

View Point



Operational Excellence through Efficient Software Testing Metrics

Ramesh Pusala

Abstract

Metrics are gaining importance and acceptance as organizations mature and strive to improve enterprise quality. Process measurement and analysis, and utilization of quantitative methods for quality management are the two key process activities at Level 4 Maturity of CMM. While the need for metrics has been recognized, implementation of structured measurement programs is lagging, especially in the software testing area.

Although many metrics have been proposed by researchers, they are either ignored or used in isolation. A software test manager must measure the effectiveness of a test process. Efficient test process measurement is essential for managing and evaluating the effectiveness of a test process. Effective measurement is the first step to make testing a true engineering discipline.

This paper addresses various aspects of a metrics program: The need for having a metrics program, implementation challenges, addressing these challenges, arriving at an ideal set of metrics, etc.

Introduction

A software test manager must have the competency to measure test processes effectively. The key to effective measurement lies in the ability to clearly identify the goals to be accomplished and the issues to be tackled. Many test managers and their organizations waste time and money in measuring more things than are necessary.

Test metrics are an important indicator of the effectiveness of a software testing process. The first step in establishing test metrics is to identify the key software testing processes that can be objectively measured. This information can be used as the baseline to define the metric(s) and to determine what information will be tracked, who will track the information and at which frequency. Then the processes necessary to effectively track, calculate, manage and interpret the defined metrics must be implemented. Areas for process improvement can be identified based on the interpretation of the defined metrics.

Paul Goodman defines software metrics as “The continuous application of measurement-based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products”.

G. Gordon Schulmeyer defines a metric as “A quantitative measure of the degree to which a system, component or process possesses a given attribute”. Although challenging, software measurement is an essential component for a healthy and high-performance software engineering culture. This paper revisits some basic software measurement principles and suggests some metrics that can help you better understand and improve operations in your organization. Measurement activities must be planned carefully because they require significant efforts to implement, and returns are realized only over a period of time.

Why do we need Test Metrics?

As we all know, a major percentage of software projects suffer from quality problems. Software testing provides visibility into product and process quality. Test metrics are key “facts” that project managers can use to understand their current position and to prioritize their activities to reduce the risk of schedule over-runs on software releases.

Test metrics are a very powerful risk management tool. They help you to measure your current performance. Because today's data becomes tomorrow's historical data, it's never too late to start recording key information on your project. This data can be used to improve future work estimates and quality levels. Without historical data, estimates will just be guesses.

You cannot track project status meaningfully unless you know the actual effort and time spent on each task as compared to your estimates. You cannot sensibly decide whether your product is stable enough to ship unless you track the rates at which your team is finding and fixing defects. You cannot quantify the performance of your new development processes without some statistics on your current performance and a baseline to compare it with. Metrics help you to better control your software projects. They enable you to learn more about the functioning of your organization by establishing a *Process Capability Baseline* that can be used to better estimate and predict the quality of your projects in the future.

The benefits of having good metrics:

- Test metrics data collection helps *predict* the long-term direction and scope for an organization and enables a more holistic view of business and *identifies high-level goals*
- Provides a *basis for estimation* and facilitates planning for closure of the performance gap
- Provides a means for *control / status* reporting
- Identifies *risk areas* that require more testing
- Provides meters to flag actions for faster, more informed *decision making*
- Quickly identifies and helps resolve potential problems and identifies *areas of improvement*
- Test metrics provide an objective measure of the *effectiveness and efficiency of testing*

The role that effective metrics can play to radically change a business' performance has been very well illustrated in the article titled *Market Busting: Strategies for Exceptional Business Growth* by Rita Gunther McGrath and Ian C. MacMillan in the Harvard Business Review, March 2005. I quote a few statements:

- Some companies focused on different key metrics than their competitors did, and in doing so, created a better business design. Still other companies helped customers change their own unit of business or key metrics.
- [Change your unit of business](#), so it more closely reflects the value created for customers. You will probably also want to change how you measure the effectiveness of your performance.
- [Dramatically change your performance on existing key metrics](#) in a way that uniquely favors your company.

Key factors to bear in mind while setting up Test Metrics

Collect only the data that you will actually need/use to make informed decisions or to alter your strategy. That is, if you are not going to change your strategy regardless of the findings, your time is better spent in testing.

Do not base decisions solely on data that is variable or can be manipulated. For example, measuring testers on the number of tests they write per day can reward them for speeding through superficial tests or punish them for tackling trickier functionality.

