EVALUATING THE COMPLEXITY OF UML CLASS DIAGRAMS

AIDA SHAFIABADY

A MASTER PROJECT SUBMITTED IN FULFILLMENT OF THE
REQUIREMENTS FOR THE AWARDS OF THE DEGREE OF
MASTER OF SOFTWARE ENGINEERING

Advanced Informatics School
Universiti Teknologi Malaysia

January 2013

ACKNOWLEDGMENT

It has been impossible to write this Master thesis without the assistance and support of people around me whom it is not possible to thank and mention all of them here.

Above all, I want to thank my academic mentor Madam Haslina Sarkan and my industrial mentor Dr. Mohd Naz'ri Mahrin for their good advice, help, support and patience. I am deeply appreciative and working with them has been a real pleasure to me. I also want to thank them for their knowledge and listening to my ideas and discuss with me about them. I also want to thank them for being so caring and kind to me, you made me feel like a friend and I am grateful for that from my heart.

I would like to appreciate academic and technological support of the Universiti Teknologi Malaysia which provided me the library facilities and also gave me this opportunity to continue my studying in higher level.

I also want to thank my fellow friends whom their encouragement throughout this work is valuable. Last, but not means least, I want to thank my dear family for their infinite support. Thank them all with all my heart.

ABSTRACT

One of the main purposes of software engineering is to improve the quality of software products. As we all know if we want to have good and acceptable software in quality criteria we must spot it from the early phases of the development life cycle. One of the key artifacts in theoretical modeling phase is class diagrams in which their quality has an important impact on the quality of our system. If they have low quality they will lead to so many problems, for instance, the construction cost will be more than the estimated one, so if we measure the quality of class diagrams we can find out if they have low or high quality and then we try to eliminate the problem in those diagrams.

One of the ways for evaluating the quality of UML class diagrams is to measure the complexity of those classes. In this thesis we tried to recommend and present a way of measuring the complexity of class diagrams with respect to the complexity metrics based on the relationship between classes. This method of measuring has many good assets and can measure the complexity of each class diagram independently by using the tool based on fuzzy logics.

ABSTRAK

Salah satu tujuan utama kejuruteraan perisian adalah untuk meningkatkan kualiti produk perisian. Seperti yang kita semua tahu jika kita ingin mempunyai perisian yang baik dan boleh diterima dalam kriteria kualiti kita mesti melihat dari fasa awal kitaran hayat pembangunan. Salah satu artifak penting dalam fasa pemodelan teori rajah kelas di mana kualiti mereka mempunyai kesan penting terhadap kualiti sistem kami. Jika mereka mempunyai kualiti yang rendah mereka akan membawa kepada banyak masalah, misalnya, kos pembinaan akan menjadi lebih daripada satu anggaran, jadi jika kita mengukur kualiti gambar rajah kelas kita boleh mengetahui jika mereka mempunyai kualiti yang rendah atau tinggi dan kemudian kita cuba untuk menghapuskan masalah dalam orang-orang rajah.

Salah satu cara untuk menilai kualiti gambar rajah UML kelas adalah untuk mengukur kerumitan kelas-kelas. Dalam tesis ini, kami cuba untuk mencadangkan dan membentangkan satu cara mengukur kerumitan rajah kelas berkenaan metrik kerumitan berdasarkan hubungan antara kelas. Ini kaedah mengukur mempunyai banyak aset yang baik dan boleh mengukur kerumitan setiap rajah kelas bebas dengan menggunakan alat yang berdasarkan bukti kabur.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

LIST OFAPPENDICES

LIST OF ABBREVATION

FDM	-	Fuzzy Decision Making
GMP	-	Generalized Modus Ponens
OBA	-	On-Board Automobile
OCL	-	Object Constraint Language
OMG	-	Object Management Group
OMT	-	Object Modeling Technique
OO	-	Object Oriented
OOIS	-	Object Oriented Information System
OOSE	-	Object Oriented Software Engineering
RUP	-	Rational Unified Process
SDD	-	Software Design Document
SLR	-	Systematic Literature Review
UML	-	Unified Modeling Language

CHAPTER 1

PROJECT OVERVIEW

1.1 Introduction

In today's competitive markets, delivering software which has high quality is not a benefit but it is one of the essentials of being successful among the other companies. For having high quality software system we have to make sure that it has been qualified from the early phases of its development life cycle. If we apply quality assurance methods at initial phases it will be more effective than applying them after the implementation of the system.

One of the most significant dependence of the quality of object-oriented (OO) software systems is the exactness of the requirements specification (Genero et al., 2000). As a result we have to focus on making the models which created in the early steps of the software development life cycle better. Class diagrams are one of the main objects in the OO model and development; it also presents the basis of the design work and the implementation process of the software. So, we can say that the quality of a system performance has influenced by the quality of the class diagrams.

