A Study of Various Static and Dynamic Metrics for Open Source Software

Ankush Vesra
Department of Computer Science
Guru Nanak Dev University
Amritsar

Rahul
Department of Computer Science
Guru Nanak Dev University
Amritsar

ABSTRACT

Software metrics is developed and utilized by the different software organizations for evaluating and assuring software code quality, operation, and maintenance. Software metrics measure various kinds of software complexity like size metrics, control flow metrics and data flow metrics. These software complexities must certainly be continuously calculated, followed, and controlled. Among the main objectives of software metrics is that pertains to a procedure and product metrics. It is definitely considered that high level of complexity in a component is bad compared to a low level of complexity in a module. Software metrics may be used in various phases of software development lifecycle. In this paper, a survey on various software metrics has been done. Moreover they are categorized into static and dynamic metrics. The paper ends with in conclusion and the near future scope to overcome some issues for the software metrics.

Keywords

Software Metrics, Static Metrics, Dynamic Metrics.

1. INTRODUCTION

Software metrics could be classified into three categories: product metrics, process metrics, and project metrics. Product metrics describe the characteristics of the product such as for instance size, complexity, design features, performance, and quality level. Process metrics may be used to enhance software development and maintenance. Examples include the potency of defect removal during development, the pattern of testing defect arrival, and the response time of the fix process. Project metrics describe the project characteristics and execution. Examples include the amount of software developers, the staffing pattern over the life cycle of the software, cost, schedule, and productivity. Some metrics participate in multiple categories. As an example, the in process quality metrics of a task are generally process metrics and project metrics.

Software quality metrics are a part of software metrics that concentrate on the product quality areas of the product, process, and project. Generally, software quality metrics are far more closely related to process and product metrics than with project metrics. Nonetheless, the project parameters such as for instance the amount of developers and their skill levels, the schedule, the size, and the corporation structure certainly affect the caliber of the product. Software quality metrics could be divided further into end-product quality metrics and in-process quality metrics. The essence of software quality engineering would be to investigate the relationships among in-process metrics, project characteristics, and end-product quality, and, on the basis of the findings, to engineer improvements in both process and product quality. Moreover, we ought to view quality from the whole software life-cycle perspective and, in this regard, we will include metrics that measure the standard degree of the maintenance process as another group of software quality metrics. In this chapter we discuss several metrics in all of three categories of software quality metrics: product quality, in-process quality, and maintenance quality.

1.1 Software Metrics

Software metrics is one consistent topic of research in software engineering. The role of software metrics is to locate significant estimates for software products and directs us in intriguing managerial and technical decisions. Software metrics have grown to be an important section of software development and are utilized during every phase of the software development life cycle. The name software metric [1][2] is connected with varied measurements of computer software and its development. Research in the region of software metrics tends to concentrate predominantly on static metrics which can be obtained by static analysis of the program artifact. To raised understand the impact of code changes and track complexity issues in addition to along with code quality software metrics are frequently utilized in the program development life cycle. Ideally, software metrics must certianly must be computed continuously through the development process to allow the perfect tracking. Moreover, software metrics must certanly be should really be definable by development teams not to only cover general factors, but to measure company, project or team specific goals.

1.1.1 General Uses of Software Metrics

Software metrics are accustomed to obtain objective reproducible measurements that may be helpful for quality assurance, performance, debugging, management, and estimating costs.

Finding defects in code (post release and just before release), predicting defective code, predicting project success, and predicting project risk

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

2. TYPES OF SOFTWARE METRICS

There are different types of software metrics defined under two categories. They are static and dynamic software metrics.

2.1 Static Metrics

Static metrics are obtainable at the early phases of software development life cycle (SDLC). These metrics deals with the structural feature of the software system and easy to gather. Static complexity metrics estimate the amount of effort needed to develop, and maintain the code. First static metric [3] (LOC/KLOC) was used to measure the productivity of a program. The most commonly used complexity metric before 1990 was cyclomatic [4] complexity that was measured by

McCabe. He uses the flow graph and some mathematical equations to compute software complexity. This metric was used in code development risk analysis, change risk analysis in maintenance and in test planning.

