ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Brief Study on Software Quality Metrics and Software Complexity Metrics in Web Application

B.Jayanthi*, Dr.K.KrishnaKumari
Assistant Professor & Ph.D Scholar, School of Computer Studies(UG), Rathnavel Subramaniam College Of Arts & Science Sulur,Coimbatore, India
Director, Department of Computer Applications (MCA) ,Rathnavel Subramaniam College Of Arts & Science Sulur, Coimbatore, India

Abstract

Software metrics is very important in research of software engineering and it has developed gradually. In this paper, software metrics definition were given and the history of and the types of software metrics were overviewed. 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. 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. This paper respectively expounds various Software Metrics & Software Complexity Metrics.

Keywords: Software Metrics, Software Complexity Metrics, Software Development Life Cycle (SDLC), Procedure Metrics, Project Metrics, Product Metrics.

Introduction

The current software engineering practices observed in the local software industry generally lacks the software metrics planning part. The specific planning required improving the software practices, to gauge the productivity of the individuals, and productivity at different phases of software development life cycle (SDLC), is generally missing. These organizations do not have any planning to gauge and control the business development issues and improve their productivities. During development phase, most of these software houses do collect data about different aspects of the project like project planning, requirement specification, testing, and bugs found. However, these organizations, generally, do not understand the importance of such data elements, how to maintain the data, how to built measurements plan, and most importantly how to use this data to improve their processes and productivity. In fact most of the time, the available data gets destroyed and hence never used. The current talk will discuss the need and importance of software metrics, the role of metrics in different phases of SDLC, and particularly the requirement phase. The requirement elicitation is considered as the most important aspect of software development, and if not handled properly, may cause severe productivity and quality issues. The talk will elaborate different metrics that may be used in requirement elicitation process and hence may help improve the requirement document resulting in improved productivity in the overall software development.

Among the things that govern our life, metrics occupies at the centre position. Some people pointed out that without metrics, there should not exist modern science. Metrics makes the abstract concept more intuitive and more understandable. It is very important to incorporate software metrics in our software lifecycle. The development of software includes requirement documents, designs, programs and tests, which all can be measured and analyzed via metrics methods. Software metrics can monitor the quality of software production and then makes forecast and maintenance easier. With the rapid development of large-scaled software, the

Complexity of software grows fast, which makes the quality more and more difficult to control. Therefore the production procedure should be checked by using metrics. In order to assure the reliability and the quality of software, correct metrics methods should be used for the procedure and quality improvement. In this article, we first introduce the software metrics

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

including the definition of metrics and Software Complexity Metrics.

Review of literature

Software metrics definition

Definition 1: Software Metrics provide a measurement for the software and the process of software production. It is giving quantitative values to the attributes involving in the product or the process.

Definition 2: Software metrics is to give the attributes some quantitative descriptions. These attributes are extracting from the software product, software development process and the related resources. They are listed as below:

- Product: the documents and programs generated during the software development process.
- Process: various processes related to software development, such as software design, implementation,
- test, and maintenance, etc.
- Resources: the supporting resources such as programmers, and cost of the product and processes, etc.

Definition 3: software measurement provides continuous measures for the software development process and its related products. It defines, collects and analyzes the data of measurable process, through which it facilitates the understanding, evaluating, controlling and improving the software product procedure.

Definition 4: According to IEEE "standard of software Quality Metrics Methodology", software metrics is a function, with input as the software data, and output is a value which could decide on how the given attribute affect the software.

Developing History of Software Metrics

The research on metrics started in 70s and at that time the development method of software was procedure-oriented.1974 Wolverton firstly used the LOC (Line of Codes) to measure the production ratio of the programmers. He promoted the criteria as person/month as the metrics unit. 1976 McCabe proposed a term "cycle" as the metrics for the software complexity based on Graph definitions. Halstead proposed a term "software sciences" to estimate the work load or work time of the programmers. It can use operator, operand and calling process to represent the software complexity**.** In 1979, Albrecht presented a method called function points, which came from the requirement specification. This method can be used in the requirement analysis phase and was very popular in America and Europe. During the decades of 80's, and 90's, software metrics had a fast growth. In 1993 scholars from Taiwan J-Y Chen and J-F Liu proposed Chen&Liu method, which used complexity, reusability, and attributes of class etc. to measure the Object- oriented software. In 1994, Chidamber Kemerer Published a set of Object-oriented metrics based on inheritance tree. And later it was named as C&K metrics methods. In 1995, Brito advocated a set of metrics based on Object-oriented attributes, called MOOD metrics. It gave 6 metrics indexes aroused from encapsulation, inheritance, coupling, and polymorphism of O-O program. In 2001, Victor and Daily proposed a method called SPECTRE which is based on software components. This method is used to estimate the development time and the scale of modules. In 2003, Hastings and Sajeev introduced a new method, which was Vector Size Measure (VSM). It was applicable for the early stage in the software lifecycle. It was used for metrics on software scale, software classification. In 2004, Arlene F. proposed a new forecast method, which was used to build an appropriate forecast on work load and track the production power based on the objects and their attributes.

Classes of software metrics

For the software metrics, there are 3 types: procedure metrics, project metrics and product metrics.

