Testing Metrics

Testing Metrics

Introduction

It is often said that if something cannot be measured, it cannot be managed or improved. There is immense value in measurement, but you should always make sure that you get some value out of any measurement that you are doing. You should be able to answer the following questions:

- What is the purpose of this measurement program?
- What data items you are collecting and how you are reporting it?
- What is the correlation between the data and conclusion?

Metrics are the means by which the software quality can be measured; they give you confidence in the product. You may consider these product management indicators, which can be either quantitative or qualitative. They are typically the providers of the visibility you need. Metrics usually fall into a few categories: project management (which includes process efficiency) and process improvement. You may use different metrics for different purposes. For example, you may have a set of metrics that you use to evaluate the output of your testing team. One such metric may be the project management measure of the number of defects found. Others may be an efficiency measure of the number of test cases written, or the number of tests executed in a given period of time.

Why Measure?

If an organization wishes to improve its level of software development in any area, it must have a scale by which to measure itself. It must know where it is, and it must know where it wants to go. Simply stated: If you cannot measure where you are, you cannot demonstrate that you are improving.

When used properly, test metrics can aid in software development process improvement by providing pragmatic, objective evidence of process change initiatives. Metrics are defined as “standards of measurement” and have long been used in the IT industry to indicate a method of gauging the effectiveness and efficiency of a particular activity within a project. Although test metrics are gathered during the test effort, they can provide measurements of many different activities performed throughout a project. In conjunction with root cause analysis, test metrics can be used to quantitatively track issues from points of occurrence throughout the development process. In addition, when test metrics information is accumulated, updated and reported on a consistent and regular basis, it ensures that trends can be promptly captured and evaluated.

What to Measure?

Any measurement program can be divided into two parts. The first part is to collect data, and the second is to prepare metrics/chart and analyze them to get the valuable insight which might help in decision making. Information collected during any measurement program can help in:

- Finding the relation between data points,
- Correlating cause and effect,
- Input for future planning.

Normally, any metric program involves certain steps which are repeated over a period of time. It starts with identifying what to measure. After the purpose is known, data can be collected and converted into the metrics. Based on the analysis of these metrics appropriate action can be taken, and if necessary metrics can be refined and measurement goals can be adjusted for the better.

Data presented by testing team, together with their opinion, normally decides whether a product will go into market or not. So it becomes very important for test teams to present data and opinion in such a way that data looks meaningful to everyone, and decision can be taken based on the data presented.

Testing Metrics

Every testing project should be measured for its schedule and the quality requirement for its release. There are lots of charts and metrics that we can use to track progress and measure the quality requirements of the release.

Metric	Purpose
Closed Defect Distribution	This chart gives information on how defects with closed status are distributed.
Defect Category	An attribute of the defect in relation to the quality attributes of the product. Quality attributes of a product include functionality, usability, documentation, performance, installation, stability, compatibility, internationalization etc. This metric can provide insight into the different quality attributes of the product. This metric can be computed by dividing the defects that belong to a particular category by the total number of defects.
Defec Cause Distribution Chart	This chart gives information on the cause of defects.
Defect Distribution Across Components	This chart gives information on how defects are distributed across various components of the system.
Defect Finding Rate	This chart gives information on how many defects are found across a given period. This can be tracked on a daily or weekly basis.
Defects/ KLOC or FP	The number of defects per 1,000 lines of code or Function Points. This metric indicates the quality of the product under test. It can be used as a basis for estimating defects to be addressed in the next phase or the next version. Ratio of the number of defects found vs. the total number of lines of code (thousands) or Function Points.
Defect Removal Efficiency	The number of defects that are removed per time unit (hours/days/weeks). Indicates the efficiency of defect removal methods, as well as indirect measurement of the quality of the product. Computed by dividing the effort required for defect detection, defect resolution time and retesting time by the number of defects. This is calculated per test type, during and across test phases.
Defect Severity	The severity level of a defect indicates the potential business impact for the end user (business impact = effect on the end user x frequency of occurrence). Provides indications about the quality of the product under test. High-severity defects mean low product quality, and vice versa. At the end of this phase, this information is useful to make the release decision based on the number of defects and their severity levels. Every defect has severity levels attached to it. Broadly, these are Critical, Serious, Medium and Low.
Defect Severity Index	An index representing the average of the severity of the defects. Provides a direct measurement of the quality of the product—specifically, reliability, fault tolerance and stability. Two measures are required to compute the defect severity index. A number is assigned against each severity level: 4 (Critical), 3 (Serious), 2 (Medium), 1 (Low). Multiply each remark by its severity level number and add the totals; divide this by the total number of defects to determine the defect severity index.

