UNIVERSITI PUTRA MALAYSIA

SOFTWARE METRICS SELECTION MODEL FOR PREDICTING MAINTAINABILITY OF OBJECT-ORIENTED SOFTWARE USING GENETIC ALGORITHMS

ABUBAKAR DIWANI BAKAR

FSKTM 2016 8

SOFTWARE METRICS SELECTION MODEL FOR PREDICTING MAINTAINABILITY OF OBJECT-ORIENTED SOFTWARE USING GENETIC ALGORITHMS

By

ABUBAKAR DIWANI BAKAR

Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia, in Fulfilment of the Requirements for the Degree of Doctor of Philosophy

March 2016

DEDICATION

This thesis is dedicated to my ever caring parents my late father Diwani Bakar, my mother Amina Kassim Omar Al-bahsany, Abeida Ahmed Alawy Al-baalawy and my late uncle Said Ali Yussuf Al-baalawy, and to my progeny Abdulsalaam, Amina, Said Aisha, Ilhaam, Zuwena and Abeida.

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment of the requirement for the degree of Doctor of Philosophy

SOFTWARE METRICS SELECTION MODEL FOR PREDICTING MAINTAINABILITY OF OBJECT-ORIENTED SOFTWARE USING GENETIC ALGORITHMS

By

ABUBAKAR DIWANI BAKAR

March 2016

Chairman : Abu Bakar Sultan, PhD
Faculty : Computer Science and Information Technology

Software development life cycle maintenance has been advocated as the critical part that consumes more time and resources. To understand the magnitude of the task to maintain the software product, software metrics have been used to make quantification based on their respective software features. To predict software maintainace, the proper metrics need to be selected to avoid the duplication or the outlying of the potential metrics. This is because on one hand, the individual metrics deals with only a single feature of the object-oriented systems, while on the other hand; the suites either contain duplicate metrics of the same goal or lack some important metrics that match the common attributes in the software products. The latest effort to solve this selection problem is the development of the metrics selection model that uses genetic algorithm (GA). However, the process failed to state clearly the encoding strategy in its initial stage.

This thesis clarifies the issue using the objective method to develop the GA metric selection model for predicting the maintainability of object-oriented systems. The study proposes the use of software metric thresholds in the classification process during the GA representation. The software metric thresholds were used as indication for identifying unsafe design in software engineering. To evaluate this technique, an experiment was conducted on two geospatial systems developed using Java programming language where the Chidamber and Kemerer (CK) metrics were used. The proposed technique was also compared to the ranking results from the experts. The comparison results obtained when compared with those of Principal Component Analysis and the complete software metric suite were very promising. Moreover, the three techniques show significant differences in both treatments when compared using analysis of variance.

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Doktor Falsafah

MODEL PEMILIHAN METRIK PERISIAN UNTUK MERAMAL KEBOLEHSELENGGARAAN PERISIAN BERORIENTASI-OBJEK MENGGUNAKAN ALGORITMA GINETIK

Oleh

ABUBAKAR DIWANI BAKAR

Mac 2016

Pengerusi : Abu Bakar Sultan, PhD
Fakulti : Sains Komputer dan Teknologi Maklumat

Penyelenggaraan dalam kitaraan hayat pembangunan perisian telah dikenal pasti sebagai bahagian yang kritikal yang menggunakan lebih banyak masa dan sumber. Untuk memahami peri pentingnya tugas penyelenggaraan produk perisian, metrik perisian telah digunakan untuk membuat pengiraan berdasarkan ciri-ciri relatif perisian. Untuk meramalkan kebolehsengaraan perisian, metrik yang sesuai perlulah dipilih untuk mengelakkan pertindihan atau mengelakkan metrik yang berpotensi daripada tersingkir. Perkara ini adalah kerana di satu pihak, setiap metrik terlibat dengan hanya satu ciri daripada sistem berorientasikan objek, manakala di pihak yang lain pula, kumpulan metriks tersebut sama ada mengandungi metrik pendua dengan matlamat yang sama ataupun ia kekurangan beberapa metrik penting yang sepadan dengan sifat-sifat umum bagi produk-produk perisian. Usaha terkini untuk menyelesaikan masalah pemilihan ini ialah pembangunan model pemilihan metrik yang menggunakan teknik algoritma genetic (GA). Walau bagaimanapun, teknik tersebut gagal menyatakan dengan jelas strategi pengekodan di peringkat awal proses tersebut.

Tesis ini memperkenalkan satu cara untuk menjelaskan isu ini menggunakan kaedah objektif untuk membangunkan model pilihan metrik GA untuk meramalkan penyelenggaraan sistem berorientasikan objek. Kajian ini mencadangkan penggunaan *threshold* metrik perisian dalam proses klasifikasi semasa perwakilan GA. *Threshold* metrik perisian digunakan sebagai sistem penggera untuk mengenal pasti reka bentuk yang tidak selamat dalam kejuruteraan perisian. Untuk menilai teknik ini, eksperimen telah dijalankan ke atas dua sistem geospatial yang telah dibangunkan dengan menggunakan bahasa pengaturcaraan Java, di mana metrik Chidamber dan Kemerer (CK) digunakan. Model ini juga telah dibandingkan dengan keputusan perarafan dari pakar-pakar. Keputusan yang diperolehi amat memberangsangkan apabila dibandingkan dengan Analisis Komponen Utama dan perisian kumpulan metrik metrik yang lengkap. Selain itu, ketiga-tiga teknik menunjukkan perbezaan yang ketara dalam kedua-dua rawatan apabila mereka dibanding menggunakan analisis varians.

