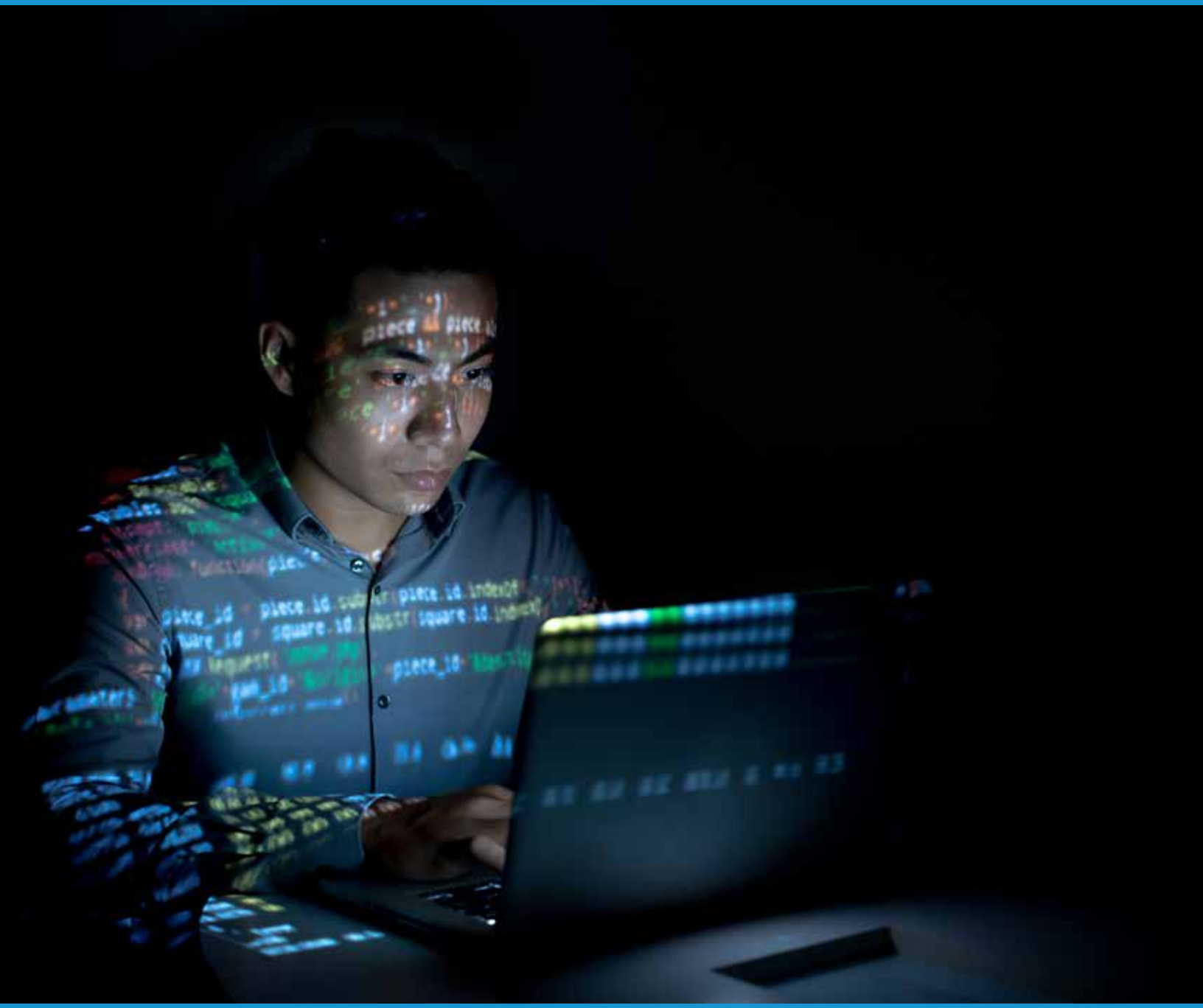
Mphasis
Unleash the Next



Software Testing Metrics

Driving Testing Services Performance

Whitepaper by:
Santosh Subramanian
Testing Services Solution Architect

Contents

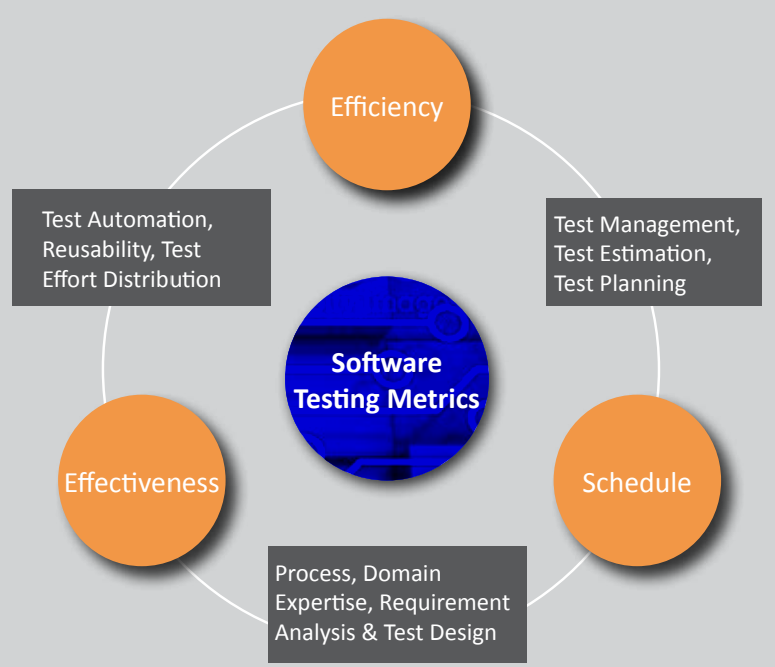
Abstract

The current decade is all about how we collect data and how we leverage to better know our customers, our competitors, the geo-political-cultural environment, etc. This focus on data has had a subliminal impact on IT organizations as well and we find an increasing focus on gathering information, mining it, analyzing it, and leveraging it to make strategic decisions on improving their service delivery performance. Consequently, senior executives pivot to their Testing Services organization to see how best they can leverage the data collected on the quality of delivered software, the testing efficiency & effectiveness, and the quality of the software development process.

When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind
- Lord Kelvin (19 century mathematical physicist and engineer)

This white paper discusses different software testing metrics, key attributes of these metrics, and how any organization can setup a good measurement program to facilitate effective decision-making.

Introduction



Types of Testing Metrics & Influencing Factors

As depicted above, a good set of software metrics can provide detailed insight into the quality of a product/service (i.e. the effectiveness) as well as efficiency of the process delivering the product/service. It also provides historical data to leverage so that to make your delivery process more predictable and your estimates reliable. The discussion of software testing metrics will be incomplete if we just mention the different types of metrics and not discuss the different factors that influence or influenced by the metrics.

For example, one of the biggest challenges IT organizations face is to get to predictable software delivery. Good metrics can not only provide the team with solid heuristics to rely on and leverage it to develop estimates, but also compare its progress against benchmarks and course correct the project at an early stage.

In the following sections we will first discuss the different usage categories for metrics. Following that, will be the key software testing metrics that should be part of any testing organization metrics program. Lastly, we will discuss how we go about building a good measurement program.

It is critically important to mention that this paper does not detail out every software testing metric. While it touches up on a few commonly used and relevant testing metrics (based on author's experience), it intends to provide the reader with a good starting point to start measuring quality of his/her organization's software quality.

Metrics Usage Patterns

Before we delve into different software testing metrics, their calculation rules, and their purpose, it is very important to understand the broad usage patterns that each of these metrics will fall into.

Lead/Lag Indicators

More often than not we try to use the number of defects raised or the number of test cases passed, as a measure of application software quality. It helps when we discuss the current status and progress of application development, but they offer minimal help in preventing defects or accelerating the testing process. This is where lead indicators come into play.