mindlance

Testing Metrics

Effort Adherence	As % of what is committed in contract. Provides a measure of what was estimated at the beginning of the project vs. the actual effort taken. Useful to understand the variance (if any) and for estimating future similar projects.
Number of Defects	The total number of defects found in a given time period/phase/test type that resulted in software or documentation modifications. Only accepted defects that resulted in modifying the software or the documentation are counted.
Review Efficiency	# of defects detected /LOC or pages reviewed per day
Test Case Effectiveness	The extent to which test cases are able to find defects. This metric provides an indication of the effectiveness of the test cases and the stability of the software. Ratio of the number of test cases that resulted in logging defects vs. the total number of test cases.
Test Case Execution Productivity	# of test cases executed per day per person.
Test Case Execution Statistics	This metric provides an overall summary of test execution activities. This can be categorized by build or release, module, by platform (OS, browser, locale etc.).
Test Case Generation Productivity	# of test cases derived per day per person.
Test Coverage	Defined as the extent to which testing covers the product's complete functionality. This metric is an indication of the completeness of the testing. It does not indicate anything about the effectiveness of the testing. This can be used as a criterion to stop testing. Coverage could be with respect to requirements, functional topic list, business flows, use cases, etc. It can be calculated based on the number of items that were covered vs. the total number of items.
TTest Effort Percentage	The effort spent in testing, in relation to the effort spent in the development activities, will give us an indication of the level of investment in testing. This information can also be used to estimate similar projects in the future. This metric can be computed by dividing the overall test effort by the total project effort.
Time to Fix a Defect	Effort required to resolve a defect (diagnosis and correction). Provides an indication of the maintainability of the product and can be used to estimate projected maintenance costs. Divide the number of hours spent on diagnosis and correction by the number of defects resolved during the same period.
Traceability Matrix	Traceability is the ability to determine that each feature has a source in requirements and each requirement has a corresponding implemented feature. This is useful in assessing the test coverage details.
Schedule Adherence	As % of what is committed in contract. Provides a variance between planned and actual scheduled followed. Useful to understand the variance (if any) and for estimating future similar projects.
Scope Changes	The number of changes that were made to the test scope (scope creep). Indicates requirements stability or volatility, as well as process stability. Ratio of the number of changed items in the test scope to the total number of items.

Testing Metrics

It is worth noting that collection and analysis of metrics will need some investment in terms resources and time. Many of the organizations assign dedicated resources for this activity. Also, metrics are as good as a defect management system. In other words, it depends how effectively the defect management system is implemented and configured to capture relevant data and that respective teams update data on a regular basis. Many of the market leading tools (both commercial and open-source) provide enhanced reporting features to generate canned reports and graphs.

Metrics Analysis

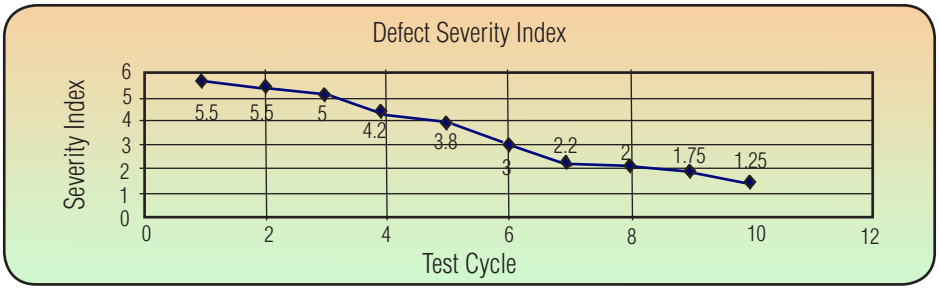
Much as the time is spent gathering or maintaining test metrics, enough time should be spent to review and interpret on a regular basis throughout the test effort, particularly after the application is released into production. During review meetings, the project team should closely examine all available data and use that information to determine the root cause of identified problems. It is important to look at several metrics, as this will allow the project team to have a more complete picture of what took place during a test.