Use statistical analysis to get a better understanding of the data. Difficult metrics data should be analyzed carefully. The templates used for presenting data should be self explanatory.

One of the key inputs to the metrics program is the defect tracking system in which the reported process and product defects are logged and tracked to closure. It is, therefore, very important to carefully decide on the fields that need per defect in the defect tracking system and then generate customizable reports.

Metrics should be decided on the basis of their importance to stakeholders rather than ease of data collection. Metrics that are not of interest to the stakeholders should be avoided.

Inaccurate data should be avoided, and complex data should be collected carefully. Proper benchmarks should be defined for the entire program.

Metrics Initiative at Infosys

There was a shared need, felt across Infosys, to have tangible and realistic metrics across all projects executed by it. These metrics were to be used to define the Process Capability Baseline (PCB) for the organization.

Arriving at a common set of metrics across the organization — spread across different locations and with projects on varied technologies, was quite a challenge.

The Process

Kick off meeting

This initiative started with a kick off meeting in which the importance and shared need of metrics was stressed upon, to get the buy-in from all sections of the team.

Taking stock of existing metrics

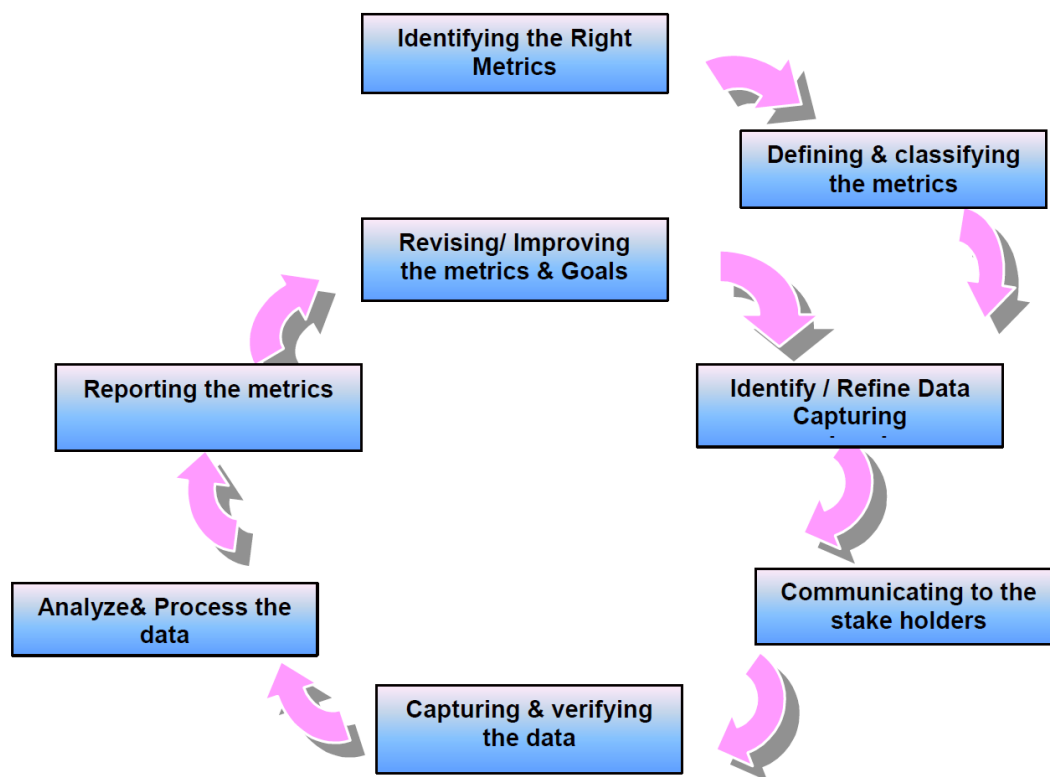
Brainstorming sessions were conducted to review the relevance and importance of existing metrics, and to identify the pain areas and additional metrics that would add value.

The Metrics Lifecycle

The process of setting up metrics involves:

- Identifying the metric
- Prioritizing metrics
- Classifying metrics that may be project specific
- Identifying data required for the metric; if data is not available, identify/setup process to capture the data
- Communicating to stakeholders
- Capturing and verifying data
- Analyzing and processing data
- Reporting

Let us take a detailed look at each of the above mentioned steps that ensure successful standardization of process.