Lead indicators measure activities that have an impact on future performance of a testing services organization. On projects Lead indicators are focused on defect prevention rather than defect detection. Examples of lead indicators are:

- Testing effort spent in requirements and design review;
- Effort spent in peer review of test cases

Point-in-Time/Trending Metrics

Point-in-time metrics enable organizations and project teams take stock of where they are in terms of the plan and make necessary adjustments so as to meet the required objectives of the project/program. Typically, such metrics are used to communicate the status of a project or program.

Trending metrics on the other hand help us understand the patterns of successes or failures, strengths or weaknesses and enable the organization or project team to react accordingly. To measure performance of the testing organization we would be focusing more on trending metrics.

Software Quality Assurance/Quality Control Metrics

Just so as to be clear, Quality Assurance focusses on having the right processes in place so that quality is built into development of the software product. Quality Control focusses on ensuring that the developed product meets the expected quality objectives.

Organizations should typically focus on Quality Assurance metrics so that they can take the steps to bring about process improvements across the SDLC while project teams focus on Quality Control metrics, will enable them to focus efforts to deliver a quality product.

Testing Metrics

When we discuss testing metrics, it is critically important to address the different tiers at which the metrics gets reported and the impact that it has on that tier. Typically these tiers are Organization, Business Unit, and Project.

This tiered approach to metrics development will enable the testing organization to deliver metrics across the IT organization. The organizational level metrics will enable Sr. Executives take up strategic decisions to improve the overall IT organization's software delivery capability. At a business unit level, it enables Sr. Managers to identify risks and/or issues using the combination of status and trend metrics and take corrective action. Each testing team with the help of the detail metrics can not only find out where testing is lacking, but also how they can help the overall team identify defects early or even prevent defect injection.

Organization Level Metrics

These metrics cut across all the testing team in the organization and will offer Sr. Executives a clear perspective on the quality of deliverables output by their IT organization. In addition, trend metrics will enable Sr. Executives to recognize how effective their organization is.

- Software stability trends
  The metrics within this category reflect the number of quality issues that have been experienced within its portfolio of applications. An upward trend over a period of time or a high number over a period of time is indicative of widespread issues within the testing organization and/or within the IT organization. It is an error to treat these metrics as issues to be resolved, rather it is important to understand the root cause(/s) and develop a plan to address them. Some of the metrics that will be included in this category are:
  - No. Of production defects reported
  - System outages/downtime
  - Post release customer feedback
- Operating cost trends
  It is important to review operating cost trends individually from CapEx as a high operating budget might be indicative of non-optimal resource utilization and/or increasing amount of effort being expended on Business As Usual (BAU) testing or heavy expense towards testing tools. Such trends suggest for an assessment of the BAU testing portfolio and identify efficiency opportunities. Some of metrics to be included in this category are:

- Manpower cost trends (operational vs. Capex)
- Manpower cost trends (full-time vs. Temporary staff)
- Licensing cost trends
- Testing infrastructure cost trends
- Ratio of testing budget against overall it budget
- Testing cost per defect

- Reusability metrics
  This metrics is typically not given the attention that it deserves at an organizational level. Primarily because in most cases each individual business unit is at its own level of maturity, there are independently operated, and technology sharing typically is at a low unless it is mandated from the top-down. Metrics included in this category are:
  - Effort savings through asset reuse
  - Time-to-market reduction

Business Unit Level Metrics

These are metrics that go across different testing teams and will enable executives and/or managers to see trends and accordingly take action that will have a mid-to-long term impact on the testing organization.