Table 1. Static Metric

Serial No..	Static Software Metric	Description
1	AHF	This metric is used to measure the invisibilities of attributes in classes. The attributed invisibility is defined as the percentage of the total classes from which the attribute is not visible.
2	AIF	Attribute inheritance factor
3	AVPATHS	Average Depth of Paths is calculated by counting the number and size of all paths from all methods, and then dividing that number by the number of methods which had other method calls. In other words, the average depth of paths from methods that have path at all.
4	ACLOC	Average lines per class: This metric gives the average Class size in terms of LOC.
5	AMLOC	Average lines per method: This metric gives the average Method size in terms of LOC.
6	PDIT	Depth of Inheritance tree: The Depth of Inheritance Tree for a Project is the deepest or maximum of all inheritance trees within the project
7	LOC	Lines of code: Number of Lines in the project, including source, whitespace and comments.
8	MHF	Method Hiding Factor is one of the important metrics of object oriented programming that is calculated by summing the visibility of each method in respect to the other classes in the project. It is used to measures the invisibilities of methods in classes. The invisibility of a method is the percentage of the total classes from which the method is not visible.
9	MIF	Method inheritance factor [5] gives the information about the impact of inheritance in your file or program. It is calculate as ratio of inherited methods to the total number of methods
10	NCLASS	It is another static metrics that count the number of classes in a program.
11	SEIMI	SEI Maintainability Index is one of the important measures of maintenance. SEIMI is a measure of the maintainability of the project, as described by the Software Engineering Institute
12	SLOC	Source lines of Code are an important measure of source line of code. Counting lines is used for estimating the amount of upholding or maintenance required and it can be used to normalize other software metrics.

2.2 Dynamic Metric

Dynamic metrics are accessible at the late stage of the software development life cycle (SDLC). These metrics capture the dynamic behavior of the system and very hard to obtain and obtained from traces of code. Dynamic metrics are derived from an analysis of code while it is executing. Software metrics for the qualitative and quantitative assessment is the combination of static and dynamic metrics for software's [6]. They provide an indication of what calls are actually taking place, the number of statements executed and what paths are being executed. Since software maintainability is an important attribute of software quality, accurate prediction of it can help to improve overall software quality[7]. Dynamic metrics include both complexity measures and measures useful in reliability modeling. Dynamic metric values are dependent on the input or test data with which system software is run.

Table 2. Dynamic Metrics

Serial No.	Dynamic Metric	Description
1	Bug Counting	error, flaw in a computer program that causes it to produce an incorrect or unexpected result, or to behave in unintended ways
2	Halstead complexity	identify measurable properties of software, and the relations between them
3	function point	**It** is a unit of measurement to express the amount of business functionality an information system provides to a user
4	Cyclomatic complexity	Indicate the complexity of a program

3. LITERATURE SURVEY

Mertoguno, J. S. et al. [8] deal with the look and modeling of a neuro-expert (NE) system for the prediction of software metrics. The NE includes two neural networks and a specialist system with fuzzy reasoning is to be able to achieve a much better evaluation of software metrics. More specifically, the significance of using both neural networks and a specialist system is to mix the adaptive nature of neural networks on different sets of data, with the high-level reasoning supplied by the expert system, for a much better overall evaluation. In this paper the very first stage modeling of the NE system is presented.

Gray, Andrew R., and Stephen G. MacDonell [9] examined the implications of using these methods and provides some recommendations concerning when they might be an appropriate. The utilization of regression analysis to derive predictive equations for software metrics has recently been complemented by increasing variety of studies using non-traditional methods, such as for example neural networks, fuzzy logic models, case-based reasoning systems, and regression trees.There's also had been an increasing degree of sophistication in the regression-based techniques used, including robust regression methods, factor analysis, and more efficient and better validation procedures. A contrast of the different techniques can also be made when it comes with regards to their modelling capabilities with specific mention of the software metrics.

