International Journal Of Engineering Research & Management Technology

ISSN: 2348-4039

Email: editor@ijermt.org March 2014 Volume-1, Issue-2 Website: ijermt.org

Software Quality Metrics: Concept and Significance

Ramesh Kumar
Research Scholar
Singhania University, Pacheri Bari,
Jhunjhunu (Rajasthan)

Dr.Rajesh Verma
Research Supervisor
(Department of Computer Sc.
Govt. College Indri, Karnal, Haryana)

Abstract:-

Software productivity and quality are important topics with significant economic importance in the modern world. Both productivity and quality should be measured with accuracy using effective metrics and proven measurement practices. The process of software development, including documentation, design, program, test, and maintenance can be measured statistically. Therefore the quality of software can be monitored efficiently. Software metrics is very important in research of software engineering and it has developed gradually. In this paper, software metrics definition, general rules and the types of software metrics and current research on software metrics were given. Software complexity measuring is the important constituent of software metrics and it is concerning the cost of software development and maintenance. In order to improve the software quality and the project controllability, it is necessary to control the software complexity by measuring the related aspects.

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. Software metrics provides measurement of the software product and the process of software production. 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.

What are Software Metrics?

Software metrics are an integral part of the state-of the- practice in software engineering. More and more customers are specifying software and quality metrics reporting as part of their contractual requirements. Industry standards like ISO 9000 and industry models like the Software Engineering Institute's Capability Maturity Model Integrated include measurement. Companies are using metrics to better understand, track, control and predict software projects, processes and products. The term software metrics means different things to different people. When we buy a book or pick up an article on software metrics, the topic can vary from project cost and effort prediction and modeling, to defect tracking and root cause analysis, to a specific test coverage metric, to computer performance modeling. These are all examples of metrics when the word is used as a noun. I prefer the activity based view taken by Goodman. He 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". Software metrics can provide the information needed by engineers for technical decisions as well as information required by management. If a metric is to provide useful information, everyone involved in selecting, designing, implementing, collecting, and utilizing it must understand its definition and purpose.

Quality of the software, different metrics look at different aspects of quality, but this aspect deals with the code. Schedule of the software project on the whole. i.e some metrics look at functionality and some look at documents produced. Cost of the software project. Includes maintenance, research and typical costs associated with a project. Size/Complexity of the software system. This can be either based on the code or at the macro-level of the project and it's dependency on other projects.

Tools for anyone involved in software engineering to understand varying aspects of the code base, and the project progress.

They are different from just testing for errors because they can provide a wider variety of information about the following aspects of software systems

General Uses of Metrics

Software metrics are used to obtain objective reproducible measurements that can be useful for quality assurance, performance, debugging, management, and estimating costs.

- Finding defects in code , predicting defective code, predicting project success, and predicting project risk

- There is still some debate around which metrics matter and what they mean, the utility of metrics is limited to quantifying one of the following goals: Schedule of a software project, Size/complexity of development involved, cost of project, quality of software

Types of Metrics:-

1. **Requirements Metrics :-**

Fairly primitive and predictive power limited.

Function Points:-

- Count number of inputs and output, user interactions, external interfaces, files used.
- Assess each for complexity and multiply by a weighting factor. Used to predict size or cost and to assess project productivity.

Number of requirements errors found (to assess quality)

Change request frequency

- To assess stability of requirements.
- Frequency should decrease over time.
- If not, requirements analysis may not have been done properly.

2. **Size of Requirements:-** used in other metrics: quality metrics, productivity metrics, cost estimation metrics
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.
4. **Process Metrics:-** Processes are set of software related activates which are used to measure the status and progress of the system design and to predict future effects. A process is usually related with some timescale. The timing can be explicit, as when an activity must be finished by a specific date, or implicit, as when one activity must be finished before another can begin. The following examples of a process related metrics that it is proposed to collect when working with object oriented software engineering. Total development time,
    - Development time in each process and sub process,
    - Time spent to modify models from previous processes,
    - Time spent in all kinds of sub process, such as use case specification, object specification, use case design, block design, block testing and use case testing for each particular object,

- Number of different kind of fault found during reviews,
- Number of change proposals on previous models,
- Cost for quality assurance,
- Cost for introducing new development process and tools.

5. **Lines of code LOC**: - Source lines of code (SLOC or LOC) are software metric used to measure the size of a software program by counting the number of lines in the text of the program's source code. SLOC is typically used to predict the amount of effort that will be required to develop a program, as well as to estimate programming productivity or effort once the software is produced. Line of code metrics have advantage

- **Scope for Automation of Counting**: As Line of Code is a physical entity, manual counting effort can be easily eliminated by automating the counting process. Small utilities may be developed for counting the LOC in a program. However, a code counting utility developed for a specific language cannot be used for other languages without modification, due to the syntactical and structural differences among languages.
- **An Intuitive Metric**: Line of Code serves as an intuitive metric for measuring the size of software due to the fact that it can be seen and the effect of it can be visualized. Function points are said to be more of an objective metric which cannot be imagined as being a physical entity, it exists only in the logical space. This way, LOC comes in handy to express the size of software among programmers with low levels of experience.

