



A Framework for Evaluating the Procedural and Object Oriented Software Metrics

Dr. Sanjeev Dhawan

Faculty of Computer Science & Engineering
University Institute of Engineering and Technology,
Kurukshetra University, Kurukshetra, Haryana, INDIA.
E-mail: rsdhawan@rediffmail.com

Kiran

Research Scholar
University Institute of Engineering and Technology,
Kurukshetra University, Kurukshetra, Haryana, INDIA.
E-mail: kiran.dhawan54@gmail.com

Abstract: Procedure oriented metrics measure different attributes of a project or smaller pieces of code. For example, a metric may measure the number of code lines, the complexity of code or the amount of comments. In Object Oriented Software Systems thousands of classes are organized into modules or packages. To measure the quality of modularization of Object oriented system a new set of metric are proposed. This paper presents the framework that will collect all the most beneficial metrics to measure the source code. In this paper various Procedural and OO metrics are implemented.

Keywords: Metrics, Procedural Metrics, OO Metrics, API (Application Programming Interface), Non API

I. Introduction

Software Metrics are standards to determine the size of an attribute of a software product and a way to evaluate it. Modern software engineering dictates that software can be organized into a set of modules. A module captures a set of design decisions which are hidden from other modules and the interaction among the modules should primarily be through module interfaces. Modules access each other's resources only through the API [3].

II. Objective

- To analyze of various procedural as well as object oriented software metrics.
- To select the most useful software metrics.
- To design an automation system that will present the software measurement analysis.

III. Metrics

Metrics are used to evaluate the software project [1]. Project based metrics keep track of project maintenance, budgeting etc. Design based metrics describe the complexity, size and robustness of object oriented and keep track of design performance [2].

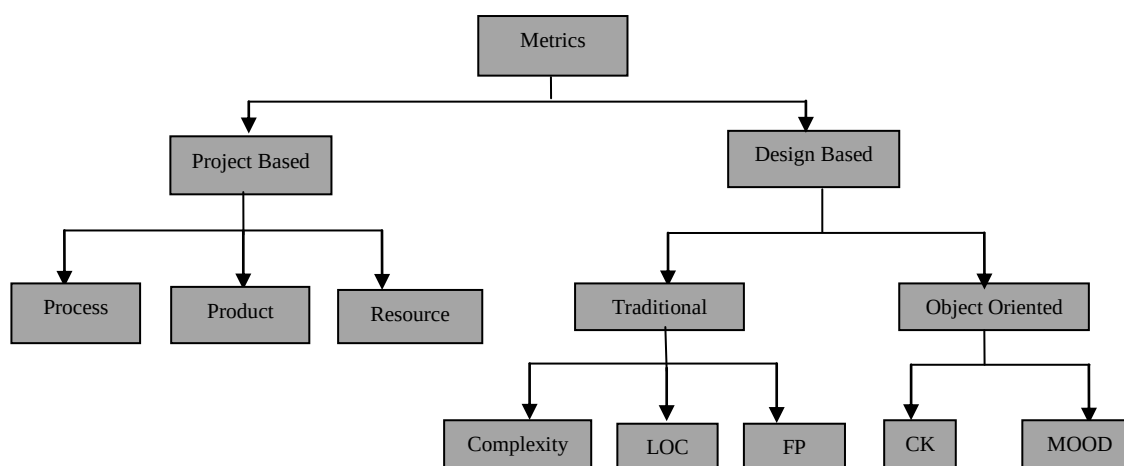


Figure 1 classification of software metrics.

1V. Procedural Metrics

Procedure oriented metrics measure different attributes of a project or smaller pieces of code. For example, a metric may measure the number of code lines, the complexity of code or the amount of comments. Traditional/Procedural metrics have been applied for the measurement of software complexity and size of structured systems since 1976 [8].

- a) Function point (FP): This metric was developed by Albrecht [9] [7]. He has proposed a measure of software size that can be determined early in the development process. Function points are intended to be a measure of program size and, thus, effort required for development.

The approach is to compute the total function points (FP) value for the project, depends 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[4].

$$FP = count\ total * [0.65 + 0.01 * (Fi)],$$

Where count total is the sum of all FP entries. The Fi ($i = 1$ to 14) are "complexity adjustment values" based on responses to the following questions:

1.Data communications	8. Online update
2. Distributed functions	9. Complex processing
3.Performance	10. Reusability
4.Heavily used configuration	11. Installation ease
5. Transaction rate	12. Operational ease
6.Online data entry	13. Multiple sites
7. End-user efficiency	14. Facilitation of change

Table 1.1 adjustment factors.

