Infosys®

White Paper

Realizing Efficiency & Effectiveness in Software Testing through a Comprehensive Metrics Model

Abstract

The ever-increasing complexity of today's software products, combined with greater competitive pressures and skyrocketing costs of software breakdown have pushed the need for testing to new heights. While the pressures to deliver high-quality software products continue to grow, shrinking development and deployment schedules, geographically dispersed organizations, limited resources, and soaring turnover rates for skilled engineers make delivering quality products the greatest challenge. Faced with the reality of having to do more with less, manage multiple projects and distributed project teams, many organizations are facing innumerable challenges in managing the quality programs for their products.

In addition, there is a continuing urge for enhancing the operational capabilities of the teams so as to be able to produce more and more with a reducing investment bucket.

Test metrics can serve as important indicator of the efficiency and effectiveness of a software testing process. Analysis of defined metrics helps identify areas for improvement and devise subsequent actions. Test Metrics Collection, Analysis and Improvement is not just a single phase in the software testing life cycle; but, acts as an umbrella of continuous improvement for whole of the testing process.

This paper illustrates a comprehensive measurement model, which can be adopted to inculcate a culture of continuous improvement in the overall software testing lifecycle.

Introduction

One of the key characteristics of a maturing discipline is a special focus on science on top of art. An enhanced focus on measurement in the field of Software Quality Engineering over the recent years has been a confirmation of above fact. The field of Software Testing in particular, being known as an Art for years has got the flavor of a scientific discipline with introduction of various testing types as well as for measuring the various aspects of these testing types.

Software metrics serve valuable information to enable quantitative managerial decision making over the entire software lifecycle. Reliable support for decision-making implies effective risk assessment and reduction. A winning software metrics model should be encompassing relevant and relatively simpler metrics to support management decisions. Such a metrics model should cover different aspects of software development and testing enabling managers make multiple sorts of forecasts, judgments' and trade-offs during the software life-cycle.

This paper illustrates the importance of measurement and metrics in the whole software testing process and provides an insight in to various process/product efficiency and effectiveness metrics.

Need for Metrics in the Software Testing Process

If you are able to measure something, you can comprehend it in a better way and know more about it. If you can comprehend more, you are in a better position to attempt to improve upon it.

Metrics help gauging the progress, quality and health of a software testing effort. Metrics can also be leveraged to evaluate past performance, current status and envisage future trends. Effective metrics are simple, objective, measureable, meaningful and have easily accessible underlying data.

Over time, software products and hence software projects have become more and more complex due to comprehensive product functionalities, advanced architectures and gigantic supported configurations. Also, with increasing business and margin pressures, the projects are required to be accomplished in lesser amount of time with fewer people. This increasing complexity over time has a tendency to impact the test coverage and ultimately affect the product quality. Other factors involved over time are the overall cost of the product and the time to deliver the software. Metrics can provide a quick insight into the status of software testing efforts, hence resulting in better control through smart decision making.

The Test Metrics – Various Dimensions

Software metrics can be broadly classified into three categories: product, process and project metrics.

Product metrics portray the characteristics of the product such as its size, complexity, design aspects, performance and quality. Process metrics can be used to measure and enhance processes of software development, maintenance and testing. Examples include defect removal effectiveness during development process, testing defect arrival pattern and defect-fix response time for maintenance process. Project metrics depict various project characteristics and its execution. Examples include number of software engineers, staffing pattern over project life cycle, cost, time and productivity.

Software testing metrics are a subset of overall software metrics umbrella that focus primarily on the quality facets of process and the product. Software testing metrics can be further classified into Process Quality Metrics and Product Quality Metrics. The core of software test engineering is to examine the relationships among various in-process metrics, project characteristics and product quality. Further, based on the findings, it aims to instill improvements in both the testing process and product quality.

In this paper, we discuss several metrics falling under each of the two sets of software quality metrics: Process quality and Product quality metrics. For each metric, we discuss its purpose, interpretation and usage. Additionally, key in-process metrics, effective for managing software testing and the in-process quality health of the project have been presented along with few useful metrics applicable for automated testing process.

Process Quality Metrics

A process can lead to a quality outcome within least time and cost only if the process itself is effective and efficient. That is one of the primary reasons why quality organizations around the world viz. SEI, have focused their efforts on gauging and enhancing process performance and maturity. Metrics play an important role, when it comes to measuring a process from various dimensions, assessing it and target improvements. Software testing process, being the core of overall software engineering life cycle has been a prime focus area with respect to enhancing efficiency and effectiveness.

Metrics for Process Efficiency

Organizations today are striving to deliver more and more with less. Achieving this objective is highly dependent upon, how efficient the various processes running in these organizations are. Hence, process efficiency depicts the ability of a process to produce desired outcome with optimum (least) no. of resources. Software testing, being the key process responsible for yielding a high-quality deliverable has been the prime focus for efficiency improvement efforts. Metrics serve as a useful guide for making sure that these improvement efforts are applied in the right direction.

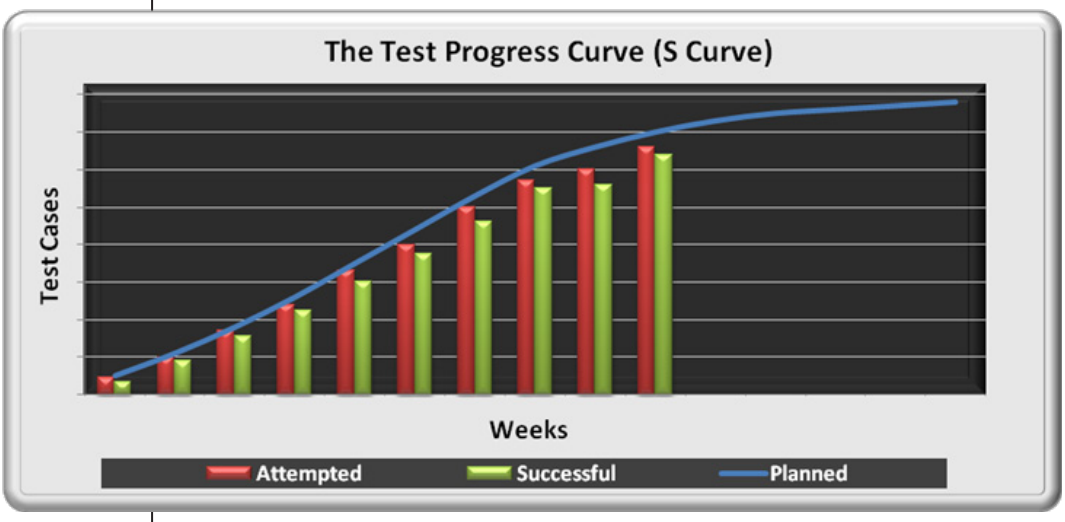
Some of the key metrics, which enable one to measure various efficiency aspects of Software testing process, are discussed below -

Test Progress Curve – S Curve

Tracking testing progress is perhaps one of the most important tracking tasks, while managing software testing. The metric utilized for this purpose is the test progress-S curve over time.

This metric depicts planned testing progress over time in terms of number of test cases completed successfully vis-à-vis number of test cases attempted

The objective of this metric is to track and compare testing progress with the plan, hence enabling teams take early actions upon any indications that the testing activity is lagging behind. When under pressurizing schedules, testing activity is the one, which generally gets impacted the most. With a formal testing progress metric deployed, it is much harder for the team to disregard the problem. Adjoining figure depicts an illustration of the test progress curve drawn during mid of the test program for a major release.

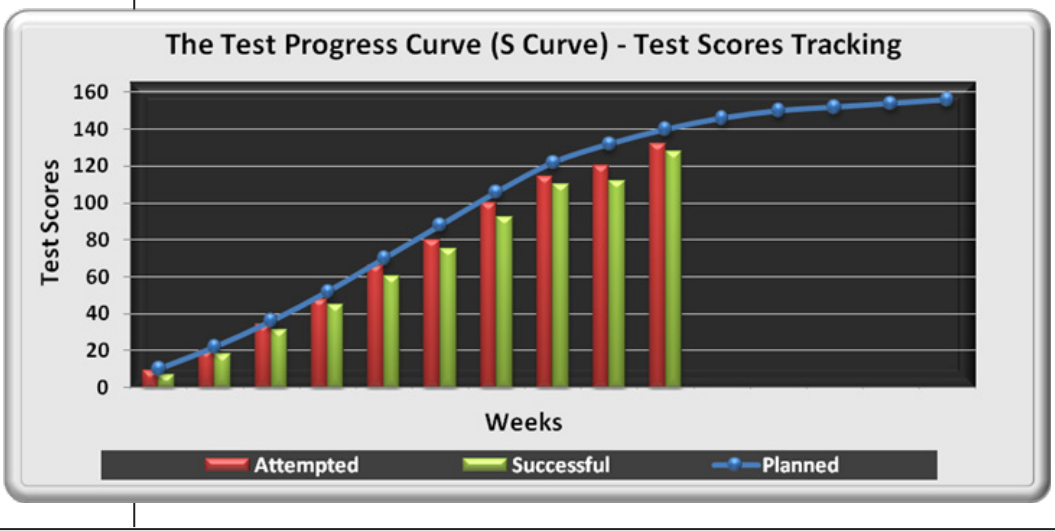


Test Scores Progress Curve – S Curve

Some software components or test cases may be more important than others; hence, it is common to assign priorities/scores to the components or test cases. When we use these priorities/test scores, it can provide a more precise way of tracking the test progress in a normalized way.

To have an S-curve, based on the test scores, the team needs to express the amount of testing done every week in terms of test scores and track weekly progress in terms of test scores.

Adjoining figure illustrates test scores level tracking for a sample software test program.



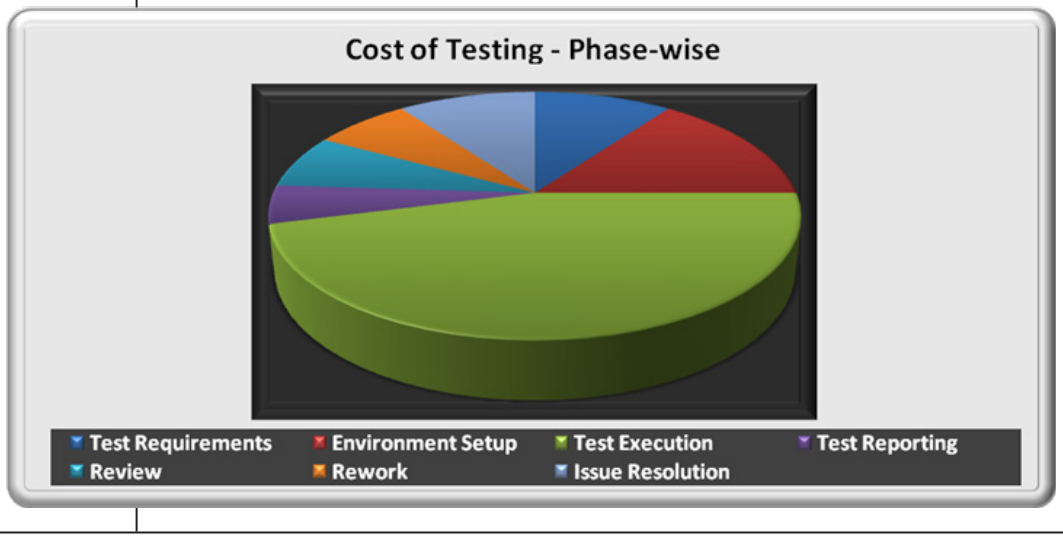
Cost of Testing - Phase-wise

Efficient utilization of the effort allocated for software testing is crucial towards ensuring maximum test coverage resulting in high-quality end-product. Hence, monitoring the effort going towards various test activities or phases and taking required improvement actions can result in an optimum execution of these activities. The metric utilized for this purpose is the phase-wise cost of testing.

