Volume 2, No. 5, May 2011

Journal of Global Research in Computer Science

ISSN-2229-371X

REVIEW ARTICLE

Available Online at www.jgrcs.info

APPLICATIONS OF SOFTWARE TESTING METRICS IN CONSTRUCTING MODELS OF THE SOFTWARE DEVELOPMENT PROCESS

Vikas Verma [1*] and Sona Malhotra [2]

[*1] M.Tech Scholar, CSE Department, U.I.E.T. Kurukshetra University, Kurukshetra, India
vik.ver86@gmail.com
[2]Assistant Professor, CSE Department, U.I.E.T. Kurukshetra University, Kurukshetra, India

Abstract: The various software testing metrics and models and to meet out the objectives this paper is confined to the Software Testing companies. The developers of software companies who are working as software testers participated in the study. The metrics relevant to process improvement can be effectively identified and tailored to the organization and its goals and to ensure consistency and completeness, measurement provides the most appropriate information. Metrics are the most important responsibility of the Test Team. Metrics allow for deeper understanding of the performance of the application and its behaviour. This paper introduces the most commonly used software metrics proposed and reviews their use in constructing models of the software development process.

Keywords: Project Metrics, Process Metrics, Organisation Metrics**,** Statistical analysis, planning.

INTRODUCTION

Software Testing Metrics are quantifiable measures that could be used to measure different characteristics of a software system or the software development process. Test metrics are known as 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. On the basis of interpretation of the defined metrics, areas for process improvement can be identified. Measurement activities must be planned carefully because they require significant efforts to implement, and returns are realized only over a period of time. [1,2]

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.
- Provide a means for control/status reporting.
- Identifies risk areas that require more testing.
- Quickly identifies and help resolve potential problems and identifies areas of improvement.
- Testing effectiveness and efficiency are provided by the test metrics.[10]

IMPORTANT FACTORS OF TESTING METRICS

- Only that data is collected that is actually needed to make decisions or to alter strategy.
- Decisions are not based solely on data that is a variable or can be manipulated.
- Statistical analysis is used to get a better understanding of data. Difficult metrics data must be analyzed carefully.
- One of the key inputs to the metric 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 fields that decide per defect in the defect tracking system and then generate customizable reports.
- Metrics should not decide on the basis of ease of data collection rather than their importance to stakeholders. Metrics that are not of interest to stakeholders must be avoided.
- Inaccurate data must be avoided and complex data must be handled carefully. Proper benchmarks must be decided for the program.[9]

SOFTWARE TESTING METRICS LIFE CYCLE

The process of setting up testing metrics involves:

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

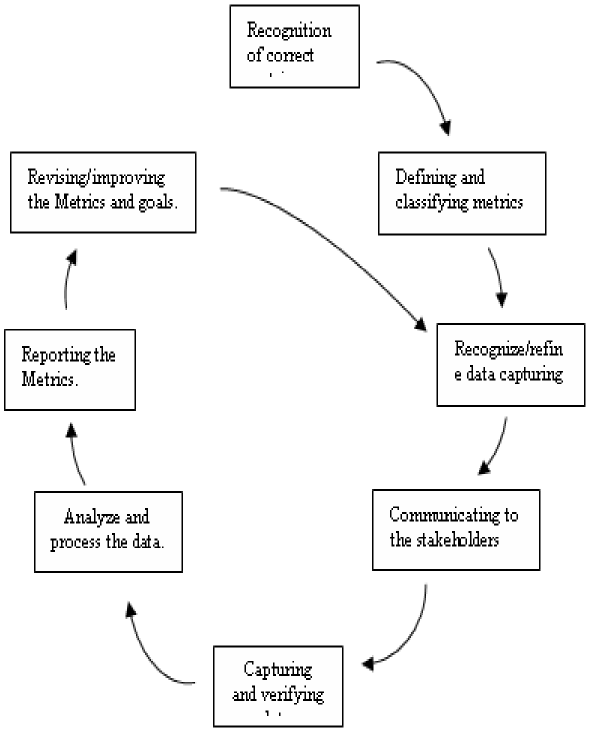
Recognition of correct
Defining and classifying metrics
Recognize/refine data capturing
Communicating to the stakeholders
Capturing and verifying
Analyze and process the data.
Reporting the Metrics.
Revising/improving the Metrics and goals.