ACKNOWLEDGEMENTS

I would like to thank many people whose contribution and collaboration have made this work a success.

First, I thank God the Almighty, for His grace to me that has always been sufficient. HIS mercy has always kept me awake in the morning after the death, and I have through the same been able to successfully face many challenges.

I wish also to acknowledge the good work and guidance from my supervisor, Associate Professor Dr. Abu Bakar Md. Sultan, who provided sound and objective intellectual advice that significantly helped me meet many challenges at all stages of my PhD study. I would like further to acknowledge the work of other members of my Supervisory Committee, particularly, Dr. Hazura Zulzalil and Dr. Jamilah Din for their inavaluable contributions towards the shaping of my research from the sctrach. Special thanks need to go to my mother Amina Kassim for her spiritual and moral support and my children Abdulsalaam, Amina, Said, Aisha, Ilhaam and Zuwena, for being a source of inspiration to me, and especially for their understanding even when we are separated thousands of miles for many years.

Finally, I thank all my friends for their constructive contributions, without which, writing of this work would be extremely difficult.

This thesis was submitted to the Senate of the Universiti Putra Malaysia and has been accepted as fulfilment of the requirement for the degree of Master of Science. The members of the Supervisory Committee were as follows:

Abu Bakar Md Sultan, PhD
Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Chairman)

Hazura Zulzalil, PhD
Associate Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

Jamilah Din, PhD
Senior Lecturer
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

BUJANG BIN KIM HUAT, PhD
Professor and Dean
School of Graduate Studies
Universiti Putra Malaysia

Date:

Declaration by graduate student

I hereby confirm that:

- this thesis is my original work
- quotations, illustrations and citations have been duly referenced
- the thesis has not been submitted previously or comcurrently for any other degree at any institutions
- intellectual property from the thesis and copyright of thesis are fully-owned by Universiti Putra Malaysia, as according to the Universiti Putra Malaysia (Research) Rules 2012;
- written permission must be owned from supervisor and deputy vice –chancellor (Research and innovation) before thesis is published (in the form of written, printed or in electronic form) including books, journals, modules, proceedings, popular writings, seminar papers, manuscripts, posters, reports, lecture notes, learning modules or any other materials as stated in the Universiti Putra Malaysia (Research) Rules 2012;
- there is no plagiarism or data falsification/fabrication in the thesis, and scholarly integrity is upheld as according to the Universiti Putra Malaysia (Graduate Studies) Rules 2003 (Revision 2012-2013) and the Universiti Putra Malaysia (Research) Rules 2012. The thesis has undergone plagiarism detection software

Signature: ________________________ Date: ________________________

Name and Matric No: Abubakar Diwani Bakari, GS29719

Declaration by Members of Supervisory Committee

This is to confirm that:

- the research conducted and the writing of this thesis was under our supervision;
- supervision responsibilities as stated in the Universiti Putra Malaysia (Graduate Studies) Rules 2003 (Revision 2012-2013) were adhered to.

Signature: ______________________________
Name of
Chairman of
Supervisory
Committee: Professor Dr. Abu Bakar Md Sultan

Signature: ______________________________
Name of
Member of
Supervisory
Committee: Associate Professor Dr. Hazura Zulzalil

Signature: ______________________________
Name of
Member of
Supervisory
Committee: Dr. Jamilah Din

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

LIST OF ABBREVIATIONS

AHF	Active hiding factor
ACO	Afferent Coupling
AIF	Attribute inheritance factor
ANOVA	Analysis of Varience
CBO	Coupling between object
CC	Cyclomatic complexity
CMC	Class method complexity
COF	Coupling factor
CK	Chidamber and Kemerer
CKJM	Chidamber and Kemerer Java metric
CTA	Coupling through abstract data type
CTM	Coupling through message passing
DIT	Depth of inheritance tree
EAs	Evolution Algorithms
EP	Evolution programming
ES	Evolution strategy
FN	False Negative
FP	False Positive
GA	Genetic algorithm
GIS	Geographic Information System
IEEE	Institute of Electrical and Electronics Engineering
ISO/IEC	International standard organisation/ International Electronic Cmmission
ISO/IEEE	International standard organisation/Institute of Electrical and Electronics Engineering
LCOM	Lack of cohesion metric
LDA	Linear Discriminant analysis
LOC	Lines of code
MEM	Maintainability estimation model
MHF	Method hiding factor
MIF	Method inheritance factor
MOOD	Model for object-oriented design
NAC	Number of ancestor classes

NCBC	Number of catch blocks per class
NDC	Number of descendent class
NLM	Number of attributes and methods number of local methods
NOC	Number of child
NOM	Number of methods
NOO	Number of Operator Overridden
NORM	Number of Overridden Methods
NPF	Number of Public Fields
NPM	Number of public method
NSM	Number of Static Methods
OGC	Open geospatial consortium
OOS	Object-oriented systems
PC	Principal Component
PCA	Principal component analysis
PF	Polymorphism factor
QMOOD	Quality model of object-oriented design
RFC	Reference for class
SAS	Statistical analysis software
TN	True Negative
TP	True Positive
WMC	Weight method per class

CHAPTER 1

INTRODUCTION

1.1 Background

This chapter explores the method of selecting software metrics to predict the degree of object-oriented software maintainability and related issues. The chapter begins with the background of the information concerning the maintenance phase for software, followed by the explanation on software measurements. The chapter also gives a list of contributions of this study, followed by an outline of the remaining chapters of the thesis. The research main argument here is that software metrics thresholds might be a useful categorisation strategy for evolutionary computation techniques.