Procedure metrics

Procedure metrics lay emphasize on the procedure of the software development. It is mainly focus on how long a procedure last, how about the cost, whether the methods used are effective, how is the result of comparing this method with alternative methods, etc. It includes the improvement of the procedure and the prediction for the future procedure. Procedure metrics is fulfilled within the whole organization. The main part of this metrics includes maturity, management, life cycle, product ratio, defect ratio, etc.. This metrics is used by high level managers to obtain the development status. It is beneficial to the control and management of the whole development procedure.

Project metrics

Project metrics is to understand and control the project situation and status. The metrics are normally accomplished for a specific project and it includes scale, cost, workload, status, production power, risk, the degree of satisfaction from clients, etc.. Project metrics is mainly to adjust the project to avoid the problems or risks and help to optimize the development plans. And thereafter project metrics is

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

to improve the quality of the product via advances in technique methods and management strategies.

Product metrics

Product metrics is to understand and control the quality of the product. It is used to predict and manipulate the quality of the product. Centring around the quality of the product, the metrics mainly include the reliability, maintainability, product scale, software complexity, portability, documents, etc.. Product metrics is to measure the medium or the final product from the phase of system requirement to the phase of system maintainability.

Software complexity metrics

Software complexity metrics is the measure of the cost consumed by developing, maintaining, and usage of the software. Complexity is the main factor that can lead to defects. The problem of reliability is intrinsically the problem of software complexity. When the complexity reaches some thresholds, the defects or faults of the software grow rapidly. The maintainability of the software is also having tight relationship with complexity..

Principles of software complexity metrics

Software complexity metrics is the main part of software measurement and is the main method to assure the quality of the software. K.Magel summarized the 6 aspects to describe the software complexity

1) The difficulty of understanding the program.
2) The difficulty of correcting the defects and maintaining the software.
3) The difficulty of explaining the software to other people.
4) The difficulty of updating the program according to some assigned rules.
5) The work load of writing programs according to the design.
6) The availability of necessary resources when programs are executing.

Unfortunately until now there is no idea and safe model for complexity metrics. Normally people use the principles listed below when considering the complexity metrics.

1) The relationship between complexity and the size of the program is non-linear.
2) The more complicated the control structures are, the more complex the program is.
3) The more complicated the data structures are, the more complex the program is.
4) Inappropriate ‘goto’ statement will make the program more complex.
5) Loop structure is more complex than selection structure, while selection structure is more complex than the sequential structure.
6) The more global variables or non-local variables there are the more complex the programs are.
7) Function parameter calls by reference are more complicated than calls by value.
8) The more connected between modules or procedures (tight coupling), the more complex the program is.
9) For OO projects, the more inheritance, more coupling between classes means more complicated the program.

Importance of complexity metrics

The importance of complexity of programs can be illustrated as below:

1) Metrics on complexity can help people predict and maintain projects. For example, the more complex the modules are, the more effort should be put on these modules for test and maintenance. For those especially complicated modules, we may figure out the ways to dissolve them and therefore reduce their complexity level, which will reduce the defects and make the module easier for test and maintenance.

2) Complexity metrics can help to evaluate the workload of programming and cost of development. Also it can help to estimate the effort put on maintenance.

3) Complexity metrics can help to select the most appropriate program with same functions. The less complexity the programs are, the better they are. Meanwhile, the complexity metrics can also be used to foresee the defects or errors

Conclusion

With the rapid advance of software, their metrics have also developed quickly. Software metrics become the foundation of the software management and essential to the success of software development. The growth of software complexity stimulates the occurrence of the software complexity metrics. The complexity of software will directly affect the eligibility, reliability of the software. Although people realize the importance of software metrics, the metrics field still needs to grow quickly to meet the requirement of software development. Lack of solid theoretic background and the finesse of methods, software metrics is still young comparing to other software theories.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

References

1. Hu Xiuwen, Tian Zhonghe, Research on Software Metrics[J].Information Technology，2003，Page(s)：58-60
2. Zeng Beiming. Research and Realization of Automatic Test Frame[D].University of Wuhan, Master Degree Dissertation，2006
3. MINGZHI MAO, XIAONAN LUO. Software Quality Management and Software Process Model [J]. Journal of Information and Computation Science, 2004, Page(s) : 203 - 207.
4. IEEE STD 1061-1992.IEEE Standard for a software quality metrics methodology [A].Institute of Electronical Engineers [M]. New York: IEEE Computer Society, 1993.
5. WolfhartB ,Goethert, Elizabeth K., Bailey Mary B., Busby, Software Effort & Schedule Measurement: A Framework for Counting Staff-hours and Reporting Schedule Information, Technical Report CMU/SEI-92-TR-021 ESC-TR-92-021.
6. M. Jorgensen, Software quality measurement, Advances in Engineering Software 30(1999), Page(s) : 907-912
7. Chidamber S R, Kemerer C F. A metrics suite for object oriented design. IEEE Transactions on Software Engineering, 1994, Page(s) :476-492
8. Lorenz M. Object-Oriented Software Development: A Practical Guide. New York: Prentice Hall, Inc., 1993
9. Kearndy J K, Sedlmeyer R L, Thompson WBetal. Software complexity measurement. Communications of the ACM, 1986, Page(s):1044-1050
10. Weyuker E.,Evaluating software complexity measures. IEEE Transactions on Software Engineering, 1988, Page(s) :1357-1365