1.2 Background of the Company

Universiti Teknologi Malaysia (UTM) is the oldest educational institute in Malaysia which is specializes in the field of technology and engineering. This university starts working since 1904. It has two main campuses; one of them is in Skudai which is the first university in Johor Bahru and it is the second largest public university in Malaysia. The other campuse is located in Kuala Lampur and it has an area of almost about 17 hectares and it has so many faculties and schools such as Advanced Informatics School (AIS).

AIS previously known as Centre for Advanced Software Engineering (CASE) which is established in 1996. In the year 2008, UTM became one of the best universities in Malaysia. During the process of improvement CASE has been promoted to the school with faculty authority and it was called Advanced Informatics School (AIS).

UTM AIS suggested both master and doctorate levels programs. We summarized all the programs which offered by UTM AIS in table 1.1.

AIS is an international campus and consist of graduates from all over the world such as Indonesia, Iran, Iraq, Libya, Sudan, Yemen, Zambia and other countries.

Table 1.1: Program offered by UTM AIS

Master Programs	Doctorate Programs
• Master of Software Engineering • Master of Science (Information Assurance) • Master of Science (Computer Systems Engineering) • Master of Philosophy	• Doctor of Software Engineering • Doctor of Philosophy

The Dean of UTM AIS is Prof. Dr. Shamsul Sahibuddin and he has this position since 2006[1]. Figure 1.1 shows organizational chart of UTM AIS.

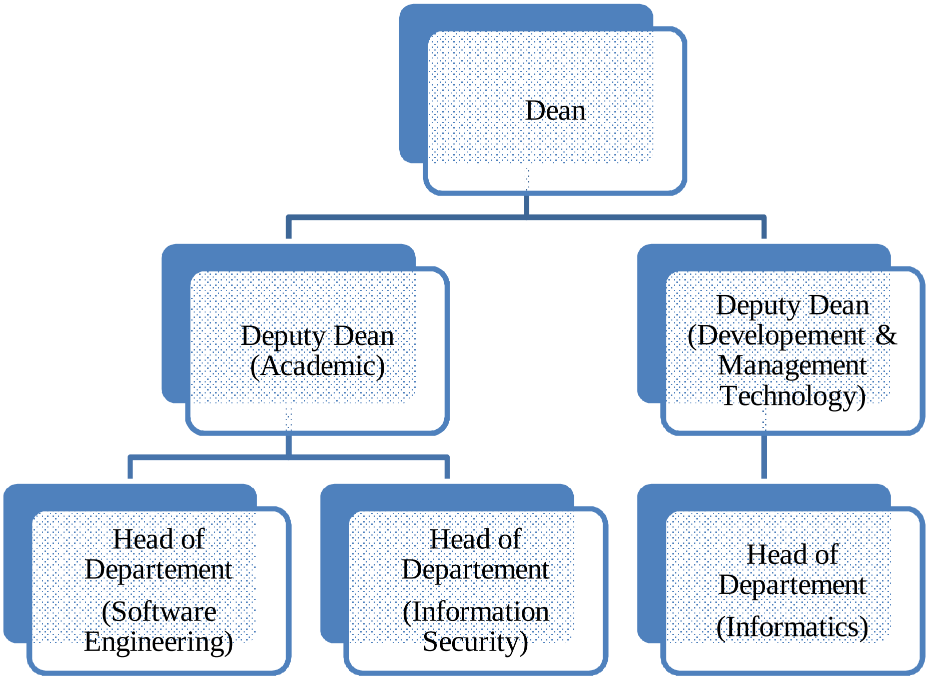
Dean
Deputy Dean (Academic)
Deputy Dean (Developement & Management Technology)
Head of Departement (Software Engineering)
Head of Departement (Information Security)
Head of Departement (Informatics)

Figure 1.1: Organizational Chart

[1] http://www.ais.utm.my/

1.3 Background of the problem

Unified Modeling Language (UML) is a all purpose standard modeling language which has a background in the object-oriented software engineering. This standard was developed by the Object Management Group (OMG), and was first appended to the list of OMG accepted technologies in 1997 (Han et al., 2009), and from that time become the industry standard for modeling software-intensive systems.

Among the 14 different types of diagrams used in UML, half of them employed to display structural information, and the other half represents general type of behavior four of which allocated to different aspects of interactions.[2]

UML not only has the capability to be exploited for forward engineering, but also beneficial on reverse engineering. This ability comes from its graphical representation of a software system (Ben-Abdallah et al., 2004). One of basic uses of UML class diagrams is to illuminate the structure of a software system that mostly leads to an appropriate understanding of it. UML class diagrams model the classes and their various relationships, such as generalization (inheritance), association (aggregation and composition), and other dependencies.