As any other engineering measurements, software metrics have become necessary tools for developing quality software products through understanding different quality objectives. The importance of quantifying the quality attributes in software engineering has been identified by Galin (2004) as the baseline for planning and predicting the quality of software for its improvement. Galin points out that "software development methods and measurements are two important allies to ensure that the quality of the software product is fulfilled". Fenton and Pfleeger (1997) go further to describe the best software developers and practitioners as those who use metrics to prove the quality of the software they design before releasing it for public use. Fenton and Pfleeger add that "this practice is used as a means of minimizing defects."

In his recent book ttiled *Why the Program Fails*, Zeller (2009) insists on measuring the software products. He finds that there is no software that contains zero defects, and the results of these defects are the source of increasing software costs. Many writers, Douce and Layzell, 1997, for example, have claimed that "much of those costs occur during the maintenance phase of the software development life cycle. About 50% to 70% of time and resources are estimated to be used in the maintenance phase of software development process." In solving the problems of cost, efforts, and time during software maintenance, it is according to Fenton and Neil, 1999, crucial to understand the extent of the handling of the software prior to its adoption. To Fenton and Neil, this can also help practitioners understand the quality of software products based on their quality attributes, which are used in quality modelling to identify the post-release fault proneness.

The nature of any software products, especially those developed using object-oriented concepts is that the code is in fact designed using more than one design attribute. For example, developers use different properties like inheritance, coupling, cohesion and other object-oriented features in the same software product. This approach is similar to the philosophy behind the development of the software metrics, whereby one software metric measures only one attribute. Consequently, it implies that there is a duplication of some metrics, through which more than one metric can be used to measure the same attribute. Therefore, to predict one quality attribute requires several metrics. Then, they need to combine the magnitude, which is essential to predict the constituent software product features.

Software metrics suite was the first attempt to counter the problem of software metrics classification (software metrics selection). Boehm (1975) and Chidamber and Kemerer (1994) were the first researchers to introduce the so-called metric suites, and later followed by other researchers like Abreu and Melo (1996), Elish and Al-Khiaty (2013) and many more. The suites faced the problem of rigidity, which pushed practitioners either to reduce software metrics to remove redundant metrics or to add more metrics to accommodate the missing ones. Michura et al. (2013), for example, made an extension of CK metric suite to conform to his requirements.

In this case where where there are dozens of metrics which can be used to measure one particular software quality attribute, there is a need to have in place mechanism that will help practitioners select software metrics that only suite measurement goals. Wang et al. (2011), suggest the elimination of the number of software metrics that are available for the particular measurement property. In their study, they reduced 98% of the metrics they collected for measuring the quality of software product. Their conclusion is that "the average of three software metrics is good enough to predict the quality of software. Thus, the selection of few suitable metrics out of dozens metrics is recommended to increase performance and to avoid redundant measurement. Therefore, the question that is raised here is how to obtain those few suitable software metrics for predicting the quality of a particular software product, out of dozens of metrics.

The evolution computational intelligence approach, as search-based strategies solves the problems associated with metrics suites. The approaches for selecting the best group of software metrics for predicting the quality of software proved to be promising. Vivanco (2010) uses genetic algorithms as a search-based strategy to solve the mentioned problems. His representation methods of encoding does not states clearly in his model while literature insists on using real data in encoding the chromosome. As by Baggen et al. (2012), insist the use of automatic and rea-world data can build the culture of trust for the final model that it has considered the quality decision on the development of that model.

This study used software metrics threshold (benchmark) ranking strategy in encoding the chromosome to propose the Genetic Algorithm software metrics selection model for predicting object-oriented software maintainability. The use of software metrics thresholds facilitates the performance and improves the accuracy of the model.

1.2 Problem Statement

The selection of the appropriate software metrics in predicting the quality of software is one of the crucial tasks in software engineering paradigm. The problem is more complicated in the availability of dozens of metrics and the inability of proposed groups of software metrics called suites. These collections of metrics failed to give clear and trusted software metrics classification goals for predicting the quality of software due to variability of attributes and features from one software product to another. As Altidor and his colegue in 2009 suggested that "the issue has forced some practitioners to make some modification to satisfy their needs" as Gray, 2008; Michura, 2013, argue. In that regard, there is therefore a need for the practitioners to propose models that can just select list of software metrics for particular software product.

Evolutionary computation, which is used in exhaustive selection problems, is the latest and hard effort employed in software metrics selection. Vivanco, (2010), uses genetic algorithms as a search-based strategy to solve the problem of rigidity identified in software metric suites. Although promising results were obtained, the representation method of encoding was not clearly stated.

The aim of this study is to propose the Genetic Algorithm (GA) software metrics selection model to predict object-oriented software maintainability. In this case, software metrics thresholds were proposed at the presentation phase of the GA. Because of the sensitivity of the availability of software metric thresholds and other challenges, study used the Chidamber and Kemmerer (CK) metrics suite as the representative for the object-oriented maintainability metrics. The results were validated and compared with real maintainability data ranked by experts in order to gain more confidence on proposed solution. The proposed model intends to help practitioners in selecting the most appropriate metrics according to the attributes in particular software, which will then facilitate faster and more accurate adoption of the maintainable software. Out of dozens of software, practitioners will be able to select only the appropriate list of metrics for particular software, which is then used for predicting the maintainability of that software.

1.3 Research Questions

To understand the problem explained above, the reseserch has been guided by the following questions:

1. Which software metrics are suitable to measure the object-oriented software maintainability based on their available thresholds in assumption that software product structure differ from one product to product?

To answer question (1) the review of related work has been conducted together the most appropriate list of software metrics that can be used to predict maintainability of particular object-oriented software products.

