UNIT – V
TEST AUTOMATION

Software test automation – Skills needed for automation – Scope of automation – Design and architecture for automation – Requirements for a test tool – Challenges in automation - Test metrics and measurements –Project, progress and productivity metrics

5.1 SOFTWARE TEST AUTOMATION

- Developing software to test the software is called automation. Test automation can help address several problems.
- Automation saves time as software can execute test cases faster than human do. The time thus saved can be used effectively for test engineers to
  - ✓ Develop additional test cases to achieve better coverage.
  - ✓ Perform some esoteric or specialized tests like ad hoc testing, or
  - ✓ Perform some extra manual testing.
- Test automation can free the test engineers from mundane tasks and make them focus on more creative tasks.
  - ✓ Automated tests 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 test cannot be executed without automation.
  - ✓ Automation means end-to-end, not test execution alone.

5.2 SKILLS NEEDED FOR AUTOMATION

Skills needed for Automation

- There are different ‐Generations of automation‖. The skills required for automation depends on what generation of automation the company is in or desires to be in the near future.
- The automation of testing is broadly classified into three generations.
  - ✓ First generation-Record and play back
  - ✓ Second generation-Data- Driven
  - ✓ Third generation- Action-Driven.

First Generation Record and Playback

- Record and playback avoids the repetitive nature of executing tests.
- A test engineer records the sequence of actions by the keyboard characters or mouse clicks and those recorded scripts are played back later, in the same orderas they were recorded. Since a recorded script can be played back multiple times, it reduces the tedium of testing functions.
- It is simple to record and save the script.
- This generation of tool has several disadvantages.
- The script may contain hard-coded values, thereby making it difficult to perform general types of tests. For example, when a report has to use the current data and time, it becomes difficult to use a recorded script. The handling error condition is left to the testers and thus, the played back scripts may require a lot of manual intervention to detect and correct error conditions.
- When the application changes, all the scripts have to be recorded, thereby increasing the test maintenance costs.

Second Generation – Data-Driven.

- This method helps in developing test scripts that generates the set of input conditions and corresponding expected output. This enables the tests to be repeated for different input and output conditions. The approach takes as much time and effort as the product.
- However the changes to application do not require the automated test cases to be changed as long as the input conditions and expected output are still valid. This generation of automation focuses on input and output conditions using the black box testing approach.

Third Generation Action-Driven

- This technique enables a layman to create automation tests. There are no input and expected output conditions required for running the tests.
- All actions that appear on the application are automatically tested, based on a generic set of controls defined for automation.
- The set of actions are represented as objects and those objects are reused.
- The user needs to specify only the operations [such as log in, download, and so on] and everything else that is needed for those actions are automatically generated.
- Automation in third generation involves two major aspects- test automation and frame work design.
- The skills needed for automation are classified into four levels for three generations as the generation of automation introduces two levels of skills for development of test cases and framework.

5.3 SCOPE OF AUTOMATION

- The automation requirements define what needs to be automated looking into the various aspects. The specific requirements can vary from product to product, from situations to situations, from time to time.
- Tips for identifying the scope for automation:
  - ✓ **Identifying the types of testing amenable to automation.**
  - ✓ **Automation areas are less prone to changes.**
  - ✓ **Automate test that pertain to standards.**
  - ✓ **Management aspects in automation**

Identifying the Types of Testing Amenable to Automation

- Certain types of tests automatically lend themselves to automation
  - ✓ Stress, reliability, scalability and performance testing.
  - ✓ Regression tests.
  - ✓ Functional tests.

Stress, Reliability Scalability and Performance Testing

- These types of testing require the test cases to be run from a large number of different machines for an extended period of time, such as 24 hours, 48 hours and so on. It is not possible to have hundreds of users trying out the product day in and day out.
- They may neither be willing to perform the repetitive tasks, nor will it be possible to find that many people with required skill set.

Regression Tests

- Regressions tests are repetitive in nature. These tests cases are executed multiple times during the product development phases.
- Given the repetitive nature of the test cases, automation will save significant time and effort in the long run.

Functional Tests

- These kinds of tests may require a complex set up and thus require specialized skill, which may not be available on an ongoing basis.
- Automating these once, using the expert skill sets, can enable using less skilled people to run these tests on an ongoing basis.

Automating Areas Less Prone To Change

- In a product scenario, the changes in requirements are quite common. In such situation, what to automate is easy answer. Automation should consider those areas where requirements go through lesser or no changes.

