ROHINI COLLEGE OF ENGINEERING & TECHNOLOGY
Near Anjugramam Junction, Kanyakumari Main Road, Palkulam, Variyoor P.O - 629401
Kanyakumari Dist, Tamilnadu., E-mail : admin@rcet.org.in, Website : www.rcet.org.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NAME OF THE SUBJECT : Software Testing

Subject code : IT6004

Regulation : 2017

UNIT V – TEST AUTOMATION

UNIT V
TEST AUTOMATION

Software test automation – skill 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.

Software Test Automation

Developing software to test the software is called test automation. Automation saves time as software can execute test cases faster than humans do.

The time saved can be used effectively by test engineers to

- Develop additional test cases to achieve better coverage
- Perform some esoteric or specialized tests like adhoc testing
- Perform some extra manual testing

Automation should have scripts that produce test data to maximize coverage of permutations and combinations of inputs and expected output for result comparison. They are called test data generator.

Advantages:

- 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 testing cannot be executed without automation
- Automation means end to end not test execution alone

Activities involved in automation

The following are the activities involved in automation

- Picking up the right product build
- Choosing the right configuration
- Performing installation
- Running the tests
- Generating the right test data
- Analyzing the test results
- Filling the defects in the defect repository

Terms used in automation:

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

Types of test cases:

1. Manual – executed manually
2. Automated – executed using automation

Same test case being used for different types of testing:

Sl.No.	Test cases for testing	Belongs to what type of testing
1	Checks whether log in works	Functionality
2	Repeat log in operation in a loop for 48 hours	Reliability
3	Perform log in from 10000 clients	Load / Stress testing
4	Measure time taken for log in operations in different conditions	Performance
5	Run log in operation from a machine running in Japanese language	Internationalization

Dimensions of test case:

1. What operations have to be tested – product specific feature
2. How the operations have to be tested (Scenario) – frame work specific requirement

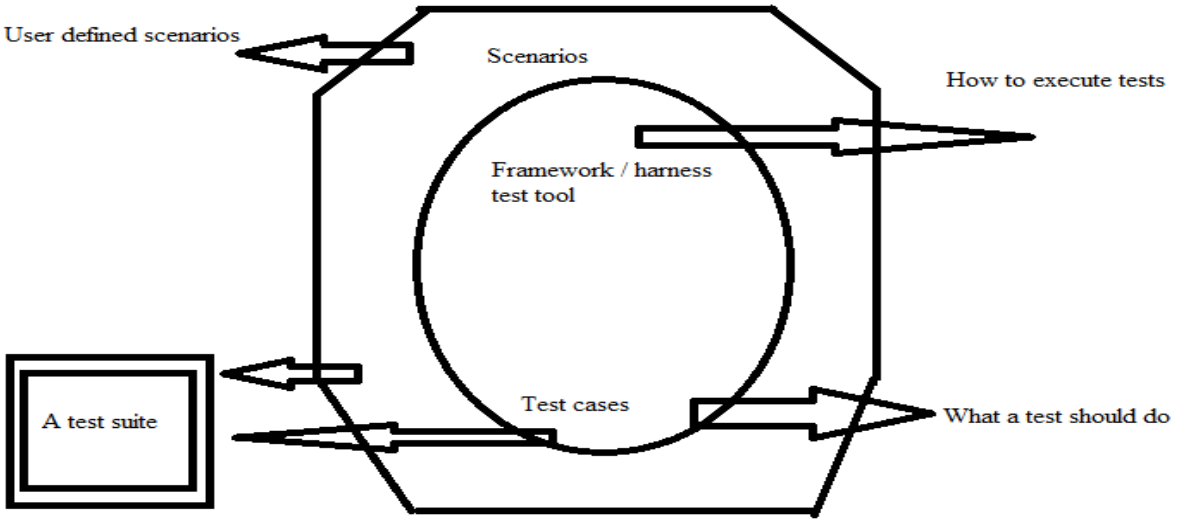
Test suite: It is a set of test cases combined with a set of scenarios

Framework for test automation

An automation framework is a platform developed by integrating various hardware and software resources and various tools and services based on a qualified set of set of assumptions. It enables efficient design and development of automated test scripts and reliable analysis of issues for the system under test.

