Survey on Impact of Software Metrics on Software Quality

Mrinal Singh Rawat[1]
Department of Computer Science
MGM's COET,
Noida, India

Arpita Mittal[2]
Department of Computer Science
IIMT,
Merrut, India

Sanjay Kumar Dubey[3]
Department of Computer Science
Amity University,
Noida, India

***Abstract*—Software metrics provide a quantitative basis for planning and predicting software development processes. Therefore the quality of software can be controlled and improved easily. Quality in fact aids higher productivity, which has brought software metrics to the forefront. This research paper focuses on different views on software quality. Moreover, many metrics and models have been developed; promoted and utilized resulting in remarkable successes. This paper examines the realm of software engineering to see why software metrics are needed and also reviews their contribution to software quality and reliability. Results can be improved further as we acquire additional experience with variety of software metrics. These experiences can yield tremendous benefits and betterment in quality and reliability.**

Keywords- Software metrics; Software quality; Software reliability; Lines of code; Function points; object oriented metrics.

I. Introduction

Software metrics are valuable entity in the entire software life cycle. They provide measurement for the software development, including software requirement documents, designs, programs and tests. Rapid developments of large scaled software have evolved complexity that makes the quality difficult to control. The successful execution of the control over software quality requires software metrics. The concepts of software metrics are coherent, understandable and well established, and many metrics related to the product quality have been developed and used.

It is essential to introduce definition of software metrics. Software metrics provides measurement of the software product and the process of software production. In this paper, the software product should be seen as an abstract object that begins from an initial statement of requirement to a finished software product, including source and object code and the several forms of documentation exhibited during the various stages of its development.

Good metrics should enable the development of models that are efficient of predicting process or product spectrum. Thus, optimal metrics should be: [1]

- Simple, precisely definable—so that it is clear how the metric can be evaluated;
- Objective, to the greatest extent possible;
- Easily obtainable (i.e., at reasonable cost);
- Valid—the metric should measure what it is intended to measure; and
- Robust—relatively insensitive to (intuitively) insignificant changes in the process or product.

II. Overview of Software Metrics

A. *Classification of Software Metrics*

There are three types of software metrics: process metrics, project metrics and product metrics. [3]

1) Process Metrics:

Process metrics highlights the process of software development. It mainly aims at process duration, cost incurred and type of methodology used. Process metrics can be used to augment software development and maintenance. Examples include the efficacy of defect removal during development, the patterning of testing defect arrival, and the response time of the fix process.

2) Project Metrics:

Project metrics are used to monitor project situation and status. Project metrics preclude the problems or potential risks by calibrating the project and help to optimize the software development plan. Project metrics describe the project characteristics and execution. Examples include the number of software developers, the staffing pattern over the life cycle of the software, cost, schedule, and productivity. [4]

3) Product Metrics:

Product metrics describe the attributes of the software product at any phase of its development. Product metrics may measure the size of the program, complexity of the software design, performance, portability, maintainability, and product scale. Product metrics are used to presume and invent the quality of the product. Product metrics are used to measure the medium or the final product.

We can find more efficient ways of improving software project, product and process management.

B. *Mathematical Analysis*

A metric has a very explicit meaning in mathematical analysis .It is a rule used to determine distance between two points. More formally, a metric is a function 'd' defined on pairs of objects p and q such that d (p, q) expresses the distance between p and q. Such metrics must satisfy certain properties: [11]

d (p,p) = 0 for all p : that is, the distance from point p to itself is zero;

d (p, q) = m (q, p) for all p and q: that is, the distance from p to q is similar to the distance from q to p;

d (p, r) ≤ d (p, q)+d (q, r) for all p, q and r: that is, the distance from p to r is no larger than the distance measured by stopping through an intermediate point.

