A Management Overview of Software Metrics and Quality

by Susan H. Johnston and Frederick G. Kohun

Note: This article is offered as a topic of general interest for M Computing *readers.-Editors*

Abstract

An important issue within the M community has been how to convince potential clients of M's performance and productivity benefits. Relevant software measurements or *metrics* focusing on quality and/or productivity can facilitate and help to quantify objectively the advantages of M. Aligning a quality metrics program to support the overall business mission may strengthen not only the information systems aspect of an organization, but also the overall business by focusing its resources. This article gives a brief overview of quality issues, metrics initiatives, management issues, various models of software metric quality initiatives, and types of metrics potentially applicable to an M development environment. A management perspective is the context with the important caveat that metrics initiatives are situation-dependent (i.e., project, application, department, organization) and management-driven, and are, for the most part, idiosyncratic and nontransferable. The better a manager knows what needs to be measured, the more effective the metrics are as a tool.

Introduction

M represents a development environment that includes a programming language, a job control language, and a database access and management system. Although M demonstrates broadly applicable productivity advantages over other development tools, its initial market was primarily in the healthcare environment. A software metrics and quality initiative can be a way of improving the M software development process within an organization. The quantitative results of such an initiative can be a significant factor in marketing M to potential clients outside of health care by emphasizing the quality of M software. There is evidence that using M allows a firm to serve greater numbers of clients with fewer employees.[1] Given this productivity advantage, a quality initiative with supportive metrics would be an effective marketing tool.

An example of a particular type of metric is the measure of defects per thousand lines of code (KLOC). The U.S. productivity quality measurement is four defects per KLOC versus Japan's and the U.K.'s defect rate of two defects per KLOC. Additionally, U.S. organizations rate lower in readiness to assimilate and use new technology.[2] With an increasingly globalized economy, U.S. software organizations could use these metrics to consider the impact of the potential entry of other developing countries into the software marketplace. Current economic, political, and technological factors within developing nations and formerly communist countries increase the potential for improved competitiveness or collaboration.[3,4]

A metrics program could be useful in organizations developing software to sell or to use in-house, or purchasing off-the-shelf software with little or no customization. The type of software-measurement system implemented depends on the type of business: Select a system that supports the overall business by providing measures that indicate business value.

Quality Issues

Of prime importance to an organization and its customers is the quality of its products or services and how these adhere to specifications, established performance criteria, or customer requirements. Measuring quality must include customer perceptions and expectations regarding product performance, i.e., the life of the product and the functions it performs versus the expected performance.

Quality performance is the interaction of the quality of the design, the quality of conformance to the design, and prevention measures against defects (faults or failures).[5] The performance standard for quality is zero defects and the actual measurement of quality is the price of nonconformance to the quality design.[6]

An organization's quality costs are actually performance measurements or metrics of the degree of excellence with which the organization produces its products or services. Quality costs are the cost of the performance needed to achieve excellence, which is to meet or exceed specifications or standards set by the firm or by the customers.

In order to facilitate quality improvements with metrics, an organization should develop a quality policy and widely publicize this policy throughout the organization. If economically feasible, a team devoted to quality improvement should be formed with employees from all departments. This team can educate employees about quality issues and instituting quality metrics. That each employee accepts the importance of individual contributions to a quality program is essential to a successful program. All these actions combined will produce continuous quality improvement.

Metrics Initiative

The objective of a software metrics quality initiative is to measure both software process and products while supporting the goals, mission, and objectives of the organization itself. This can be accomplished by starting with a business model and identifying software measurements that can be used within or in support of business measurements to support overall business goals.

A business model is needed that combines financial and nonfinancial performance measurements, so that costs of quality can be included in reports to management. Performance and accountability for results must be measured within the model.[7] The model could start with the formal organization chart but decision making does not always follow the formal organization chart. The information systems staff should be able to suggest where systems can support the decision-making process, such as determining what types of measurements are needed and how to measure for software quality and overall organizational quality. Regular assessments of measured results will ensure the usefulness and relevance of the measurements.