- Test effectiveness trends
  These trend metrics will let the Program/Business Unit/ Departmental managers know whether the testing organization is effective in defect identification. More importantly, it will also help in determining how late in the testing life cycle were these defects detected. Some of metrics included in this category are:
  - Defect removal efficiency
  - Test coverage
  - Defect injection rate
- Test efficiency trends
  These metrics will help the testing organization determine if it can optimize the cost of testing and the time-to-market. Some of the metrics that will help here are:
  - Mean time to detect
  - Defect density
  - Test case design/execution productivity
- Schedule variance trends
  These metrics will help the testing organization determine if there are challenges with either the estimation for the Testing effort, or the testing planning exercise, or with other factors such as build planning, environment issues, etc.
  - Schedule variance
  - Effort variance
  - Mean time to repair
- Test automation metrics
  It is challenging for departmental managers to see why they are continuing to incur automation costs for application that have been already delivered to production or in case of new development, why the investment in automation is not yielding benefits in terms of reduced cost of testing. A combination of these metrics should enable the testing organization make a strong case of continuous commitment to test automation.
  - Automation testing test coverage
  - Automated test development productivity
  - Automation maintenance effort
  - Automation ROI
- Capability Metrics
  It is important for any Business Unit to understand the true capabilities of the team. It is critically important to give some thought to this metrics, in terms of how well does it align with the objectives and needs of the organization and business unit.
  - Certifications metrics
  - Resource skill index
  - Resource fulfillment index

Project Level Metrics

Typically, individual testers within a team are so down in the trenches that they fail to see the big picture. These metrics when presented along with a snapshot of where the project is against the plan, should provide every tester a clear perspective on the state of project.

Status metrics may hide the true state of the project, as they represent a point-in-time picture and how it track against plan. However, in combination with these metrics, testing team members can truly assess the Qualitative state of the project. Some of the metrics included in this category are:

Test Effectiveness

- Code coverage
- Defect detection efficiency
- Defect acceptance ratio
- Exploratory testing effectiveness

Test Efficiency

- Test case design/execution productivity
- Automated test development productivity
- Defect aging
- Time distribution metrics

Schedule

- Effort variance
- Change request effort ratio

When we talk about project-level metrics, it is also important to align the metrics to the development methodology such as Agile, Waterfall, or Iterative. The above list of metrics spans across these methodologies and should be augmented with metrics specific for the project.

Metrics Explained

The below table provides a brief on each of the metrics referred to above along with an explanation of how each is calculated:

Metrics	Brief Description	Calculation Logic
No. of production defects reported	Count of defects reported in production by severity month over month (MoM)	Not applicable
System outages/ down time	No. of outages by application MoM	Not applicable
	Cost of system outage/ downtime	Cost to business due to downtime + cost of effort spent in restoring application
Customer acceptance index	A measure of feedback from the user community	
Testing infrastructure cost trends	Cost of all environments (dev, test, UAT, PreProd, etc.) By application area QoQ	Not applicable
Ratio of testing budget against overall it budget	A trend of the ratio of overall testing budget against overall it spend	
Testing cost per defect	This essentially indicates how efficient is the testing team operating in terms of identifying defects. This metric in conjunction with the defect detection efficiency (DDE – discussed below) indicates how effective the testing team is.	Overall cost of testing during a given period/# of defects identified by the testing team during that same period
Time-to-market reduction	It is the length of time it takes from a product being conceived until it is released to production	Time in weeks/months saved on a project/ program by using the asset
Defect removal efficiency	% Indicating if testing team was efficient in controlling defect leakage within a testing phase	$(D_{FiP}/(D_{FiP} + D_{FaP})) * 100$ Where D_{FiP}= Defect detected & fixed in a SDLC phase D_{FaP} = SDLC phase Defect detected beyond the SDLC phase

