Correlation of Object-Oriented Design Metrics with Maintainability

Hnin Pwint Phyu
University of Computer Studies, Yangon
ahpwint@gmail.com

Thi Thi Soe Nyunt
University of Computer Studies, Yangon
thithisn@gmail.com

Abstract

Software maintainability prediction enables organizations to predict the maintainability of the software systems and can then help in reducing the maintenance effort and thus, reducing the overall cost and time spent on a software project. Besides, measuring structural design properties of a software system, such as coupling, cohesion, or complexity, is a promising approach towards early quality assessments. To use such measurement effectively, quality models are needed that quantitatively describe how these internal structural properties relate to relevant external system qualities such as reliability or maintainability. In this paper, although the framework of design prediction approach is proposed which consists of two main phases, we only focus on a set of object-oriented design metrics that can be used to evaluate and correlate object- oriented design maintainability.
Key Words: object-oriented design, design prediction maintainability, design metrics

1. Introduction

Software maintainability is defined by IEEE standard glossary of Software Engineering as "the ease with which a software system or component can be modified to correct faults, improve performance or other characteristics, or adapt to a changed environment" [14]. The software maintainability is an important characteristic of a software system which is intended to help in reducing a system's tendency, and to indicate when it becomes cheaper and less risky to rewrite the code instead to change it. In order to identify and fix a fault within software, maintainability is required.

Every software quality model has some characteristics and sub-characteristics, which affect software quality. The quality of the software can be measured by using software quality characteristics. Quality measures of object-oriented code or design artifacts usually involve analyzing the structure of these artifacts with respect to the interdependencies of classes and components as well as their internal elements (e.g., inner classes, data members, methods)[7]. Design metrics play a vital role in helping developers to appreciate design aspects of software i.e. improve software quality and developer productivity [13]. When an object-oriented software becomes bigger and bigger, duplicated elements start to appear, cause decreasing the reliability and maintainability of the software. Thus, the present paper proposes a design prediction approach and emphasizes on software maintainability characteristics with the use of Formal Concept Analysis (FCA), Relational Concept Analysis (RCA), and object-oriented design (OOD) metrics for design prediction. The uses of object-oriented design metrics include inheritance related measures, cohesion measures and coupling measures such as DIT, NOC, MIF, AIF, LCOM and CBO to predict maintainability.

2. Related Work

The maintainability of software is probably the element that can be approached the best at

the level of the software's design and actual code. Therefore, some studies investigated the maintainability characteristics. The paper [11] developed a multivariate linear model 'Maintainability Estimation Model for Object-Oriented software in Design phase' (MEMOOD) and estimates the maintainability of class diagrams in terms of their understandability and modifiability. The overview of various quality models in which maintainability is described and also provides the analysis of maintainability in various quality models is in [14].

Paper [9] aims to predict object- oriented software quality by estimating the number of faults and the number of lines changed per class with the use of object-oriented metrics. The study of paper [15] investigates the connection between design patterns, object-oriented (OO) quality metrics and software maintainability and describes ISO 9126-1 quality model characteristics and sub-characteristics. ISO 9126-1 quality model characteristics and sub-characteristics are applied in the proposed system. Briand and Wust [3] discuss measuring structural design properties of a software system, such as coupling, cohesion, or complexity, is a promising approach towards early quality assessments and also present overview of correlational studies of different metrics.

This paper is organized as follows. Related work of the proposed system is presented in section 2. In section 3, the background theory of design metrics and proposed objected-oriented design metrics are discussed. The architecture of the proposed design prediction approach is detailed in section 4. Then, the framework of correlation with metrics and maintainability is presented in section 5 and conclusion is described in section 6.

3. Design Metrics

In fact, object-oriented development requires not only a different approach to design and implementation, but also a different approach to software metrics. Object-oriented design includes attributes, methods, objects (classes), relationships and class hierarchies. Object-oriented design metrics are essential part of software environment. A set of object-oriented metrics that can be used to measure the quality of an object-oriented design is described below. These metrics look at the quality of the way the system is being built. Design metrics can be divided into

- Traditional Metrics
- Object-oriented Design Metrics.