An automation framework is primarily designed to do the following:

- Enhance efficiency during the design and development of automated test scripts by enabling the reuse of components or code
- Provide a structured development methodology to ensure uniformity of design across multiple test scripts to reduce dependency on individual test-case developers
- Provide reliable issue detection and efficient root-cause analysis with minimum human intervention for the system under test
- Reduce dependence on subject matter experts by automatically selecting the test to execute according to test scenarios and dynamically refining the test scope according to changes in the test strategy or conditions of the system under test
- Improve the utilization of various resources in each cycle to enable maximum returns on effort and also ensure an uninterrupted automated testing process with minimum human intervention



Test Automation Framework

Skills needed for Automation

Generations of Automation:

1. First generation – record and playback
2. Second generation – Data driven
3. Third generation – Action driven

First Generation:

A test engineer records the sequence of actions by keyboard characters or mouse clicks and those recorded scripts are played back in the same order as they were recorded.

Disadvantages:

Script contains hard-coded values which is difficult for error handling.

Second Generation:

This method helps in developing test scripts that generates the set of input conditions and corresponding expected output

Disadvantages:

This approach use much time and efforts

Third Generation:

This approach has no input and expected output condition for running the tests. All actions that appear on the application are automatically tested based on generic set of controls defined for automations.

Advantages:

Scenarios for test execution can be dynamically changed.

Automation – first generation	**Automation – second generation**	**Automation – third generation**	
Skills for test case automation	**Skills for test case automation**	**Skills for test case automation**	**Skills for framework**
Scripting languages	Scripting languages	Scripting languages	Programming languages
Record-playback tools usage	Programming languages	Programming languages	Design and architecture skills for framework creation
	Knowledge of data generation techniques	Design and architecture of the product under test	Generic test requirements for multiple products
	Usage of the product under test	Usage of the framework	

Testing automation is an important concern of businesses, and a growing field in which IT professionals are able to make a name for themselves. The most important skills needed are as follows,

i) **Configuration Management (CM)**: Software Experience Mitigating development drift is essential for implementing reliable automated procedures, so familiarity and skill with some of the more common CM tools is important. These programs include names like Puppet and Chef, and they help to manage large systems and high volume databases.

ii) **Troubleshooting**: Testing in general is the use of troubleshooting methodology on a large scale to find errors that have not been made apparent, and automated testing is the use of programming tools to facilitate this troubleshooting process.

iii) **Development Methodology**: Far beyond the need for technical knowledge, proper testing design requires integrating a large number of systems together, while incorporating a multitude of workers and inputs.

iv) **Coding and Scripting Expertise**: There are a few programming languages which are essential when dealing with automated tests, or simply automation in general. These are generally object oriented with powerful functions already included, or they are designed for easier understanding and to ease the integration of coding from multiple programmers.

v) **Certifications**: While certifications are mostly official recognition of skill with certain hardware or software, the importance placed on them by businesses indicates how important the skills are, even in specific fields, like automation. The top certifications include ones such as PMP, MCSA, VMP, CCNP, various standard certifications, and CompTIA Server+. The PMP (Project Management Professional) certification also relates back to number three on this list.

Scope of Automation

The following are some generic tips for identifying the scope for automation,

Identifying the types of Testing Amenable to Automation

Stress, reliability, scalability and performance testing: Test cases belonging to these testing types are the first candidates for automation

Regression tests: They are repetitive in nature, the test cases are executed multiple times automation of test cases will save time and effort

Functional tests: These kind of tests may require complex setup and require specialized skill, automating the test can enable the less skilled people to run these tests.

Automating areas less prone to change

The basic functionality of the product never changes, hence while automating it has to be considered first

Automate test that pertain to Standards

One of the tests that products may have to undergo is compliance to standards. These tests undergo relatively less change. Automating these tests provides dual advantages. Test suites developed for standards are not only useful for product testing but can also be sold as test tools for the market.

Testing for standards have certain legal and organizational 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 the supporting organization before the release of software and hardware. This is called certification testing and requires perfectly compliant results every time the tests are executed.