Metrics	Brief Description	Calculation Logic
Test coverage	% of requirements that were tested within a testing phase	(No. of requirements traced back for all test cases executed and passed)/(total no. Of requirements specified) * 100
Defect injection rate	Rate at which new defects are introduced into the system	No. of defects accepted/ (total # of development hours & test execution hours spent)
Mean time to detect	Determines the time that it takes for testing team to identify a defect	No. of defects accepted/ (total # of test execution hours spent)
Defect density	Communicates how many defects were identified in relation to the size of the application	No. of accepted defects/ size of the application
Test case design/ execution productivity	Number of test cases developed per person hour of effort	No. of test case developed/(effort spent on test case development)
Test case design/ execution productivity	Number of test cases executed per person hour of effort	No. of test case executed/ (effort spent on test case execution)
Automated test development productivity	Number of test scripts developed per person hour of effort	No. of test scripts developed/(effort spent on test script development)
Automation maintenance effort	Number of hours expended towards changing existing test scripts over a period	Not applicable
Automation ROI	lindicates if the investment in automation is worth the cost savings that have been achieved due to automation. If ROI = 1 – indicates no savings ROI > 1 – indicates benefit in automating the test suite ROI < 1 – indicates no value in automating the test suite.	(Cost for manual test suite execution – cost for automated test suite execution)/(cost of automation suite development + cost of license + cost of automation suite maintenance + cost of execution)

Metrics	Brief Description	Calculation Logic
Code coverage	Typically created as part of unit testing, is a measure of how extensively is the developed code tested	Not applicable
Defect detection efficiency	% of total # of defects reported	No. of accepted defects found during testing state/ (no. of accepted defects found during testing stage + No. of accepted defects after testing stage) * 100
Exploratory testing effectiveness	Exploratory testing efficiency is the percentage of defects detected in exploratory testing to the total defects detected in the test phase	Defects detected in exploratory testing /total number defects detected in manual, automated and exploratory testing
Time distribution metrics	This can give great insight on how changes in the testing process affects the test projects. If we have to carve out a pie chart from breakdown of total time in testing cycle, we can get good insight into where most of time is spent and correct.	
Defect acceptance ratio	Percentage of defects reported and accepted as valid defects	(No. of accepted defects found during testing state/ (no. of defects reported during testing stage))* 100
Effort variance	% Difference between actual testing effort and estimated testing effort, expressed as a % of actual testing effort	((Actual testing effort – estimated testing effort)/ actual testing effort)*100
Change request effort ratio	Goal is to evaluate requirements stability and accordingly take actions to improve quality of prod/ requirements documentation	Actual effort on changes/ total actual efforts in a testing project
Defect aging	Duration in days or weeks for how long a defect is open i.e. From the time the defect was reported until the defect has been fixed	(Date defect fixed – date defect reported)/s

Setting Up a Metrics Program

It is common industry practice to setup a metrics program based on the GQM (Goals – Question – Metric) paradigm by Victor Basili. The objective of the methodology is to have at a conceptual level the objectives that organization is trying to achieve; then develop a list of questions that can be evaluated against those goals; that will essentially lead to the operational data sets (measures) that can help answer those questions.

The below figure graphically represent an approach that we recommend to setup a metrics program for any testing organization.

Analyze Current State

Any implementation program that does not take into consideration the current state is working outside of context and is expected to have significant challenges in shifting the organizational mindset required to operate a successful metrics program. The current state analysis would include:

- Understanding the current testing process
- Gathering information on the testing tools available and used
- Listing the current set of metrics data captured

Establish Organizational Goals

Understanding the business objectives of establishing a metrics program would enable the team to develop the right set of measures. These goals can be broad and generic or be specific to start with. To develop a Software Testing Metrics program, it is important at this stage to translate these into a set of goals that is influenced/ controlled by the testing organization. Some examples of such goals are:

Business Goal – Improve Customer Satisfaction rating by 15% in 2 years

Testing Goal – Reduce Defects in Production by 95% in 2 years

Testing Goal – Reduce System Downtime to 99.9% in 18 months

Business Goal – Improve time-to-market by 40% in 1 year
Testing Goal – Decrease testing cycle time by 50% for a release in 1 year

Business Goal – Reduce IT Operational Cost by 10% in 20 months

Testing Goal – Reduce testing effort for Business As Usual releases by 20% in 18 months