A limitation of traditional accounting indicators is the delay in what they report. Traditional accounting measures are the result of decisions or the consequences of them. They are financial measurements which provide some guidance for the future, but do not indicate the next steps to be taken. Performance metrics, however, can be used to support quality, customer satisfaction, and innovation.[8] Software metrics help to pinpoint areas of the software process or software products that may need improvement.

Roles of Management and Employees

One essential element in the success of any software-measurement program is the commitment of top management, which should be communicated to employees both formally and informally. But quality should also become part of the information systems department culture to the point that it fosters employee suggestions for innovations. Management support and employee involvement (especially in creating the measurement program) are both critical success factors.

Cost/Benefit and User Satisfaction

What is the cost of the software quality-measurement process versus the benefits? The time invested in measuring for defects (software metrics) must be compared against the benefits derived from identifying and correcting the defects. One organization reduced its bidding rate 40 percent over ten years by instituting a software quality and productivity program. Obviously some measurements can be used to support the short-term goals of an organization, but as this example shows, improvement may be a long-term iterative process in which measurements will be initially defined and then redefined. Although a base set of measurements can be instituted, most organizations will develop unique metrics for their particular purposes.

There are also quality issues of user satisfaction. Most organizations have internal and external users or customers. User-satisfaction surveys traditionally focus on external users, but internal customers must also be considered when using metrics; if internal users' needs are met, an organization can better serve its external users. Can the software or system be considered a success if neither are satisfied?

When measuring cost against benefits and user satisfaction, efficiency, cycle time, and defects are important factors.[9] Questions to consider include: How active is the organization's quality assurance group? How efficient is the organization at defect removal? Common sense will dictate the best approach for addressing quality issues, especially in small organizations that might not have the staff or resources for a separate quality assurance group.

Benchmarks

Benchmarks help evaluate such factors as the length of time between software quality improvement and increased customer satisfaction, and the balance among cost of quality, customer satisfaction, and profitability. A baseline benchmark can be performed on the current state of the software development process, including the traditional quality-report elements of prevention, appraisal, internal failure and external failure.[10] Continual improvements in quality activities for the software process can be measured against this baseline.[11]

Prevention costs involve quality training, software reliability engineering, and prototype or pilot studies, and focus on system analysis, design, and development. Appraisal costs involve inspections and testing, especially reliability testing and defect measurements. Internal failure costs—those occurring prior to product distribution—involve rework, repair, and possibly cancellation of a project. External failure costs —those occurring after product distribution—involve customer complaints and dissatisfaction, repairs or replacement of software, and possible warranty costs. All of these costs should be included in a cost-of-quality report.[12]

Various issues and questions should be considered during the benchmark. Is there a system in place for defect-tracking or a quality-tracking system? Is the system manual or automated? Identification of where the error or defect was introduced within the systems development life cycle and where it was detected should be the type of functionality included in the defect tracking system.[13] The quality tracking system should include the length of time needed to resolve the defect or error. Identifying the applications with the greatest number of problems should be a priority. Cost issues per defect should be considered, if they can be identified. What types of inspections or walkthroughs are conducted? How is current software scored or measured? Scoring the programs that are taken out of production and not placing them back into production unless they meet or exceed the already achieved score would tend to significantly reduce the bugs introduced during maintenance procedures.[14]

Understanding the internal processes of an organization is key to ensuring that the tools implementing the metrics will conform to the process and report results accurately. A metrics baseline should include process modeling—modeling tasks, relationships, and tools used for current processes within an organization. A generic process model can be developed and "tuned" for specific projects within an organization.[15] Metrics can track a project at all stages, from the requirements phase through the analysis, design, coding, testing, and maintenance phases.