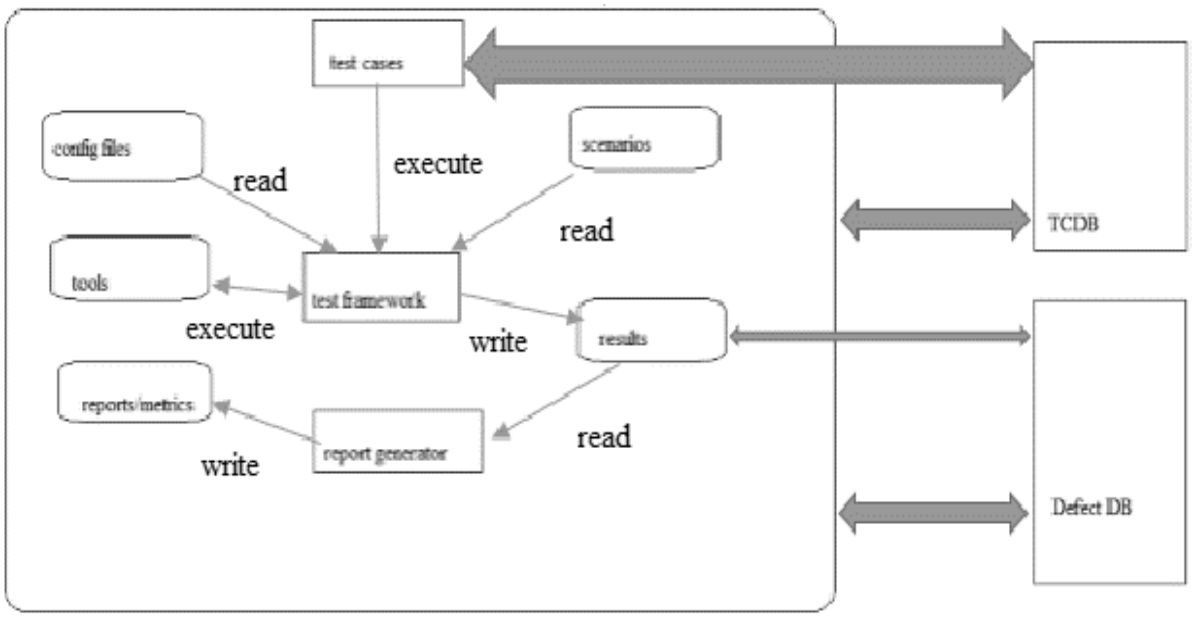
Management aspects in automation

Prior to starting automation, adequate effort has to be spent to obtain management commitment. Automation generally is a phase involving a large amount of effort and is not necessarily a onetime activity. Since it involves significant effort to develop and maintain automated tools, obtaining management commitment is an important activity.

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.

Design and Architecture for Automation

Design and architecture is an important aspect of automation. In the below figure, the thin arrows represent internal interfaces and the direction of flow and thick arrows show the external interfaces



Components of test automation

Architecture for test automation involves two major heads:

1. A test infrastructure that covers a test case database
2. A defect database or defect repository

The different interaction modules are as follows,

External Modules

There are two modules that are external modules to automation

- TCDB
- Defect DB

All the test cases, the steps to execute them, and the history of their execution are stored in TCDB. The test cases in TCDB can be manual or automated. The interface shown by thick arrows represent the interaction between the TCDB and the automation framework only for automated test cases. Manual test cases do not need any interaction between the TCDB and the framework.

Defect DB contains details of all the defects that are found in various products that are tested in a particular organization. It contains defects and all the related information. Test engineers submit the defects for manual test cases. For automated test cases, the framework can automatically submit the defects to the defects DB during execution.

Scenario and Configuration file modules:

Scenarios are information on "how to execute a particular test case"

Configuration file contains a set of variables that are used in automation. The variables could be for the test framework or for other modules in automation such as tools and metrics or for the test suite or for a set of test cases or for a particular test case.

Configuration file is important for running the tests for various input and output conditions and states. The values of variables in this configuration can be changed dynamically to achieve different execution, input, output and state conditions.

Test cases and test framework modules:

Test case means the automated test cases that are taken from TCDB and executed by the framework. 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 execute them. The variables and their defined values are picked up by the test framework and the test cases are executed for those values.

Tools and Results Modules:

When a test framework performs its operations, there are a set of tools that may be required. These tools helps in performing the various test related activities.

When test cases are stored as source code files in TCDB, they need to be extracted and compiled by build tools. In order to run the compiled code, certain runtime tools and utilities may be required

When a test framework executes a set of test cases with a set of scenarios for the different values provided by the configuration file, the results for each of the test case along with scenarios and variable values have to be stored for future analysis and action.