Subramanian, Girish, and William Corbin [10] centered on analyzing certain software metrics in a object-oriented (OO) environment. The metrics collected and analyzed includes size,quantity of message (NOM) sends, reuse, inherited methods, and hierarchical nesting level. The website used could be the factory systems department of a big sizable manufacturing company. This department uses SmallTalk whilst because the OO programming language to implement the OO design paradigm. Using automated tools developed in SmallTalk, these metrics were collected from three domain applications comprising 600 classes. Four propositions are empirically tested and the outcomes provided in this study.

Olague, Hector M. et al. [11] explored the power of those three metrics suites to predict fault-prone classes using defect data for six versions of Rhino, an open-source implementation of JavaScript written in Java. They figured the CK and QMOOD suites contain similar components and produce statistical models which can be effective in detecting error-prone classes. In addition they conclude that the class components in the MOOD metrics suite are negative class fault-proneness predictors. Analyzing multivariate binary logistic regression models across six Rhino versions indicates these models might be useful in assessing quality in OO classes produced using modern highly iterative or agile software development processes.

Shatnawi, Raed, and Wei Li [12] examined three releases of the Eclipse project and discovered that however while some others metrics can still predict class error proneness in three error-severity categories, the accuracy of the prediction decreased from release to release. Furthermore, they discovered that the prediction can't be used to construct a metrics model to recognize error-prone classes with acceptable accuracy. These findings claim that as something evolves, the utilization of some commonly used metrics to recognize which classes are far more susceptible to errors becomes increasingly difficult and they need to seek alternative methods (to the metric-prediction models)to discover error-prone classes should they want high accuracy.

Honglei et al. [13] gave , software metrics definition and the real history of and the types of software metrics were overviewed. Software complexity measuring may be the important constituent of software metrics and it's concerning the price of software development and maintenance. To be able to improve the software quality and the project controllability, It's necessary to manage the software complexity by measuring the related aspects. This paper respectively expounds McCabe methods and C&K metric method for types of complexity metrics.

Mohsin, Shaikh, and Zeeshan Kaleem [14] suggested an approach which explores effective code comprehension by combining Software metrics and technique called Program Slicing. Program slicing is static program analysis process for code automation which could develop efficient measures for coupling, cohesion, complexity. Such novel design of software metrics with analytical approach can insure reliable development of software system.

Catal, Cagatay et al. 15[] centered on case studies of five public NASA datasets and details the construction of Naive Bayes-based software fault prediction models both before and after applying the proposed noise detection algorithm. Experimental results show this noise detection approach is quite effective for detecting the class noise and that the performance of fault predictors utilizing a Naive Bayes algorithm with a logNum filter improves if the class labels of identified noisy modules are corrected.

Takai, Yasunari et al. [16] centered on latent faults detected by static analysis techniques. The coding checker is popular|to locate coding standards violations which are strongly associated with latent faults. In this paper, they proposed new software metrics centered on coding standards violations to fully capture latent faults in a development. They analyzed two open source projects by utilizing proposed metrics and discuss the effectiveness.

Debbarma, Mrinal Kanti et al. [17] discussed the different metrics and comparison between both static and dynamic metrics. They tried to judge and analyze different aspects of software static and dynamic metrics in regression testing that provides of estimating the time and effort required for testing.

Suresh, Yeresime et al. [18] evaluated software like ATM using available subset of metrics from traditional and object-oriented methodology. The standard metrics such as for instance cyclomatic complexity, size and comment percentage are accustomed to compute the software complexity. This paper also analyses a popular subset of object-oriented metrics like the Chidamber and Kemerer metric suite to compute the system reliability. The metric values are evaluated for a actual life application, which supports us to understand the complexity and the reliability of the ATM software.

