**What is software testing?**

**Software testing** is a procedure, in which every one of the stages like test planning, test development, test execution, result investigation, tracking of bug and reporting are accomplished effectively.

Validation and verification of Software is ***software testing in software engineering*.**

Testing is a critical segment in the software development life cycle particularly with bugs and imperfections being boundless and phenomenal.

*Software Testing is a continuous procedure and it ought to be conducted all through the development to guarantee that the application functions as per the expectations*.

**Software Testing as a Process**

Software testing is a comprehensive procedure that incorporates connected processes. Three things you check through these procedures: Software completeness in regards to functional necessities; discovering errors that are technical in nature and guaranteeing the software is without bug; evaluating ease of use, execution, security, localization, installation and compatibility.   The software can go overall, in segments, or within a live framework. However, if a product needs to be useful, it needs to go through every one of the tests. After every step of testing, the software goes for amendments. Subsequent to settling these mistakes, the software testing team again runs the following series of tests. This cycle proceeds until the point where the software reaches the desired level of quality.

**Role of Metrics Reporting in Software Testing**

Metrics in software testing can be characterized as standards of measurement. Software metrics are utilized to quantify the nature and quality of the project. In simple words, metrics is a unit utilized for describing an attribute. Thus, the metric is a scale for measurement.

**Some example of Test Metrics**

* Test Execution summary – Pass vs Fail vs Blocked %
* Defects per modul
* Test Coverage percentage
* Defects by Priority
* Requirement to test case traceability

**What is Software Test Measurement?**

Software testing measurement is the quantitative sign of degree, amount, capacity, dimension or size of some attribute of a certain process or product.

***Let’s take a brief look into what software testing is all about with the assistance of 5 Ws (Who, What, When, Where and Why) and 1 H (How).***

5Ws and 1H explained through a project management-based scenario.

Taking a Software testing in software development project, for instance, we consider a situation where development team has discovered that technology they are utilizing isn’t completely perfect with the current frameworks of their client, as was already thought. So the task team can consider utilizing 5Ws 1H (or 2H) to comprehend the problem and its degree of impact.

**Application of the first W – ‘’What**”

What do the users do? What are the objectives they have to achieve? You have to understand their tasks all in all and also the assignments that relate to the software testing or framework that you’re planning.

**Usefulness Vs Performance Vs Load Vs Security Testing (What)**

An imperative assignment for testers is to confirm if the application complies with its determinations and necessities. Utilizing either Manual or Automated software testing philosophies, testers would need to test the applications UI, the APIs, the systems networking to guarantee that the application functions in the manner in which it is required to.

Testers do not just need to guarantee if the application can deal with a lot of information they also need to validate that numerous users are going to access the system. Software performance testing is done to benchmark the application performance in certifiable conditions and to recognize the bottlenecks that can block performance.

*In this regard, the testing team can make the following questions to create an understanding of the crucial problem and extent of the problem.*

* What is the technology we are utilizing for **software testing** development?
* What technologies were considered at first for this development project?
* What was known about the current system(s) of the client?
* What checks and confirmations were done to affirm the similarity of the technology being manufactured and the arrangement compatibility of new technology on existing client system(s)?
* What (assuming any) affirmed or unapproved adjustments/changes have been made after dialogs on technology similarity since the beginning of the task?
* What expertise is accessible to the team to enable them to comprehend the similarity problem?
* What are the acknowledged procedures for overseeing similarity and technological collaborations?
* What rules or [standard operating procedures](https://en.wikipedia.org/wiki/Standard_operating_procedure) (SOPs) are accessible to manage such problems?
* What actions were taken once the problem was detected during testing?

**Application of the second W ‘Why’**

The task team may ask ‘Why’ questions to get a more granular comprehension of the problem and look to clarify triggers or drivers that may have added to the problem. A portion of the questions for software testing that can be asked are:

* Why has it happened that two technologies are currently observed to be inconsistent?
* Why was this problem not detected before or toward the beginning of the software testing or project?
* Why technologies can’t be made good?
* Why were quality affirmation forms not ready to recognize the problem?
* Why were project teams or specialists engaged with the task not ready to identify the problem?

**Application of the third W “When”**

Test-First or Test-Last (When)

**Software testing** can either be composed before the code is finished or after the code is finished. Test-Last help in checking that the code fills in obviously while Test-First recognizes how the code should function. Composing the test initially has its very own favorable circumstances despite the fact that it might appear to be variant. It keeps imperfections and bugs from entering the code and later aides in outlining.