This metric depicts a phase-wise distribution of the overall effort going towards software testing.

The objective of this metric is to help identify intensive effort areas and target improvement actions in order to bring execution optimization. For example, if based on the analysis of this metric, it is found that the "Environment Setup" activity is taking too much of a time, some improvement actions can be devised and deployed in order to optimize this activity, which may involve automating the steps required for environment setup or having team members work in parallel on multiple test setups to save effort.

Adjoining figure depicts an illustration of the effort distribution for various commonly used testing phases in a test program. Please note that these phases may vary based on the nature of a project.



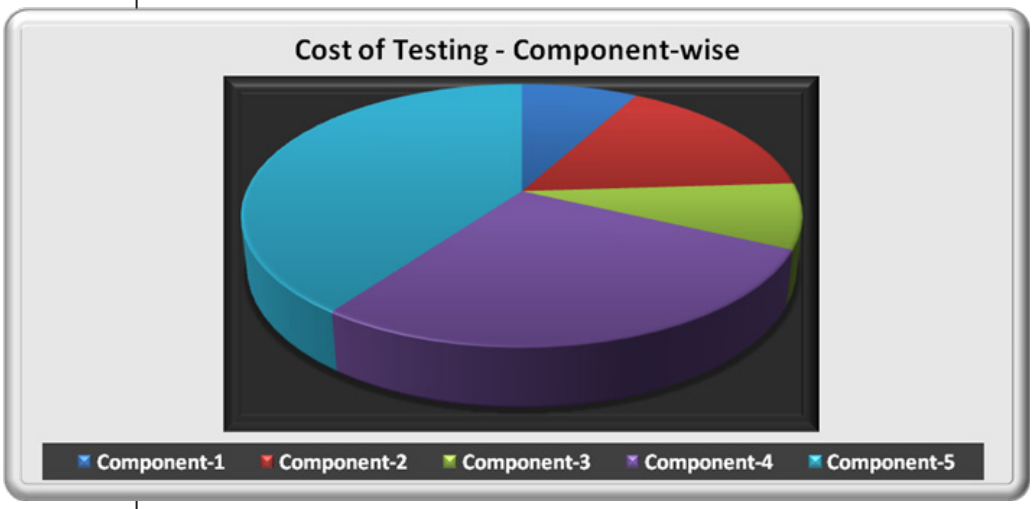
Cost of Testing - Component-wise

Analogous to monitoring phase-wise testing effort, tracking the effort going towards testing of various product components can help identify features/components taking more testing effort and devise optimization actions. The metric utilized for this purpose is the feature/component-wise cost of testing.

This metric depicts a component-wise distribution of the overall effort going towards software testing.

The objective of this metric is to help identify product components having intensive testing effort areas and identify improvement actions. For example, adoption of testing tools or automation can help reduce testing effort for any particular component.

Adjoining figure depicts an illustration of the effort distribution for various product components.

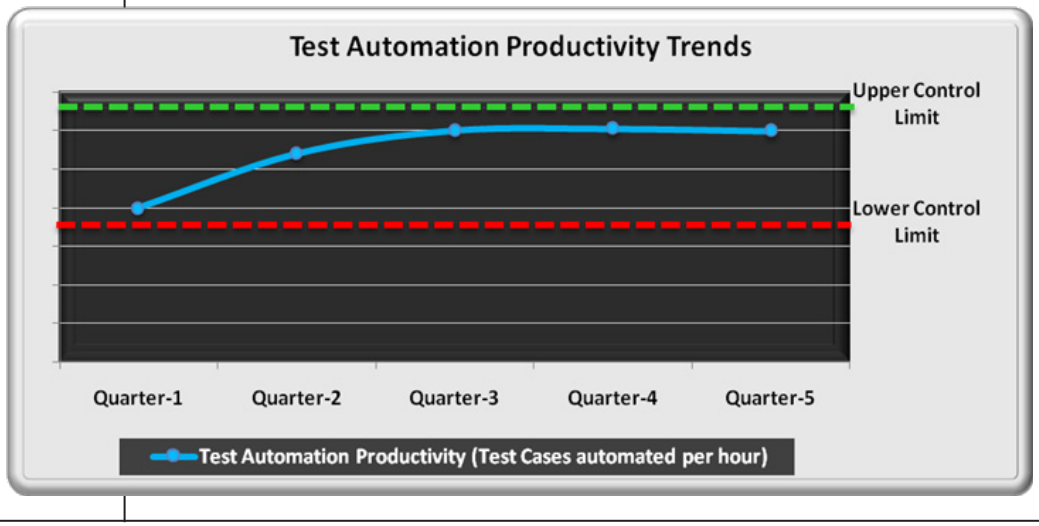


Test Execution Productivity Trends

A helpful measure of test team's efficiency is the productivity at which the team carries out test execution. The test execution productivity may be defined as the average no. of test cases executed by the team per unit of time, where the unit may be hour, day or week, but is generally taken as an hour or a day.

The metric depicts a trend of test execution productivity delivered by the team over multiple test cycles, weeks or any other time interval.

The objective of this metric is to help identify any problematic areas impacting team's productivity and take remedial actions, where feasible. Based on past experience or set baselines, upper and lower control limits can be set for the productivity and special focus be given to situations, wherein the productivity falls below lower control limits or goes above upper control limits. Adjoining figure depicts an illustration of this metric.

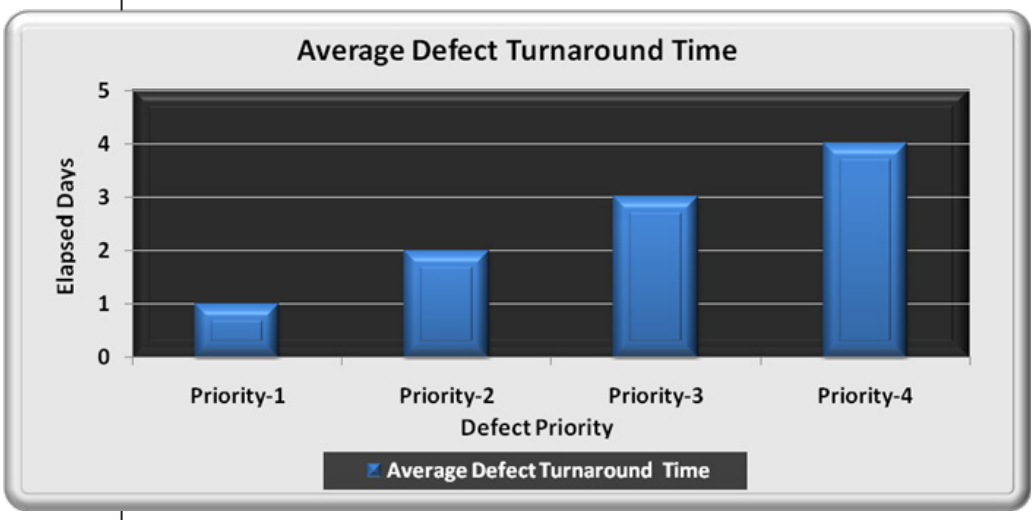


Average Defect Turnaround Time

One of the key activities carried out by test teams is verification of defects, which circle back to them after getting fixed by development team. This activity generally needs close attention by test teams especially for defects having higher priorities. An indicator of test team's operational efficiency is the average time taken by the team for verification of these defects.

The metric depicts the average verification or turnaround time taken by the team to verify defects of various priorities.

Based on past experience or set baselines, upper and lower control limits can be set for the average verification times and situations having instances falling outside control limits need to be analyzed. Adjoining figure depicts an illustration of this metric.

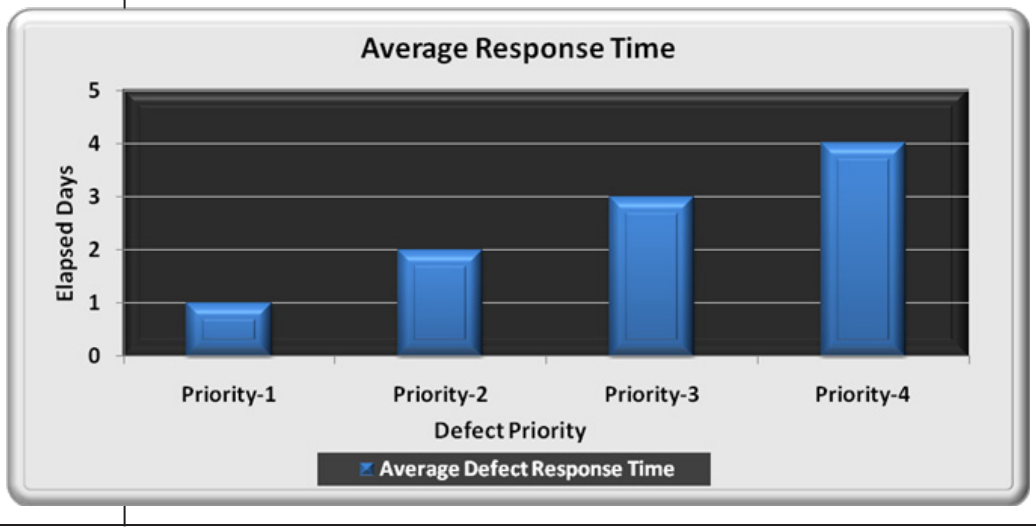


Average Defect Response Time

Some of the defects logged by the test team might come back to them requesting for additional information in case any more information is required pertaining to the defect. Responding with the required additional information as early as possible is imperative to quick closure of these defects. An indicator of test team's operational efficiency is the average time taken by the team for responding to the defects falling under "Require More Information" category.

The metric depicts the average response time for the test team to get back with additional information for defects of various priorities.

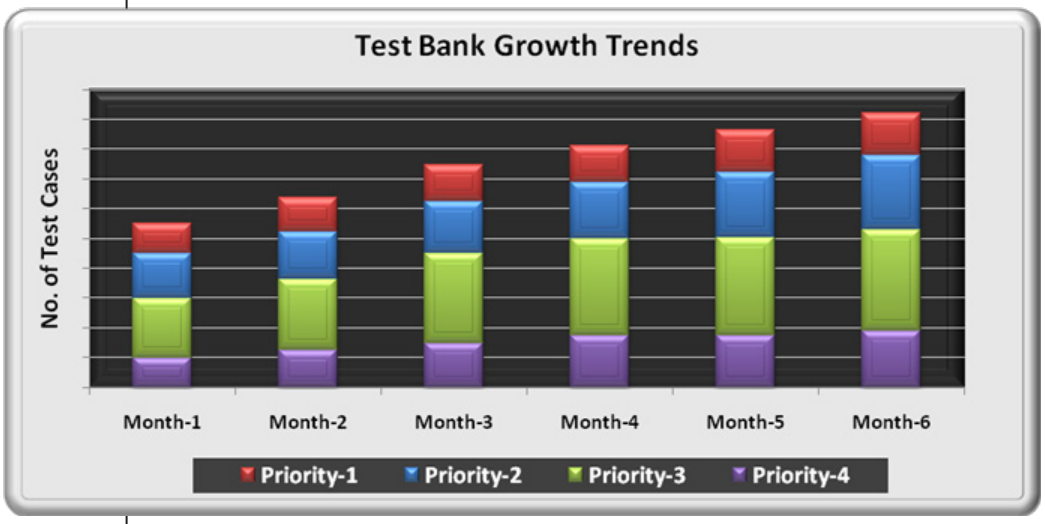
Based on past experience or set baselines, upper and lower control limits can be set for the average response times and situations having instances falling outside these control limits need to be analyzed. Adjoining figure depicts an illustration of this metric.