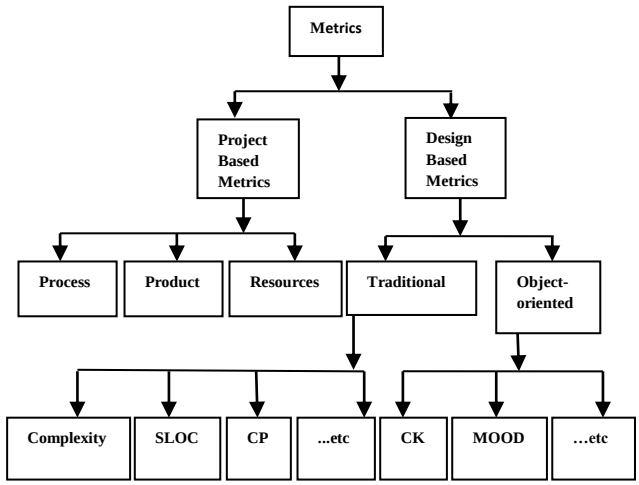

Figure.1. Metrics Hierarchy

SLOC – Line Counts (size)
CP - Comment Percentage
CK - Chidamber and Kemerer
MOOD- Metrics for Object Oriented Design

The metrics for object-oriented design focus on measurements that are applied to the class and design characteristics. For example, metrics proposed by Chidamber & Kemerer metrics (CK metrics, 1994), MOOD metrics, Lorenz and Kidd metrics etcs. CK metrics are the most popular among them. Another comprehensive set of metrics is MOOD metrics. Chidamber and Kemerer proposed six metrics are Weighted Method per Class (WMC), Depth of Inheritance Tree (DIT), Number of Children (NOC), Coupling Between Objects (CBO), Response for a Class (RFC) and Lack of Cohesion in Methods (LCOM).

The MOOD **(**Metrics for Object-Oriented Design**)** metrics set refers to a basic structural mechanism of the OO paradigm as encapsulation, inheritance, polymorphisms, message-passing and are expressed as quotients. The set includes the following metrics are Method Hiding Factor (MHF), Attribute Hiding Factor (AHF), Method Inheritance Factor (MIF), Attribute Inheritance Factor (AIF), Polymorphism Factor (PF) and Coupling Factor (CF).

3.1. The Proposed Object-Oriented Design Metrics for Prediction

The selected object-oriented design metrics are intended to be applied to the concepts of cohesion (classes), coupling, and inheritance for prediction of the maintainability of the class hierarchy design.

3.1.1. Inheritance Metric: Inheritance is a type of relationship among classes that enables programmers to reuse previously defined objects including variables and operators.
Depth of Inheritance Tree (DIT): The DIT will be the maximum length from the node to the root of the tree. DIT is a measure of how many ancestor classes can potentially affect this class.
Number of Children (NOC): The number of children is the number of immediate subclasses subordinate to a class in the hierarchy.
Method Inheritance Factor (MIF): MIF is defined as the ratio of the sum of the inherited methods in all classes of the system under consideration to the total number of available methods for all classes.
Attribute Inheritance Factor (AIF): AIF is defined as the ratio of the sum of inherited attributes in all classes of the system under consideration to the total number of available attributes for all classes.

3.1.2. Cohesion Metric:
Lack of Cohesion in Methods (LCOM) is the number of pairs of methods in the class using no attributes in common, minus the number of pairs of methods that do. Low cohesion increases complexity, thereby increasing the likelihood of errors during the development process.

3.1.3. Coupling Metric:
Coupling between object classes (CBO) is a count of the number of other classes to which a class is coupled. The larger the number of couples, the higher the sensitivity to changes in other parts of the design and therefore maintenance is more difficult.

