Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 3, March 2014, pg.650 – 654

RESEARCH ARTICLE

A Metric Level Analysis for Aspect Oriented Programs

Annu
Student, M.Tech
Deptt. Of Computer Sc., Baba Mast Nath Engg. College
Rohtak, Haryana
cse.annu@gmail.com

Sunita
Asstt. Professor,
Deptt. Of Computer Sc. & App., College,
Shri Baba Mastnath Engineering College,
Rohtak, Haryana
Kashisuni5@gmail.com

Abstract*:* To analyze the quality or the reliability of a software system, it is required to measure the software system in quantized form. Software metrics is one of the most traditional and effective way to measure the software system. Different kind of software system such as object oriented, aspect oriented or the embedded system, all requires some software metrics to analyze the software quality and complexity using software metrics. In this paper, some of the common aspects of software metrics are been discussed in this section. The paper has defined the some core metrics associated with any software system.

Keywords: Aspect Oriented, Software Measurement, Software Metrics, Functional Point

I. INTRODUCTION

The software measurement process should be an objective, a method for quantifying, assessing, adjusting, and finally improving the development process. Data is collected on the basis of development issues, concerns, and questions. Then analysis is done according to the software development process and products. The measurement process is used to find the quality, progress, and the performance of the software throughout all life cycle phases. This measurement process consists of collecting and receiving the actual data (not just graphs or indicators), and then analysis is done on that data[1][2]. The key components of an effective measurement process are:

- The software development issues are clearly defined and the measures (data elements) are required to provide insight into those issues.

- Processing of data which has been collected into graphical or tabular reports (indicators) to help in issue analysis.
- Indicators are analyzed to provide insight into development issues
- Implement of process improvements is done on the basis of analyzed results and then new issues and problems are identified.

A) Software Measurement

Measurement is introduced by information technology organizations to better understand, evaluate, control and predict software processes. Measurement as the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules(Gurdev Singh et al. 2011). Measurement can be divided into two ways: - (a) Direct measures: It includes software process (e.g., cost and effort applied) and product (e.g., lines of code (LOC) which are produced, execution speed, and defect rate)[3][4].
(b) Indirect measures: It includes the product functionality, complexity, efficiency, reliability, maintainability and many others(R. S. Pressman,2005). In software engineering, a measure gives a quantitative indication of the amount, dimension, capacity, or size of some attribute of a process or product. Measurement is to determine a measure. The ultimate goal of software measurement is not just to find measures, its goal also consists to build and validate the hypotheses and increase the body of knowledge about software engineering. This body of knowledge is further used to understand, monitor, control, and improve the software products and processes. So to build measure is a necessary part of measurement, but it's not the end result. Measurement is used to find and predict the quality of the current product or process. It is also used to improve the quality of product/process[5][6].

B) Software Metrics

Metrics which are used to evaluate the software processes, products and services is referred to as the software metrics. Metrics can help the management as well as the engineers to maintain their focus on their goals. Software metrics programs should be designed as such that they give specific information which is necessary to manage software projects and enhance the quality software engineering processes and services. Organizational, projects, and the goals are determined in advance and then the metrics are taken on the basis of those goals. These metrics are used to determine the effectiveness in achieving the goals. Traditional design techniques separate data and procedures on the other hand object-oriented designs combine these. There are multiple dimensions which are there in OO metric if it is to provide accurate effort prediction or productivity tracking. It is necessary to measure the amount of raw functionality a software offers, at the same time it is equally important to include information about communication which is there between the objects and reuse through inheritance in the 'size' as well[6][7].

In this section, the exploration of the Software measurement process is defined. The metric based analysis of the software system for aspect oriented programs is discussed in this section. In section II, the work defined by the earlier researchers is discussed. In section III, different Metrics associated with aspect oriented programming is discussed. In section IV, the conclusion obtained from the work is presented.

II. CLASSIFICATION OF SOFTWARE METRICS

Software metrics are standards to determine the size of an attribute of a software product and at the same time to evaluate it(R. S. Pressman,2005). In different phases of software, metrics can be grouped as Process metrics, Product metrics, and Resource metrics. These metrics are described as follows: -

A) Product type metrics:

It measures the products of software. Example: source code and design documents. There is a wide range of metrics to measure the software product. Product metrics are defined as the measure of product's external attributes or its internal attributes (Gurdev Singh,2011). Product performance is basically the measure of external attributes in the actual environment where the product has to actually run. These measures consist of software usability and re-usability, portability and efficiency. Example of internal product attributes are: size of the software, correctness, complexity, bugs and testability[8].

B) Process type metrics:

It measures the development of software process. Ex: type of methodology, overall development time, number of changes made to documents and number of bugs recovered during testing. An example of process metrics is Source Line of Code (SLOC) metric(Gurdev Singh,2011). This metric only count the lines of source code, by counting it can give an indication of effort made to develop that code.

C) Resource metrics:

These metrics are more concerned with the managers for the estimation of resources which are required for software project. These resources are man-powers (developers), physical resources, for example, material, computers and methods. These metrics can be also classified under process metrics class(R. S. Pressman,2005).

III SOFTWARE METRICS

Metrics are a tool for quality control and project management. Procedure oriented metrics are used to measure different attributes of a project or the smaller piece of code. Example, a metric available may measure the number of code lines, the amount of comments and the complexity of code. Metrics are helpful in finding areas that are more prone to problems. Metrics can be tracked across multiple team or they can be used to monitor the development of a single system[9][10][11].