If metrics have been gathered across several projects, a comparison should be done between the results of the current project and the average or baseline results from the other projects. Determine if the current metrics are typical of software projects in your organization. If development process changes were made for the current project, note if there were any visible effects on the metrics.

Let us assume that as part of the organization's test metrics program, the following metrics are collected by the test team.

#	Metric Name	Metric Definition
1	Defect Severity Index	Weighted average index of the Severity of defects. A higher severity defect gets a higher weight. S1 is a show stopper, S2 is high severity, S3 is medium & S4 is low. Ideally, this should slope down as test cycles progress.

For instance, if the test team has generated the following metrics:



Looking at the graphs one can safely deduce the following.

Defect Severity Index:

What does the graph indicate? The defect severity index is sloping down consistently. This indicates an increasingly favorable trend. As the test cycle progresses (from cycle 1 to cycle 10), the severity index is sloping which suggests increasing quality of the application (as lesser number of critical and high severity defects are being reported).

This is what it could mean: While a fall in the defect severity index is definitely a good trend, looking at this index in isolation could be misleading. Following factors need to be considered in order to have a meaningful analysis.

- Number of defects logged - let us consider an example where the test team executed two cycles of testing (assuming other things as constant). The number of defects logged against each of these cycles along with the calculated severity index is shown below.

Number of Defects		
Defect Severity	**Cycle 1(# of defects)**	**Cycle 2(# of defects)**
s1	5	5
s2	10	15
s3	50	30
s4	100	100
Severity Index	1.52	1.50

At first thoughts, when we compare cycle 1's Severity Index with cycle 2's Severity Index, cycle 2 looks to be favorable (as the severity index is lower). If you go into the details of the number of defects logged and their severity, the picture turns out to be the opposite. While the total number of Severity 1 and Severity 2 defects for cycle 1 is 15, the number of Severity 1 and Severity 2 defects for cycle 2 is 20. In terms of quality, cycle 1 is better than cycle 2 as cycle 1 has lesser number of high severity defects (though the total number of defects logged in cycle 1 is more than cycle 2 defects and the severity index is greater than cycle 2 severity index). Test coverage has a similar impact. A lower test coverage coupled with reducing severity index would not be a healthy trend.

- Defect Severity let us consider another example where the test team executed two cycles of testing (assuming other things as constant). The severity of defects logged against each of these cycles along with the calculated severity index is shown below.

Severity of Defects		
Defect Severity	**Cycle 1(# of defects)**	**Cycle 2(# of defects)**
s1	4	0
s2	4	0
s3	42	75
s4	27	2
Severity Index	1.81	2.03

Looking at the severity index, it looks like cycle 1 is better than cycle 2 (as the severity index is low for cycle 1 compared to cycle 2). However, cycle 2 is better than cycle 1 as total number of Severity 1 and Severity 2 defects is zero compared to a total of 8 severity 1 and severity 2 defects of cycle 1. Just because the severity index is low, do not believe the quality of the application is better than the earlier cycle.

Testing Metrics

To summarize, while the above defined metrics are a very small subset of the large number of metrics possible to generate, it definitely helps us understand an important aspect of test reporting. Relying on test metrics is always better; however, teams depending on the metrics need to have a better understanding of the different components and factors which affect these metrics. Any test report that presents metrics along with detailed root cause analysis by considering different point of view is important to ensure, team does not face surprises in the future.

Conclusion

The metrics provided by testing offer a major benefit to executives: visibility into the maturity and readiness for release or production, and visibility into the quality of the software product under development. This enables effective management of the software development process, by allowing clear measurement of the quality and completeness of the product.