Test-Last help in checking that the code fills in of course while Test-First recognizes how the code should function. Composing the test initially has its own points of interest in spite of the fact that it might appear to be atypical. It keeps imperfections and bugs from entering the code and later aides in outlining.

By utilizing ‘When’ questions, project teams can time stamp the events and comprehend the connections among different events that may have affected the rise of the incompatibility problem.

* When the problem or need for software testing was originally identified?
* When was client system(s) information/architecture assessed?
* When were similarities problems mapped and examined?
* When any similarity tests were done and similarity was discovered satisfactory?
* When was any incongruence affect appraisal done?
* When was the problem taken up with client or client educated?

**Application of the fourth W “Who”**

Who are the users? What are their characteristics? What learning and experience do they convey to their tasks? Are there any other groups of users? Assuming this is the case, what separates them from one another, and which client bunches are generally vital? The subject of whose desires to meet dependably emerge. Would it be those of the Developers or of the Users? Users and engineers have their very own assumptions regarding the application and the codes. The desires from the two sides during software testing ought to be weighted legitimately. The project team can make questions to distinguish the people involved in contributing to the particular problem. A few questions that might be inquired:

* Who is in charge of guaranteeing technological similarity within the project team?
* Who is in charge of data gathering and client system(s) mapping on task team side?
* Who is in charge of plan and change administration endorsements?
* Who is in charge of giving client system(s) data and specialized points of interest to the task team from the client side?
* Who is in charge of regulating the similarity of software testing and detailing?
* Who is in charge of problem identification and acceleration?
* Who is in charge of problem firefighting and contact?
* Who detected the problem or came to know the problem first and who were educated first?
* Who heightened the problem and educated the concerned parties?

**Application of the fifth W “Where”**

By asking ‘Where’ questions, the project team can improve the handle of the source(s) of the problem. A few questions that can be inquired during this stage of software testing are:

* Where is the client system(s) found?
* Where is development teams found?
* Where is quality confirmation for software testing found?
* Where is documentation on framework similarity kept?
* Where are any endorsement/change administration documentation on framework engineering and similarity kept?

**Now Application of the H “How”**

In the metric reporting of 5W and 1H, anything that disappoints the client is a defect, thus understanding the client and client prerequisites is the most critical problem in building up a metric reporting software testing culture. This is a critical thinking administration philosophy that can be connected to a business procedure to recognize and take out the main drivers of defects within software development, eventually enhancing the key software features and sparing expense for the association. In such a manner, the fundamental objective of metrics reporting is that any software development in an association should be monetarily suitable.

5W and 1H metric reporting is a management logic, enables an association to apply a restrained, information driven methodology that consistently acquires development process execution by lessening the inconstancy in every software design. This makes a culture in an association went for figuring out how to fabricate forms that convey the business yield with immaculate quality. Metrics reporting additionally centers on estimating and controlling the variety at each phase of **software testing**and its development.

To comprehend the sequence of differences between related events, the project team can ask how to addresses the 5W’s, for example –

* How did the problem occur?
* How has the succession of occasions prompted the discovery of the problem?
* How are similarities problems dealt with and enter exercises detected in the task?
* How are similarities documentations arranged, shared and put away?
* How is similarity testing performed?
* How the capability of inconsistencies limited and superior comprehension is assembled about mechanical similarity?
* How are jobs and duties inside task portrayed and how is responsibility guaranteed?
* How is detach between project teams limited and legitimate correspondence kept up?