Figure 1: Software Testing Metrics Life Cycle

Types of Testing Metrics

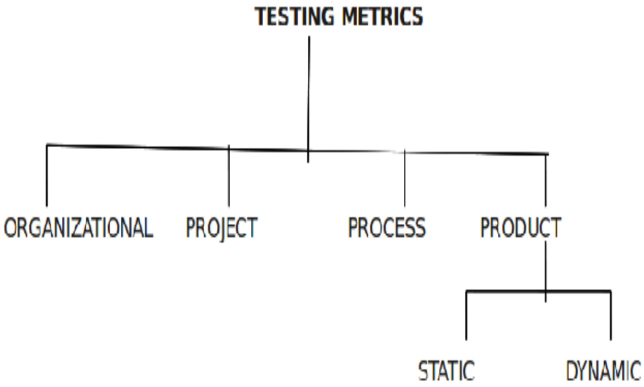

Figure 2: Classification of Testing Metrics

Organisation Metrics: Metrics at the level of organization are useful in overall project planning and management. Some of these metrics are obtained by aggregating compatible metrics across multiple projects. An organizational level test plan needs to incorporate metrics to make the testing activities visible available to process improvements.[3]

Project Metrics: Project test metrics relates to specific project. These are useful in the monitoring and control of a specific project. The ratio of actual to planned system test effort is one project metric. Test effort could be measured in terms of tester-man-months. At the start of the system test phase, for example, the project manager estimates the total system test effort. The ratio of actual to estimated effort is zero prior to the system test phase. This ratio builds up over time. Tracking the ratio assists the project manager in allocating testing resources.[4]

Process Metrics: Every project use some test processes. The big-bang approach is one process sometimes used in relatively small single-person projects. Several other well organized processes exist. The goal of project testing metrics is to access the goodness of the process.[5]

- Effort required in the process.
- Time to produce the product.
- Effectiveness of defect removal during development.
- Number of defects found during testing
- Maturity of process.

Product metrics

Product metrics relates to specific product such as compiler for programming language. These are useful in making decisions related to the product. Product metrics describe the characteristics of the product such as size, complexity, design features, performance, and quality level. Product metrics characterize some aspect of the structure of a software product, such as a requirements specification, a design, or source code. They are also commonly known as complexity metrics. Product complexity related metrics abound. Two types of product metrics are:[6]

1. Cyclomatic Complexity
2. Halstead Metrics for Software Testing.

Static & Dynamic Metrics

Static metrics are those computed without having to execute the product. Number of testable entities in an application is an example of a static product metric. Dynamic metrics require code execution. For example, the number of testable entities actually covered by a test suite is a dynamic product metric. One could apply the notions of static and dynamic metrics to organization and project. For example, the average number of testers working on a project is a static project metric. Number of defects remaining to be fixed could be treated as dynamic metric as it can be computed accurately only after a code change has been made and the product retested.[7,8,11]

CONCLUSION

It is easy to understand and applicable on the software, so favourable among software professionals. It is most prevalent and important testing metrics used in organizations. The least used testing metrics found in empirical investigation is Halstead metrics. It is complex measure to calculate and time consuming; organizations do not prefer it as very less testers have knowledge about it. Effective management of any process requires quantification, measurement, and modeling. Software metrics provide a quantitative basis for the development and validation of models of the software development process. Testing Metrics are used to improve software productivity and quality. This paper introduces the most commonly used software metrics proposed and reviews their use in constructing models of the software development process. Although current metrics and models are certainly inadequate, a number of organizations are achieving promising results through their use. Software Testing Metrics have rarely been used in any regular, methodical fashion.

REFERENCES

[1] Emam K. El, “A Methodology for Validating Software Product Metrics”, Ottawa, Ontario, Canada, National Research Council of Canada, June 2000.

[2] El-Far Ibrahim K. and Whittaker James A., “Model-based Software Testing” Encyclopedia on Software Engineering, Wiley, 2001, pages 1-22.

[3] Kaner Cem, Senior Member, IEEE, and Walter P. Bond, “Software Engineering Metrics: What Do They Measure and How Do We Know?” 10TH INTERNATIONAL SOFTWARE METRICS SYMPOSIUM, 2004.

[4] Li Fu-Shiau, Ma Wei-Ming, Chao Architect, “Architecture Centric Approach to Enhance Software Testing Management”, Eighth International Conference on intelligent Systems Design and Applications, pages 654-659.

[5] Lu Yongzhong, Song Simeng Da, Junli Jun Yang, Nie Songlin, “Research on a Behavior-Based Software Test Process Model”, International Symposium on Computer Science and Computational Technology, 2008, Pages 114- 117.

[6] Nagappan N., Williams L., Vouk M., Osborne, “Early Estimation of Software Quality Using In-Process Testing Metrics: A Controlled Case Study”, Third Software Quality Workshop, 2005, pages 46-52.

[7] Paul Raymond A., Kunii Tosiyasu L., Shinagawa Yoshihisa, and Khan Muhammad F., “Software Metrics Knowledge and Databases for Project Management” IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL.11, NO.1, JANUARY/FEBRUARY 1999, pages 255-264.

[8] Ripasi Tabor, “Software Testing- State of the Art and Current Research Challenges” 5th International Symposium on Applied Computational Intelligence and informatics, pages 47-50, May 28-29 2009.

[9] Schneidewind Norman F., “Measuring and Evaluating Maintenance Process using Reliability, Risk and Test Metrics”, IEEE Explore, 1989, pages 232-239.

[10] Wasif Afzal and Richard Torkar, “Incorporating Metrics in an Organizational Test Strategy” IEEE International Conference on Software Testing Verification and Validation Workshop, 2008.

[11] Aggarwal K. K. & Singh Yogesh, “Software Engineering”, New Age International, New Delhi, 2005.