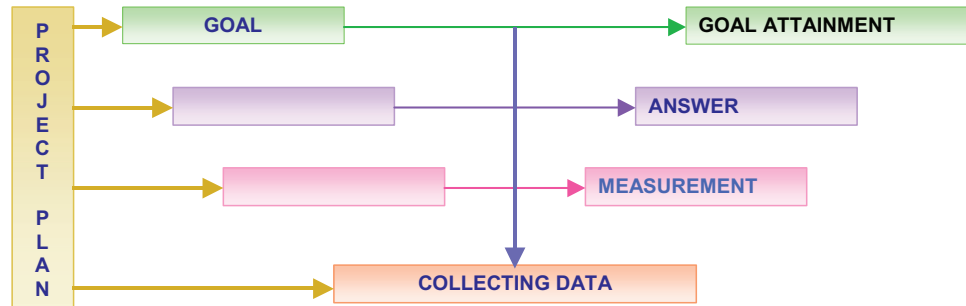


Step 1: Identify the right metrics

The right metrics can be identified only after:

- Deciding the audience (executive team, test team, etc)
- Identifying the metrics which capture the status of each type of testing
- Ensuring that all the different categories of metrics are considered based on project needs
- Setting up easy mechanisms for data collection and data capture
- Analyzing the value/benefit of each metric and the project lifecycle phase in which it provides the most value
- Identifying the goals or problem areas where improvement is required
- Refining the goals, using the “Goal-Question-Metric” technique

“Goal-Question-Metric” method for identifying metrics



Goal-Question-Metric (GQM) is an excellent technique for selecting appropriate metrics to meet your needs. The GQM works like this: Begin by selecting a few projects or organizational goals. State the goals in as quantitative and measurable terms as possible. Then, find out what must be changed to reach that goal, and finally, define what is to be measured to quantify progress made to achieve the goal.

- GQM is a systematic approach for integrating goals to the process
- The metrics relevant to process improvement can be effectively identified and tailored to the organization and its goals
- Measurement provides the most appropriate information to ensure consistency and completeness in the quest for goal attainment

Step 2: Define and classify metrics

- Provide the definition for each metric
- Define the benchmark or goal for each metric
- Verify whether the goals or benchmarks are realistic, by comparing with industry standards or with data of similar projects within the organization
- Based on the type of testing, metrics are mainly classified into:
 - Manual testing
 - Automation testing
 - Performance testing
- Each of these are further categorized based on the focus area:
 - Productivity
 - Quality
 - People
 - Environment / Infrastructure
 - Stability
 - Progress
 - Tools
 - Effectiveness

Step 3: Identify / refine the data capturing mechanism

Based on data capturing methodology, metrics can be classified as:

- **Base metrics** - Metrics for which data can be captured directly (like time, effort, defects, test execution details, test case preparation details, etc.)
- **Derived metrics** - Derived from base metrics (productivity, quality, etc.)

After identifying suitable metrics, the data that is required for each metric must be analyzed by:

- Identifying the source of data for each of the base metrics
- Defining the common template for capturing all base metrics

- Obtaining feedback from the team which captures the data
- Streamlining significant data
- Defining efficient means to extract relevant data:
 - To avoid ambiguous data, the measurement value type has to be verified for data consistency
 - The technique for pulling out relevant data is based on the following criteria to aid the capture of relevant metrics

Data	Relevant	Reliable	Valid	Practical	Measurable	Comparable
	Is it meaningful data, linked to the goal	Consistent with respect to time	Measures the outcome	Economical and easy to collect data	Can be measured	Can be compared

This process step strives for continuous improvement of the data capturing methodology by reducing the data capturing effort and eliminating sources of inaccurate data.

Step 4: Communication

To ensure better end-results and to increase buy-in, the metrics identification and planning process must involve all stakeholders.

- Communicate the need for metrics to all the affected teams
- Educate the testing team regarding the data points that need to be captured for generating the metrics
- Obtain feedback from the stakeholders
- Communicate the expectations to stakeholders — how often the data needs to be collected, how often the reports will be generated, etc.