4. The Architecture of Proposed Design Prediction Approach

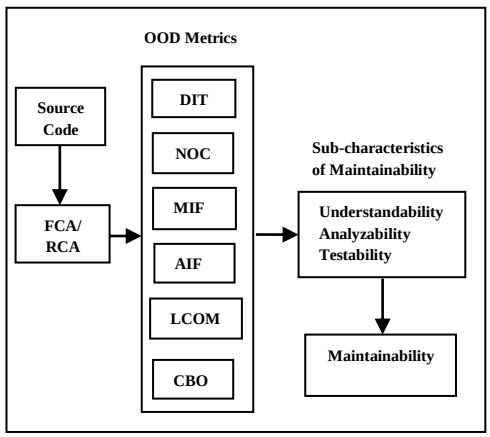

Figure.2. Architecture of Design Prediction Approach

The design prediction approach is proposed which is intended to make the object-oriented design easier to understand, maintain and reuse. The aim of the proposed approach is to predict maintainability of design by making well designed class hierarchy and measuring and correlating with design metrics.

In the first step**,** the proposed system takes as input source code, encodes it into FCA (or RCA) contexts, generates the corresponding concept lattices, and produces the class diagram as output.

Finally, the maintainability of the output class diagram is predicted with object-oriented design metrics especially with inheritance, cohesion,

coupling metrics. In addition, sub- characteristics and characteristics of maintainability are correlated with design metrics.

Formal concepts naturally endow "cohesiveness" because their extents comprise members sharing all the properties in the respective intents. Furthermore, in an attempt to reduce coupling in the resulting OO code, the links between class members are considered. RCA provides a particularly suitable framework for the redistribution because it can discover strongly related sets of individuals with respect to shared properties and inter-individual links and hence supports the search of cohesive subsets of class members.

Cohesion and Coupling can affect the maintainability of the software design. So, design could be improved by redistributing class members among existing or new classes to increase cohesion and/or decrease coupling with the use FCA and RCA. This makes to improve internal quality of the software design that affects maintainability. Sub-characteristics of maintainability such as understandability, analyzability and testability can achieve from measuring with OOD metrics of design. These directly affect maintainability characteristics.

5. The Framework of Correlation with Metrics and Maintainability

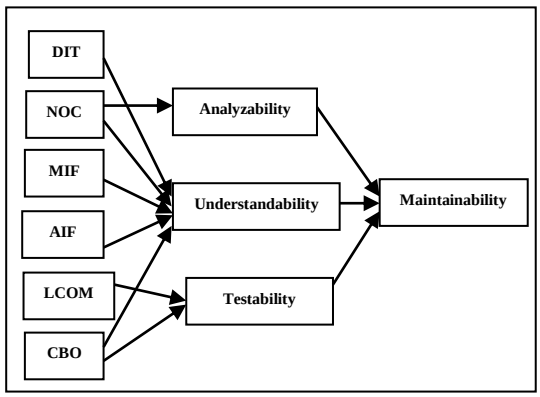

Figure.3. Correlation of Metrics and Maintainability

To correlate maintainability characteristics with understandability, analyzability and testability of class diagram are being quantified in terms of the result of measuring (NOC, DIT, MIF, AIF, CBO and LCOM) design metrics respectively as shown in figure. A set of six metrics with its threshold values are defined to evaluate maintainability of the software design. ISO 9126-1 quality model characteristics and sub-characteristics are applied in this work. Understandability, analyzability and testability are used as sub-characteristics of maintainability because analyzability is the ability to find the reasons of the failures, testability is the verifiability of the new changes in the software and maintainability is affected by understandability.

NOC: This can be considered as a negative impact on understandability and analyzability of the software. NOC measures the breadth of a class hierarchy, where DIT measures the depth. NOC and DIT are closely related. High NOC has been found to indicate fewer faults. The class is potentially influencing a large number of descendant classes. This can be a sign of poor design. So, redesigning may be required.