2. What are the thresholds for the software metrics identified in the first research question (1)?

Another two reviews were conducted to gather information on the most appropriate software metrics that can be used to predict the maintainability of particular object-oriented software products and to identify their thresholds (benchmark). Results were based on theoretical and experimental validation process in the related literatures.

1.4 Research Objectives

The objective of this research is to propose GA-based software metrics selection model to predict object-oriented software maintainability.

In achieving this main objective, the study proposes the following sub-objectives:

1. To standardardise software metrics thresholds values for the six metrics in Chidamber and Kemmerer (CK) suite to determine the acceptance values.
2. To generate objective thresholds (benchmark) encoding strategy for the representation phase in the GA.
3. To implement GA-based software metrics selection model to predict maintainability of object-oriented software.

1.5 Scope and Limitations of the Research

This research proposed the software metrics selection model using evolutionary algorithm method. GA, which is one of the evolution strategies, is used to classify the software metrics in predicting maintainability of object-oriented software systems. The object-oriented software system is software technology, which involves many measurable features in software engineering. Geographical Information Software (GIS)-based software was used as case studies. Two geospatial software systems have been used to provide the researcher the chance to compare the software metrics selection performances within the same development context. This extends confidence prior to using software products that have been developed using another technology. Moreover, the geospatial software used was developed using open-source software technology. Apart from universal advantages of open-source technology, open access privillage of source code makes it easier to generate software metrics values.

1.6 Contributions of the Research

The use of software metrics thresholds (benchmark) in metrics selection model has the intuitional impacts in software engineering. Having used the software metrics thresholds, the study has successfully developed the software metrics selection model, which facilitates the selection of most effective software metrics in predicting maintainability to accomplish the adoption of the quality software. The use of the threshold prediction model seems to be a promising input in the software engineering paradigm. This encoding strategy has employed the practical use of empirical data that has also shown the great achievement in model performance. Therefore, this new introduced technique supports evolutionary algorithms representation encodings process. In addition, the use of GA as a search technique has put in the philosophy of traditional evolution from biology into the software metrics paradigm using software metrics thresholds.

In addition, upon knowing the maintainability efforts for particular softwares, software practitioners will probably be informed about technical and business aspects of the software product to be adopted. Practitioners who work as software maintainer will easily review the performance of the software product to ensure that the quality criteria have been met. For organizations, Baggen et al. (2012), indicate that “the maintainability models might guide the maintainers in organizing a better way of maintaining the resources”. As for the individual practitioners, they will probably make wiser decisions in adopting the suitable software product that is easily maintained based on available resources.

Moreover, the model can be used by the organisations to evaluate their existing software system to see whether it conforms to the dynamic quality standards from time to time due to error correction or incorporation of the new requirements.

1.7 Organization of the Thesis

The next chapter discusses the previous related literature on software maintenance, software metrics, software metrics threshold and evolutionary algorithms techniques. Chapter 3 discusses methodology used during the research phases. Chapter 4 discusses the preliminary results and provides a discussion on the software metrics thresholds encoding strategy. Chapter 5 presents the proposed models and their validation process. Chapter 6 presents the study findings and discussions. Finally, Chapter 7 concludes the research.

REFERENCES

Abdi H., and Williams, L. J. (2010). Principal component analysis, *Wiley Interdisciplinary Reviews: Computational Statistics, 2*, 433–459.

Abreu, F. B., and Melo, W. (1996). Evaluating the impact of object-oriented design on software quality. *Third International Symposium on Software Metrics: From Measurement to Empirical Results* (Mar 25-26, Berlin, Germany), *3*, 90-99.

Abreu, F., and Carapuca, R. (1994). Object-oriented software engineering: Measuring and controlling the development process. *Proceedings of the 4th International Conference* on Software Quality, McClain, VA, USA, 3-5 October 1994.

Aggarwal, K. K., Singh, Y., Kaur, A., and Malhotra, R. (2008). Application of artificial neural network for predicting maintainability using object oriented metrics. *World Academy of Science, Engineering and Technology, 22*, 1071-1075.

Al-Badareen, A. B., Selamat, M. H., Jabar, M. A., Din, J., and Turaev, S. (2011). The impact of software quality on maintenance process. *International Journal of Computers,* 5, 183-190.

Al-Dallal, J. (2010). Mathematical validation of object-oriented class cohesion metrics. *International Journal of Computers, 4,* 788-804.

Altidor, W., Khoshgoftaar, T. M., and Van Hulse, J. (2009). Empirical study on wrapper-based feature ranking. *International Conference on Tools with Artifficial Intelligence, 2-4 November (ICTAI), 21,* 75-82.

Anda, B. (2007). Assessing software system maintainability using structural measures and expert assessments. *IEEE International Conference on Software maintenance*, October 2-5 (ICSM), 204-213.

Araújo, P., Antonio, M., Travassos, G. H., and Kitchenham, B. (2005). *Evolutive maintenance: Observing object-oriented software decay*. Technical Report, COPPE/UFRJ. Keele University.

Armstrong, D. (2006). The quarks of object-oriented development, *Communications of the ACM,* 49:123–128.

Bäck, T. (1996). *Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms*. New York: Oxford University Press.

Baggen, R., Correia, J. P., Schill, K., and Visser, J. (2012). Standardized code quality benchmarking for improving software maintainability. *Software Quality Journal, 20*, 287–307.

Baker, A. L., Bieman, J. M., Fenton, N., Gustafson, D. A., Melton, A., and Whitty, W. (1990). A philosophy for software measurement. *Journal of Systems and Software, 12*, 277-281.