Step 5: Capture & verify data

- Ensure that the data capturing mechanism is set up and streamlined
- Communicate and give proper guidelines to the team members on the data that is required
- Set up verification points to ensure that all data is captured
- Identify the sources of inaccurate data for each base metric and take corrective steps to eliminate inaccuracies
- For each base metric, define a source of data and the procedure to capture data
- Ensure that minimum effort is spent on capturing the data by automating the data capturing process wherever possible (If you are using TestDirector for test management, data can be extracted by using TestDirector APIs)
- Capture the data in a centralized location easily accessible to all team members
- Collect the data with minimal manual intervention

Step 6: Analyze & process the data

- Once the data is captured, the data must be analyzed for completeness
- Verify whether the data filled is accurate and up-to-date
- Define the process/ template in which the derived data must be captured
- Calculate all the metrics (derived metrics) based on the base metrics
- Verify whether the metrics are conveying the correct information
- Automate the process of calculating derived metrics from the base metrics to reduce effort

Step 7: Reporting metrics

- Develop an effective approach for reporting, like a metrics dashboard
- It is advisable to obtain feedback from stakeholders and their representatives on the metrics to be presented by providing samples
- Metrics should be presented based on the audience and in a consistent format
- Reports should contain the summary of observations

- Reporting should be in a clearly understandable format, preferably graphs and charts with guidelines to understand the report
- Reports should clearly point out all the issues or highlights
- Based on the request, user should be able to access the data
- Reports should be presented in such a way that metrics are compared against benchmarks and trends shown
- Reports should be easily customizable based on the user requirement
- Ensure that the efforts spent on reporting is minimal; wherever required try to automate (if it is a Microsoft Excel based report, through the usage of macros)

Step 8: Continuous Improvement

- Continuous improvement is the key to the success of any process
- After successful implementation of metrics and after achieving the benchmark, revisit the goals and benchmarks and set them above industry standards
- Regularly collect feedback from the stakeholders
- Metrics can be added/deleted/modified based on the need
- Metrics reports must be accessible to everyone
- Evaluate new metrics to capture
- Refine the report template
- Ensure that the effort for capturing and reporting metrics is minimal

Challenges in implementation of a metrics program

While more and more organizations now appreciate the importance of proper metrics, there are lots of challenges in the successful implementation of a metrics program. According to the metrics guru Howard Rubin, up to 80 percent of all software metrics initiatives fail within two years.

To avoid the common pitfalls in test metrics, the following aspects need to be considered:

- **Management Commitment:** To be successful, every process improvement initiative needs strong management commitment in terms of owning and driving the initiative on an ongoing basis.
- **Measuring too much, too soon:** One can identify many metrics that can be captured in projects, but the key is to identify the most important ones that add value. It can be done in an incremental fashion, adding new metrics as the project progresses and as the team starts reaping the benefits of the metrics.
- **Measuring too little, too late:** The other mistake teams make is to collect few metrics too late in the process. This does not provide the right information for proper decision making.
- **Wrong metrics:** If the metrics do not really relate to the goals, it does not make sense to collect them.
- **Vague metrics definitions:** Ambiguous metric definitions are dangerous, as different people may interpret them in different ways, thus resulting in inaccurate results.
- **Using metrics data to evaluate individuals:** One of the primary reasons for a metrics program being not appreciated and supported by all levels of the team is the fear that the data may be used against them. So, never use the metrics data to evaluate a person.
- **Using metrics to motivate rather than to understand:** Many managers make the mistake of using metrics to motivate teams or projects. This may send signals that metrics are being used to evaluate individuals and teams. So the focus must be on understanding the message given by the metrics.
- **Collecting data that is not used:** There may be instances where data is collected but not really used for analysis; avoid such situations.

- **Lack of communication and training:** Proper communication is the key to the success of the metrics program. All stakeholders need to understand the relevance of any data that is collected.
 - **Explain why:** There is a need to explain to a skeptical team why you wish to measure the items you choose. They have the right to know your intention and why you think the data will be valuable. Use the data that is collected, and do not let it go unused.
 - **Share the results:** Your team will be more motivated to participate in the measurement activities if you inform them about how you have used the data. Share summaries and trends with the team at regular intervals and get them to help you understand the data. Let them know whenever you use their data to answer a question, make a prediction or a decision.
 - **Define data items and procedures:** It is more difficult and time-consuming to precisely define the data items and metrics than you might think. However, if you do not pin these definitions down, participants may interpret and apply them in different ways. Define what you mean by a “test case or test case point,” spell out which activities go into the various work effort categories, and agree on what a “defect” is. Write clear, succinct procedures for collecting and reporting the measures you select.
 - **Obtain “buy-in:** To have ‘buy-in” to the goals and the metrics in a measurement program, team members need to have a feeling of ownership. Participation in the definition process of the metrics will enhance this feeling of ownership. In addition, people who work in a process on a daily basis will have intimate knowledge of that process. This gives them a valuable perspective on how the process can best be measured to ensure accuracy and validity, and how best to interpret the measured result to maximize usefulness.
- **Misinterpreting metrics data:** While analyzing metrics one should be able to distinguish between the trends shown. Due to noise factors, metrics may mislead at times. One should not take decisions based on such variations.