Test Bank Growth Trends

The test banks which act as the basis of test activities performed by the test teams need to be enhanced on a continual basis to reflect enhanced product functionality as well with additionally devised test scenarios. The rate at which these test banks grow serves as an indicator of test team's performance.

While assessing overall test bank growth trends, it makes sense to also observe the trend in the growth of test cases of various priority levels. For instance, an increase in no. of highest priority (Priority-1) test cases may be regarded better as compared to a similar increase in lower priority test cases. Adjoining figure depicts an illustration of this metric.



Metrics for Process Effectiveness

Process effectiveness refers to the discipline and control deployed over a process, which in turn enables the process to yield a high-quality output. Measurement enables one to assess the health of currently deployed process discipline & control, and bring-in required improvement actions.

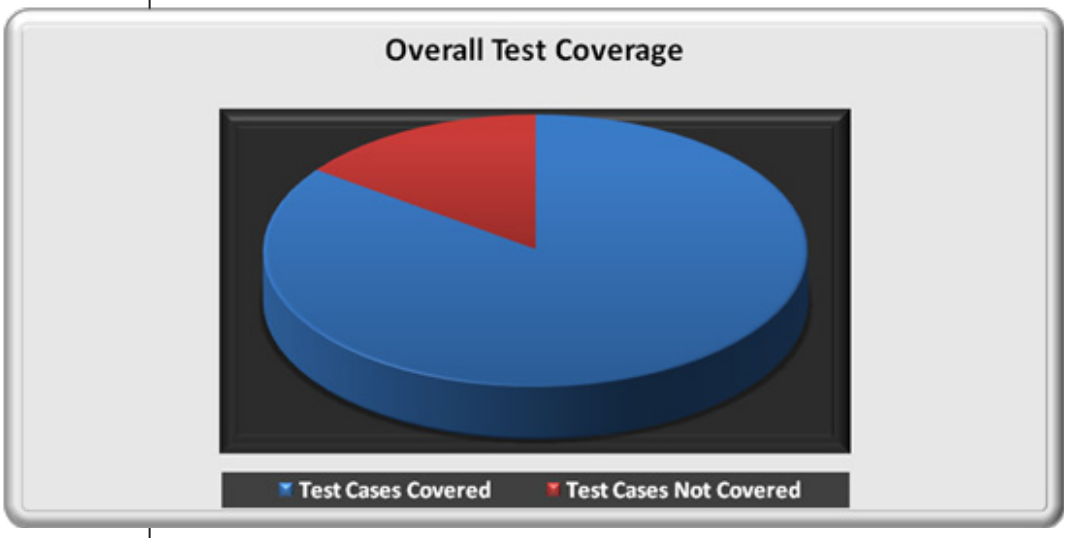
Some of the key metrics, which enable one to measure various aspects of Software testing process effectiveness, are discussed below -

Test Coverage (Overall)

With ever increasing test banks, huge configuration support matrices and too frequently changing requirements, it becomes intuitively important for the test manager to smartly strategize in order to manage the resources at hand so as to ensure an optimum coverage of the test banks and supported configurations. Hence, test coverage serves as an important measure of the effectiveness of the testing process.

The metric measures the amount (percentage or number) of components/test cases/configurations actually tested successfully vis-à-vis the total no. of components/test cases/configurations for the product.

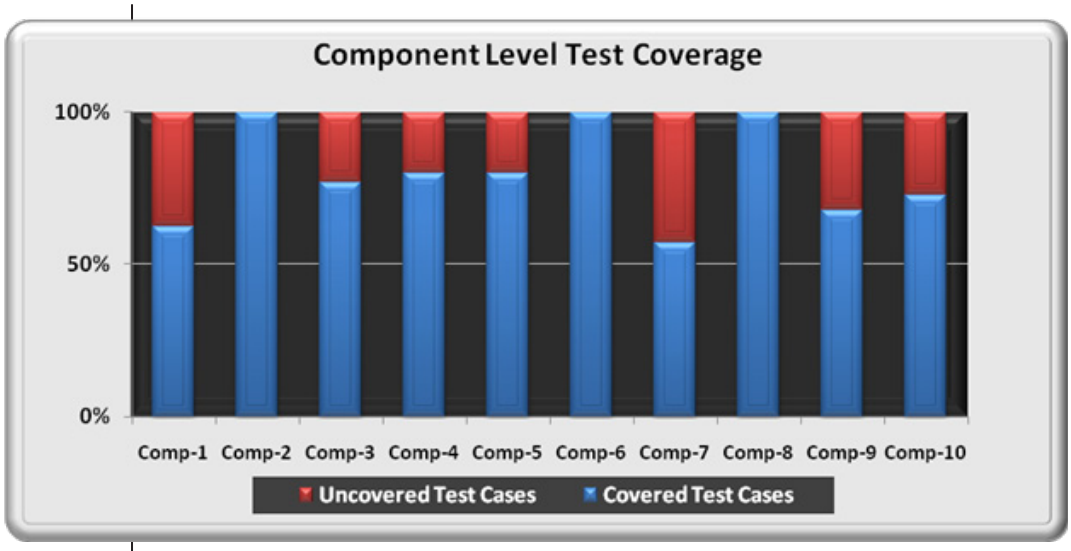
This metric provides the management team with a confidence on the product quality and acts as an important indicator for the effectiveness of the testing process. Adjoining figure illustrates the metric at a glance.



Test Coverage (Component-wise)

In addition to tracking the test coverage at overall product level, test managers do track the coverage at per product component level in order to have a confidence on the component-level quality. This metric along with overall test coverage serves as an important indicator of the effectiveness of testing process.

The metric measures the number (percentage) of test cases actually tested successfully vis-à-vis the total no. of test cases for each product component. Adjoining figure illustrates the metric at a glance.



Defect Removal Efficiency (DRE)

Another important metric for gauging the effectiveness of testing process and for tracking quality through the testing life cycle is Defect Removal Efficiency (DRE). DRE is used to determine the effectiveness of testing team's defect removal efforts. It serves as an indirect indicator of the quality of the product.

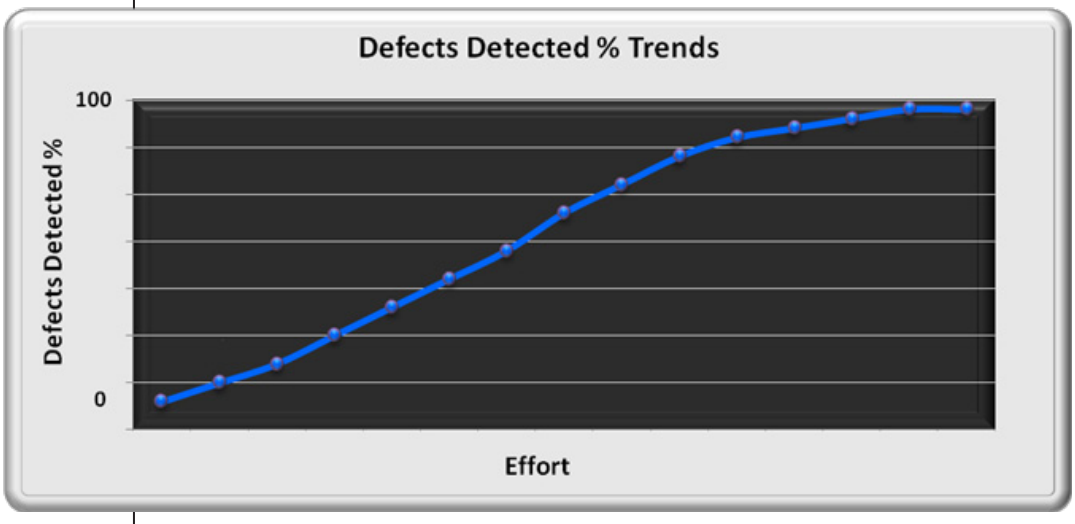
The metric measures the percentage of defects caught by the testing team vis-à-vis the total no. of defects reported against the product, both during the testing life cycle and post delivery to end-customer. Higher the percentage of DRE, the more positive impact on product quality. This is because it represents the ability of testing team to catch product defects prior to product delivery.

DRE (%) = [TD / (TD + AD)] * 100,

Where TD = # of defects found during testing, AD = # of defects found after delivery

The highest possible value of DRE is "1" or "100%", though it is quite unlikely to achieve. DRE can also be measured during various phases of software development life cycle. In such a case, a low DRE reported during analysis and design phases for instance, may signify that the technical review process needs to be strengthened.

Calculating DRE post release of a product and analysing the root causes behind various defects reported by end customers may help strengthen the testing process in order to achieve higher DRE levels for future releases.



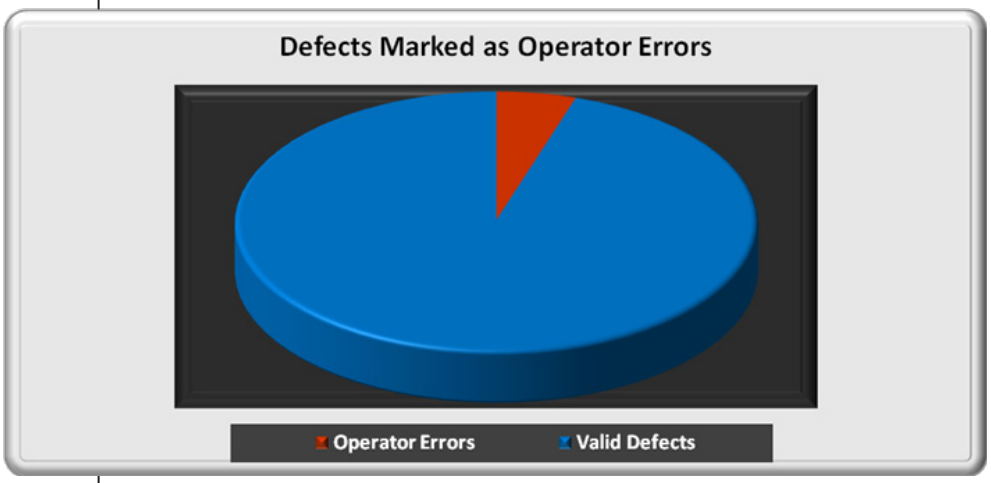
Defects Marked as Operator Errors

An indicator of testing team's work quality is the extent of defects logged by them turning out to be invalid defects. There may be some defects out of the several defects logged by the testing team, which are designated as "Operator Errors" due to several reasons viz. incomplete understanding of product requirements, invalid test environment configuration etc.