Bakota, T., Ferenc, R., Gyimóthy, T., Riva, C., and Xu, J. (2006). Towards portable metrics-based models for software maintenance problems. *Proceedings of the 22nd IEEE International Conference on Software Maintenance* (ICSM'06), *2*, 483-486.

Baldassarre, M. T., Bianchi, A., Caivano, D., and Visaggio, C. A. (2003). Full reuse maintenance process for reducing software degradation. *Proceedings of the Seventh European Conference on Software Maintenance and Reengineering, March* 26-28, *7,* 289-298.

Bandyopadhyay, S., and Saha, S. (2013). *Unsupervised classification: Similarity measures, classical and metaheuristic approaches, and applications*. New York: Springer.

Bansiya, J., and Davis, C. G. (2002). A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on Software Engineering, 28*, 4-17.

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

Benlarbi, S., Emam, K., Goel, N., and Rai, S. (2000). Thresholds for object-oriented measures. *Proceedings of the 11th International Symposium on Software Reliability Engineering.* IEEE Computer Society, Washington, DC, USA, (ISSRE) *11*, 24-38.

Bennett, K. H., and Rajlich, V. T. (2000). Software maintenance and evolution: A road map, *Proceeding of the Conference on the Future of Software Engineering, New York, USA*, 73-87

Beyer, H. G. (2001). *The theory of evolution strategies*. New York: Springer.

Boehm, B. W., (1975). Software design and structuring. In E. Horowitz, (Ed.), *Practical strategies for developing large software systems.* Reading, MA: Addison-Wesley.

Bouktif, S., Kegl, B., and Sahraoui, H. (2002). Combining software quality predictive models: An evolutionary approach. *Proceedings of an International Conference on Software Maintenance*, 385-392.

Briand, L., Bunse, L., Daly, J., and Differding, C. (1997). An experimental comparison of the maintainability of object-oriented and structured design documents. *Proceedings of International Conference on Software Maintenance*, 1-3 Oct. 1997, Bari, Italy, 130-138.

Briand, L. C. Wust, J. Daly, J. W. and Porter, D. V. (2000). Exploring the relationships between design measures and software quality in object-oriented systems. *Journal of Systems and Software, 51*, 245-275.

Carey, D. (1996). Is "Software quality" intrinsic, subjective or relational. *Software Engineering Notes*, *21*(1), 74-75.

Cartwright, M., and Shepperd, M. (2000). An empirical investigation of an object-oriented software system. *IEEE Transaction Software Engineering, 26*, 786-796.

Cheong, R. A. (2007). Comparison between genetic algorithms and evolutionary programming based on cutting stock problem. *Engineering Letters*, *14*.

Chhikara, A., Chhillar, R. S., and Khatri, S. (2011). Evaluating the impact of different types of inheritance on the object oriented software metrics. *International Journal of Enterprise Computing and Business Systems, 1*, 1-7.

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

Counsell, S. (2008). An analysis of faulty and fault-free C++ classes using an object-oriented metrics suite. *Innovative Techniques in Instruction Technology, E-learning, E-*assessment*, and Education*, 520–525. Springer Link

D'Ambros, M. (2010). *On the evolution of source code and software defects.* (Unpublished PhD thesis). Università della Svizzera Italiana.

Dagpinar, M., and Jahnke, J. H. (2003). Predicting maintainability with object-oriented metrics – An empirical comparison. *Proceedings of 10th Working Conference on Reverse Engineering, November* 13 - 17 (WCRE), *10*, 155 - 164.

Daly, J., Brooks, A., Miller, J., Roper, M., and Wood, M. (1996). Evaluating inheritance depth on the maintainability of object-oriented software. *Empirical Software Engineering*, *1(*2), 109-132.

Domingos, P. (2012). A few useful things to know about machine learning. *Magazine Communications of ACM, 55*, 78-87.

Douce, C., and Layzell, P. J. (1997). Maintenance of object-oriented C++ software: A protocol study. *IEEE Computer Society 2nd International Workshop on Empirical Studies of Software Maintenance, WESS* 2: 115 - 119.

Dromey, R.G. (1996). Cornering the Chimera [software quality], *IEEE Software,13*, 33-43.

Elish, M., *and* Al-Khiaty, M. (2013). A suite of metrics for quantifying historical changes to predict future change-prone classes in object-oriented software. *Journal of Software: Evolution and Process, 25*, 2013.

Fenton, N. E. (1991). *Software metrics: A rigorous approach*. London: Chapman and Hall.

Fenton, N. E., and Neil, M. (1999). Software metrics: Successes, failures, and new directions. *Journal of Systems and Software, 47*, 149-157.

Fenton, N. E, and Ohlsson, N. (2000). Quantitative analysis of faults and failures in a complex software systems. *IEEE Transactions on Software Engineering, 26*, 797-814.

Fenton, N. E., and Pfleeger, S. L. (1997). *Software metrics: A rigorous and practical approach* (2nd ed.), London: International Thomson Computer Press.

Ferreira, K., Bigonha, M., Bigonha, R., Mendes, L., and Almeida, H. (2012). Identifying thresholds for object-oriented software metrics, *Journal of Systems and Softwar*e, *85*, 244-257.

Fogel, L. J. (1999). *Intelligence through simulated evolution: Forty years of evolution programming*. New York: Wiley-Interscience publication.

Galin, D. (2004). *Software Quality Assurance: From Theory to Implimentation*, Addison Wesley.

Gao, K., Khoshgoftaar, T. M., Wang, H., and Seliya, N. 2011. Choosing software metrics for defect prediction: An investigation on feature selection techniques. *Software: Practice and Experience Software, 41*, 579-606.

