Novel Approach to Software Metrics

Reena Sharma, R.S.Chhillar

Abstract : Software metrics are becoming important day by day. Many metrics have been defined and have been related to class coupling cohesion etc. First of all it is difficult to choose the correct metrics for particular software and secondly most of the metrics only cater to requirements phase. It is to be understood that the importance of software metrics cannot be undermined in the design and implementation phase also. The work discusses the various techniques, their merits and demerits and intends to propose a new system for measuring the goodness of implementation phase. The concept of Object Oriented Software Metrics has also been explored. The proposed metrics uses the concept of Genetic Algorithms, which are based on the theory of natural selection. Thus, the work intends to introduce natural selection techniques for measuring the quality of software.

Keywords : Software Metric, Measure, Object Oriented Metrics, Genetic Algorithms

I. INTRODUCTION

It is said that “what can be measured can be studied”. The concept is valid for all types of engineering including software engineering. Measurements in software are done using Software metrics. Software metric is a measure of some property of a piece of software or its specifications. Since quantitative measurements are essential in all sciences, there is a continuous effort by computer science practitioners and theoreticians to bring similar approaches to software development. The use of software metrics is primarily to determine the quality of software. However, they are also used not only to predict the quality of product or process, but also to improve the quality. There are many types of metrics ranging from object, component and aspect metrics. The following work throws some light on the various metrics also. The goal of the present work is Obtaining objective, reproducible and quantifiable measurements, which may have numerous valuable applications in schedule and budget planning, cost Estimation, quality assurance testing, software debugging, software performance optimization, and optimal personnel task assignments. The work proposes the use of genetic Algorithms in Metrics. The work proposes the use of Genetic Algorithms in Software metrics. Genetic Algorithms are based on the theory of natural selection and the survival of the fittest. The concept is to prioritize the various elements in a software and then apply GA to select the fittest whose value is representational as per as software is concerned. The idea can be a turning point as per as software measurements are concerned.

II. CLASSIFICATION OF METRICS

Software is robust if it behaves “reasonably”, even in circumstances that were not anticipated in the requirement specification-for example, when it encounters incorrect input data or some hardware malfunction [1]. On the part of programmer it is naïve to accept ideal input from a common, unsophisticated user. The software should be capable of diagnosing certain classes of errors. Robust design reads out error circumstances to be estimated and error-handling paths to be set up to deflect. This is attained by exception-handling mechanism. Many metrics have been proposed by researchers which measure the desirable characteristics of software one of which is exception handling. A set of metrics has been proposed in this work to measure the robustness of design. The metrics are proposed is to be analyzed on sample data set.

APT (applied psychological technology) metrics are desirable to develop proficient software system. The role of Object-oriented metrics in this portion cannot be undermined. One of the most important parts this work evaluates is the object-oriented software metrics approach to describe the characteristics of the software system.

Object-oriented systems are efficient software systems and have reduced size. The classes, number of methods in a class, the interaction of these methods are some of the factors that can be considered while evaluating the system. These factors are capable of providing comprehensive descriptions of software’s configuration. The objects in these software systems cooperate while remaining in their own local state. The object-oriented features can be used to increase the efficiency of object-oriented systems. Software metrics help us to determine the software quality and the cost. The metrics indicating the software quality are measured in the early phases while the object-oriented software analysis normally is used in later phase of SDLC. The work compares the conventional metrics used in the implementation phase and Object Oriented Metrics.

The Present work clubs the above concepts with the theory of natural selection to propose an altogether new concept. The metrics proposed can be applied to large software and change the way we look at measurements till now.

III. MOTIVATION

The inputs to the implementation phase of a Software Design Life Cycle are Number of Implementation Plans,

Manuscript received on July, 2012.

Reena Sharma, Department of computer science & application M.D. university, Rohtak-124001, Haryana,india

R.S.Chhillar. Department of computer science & application M.D. university, Rohtak-124001, Haryana,india.

Software Requirements Documents, Number of Software Design Documents, Design Documents etc. [10].
The outputs of the phase include the Number of Updated Implementation Plans, Number of Test Plans and the Number of Test Procedure Documents. So the various things that we need to measure are as follows:

- The time require to complete the Implementation Phase
- The Cost for Implementation Phase
- The Staff Size for Implementation Phase

In order to measure the above things matrices are needed. Some of the works have analyzed and studied the above factors. According to most of the works Defect Metrics, the Lines of Code (LOC), and the Halstead product metric assist in the measurements of the implementation phase.

To calculate the number of defects in the system based on the number of Function Points the rule given by Capers Jones is used. According to the rule the potential number of defects is proportional to FP125, FP stands for Function points. The Lines of Code (LOC) metric specifies the number of lines that the code has except for the comments [8]. The LOC metric is often presented on thousands of lines of code (KLOC). LOC is used during the testing and maintenance phases. Several LOC tools are enhanced to recognize the number of lines of code that have been modified or deleted from one version to another. But there are some reservations as per LOC are concerned.