A) LOC

Lines of code (or LOC) are the most widely used metric for measuring the size of the program. This metric is used to measure the quantitative characteristics of program source code. This metric is used to count the lines of the source code(R. S. Pressman,2005). LOC is defined as the count of any line that is not a blank or comment line, irrespective of the number of statements per line. LOC is programming language dependent and is commonly used software size metric. Levitin has stated that LOC is a poorer measure of size than Halstead's program length, N.

B) Functional Point

This metric was developed by Albrecht(Albrecht,1983). He has proposed a measure of software size which can be determined early in the development process. Function points are intended to be a measure of program size and the effort required for development.

Complexity function types	Low	Average	High
External inputs	3	4	6
External outputs	4	5	7
Internal logical files	7	10	15
External interface files	5	7	10
External inquiries	3	4	6

Table1. Weighting Factors

This approach calculates the total function points (FP) value for the project, which is dependent on the counts of distinct (in terms of format or processing logic) types in the following five classes input, output, external interface files, internal files, and external inquiries(R. S. Pressman,2005).

$$FP = \text{count total} * [0.65 + 0.01 * (Fi)]$$

Here count total is the sum of all FP entries and Fi (i = 1 to 14) are "complexity adjustment values."

After the calculation of function points they are used in a manner analogous to LOC as a way to normalize measures for software quality, productivity and the other attributes:

- Errors per FP.
- Defects per FP.
- FP per person-month.

Function points and LOC based metrics are considered to be relatively accurate predictors of software development effort and cost.

C) Bang Metrics

The Bang metric was first described by DE Marco (DeMarco et al. 1986). As the function the bang metric, point metric can be used to develop an indication of the size of the system. It measures the total functionality of the software system which is being provided to the user. Bang metrics can be calculated from particular algorithm and data primitives available from a set of formal specifications for the software. Various primitives are Functional primitives (FuP), Data elements (DE), Objects (OB), Relationships (RE), Status (ST), and Transitions (TR).

D) Cyclomatic Complexity

Thomas McCabe introduced a metric in 1976 which was based on the control flow structure of a program(Albrecht,1983). This metric is called as the McCabe cyclomatic complexity and it is the most widely used complexity metric throughout. Given any computer program, McCabe method maps a program to connected an a directed graph. The nodes in the graph are used to represent decision or control statements. The edges represent the control paths which define the program flow(T. J. McCabe,1976). Cyclomatic complexity is calculated by the following method:

$$V(G) = e - n + 2$$

Where *e* represents the number of edges, and *n* are the number of nodes present in the graph. McCabe proposed that v (G) can be used as a measure of program complexity. McCabe's Cyclomatic complexity metric is related to debugging performance, programming effort and the maintenance effort. Programs which consists of only binary decision nodes we do not construct a program control graph to compute v (G) (R. S. Pressman,2005). We only count the number of predicates or binary nodes and we add one to it.

$$V(G) = p + 1$$

Where, v (G) is defined as the Cyclomatic Complexity, and p here denotes the number of binary nodes or predicates. McCabe compared his Cyclomatic complexity with the frequency of errors and suggested the following relationship between Cyclomatic complexity and code complexity for a function.

Cyclomatic Complexity	**Code Complexity**
1-10	A simple program, without much risk
11-20	More complex, moderate risk
21-50	Complex, high risk
50 +	Untestable, very high risk

Table 2. McCabe Cyclomatic complexity ranges

l prediction of the sentiment feature will be done.

IV. CONCLUSION

In this paper, a study different metrics associated with a software project are discussed. These are the core metrics that are used to analyze the software system for aspect oriented as well as object oriented software systems.

REFERENCES

[1] Albrecht, A. J. and J. E. Gaffney. Jr.(1983) "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation", IEEE Trans. Software Eng. SE-9, Nov. 1983.

[2] Barry W. Boehm, University of Southern California, Ricardo Valerdi, (2008)" Achievements and Challenges in Cocomo- Based Software Resource Estimation", IEEE Transactions on Software Engineering.

[3] Chhikara,Chhillar, and Khatri, (2011)"Evaluating the Impact of Different Types of Inheritance on the Object Oriented Software metrics", International Journal of Enterprise Computing and Business Systems ISSN: 223-8849 Volume 1 Issue 2.

[4] DeMarco, Tom, "Controlling Software Projects", Yourdon Press, New York, 1986.

[5] E Da-wei and Xiamen, (2007)" The Software Complexity Model and Metrics for Object-Oriented",IEEE Transactions on Software Engineering.

[6] Elaine J. Weyuker,(2001)"Evaluating Software Complexity Measures", IEEE Transactions on Software Engineering.

[7] Geoffrey K. Gill and Chris F. Kemerer,(1991)" Cyclomatic Complexity Density and Software Maintenance Productivity", IEEE Transactions on Software Engineering, vol. 17, no. 12, December 1991

[8] Gurdev Singh, Dilbag Singh, and Vikram Singh,(2011)" A Study of Software metrics", IJCEM International Journal of Computational Engineering & Management, Vol. 11, January

[9] Lalji Prasad, and Aditi Nagar, (2009)" Experimental Analysis of Different Metrics (Object-Oriented and Structural) Of Software" First International Conference on Computational Intelligence.

[10] Linda L. Westfall Principle Software Measurement Services Plano, TX 75075," Seven Steps to Designing a Software Metric.

[11] Lou Marco, "Measuring software complexity", Enterprise Systems Journal, 1997.

[12] Michelle Cartwright and Martin Shepperd,(2000)"An Empirical Investigation of an Object-Oriented Software System" IEEE Transactions on Software Engineering.

[13] M.P. Thapaliyal Garima Verma,(2010)"Software Defects and Object Oriented Metrics – An Empirical Analysis", International Journal of Computer Applications (0975-8887) Volume 9– No.5, November 2010.