical representations of the program, and identifying the dead code. All these tasks enable the inspection team to analyse the code extensively.

Tools for test planning: -
The types of tools required for test planning are:
1. Templates for test plan documentation
2. Test schedule and staffi ng estimates
3. Complexity analyser

Tools for test design and development: -
Discussed below are the types of tools required for test design and development.

Test data generator: - It automates the generation of test data based on a user defined format. These tools can populate a database quickly based on a set of rules, whether data is needed for functional testing, data-driven load testing, or performance testing.

Test case generator: - It automates the procedure of generating the test cases. But it works with a requirement management tool which is meant to capture requirements information. Test case generator uses the information provided by the requirement management tool and creates the test cases. The test cases can also be generated with the information provided by the test engineer regarding the previous failures that have been discovered by him. This information is entered into this tool and it becomes the knowledge-based tool that uses the knowledge of historical figures to generate test cases.

Test execution and evaluation tools: -
The types of tools required for test execution and evaluation are

Capture/playback tools: - These tools record events (including keystrokes, mouse activity, and display output) at the time of running the system and place the information into a script. The tool can then replay the script to test the system.

Coverage analysis tools: - These tools automate the process of thoroughly testing the software and provide a quantitative measure of the coverage of the system being tested. These tools are helpful in the following:
- Measuring structural coverage which enables the development and test teams to gain insight into the effectiveness of tests and test suites.
- Quantifying the complexity of the design
- Help in specifying parts of the software which are not being covered
- Measure the number of integration tests required to qualify the design
- Help in producing integration tests
- Measuring the number of integration tests that have not been executed
- Measuring multiple levels of test coverage, including branch, condition, decision/condition, multiple conditions, and path coverage.

Memory testing tools: - These tools verify that an application is properly using its memory resources. They check whether an application is:

- Not releasing memory allocated to it
- Overwriting/over reading array bounds
- Reading and using uninitialized memory

Test management tools: - Test management tools try to cover most of the activities in the testing life cycle. These tools may cover planning, analysis, and design. Some test management tools such as Rational's TestStudio are integrated with requirement and configuration management and defect tracking tools, in order to simplify the entire testing life cycle

Network-testing tools: - There are various applications running in the clientserver environments. However, these applications pose new complexity to the testing effort and increases potential for errors due to inter-platform connectivity. Therefore, these tools monitor, measure, test, and diagnose performance across an entire network including the following:

- Cover the performance of the server and the network
- Overall system performance
- Functionality across server, client, and the network

Performance testing tools: - There are various systems for which performance testing is a must but this becomes a tedious job in real-time systems. Performance testing tools help in measuring the response time and load capabilities of a system.

SELECTION OF TESTING TOOLS: -

The big question is how to select a testing tool. It may depend on several factors. What are the needs of the organization; what is the project environment; what is the current testing methodology; all these factors should be considered when choosing testing tools. Some guidelines to be followed by selecting a testing tool are given below.

1. **Match the tool to its appropriate use: -** Before selecting the tool, it is necessary to know its use. A tool may not be a general one or may not cover many features. Rather, most of the tools are meant for specific tasks. Therefore, the tester needs to be familiar with both the tool and its uses in order to make a proper selection.

2. **Select the tool to its appropriate SDLC phase: -** Since the methods of testing changes according to the SDLC phase, the testing tools also change. Therefore, it is necessary to choose the tool according to the SDLC phase, in which testing is to be done.

3. **Select the tool to the skill of the tester: -** The individual performing the test must select a tool that conforms to his skill level. For example, it would be inappropriate for a user to select a tool that requires programming skills when the user does not possess those skills.

4. **Select a tool which is affordable: -** Tools are always costly and increase the cost of the project. Therefore, choose the tool which is within the budget of the project. Increasing the budget of the project for a costlier tool is not desired. If the tool is underutilization, then added cost will have no benefits to the project. Thus, once you are sure that a particular tool will really help the project, then only go for it otherwise it can be managed without a tool also.

5. **Determine how many tools are required for testing the system: -** A single tool generally cannot satisfy all test requirements. It may be possible that many test tools are required for the entire project. Therefore, assess the tool as per the test requirements and determine the number and type of tools required.

6. **Select the tool after examining the schedule of testing: -** First, get an idea of the entire schedule of testing activities and then decide whether there is enough time for learning the testing tool and then performing automation with that tool. If there is not enough time to provide training on the tool, then there is no use of automation.

COSTS INCURRED IN TESTING TOOLS: -

Automation is not free. Obviously employing the testing tools incur a high cost. Moreover, before acquiring the tools, significant work is required. Following are some facts pertaining to the cost incurred in testing tools

1. **Automated script development: -** Automated test tools do not create test scripts. Therefore, a significant time is needed to program the tests. Scripts are themselves programming languages. Thus, automating test execution requires programming exercises.

2. **Training is required: -** It is not necessary that the tester will be aware of all the tools and can use them directly. He may require training regarding the tool, otherwise it ends up on the shelf or implemented inefficiently. Therefore, it becomes necessary that in a new project, cost of training on the tools should also be included in the project budget and schedule.

3. **Configuration management: -** Configuration management is necessary to track large number of fi les and test related artifacts.