The most significant contribution to the Implementation phase metrics was by Halstead [11]. According to him a program could be measured by counting the number of operators and operands. He defined a set of formulas to calculate the vocabulary, the length and the volume of the software program [11].

IV. PROGRAM VOCABULARY AND OBJECT-ORIENTED METRICS

4.1. Vocabulary

The program vocabulary is given by the number of unique operators plus the number of unique operands

n = n1+n2

n = program vocabulary

n1 = number of unique operators

n2 = number of unique operands

4.1.1. Program Length

The program length is the total usage of all the operators appearing in the implementation plus the total usage of all operands appearing in the implementation.

N =N1+N2

N = program length

N1= all operators appearing in the implementation

N2 = all operands appearing in the implementation

4.1.2 Program Volume

The program volume is defined as the size of the program. This definition is defined by equation 11.

V= N log2 n

V = program volume

N = program length

n = program vocabulary

4.2. Object Oriented Metrics

Chidambaram and Kemmerer (CK) defined six metrics [5]. They are Weighted Methods per Class, Response sets for Class, Lack of Cohesion in Methods, coupling between Object Classes, Depth of Inheritance Tree of a class and Number of Children of a class. CK metrics are widely used to measure the design complexity [6], [7], [8]. There have been many endeavors to verify the metrics. Several investigational studies have been carried out to validate CK metrics [9]. The summary of CK metrics is given as follows.

4.2.1. Weighted Methods per Class:

Number of methods of a certain class without inherited methods

4.2.2. Response set For Class:

Number of methods that can be performed by a certain class regarding a received message

4.2.3. Lack of Cohesion in Methods

Number of disjunctive method pairs of a certain class

4.2.4. Coupling between Objects and Classes

Number of couplings between a certain class and all other classes

4.2.5. Depth of Inheritance :

Maximal depth of a certain class in an inheritance structure

4.2.6. Number of Children of a Class:

Number of direct subclasses of a certain class

The CK metrics are aimed at the design of object oriented system rather than implementation. The present work intends to use these for the implementation phase as well.

V. INTENDED WORK

The intended work is to develop new metrics which support the work of Halstead and still be more comprehensive and better. The metrics being developed takes into account the number of functional units and the type of coupling [12]. The type of coupling determines the fitness of a module. This can be the basis of the fitness function of the genetic algorithm part. The value of delta in the fitness function is proportional to the coefficient obtained by the type of coupling. The process of initial population generation will be followed by crossover. The crossover chromosomes will find new functional units. The present work does not make use of mutation, as its need was not realized in the analysis. The overall metrics will serve the purpose of exhaustively defining the software implementation. The metrics also take into account the type of coupling which is rarely done till now. The designing and the formulas' have been derived. The work is at present in the testing phase. The metrics proposed are being checked against practical examples to gauge their behavior. The work takes into account the shortcomings in the present metrics and tries to remove then as much as possible.

REFERENCES

1. An Introduction to Object oriented Programming and Smalltalk. Pinson Lewis and Richard S. Wiener Addison- Wesley pp 49-60, 1988.
2. S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design," IEEE Transactions on Software Engineering, Vol. 20, No. 6, pp. 476–493, 1994.

3. A Comprehensive Assessment of Object-Oriented, Software Systems Using Metrics Approach, Sanjay Kumar Dubey et al., International Journal on Computer Science and Engineering, Vol. 02, No. 08, 2010, 2726-2730
4. Weyuker's Properties, Language Independency and Object Oriented Metrics, Published in: · Proceeding ICCSA '09 Proceedings of the International Conference on Computational Science and Its Applications: Part II Pages 70 - 81
5. A metrics suite for object oriented design, Software Engineering, IEEE Transactions on, March 1995, Churcher, N.I. ,Shepperd, M.J.; Chidamber, S. ; Kemerer, C.F., Volume 21 , Issue: 3, Pages: 263-265
6. L. Prechelt, B. Unger, M. Philippsen and W. Tichy, “A controlled experiment on inheritance depth as a cost factor for code maintenance”, The Journal of Systems and Software, Vol. 65, 2003, pp. 115-126.
7. M. Alshayeb, and M. Li, “An Empirical Validation of Object-Oriented Metrics in Two Different Iterative Software Processes”, IEEE Transactions on Software Engineering archive, Vol. 29, 2003, pp.1043 – 1049.
8. M. Cartwright, An Empirical view of inheritance, Information and Software Technology, Vol. 40, No. 4, 1998, pp. 795-799.
9. M. Tang, M. Kao and M. Chen, An Empirical Study on Object-Oriented Metrics, 6th IEEE International Symposium on Software Metrics, 1998.
10. Impact of Software Metrics on Object-Oriented Software Development Life Cycle, International Journal of Engineering Science and Technology, Vol.2 (2), 2010, 67-76
11. Factor analysis of source code metrics, D Coupal, Journal of Systems and Software, 1990 – Elsevier
12. Regression Testing Using Coupling and Genetic Algorithms, IJCSIT 3(1) ,Pages : 3255 – 3259, Harsh Bhasin, Manoj