A prediction system comprise of a mathematical model along with a set of prediction processes for determining unknown parameters and depicting the results. The model should not be complicated for use. Suppose we want to predict the number of pages, P that will print out as a source code program, so that we can bring sufficient paper or calculate the time the program will take for printing. We can use a simple model,

$$P = x/a \tag{1}$$

Where x is a variable, acts as a measure i.e. length of source code program in LOC (line of code), and ‘a’ is a constant that represents the average number of lines per page. There are number of models to determine effort estimation; from analogy based estimation to parametric models. A generic model can be used to estimate effort predication.

$$E = aS^{b} \tag{2}$$

Where a and b are constants. E is effort in person-months. S is the size of source code in Line of code.

III. Importance of Software Quality

In recent times the importance of software quality has come to light when random errors on a say a telephone bill, or on a bank statement were randomly attributed to a bug in the “computer code” or using the ignorant adage of “the computer does things” without making an effort to undermine the cause of the problem or even separating it by hardware or software. The problem arises when “computer errors” creep into highly critical aspects of our lives involving situations where a small error can lead to a cataclysmic chain of events. Bearing all this in mind, the importance of enforcing software quality in computer practices has become highly important. Seeing the penetration of computer code into everyday objects like washing machines, automobiles, refrigerators, toys and even things like the mars rover, any system be it a large one or a small system running embedded IC technology, ensuring the highest levels of software quality is paramount.

However, that brings us to the next logical question, how do we assess the quality of something intangible like software quality? This is a highly subjective question whose answer will vary according to the situation. For example, a small word processing error in a student’s assignment will not be a huge issue. But a slight code error in a space shuttle’s guidance computer might be mission critical and endanger human lives.

Hence in terms of software quality, it is imperative that we understand that it’s impossible to have a boilerplate definition or meaning of software quality. The definition will differ according to factors like quality of products and business. Also crucial is the proper setting of goals as well as proactive monitoring of quality factors and goals making sure that that the goals set are resolved and completed in the given timelines and specifics. Views on Software Quality

Software quality, as stated earlier, depends on a number of factors. Also as theorized by David & Garwin, quality is a complex as well as multifaceted concept, which can be viewed according to different points of view as follows

1) *User View*

The user viewpoint of software quality tends to be a lot more concrete and can be highly subjective depending upon the user. This view evaluates the software product against the user’s needs. In certain types of software products like reliability performance modelling and operational products, the user is monitored according to how they use the product.

2) *Manufacturing View*

This viewpoint looks at the production aspect of the software product. It basically stresses on enforcing building the product without any defects and getting it right the first time rather than subsequently making a defective product and spending valuable project time and more importantly costs ironing out the defects or bugs at a later stage. Being process based, this viewpoint focuses on conformity to the process, which will eventually lead to a better product.

Models such as ISO 9001 as well as the Capability Maturity Model do encompass this viewpoint that stress on following the process as opposed to going by specification. However, that being said, the theory that following the best and high quality manufacturing process will automatically lead to a better product cannot be inferred. The critic’s viewpoint is that following an optimized and high quality product manufacturing method can also lead to the standardization of a product making it more of a commodity rather than a standout product.

That being said, there have been a lot of industry example where the philosophy of “doing it right” the first time been profitable. Also both the models CMM as well as the ISO, indirectly do imply by following the principle of “Documenting what you do and doing what you say” helps in improving the product quality.

3) *Product View*

The product viewpoint looks at the internal features as well as the characteristics of the product. The idea behind this viewpoint is that in case a product is sound in terms of the features and functionality it offers, and then it will also be favourable when viewed from a user viewpoint in terms of software quality. The idea is that controlling the internal product quality indicators will influence positively the external product behaviour (user quality) There are models trying to link both the views of software quality but more work is needed is this area.

4) *Value based view*

The value-based view becomes important when there are lots of contrasting views, which are held by different departments in an organization. For example, the marketing department generally take a user view and the technical department will generally take a product-based view. Though initially these contrasting viewpoints help to develop a 360-

degree product with the different viewpoints complementing each other, the later stages of the software product development might have issues

The issues arise when there might be a set of change proposed to a certain view that can end up throwing a conflict in the other view. For example, say the marketing department (user view) want changes to the user interface that are not technically feasible (product view).