Quality in software products is characterized by the presence of different external and internal attributes (Genero et al., 2004) which external quality attributes are those attributes that can be measured with respect to how the product relates to its environment such as functionality, reliability, usability but an internal quality attributes are those that

[2] Structural diagrams are Package Diagram, Component Diagram, Object Diagram, Deployment Diagram, Class Diagram, Composite Structure Diagram, Profile Diagram. And behavioral diagrams are Use Case Diagram, State Machine Diagram, Activity Diagram, Interaction Overview Diagram, Communication Diagram, Sequence Diagram, Timing Diagram.

can be measured purely in terms of the product and can be measured by examining the product on its own, separate from its behavior such as complexity.

For having a good software design, software must be planned from early stages. For example, if a class diagram is not designed with sufficient accuracy, we could not reach to the ultimate goal of the project, so quality checks can be done on these diagrams to assess the upcoming properties of the object-oriented information systems, such as business and other kinds of database applications, to be developed. The quality is usually assessed during the early development phase like requirement specification and conceptual modeling to avoid poor quality models that will result in inadequate implementation problems.

1.4 Problem Statement

Building software models before implementing them has become widely accepted in the software industry. Object models, graphically represented by class diagrams, and their quality can have a significant impact on the quality of the software which is implemented (Zhou and Xu, 2005). The problem here is sometimes the final software construction cost is more than what we estimated and also it has a low quality and this problem is originated from the design work, to avoid these matters we have to find early indicators of quality based such as class diagrams, and here is the basis where evaluation is necessary because it can allow us to investigate the complexity of such diagrams in initial phases.

1.5 Objectives of the study

This Master report aims:

I. To investigate the current UML class diagram complexity measurement.
II. To apply and evaluate the effectiveness of current UML class diagrams complexity.
III. To improve the complexity evaluation of the UML class diagram.

1.6 Scope of the study

This research is focused on evaluating the quality based on the complexity of the UML class diagrams which describes the structure of a system by showing the system's classes, their attributes, and the relationship among the classes. We focus our work on UML class diagram complexity measurement.

In this research the first step consists of investigation, where we will try to find out and clear about the methods that software engineers use for measuring the complexity of class diagrams in UML. The next step is to apply and evaluate the effectiveness of current UML class diagrams complexity, in this part we will see if the current methods are meaningful and sufficient enough for evaluating the complexity of class diagrams. And at the end we will go to analysis the result of evaluating the complexity on class diagrams based on the On-Board Automobile (OBA) project so our case study in this thesis is OBA’s class diagrams of seven groups.

1.7 Project Deliverables

Each of our objectives has their own result and deliverable that you can see them all in table 1.2.

Table 1.2: Project Deliverables

NO.	OBJECTIVE	DELIVERABLE
1	To investigate the current UML class diagram complexity measurement	Report
2	To apply and evaluate the effectiveness of current UML class diagrams complexity	Report of the results
3	To improve the complexity evaluation of the UML class diagram	An improved method

REFERENCES

Ali, S., Briand, L. C., Hemmati, H., and Panesar-Walawege, R. K. (2010). A systematic review of the application and empirical investigation of search-based test case generation. *Software Engineering, IEEE Transactions on.* 36(6), 742-762.

Bagheri, E., and Gasevic, D. (2011). Assessing the maintainability of software product line feature models using structural metrics. *Software Quality Journal.* 19(3), 579-612.

Bakht, A. J., Rad, M. F., and Akbari, F. (2010). Using Fuzzy Multiple Criteria Decision Making in Evaluation of Software Quality. *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*. 1-7.

Basili, V. R., Briand, L. C., and Melo, W. L. (1996). A validation of object-oriented design metrics as quality indicators. *Software Engineering, IEEE Transactions on.* 22(10), 751-761.

Bellman, R. E., and Zadeh, L. A. (1970). Decision-making in a fuzzy environment. *Management science.* 17(4), B-141-B-164.

Ben-Abdallah, H., Bouassida, N., Gargouri, F., and Ben-Hamadou, A. (2004). A uml based framework design method. *Journal of Object Technology.* 3(8), 97-120.

Bolboaca, S. D., and Jantschi, L. (2006). Pearson versus Spearman, Kendall's tau correlation analysis on structure-activity relationships of biologic active compounds. *Leonardo Journal of Sciences.* 5(9), 179-200.

Booch, G., Maksimchuk, R., Engle, M., Young, B., Conallen, J., and Houston, K. (2007). *Object-oriented analysis and design with applications*. Addison-Wesley Professional.

Briand, L. C., Morasca, S., and Basili, V. R. (1996). Property-based software engineering measurement. *Software Engineering, IEEE Transactions on.* 22(1), 68-86.

Briand, L. C., Morasca, S., and Basili, V. R. (2002). An operational process for goal-driven definition of measures. *Software Engineering, IEEE Transactions on.* 28(12), 1106-1125.