Garge, R. K., Sharma, K., and Nagpal, C. (2013). Ranking of Software Engineering Metrics by Fuzzy-based Matrix Methodology. *Software Testing, Verification and Reliability*, 23: 149-168.

Garnett, J. (n.d.). GeoTools user guide. Open Source Geospatial Foundation. Retrieved from http://geotoolsnews.blogspot.com/2011_10_01_archive.html.

Geoserver. (2013). *Geoserver user manual*. Retrieved from http://docs.geoserver. org/stable/en/ developer/

German, D.M. and Hindle, A. (2005). Measuring fine-grained change in software: towards modification-aware change metrics, Software Metrics. *11th IEEE International Symposium*, 10-28

Goel, B. M., Bhatia, P. K. (2012). Analysis of reusability of object-oriented system using CK metrics. *International Journal of Computer Applications,* 60: 32-36.

Gray, A. R., and MacDonell, S. G. (1997). A comparison of techniques for developing predictive models of software metrics. *Information and Software Technology,* 39: 425-437

Gray, L. C. (2008). *A Coupling complexity metric suite for predicting software quality*, Master thesis, California Polytechnic State University.

Guo, L., Ma, Y., Bojan, C., and Singh, H. (2004). Robust prediction of fault-proneness by random forests. *Proceedings of the 15th International Symposium on Software Reliability Engineering* (ISSRE), 417-428.

Gyimothy, T., Ferenc, R., and Siket, I. (2005). Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Transactions Software Engineering,* 31: (10), 897–910.

Halstead, M. (1977). *Elements of software science.* New York: North-Holland.

Hanley, J., and McNeil, B. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143: 29–36.

Harrison, H., Counsell, S., and Nithi, R. (2000). Experimental assessment of the effect of inheritance on the maintainability of object-oriented systems. *The Journal of Systems and Software,* 52: 173-179.

Hassaine, S., Guéhéneuc, Y., Hamel, S., and Antoniol, G. (2012). ADvISE: Architectural decay in software evolution. Published in *Software Maintenance and Reengineerin (CSMR), 2012 16th European Conference on Software Maintenance and Reengineering*, 27-30 March 2012, 16: 267-276.

Hutton, L. (1998). Does OO sync with how we think? *IEEE Software,* 46-54.

Haupt, R. L., and Haupt, S. E. (2004). *Practical genetic algorithms*. New York: John Wiley and Sons.

Heitlager, I., Kuipers, T., and Visser, J. (2007). A practical model for measuring maintainability. *Proceedings of 6th International Conference on Quality of Information and Communications Technology, IEEE Computer Society, Washington,* DC, USA, (QUATIC), 6: 30-39.

Henderson-Sellers, B., Constantine, L., and Graham, I. M. (1996). Coupling and cohesion: Towards a valid metrics suite for object-oriented analysis and design. *Object Oriented Systems*, 3: 143-158.

Herbold, S. Grabowski, J., and Waack, S. (2010). *Calculation and optimization of thresholds for sets of software metrics.* Technical Report, No. IFI-TB-2010-01, ISSN 1611-1044.

Hitz, M., and Montazeri, B. (1996). Chidamber and Kemerer's metrics suite: A measurement theory perspective. *IEEE TSE, 22*, 267–271.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor: The University of Michigan Press.

ISO/IEC 20000-1:2011 *Service management system requirements*. Updated at 2011-04-12 (replacing ISO/IEC 20000-1:2005)

Johny, A. P. (2013). Predicting reliability of software using thresholds of CK metrics. *International Journal of Engineering Research and Technology, 2*, 1136-1145.

Jolliffe, I. T. (1982). A note on the use of principal components in regression. *Journal of the Royal Statistical Society, Series C,* 31: 300–303.

Kanmani, S., Uthariaraj, V. R., Sankaranarayanan, V., and Thambidrai, P. (2007). Object-oriented software fault prediction using neural networks. *Information and Software Technology, 49*, 483-492.

Kaur, S., Singh, S., and Kaur, H. (2013). A quantitative investigation of software metrics thresholds values at acceptable risk level. *International Journal of Engineering Research and Technology*, *2*.

Khosravi, K., and Guéhéneuc, Y. (2005). On issues with software quality model. *Workshop on Quantitative Approaches in Object-Oriented Software Engineering Glasgow, Scotland (ECOOP), 9*, 70-83.

Kitchenham, B. (2010). What's up with software metrics? A *preliminary mapping study, Journal of Systems and Software,* 83: 37–51.

Kitchenham, B., Pfleeger, S. L., and Lawrence, S. (1996). Software quality: The elusive target. *IEEE Software,* 13: 12-21.

Kitchenham, B., Pfleeger, L., and Fenton, N. (1995). Towards a frame-work for software measurement validation. *IEEE Transaction Software Engineering*, 12: 929-944.

Knauf, R., Tsuruta, S., and Gonzalez, A. J. (2005). Overcoming human weaknesses in validation of knowledge-based systems. *Leipziger Informatik-Tage,* 72: 254-263.

Kwon, I., and Lee, J. (1996). Adaptive simulated annealing genetic algorithm for system identification. *Engeneering Application Artificial lntelligence,* 9: 523-532.

Lanza, M., and Marinescu, R. (2006). *Object-oriented metrics in practice: Using software metrics to characterize, evaluate, and improve the design of object-oriented systems*. Heidelberg, Germany: Springer-Verlag.