Identify Measures

This stage consists of three individual steps:

Develop Metrics List

At this point, it is important to take the testing goals that were created in the previous step to develop a list of metrics. Some examples of these translating goals to metrics are:

Testing Goal – Reduce Defects in Production by 95% in 2 years

Metrics – Defect Detection Efficiency, Defect Removal Efficiency, Requirements Coverage

Testing Goal – Decrease testing cycle time by 50% for a release in 1 year

Metrics – Test Case Design/Execution Productivity, Automated Testing Test Coverage

Conduct Gap Analysis

Using the current state analysis determine which of these metrics can be captured with the current testing process and data capture mechanisms in place.

Capture Current State Measurements

For each of the identified metrics and those that exist today, list the actual max and min limits based on the data available.

This three step formal process of identifying measures will help immensely as we move towards developing an implementation plan for establishing the metrics program.

Develop Implementation Plan

It is critically important to understand that putting a metrics program in place without establishing a process to capture the required data is meaningless. It is therefore necessary for the implementation plan to take into account the three dimensions of a testing organization.

Process

The implementation plan should detail the necessary changes that are required, if any, to the testing process that is currently in place so that the measures that are required to develop the metrics program are in place.

Tools

The plan should take into consideration implementation of any tools that need to be implemented for the measures to be captured effectively, automation of data capture, and for the reporting of the metrics data.

People

The implementation plan should detail any training that is required for the testing organization to capture the data that is required and for use of the new tools. It is also required for the implementation to build-in some time for familiarization and implementation of the new tools & processes.

Data Governance Framework

Any metrics reporting is only as effective as the data that goes into the report. Therefore a very critical step in establishment of the metrics program is the data governance framework. Governance framework needs to include:

- Data Standardization – Is there a common taxonomy for the metrics data across the organization
- Data Integrity – Is the data estimated, guesstimated? If yes, how close does it reflect the reality across the organization?
- Data Freshness – Data reflected in the metrics reports should not be stale so that change actions taken in light of the report is relevant to the organization

Implement Metrics Program

With the pre-work that is done up to this point, this process becomes more straightforward. The implementation is then a matter of reviewing the process to capture measurement, the process that goes into analysis of those measurements, and the reporting of the metrics.

The implementation process should include feedback from all the different stakeholders in the organization and constant review of the metrics to ensure that it aligns with the business objectives.

Setup Review & Refine Process

As it is with every other system, as the organization improves its performance through the use of metrics program or otherwise, a period review of the metrics system is required. Therefore, as part of the Metrics Program setup, it is absolutely essential to put a short-term and a long-term review process in place.

The objective of the short-term review process is to ensure that the measurements, the analysis, and the reporting processes are enabling the organization to validate the actions taken towards testing improvements are yielding the desired results.

The long-term review process is to conduct a retrospective on the implementation of the metrics program and see how it needs to refine the process of identify new goals, measures, metrics for a more matured metrics program.

Conclusion

A Metrics Program is essential to evaluating where we are on a project, how we are doing as a testing organization, and how we are serving our business community.

Many a times Sr. Executives start with the process of making changes to the testing organization based on their perception of the issues that they hear from the business or development teams or that they glean from basic
metrics that are available to them. Without a proper metrics program in place, it is not clear if there are underlying issues and there is no way to know whether the change strategy is actually having an impact until much energy and resources have been expended.
At the other end of the spectrum is, the approach to develop a bunch of metrics and start managing the testing organization based on some or all of it. This approach has in many cases led to disgruntled testing organization (e.g. test case execution rate), who believe they are judged unfairly or in other cases led to encouraging bad behavior within the testing team (e.g. tracking total defects reported).

