International Journal of Contemporary Research in Computer Science and Technology (IJCRCST) e-ISSN: 2395-5325
Volume2, Issue 6 (June '2016)

THE METRICS SUITE FOR MAINTAINABILITY MODEL OF ASPECT ORIENTED SYSTEMS

Dr Ananthi Sheshasaayee,
Associate Professor and Head,
PG and Research Department of Computer Science,
Quaid E Millath Govt College for Women,
Chennai,India.

Roby Jose,
Research Scholar,
PG and Research Department of Computer Science,
Quaid E Millath Govt College for Women,
Chennai,India.

Abstract: Aspect Oriented Software Development is gaining wide attention because of its ability to support modularization. The modularization is achieved in aspect Oriented paradigm by the feature separation of concerns that are scattered over the system. Aspect Oriented Software Development takes in software abstractions at new dimensions. Since this is a new paradigm more studies are required to arrive at models that assess the external quality attributes. For the predecessor, object oriented technology; studies have proven that the software metrics are capable of predicting maintainability. So for assessing the maintainability of aspect oriented systems product metrics based models can be employed. This paper attempt to figure out metrics that can be used to build the maintainability model of the aspect oriented software.

Keywords: Aspect Oriented metrics, Aspect Oriented Programming, Maintainability metrics, Software maintenance, Software Product metrics.

I.INTRODUCTION

Design of a maintainable aspect oriented system requires the developer to be aware of the phenomena that are observable while the system concerns are being evolved. One such observational phenomenon is dynamic metrics of software. Since Aspect Oriented Software Development (AOSD) is one of relatively new branch, it should deal with various challenges when compared to the precursor object oriented software development. The static Object Oriented Programming model can be altered to meet the new code prerequisite to accomplish the separation of concerns to achieve modularity.

Thus aspect orientation complements object orientation and not replaces. Aspect Oriented metrics had been used in various studies to assess the strength of the approach as compared to object oriented approach. Till date no studies have suggested with theoretical validation an entire set of metrics that is capable of predicting maintainability of aspect oriented systems, whereas object oriented approach achieved this by means of Li and Henry metrics suite.

The maintenance cost can be controlled if software metrics is utilized during the development phase [6]. Studies [11] performed in the direction of Object Oriented software metrics have shown that metrics can be used as the predictors of maintenance efforts. As Aspect Oriented Software Development is an emerging paradigm a study on maintainability and its associated metrics need to be meticulously performed.

The rest of the paper is organized as follows. Section 2 presents background information about aspect oriented paradigm. Section 3 proposes metrics that can be used for aspect oriented maintainability prediction. The paper is concluded in section 4 with future research indications.

II.ASPECT ORIENTATION

AOSD is a promising paradigm that allows separation of concerns. A concern is a part of the problem that is to be treated as only conceptual unit [8]. These concerns are termed crosscutting concerns as they cut across modularity of other concerns. Aspect Orientation has been proposed as a method for improving separation of concerns [8, 9] in the structure of Object Oriented Software. AOSD uses aspects as new abstraction and make available mechanisms for creating aspects and components at join points. In Aspect-Oriented Programming (AOP) modularization is achieved by means of language abstractions that contribute to the modularization of crosscutting concerns (CCs). Crosscutting Concerns are concerns which cannot be accurately modularized by using traditional programming paradigms [20]. If not with proper language abstractions, crosscutting concerns become scattered and tangled with other parts of the source code thereby affecting quality attributes like maintainability and reusability. In AOP, there exist clear distinction between base concerns and crosscutting concerns. The base concerns (or Core-concerns) are those which the system was originally designed to deal with. The crosscutting concerns are the concerns which affect on other concerns. Some instances of crosscutting concerns include global restrictions, data persistence, authentication and access control.