Lehman, M. M., Ramil, J. F., and Sandler, U. (2001). An Approach to Modelling Long-term Growth Trends in Software Systems. *Proceedings International Conference on Software Maintenance, Florence, Italy*, 219–28.

Li, W. (1998). Another metric suite for object-oriented programming. *Journal of Systems and* Software, 44: 155-162.

Li, W., and Henry, S. (1993). Object-oriented metrics that predict maintainability. *Journal of System Software,* 23: 111-112.

Lincke, R., Lundberg, J., and Löwe, W. (2008). Comparing software metrics tools. *ISSTA '08 Proceedings of the 2008 international symposium on Software testing and analysis*, 131-142.

Linda, P. E., and Chandra, E. (2010). Class break point determination using CK metrics thresholds. *Global Journal of Computer Science and Technology*, *10*.

Lorenz, M., and Kidd, J. (1994). *Object-oriented software metrics*. Englewood Cliffs NJ: Prentice-Hall.

Luijten, B., and Visser, J. (2010). Faster defect resolution with higher technical quality of software. *International workshop on software quality and maintainability (SQM 2010), March 15, 2010, Madrid, Spain, IEEE Computer Society Press* 4: 11-20.

Mago J., and Kaur P. (2012). Analysis of quality of the design of the object-oriented software using fuzzy logic. *IJCA Proceedings on International Conference on Recent Advances and Future Trends in Information Technology, iRAFIT(3), 21-25, April 2012*. Published by Foundation of Computer Science, New York, USA.

Malhotra, R., and Chang, A. (2012). Software maintainability prediction using machine learning algorithms. *Software Engineering: An International Journal*, 2: 19-36.

Malhotra, R.. Singh, N., and Singh, Y. (2011). Genetic algorithms: Concepts, design for optimization of process controllers. *Computer and Information Science,* 4: 39-54.

Mao, Y., Sahraoui, H. A., and Lounis, H. (1998). Reusability hypothesis verification using machine learning techniques: A case study. *Proceedings of 13th IEEE International Conference on Automated Software Engineering,* 13-16 October 1998. Honolulu, HI. 13:84-93. doi:10.1109/ASE.1998.732582

Martino, M., Hernandez, G., Fiori, M., and Fernandez, A. (2013). A new framework for optimal classifier design. *Pattern Recognition, 46*, 2249-2255.

Matthews, J. W. (1975). *Epitaxial growth*. New York: Academic Press.

McCabe, T. J. (1976). A complexity measure. *Proceedings of the 2nd International Conference on Software Engineering,* 2: 308-320.

McCall, J. A., Richards, P. K., and Walters, G. F. (1977). Factors in software quality, Vol. 1(2*). National Technology Information Service. Springfield*, VA.

McDermid, J., and Bennett, K. H. (1999). Software engineering research in the UK: A critical appraisal. *IEE Proceedings – Software,* 146: (4), 179-186.

Melanie, M. (1998). *An introduction to genetic algorithms*, Cambridge, MA: The MIT Press.

Meyer, C. D. (2000). *Matrix analysis and applied linear algebra*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Michalewicz, Z. (1996). *Genetic algorithms + data structure = evolution programs* (3rd., rev. and ext. ed.). New York: Springer-Verlag.

Michura, J., Capretz, M. A., and Wang, S. (2013). Extension of object-oriented metrics Suite for software maintenance. *ISRN Software Engineering, 2013*, 1-14.

Misra, S. C., and Bhavsar, V. C. (2003). Relationship between selected software measures and latent bug-density: Guidelines for improving quality. *Computer Science and its Applications (ICCSA), Lectures Notes in Computer Science* (LNCS), 724-732.

Mizuno, O., and Hata, H. (2010). An empirical comparison of fault-prone module detection approaches: Complexity metrics and text feature metrics. *Proceedings of IEEE 34th Annual Computer Software and Applications Conference, 34*, 248-249.

Nagappan, N. and Ball, T. (2005). Static analysis tools as early indicators of pre-release defect density. *In Proceedings of ICSE 2005 (27th International Conference on Software Engineering), ACM,* 580–586.

Nelson, A. L., Barlow, G. J., and Doitsidis, L. (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems,* 57: 345-370.

Okike, E. (2010). A proposal for normalized lack of cohesion in method (LCOM) metric using field experiment. *IJCSI International Journal of Computer Science Issues,* 7: (4), 19-27.

Olague, H. M., Etzkorn, L. H., Gholston, S., and Quattlebaum, S. (2007). Empirical validation of three metrics suites to predict fault-proneness of object-oriented class development using highly iterative or agile software development processes. *IEEE Transactions Software Engineering,* 33: 402-419.

Olmo, F., and Gaudioso, E. (2008). Evaluation of recommender systems: A new approach. *Expert Systems with Applications,* 35, 790–804.

Pandey, M. H., Dixit, A., and Mehrotra, D. (2012). Genetic algorithms: Concepts, issues and a case study of grammar induction. *Preceedings of the CUBE International Information Technology Conference* (CUBE), ACM, New York, 263-271.

Park, R. E., Goethert, W. B., and Florac, W. A. (1996). *Goal-driven software measurement: A guidebook*. Handbook CMU/SEI-96-HB-002.

Petridis, V., Kazarlis, S., and Bakirtzis, A. (1998). Varying fitness functions in genetic algorithm constrained optimization: The cutting stock and unit commitment problems. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics,* 28*:*(5), 629-640.

Poornima, U. S. (2011). Unified design quality metric tool for object-oriented approach including other principles. *International Journal of Computer Applications,* 26: 1-4.