4. **Learning curve for the tools: -** There is a learning curve in using any new tool. For example, test scripts generated by the tool during recording must be modified manually, requiring tool-scripting knowledge in order to make the script robust, reusable, and maintainable.

5. **Testing tools can be intrusive: -** It may be necessary that for automation some tools require that a special code is inserted in the system to work correctly and to be integrated with the testing tools. These tools are known as intrusive tools which require addition of a piece of code in the existing software system. Intrusive tools pose the risk that defects introduced by the code inserted specifically to facilitate testing could interfere with the normal functioning of the system.

6. **Multiple tools are required: -** As discussed earlier, it may be possible that your requirement is not satisfied with just one tool for automation. In such a case, you have to go for many tools which incur a lot of cost.

GUIDELINES FOR AUTOMATED TESTING: -

Automation is not a magical answer to the testing problem. Testing tools can never replace the analytical skills required to conduct testing and manual testing. It incurs some cost as seen above and it may not provide the desired solution if you are not careful. Therefore, it is necessary that you carefully plan the automation before adopting it. Decide which tool and how many tools are required, how much resources are required including the cost of the tool and the time spent on training.

The guidelines to be followed if you have planned for automation in testing are

1. **Consider building a tool instead of buying one, if possible: -** It may not be possible every time. But if the requirement is small and sufficient resources allow, then go for building the tool instead of buying, after weighing the pros and cons. Whether to buy or build a tool requires management commitment, including budget and resource approvals.

2. **Test the tool on an application prototype: -** While purchasing the tool, it is important to verify that it works properly with the system being developed. However, it is not possible as the

system being developed is often not available. Therefore, it is suggested that if possible, the development team can build a system prototype for evaluating the testing tool.

3. **Not all the tests should be automated: -** Automated testing is an enhancement of manual testing, but it cannot be expected that all test on a project can be automated. It is important to decide which parts need automation before going for tools. Some tests are impossible to automate, such as verifying a printout. It has to be done manually.

4. **Select the tools according to organizational needs: -** Do not buy the tools just for their popularity or to compete with other organizations. Focus on the needs of the organization and know the resources (budget, schedule) before choosing the automation tool.

5. **Use proven test-script development techniques: -** Automation can be effective if proven techniques are used to produce efficient, maintainable, and reusable test scripts. The following are some hints:
   - Read the data values from either spreadsheets or tool-provided data pools, rather than being hard-coded into the test-case script because this prevent test cases from being reused. Hard-coded values should be replaced with variables and whenever possible read data from external sources.
   - Use modular script development. It increases maintainability and readability of the source code.
   - Build library of reusable functions by separating the common actions into shared script library usable by all test engineers.
   - All test scripts should be stored in a version control tool.

6. **Automate the regression tests whenever feasible: -** Regression testing consumes a lot of time. If tools are used for this testing, the testing time can be reduced to a greater extent. Therefore, whenever possible, automate the regression test cases.

OVERVIEW OF SOME COMMERCIAL TESTING TOOLS: -

- **Mercury Interactive's WinRunner: -** It is a tool used for performing functional / regression testing. It automatically creates the test scripts by recording the user interactions on GUI of the software. These scripts can be run repeatedly whenever needed without any manual intervention. The test scripts can also be modified if required because there is support of Test Script language (TSL) with a 'C' like syntax. There is also provision for bringing the application to a known state if any problem has occurred during automated testing. WinRunner executes the statements by default with an interleaving of one second. But if some activities take more time to complete, then it synchronizes the next test cases automatically by waiting for the current operations to be completed.

- **Segue Software's SilkTest: -** This tool is also for functional/regression testing. It supports 4Test as a scripting language which is an object-oriented scripting language. SilkTest has a provision for customized in-built recovery system which helps in continuing the automated testing even if there is some failure in between.

- **IBM Rational SQA Robot: -** It is another powerful tool for functional/regression testing. Synchronization of test cases with a default delay of 20 seconds is also available.

- **Mercury Interactive's LoadRunner: -** This tool is used for performance and load testing of a system. Generally, the tool is helpful for client/server applications of various parameters with their actual load like response time, the number of users, etc. The major benefit of using this tool is that it creates virtual users on a single machine and tests the system on various parameters. Thus, performance and load testing is done with minimum infrastructure.

- **Apache's JMeter: -**This is an open-source software tool used for performance and load testing.

- **Mercury Interactive's TestDirector: -** TestDirector is a test management tool. It is a web-based tool with the advantage of managing the testing if two teams are at different locations. It manages the test process with four phases: specifying requirements, planning tests, running tests, and tracking defects. Therefore, there is advantage that the tests are planned as per the requirements evolved. Once the test plan is ready, the test cases are executed. Defect-tracking can be done in any phase of test process. During defect-tracking, new bug can be reported; assign responsibility to someone for bug repair; assign priority to the bug; bug repair status can be analysed, etc. This tool can also be integrated with LoadRunner or WinRunner.

REVIEW QUESTIONS: -

1. What is the need of automating the testing activities?
2. What is the difference between static and dynamic tools?
3. Analyse and report which tools are available for test planning?
4. Analyse and report which tools are available for test design?
5. What are the guidelines for selecting a testing tool?
6. What are the costs incurred in adopting a testing tool?
7. Is a single testing tool sufficient for all testing activities?
8. Make a list of some important testing tools which are open-source on the Internet. Use them for testing your software.