This metric to measure testing team's work effectiveness is percentage of defects marked as operator errors.

If you see a good percentage of defects getting marked as "Operator Errors", it indicates that the understanding of the test engineer on the functionality may be low or there have been gaps in the requirements document resulting in an improvement action for the development team.

Based on the root cause analysis carried out for various "Operator Errors", improvement actions can be initiated to improve the testing process resulting in reduced "Operator Errors" for future testing efforts. Adjoining figure depicts an illustration of this metric.

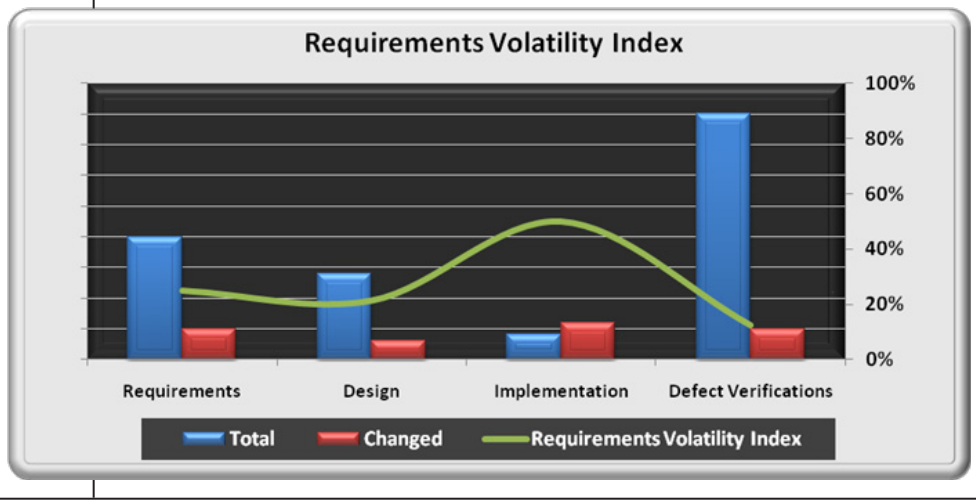


Requirements Volatility Index

During the testing life cycle, there generally are instances when the test team needs to respond dynamically to changing requirements, shifting designs, multiple new product builds handed over for test execution and fixed defects failing verifications. Test managers need to revisit their strategies in order to respond effectively to these changes. The amount of these changes acts as an environment complexity factor in which test teams operate and serves as a performance indicator for the test teams amidst these complexities.

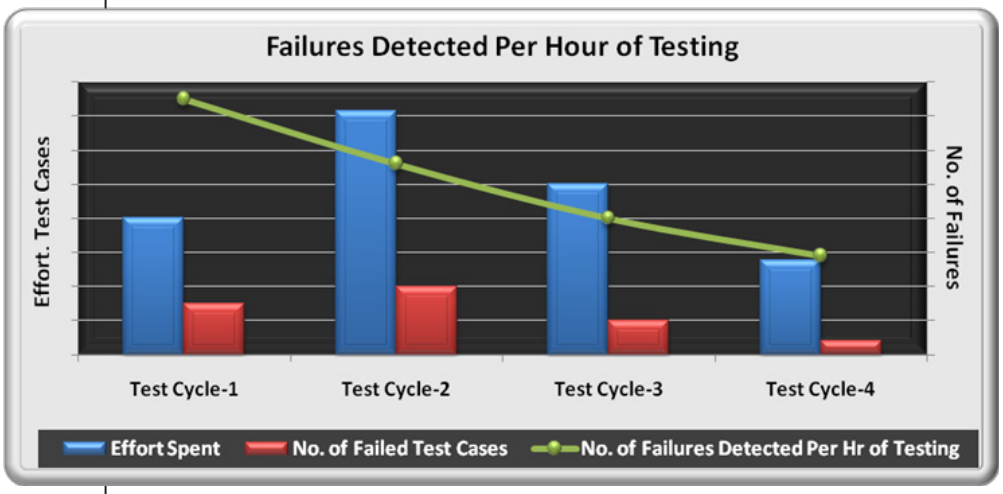
The metric to track all these changes is the "Requirements Volatility Index", which can be tracked and compared for various types of changes.

While this metric serves as an indicator of the test team's performance on one hand, it also equips test team to raise their concerns on required improvements in the development process. Adjoining figure depicts an illustration of this metric.



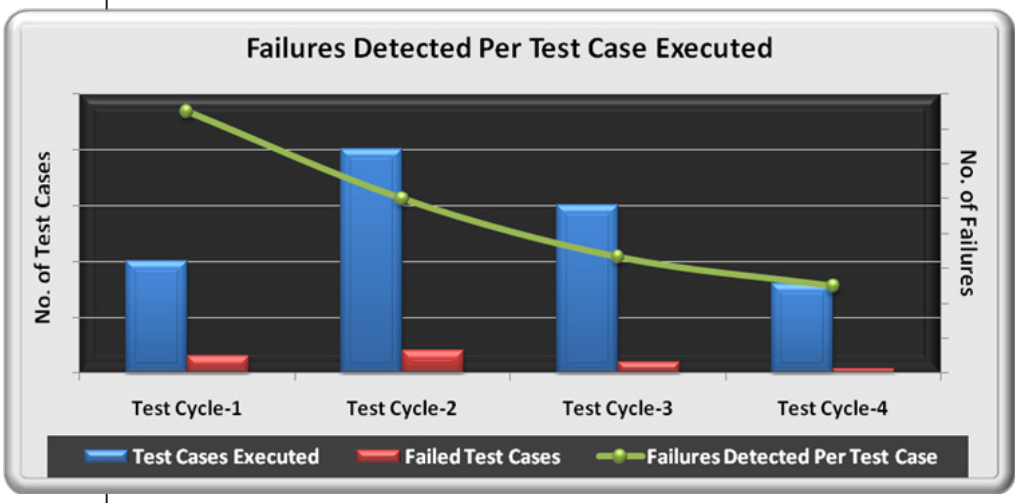
Failed Test Cases/ Hr

The effectiveness of the effort going in to test activities can be measured based on the no. of failures detected within this effort. A useful metric to gauge this effectiveness is the no. of failures detected per unit of time. As we move from initial stages of the software testing life cycle to the final stages, there should be a decreasing trend in the value of this metric. Adjoining figure depicts an illustration of this metric.



Failed Test Cases/ Total Test Cases Executed

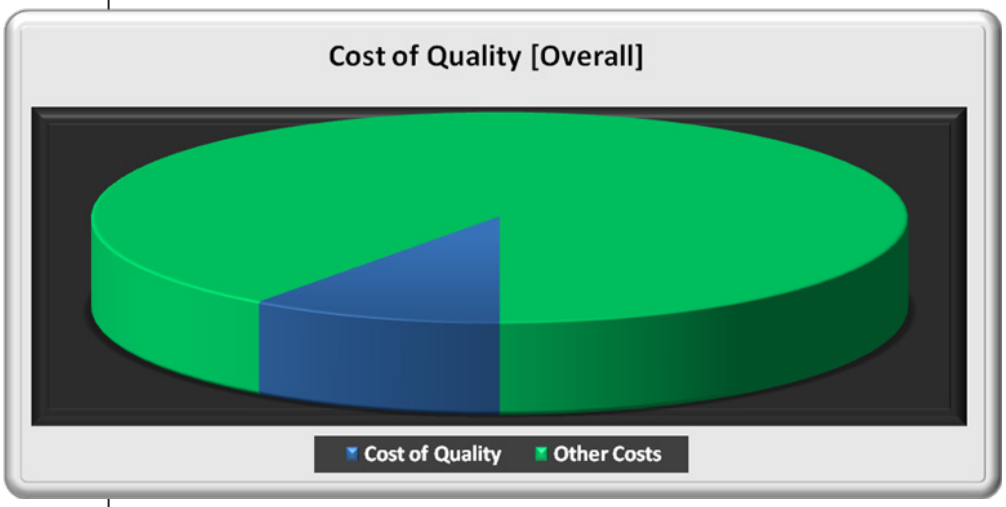
Another metric in this direction is the average no. of failures detected per test case executed, which indicates the effectiveness of the test bank and the specifc set of test cases selected for execution. Similar to the earlier metric, there should be a decreasing trend in the value of this metric as we move from initial stages of the software testing life cycle to the final stages,. Adjoining figure depicts an illustration of this metric.

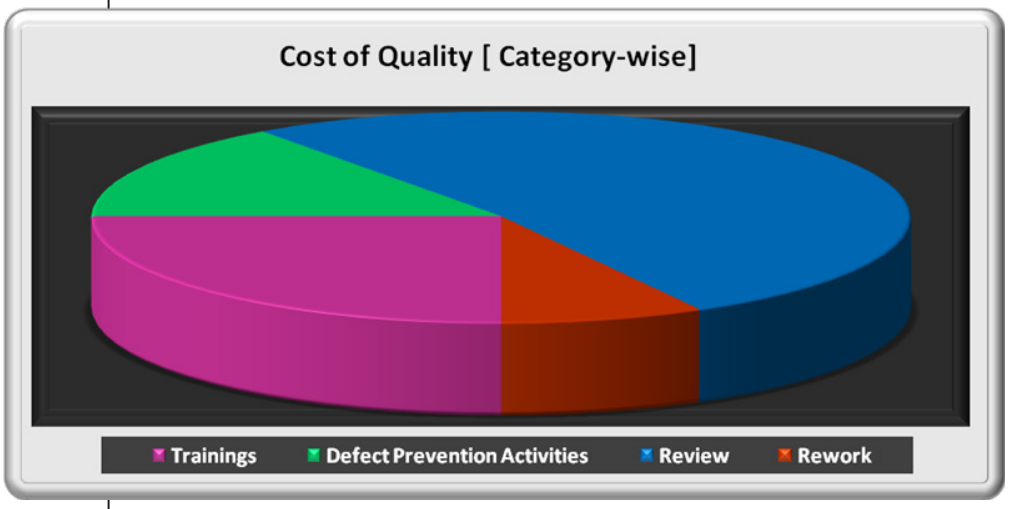


Cost of Quality

The software development life cycle concept of "Cost of Quality" does apply to software testing life cycle as well. As we recognnize "Cost of Quality", it is the cost of NOT creating a quality product or service right the first time. Analogously, in the context of software testing, "Cost of Quality" is the cost of not executing any of the testing activities right the first time viz. incorrect execution of a test case resulting in rework to reexecute the same test case.

"Cost of Quality" in the context of software testing does comprise of three types of costs viz. Prevention, Appraisal and Failure. Here, Prevention refers to various activities carried out to prevent any defects in the testing process viz. team trainings, creation of checklists and templates and various Defect Prevention activities carried out in the project etc. Appraisal costs include the effort going in to various reviews and checks conducted to ensure that the testing process proceeds smoothly viz. review of various testing artifacts, review of test execution techniques etc. Failure costs include the effort going in to rework required due to inadequate testing process and artifacts.





Tracking the metric of "Cost of Quality" for the testing process enables test managers to find areas of waste and take remedial actions in the form of strengthening of Prevention activities in order to reduce overall "Cost of Quality". Adjoining figures depict illustrations of "Overall Cost of Quality" and "Categories of Cost of Quality".