The results that come out of the tests run by the test framework should not overwrite the results from the previous test run. The history of all the previous test run should be recorded and kept as archives.

Report Generator and Report/Metrics Modules:

Once the results of a test run are available, the next step is to prepare the test reports and metrics. Preparing report is a complex and time consuming effort and hence it should be part of the automation design. There should be customized reports like

- Executive report – gives high level status
- Technical report – gives a moderate level of detail
- Debug report – generated for developers to debug the failed test cases and the product.

The periodicity of the report is different such as daily, weekly, monthly and milestone report.

The module that takes the necessary inputs and prepares a formatted report is called a report generator. Once the results are available, the report generator can generate metrics. 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

From the context of software testing a tool is an artifact that assists in the test related activities like planning, requirement analysis, executing the test, defect recording and analysis. The necessary requirements of a good testing tool are as follows

- No hard coding in the test suite : Data should not be embedded in the code subjected to test.
- Test case / suite expandability: The collection of test cases used for testing the software should allow further addition of new test cases.
- Reuse of code for different types of testing, test cases: The code used for testing should adhere to all types of testing and for all test cases in a test suite.
- Automatic setup and cleanup: The tool should start the test and restore the test environment automatically after the test.

- Independent test cases: The test cases employed should not dependent on any other test cases for its functioning and should be independent of one another.
- Selective execution of test cases: The tool should support selective execution of the test cases from the vast collection of test cases.
- Random execution of test cases: The tool should be able to execute the rest cases in a random manner.
- Parallel execution of test cases: The tool should allow the parallel execution of the test cases.
- Reporting scheme: The tool should include a reporting scheme to report the test outcomes
- Independent of languages: The tool should be independent of the programming languages employed.

Example

1) WebLOAD: WebLOAD lets to perform load and stress testing on any internet application using Ajax, Adobe Flex, .NET, Oracle Forms, HTML5 and many more technologies.

2. LoadComplete: LoadComplete enables you to create and execute realistic load tests for websites and web apps.

3. Apache JMeter: It is a Java platform application. It is mainly considered as a performance testing tool and it can also be integrated with the test plan.

4. HP LoadRunner: This is a HP product which can be used as a performance testing tool. Also, it is very much useful in understanding and determining the performance and outcome of the system when there is actual load.

5. Rational Performance Tester: The Rational performance tester is an automated performance testing tool which can be used for a web application or a server based application where there is a process of input and output is involved.

6. NeoLoad: This is a tool used for measuring and analysing the performance of the website. The performance and the end result can be evaluated by using this tool and any further steps can be taken.

7. LoadUI: Load UI is yet another open source and load testing software used for measuring the performance of the web applications. This tool works effectively when it is integrated with the functional testing tool soapUI.

8. WAPT: WAPT refers to the Web Application Performance tool. These are scales or analysing tools for measuring the performance and output of any web application or web related interfaces.

9. Loadster: Loadster is a desktop based advanced HTTP load testing tool. The web browser can be used to record the scripts which is easy to use and record.

10. LoadImpact: LoadImpact is a load testing tool which is mainly used in the cloud-based services. This also helps in website optimization and improvising the working of any web application. This tools generates traffic to the website by simulating users so as to find the stress and maximum load it can work.

11. Testing Anywhere: Test anywhere is an automated testing tool which can be employed for testing the performance of any web sites, web applications or any other objects. Many developers and testers make use if this tool to find out any bottlenecks in their web application and rectify them accordingly.

12. Appvance Next Generation App Performance Testing Platform: Appvance Performance Cloud is a broad-scale platform targeted at enterprise clients which can fully exercise apps from beginning (at the UX) to end. The platform is used to surface deep app and site issues from functional to load and stress and performance to APM.

13. OpenSTA Open STA stands for Open System Testing Architecture: This is a GUI based performance tool used by application developers for load testing and analyzing. This is believed to be a complex tool among the all other performance testing tools.