This is where a value-based view comes into play helping resolve such conflicts so that the software product is not delayed indefinitely. The value-based viewpoint looks the conflict with a cost to benefit angle. It help in resolving such issues by looking at the issue in relation to terms like costing, constraints, resources, time. Using this viewpoint, it's possible to resolve interview conflicts helping to keep the software product on track and within initial cost and timeline estimates.

IV. Case Study on Software Quality

The Boeing 777 project – Boeing with its 777 airplane project was a giant leap forward in the direction of Software quality and is compelling case and point in the importance in reinforcing strong software quality management. With almost 2.5 million lines of code written for the new jetliner's state of the art avionics and other on board software, it was super critical to ensure best software quality practices and implementation. Complications like an extensive network of third party suppliers who would supply crucial components for the 777 made it a large challenge to ensure that deadlines are met without a compromise on software quality as a whole. [15]

Measures taken by Boeing – interestingly, at the beginning of the 777 project, since there was extensive vendor fragmentation, each vendor was using different measures and metrics to keep in track of software quality and measure the status of the work. As a result, this soon snowballed into a situation where due to non-standard practices being followed, it was extremely hard to understand the progress of the project as a whole. Therefore, around the 777 project's midway point, Boeing implemented measures, which called for uniformity in reporting as well as monitoring all variables related to the project status and software quality. A uniform use of metrics like came into effect which made the suppliers report around the simple metrics like test definition, resource utilization, test execution as well as detailed plans for the software coding and design.

As a result, since the reporting was uniform as well as the enforcement of these metrics was universal for the of Boeing's vendors, each vendor was now reporting on a bi weekly basis as which now contained information about completed code, testing as well as design. This not only lowered the effort on Boeing's part in consolidating the fragmented data (as was happening previously) but also allowed Boeing to adjust its own plans in sync with the vendor's estimates and hence keep the project on schedule.

Key Takeaways – Boeing realized early enough of the importance of enforcing a uniform set of metrics. Also vital learnings from Boeing's experience is that done properly, enforcing software quality in a project ensures that program risk points can be identified early which would allow a reasonable time to apply corrective measures without delaying a project indefinitely. Additional key points are the implementation of metrics allowed each project point to be having a check and balance so that the project flows smoothly without any major roadblocks. A good consequence of the metrics implementation was the streamlining and the regularity of communications between Boeing and its vendors, which was touted as being of equal importance to the metrics as well. Clear goals, milestones and constant monitoring of the key metrics around software design coding and testing made sure the 777 project was a success.

V. Comparison of Software Metrics- Strengths and Weaknesses

The software industry does not have standard metric and measurement practices. Most of the software metric has multiple definitions and ambiguous rules for counting. There are also important subject issues that do not have specific metrics, such as quantifying the volume or quality levels of databases, web sites and data warehouses. There is a lack of strong empirical data on software costs, schedules, effort, quality, and other tangible elements, which results in metric problems. [12]

A. Source Code Metrics

"Source lines of code" or SLOC was the first metric developed for quantifying the outcome of a software project. The divergent "lines of code" or LOC has similar meaning and is also widely acceptable. "Lines of code" could be defined either:

- A physical line of code.
- A logical line of code.

Physical lines of code are sets of coded instructions terminated by hitting the enter key of a keyboard. Physical lines of code and logical lines of code are almost identical for some languages, but for some languages there can be considerable differences. Generally, the difference between physical lines of code and logical lines of code is often excluded from the software metrics literature.

Strengths of physical lines of code (LOC) are:

- It is easy to measure.
- There is a scope for automation of counting.
- It is used in a verity of software project estimation tools.

Weaknesses of physical LOC are:

- It may include significant "dead code."
- It may include white spaces and comments.
- This metric is vague for software reuse.
- It does not function for a few "visual" languages.
- Direct conversion to function points is erroneous.
- It is inconsistent for direct conversion to logical statements.

Strengths of the logical LOC are:

- It omits dead code, blanks, and comments.
- Mathematical conversion of logical statements into function point metrics is possible.
- Logical LOC are used in many software project estimation tools.