Product Quality Metrics

While the process of software testing plays a crucial role towards ensuring a high quality end product, it is vital to measure in-process quality throughout the testing life cycle to enable taking any remedial actions well in time. Software metrics play a key role helping teams drive various product quality improvement initiatives.

Software quality metrics can be divided into in-process quality metrics and end-product quality metrics. The core of software quality engineering is to examine the relationships amongst in-process metrics, end-product quality and project characteristics and based on the conclusions, to engineer perfection in the product quality.

Some of the key metrics, which enable one to measure in-process and end-product quality, are discussed below -

Metrics for Software Product Health

Testing Defect Arrival Rate

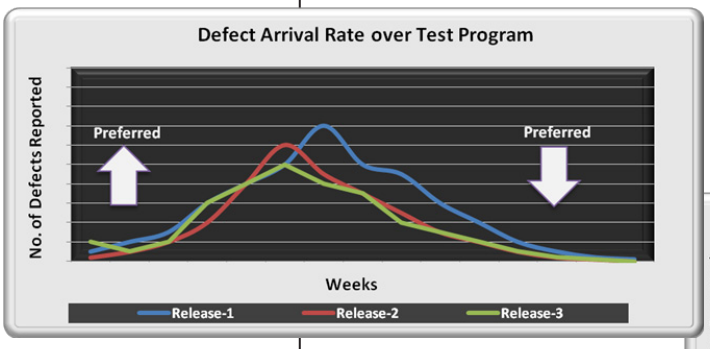
Defect tracking and management is the key to driving software product quality through the software testing life cycle. One of the metrics used in this direction is the pattern of defect arrivals over time. Even with similar overall defect counts during the test program, distinct patterns in defect arrival rates may mean different states of end-product quality.

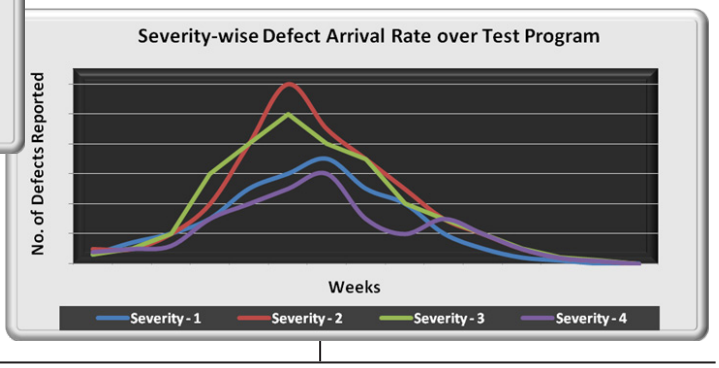
While tracking and analyzing defect arrival rate for a particular release, it is advisable to compare this trend with a set baseline (trends for a prior release, similar project or an ideal trend), if such a data is accessible. In the absence of a baseline, it is advisable to set some anticipated level of defect arrivals at certain points in the overall project schedule for example, midpoint of test cycle, entry in to system test cycle etc. Adjoining figure depicts an illustration of this metric.

A defect arrival pattern with higher arrivals during earlier stages, an earlier peak and a decline during the final stages (compared to the baseline) is regarded as positive. Also, another way to term a defect arrival pattern positive may be a pattern that is constantly lower than the baseline throughout the testing life cycle and hence signifies that testing effectiveness is at least as good as the previous one. The concluding end of the defect pattern curve is especially important as it acts as a prime indicator of the product quality in the field. An elevated defect activity just before the product release is generally a sign of quality issues.

Based on the defect arrival patterns observed, an in-depth root cause analysis needs to be carried out to identify improvement actions. The teams should try to control defect arrival rates only in a natural way, which is a function of the test effectiveness, test progress and inherent quality of the product. Another related advice is to open defects as soon as they are found so they do not unnecessarily get delayed.

Another associated metric is the defect arrival pattern for defects of various severity levels. Getting higher severity defects during final stages of testing life cycle might hamper product release in the market or some components not getting in to the finally released product.

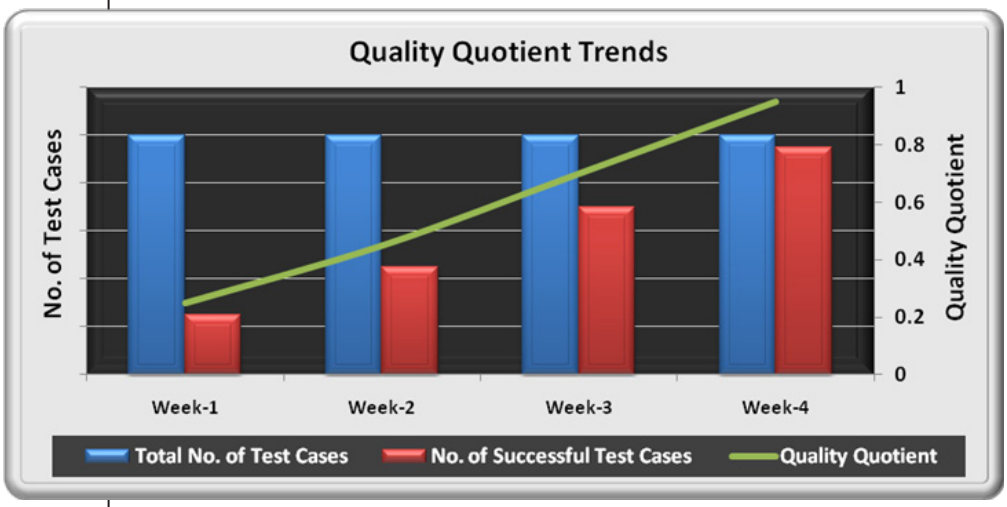




Quality Quotient Trends

A common question, which gets raised commonly to test managers by the top executives, is "How do you rate the current quality of the product being developed and tested?" There can be qualitative answers to this question. The metric of "Quality Quotient" enables test managers to answer this question with quantitative evidence and also manage the product quality as we move ahead through the test program. This metric provides indications about the amount of functionality that has successfully been demonstrated.

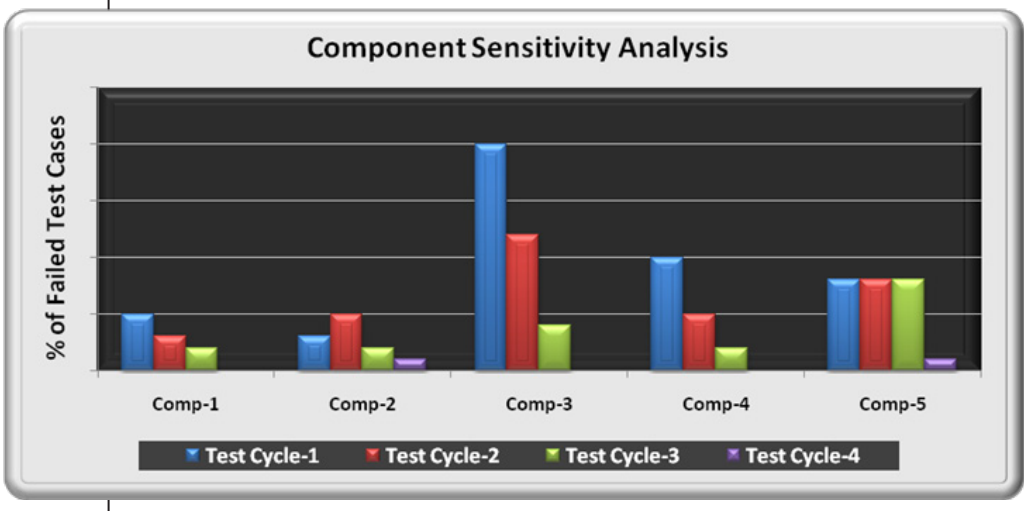
"Quality Quotient" is the number of test cases successfully executed (without any defects) versus the total number of test cases. Tracking and reporting this metric throughout the test program enables management to get "At a Glance View" of the product health and take required actions. Adjoining figure provides an illustration of this metric.



Component Sensitivity Analysis

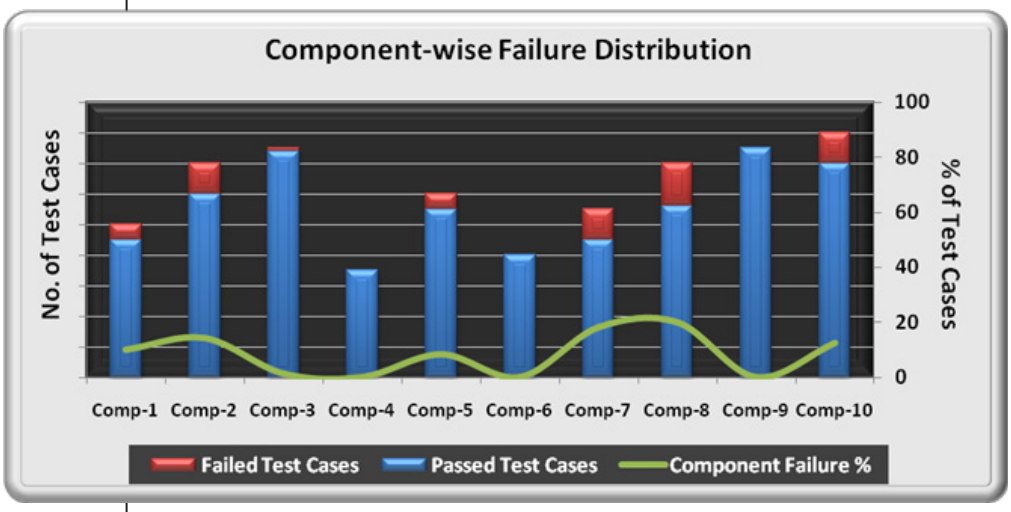
It is vital that all the components of a product should be of high quality for the product to be regarded as a high-quality product.

A metric to track the quality state of various components is "Component Sensitivity Ananlysis", which tracks the failures at each component level as we move from one to other test cycle in the whole test program. In an ideal scenario, the percentage of failures should demonstrate a reducing trend as we move from initial stages of testing life cycle to the final stages. Components, which are more defect prone in the final stages of testing life cycle are candidates of causing failures, when released to the field and may be termed as "Sensitive Components". Adjoining figure depicts an illustration of this metric. The sensitive components require specilized focus and efforts to ensure that they do not hamper overall product quality.



Component-wise Defect Distribution

A metric closely associated with the metric of "Component Sensitivity Ananlysis" is "Component-wise Defect Distribution". This metric serves an agreegate percentage of failures observed at each component level during whole of the test program. This metric along with the above, helps test teams to designate sensitive features and have a special focus towards those features in future test cycles or test programs for future enhancement releases of the same product. The two metrics also give a predictability of how various components are likely to behave post release to end customers. Adjoining figure depicts an illustration of this metric.



