OBJECT ORIENTED SYSTEM USING SOFTWARE MATRICES

G. Rekha, Research scholar, Dept of CSE, Sunrise University, Alwar, Rajasthan

Dr.R.K Pandey, Supervisor, Dept of CSE, Sunrise University, Alwar, Rajasthan

Abstract

Object oriented design is becoming more popular in software development environment and object oriented design metrics is an essential part of software environment. This study focus on a set of object oriented metrics that can be used to measure the quality of an object oriented design. The metrics for object oriented design focus on measurements that are applied to the class and design characteristics. These measurements permit designers to access the software early in process, making changes that will reduce complexity and improve the continuing capability of the design. This report summarizes the existing metrics, which will guide the designers to support their design. We have categorized metrics and discussed in such a way that novice designers can apply metrics in their design as needed.

Introduction

Object oriented analysis and design focuses on objects as the Primary agents involved in a computation; each class of data and related operations are collected into a single system entity. The meth‟s focus on internal object structures that reflect the Complexity of each individual entity, such as methods and classes, and on external complexity that measures the interactions among entities, such as coupling and inheritance. The metrics measure computational complexity that affects the efficiency of an algorithm and the use of machine resources, as well as Psychological complexity factors that affect the abilip1 of a programmer to create, comprehend, modify and maintain software. But as important as the metrics chosen is What the metrics “tell” the

developers and managers about the quality and object oriented structure of the design and code; metrics Without interpretation guidelines are of little value.

Metrics for object oriented development is a relatively new field of study, in this paper, there are six sections, second section defined overview of object oriented structures, third section defined metrics for object oriented in which three are traditional and ten are new metrics for object oriented, fourth section

defined interpretation guidelines for all metrics, fifth section defined conclusion and in sixth section mention all references.

Literature Review

Lee *et.al.*[1995] defined the *Information Flow-based Coupling* (ICP) as the counts of the sort of measure using information flow between classes but it is also calculated from the class methods not from object invocations.

Li *et.al.* [1993a] defined the *Message Passing Coupling* (MPC) as the count of the number of send statements that is found in methods of one class to other classes. Counting the number of *send* statements does not reflect the actual number (frequency) of execution of that send statement.

Chidamber*et.al.* [Chidamber1994] introduced the *Response for Class* (RFC) as a measure of the number of methods that can potentially be executed in response to a message received by an object of that class. Characterized as "*potentially*", this number does not reflect the actual invocations due to messages received by an object and hence is not a dynamic measure.

Chidamber *et.al.* [Chidamber1994] defined

Coupling between Object classes (CBO) as "*the count of the number of classes to which it is coupled*" and further elaborated in the definition as "*two classes are coupled when methods of one class use methods or instance variables defined by the other class*". Obviously, it is a measure of coupling between object classes not between objects themselves (as the acronym CBO may indicate). This measure is not a dynamic measure of coupling because it does not count the number of invocations during execution, but it counts the number of methods and variables invoked.

Metrics for Object Oriented Systems

The metrics evaluate the object oriented concepts: methods, classes, coupling, and inheritance. This chapter spells out three

aditional metrics and six additional metrics specifically for object oriented systems. Table 2 presents an overview of the object oriented metrics. The first three metrics in Table 2 are examples of traditional metrics applied to the object oriented structure of methods instead of functions or procedures. The next six metrics are specifically for object oriented systems and the object oriented construct applicable is indicated.

A. Traditional Metrics

There are many metrics that are applied to traditional functional development. Here three traditional metrics that is applicable to object oriented development: Complexity, Size, and Readability. To measure the

complexity, the cyclomatic complexity is used.

1) Cyclomatic Complexity (CC) Cyclomatic complexity (McCabe) is used to evaluate the complexity of an algorithm in a method. It is a count of the number of test cases that are needed to test the method comprehensively. The formula for calculating the cyclomatic complexity is the number of edges minus the number of nodes plus 2. For a sequence where there is only one path, no choices or option, only one test case is needed. An IF loop however, has two choices, if the condition is true, one path is tested; if the condition is false, an alternative path is tested. Figure 2 shows examples of calculations for the cyclomatic complexity for four basic programming structures.