DIT: Higher DIT decreases understandability and latter indicates that higher DIT increases fault proneness therefore, maintainability.

MIF and AIF: MIF and AIF should be in a reasonable range, not too low and not too high either. Too high a value indicates either superfluous inheritance or too wide member scopes. A low value indicates lack of inheritance or heavy use of Overrides/Shadows. MIF and AIF seem to increase with increasing maintainability.

LCOM: the higher the LCOM, the lower the quality of the system.

CBO: High CBO is undesirable. The larger the number of couples, the higher the sensitivity to changes in other parts of the design, and therefore maintenance is more difficult. High CBO signals poor and complex design, decreases modularity and reuse, complicate testing of the class and as a result decreases understandability and testability. Excessive coupling between object classes is detrimental to modular design and prevents reuse.

6. Conclusion

As object-oriented programming languages and development methodologies moved forward, a significant research effort was also dedicated to defining specific quality measures and building quality models based on those measures. Prediction using object-oriented design metrics should be used for obtaining assurances about software quality. It is believed that predicting the maintainability of the design will help software designers and maintainers to alter the architecture of the software system for better performance that leads to the overall reduction of maintenance time and costs. Moreover, correlation of design metrics with sub-characteristics and characteristics of maintainability is a crucial task for prediction. Thus, this study focuses on object-oriented design metrics (inheritance, cohesion and coupling) that can be worn to measure the quality of an object-oriented design and correlate sub-characteristics with maintainability characteristics. Java Open Source Projects will be analyzed for the proposed system. Then, the prediction approach will be validated in future plan.

References

[1] O. Adekile, "O*bject-Oriented Software Development Effort Prediction Using Design Patterns From Object Interaction Analysis*", 2008.

[2] J. Bansiya, "*A Hierarchical Model for Object-oriented Quality Assessment*", IEEE, 2002.

[3] L. C. Briand, J. Wust, "*Empirical Studies of Quality Models in Object-Oriented Systems.*"

[4] J. Falleri, M. Huchard, C. Nebut, "*A generic approach for class model normalization*."

[5] M. Genero, M. Piattini and C. Calero, "*Early Measures for UML Class Diagrams*", L' Objet Volume 6 – No. 4/2000.

[6] M. Genero and M. Piattini, "*Empirical validation of measures for class diagram structural complexity through controlled experiments.*"

[7] S. Muthanna, K. Kontogiannis, K.Ponnambalam, B. Stacey, "*A Maintainability Model for Industerial Software Systems Using Design Level Metrics*", IEEE, 2000.

[8] C. Neelamegam, Dr. M. Punithavalli, "*A Survey - Object Oriented Quality Metrics*", in Global Journal of Computer Science and Technology, p 183.

[9] T. Quah, M. M. Thet Thwin, "*Application of Neural Networks for Software Quality Prediction Using Object-Oriented Metrics*", IEEE, 2003.

[10] M. Riaz, E. Mendes, E. Tempero, "*A Systematic Review of Software Maintainability Prediction and Metrics*", IEEE, 2009.

[11] S. W. A. Rizvi and R. A. Khan, "*Maintainability Estimation Model for Object-Oriented Software in Design Phase (MEMOOD)*", in Journal of Computing, Volume 2, Issue 4, 2010.

[12] A. Shaik, C. R. K. Reddy, and B. Manda, "*An Empirical Validation of Object Oriented Design Metrics in Object Oriented Systems*", in JETEAS, 2010.

[13] H. A. Sahraoui, R. Godin, T. Miceli, *"Can Metrics Help Bridging the Gap between the Improvement of OO Design Quality and Its Automation."*

[14] R. Saini, S. K. Dubey, A. Rana "*Analytical Study of Mainability Models for Quality Evaluation*", in IJCSE, 2011.

[15] T. Turk, "*The Effect of Software Design Patterns on Object- Oriented Software Quality and Maintainability*", 2009.