Additional Defect Management Metrics

As mentioned earlier, effective defect tracking, analysis and reporting are critical steps in the testing process. A well-defined method for defect management will benefit more than just the testing team. The defect metrics are a key parameter for evaluating the health of the product at any stage during test execution. Some more metrics, which enable effective defect management are discussed below. -

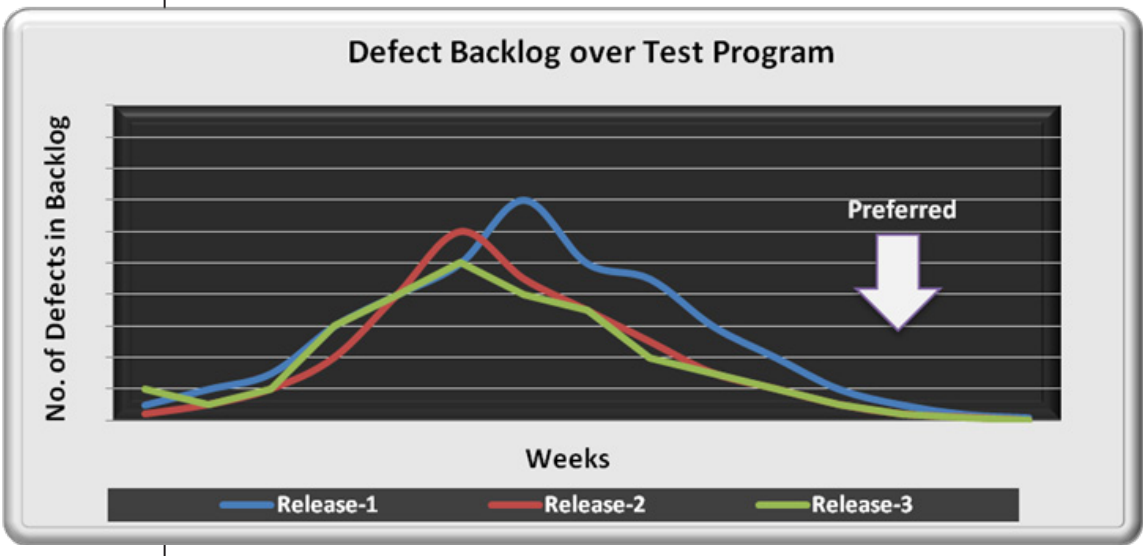
Testing Defect Backlog over Time

Defect backlog is the number of testing reported defects yet to be fixed at any given point of time. Defect backlog tracking and management is vital from the standpoint of both test progression and customer reported issues. Considerable number of unresolved defects during development cycle will hamper test progress. During final stages, when the product is nearing shipment, a high defect backlog means already hanging backlog defects getting reported by customer resulting in customer dissatisfaction.

During early test cycles, the focus should be on test execution effectiveness and efficiency and defect detection should be to the maximum possible extent. The focus of development team during this stage should be to shorten turnaround time for the critical defects that obstruct testing progress instead of the whole defects backlog. As testing approaches completion, strong focus should be given towards reduction in defect backlog.

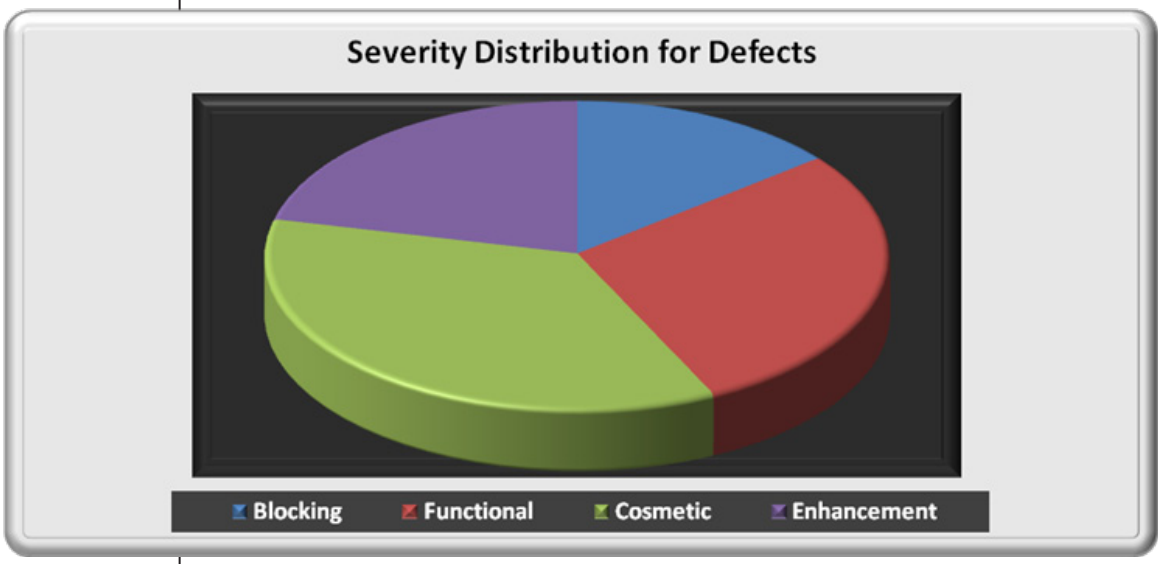
Adjoining figure depicts an illustration of comparative defect backlog trends amongst three releases. Here again, release-to-release comparisons and actual trends versus targeted trends should be the key objectives.

Also, this is to be noted that for this metric, focusing only on reducing overall defect numbers might not be sufficient. Besides the overall reduction, determining which specific defects to be fixed on priority is really significant in terms of achieving early software product stability. Towards this, the expertise of development and test teams is crucial.



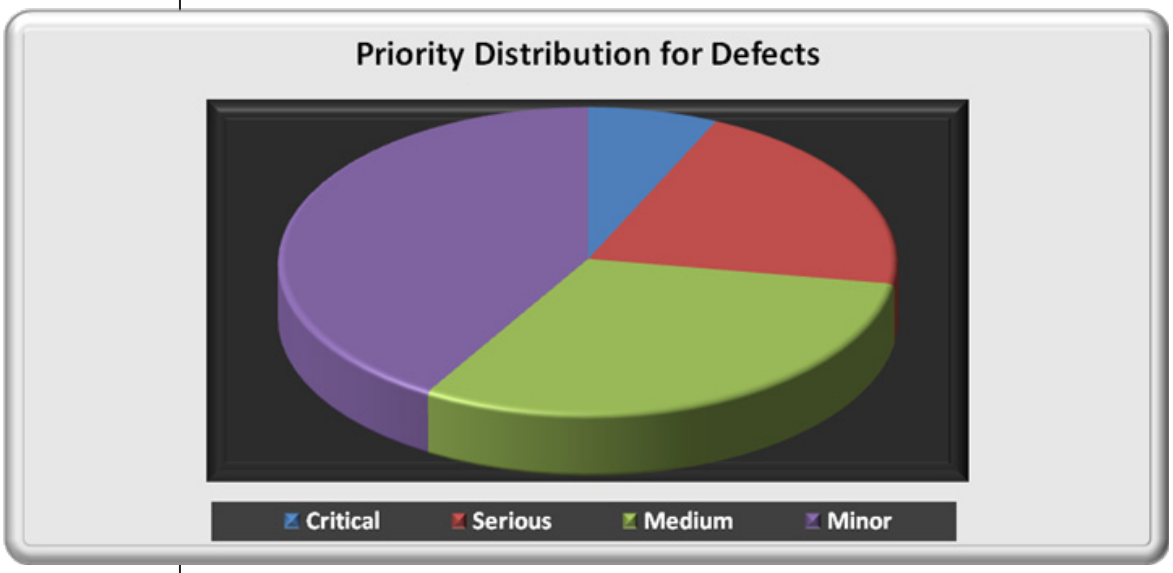
Defects Severity Distribution

At any point during the testing life cycle, tracking severity distribution of defects outstanding against the product provides an accurate and quick insight in to the product's health and enables team focus efforts in the right product areas. For instance, Blocking and High-Priority functional defects will generally be prime areas of focus to work upon by the development team, while some of the cosmetic defects and enhancement requests may be chosen to be deferred.



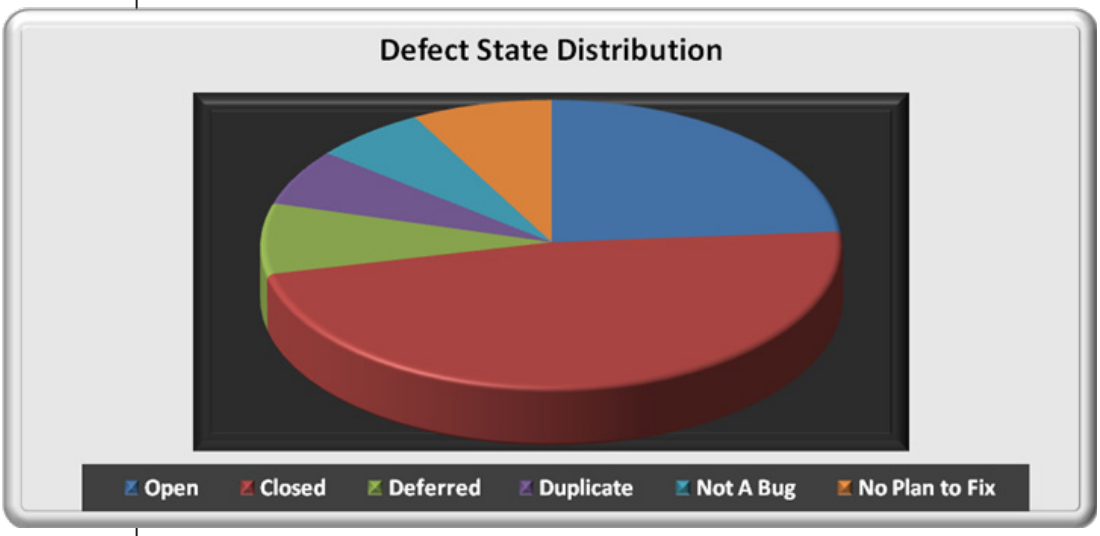
Defects Priority Distribution

Another defect categorization, which is commonly used in combination with the Defects Severity Distribution, is the priority distribution of defects outstanding against the product. The priority distribution of defects, when looked at in conjunction with the severity distribution serves as a guide for the development team, while deciding on the defects to be taken up for fixing. During the product final stages, this metric helps Final Gate Review teams to make an assessment on release readiness for the product.



Defect State Distribution

One of the key objectives of defect management activities is to bring various defects logged against the product to a logical closure. The metric used to analyze the defects situation at hand is "Defect State Distribution". This metric provides a quick snapshot of the overall defect status of the product. The metric serves as a useful reference during "Defect Triage" meetings, where representatives from various teams, viz. development, testing, product management, documentation etc. participate and a joint decision is taken on the action to be taken for various defects in "Open" state along with the decisions to defer any particular defect or decide not to fix. As indicated earlier, defects in "Not A Bug" state serve as a quality indicator of the test team's performance.