Ors, Kilyen Attila, and Barabas Laszlo [19] presented an interpreter framework created for measuring static and dynamic characteristics of a Scade model. Though some of the software metrics have grown to be industrial standards in software development and for popular languages there's an assortment of software measurement tools, for Scade you will find no such tools. The main achievement is they developed an interpreter for metrics, and they provided quick access for the information gained from these measurements. Additionally they implemented a few of the canonical software metrics like Cyclomatic complexity and Halstead's Software Science.

Singh, Pradeep Kumar, and Om Prakash Sangwan [20] emphasized on a new framework to gain access to the Aspect Oriented Software's (AOS) using software metrics. Software metrics for the qualitative and quantitative assessment may be the combination of static and dynamic metrics for software's. It is located from the literature survey that till date most the framework only considered the static metrics based assessment for aspect oriented software's. Within their work they've mainly considered the set of static metrics along side dynamic software metrics specific to AspectJ. This framework may give a new research direction while predicting the software attributes because earlier dynamic metrics were neglected while evaluating the standard attributes like maintainability, reliability, understandability for AO software's. Centered on basic fundamentals of software engineering dynamic metrics are equally important as well as static metrics for software analysis. An identical concept is borrowed to use on aspect oriented software development by the addition of dynamic software metrics. Presently they've only proposed a construction and model using the static and dynamic metrics for the assessment of aspect oriented system but nonetheless the proposed approach have to be validated.

Yadav, Harikesh Bahadur, and Dilip Kumar Yadav [21] proposed a fuzzy logic based model for predicting software defect density indicator at each phase of the SDLC. The predicted defects of twenty different software projects are observed very close to the particular defects detected during testing. The predicted defect density indicators are very useful to analyze the defect severity in various artifacts of SDLC of a software project.

Yadav, Harikesh Bahadur, and Dilip Kumar Yadav [22] discussed that the quantity of the program defect prediction model using software metrics has been proposed in last two decades. However, predicting software defect by taking all the software metrics (traditional, object oriented and process) is computationally complex. Therefore, an intelligent choice of metrics plays an important role in improving the program quality. In the first phases of the program development life cycle, software metrics are related to uncertainty and could be assessed in linguistic terms. Construction of membership function is essential because the success of a technique depends upon the membership functions used. Therefore, in this paper, a methodology has been proposed to create the membership functions of software metrics.

4. CONCLUSION AND FUTURE SCOPE

In this paper, a survey on various software metrics has been done. The metrics has also been divided into static and dynamic metrics**.** Static metrics are obtainable at the early phases of software development life cycle (SDLC). These metrics deals with the structural feature of the software system and easy to gather. Static complexity metrics estimate the amount of effort needed to develop, and maintain the code. Dynamic metrics are accessible at the late stage of the software development life cycle (SDLC). These metrics capture the dynamic behavior of the system and very hard to obtain and obtained from traces of code. The various types of the metrics has also been mentioned in this paper under these two different categories.

5. ACKNOWLEDGEMENTS

I would like to thanks to God, all my teachers, family and friends who guide and support me to write this paper. They always stand with me and always help me to do the work.

6. REFERENCES

[1] H F Li, W K Cheung "An Empirical Study of Software Metrics" Software Engineering IEEE Transactions on (1987) Volume: SE-13, Issue: 6, Pages: 697-708.

[2] N E Fenton "Software Metrics" Conference Proceedings of on the future of Software engineering ICSE 00(2000) Volume: 8, Issue: 2, Publisher: ACM Press [3] Kuljit Kaur Chahal, Hardeep Singh "Metrics to study

[3] Li, H.F., Cheung, W.K. "An Experimental investigation of software metric and their relationship to software development effort", IEEE Transaction on software engineering 649-653, Piscataway, NJ, USA.