Brito e Abreu, F., and Melo, W. (1996). Evaluating the impact of object-oriented design on software quality. *Software Metrics Symposium, 1996., Proceedings of the 3rd International*. 90-99.

Cabot, J., Clarisó, R., and Riera, D. (2008). Verification of UML/OCL class diagrams using constraint programming 73-80.

Chidamber, S. R., and Kemerer, C. F. (1994). A metrics suite for object oriented design. *Software Engineering, IEEE Transactions on.* 20(6), 476-493.

Dawson, C. W. (2000). *The essence of Computing projects: a student's guide*. Prentice Hall.

Eick, S. G. (1997). Information visualization. *IEEE Computer Graphics and Applications.* 272, 97.

Farhad, J. (2002). The UML Extension Mechanisms. *Department of Computer Science, University College London*.

Garcıa, Y. A. I. (2009). Complexity boundaries for full satisfiability. *A A.* 1, 1.

Genero, M., Fernandez, A. M., Nelson, H. J., Poels, G., and Piattini, M. (2009). A Systematic Literature Review on the Quality of UML Models. *Working Papers of Faculty of Economics and Business Administration, Ghent University, Belgium*.

Genero, M., Manso, E., Visaggio, A., Canfora, G., and Piattini, M. (2007). Building measure-based prediction models for UML class diagram maintainability. *Empirical Software Engineering.* 12(5), 517-549.

Genero, M., Piattini-Velthuis, M., Cruz-Lemus, J. A., and Reynoso, L. (2004). Metrics for UML models. *UML and Model Engineering.* 5(1), 43.12.

Genero, M., and Piattini, M. (2001). Empirical validation of measures for class diagram structural complexity through controlled experiments.

Genero, M., Piattini, M., and Calero, C. (2000). Early measures for UML class diagrams. *L'Objet.* 6(4), 489-505.

Genero, M., Piattini, M., and Calero, C. (2002). Empirical validation of class diagram metrics 195-203.

Genero, M., Piattini, M., and Calero, C. (2005). A survey of metrics for UML class diagrams. *Journal of Object Technology.* 4(9), 59-92.

Han, X., Shang, L., and Bo, W. (2009). A tool for the application of software metrics to UML class diagram 181-184.

Henderson-Sellers, B. (1995). Object-oriented metrics: measures of complexity.

Kaneiwa, K., and Satoh, K. (2010). On the complexities of consistency checking for restricted UML class diagrams. *Theoretical Computer Science.* 411(2), 301-323.

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University.* 33, 2004.

Kitchenham, B., Linkman, S., and Law, D. (1997). DESMET: a methodology for evaluating software engineering methods and tools. *Computing & Control Engineering Journal.* 8(3), 120-126.

Kobryn, C. (1999). UML 2001: a standardization odyssey. *Communications of the ACM.* 42(10), 29-37.

Manso, M., Genero, M., and Piattini, M. (2003). No-redundant metrics for UML class diagram structural complexity 1029-1029.

Marchesi, M. (1998). OOA metrics for the Unified Modeling Language 67-73.

O'Hagan, M. (2000). A fuzzy decision maker. *Proc. Fuzzy Logic '93 (Computer*.

Ortega, M., Pérez, M., and Rojas, T. (2003). Construction of a systemic quality model for evaluating a software product. *Software Quality Journal.* 11(3), 219-242.

Pavlova, M. (2010). http://sourcemaking.com/.

Pilone, D., and Pitman, N. (2005). *UML 2.0 in a Nutshell (In a Nutshell (O'Reilly)).* O'Reilly Media, Inc.

Ribeiro, R. A. (1996). Fuzzy multiple attribute decision making: a review and new preference elicitation techniques. *Fuzzy Sets and Systems.* 78(2), 155-181.

Sheldon, F. T., and Chung, H. (2006). Measuring the complexity of class diagrams in reverse engineering. *Journal of Software Maintenance and Evolution: Research and Practice.* 18(5), 333-350.

Tang, M. H., Kao, M. H., and Chen, M. H. (1999). An empirical study on object-oriented metrics. *Software Metrics Symposium, 1999. Proceedings. Sixth International.* 242-249.

Weyuker, E. J. (1988). Evaluating software complexity measures. *Software Engineering, IEEE Transactions on.* 14(9), 1357-1365.

Yi, T. (2009). Measuring Complexity of Relationships among Classes 1-4.

Yi, T. (2010). Comparison research of two typical UML-class-diagram metrics: Experimental software engineering V12-86-V12-90.

Yi, T., Wu, F., and Gan, C. (2004). A comparison of metrics for UML class diagrams. *ACM SIGSOFT Software Engineering Notes.* 29(5), 1-6.

Zhou, Y., and Xu, B. (2005). Measuring structural complexity for class diagrams: an information theory approach 1679-1683.