Aspect-Oriented Programming languages have the flexibility that accede programmers to design and implement crosscutting concern detached from the base concerns. The AOP compiler is provided with the facility to weave the decoupled concerns together in order to attain a correct software system. Even though, there is a complete separation of concerns at the source-code level, the final release delivers the functionality expected by the users. The basic concepts of AOP are illustrated with AspectJ language [7], which is an aspect-oriented extension for Java, allowing the Java code to be compiled seamlessly by the AspectJ compiler. The main constructs in this language are: Aspect - a structure to

represent a crosscutting concern; Pointcut - a rule used to capture join points of other concerns; Join Point- A join point is a point of interest in some object of the software lifecycle through which two or more concerns may be composed; JoinPoint model-A JoinPoint model provides the common frame of reference to enable the definition of the structure of aspects; Advices - types of behavior to be executed when a join point is captured; and intertype declarations - the ability to add static declarations from the outside of the affected code.

III.METRICS SELECTION

Since a software quality prediction model is based on the knowledge stored in the known software metrics, the selection of the specific set of metrics becomes an integral component of the model-building process [3]. In other words, the quality of the software measurement data plays an important role in quality prediction. It has been implied that a software system can be analyzed from several standpoint. To find out the group of metrics is the initial stride in the maintainability assessment using metrics. This set of metrics should be a sign of characteristics of the standpoints with which the system is analyzed. Those metrics that provide redundant information (capable of capturing functionality same as other metrics already in the set) should be discarded.

The position paper is interested on examining the metrics for maintainability at the code level. The study [1] proposed a metrics suite for the maintainability studies of the aspect oriented systems. This set of metrics is listed below.

1. Loose Class Cohesion
2. Concern Diffusion over operation
3. Coupling on Advice execution
4. Concern Diffusion over Lines of code
5. Crosscutting degree of Aspects
6. Coupling on Intercepted modules
7. Coupling on Field access
8. Weighted operations in module
9. Response for a module
10. Coupling between modules
11. Vocabulary size
12. Degree of scattering
13. Concentration
14. Lack of cohesion in operations

The primary aim of this study is to identify a minimal set of metrics that are easy to obtain and useful in predicting software maintainability. With this objective in mind, metrics which are hard to compute (Loose class cohesion, Concern diffusion over lines of code and concern diffusion over operations) were identified. The metric Vocabulary size[4], degree of scattering[2] and concentration [2] pertain to the requirements engineering phase of the software development. The proposed study aims at finding the metrics at the code level capable of predicting maintainability. In the literature it is shown that size [11] influences the quality maintainability. So the metric Lines of class code (LOCC)[13] is added to the primary suite, which was not available in[1]. Following the selection and omission the rough primary set consist of nine metrics which are grouped into Complexity, Coupling, Cohesion and Size.

A.Complexity

The two metrics that fall in this category are Weighted Operations in Module (WOM) and Response for a Module (RFM). WOM counts number of operations in a given module [5]. WOM captures the internal complexity of a module in terms of the number of implemented functions. RFM is number of methods and advices potentially executed in response to a message received by a given module [5].RFM measures the potential communication between the given module and the other ones. The main adaptation necessary to apply it to AOP software is associated with the implicit responses that are triggered whenever a pointcut intercepts an operation of the given module [5].

B.Coupling

The five coupling metrics used in the study are Coupling on Advice Execution (CAE), Coupling on Intercepted module (CIM),Coupling on Field Access(CFA), Crosscutting degree of Aspect(CDA) and Coupling between Modules(CBM).
CDA is a number of modules affected by the pointcuts and by the introductions in a given aspect [5]. CIM is a number of modules or interfaces explicitly named in the pointcuts belonging to a given aspect. CAE is a number of aspects containing advices possibly triggered by the execution of operations in a given module [5]. CFA is a number of modules or interfaces declaring fields that are accessed by a given module [5]. CBM is a number of modules or interfaces declaring methods or fields that are possibly called or accessed by a given module.

C.Size

The lone size metric LOCC counts number of perfect lines of class code including class, aspect, method and advice headers.

D.Cohesion

The single metric in this category also LCOO is number of pairs of operations working on different class fields minus pairs of operations working on common data structure.