Power, D. (2011). Metrics evaluation: From precision, recall and F-measure to ROC informedness, markedness and correlations. *Journal of Machine Learning Technologies,* 2: 37-63.

Rand, W., and Wilensky, U. (2006). *Verification and validation through replication: A case study using Axelrod and Hammond's ethnocentrism model*. North American Association for Computational Social and Organization.

Ratzinger, J. Gall, H. and Pinzger, M. (2007). Quality Assessment Based on Attribute Series of Software Evolution. *In Proceeding Working Conference on Reverse Engineering (WCRE07), Vancouver, Canada.*

Riaz, M., Mendes, E., and Tempero, E. (2009). A systematic review of software maintainability prediction and metrics. In Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement (pp. 367-377). doi:10.1109/ESEM.2009.5314233

Rizvi, S. W., and Khan, R. A. (2010). Maintainability estimation model for object-oriented software in design phase (MEMOOD). *Journal of Computing*, *2*(4), 26-32.

Rosenberg, L. (1998). Applying and interpreting object-oriented metrics. *Software Technology Conference.*

Rosenberg, L., Stapko R., and Gallo A. (1999). Object oriented metrics for reliability. *IEEE International Sysmposium on Software Metrics,* 1-8.

Rothlauf, F. (2006). *Representations for genetic and evolutionary algorithms* (2nd ed.). New York: Springer-Verlag.

Sahraoui, H. A., Boukadoum, M., and Lounis, H. (2001). Building quality estimation models with fuzzy threshold values, L'objet, *Edition Hermès Sciences,* 17: 535-554.

SAS Institute Inc. (2010). *SAS/STAT user's guide*. Cary, NC. USA

Schröder, G., Thiele, M., and Lehner, W. (2011). Setting goals and choosing metrics for recommender system evaluations. *UCERSTI2 Workshop at the 5th ACM Conference on Recommender Systems,* Chicago, October 23rd.

Shatnawi, R., Li, W., Swain, J., and Newman, T. (2010). Finding software metrics threshold values using ROC curves, *Journal Software Maintenance and Evolution, Research and Practice,* 22: 1-16.

Singh, P., Chaudhary, K., and Verma, S. (2011). An investigation of the relationships between software metrics and defects. *International Journal of Computer Applications, 28*, 13-17.

Soi, I. (1985). Software complexity: An aid to software maintainability. *Microelectronics Reliability, 25*, 223-228.

Sommerville, I. (2011). *Software engineering,* (9th ed.). Addison Wesley.

Spinellis, D. (2005). Tool writing: A forgotten art? *IEEE Software*, 22: 9-11.

Subramanyam, R., and Krishnan, M. (2003). Empirical analysis of CK metrics for object-oriented design complexity: Implications for software defects. *IEEE Transactions Software Engineering,* 29: (4), 297–310.

Svitak, J. J. (2008). *Genetic algorithms for optical character recognition.* Ann Arbor, MI: ProQuest LLC.

Thirugnanam, M., and Swathi, J. N. (2010). Quality metrics tool for object oriented programming. *International Journal of Computer Theory and Engineering,* 2: 712-717.

Ujhazi, B., Ferenc, B., Poshyvanyk, D., and Gyimothy, T. (2010). New conceptual coupling and cohesion metrics for object-oriented systems. *Proceedings of the 2010 10th IEEE Working Conference on Source Code Analysis and Manipulation, Timisoara*, (12-13 September 2010).

Unger, B., and Prechelt, L. (1998). *The impact of inheritance depth on maitenance tasks – Detailed description and evaluation of two experiment replications.* Technical Report 19/1998, Facultat fur Informatik, Universitaet Karlsruhe.

Vivanco, R. (2010). *Improving predictive models of software quality using Search-Based Metric Selection and Decision Trees*. PhD Thesis, University of Manitoba.

Wang, H., Khoshgoftaar, T. M., and Seliya, N. (2011). How many software metrics should be selected for defect prediction. *Proceedings of the 24th International Florida Artificial Intelligence Research Society Conference,* 24: 69-74.

Wein, L. M., Wu, J. T., and Kirnl, D. H. (2003). Validation and analysis of a mathematical model of a replication-competent oncolytic virus for cancer treatment: Implications for virus design and delivery. *Cancer Research,* 63: (6),1317–1324.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. New York: Springer.

Wolpert, D. H., and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1: (1) 67-82.

Wu, F. (2011). Empirical validation of object-oriented metrics on NASA for fault prediction. In H. Tan and M. Zhou (Eds.), *Advances in information technology and education* (pp. 168-175). Heidelberg, Germany: Springer Berlin Heidelberg.

Xing, F., Guo, L., and Lyu, M. R. (2005). A novel method for early software quality prediction based on support vector machine. *Proceedings of IEEE International Symposium on Software Reliability Engineering* (ISSRE), 213-222.

Zeller, A. (2009). *Why programs fail: A guide to systematic debugging* (2nd ed.). Burlingotn MA: Elsevier.

Zhang, Y., Wang, K., Shaw, D., Miles, J., Parmee, I., and Kwan, A. (2006). Representation and its impact on topological search in evolutionary computation. *Joint International Conference on Computing and Decision making in Civil and Building Engineering,* June 14-16, 2006 – Montreal, Canada, 2359- 2368

Zhou, Y., and Leung, H. (2006). Empirical analysis of object-oriented design metrics for predicting high and low severity faults. *IEEE Tranactions on Software Engineering, 32*, 771-789.

Zimmer, R., Holte, R., and MacDonald, A. (1997). The impact of representation on the efficacy of artificial intelligence: The case of genetic algorithms. *AI and Society,* 1: 76-87.