Selecting a test tool

Tools play a vital role in automating the test. In order to succeed in test automation, proper attention should be given in selecting the correct testing tool. This is because of the following reasons

- Free tools are not well supported and get phased out soon
- Developing in-house tools takes time
- Test tools sold by vendors are expensive
- Test tools require strong training
- Test tools generally do not meet all the requirements for automation
- Not all test tools run on all platforms

Criteria for selecting Test tools:

While there are so many tools available in the market for test automation an appropriate tool for test automation is selected based on the following factors

1. Meeting requirements
2. Technology expectations
3. Training
4. Management aspects

Issues in selecting a testing tool:

Meeting requirements	Technology expectations	Training/Skills	Management aspects
Checking whether the tools meet requirements, involving effort and money	Extending the test tool is difficult	Lack of trainers for test tools	Test tools requires system upgrades
Test tools are not fully compatible with products	Requires instrumented code to be removed for certain tests	Test tools requires people to learn new language/scripts	Migration to other test tools difficult
Test tools are not tested with the same seriousness as products for new requirements	Test tools are not cross platform		Deploying tool requires huge planning and effort
Difficult to isolate problems of product and test suites; change in product causes test suite to be changed			

Steps for tool selection and deployment:

1. Identify the test suite requirements among the generic requirements. Add other requirements
2. Make sure experiences are taken care of
3. Collect the experiences of other organizations which used similar test tools
4. Keep a checklist of questions to be asked to the vendors on cost/effort/support
5. Identify list of tools that meet the above requirements
6. Evaluate and shortlist one/set of tools and train all test developers on the tool
7. Deploy the tool across test teams after training all potential users of the tool

Challenges in Automation

Test automation presents some very unique challenges.

- Management commitment
- Automation takes time and effort and pays off in the long run
- It requires significant initial outlay of money
- It requires a steep learning curve for test engineers
- Management should have patience and persist with automation

Other challenges of automation are as follows,

1) Testing the complete application:

There are millions of test combinations. It's not possible to test each and every combination both in manual as well as in automation testing.

2) Misunderstanding of company processes:

There are some myths in testers that they should only go with company processes even these processes are not applicable for their current testing scenario. This results in incomplete and inappropriate application testing.

3) Relationship with developers:

Big challenge. Requires very skilled tester to handle this relation positively and even by completing the work in testers way. There are simply hundreds of excuses developers or testers can make when they are not agree with some points. For this tester also requires good communication, troubleshooting and analyzing skill.

4) Regression testing:

When project goes on expanding the regression testing work simply becomes uncontrolled. Pressure to handle the current functionality changes, previous working functionality checks and bug tracking.

5) Lack of skilled testers:

while selecting or training testers for their project task in hand. These unskilled fellows may add more chaos than simplifying the testing work. This results into incomplete, insufficient and ad-hoc testing throughout the testing life cycle.

6) Testing always under time constraint:

There is huge list of tasks that need to complete within specified time. This includes writing, executing, automating and reviewing the test cases.

7) Which tests to execute first?

How to take decision which test cases should be executed and with what priority? Which tests are important over others? This requires good experience to work under pressure.

8) Understanding the requirements:

Some times testers are responsible for communicating with customers for understanding the requirements. Testers require good listening and understanding capabilities.

9) Automation testing:

Many sub challenges –

Should automate the testing work?

Till what level automation should be done?

Do you have sufficient and skilled resources for automation?

Is time permissible for automating the test cases?

Decision of automation or manual testing will need to address the pros and cons of each process.

10) Decision to stop the testing:

When to stop testing? Very difficult decision. Requires core judgment of testing processes and importance of each process. Also requires ‘on the fly’ decision ability.

11) One test team under multiple projects:

Challenging to keep track of each task. Communication challenges. Many times results in failure of one or both the projects.

12) Reuse of Test scripts:

Application development methods are changing rapidly, making it difficult to manage the test tools and test scripts. Test script migration or reuse is very essential but difficult task.

13) Testers focusing on finding easy bugs:

If organization is rewarding testers based on number of bugs (very bad approach to judge testers performance) then some testers only concentrate on finding easy bugs those don't require deep understanding and testing. A hard or subtle bug remains unnoticed in such testing approach.

14) To cope with attrition:

Increasing salaries and benefits making many employees leave the company at very short career intervals. Managements are facing hard problems to cope with attrition rate. Challenges – New testers require project training from the beginning, complex projects are difficult to understand, delay in shipping date!