Software metrics have a very important role in software development and software quality assurance activities. A software quality prediction model is often based on knowledge extracted from software measurement data. The quality of the underlying software measurement data is thus critical. An intelligent selection of software metrics before training a maintainability prediction model is likely to improve the end result—by removing redundant and less important features. For this machine learning techniques can be involved.

IV.ANALYTICAL EVALUATION PROPERTIES

It is stated that[10] several researches suggest properties that metrics ought to possess. The well known approach for the analytical evaluation of software metrics is the Weyuker criteria [12]. This criterion properly define the informal properties as necessitated by the metrics. The Weyuker criteria [12] are outlined as follows,

(a)Non-coarseness: Non-coarseness property requires that a particular metric value should be different among modules, if not the metrics is not meaningful.

(b)Non-uniqueness: For a metric, two different modules can have the same value is defined by the property non-uniqueness.

(c) Design details are important: The property notify that dissimilarity in design will yield different values for a metric for two different modules.

(d) Monotonicity: The value of a particular metric during the composition of two modules can never be less than the metric values of each individual module.

(e)Non-equivalence of interaction: This property considers that the composition of first and second module result in different values for the same metric when compared with the value for the composition of first and third module.

(f)Interaction increases complexity: When two modules are composed, the metric value can increase is insisted by the property.

V.CONCLUSION

Software maintainability is a continuous and challenging task. The overall cost of maintenance can be reduced by calculating the maintenance at the design level which enables the software developers and maintainers to adjust the software architecture for better performance. A high-quality maintainability model based on metrics enable developers to guide their efforts and make available for them the greatly essential feedback. This paper proposes a primary rough set of metrics that can be used to study the calculation of aspect oriented software maintainability. Further statistical analysis based investigations are required to assess and select a minimal set of metrics which displays preferred properties such as low correlation between one another and those that can be related to the different design properties of the system.

VI.REFERENCES

[1] Saraiva, Juliana, et al. "Aspect-oriented software maintenance metrics: A systematic mapping study." Evaluation & Assessment in Software Engineering (EASE 2012), 16th International Conference on. IET, 2012.

[2] Freitas, Tássia AV, et al. "Metrics for Evaluation of Aspect-Oriented Middleware." Software Engineering, 2009. SBES'09. XXIII Brazilian Symposium on. IEEE, 2009.

[3] Pfleeger SL. Software metrics: Progress after 25 years? IEEE Software 2008; 25(6):32–34.

[4] Mussbacher, Gunter, et al. "Visualizing Aspect-Oriented Goal Models with AoGRL." Requirements Engineering Visualization, 2007. REV 2007. Second International Workshop on. IEEE, 2007.

[5] Ceccato, Mariano, and Paolo Tonella. "Measuring the effects of software aspectization." 1st Workshop on Aspect Reverse Engineering. Vol. 12. 2004.

[6] Bandi, R.K, Vaishnavi V.K.,Turk, D.E.,.Predicting maintenance performance using object-oriented design complexity metrics. IEEE Transaction on Software Engineering 29(1) 2003,77-87

[7] Kiczales, Gregor, et al. "An overview of AspectJ." ECOOP 2001—Object-Oriented Programming. Springer Berlin Heidelberg, 2001. 327-354.

[8] Tarr, Peri, et al. "N degrees of separation: multi-dimensional separation of concerns." Proceedings of the 21st international conference on Software engineering. ACM, 1999.

[9] Kiczales, Gregor, et al. Aspect-oriented programming. 11th European Conference on Object Oriented Programming in:LNCS,vol.1241,Springer Verlag,1997,pp 220-242 1997.

[10] Chidamber, Shyam R., and Chris F. Kemerer. "A metrics suite for object oriented design." Software Engineering, IEEE Transactions on 20.6 (1994): 476-493.

[11] Li, Wei, and Sallie Henry. "Object-oriented metrics that predict maintainability."Journal of systems and software 23.2 (1993): 111-122.

[12] Weyuker, Elaine J. "Evaluating software complexity measures." Software Engineering, IEEE Transactions on 14.9 (1988): 1357-1365.

[13] http://aopmetrics.tigris.org/metrics.html