Summary of activities involved in successfully implementing Test Metrics

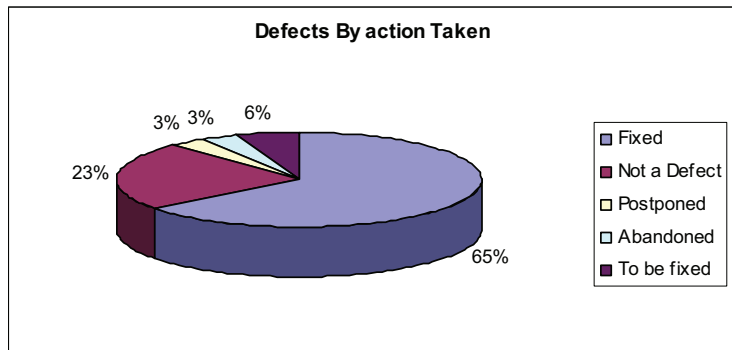
- Decide the audience
- Understand the expectation (problem areas/pain points where improvement is required)
- Identify whether it is derived/inferred information (metric) or direct (basic) metric
- Identify the value/benefit of each metric
- Identify benchmark/goals for each metric
- Understand the definition of each metric
- Categorize the metrics based on audience, type of testing
- Ensure that the data capturing method is identified for each metric
- Identify side effects (inaccurate data) for each metric and define the steps to correct them Communicate the benefits of each metric to the stakeholders
- Define a template for presenting the metrics (graphs or tabular formats)
- Automate the metrics process to minimize human efforts and errors

Key metrics for software testing

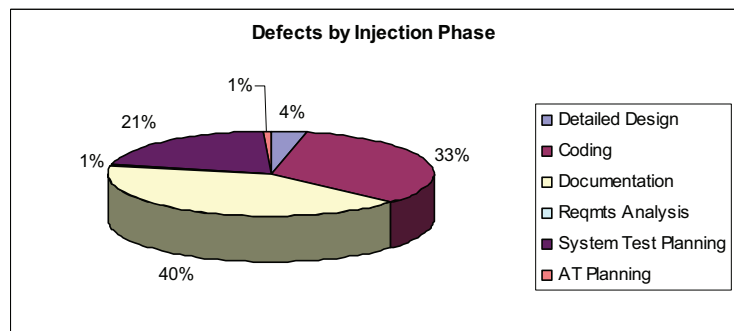
Test progress tracking metric: Track the cumulative test cases or test points — planned, attempted and successful, over the test execution period. Plot these three parameters on the Y axis and the X axis can be the timeline. The purpose of this metric is to closely monitor the testing progress. Any slippages from the plan can be clearly seen, thus helping to take corrective action.

Defect Metrics

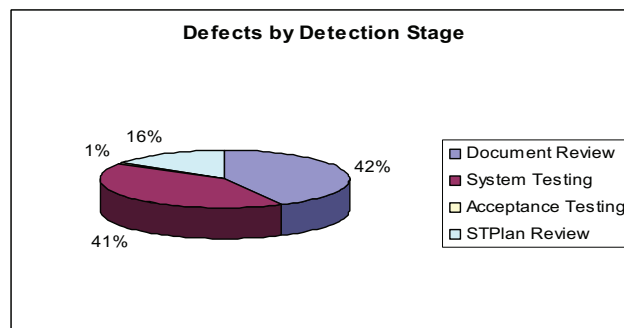
1. **Defects by action taken:** This metric is an indicator of the effectiveness of the testing process and test execution. If you see a good percentage of the defects marked as “not a defect”, it indicates that the understanding of the test engineer on the functionality is low, or that the requirements document is not clear.