Weaknesses of logical LOC are:

- It can be difficult to measure.
- These are not comprehensively automated.
- These are ambiguous for a number of "visual" languages.
- This metric is vague for software reuse.
- Direct conversion to the physical LOC metric may be erroneous.

"Measuring software productivity by lines of code is like measuring progress on an airplane by how much it weighs." – Bill Gates.

It is prudent to focus more on building expertise on Function Point Analysis and use it effectively.

B. *Function Point Metrics*

The function point analysis to measure software application is enumerated from analysis of the requirements and logical design of the application. Function Point count can be applied to Development projects, Enhancement projects, and existing applications as well. [13] There are five key elements of Function Point Analysis, which capture the functionality of the application. These are:

- External Inputs (EIs),
- External Outputs (EOs)
- External Inquiries (EQs)
- Internal Logical Files (ILFs) and External
- Interface Files (EIFs).

First three elements are of Transactional Function Types and last two are of Data Function Types. Function Point Analysis consists of performing the following steps:

- Determine the type of Function Point count
- Determine the application boundary
- Identify and rate transactional function types to calculate their contribution to the Unadjusted Function Point count (UFP)
- Identify and rate the data function types to calculate their contribution to the UFP
- Determine the Value Adjustment Factor (VAF) by using General System Characteristics (GSCs)

Finally, calculate the adjusted Function Point count

When we examine the patterns of strengths and weaknesses of function point metrics, we observe that for economic studies and for studies that include non-coding work such as specifications, function points are clearly superior to lines of code metrics. [12]

Strengths of function point metrics are:

- It stays stable regardless of programming languages used.
- It can compute non-coding activities such as documentation.
- It can measure non-coding defects in requirements and design.
- These are useful for software reuse analysis.
- Function points are used for object-oriented economic studies.
- These are supported by a lot of software cost estimating tools.
- Mathematical conversion of function points into logical code statements is very easy.

Weaknesses of function point metrics are:

- Function Point counting requires good deal of experience.
- Function point counting can be protracted and pricey.
- Function point counting automation is of indefinite accuracy.
- Function point counts are unreliable for those projects that are below 15 function points in size.
- Function point variant have no conversion rules to IFPUG function points.

C. *Object-Oriented Metrics*

In today's software development environment, Object-oriented analysis and design concepts are well known. Object-Oriented Analysis and Design of software provide many advantages such as reusability, decomposition of problem into easily understandable object and the aiding of future modifications. Object-oriented software development requires a diverse approach from more traditional functional decomposition and dataflow development methods. But the OOAD software development life cycle is not easier than the typical procedural approach. Therefore, it is necessary to provide dependable guidelines that one may follow to help ensure good OO programming practices and write reliable code. Object-Oriented programming metrics is an aspect to be considered. Metrics should be a set of standards against which one can measure the effectiveness of Object-Oriented Analysis techniques in the design of a system. [2]

Strengths of OO metrics are: [12]

- The OO metrics are psychologically attractive within the OO community.

- The OO metrics come out to be able to differentiate simple from complex OO projects.
- Weaknesses of OO metrics are:
- The OO metrics do not support studies outside of the OO paradigm.
- The OO metrics have not yet been applied to testing.
- The OO metrics have not yet been applied to maintenance.
- The OO metrics have no conversion rules to lines of code metrics.
- The OO metrics have no conversion rules to function point metrics.
- The OO metrics lack automation.
- The OO metrics are difficult to enumerate.
- Software project estimation tools do not support the OO metrics.

OO metrics are not linked to all other known software metrics. There are no conversion rules between the OO metrics and any other metrics, so it is complicated to perform alongside comparisons between OO projects and conservative projects using the currently available OO metrics.

VI. Future Scope

Looking at rising demand for the implementation and successful case studies of software quality, it is safe to conclude that in the coming years, software metric's importance will increase multifold as industry leaders like embrace newer and more stringent approaches to monitoring, improving as well as delivering better software quality in products as well as processes. A number of metrics are proposed and exercised for measuring the quality of a system before implementation. Future research directions include improvement in existing metrics based on the nature and magnitude of the problem statement. There is a scope for various tools to support software project development reducing time, effort and cost of the project in consistent manner.