- Normally change in requirements cause scenarios and new features to be impacted, not the basic functionality of the product.

Automate Tests That Pertain To Standards

- One of the tests that products may have to undergo is compliance to standards. For example, a product providing a JDBC interface should satisfy the standards of IDBC tests. These tests undergo relatively less changes. Even if they do change, they provide backend compatibility by which automated scripts will continue to run.
- Automating for standards provides dual advantage. Test suites developed for standards are not only used for product testing but also be sold as tools for the market.
- A large number of tools available in the commercial market were internally developed for in-house usage.
- Hence automating for standards creates a new opportunities for them to be sold as commercial tools.
- Testing for standards have certain legal and organization requirements. To certify the software or hardware, a test suite is developed and handed over to different companies. The certification suites are executed every time by supporting organization before the release of software and hardware. This is called certification testing.

Management Aspects in Automation

- What to automate - it takes into account the technical and management aspects, as well as long-term version. Adequate effort has to be spent to obtain management commitment. The automated test cases need to be maintained till the product reaches obsolescence.
- Automation involves effort over an extended period of time, management permissions are only given in phases and part by part. Hence, automation effort should focus on those areas for which management commitment exists already.
- Return on investment is another aspect to be considered seriously. Effort estimates for automation should give a clear indication to the management on the expected return on investment.

5.4 DESIGN AND ARCHITECTURE FOR AUTOMATION

- The work on automation can go simultaneously with product development and can overlap with multiple releases of the product.
- Like multiple-product releases, automation also has several releases. One specific requirement for automation is that the delivery of the automated tests should be done before the test execution phase so that the deliverables from automation effort can be utilized for the current release of the product.
- The requirements for automation span multiple phases for multiple releases, like product requirements.
- Test execution may stop soon after releasing the product but automation effort continues after a product release

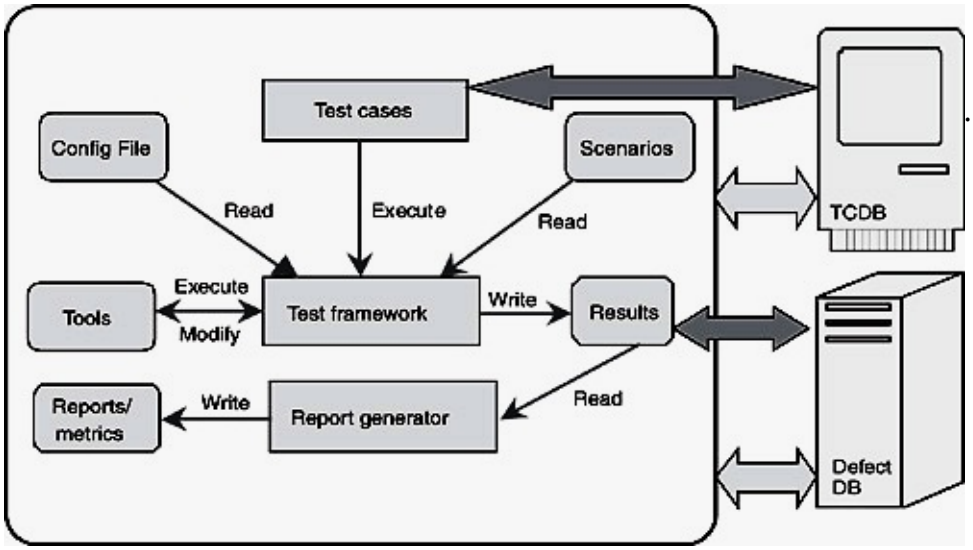

Figure 5.1: Components of Test automation

Selecting a Test Tool:

- Having identified the requirements of what to automate, a related question is the choice of an appropriate tool for automation. Even though we have listed these as two sequential steps, oftentimes, they are tightly interlinked.

Automation for Extreme Programming Model:

- Unit test cases are developed before coding phase starts;
- Code is written for test cases and are written to ensure test cases pass;
- All unit tests must run 100% all the time.
- Everyone owns the product; they often cross boundaries.

Modules:

- External Modules
- Scenario and configuration file modules
- Test cases and test framework modules
- Tools and results modules
- Report generator and report / metrics modules

External Modules

- There are two modules that are external modules to automation TCDB and defect DB, all the test cases, the steps to execute them and the history of their execution are stored in the TCDB.

- The test cases in TCDB can be manual or automated. The interfaces are shown by thich arrows represents the interaction between TCDB and the automation framework onl for automated test cases.