Metrics Examples

Two examples of metrics models are the Software Engineering Institute's Capabilities Maturity Model (CMM) and the U.S. Department of Defense's Software Readiness Growth Model (DSRGM). The CMM defines five levels of organizational software-process maturity: initial, repeatable, defined, managed, and optimizing. Companies that focus more on front-end activities such as design and analysis, and also on testing, rate higher in maturity ratings. The DSRGM is a methodology for measuring risk associated with software acquisition. While the CMM evaluates an organization's maturity in the software-development process, the DSRGM measures the development process and the products of that process. The DSRGM is domain-independent and can be used in both the private and public sectors.[16] An organization can tailor the DSRGM to its particular circumstances, creating specific and therefore more productive measurements.

Continuous Process

A measurement system can identify problems as well as success factors within system applications. It can be the basis for a continuous improvement process with unique methods for developing and maintaining software.[17] Objectives should be set, analyzed, and refined until a workable set of process objectives is obtained for the organization, an ongoing process. In the long term, productivity as well as quality should increase, software costs should decrease, and improvements in scheduled software deliveries should be noted.

Just as business issues evolve over time, so should software metrics be adapted to support the strategic issues of an organization. There are many stories about reports that are generated for years without ever being used. Metrics should be monitored to ensure that they are being used and, even more importantly, that they are actually measuring and reporting as intended.

Types of Metrics

Numerous types of metrics are currently in use throughout the world, each with a unique application. Settling upon a particular metric should take into account the differences between metrics for established software and start-up software, especially if a new language or new paradigm, such as object-oriented, is used. The M language, for example, has some unique properties such as abbreviated commands and the indirection operator which can affect metrics.[18]

Following are examples of types of metrics that an organization might consider when developing a software quality initiative:

Metrics targeted toward defects:

- Number of defects per program or module;
- Frequency of occurrence of defects;
- Origin of defects (stage of development process);
- Reason for defects—not to place blame, but to correct the cause;
- Type of defects (accumulated also);
- Defects requiring rewrites; and
- Severity of defects and effects on customer (internal or external).

Metrics targeted toward time or usage:

- Actual performance compared with estimated performance;

- Schedule adherence;
- Required time to fix defects (turnaround time);
- Actual cost of defects—should trace ripple effect throughout organization in order to accumulate total cost of defects;
- Resource utilization to fix defects; and
- Effort metrics—labor time.

For reliability:

- Mean time to failure;
- Target modules with defect count or density;
- Target code that has been tested;
- Number of modifications to modules;
- Assess complexity of modules; and
- Operational profiles (usage pattern of software which can affect reliability).[19]

For productivity:

- Lines of code (if comparing—language used should be the same); and
- Function points.[20]

For reuse:

- Reuse metrics measure the amount of code within a system that has been reused from another application or a reuse library. Because reuse of existing code can significantly reduce the amount of coding necessary for an application, reuse should be a requirement rather than an option.

For customer satisfaction:

- Use of surveys for both external and internal customers;
- Customer defect reports (target code); and
- Assess defect type.

Management Issues

Software problems such as the failure to meet deadlines or large defect numbers are sometimes viewed as management problems.[21] To improve a software-development process, measurements of the current situation are needed as a baseline benchmark. It makes good economic sense for information systems managers to consider instituting a software-quality initiative based on relevant and useful software metrics.

The following steps may be useful when implementing a quality-metrics project:

1. Start the process on a small scale.
2. Publicize the organizational metrics program and its rationale.
3. Secure visible management backing and, more importantly, compliance in decision making.
4. Timely feedback is essential.
5. Change what does not work, an iterative process.
6. Dedicate a portion of the staff to metrics, if possible.
7. Provide positive feedback to individual employees.
8. Keep metrics visible in meetings, presentations, and publications.
9. Use software-oriented metrics in quality measurements throughout the organization.
10. Address employees' fear of metrics by focusing on quality improvement, not individual performance. Ease of use of metrics tools should be emphasized. Many measurement programs have failed due to lack of employee acceptance of, involvement with, and use of the tools. Resistance to a metrics project can be open, such as questioning its cost or apparent benefits, or hidden, which is more insidious and damaging, and can come from either staff or management.