6. **Design metrics** – computed from requirements or design documents before the system has been implemented
7. **Communication Metrics** – Our hypothesis is that metrics applied to communication artifacts provide more insight into a software development process than outcome metrics. Communication metrics are available earlier and are easier to collect than outcome metrics. Communication records cover more aspects of the project than technical and refer to actual events, instead of planned events. Communication metrics also provide finer granularity at the participant level: electronic mail and memoranda have explicit authors. On a document or software artifact, the individual contribution is usually more difficult to establish.

Current Research in Software Metrics

1. **Fault prediction models**

- Allow organizations to predict defects in code before software has been released.
- There is debate around aggregating the models or modeling only some of the defects more accurately.
- Part of the debate stems from the need for more research in software decomposition and how to decompose software for better quality systems.

2. **Models are used to explain aspects of a software system numerically in at a statistically significant level.**

- Human judgment and software metrics
- Research that is trying to address the human judgment side of metrics processing.
- Working to address the expensive problem of failing projects that I
- mentioned earlier, costs $55 billion annually.
- Software metrics for different types of software defects
- Breaking down the defects that software is measured for will give a better view of the particular type of defect you are interested in.
- Frameworks for understanding metrics and making sure that we are using them correctly
- One framework was shown earlier, more information from those researchers is to follow

Software Metrics: Vision for the Future

- Paper works to address the problem "the low impact that software metrics has on current software development"
- Failing projects are an expensive problem that costs around 55$ billion dollars each year.
- One of the reasons that cognitive psychologists believe this is happening is from various problems with management, and developer error.
- Within the problems with management is how they interpret the metrics, which they base their decision on.
- Software metrics are statistical predictions and estimations, and not just a number. The numbers have three dimensions error, bias, and variance or scatter. A human typically ignores these dimensions for simplicity and with the loss of information comes over optimism and overconfidence.
- Research is being done to use metacognition experiment results in application to software metrics interpretation and use in project planning and reflection.

- Researchers Carolyn Mairand Martin Shepperd want to include the perspective on how people actually employ software metrics results currently so they can understand how to make those metrics better.

Conclusion:-

Over the past years the software industry has become one of the largest industries in human history, and software applications have changed every aspect of business, government, and military operations. No other industry has had such a profound change on human communication and human knowledge transfer. But in spite of the many successes of the software industry, software applications are characterized by poor quality when released, frequent cost and schedule overruns, and many cancelled projects.

Further, software is one of the most labor-intensive industries in history and approaches cotton cultivation in total work hours to deliver a product. In order to solve the problems of software and convert a manual and expensive craft into a modern engineering profession with a high degree of manufacturing automation, the software industry needs much better metrics and measurement disciplines, and much more in the way of standard reusable components. A metrics program that is based on the goals of an organization will help communicate, measure progress towards, and eventually attain those goals. People will work to accomplish what they believe to be important. Well-designed metrics with documented objectives can help an organization obtain the information it needs to continue to improve its software products, processes, and services while maintaining a focus on what is important Better measures and better metrics are the stepping stones to software engineering excellence. It is hoped that this report will highlight both measurement problems and also increase the usage of effective metrics such as function points and defect removal efficiency.

References :

1. Paul Goodman, Practical Implementation of Software Metrics, McGraw Hill London
2 Vasilescu, B., Serebrenik, A., van den Brand, M. (2011). By No Means: A Study on Metrics. WETSoM'11, Waikiki, Honolulu, HI USA.
3 Mair, C., Shepperd, M. (2011) Human Judgement and Software Metrics: Vision for the Future. WETSoM'11, Waikiki, Honolulu, HI USA.
4 Norman E. Fenton, Software Metrics, A Rigorous Approach, Chapman & Hall, London.
5 Boehm, Barry Dr.; Software Engineering Economics; Prentice Hall, Englewood Cliffs, NJ;
6 N. E. Fenton, "Software Metrics: Successes, Failures & New Directions," presented at ASM 99: Applications of Software Measurement.

7 Misirli, A., Caglayan, B., Miransky, A., Bener, A., Ruffolo, N. (2011). Different Strokes for Different Folks: A Case Study on Software Metris for Different Defect Categories. WETSoM'11 (May 24, 2011), Waikiki, Honolulu, HI USA.

8 Agile Software Development: Principles, Patterns and Practices. Pearson Education. Robert Cecil Martin

9 Nagappan, N., Ball, T. (2007) Using Software Dependencies and Churn Metrics to Predict Field Failures: An Empirical Case Study. IEEE First International Symposium on Empirical Software Engineering Measurement.

10 Kaner, C., Bond, W. Software Engineering Metrics: What Do They Measure and How Do we Know?