- Defect DB on defect database or defect repository contains details of all the defects that are found in various products that are tested in a particular organization.

Scenario and configuration file modules

- Scenarios are nothing but information on ―how to execute a particular test case‖.
- A configuration file contains a set of variables that are used in automation.
- The values of variables in this configuration file can be changed dynamically to achieve different execution, input, output and state conditions.

Test cases and test framework modules

- A test case means the automated test cases taken from TCDN and executed by the framework. A test case is an object for execution for other modules in the architecture and does not represent any interaction by itself.
- A test framework is a module that combines ‗what to execute' and ‗how they have to be executed'. It picks up the specific test cases that are automated from TCDB and picks up the scenarios and executes them.

Tools and results modules

- When a test framework performs its operations, there are a set of tools that may be required.

Report Generator and Report / Metrics Modules

- The module that takes the necessary inputs and prepares a formatted report is called a report generator.
- All the reports and metrics that are generated are stored in the reports/metrics module of automation for future use and analysis.

5.5 REQUIREMENTS FOR TEST TOOL

Generic Requirements for Test Tool/Framework

- While illustrating the requirements, we have used examples in a hypothetical metal language to drive home the concept.

Requirement 1: No Hard Coding In the Test Suite

- One of the important requirements for a test suite is to keep all variables separately in a file.
- By following this practice the source code for the test suite need not be modified every time it is required to run the tests for different values of the variables.
- This enables a person who does not know the program to change the values and run the test suite.
- The variables for the test suite are called configuration variables.
- The file in which all variable names and their associated values are kept is called configuration file.

Requirement 2: Test case / suite expandability

- In a product scenario involving several releases and several test cycles and defect fixes, the test cases go through large amount of changes and additionally there are situations for the new test cases to be

added to the test suite. The test case modification and new test case insertion should not result in the existing test cases failing.

- Test tools are not only used for one product having one test suite. Hayes is used for various products that may have multiple test suites. It is important for the test suits be added to the framework without affecting other test suites.

Requirement 3: Reuse Of Code For Different Types Of Testing, Test Cases.

- The functionality of the product when subjected to different scenarios becomes test cases for different type of testing. This encourages the reuse of code in automatic.
- By allowing the objectives of framework and test suites to take care of the ‐how‖ and ‐what‖ portions of automation respectively.
- Reuse of test cases can be increased. The reuse of code is not only applicable to various types of testing; it is also applicable for modules within automation.

Requirement 4: Automatic Set Up and Clean Up

- For each test case there could be some prerequisite to be met today before they are run.
- The test cases may expect some objects to be created or certain portion of the product to be configured in a particular way. If this portion is not met by automation, then it introduces some manual intervention before running the test cases.
- Each test program should have a set up program that will create the necessary setup before executing the test cases. The test frame should have the intelligence to find out what test cases are executed and call the appropriate set up program.

Requirement 5: Independent Test Cases

- The test cases need to be independent not only in design phase, but also in the execution phase. To execute a particular test case, it should not expect any other test case to have been executed before nor should it implicitly assume that certain other test case will be run after it.
- Each test case should be executed alone; there should be no dependency between test cases such as test case -2 to be executed after test case 1 and so on.
- This requirement enables the test engineer to select and execute any test case at random without worrying about other dependencies.

Requirement 6: Test Case Dependency

- Making test cases independent enables anyone case to be selected at random and executed. Making a test case dependent on another makes it necessary for a particular test case to be executed before or after a dependent test case is selected for execution.
- A test tool or a framework should provide both features. The framework should help to specify the dynamic dependencies between test cases.

Requirement 7: Insulating Test Cases during Execution

- Insulating test cases from the environment is an important requirement for the framework or test tool. At the time of test case execution, there could be some events or interrupts or signals in the system that may affect the execution.
- Consider the example of automatic pop-up screens on web browsers, when such pop-up screens happen during execution, this affects test case execution as the test suite may be expecting some other screen based on an earlier setup in the test case.
- Hence to avoid test cases failing due to some unforeseen events the framework should provide an option for user to block some of the events.

Requirement 8: Coding Standards and Directory Structures.

- Coding standard and proper directory structures for a test suite may help the new engineers in understanding the test suite fast and help in maintaining the test suite.

Requirement 9: Selective Execution of Test Cases.

- A framework may have multiple test suites; a test suite may have multiple test programs and a test program may have multiple test cases.