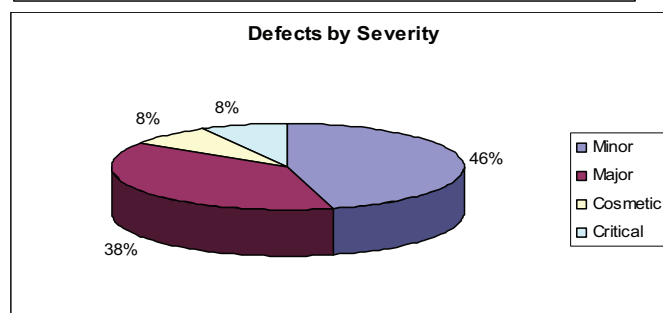
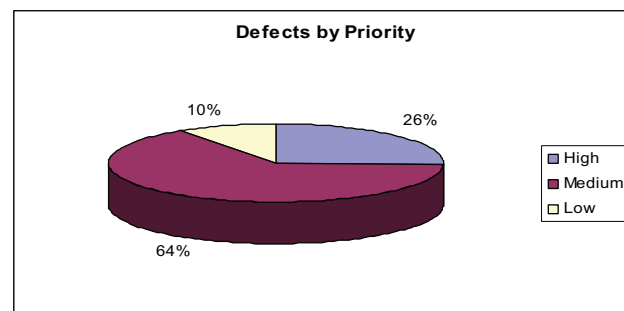
2. **Defects by injection phase:** This metric is an indicator of the most problematic phase in the software development



3. **Defects by detection phase:** This metric is an indicator of the most efficient phase of software development, in terms of finding and removing bugs.



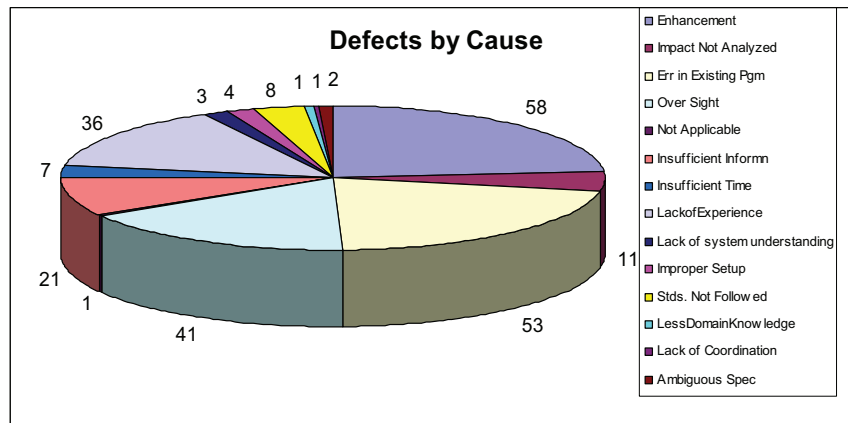
4. **Defects by priority:** It is very important to classify defects by priority and severity in order to get an understanding of the quality of the software. At times, a combination of the two classifications can be used to decide on the release of the software.



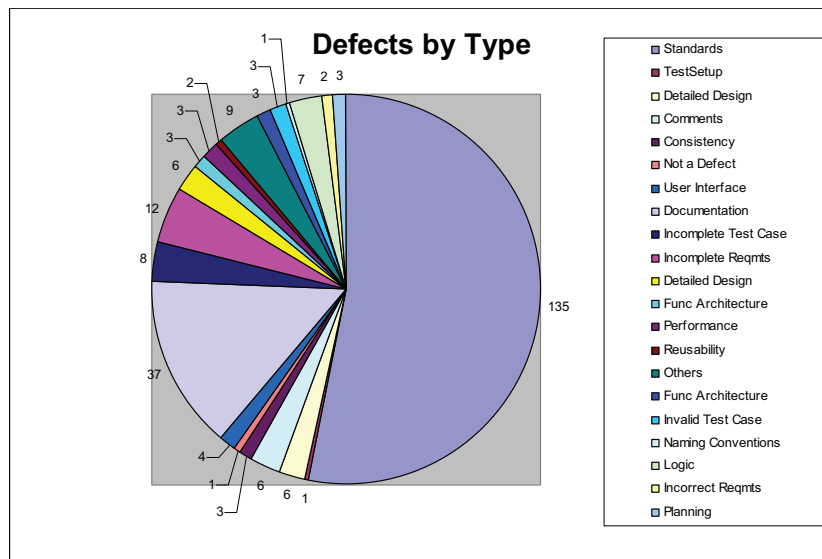
Release criteria: The matrix shown below can be used to define the release criteria of the software. If a defect falls in the shaded region, the software should not be released or will need a good reason for the defect to be waived off.