Each of these questions is answered using a scale that ranges from 0 (not important or applicable) to 5 (absolutely essential).

V. Object Oriented Metrics

Procedural metrics do not capture unique aspects of Object Oriented Programs. Object Oriented Metrics plays a pivotal role in the development of fault free software product. Object Oriented Metrics are mainly designed for Object Oriented Systems [4]. Object oriented design and development are popular concept in today's software development. Object oriented design and development focuses on objects as the prime agents involved in the computation; each class of data and related operations are collected into a single system entity. The main advantage of object oriented design is its modularity and reusability. Object oriented metrics are used to measure properties of object oriented designs. Berard [10] defines five characteristics of object oriented paradigm which differentiates them from traditional structured approach are: localization, encapsulation, and information hiding, inheritance, and object abstraction techniques.

- a) Coupling Based Structural Metrics: This metric is proposed by Chidamber and Kemerer [CHI 94]. Two classes are coupled when methods declared in one class use methods or instance variables of the other class. CBO metric measure the effort required to test the class [6]. Coupling-Based structural metrics provide various measures of the function-call traffic through the API's of the modules in relation to the overall function-call traffic. For that following factors are calculated [11]: (i) Module interaction index , and (ii) Non-API Function Closedness Index

- (i) Module Interaction Index: This metric calculates how effectively a module's API functions are used by the other modules in the system. Suppose module m has n API functions. Also assume that the system S has z numbers of modules from module1 to module z that calls one or more API functions of module [11]. Module Interaction Index (MII) for a given module m and for the entire software system S by

$$MII(m) = (n1+n2+...nz) / (n * z)$$

0, if n i. e. number of API function is zero [5].

If module api is the total number of modules having more than zero API functions. Then

$$MII(S) = SUM [MII(m)i] / module\ api, \text{ Where } i = 1 \text{ to module } api$$

- (ii) Non-API Function Closedness Index: This metric calculates the index factor for module communication and how well API functions of modules are used by the other modules in the system for communication. In this intermediate state, there may exist functions that participate in both intermodule and intramodule call traffic. This traffic can be measured by using a metric "Non-API

Function Closedness Index,” or NC. Let API function are represented as function api and non API functions as function napi for given module [5].

Then total function will be

$$function = function\ api + function\ napi$$

Total number of modules is m.

$$NC(m) = function\ napi / (function - function\ api)$$

0, if the non API functions are zero.

$$NC(S) = SUM [NC(m)i] / M, \text{ Where } i = 1 \text{ to } M$$

In object oriented software, functions will be either API or non API type of functions. And non API functions are not used outside the module. Then function - function api will be equal to function napi. So that NC (m) = 1. Here sometimes the value of the NC (m) can be between 0 and 1.

VI. Implementation

Implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods and evaluation of methods. Therefore, a large number of metrics has been studied. The implementation plan that follows is presented as a sequence of distinct steps. This part presents the implementation of different procedural and object oriented metrics. The implementation of these metrics is carried out using PC with Intel Pentium Dual Core Processor running on 1.60 GHz. The software modules are designed by using C# on .NET platform.