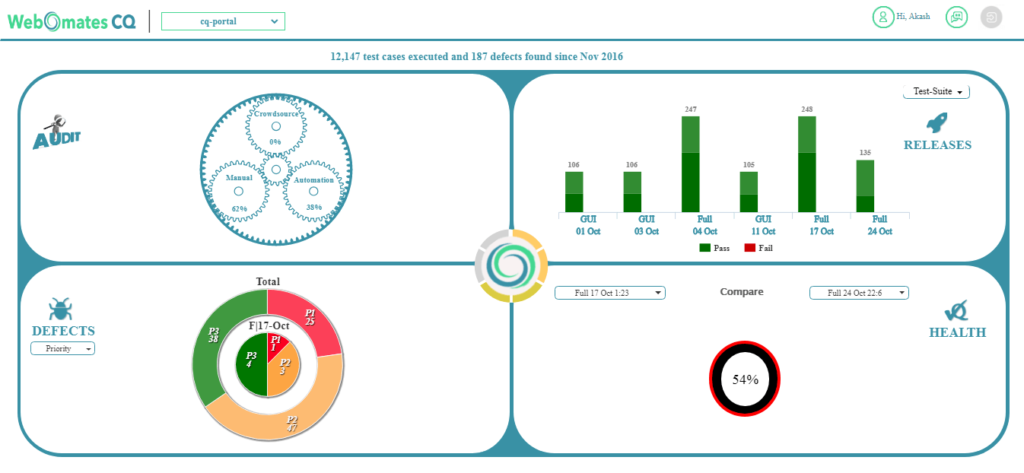
**You May Like to Read**

* [Black – Box Testing Explained](https://www.webomates.com/blog/manual-testing/black-box-testing/)
* [Uncovering the Meaning of Exploratory Testing](https://www.webomates.com/blog/exploratory-testing/uncovering-the-meaning-of-exploratory-testing/)

**How 5W and 1H are Important for software testing?**

Utilizing the 5W-1H strategy for **software testing**, one can determine the different questions and arrive at the metric and its elucidation/significance for the project. The above model has an illustrative rundown of questions for a metric and same can be assisted for any given metric. It is additionally prescribed to map and delineate metric to a certain value and hence bring it closer to the business value. Metric driven project and **defect management** are generally prevalent as long as the correct and proper metrics are chosen.

Specifically, problems solving and quality management methods could help enhance the general adequacy of project delivery. 5Ws and a 1H procedure is basic, legitimate and has what it takes to be utilized with least expense to the project. Utilizing it on normal basis could help project teams to wind up normally more problems in quick time. It is important to note that the list of questions utilized above is for illustration purpose only and ought not to be viewed as complete or generally accepted.



The UI above provides a dashboard view of the [execution of a regression](https://www.webomates.com/blog/software-testing/overlapped-regression-for-agile-teams/) in [Webomates CQ Portal](https://cq-portal.webomates.com/):

1. Number of test cases that passes/Failed ( top right)
2. Number of defects found in the current execution and overall ( bottom left quadrant )
3. Number of test cases Passed/Failed as compared to the the last regression ( bottom right)

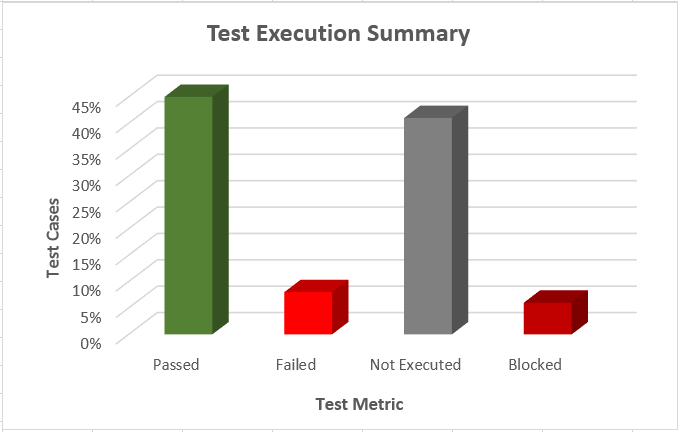
**Why do Test Metrics?**

We cannot improve what we cannot measure and Test Metrics causes us to do precisely the same.

* Take decision for next set or phase of activities
* Evidence of the case or forecast
* Understand the kind of development required
* Enables decision making or process or technology change

For example the chart below shows text execution metrics for a regression run. There are four categories for all the test cases in the regression execution:

1. Passed. Executed correctly and passed all validation point
2. Failed. did not provide correct results for one or more validation points
3. Not executed. The system was unable to perform this test case
4. Blocked. UI blockage or change prevented these test cases from executing



**Advantages of Software Testing through Metrics Reporting**

Since **software testing** through metrics reporting helps in distinguishing and settling bugs before the product gets operational, the possibility of failure can be decreased significantly. **Importance of testing**, therefore, plays a crucial role as it is done keeping the end user in the picture. With prescience for the situations that the end user will potentially confront, tests are done and their exactness and proficiency are noted for any disparities.

Software testing is an important aspect of the product development process. It generally performs a root cause analysis for in turn helps in making it more effective and efficient. Software testing gives a truly necessary confirmation that it will work appropriately and its performance won’t get influenced because of any integration that it may be subjected to.

**There are various benefits of software testing through metrics reporti**

* Maximum coverage
* Low Maintenance Cost
* Easy Reporting
* Minimum Manual Effort
* Scenario Recovery