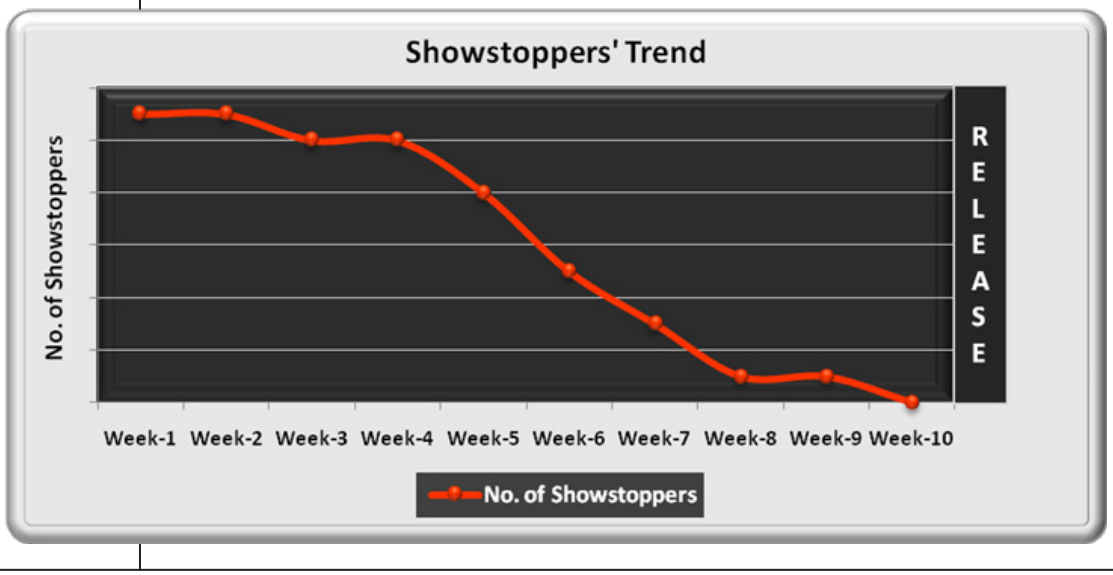


Showstoppers' Trend

A Showstopper is a defect, which is generally having highest priority and severity and which may make a product dysfunctional and hence, hamper the product release in to the market. Sometimes, the defects, which put a halt on the further progress of testing, are also termed as showstoppers during early stages of testing.

Tracking showstoppers is very important because severity and impact of various defects may vary. Regardless of the total defect volume, only a few showstoppers can hamper further progress of testing or hold a product release.

Tracking of showstopper defects generally starts at the middle of the test program, when Defect Triage meetings start happening. All problems on the showstoppers' list must be resolved before the product can be shipped. Adjoining figure depicts an example of this metric



Metrics for Automated Testing

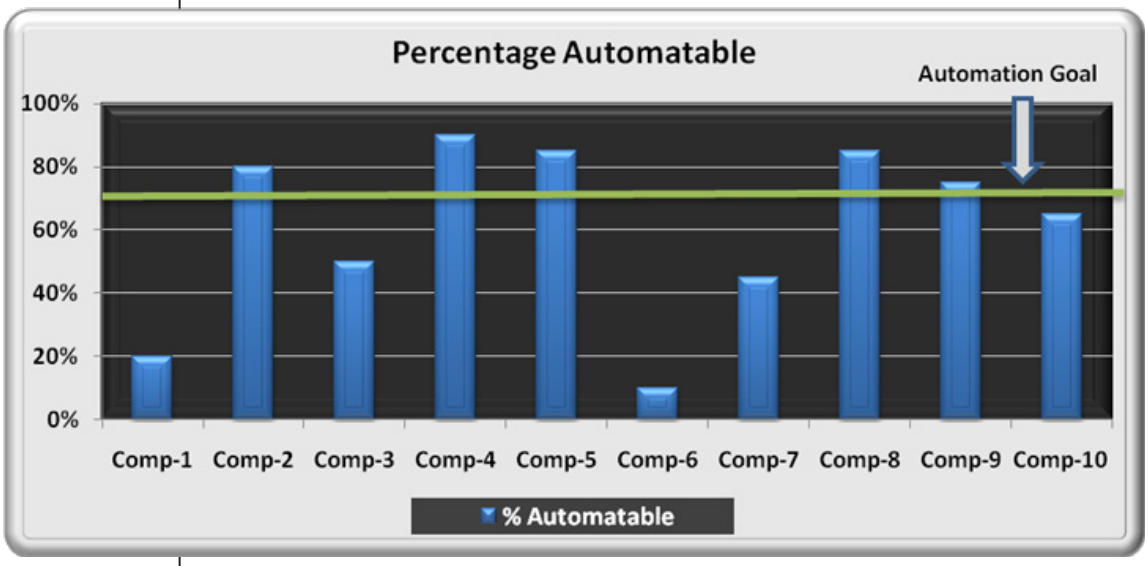
Test Automation, being a prime focus of any Testing endeavor forms a core part of the overall testing process. Just like the testing process, test automation process too has associated levers, which if tuned properly may contribute in a large way to the efficiency and effectiveness improvements for the testing process. As part of a successful Test Automation initiative, it is vital that appropriate goals and strategies are defined upfront and then put in to practice. There are several metrics related to Automated Testing, which enable us to gauge progress against the set goals and strategies during implementation and help bring-in continuous improvements.

Some of the key metrics, which enable one to measure various efficiency and effectiveness aspects of Test Automation process, are discussed below -

Automatable Percentage

When kick-starting a test automation effort, an assessment has to be made on the extent of the test bank, which can be automated or "is automatable". This assessment does take in to account multiple factors related to product stability and cost-benefit analysis. Given enough creativity and resources, one can claim that everything is automatable (which is in fact true as well!). But, the final decision on whether to select any component for automation or not will largely depend up on the benefits seen out of the automation vis-à-vis the cost involved in automating. Overall and apart from this criterion is the automation goal set by the organization. In case, the "Percentage Automatable" for any particular component significantly falls below the automation goal set by the organization, that component may not be a good candidate for automation.

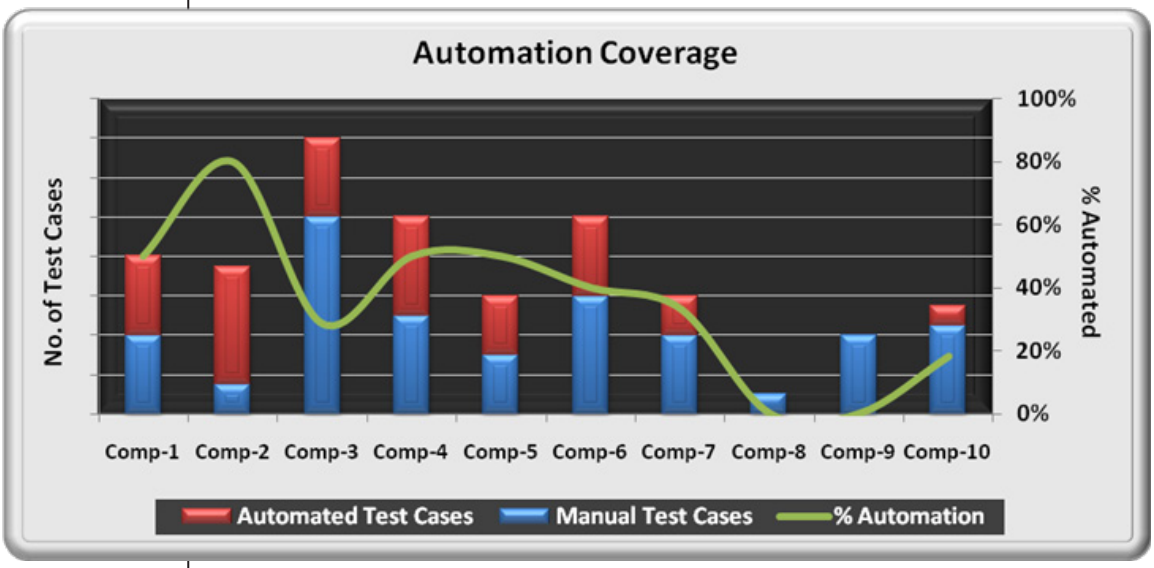
This metric can be used to assess the % automatable of various products or components within an organization and set the automation goal.

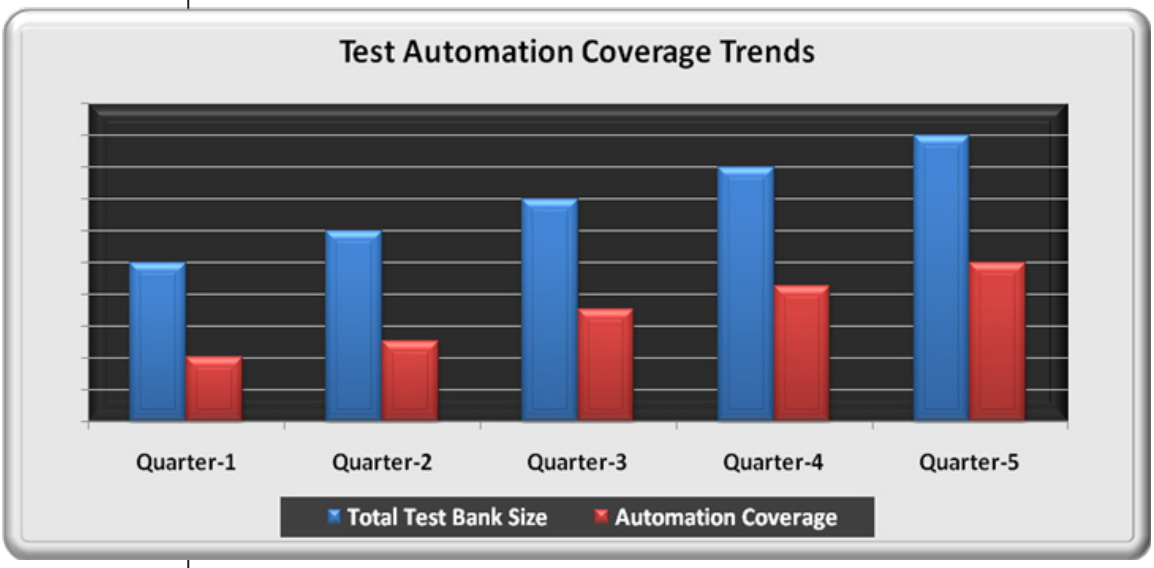


Test Automation Coverage

This metric helps test teams understand the extent of automation done at each component level and enables identify areas, where automation is low and can targeted for future automation efforts. Adjoining figure depicts an illustration of this metric.

Another coverage metric that facilitates management observe the growth trends in test automation vis-à-vis the growth trends in test banks over multiple time intervals, for example months or quarters. In case any lags are identified, appropriate actions can be taken to bring the activity back on track.