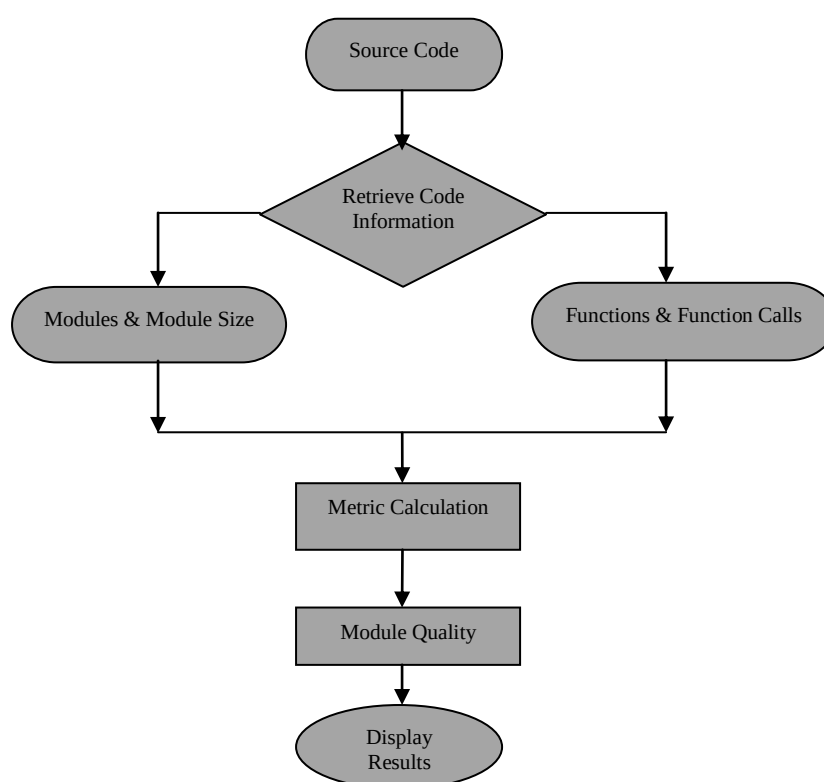


Figure 2 flow chart shows the implementation stage.

The flow chart 2 represents the steps that are taken to implement these metrics and description of these steps is given below:

STEP 1: Getting the source code.

STEP 2: Retrieve code information like number of lines in the code, total number of functions (API and Non API functions), and the number of modules from the code.

STEP 3: Applying the metric to the retrieved code information and calculate the results.

STEP 4: Display the result in the form of snapshots.

VII. Results

The figure 3 represents the working of Function Point which is used to measure the size of the code. Function Point depends on the input, output, external interface files, internal files, and external inquiries. First the size of

the project is defined whether it is small, medium and large then value for the 14 adjustments factors are entered by the user. After submitting these entries, Function Point is calculated by predefined formula. Figure 4 represents the results of Object Oriented Metrics. This snapshot gives complete detail of selected folder like number of files in that folder and the file information its creation time, full path and the result of Coupling Based Structural Metrics. Value of this metric based on the number of lines in the file, number of functions (API, Non API), and on the number of modules. This metric shows how the modules interact with each other through the Application Programming Interface (API).

Figure 3 Function Point analysis.

Figure 4 results of Object Oriented Metric.

VIII. Conclusions and Future Work

In this paper both procedural and object oriented metrics are considered. Procedural oriented metrics are not applicable to object oriented systems. For, object oriented systems different approaches are used to measure the size and complexity of code. Above discussed object oriented metrics shows how the modules interact with each

other through API functions. In this paper the information about the software metrics is extracted from the source program, the work can be extended to derive the information from the object programs. Some other object oriented metrics can also be collected to get the better picture of software analysis.

IX. References

- [1] Linda L. Westfall Principle Software Measurement Services Plano, TX 75075,” Seven Steps to Designing a Software Metric.
- [2] Booch, G: “Object-Oriented Analysis and Design with Applications”, 2nd ed., Benjamin Cummings, 1994
- [3] Sarkar S., Kak A. C. and Rama G. M, “API-Based and Information-Theoretic Metrics for measuring the Quality of SoftwareModularization” IEEE Trans. Software Eng., vol. 33, no. 1, pp.14-30.
- [4] Chhikara, Chhillar, and Khatri,” 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 July 2011.
- [5] Priya Walde and S.V.Kulkarni,” Metrics for measuring the quality of object oriented software modularization”, Emerging Trends in Computer Science and Information Technology -2012(ETCSIT2012)
- [6] Harrison, R., Counsell, S.J. Nithi, R.V: "An Investigation into the Applicability and Validity of Object-Oriented Design Metrics", technical Report.
- [7] Albrecht, A. J. and J. E. Gaffney. Jr. “Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation”, IEEE Trans. Software Eng. SE-9, 6, pp. 639-648, Nov. 1983.
- [8] McCabe and Associates,: “Using McCabe” QA 7.0, 1999, 9861 Broken Land Parkway 4th Floor Columbia, MD 21046
- [9] Albrecht, A.J., “Measuring Application Development Productivity,” Proc. IBM Application Development Symposium, Monterey, CA, October 1979, pp. 83–92.
- [10] Berard, E., Metrics for Object-Oriented Software Engineering, an Internet posting on comp. software-eng, January 28, 1995.
- [11] Chidamber, S.R. and C.F. Kemerer, “A Metrics Suite for Object-Oriented Design,” IEEE Trans. Software Engineering, vol. SE-20, no. 6, June 1994, pp. 476–493.