Requirement 10: Random Execution of Test Cases

- The same test engineer may sometime need to select a test case randomly froma list of test cases. Giving a set of test cases and expecting the test tool to select the test case is called random execution of test cases.

Requirement 11: Parallel Execution of Test Cases

- There are certain defects which can be unearthed if some of the test cases are run at the same time. In a multi-tasking and multi-processing operating systems it is possible to make several instances of the tests and make them run in parallel.

Requirement 12: Looping the Test Cases.

- Reliability testing requires the test cases to be executed in loop. There are two steps of loops that are available. One is the iteration loop which gives the number of iteration of the particular test case to be executed.
- The other is the limited loop, which keeps executing the test case to be executed in a loop till the specified time duration is reached.

Requirement 13: Grouping Of Test Scenarios.

- The group scenarios allow the selected test cases to be executed in order random, in a loop all at the same time. The grouping of scenarios allows several tests to be executed in predetermined combination of scenarios.

Requirement 14: Test Cases Execution Based On Previous Results

- Automation may not be of much help if the previous results of test execution are not considered for the choice of tests.

Requirement 15: Remote Execution of Test Cases.

- The central machine that allocates test to multiple machines and co-ordinate the execution and result is called test console or test monitor.
  - ✓ It should be possible to execute/stop the test suite on any machines/set of machine from the test console.
  - ✓ The test results and logs can be collected from the test console.
  - ✓ The progress of testing can be found from the test console.

Requirement 16: Automatic Archival Of Test Data

- Archival of test data must include
  - ✓ What configuration variable was used?
  - ✓ What scenario was used and
  - ✓ What programs were executed and from what path.

Requirement 17: Reporting Scheme

- Every test suite needs to have a reporting scheme from where meaningful reports can be extracted.
- The report generator is designed to develop dynamic reports; it is very difficult to say what information is needed and what not.
- Audit logs are very important to analyze the behavior of a test suite and a product.
- A reporting scheme should include
  - ✓ When the framework, scenario test suite, test program, and each test case were started/completed.
  - ✓ Result of each test case
  - ✓ Log messages.
  - ✓ Category of events and log of events.
  - ✓ Audit reports.

Requirement 18: Independent of Language

- A framework or test tool should provide a choice of language and scripts that are popular in the software development area. Irrespective of the languages/scripts are used for automation, the framework should function the same way, meeting all requirements.
- Many test tools force the test scripts to be used. This needs to be avoided as it affects the momentum of automation because a new language has to be learned.

- ✓ A framework should be independent of programming languages and scripts.
- ✓ A framework should provide choice of programming languages, scripts, and their combinations.
- ✓ A framework or test suite should not force a language/script.
- ✓ A framework or test suite should work with different test programs written using different languages and scripts.
- ✓ The internal scripts and options used by the framework should allow the developers of a test suite to migrate to better frame work.

Requirement 19: Portability to Different Platforms.

- ❖ Products being cross-platform and test framework networking on some of the platforms are not good for automation. Hence it is important for the test tools and framework to be cross-platform and be able to run on the same diversity of platforms and environments under which the product under test runs.

  - ✓ The framework and its interface should be supported on various platforms.
  - ✓ Portability to different platforms is a basic requirement for the test tool/test suite.
  - ✓ The language/script used in the test suite should be selected carefully so that it runs on different platforms.
  - ✓ The language/script written for the test suite should not contain platform specific calls.

5.6 CHALLENGES IN AUTOMATION

- ❖ There are number of problems that may be encountered in trying to automate testing. Having some idea of the type of problems that encounter should help in implementing effective automation regime. Few common problems are described below.

Unrealistic expectations:

- ❖ Generally there is a tendency to be optimistic/have high expectation about what can be achieved by a new test tool. It is human nature to hope that this new test solution will at last solve all of the problems we are currently experiencing.
- ❖ Vendors usually emphasize the benefits and successes, and may play down the amount of effort needed to achieve the desired benefits. If management expectations are unrealistic, then no matter how well the tool is implemented from a technical point of view, it will not meet expectations.

Expectation that automated tests will find a lot of new defects:

- ❖ A test might more likely find a defect the first time it is run. If a test has already run and passed, running the same test again is much less likely to find a new defect (unless the test is exercising code that has been changed or could be affected by a change made in a different part of the software, or is being run in a different environment).
- ❖ Test execution tools are _record – replay‘ tools, i.e. regression testing tools. Their use is in repeating tests that have already run. This is a very useful thing to do, but it is not likely to find a large number of new defects, particularly when run in the same hardware and software environment as before.