Test metrics and measurements:

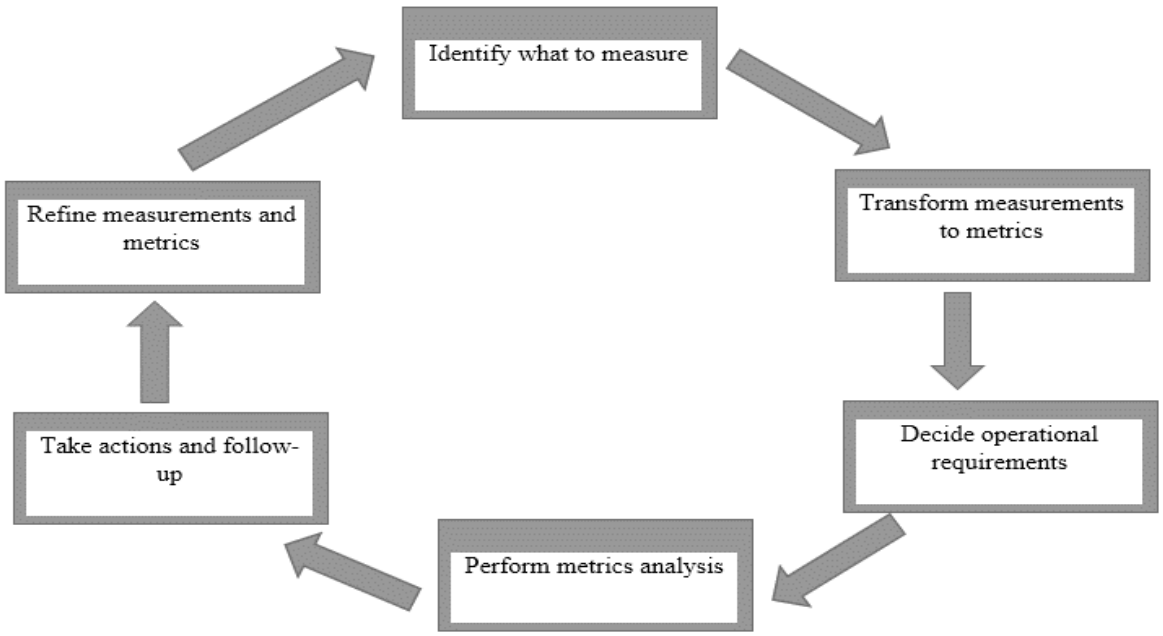
In order to track a project performance and monitor its progress

- The right parameters must be measured
- The right analysis must be done on the data measured
- The result of the analysis must be presented in an appropriate form to the stakeholders to enable them to make the right decisions on improving product or process quality

Effort – the actual time that is spent on a particular activity or a phase

Elapsed days – it is the difference between the start of an activity and the completion of the activity

Schedule – elapsed days for a complete set of activities



Steps in a metrics program

Importance of Metrics in Testing:

Metrics are needed to know test case execution productivity and to estimate test completion date Testing alone can't determine the date at which the product can be released. The number of days to fix all outstanding defects is another crucial data point. The number of days needed for defects fixes needs to take into account the "outstanding defects waiting to be fixed" and a projection of "how many more defects that will be unearthed from testing in future cycles".

The defect trend collected over a period of time gives a rough estimates of the defects that will come through future test cycles. Hence metrics helps in predicting the number of defects that can be found in future test cycles.

Days needed to complete testing = Total test cases yet to be executed / Test case execution productivity

Total days needed or defect fixes = (Outstanding defects yet to fixed + Defects that can be found in future test cycles) / Defect fixing capability

Days needed for release = Max (Days needed for testing, days needed for defect fixes)

Days needed for release = Max (Days needed for testing, (days needed for defect fixes + days needed for regressing outstanding defect fixes))

Metrics in testing helps in identifying

- When to make the release
- What to release
- Whether the product is being released with known quality

Types of Metrics:

Metrics can be classified as (i) product metrics and (ii) process metrics

Product metrics can be further classified as

1. Project metrics – indicates how project is planned and executed
2. Progress metrics – tracks how the different activities of the project are progressing
3. Productivity metrics – helps in planning and estimating of testing activities

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 into size estimated, 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 baselined 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.

Effort and schedule are two factors to be tracked for any phase or activity. In an ideal situation if the effort is tracked closely and met then the schedule can be met. The schedule can also be met by adding more effort to the project. If the release date (schedule) is met by putting more effort then the project planning and execution cannot be considered successful.