Defect Severity >	Critical	Major	Cosmetic	Minor
Priority				
High				
Medium				
Low				

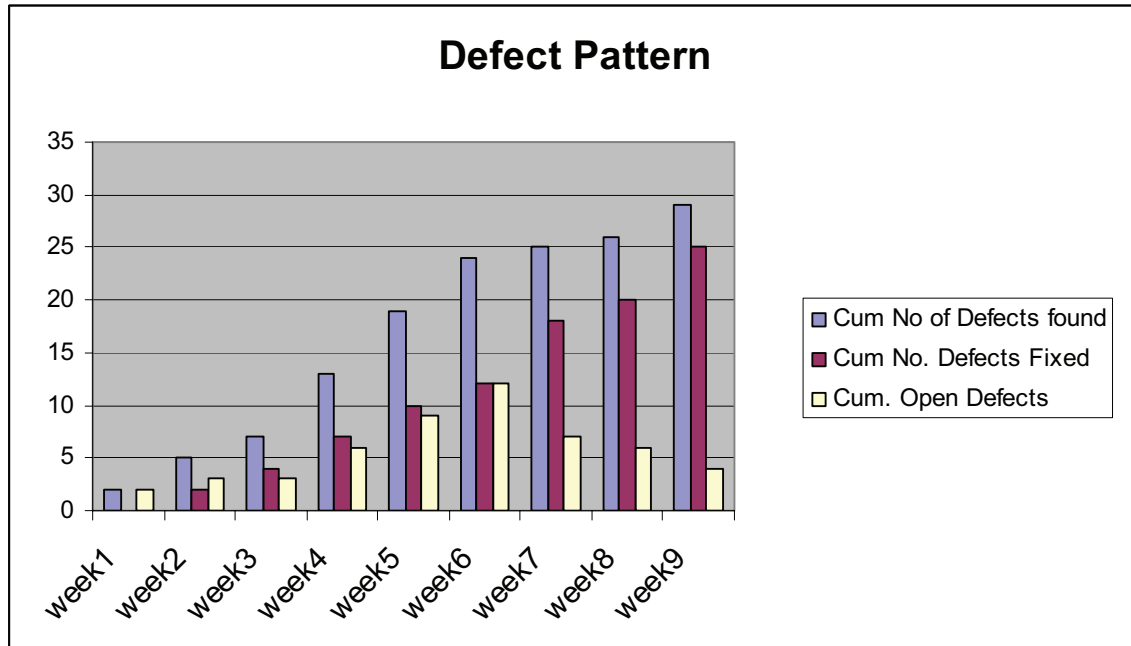
Defect by cause: This metric will help the development team and the test teams to focus on areas for improvement. Take the graphic shown below for instance, it indicates that 41 defects were due to low domain knowledge. As a project manager, you would like to focus on improving the domain skills of your team.



Defect by type: This metric can be a good pointer to areas for improvement. The graphic below shows the review defects and indicates that this project needs to focus more on following standards.



Defect pattern: This metric shows the pattern in which defects are found. Cumulative number of defects found in a week can be plotted against the weeks left to end the test execution cycle. To extend this further, the cumulative number of defects fixed in each week can also be plotted on the Y axis to give an idea on the open defects in a week.



References:

- Stephen H. Kan, [Metrics and Models in Software Quality Engineering](#), Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 2002
- Paul Goodman, 1993, [Practical Implementation of Software Metrics](#), McGraw Hill, London.
- G. Gordon Schulmeyer, James I. McManus, [Handbook of Software Quality Assurance, 3rd Edition](#), Prentice Hall PTR, Upper Saddle River, NJ, 1998.

About the Author

Ramesh Pusala is a Senior Test Manager at Infosys. He has 10 years of IT experience encompassing software testing, development, configuration management, customer and production support, quality, etc. He is a DFSS Six Sigma Black Belt from General Electric.



For more information, contact askus@infosys.com

About Infosys

Many of the world's most successful organizations rely on Infosys to deliver measurable business value. Infosys provides business consulting, technology, engineering and outsourcing services to help clients in over 30 countries build tomorrow's enterprise.

For more information about Infosys (NASDAQ:INFY), visit www.infosys.com.