- Knowing that a set of tests has passed again gives confidence that the software is still working as well as it was before, and that changes elsewhere have not had unforeseen effects.

Poor testing practice:

- If testing practice is poor, with poorly organized/designed tests, little or inconsistent documentation and tests that are not very effective at finding defects, automating those tests is not a good idea.

Maintenance of automated tests:

- When software is changed it is often necessary to update some, or even entire test suite, so they can be re-run successfully. This is particularly true for automated tests.
- Test maintenance effort is the biggest challenge and often reason to truncate many test automation initiatives. When it takes more effort to update the tests than it would take to re-run those tests manually, test automation will be stopped.

False sense of security:

- Just because a test suite runs without finding any defects, it does not mean that there are no defects in the software. The tests may be incomplete, or may contain defects themselves.
- If the expected outcomes are incorrect, automated tests will simply preserve those defective results.

Technical problems of tools:

- Commercial test execution tools are software products, sold by vendor companies, they are not immune from defects or problems of support. Interoperability of the tool with other software, either your own applications or third-party products, can be a serious problem. Many tools look ideal on paper, but simply fail to work in some environments.
- In addition to technical problems with the tools themselves, we may experience technical problems with the software we are trying to test. If software is not designed and built with testability in mind, it can be very difficult to test, either manually or automatically. Trying to use tools to test such software will add complication which will only make test automation even more difficult.

Organizational issues:

- Automating testing is not a trivial exercise, and it needs to be well supported by management and implemented into the culture of the organization. Time must be allocated for choosing tools, for training, for experimenting and learning what works best, and for promoting tool use within the organization.
- Test automation is an infrastructure issue, not just a project issue. In large organizations, test automation can rarely be justified on the basis of a single project, since the project will bear all of the start-up costs and teething problems and may reap little of the benefits.
- If the scope of test automation is only for one project, people will then be assigned to new projects, and the automation initiative will be lost.

5.7 TEST METRICS AND MEASUREMENTS

A Metric is a quantitative measure of the degree to which a system, system component, or process possesses a given attribute.

Metrics can be defined as "STANDARDS OF MEASUREMENT"

- Software Metrics are used to measure the quality of the project. Simply, Metric is a unit used for describing an attribute. Metric is a scale for measurement.
- Suppose, in general, ‒Kilogram‖ is a metric for measuring the attribute ‒Weight‖. Similarly, in software, ‒How many issues are found in thousand lines of code?‖, here No. of issues is one measurement & No. of lines of code is another measurement. Metric is defined from these two measurements.
- **Test metrics example:**
  - ✓ How many defects are existed within the module?
  - ✓ How many test cases are executed per person?
  - ✓ What is the Test coverage %?

What is Software Test Measurement?

- Measurement is the quantitative indication of extent, amount, dimension, capacity, or size of some attribute of a product or process.
- Test measurement example: Total number of defects.
- The measurement of key parameters is an integral part of tracking.
  - ✓ Measurements first entail collecting a set of data. But, raw data by itself may not throw light on why a particular event has happened.
  - ✓ The collected data have to be analyzed in totality to draw the appropriate conclusions.

Why Metrics in Testing?

- Testing is the penultimate phase before product release, it is essential to measure the progress of testing and product quality.
- Tracking test progress and product quality can give a good idea about the release—whether it will be met on time with known quality. Measuring and producing metrics to determine the progress of testing is thus very important.
- Knowing only how much testing got completed does not answer the question on when the testing will get completed and when the product will be ready for release.
- To answer these questions, one needs to know how much more time is needed for testing.
- To judge the remaining days needed for testing, two data points are needed - remaining test cases yet to be executed and how many test cases can be executed per elapsed day.
- The test cases that can be executed per person day are calculated based on a measure called **test case execution productivity**.
- This productivity number is derived from the previous test cycles. It is represented by the formula, given alongside in the margin.
- Generation of Software Test Metrics is the most important responsibility of the Software Test Lead/Manager.
- Test Metrics are used to,
  - ✓ Take the decision for next phase of activities such as, estimate the cost & schedule of future projects.

- ✓ Understand the kind of improvement required to success the project
- ✓ Take decision on process or technology to be modified etc.

Importance of Software Testing Metrics:

- ❖ As explained above, Test Metrics are the most important to measure the quality of the software.

Suppose, if a project does not have any metrics, then how the quality of the work done by a Test analyst will be measured.