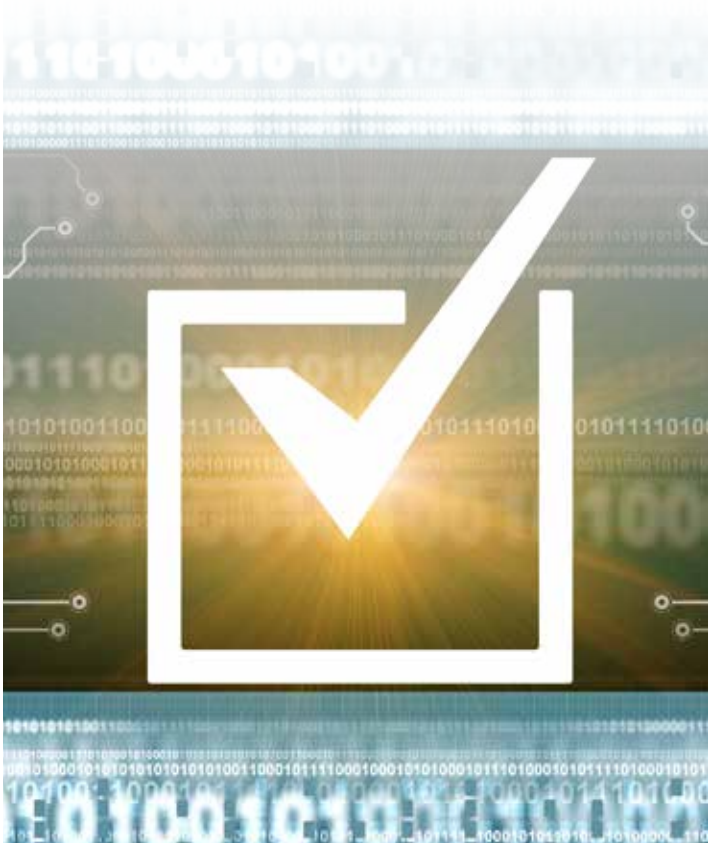
It is therefore critically important for any testing organization that is looking to improve its performance to:

1. Take a concerted effort in bringing together all the stakeholders – business teams, IT management, development team, testing team, and other support teams, to develop a set of goals to measure performance
2. Employ and/or leverage the right tools to automate as much of the data capture and reporting as possible
3. Develop a set of measures that are aligned with the business performance goals set for the organization

Last Line; implementing a robust metrics program is not a one-time event, it is a journey that requires experimentation and learning.

References

- "Economics of Software Quality" – Capers Jones & Olivier Bonsignour
- "Managing the Testing Process" – Rex Black
- "Metrics and Performance Measurement System for the Lean Enterprise" - Professor Deborah Nightingale
- "Better Software Magazine" – various articles
- "Testing Experience" – various articles



Santosh Subramanian

Testing Practice

Santosh has over 22 years of experience in IT consulting with Fortune 500 corporations. He has successfully built and operated multiple testing organizations across different industries including Financial Services, Insurance, and Logistics. Since joining Mphasis in August 2012, Santosh has successfully led opportunities from pre-sales to proposal creation, to contract negotiation, transition, and testing services delivery. Currently, he is working with our strategic accounts providing point testing solutions.

About Mphasis

Mphasis is a global Technology Services and Solutions company specializing in the areas of Digital and Governance, Risk & Compliance. Our solution focus and superior human capital propels our partnership with large enterprise customers in their Digital Transformation journeys and with global financial institutions in the conception and execution of their Governance, Risk and Compliance Strategies. We focus on next generation technologies for differentiated solutions delivering optimized operations for clients.

For more information, contact: marketinginfo@mphasis.com

USA
460 Park Avenue South
Suite #1101
New York, NY 10016, USA
Tel.: +1 212 686 6655

UK
88 Wood Street
London EC2V 7RS, UK
Tel.: +44 20 8528 1000

INDIA
Bagmane World Technology Center
Marathahalli Ring Road
DoddanakundhiVillage, Mahadevapura
Bangalore 560 048, India
Tel.: +91 80 3352 5000

www.mphasis.com

VAS 10/01/17 US LETTER BASIL 4197