A method with a low cyclomatic complexity is generally better. This may imply decreased testing and increased understandability or that decisions are deferred through message passing, not that the method is not complex. Cyclomatic complexity cannot be used to measure the complexity of a class because of inheritance, but the cyclomatic complexity of individual methods can be combined with other

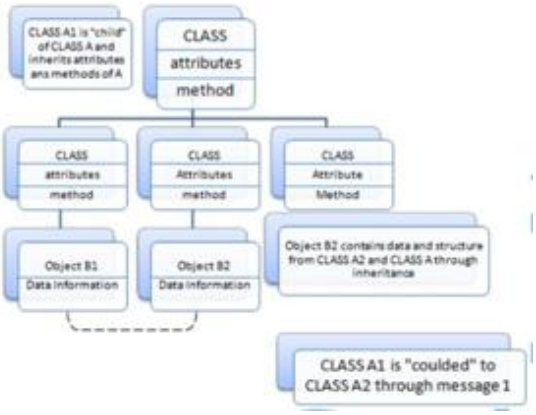

Figure 1: Pictorial Description of Object Oriented Terms

Object-Oriented Specific Metrics

Object-Oriented Analysis and Design of software provide many benefits such as reusability, decomposition of problem into easily understood object and the aiding of future modifications. But the OOAD software development life cycle is not easier than the typical procedural approach. Therefore, it is necessary to provide dependable guidelines that one may follow to help ensure good 00 programming practices and write reliable

code. Object-Oriented programming metrics is an aspect to be considered. Metrics to be a set of standards against which one can measure the effectiveness of Object- Oriented Analysis techniques in the design of a system. 00 metrics which can be applied to analyze source code as an indicator of quality attributes. The source code could be any OO language.

Methodology

The methodology followed in this paper is causal mapping, a qualitative approach used to identify the thought process of individuals related to accomplishing a goal or reaching a decision. The foundations for this approach, pioneered by Axelrod (1976), state that to comprehend the decision making process of experts, we must understand the causal links that they use to reach their decisions.

Enhancements of this technique that make it more productive for business and MIS research have led to modifications of Axelrod's original contribution. For example, Nelson et al. (2000) have developed an approach they call Revealed Causal Mapping (RCM) methodology, which they apply specifically to the identification of factors that constitute expertise in the area of software operations support (code maintenance). Their approach uses the concept of a revealed map, implying that the true causal map for any individual is strictly held within the subject's mind. All we can see and understand is the portion of that map that they choose to reveal. RCM uses traditional interview-ased techniques to gather this data from participants. Collective Causal Mapping Methodology (CCMM) (Scavarda et al., 2006), which is the methodology employed in this research, takes a more virtual approach, using webbased interactions with participants as opposed to traditional interviews. Through webbased interviews and interactions with software maintenance experts, we identify and rank order a set of factors that contribute to corrective maintenance effort. CCMM provides a complete set of guidelines defining the study progression, including how to construct the web-based interview instruments, techniques for coding the resulting unstructured data, and organizing this data into a weighted causal map. The web-based interaction paradigm of the CCMM has certain advantages over a traditional interview-based technique. It allows the researcher to work with a larger, more geographically dispersed pool of experts. The experts can remain completely anonymous, and because all

communication is handled electronically, there are no interactions directly among the respondents. This eliminates the possibility of groupthink, which can negatively impact the exchange of ideas in direct group interaction. The summary of the proposed work is discussed below: OOAD is the technique to measure the quality of the software using different OOAD metrics. OOAD metrics are of numerous to measure the software in different ways. In this proposed work, it is to be developed with new OOAD metrics to determine the quality and error rate of the software to enhance the quality of the software.

Result &Decission

The initial phase of this research produced a total of 17 factors that are reported to impact corrective maintenance effort. These factors are illustrated in Figure 1 below and their definitions are provided in Table B.1 (see Appendix B). The factors in this table are not presented in any rank order, but rather grouped into categories. To define these categories, two researchers independently arranged the factors into groups based on the general characteristics of each factor. The categories produced by the researchers were consistent with each other and therefore the categories were adopted for classification purposes. Table B.1 presents the definition of each node, as well as its relationship to maintenance effort, as reported by the experts who provided input to this study.