[4] Thomas J McCabe, "A Complexity Measure", IEEE Transaction on Software Engineering, Vol. SE-2 No. 4 [308-320]

[5] KP Srinavan, Dr. T Devi, "Design and Development of procedure for new object oriented design metrics', IJCA, Vol. 24, No. 8, Jun 2011

[6] Singh, Pradeep Kumar, and Om Prakash Sangwan. "Aspect Oriented Software Metrics Based Maintainability Assessment: Framework and Model." (2013): 1-07.

[7] Kaur, Arvinder, Kamaldeep Kaur, and Kaushal Pathak. "Software maintainability prediction by data mining of software code metrics." In *Data Mining and Intelligent Computing (ICDMIC), 2014 International Conference on*, pp. 1-6. IEEE, 2014.

[8] Mertoguno, J. S., R. Paul, N. G. Bourbakis, and C. V. Ramamoorthy. "A neuro-expert system for the prediction of software metrics." *Engineering Applications of Artificial Intelligence* 9, no. 2 (1996): 153-161.

[9] Gray, Andrew R., and Stephen G. MacDonell. "A comparison of techniques for developing predictive models of software metrics." *Information and software technology* 39, no. 6 (1997): 425-437.

[10] Subramanian, Girish, and William Corbin. "An empirical study of certain object-oriented software metrics." *Journal of Systems and Software* 59, no. 1 (2001): 57-63.

[11] Olague, Hector M., Letha H. Etzkorn, Sampson Gholston, and Stephen Quattlebaum. "Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes." *Software Engineering, IEEE Transactions on* 33, no. 6 (2007): 402-419.

[12] Shatnawi, Raed, and Wei Li. "The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process."*Journal of systems and software* 81, no. 11 (2008): 1868-1882.

[13] Honglei, Tu, Sun Wei, and Zhang Yanan. "The research on software metrics and software complexity metrics." In *Computer Science-Technology and Applications, 2009. IFCSTA'09. International Forum on*, vol. 1, pp. 131-136. IEEE, 2009.

[14] Mohsin, Shaikh, and Zeeshan Kaleem. "Program Slicing Based Software Metrics towards Code Restructuring." In *Computer Research and Development, 2010 Second International Conference on*, pp. 738-741. IEEE, 2010.

[15] Catal, Cagatay, Oral Alan, and Kerime Balkan. "Class noise detection based on software metrics and ROC curves." *Information Sciences* 181, no. 21 (2011): 4867-4877.

[16] Takai, Yasunari, Takashi Kobayashi, and Kiyoshi Agusa. "Software Metrics based on Coding Standards Violations." In *Software Measurement, 2011 Joint Conference of the 21st Int'l Workshop on and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA)*, pp. 273-278. IEEE, 2011.

[17] Debbarma, Mrinal Kanti, Nirmalya Kar, and Ashim Saha. "Static and dynamic software metrics complexity analysis in regression testing." In *Computer Communication and Informatics (ICCCI), 2012 International Conference on*, pp. 1-6. IEEE, 2012.

[18] Suresh, Yeresime, Jayadeep Pati, and Santanu Ku Rath. "Effectiveness of software metrics for object-oriented system." *Procedia Technology* 6 (2012): 420-427.

[19] Ors, Kilyen Attila, and Barabas Laszlo. "Scade interpreter for measuring static and dynamic software metrics." In *Intelligent Systems and Informatics (SISY), 2013 IEEE*

11th International Symposium on, pp. 123-128. IEEE, 2013.

[20] Singh, Pradeep Kumar, and Om Prakash Sangwan. "Aspect Oriented Software Metrics Based Maintainability Assessment: Framework and Model." (2013): 1-07.

[21] Yadav, Harikesh Bahadur, and Dilip Kumar Yadav. "A fuzzy logic based approach for phase-wise software defects prediction using software metrics."*Information and Software Technology* 63 (2015): 44-57.

[22] Yadav, Harikesh Bahadur, and Dilip Kumar Yadav. "Construction of Membership Function for Software Metrics." *Procedia Computer Science* 46 (2015): 933-940.