- ❖ For Example: A Test Analyst has to,
  - ✓ Design the test cases for 5 requirements
  - ✓ Execute the designed test cases
  - ✓ Log the defects & need to fail the related test cases
  - ✓ After the defect is resolved, need to re-test the defect & re-execute the corresponding failed test case.
- ❖ In above scenario, if metrics are not followed, then the work completed by the test analyst will be subjective i.e. the test report will not have the proper information to know the status of his work/project.
- ❖ If Metrics are involved in the project, then the exact status of his/her work with proper numbers/data can be published.
- ❖ In the Test report, we can publish:
  1. How many test cases have been designed per requirement?
  2. How many test cases are yet to design?
  3. How many test cases are executed?
  4. How many test cases are passed/failed/blocked?
  5. How many test cases are not yet executed?
  6. How many defects are identified & what is the severity of those defects?
  7. How many test cases are failed due to one particular defect? etc.

- ❖ Based on the project needs we can have more metrics than above mentioned list, to know the status of the project in detail.
- ❖ Based on the above metrics, test lead/manager will get the understanding of the below mentioned key points.

  a) %ge of work completed
  b) %ge of work yet to be completed
  c) Time to complete the remaining work
  d) Whether the project is going as per the schedule or lagging? etc.

- ❖ Based on the metrics, if the project is not going to complete as per the schedule, then the manager will raise the alarm to the client and other stake holders by providing the reasons for lagging to avoid the last minute surprises.

Types of Manual Test Metrics:

- Testing Metrics are mainly divided into 2 categories.

1. Base Metrics
2. Calculated Metrics

Base Metrics:

- Base Metrics are the Metrics which are derived from the data gathered by the Test Analyst during the test case development and execution.
- This data will be tracked throughout the Test Life cycle. i.e. collecting the data like, Total no. of test cases developed for a project (or) no. of test cases need to be executed (or) no. of test cases passed/failed/blocked etc.

Calculated Metrics:

- Calculated Metrics are derived from the data gathered in Base Metrics. These Metrics are generally tracked by the test lead/manager for Test Reporting purpose.

Metrics Life Cycle:

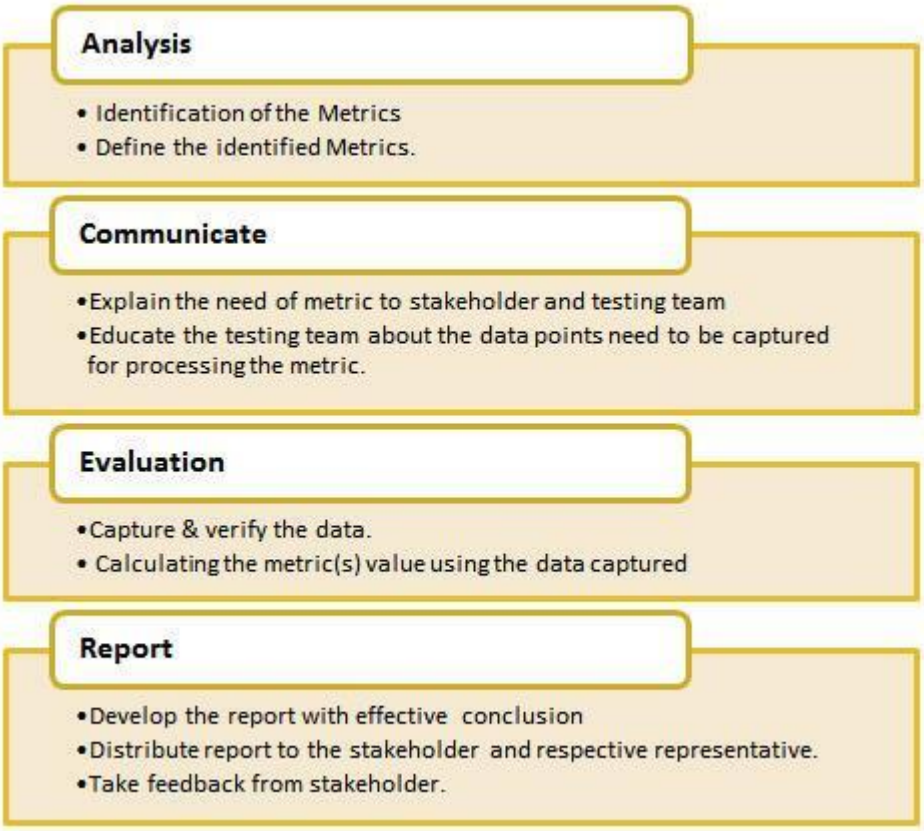

Figure 5.2: Examples of Software Testing Metrics

Examples of Software Testing Metrics:

- Example: To calculate various test metrics used in software test reports:

Below is the table format for the data retrieved from the test analyst who is actually involved in testing:

S.No.	Testing Metric	Data retrieved during test case development & execution
1	No. of Requirements	5
2	Avg. No. of Test cases written per Requirement	20
3	Total no. of Test cases written for all requirements	100
4	Total no. of Test cases Executed	65
5	No. of Test cases Passed	30
6	No. of Test cases Failed	26
7	No. of Test cases Blocked	9
8	No. of Test cases un executed	35
9	Total No. of Defects identified	30
10	Critical Defects count	6
11	High Defects Count	10
12	Medium Defects Count	6
13	Low Defects Count	8

Table 5.1: Testing Metrics

5.8 PROJECT, PROGRESS AND PRODUCTIVITY METRICS

Types of Metrics:

- Metrics can be classified into different types based on what they measure and what area they focus on. At a very high level, metrics can be classified as product metrics and process metrics.
- Product metrics can be further classified as,

Project Metrics:

- A typical project starts with requirements gathering and ends with product release. All the phases that fall in between these points need to be planned and tracked.
- In the planning cycle, the scope of the project is finalized. The project scope gets translated to size estimates, which specify the quantum of work to be done.
- This size estimate gets translated to effort estimate for each of the phases and activities by using the available productivity data available. This initial effort is called base lined effort.
- As the project progresses and if the scope of the project changes or if the available productivity numbers are not correct, then the effort estimates are re-evaluated again and this re-evaluated effort estimate is called revised effort.
- The estimates can change based on the frequency of changing requirements and other parameters that impact the effort.

Progress Metrics:

- Any project needs to be tracked from two angles. One, how well the project is doing with respect to effort and schedule.
- The other equally important angle is to find out how well the product is meeting the quality requirements for the release. There is no point in producing a release on time and within the effort estimate but with a lot of defects, causing the product to be unusable.
- One of the main objectives of testing is to find as many defects as possible before any customer finds them. The number of defects that are found in the product is one of the main indicators of quality.
- Defects get detected by the testing team and get fixed by the development team. Defect metrics are further classified in to test defect metrics (which help the testing team in analysis of product quality and testing) and development defect metrics (which help the development team in analysis of development activities).

Productivity Metrics:

- Productivity metrics combine several measurements and parameters with effort spent on the product. They help in finding out the capability of the team as well as for other purposes, such as
  - ✓ Estimating for the new release.
  - ✓ Finding out how well the team is progressing, understanding the reasons for (both positive and negative) variations in results.
  - ✓ Estimating the number of defects that can be found.
  - ✓ Estimating release date and quality.
  - ✓ Estimating the cost involved in the release.

QUESTION BANK

PART – A

Software test automation

1. What is Test Automation?

Automate running of most of the test cases that are repetitive in nature. Developing software to test the software is called test automation.

2. What is Test Case?

A test case is a set of sequential steps to execute a test operating on a set of predefined inputs to produce certain expected outputs.

3. What are the advantages in test automation?

The time saved through automation testing can be used effectively for test engineers to

1. Develop additional test cases to achieve better coverage.
2. Perform some esoteric or specialized tests like ad hoc testing, or
3. Perform some extra manual testing.

Skills needed for automation

4. What are the skills required for automation?

The skills required for automation depends on what generation of automation the company is in or desires to be in the near future.

The automation of testing is classified into three generations.

- ✓ First generation-Record and Playback.
- ✓ Second generation-Data driven.
- ✓ Third generation-Action driven.

Scope of automation -Challenges in automation

5. Give some suggestions for identifying scope for automation?[May 2017]

- ✓ Identifying the types of testing amenable to automation.
- ✓ Automation areas are less prone to changes.
- ✓ Automate test that pertain to standards.
- ✓ Management aspects in automation

6. What kind of tests can lend themselves to automation?

Certain types of tests automatically lend themselves to automation. They are

- Stress, reliability, scalability and performance testing.
- Regression tests.
- Functional tests.

Design and architecture for automation

7. How automation for Extreme Programming Model is done?

Automation for Extreme Programming Model:

- Unit test cases are developed before coding phase starts;
- Code is written for test cases and are written to ensure test cases pass;
- All unit tests must run 100% all the time.
- Everyone owns the product; they often cross boundaries.