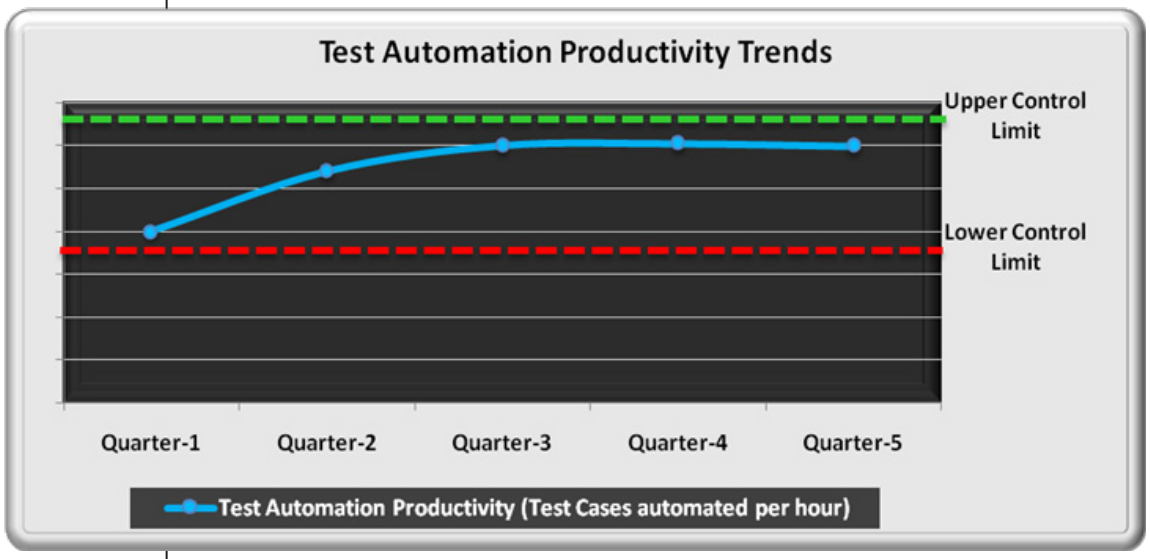


Test Automation Productivity Trends

Similar to the metric of "Test Execution Productivity", a useful measure of test automation team's efficiency is the productivity at which the team automates test cases. The test automation productivity may be defined as the average no. of test cases automated by the team per unit of time, where the unit may be hour, day or week, but is generally taken as an hour or a day.

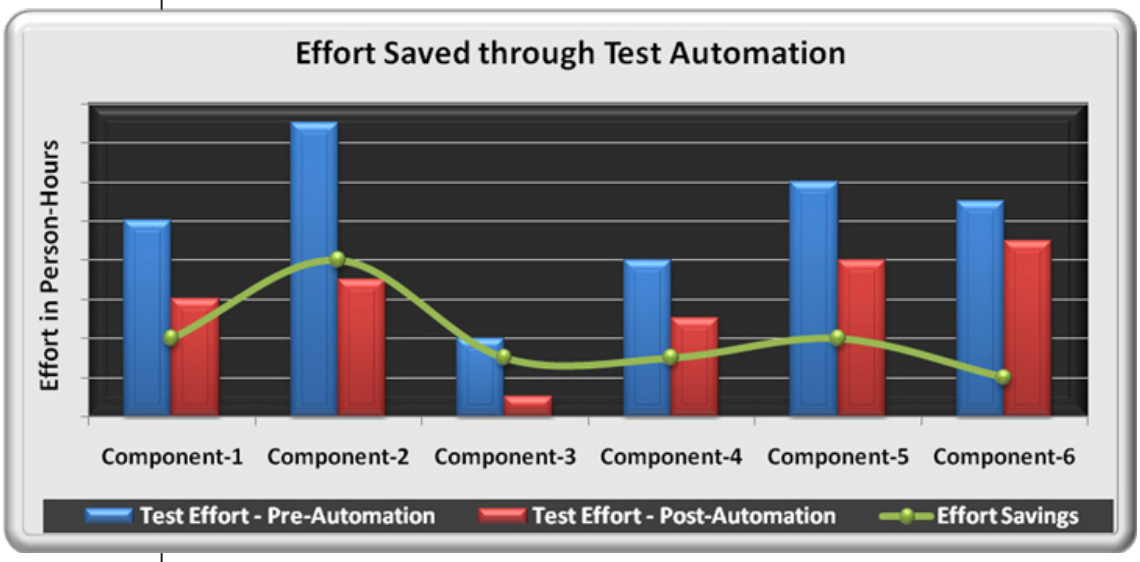
The metric depicts a trend of test automation productivity delivered by the team over multiple test automation exercises, weeks or any other time interval.

The objective of this metric is to help identify any problematic areas impacting team's productivity and take remedial actions, where feasible. Based on past experience or set baselines, upper and lower control limits can be set for test automation productivity and special focus be given to situations, wherein the productivity falls below lower control limits or goes above upper control limits. Adjoining figure depicts an illustration of this metric.



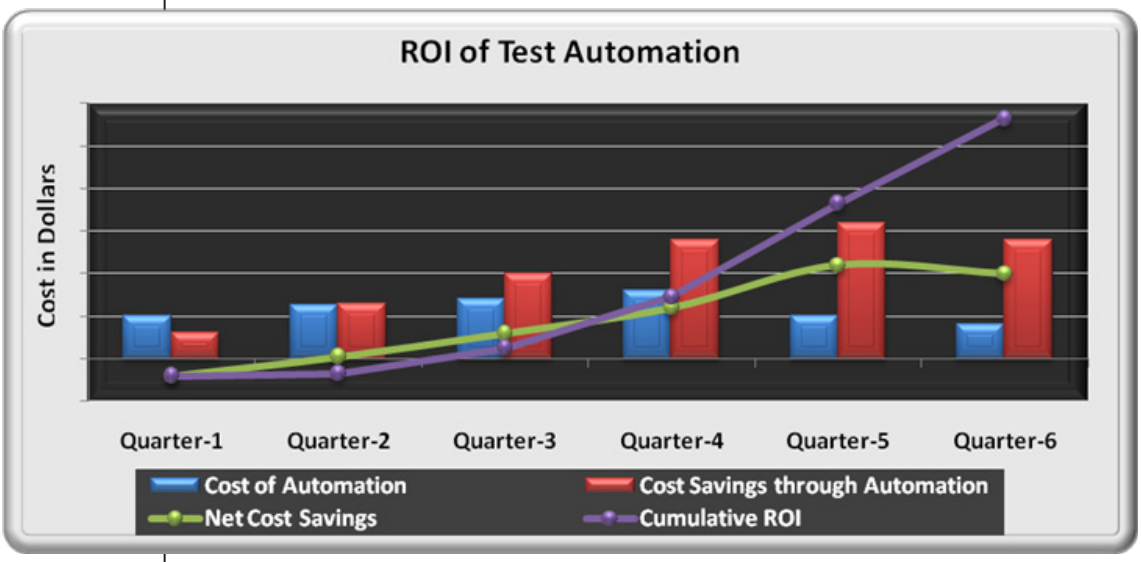
Effort Savings through Test Automation

It is essential to measure and track the benefits of any test automation effort in order to assess whether the earlier decisions taken up for automating any particular test bank have been beneficial. The metric, which enables test teams measure benefits out of test automation, is how much effort they are able to save through the usage of automation per time interval. Adjoining figure depicts an illustration of this metric. The metric shows component-wise savings in execution time achieved through test automation usage. The metric helps evaluate benefits of test automation and validate your automation selection decisions taken in the past to be better equipped to decide on future automation roadmaps.



ROI of Test Automation

Test Automation consumes a considerable amount of cost and effort in the form of people, automation frameworks and required tools. Hence, carrying out a pre and post automation Return-On-Investment (ROI) analysis is imperative to predict well and validate your decisions. A well thought out decision to automate will take in to account the one-time cost for automation, long-term costs to be incurred for maintaining the automation and the long-term benefits for automation. Such an analysis equips teams to take management buy-ins while entering in to any automation initiatives. Generally, any new automation effort will incur more costs vis-à-vis the returns or cost savings in the initial few months or years and it is only after some time that the real benefits of automation efforts may be derived. It is to be noted here that for products having a longer life-span these returns will start rising exponentially in later years of the product life span. Adjoining figure depicts an illustration of an example ROI scenario.



Tips for institutionalizing a

Successful Metrics-oriented Culture

Reluctance is often a software practitioner's first response to any new metrics endeavor. People may get worried, the data might be used against them or it will consume too much of their time to collect and analyze data.

Institutionalizing a software measurement culture and tackling such resistance will require a conscientious and consistent direction from leaders who are committed to deploying these metrics and aware of these concerns.

You must educate your team about the true intent behind the metrics program, why measurement is important and metrics data will never be used metrics either to punish or reward individuals and then making sure that you practice this saying.

Also, Software measurement doesn't need to be a time consuming exercise. Once they get closely knitted with your process, generating them will rarely demand additional time and effort and you will eventually be saving tremendously on the time you used to spend earlier on decision making. Moreover, several commercial tools are available for integration in to the project processes, which will make various types of data generation seamless. Some of the examples of such tools include - daily time tracking tools, defect/problem tracking tools, code coverage tools etc.

Albert Einstein once said "Make everything simple as possible, but not simpler."

Because deploying the measurement culture and infrastructure will be a long term pursuit, its best to first select a basic set of metrics to start with. Once the team becomes accustomed to the idea of measurement and some amount of momentum is established, one can bring in newer metrics, which will provide additional information one needs to manage the projects and organization effectively. Always choose a balanced set of measurements that help prevent dysfunctional behavior by tracking team's performance in several complementary aspects of their work. For example, if team productivity is being measured, do measure quality as well.

Choosing the right metrics set is equally important. The metrics set should not be a real "Big One" to measure everything. It should comprise of various necessary and sufficient metrics, which help measure the vital project attributes.

The intent behind collecting various metrics should be shared with the team so that they get motivated to participate in various measurement activities. Finally, trends in the data over time are more significant than individual data points. Measurement only will not improve performance, but it serves information that will help management and team focus on improvement efforts.

Summary

In this paper, we discussed about a comprehensive set of process and product quality metrics for the software testing life cycle along with few useful metrics applicable for automated testing process. There are certainly many more metrics for software testing, which can be derived and are not covered here. Also, all the metrics presented may not be viable or useful to measure for a particular software testing process. It is very evident that it is the effectiveness of the metrics model that actually matters, not the number of metrics used. The key objective of this paper is to serve as a guide to understanding the significance of the measurement model. At the same time, one should take a dynamic and flexible approach; that is to say, tailor the metrics based on the needs of specific project, product and organization.

A buy-in of the team must be obtained in order for the metrics to be successful and effective. Metrics are a means to reach a goal – "project success" and not an end in itself. A project team, which has logical understanding of the metrics model and data, will be able to make the right sight use of this tool for effective decision making. Per se, the usage of specific metrics cannot be directed from the top to down. While effective metrics can serve as a useful tool for software practitioners, they do not robotically lead to improvement in testing and process/product quality.

A key point to be noted is that after going through all measurements and metrics, and qualitative indicators, the team needs to go one level up and take a big-picture view in order to come to a final analysis. The final judgment and decision making should be driven by analysis and not only data. Metric assist in decision making, but do not substitute it.

Continuous improvement is the key to success of any process. Having illustrated the metrics model in place, we will have to continuously enhance the metrics model to strive for continuous improvement.

As H. James Harrington truly said - "The journey towards excellence is a never ending job".

References

- Metrics and Models in Software Quality Engineering – Second Edition by Stephen H. Kan
- Software Engineering: A Practitioner's Approach, 6/e by Roger S Pressman, R.S. Pressman and Associates, ISBN: 0072853182, Copyright year: 2005
- Implementing Automated Software Testing: How to Save Time and Lower Costs While Raising Quality by Elfriede Dustin Thom Garrett Bernie Gauf

About the Author

Mandeep Walia is a Group Project Manager with Infosys Technologies Ltd. He has over 13 years of IT experience encompassing Software Development, Maintenance, Testing and Professional Services. He is certified as a Project Management Professional (PMP) by PMI and a Certified Software Quality Analyst (CSQA) from QAI, USA. During his career at Infosys, Mandeep has managed multiple large and complex software programs for Fortune 500 companies.