VII. Summary and Conclusion

With the rapid advancement in software industries, software metrics have also developed fast. Software metrics become the basis of the software management and crucial to the accomplishment of software development. It can be anticipated that by using software metrics the overall rate of progress in software productivity and software quality will improve. If relative changes in productivity and quality can be determined and studied over time, then focus can be put upon an organization's strengths and weaknesses. Although people appreciate the significance of software metrics, the metrics field still needs to mature. Each of the key software metrics candidates has broken into many competing alternatives, often following national restrictions. There is no adequate international standard for any of the extensively used software metrics. Absence of firm theoretic background and the assurance of methods, software metrics are still young in comparison of other software theories.

References

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lips Software Metrics SEI Curriculum Module, SEI-CM-12-1.1, December 1988

[2] "Software Quality Metrics for Object Oriented System Environments", June 1995, National Aeronautics and Space Administration, Goddard Space Flight Center, Greenbelt Maryland

[3] Tu Honglei1, Sun Wei1, Zhang Yanan1, "The Research on Software Metrics and Software Complexity Metrics", International Forum on Computer Science-Technology and Applications, 2009.

[4] http://www.pearsonhighered.com/samplechapter/0201729156.pdf.

[5] http://www.wohlin.eu/Articles/ICGSE11.pdf.

[6] Barbara Kitchenham, Shari Lawrence," Quality: The Elusive Target", IEEE Software-Vol. 13, No. 1: JANUARY 1996, pp. 12-21.

[7] Henrike Barkmann, Rudiger Lincke and Welf L owe, "Quantitative Evaluation of Software Quality Metrics in Open-Source Projects".

[8] Dindin Wahyudin, Alexander Schatten, Dietmar Winkler, A Min Tjoa, Stefan Biff,"Defect Prediction using Combined Product and Project Metrics ", March 2008.

[9] Nachiappan Nagappan, Brendan Murphy, and Victor Basili, "The Influence of Organizational Structure On Software Quality: An Empirical Case Study", January 2008.

[10] David I. Heimann,"Implementing Software Metrics at a Telecommunications Company" – A Case Study,2oo4

[11] Rakesh.L , Dr.Manoranjan Kumar Singh ,and Dr.Gunaseelan Devaraj ,"Software Metrics: Some degree of Software Measuremen and Analysis ",(IJCSIS) International Journal of Computer Science and Information Security, Vol. 8, No. 2, 2010

[12] Capers Jones, Chief Scientist Emeritus, "Strengths and Weaknesses of Software Metrics", Version 5, March 22, 2006.

[13] Kurmanadham V.V.G.B. Gollapudi, "Function Points or Lines of Code? – An Insight"

[14] Arti Chhikarai, R.S.Chhillar , "Impact of Aspect Orientation on Object Oriented Software Metrics". Vol. 2 No. 3 Jun-Jul 2011, Indian Journal of Computer Science and Engineering (IJCSE)

[15] Lytz, R., Software Metrics for the Boeing 777: A Case Study, Software Quality Journal, 4, 1-13 (1995)

Authors Profile

[1]Ms. Mrinal Singh Rawat is Assitant Professor in the Department of Computer Science and Engineering in MGM's COET, Noida, UP, INDIA. Her Research activities are based on Software Engineering and Reliability Engineering.She is pursuing her M.Tech in Computer Science and Engineering from Amity University.

[2]Ms. Arpita Mittal is working as Assistant Professor in Department of Computer Science at IIMT Merrut, UP, INDIA. Her Research activities are based on Software Engineering and Software Tesing. She is pursuing her M.Tech in Computer Science and Engineering from Amity University.

[3]Mr. Sanjay Kumar Dubey is working as Assistant Professor in Department of Computer Science and Engineering in Amity University Noida, UP, INDIA. His Research area includes Software Engineering and Usability Engineering.He is pursuing his Ph.D in Computer Science and Engineering from Amity Unversity.