8. What are the modules involved in designing the architecture for automation?

Modules:

- ✓ External Modules
- ✓ Scenario and configuration file modules
- ✓ Test cases and test framework modules
- ✓ Tools and results modules
- ✓ Report generator and report / metrics modules

9. What is test framework?

A test framework is a module that combines ‒what to execute' and ‒how they have to be executed‖. It picks up the specific test cases that are automated from TCDB and picks up the scenarios and executes them.

10. What is defect database?

Defect DB on defect database or defect repository contains details of all the defects that are found in various products that are tested in a particular organization.

11. What is Report Generator and Report / Metrics Modules?

The module that takes the necessary inputs and prepares a formatted report is called a report generator. All the reports and metrics that are generated are stored in the reports/metrics module of automation for future use and analysis.

Requirements for a test tool

12. Mention the some requirements for a test tool.

- ✓ Independent Test Cases
- ✓ No Hard Coding In the Test Suite
- ✓ Reuse Of Code For Different Types Of Testing, Test Cases
- ✓ Automatic Set Up and Clean Up
- ✓ Test Case Dependency
- ✓ Insulating Test Cases during Execution

13. What is Test console?

The central machine that allocates test to multiple machines and co-ordinate the execution and result is called test console or test monitor.

1. It should be possible to execute/stop the test suite on any machines/set of machine from the test console.
2. The test results and logs can be collected from the test console.
3. The progress of testing can be found from the test console.

14. What are the items that should have in a test data archive?

Archival of test data must include

1. What configuration variable was used?
2. What scenario was used and
3. What programs were executed and from what path.

Challenges in automation

15. Mention some challenges that are faced during test automation.[May 2017] [Nov 2017]

1. Unrealistic expectations.
2. Expectation that automated tests will find a lot of new defects.
3. Poor testing practice
4. Maintenance of automated tests
5. False sense of security.
6. Technical problems of tools
7. Organizational issues

Test metrics and measurements

16. Why Metrics in Testing?

Testing is the penultimate phase before product release, it is essential to measure the progress of testing and product quality.

Tracking test progress and product quality can give a good idea about the release—whether it will be met on time with known quality.

Measuring and producing metrics to determine the progress of testing is thus very important.

17. Name the two major test metrics. [Nov/Dec 2009]

At a very high level, metrics can be classified as

- ✓ product metrics
- ✓ process metrics

Project, Progress and Productivity metrics

18. How Defect metrics is classified?

Defect metrics are further classified into

- ✓ **Test defect metrics** (which help the testing team in analysis of product quality and testing)
- ✓ **Development defect metrics** (which help the development team in analysis of development activities)

19. What is meant by test case execution productivity? [May/ Jun-2012]

The test cases that can be executed per person day are calculated based on a measure called test case execution productivity. This productivity number is derived from the previous test cycles.

20. Differentiate between project monitoring and project controlling. [Nov / Dec 2012]

Project Monitoring	Project Controlling
It refers to the activities and tasks managers engage into periodically check the status of each project Reports are prepared that compare the actual work done to the work that was planned	It consists of developing and applying a set of corrective actions to get a project on track when monitoring shows a deviation from what was planned

PART – B

1. **Write short notes on software test automation. [4M]** *[Refer Pg.no:157]*
2. **Explain the skills needed for automation. [May 2017-16M]** *[Refer Pg.no:157]*
3. **Explain in detail about scope of automation.[8M]** *[Refer Pg.no:159]*
4. **Explain the design and architecture for automation. [Nov 2017] [16M]** *[Refer Pg.no:160]*
5. **Write short notes on Testing tools.[Nov / Dec 2009 – 8M]** *[Refer Pg.no:162]*

   (Or)

   List the requirements of test tool. Explain with 5 suitable examples. [May/ Jun-2012 – 8M]
6. **Write short notes on challenges faced by tester in automation process. [8M]** *[Refer Pg.no:166]*
7. **Write short notes on Test metrics. [Nov / Dec 2009 – 8M][May 2017-16M]** *[Refer Pg.no:168]*

   (Or)

   Narrate about the metrics or parameters to be considered for evaluating the software quality. [Nov/Dec 2012 -16M] [Nov 2017-16M] *[Refer Pg.no:168]*
8. **Explain the types of Product metrics. [8M]** *[Refer Pg.no:171]*