If planned effort and actual effort are the same but if the schedule is not met then too the project cannot be considered successful. Hence it is a good idea to track both effort and schedule in project metrics.

Inputs to project metrics:

1. The different activities and the initial baselined effort and schedule for each of the activities
2. The actual effort and time taken for the various activities
3. The revised estimate of effort and schedule

Progress Metrics:

Progress metrics reflects the defects of a product. Defects get detected by the testing team and get fixed by the development team. The defect metrics are further classified into test defect metrics and development defect metrics. How many defects have already been found and how many more defects may get unearthed are two parameters that determine product quality and its assessment.

The progress chart gives the pass rate and fail rate of executed test cases, pending test cases, and test cases that are waiting for defects to be fixed. A scenario represented by a progress chart shows progressing in testing as well as improvement in quality of the product. On the other hand if the chart had shown a trend that as weeks progress, the not run cases are not reducing in number or the blocked cases are increasing in number or pass cases are not increasing, then it would clearly point to quality problems in the product that prevent the product from being ready for release

Defect priority and defect severity

Priority	What it means
1	Fix the defect on highest priority; fix it before the next build
2	Fix the defect on high priority before the next test cycle
3	Fix the defect on moderate priority when time permits, before the release
4	Postpone this defect for next release or live with this defect

Severity	What it means
1	The basic product functionality failing or product crashes
2	Unexpected error condition or a functionality not working
3	A minor functionality is failing or behaves differently than expected
4	Cosmetic issue and no impact on the users

Test Defect Metrics

Test defect metric is a quantitative measure of the degree to which a system, system component, or process possesses defects. The various metrics are as follows

- Defect find rate
  It is the total number of defects found in the units divided by the total number of units tested.
- Defect fix rate
  It is the number of defects fixed divided by the total number of defects found out.
- Outstanding defects rate
  The number of defects present in the unit after subtracting the defects fixed in the unit from the total number of defects in the unit.
- Priority outstanding rate
  The number of high priority defects present in the unit after eliminating the low priority defects present in it divided by the outstanding defects in the unit.

Defect trend

It is a chart that shows the rate at which defects are being activated, the rate at which defects are being terminated or closed, and the drift for the total number of active defects.

- Defect classification trend
  It is a chart that depicts the total number of defects in an unit as extreme, critical, important, minor and cosmetic
- Weighted defects trend
  It is a chart that provides information about how the various defect types contribute to the total number of defects. It helps in quick analysis of defects.

Development Defect Metrics:

It is the quantitative measure of the degree of the defects detected in the development stage. The various metrics are

- Component wise defect distribution
- Defect density and defect removal rate
- Age analysis of outstanding defects
- Introduced and reopened defects trend

Defects per KLOC = (Total defects found in the product) / (Total executable AMD lines of code in KLOC)

Defect removal rate

It is the ratio of defects resolved to total number of defects found in the unit under test.

Defect Classification

Defect classification	What it means
Extreme	• Product crashes or unusable • Needs to be fixed immediately
Critical	• Basic functionality of the product not working • Needs to be fixed before the next test cycle starts
Important	• Extended functionality of the product not working • Does not affect the progress of testing • fix it before the release
Minor	• product behaves differently • no impact on the test team or customers • fix it when time permits
Cosmetic	• minor irritant • need not be fixed for this release

Productivity Metrics:

It combines several measurements and parameters with effort spent on the product. It helps in finding out the capability of the team as well as for other purposes such as

1. Estimating for the new release
2. Estimating the number of defects that can be found
3. Estimating release date and quality
4. Estimating the cost involved in the release

Metrics:

- Defects per 100 Hours of testing
- Test cases executed per 100 Hours of testing
- Test cases developed per 100 Hours of testing
- Defects per 100 Test cases
- Defects per 100 Failed Test cases

Defects per 100 hours of testing = (Total defects found in the product for a period / Total hours spent to get those defects) * 100

Test cases executed per 100 hours of testing = (Total test cases executed for a period / Total hours spent in test execution) * 100

Test cases developed per 100 hours of testing = (Total test cases developed for a period /Total hours spent in test case development) * 100

Defects per 100 test cases = (Total defects found for a period / Total test cases executed for the same period) * 100

Defects per 100 failed test cases = (Total defects found for a period / Total test cases Failed due to those defects) * 100