Conclusion

Software metrics can be a useful way to promote M's benefits. Metrics can help to provide consistent and objective measurements for performance comparisons. They can be used as a marketing tool in the bidding process, to improve process quality and efficiency, and to assess and improve user satisfaction.

The best approach to a software measurement program is a common-sense approach—analyzing in a different way data that are already being captured. Also, emphasis on quality improvement is paramount for successfully implementing and continuing a software quality-improvement program.

Finally, metrics programs are situation-dependent. There is no universal formula, statistical application, or audit program that is applicable to every project in every organization. An organization must know what it wants to measure and why it wants to measure it for any metrics initiative to be ongoing and successful.

Research for this article was partially funded by the Department of Veterans Affairs, Washington Information Service Center.

Susan Johnston was the 1993 winner of the Michael S. Distaso Award and was a member of the 1994 MTA Annual Meeting program committee.

Frederick Kohun, Ph.D., has been head of the Computer and Information Systems Department at Robert Morris College (Pittsburgh) for the past five years.

Endnotes

1. D. Gall, "Dollars and Sense: How M Compares with COBOL and Relational Database Software," *M Computing* 1:5 (November 1993).
2. H.A. Rubin, Ph.D., "Software Process Maturity: Measuring Its Impact on Productivity and Quality," in *Proceedings, First International Software Metrics Symposium* (Baltimore, Maryland: sponsored by IEEE Computer Society Technical Committee on Software Engineering, May 1993).
3. L. Press, "Software Export from Developing Nations," *Computer* 26:12 (December 1993).
4. E. Yourdon, *Decline and Fall of the American Programmer* (Englewood Cliffs, NJ: Yourdon Press, 1993).
5. L. Augenblick, "Barriers to Quality," *Personnel Journal* 69:30 (May 1990).
6. Augenblick.
7. R.G. Eccles and P.J. Pyburn, "Creating a Comprehensive System to Measure Performance," *Management Accounting* 74:4 (October 1992).
8. Eccles.
9. L.R. White, "Evaluating and Measuring Outsource Agreements," in *Proceedings, Fourth International Conference on Applications of Software Measurement* (Orlando, Florida: sponsored jointly by the American Society for Quality Control, Software Division; Centre for Software Reliability; and Software Quality Engineering, November 1993).
10. R.W. Hilton, *Managerial Accounting* (New York, NY: McGraw-Hill, Inc., 1991).
11. R.K. Youde, CMA, "Cost-of-Quality Reporting: How We See It," *Management Accounting* 73:7 (January 1992).
12. Hilton.
13. Rubin.
14. White.
15. Rubin.
16. D.R. Castellano, P.E. Janusz, and S.L. Tolochko, "Readiness Growth Model: A Quantitative Analysis of Software Risk," (Technical Report ARPAD-TR-93003, Department of Defense Publication, U.S. Army Armament Research, Development and Engineering Center, October 1, 1993).
17. W.S. Humphrey, *Managing the Software Process* (Reading, MA: Addison-Wesley Publishing Co., 1989).
18. E.F. Dindal, "Software Metrics in the MUMPS Environment," *MUG Quarterly* 21:3 (June 1991).
19. N.B. Harrison and B.D. Juhlin, Ph.D., "Customer Driven Reliability Metrics," in *Proceedings, Fourth International Conference on Applications of Software Measurement* (Orlando, Florida: sponsored jointly by the American Society for Quality Control, Software Division; Centre for Software Reliability; and Software Quality Engineering, November 1993).
20. C. Jones, "Sick Software," *Computerworld* 27:50 (December 13, 1993).
21. Jones.