While the rank-ordered list, as illustrated in Table 1 (below), is interesting with regard to the categories that emerged at both the top and the bottom of the list, it is not surprising that the weighted standardized response does not exhibit a high degree of variability since all of these factors had been previously identified by experts as causal to maintenance effort.

MIF/AIF are the measure of inheritance which shows relations of generalization and specialization, Increased MIF/AIF will create low under stability and testability of the system. In MOOD Metrics, MHF is having value 0.89 meaning little functionality i.e.,

Interface is provided by classes rather than functionality. Designing of attributes or data hiding is shown by AHF 0.95 which means that using class methods data can be accessed. In the current work MIF value is 1.8 which shows system is less specialized as methods are inherited and functionalities are reused. MIF value 1.8 and AIF value 0.6 shows that reuse of functionality is higher than reuse of information or data. A PF value 0.1 indicates that system uses less polymorphism with this value and it is verified that RMI classes provide reuse of code but it doesn‟t support to multiple functionalities for an operation call. DIT metric value indicates maximum path from root to leaf and in our case the value is 3 which indicate average 3 levels of inheritance hence optimum reuse of code and clear understandability of system (RMI classes). NOC 16 indicates large amount of responsibility associated with a class (average 16 children per class).

MPC, message passing coupling 0 indicates there is no dependency among the classes in RMI. NOA, number of attributes per class 9 and NOM (number of method per class) 15 indicates complex class design. The value 3 of ANA indicates an acceptable design complexity in JAVA RMI classes.

Conclusion

Object oriented metrics exist and do provide valuable information to object oriented developers and project managers. A combination of “traditional” metrics and metrics that measure structures unique to object oriented development most effective. This allows developers to continue to apply metrics that they are familiar with, such as complexity and lines of code to a new development environment. However, new concept is to develop project in object oriented paradigm

METRIC	OBJECTIVE
Cyclomatic Complexity	Low
Lines of Code/Executable Statements	
Comment Percentage	LOW
Weighted Methods per Class	~20—30 %
Response for a Class	LOW
Lack of Cohesion of Methods Cohesion of Methods	Low
Coupling Between Objects	
Depth of Inheritance	Low High
Number of Children	Low
Application Granularity	Low (trade-off)

Average Method Complexity	V Low (trade-off)
Factoring Effectiveness	High
Degree of Cohesion of Objects	Low
	High
	V Low (trade-off)

Table 3: Interpretation Guidelines

and uses inheritance, coupling, cohesion, methods and classes, metrics are needed to evaluate the effectiveness of their application. Metrics such as Weighted Methods per Class, Response for a Class, and Lack of Cohesion are applied to these areas. The application of a hierarchical structure also needs to be evaluated through metrics such as Depth in Tree and Number of Children. At this time there are no clear interpretation guidelines for these metrics although there are guidelines based on common sense and experience.

References

- K.K.Aggarwal and YogeshSingh ,Software Engineering, New Age International (P)Limited, Publishers ,2000.
- Shyam R. Chidamber, Chris F. Kemerer, Towards A Metrics Suite For Object Oriented Design, OOPSLA‟91, pp. 197-211, 1999.
- Henderson-Sellers, L. L. Constantine, I. ANSI/IEEE (1983) „IEEE Standard Glossary of Software Engineering Terminology ANSI/IEEE Standard 729-983.
- Ole-Johan Dahi, Kristen Nygaard, How Object Oriented Programming Started, http://heim.ifi.uio.no i-kristen / FORSKNINGSDOK_MAPPE/F_OO start. html, 2004-06-04.
- David N. Card, Khaled El Emam, Betsy Scaizo, Measurement of Object Oriented Software Development Projects, Software Productivity Consortium, Herndon, Virgina, 2001. -162.
- M. Graham, Coupling and Cohesion (Towards a Valid Metrics Suite for Object Oriented Analysis and Design), Object Oriented Systems Vol 3 pp 143-158, 1996.

OBJECT ORIENTED SYSTEM USING SOFTWARE MATRICES

*****