MODULE 4: TEST AUTOMATION

-By
Asst Prof. Rohini M. Sawant

Automation and Testing Tools

- 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.
- 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.
- 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

- **Reduction of testing efforts:** SDLC for product development aiming at software testing and quality assurance, has to undergo verification and validation activities in each phase as per the V-V model. This has number of test activities and number of test cases in each test activity. Now if some of these are implemented with testing tools aiming towards automation, it will save a lot of testing efforts of the testers.
- **Reduction of testers involvements:** if the test cases are automated, they just can be monitored by semi-skilled testers and the main tester involvement being reduce can use their time for design and development.
- **Facilitates regression testing:** regression testing is most time consuming and boring as a lot of repetition is involved in it. These repetition if automated, will save a lot of time of the testers.
- **Avoids human mistakes:** manual testing if done without 100% involvement will introduce errors and for repeating same time of test cases will increase this. All such problems can be overcomed by automated tools.

Need for Automation

- **Reduces overall cost of software:** testing tools are costly, but by using the testing tools for repeated test, the testing time ,efforts etc are reduced. The testers get time for doing other fruitful activities of design and planning. This lets to overall quality development which thus brings saving in overall cost.

- **Simulated testing:** in case of integration testing, interfaces and data has to be stimulated by studs and drivers. This work can be made more easy by automation then doing it manually. Large number of virtual interfaces(fan-in and fan-out) can be implemented automatically much more easily as compared to manually.

- **Internal testing:** in a testing activity, some internal testing activities may have to be performed, like memory leakage check, coverage check or data coverage check etc. This activities if done manually will be cumbersome, Time consuming and may not be accurate. This can be automated for more accurate results and time saving.

- **Test Enablers:** stubs and drivers are some of the test enablers. Making their codes overhead on the tester. This work can be done by automated tools in a better and more efficient way.

- **Test cases design:** initially test cases design started manually but by experience and knowledge the process can be automated under complete control of the tester.

CATEGORIZATION OF TESTING TOOLS

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:

Control flow analysis This analysis detects loops with multiple exits and entry points and unreachable code.

Data use analysis It detects all types of data faults.

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.

Path analysis It identifies all possible paths through the program and unravels the program's control.

CATEGORIZATION OF TESTING TOOLS

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 [66] 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.

CATEGORIZATION OF TESTING TOOLS

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

CATEGORIZATION OF TESTING TOOLS

Now, we will discuss the testing tools based on these testing tasks.

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.

CATEGORIZATION OF TESTING TOOLS

Tools for test planning The types of tools required for test planning are:

1. Templates for test plan documentation
2. Test schedule and staffing 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.

CATEGORIZATION OF TESTING TOOLS

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.

CATEGORIZATION OF TESTING TOOLS

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.
- 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/overreading array bounds or Reading and using uninitialized memory
- 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

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.

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.

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.

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 under utilization, 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.

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.

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

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.

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.

Configuration management Configuration management is necessary to track large number of files and test related artifacts.

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.

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.

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

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.

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.

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.

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.

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.