# Application of Ensemble Techniques in Predicting Object-Oriented Software Maintainability

Hadeel Alsolai
1 Computer Science and Information system
Princess Nourah Bint Abdulrahman University
Riyadh, Saudi Arabia
2 Computer and Information Sciences
University of Strathclyde
Glasgow, United Kingdom
hadeel.alsolai@strath.ac.uk

Marc Roper
Computer and Information Sciences
University of Strathclyde
Glasgow, United Kingdom
marc.roper@strath.ac.uk

## ABSTRACT

While prior object-oriented software maintainability literature acknowledges the role of machine learning techniques as valuable predictors of potential change, the most suitable technique that achieves consistently high accuracy remains undetermined. With the objective of obtaining more consistent results, an ensemble technique is investigated to advance the performance of the individual models and increase their accuracy in predicting software maintainability of the object-oriented system. This paper describes the research plan for predicting object-oriented software maintainability using ensemble techniques. First, we present a brief overview of the main research background and its different components. Second, we explain the research methodology. Third, we provide expected results. Finally, we conclude summary of the current status.

## CCS CONCEPTS

D.1.5 [**Programming Techniques**]: Object Oriented Programming, D.2.2 [**Software Engineering**]: Design Tools and Techniques --- object-oriented design method, K.6.4 [**Management of Computing and Information Systems**]: System Management—Quality Assurance.

## KEYWORDS

individual model; ensemble model; software maintainability; prediction; Object-oriented system.

**ACM Reference format:**

EASE '19, April 15–17, 2019, Copenhagen, Denmark

ACM ISBN 978-1-4503-7145-2/19/04.
https://doi.org/10.1145/3319008.3319716

## 1 Introduction

Software Quality Assurance (SQA) is defined as the group of activities that guarantee software meets a certain quality level [1]. Software maintainability is one of the essential attributes in evaluating the SQA, and it begins as soon as the system has been produced. It has a vital role in enabling frequent changes to the software to meet customer requirements, adapting to environmental changes, accommodating new hardware or in developing new features. Software maintenance consumes the largest amount of cost, time and effort during the software development life-cycle (SDLC) [2]. Jones reported that maintenance was observed to consume about 75 % of total project cost, and the cost of maintaining source code is ten times than the cost of developing source code [3].

Controlling costs, time and effort are the significant components for ensuring SQA. These are performed by determining appropriate measures: as T. DeMarco [4] stated that “you cannot control what you cannot measure”. Software metrics are quantitative measures which may be employed to evaluate the quality of the software. In particular, Object-Oriented (OO) metrics are used to measure aspects of the source code of software systems (e.g. cohesion, size and inheritance depth). Consequently, several studies have employed a variety of OO metrics to evaluate the concept of software maintainability, such as Chidamber and Kemerer (C&K) metrics and Li and Henry (L&H) metrics [2, 5]. For example, the L&H metrics can be utilised as predictors of software maintenance effort as they have been shown to exhibit a strong relationship with the number of changes in the source code [2, 6-8]. Change maintenance effort is a well-known software maintainability measure that calculates the number of modifications made per class during the maintenance period [2]. A higher number of changes requires greater maintenance effort, which implies a lower level of maintainability.

Object-oriented (OO) systems are structured around objects and classes that have different characteristics (i.e. encapsulation, coupling and inheritance). These systems are written in various programming languages, such as Java, C++ and C#. Many OO systems are available in open-source projects (e.g. Github or SourceForge) and are used commonly by various organisations. With the growing use of OO systems, organisations have needed to further develop and change systems which in turn leads to an increase in their complexity [9].

Much attention has been paid to predicting software maintainability using machine learning techniques. Accurate predictions help to play an increasingly important role in software project management tasks: allocating developers; identifying resources; supporting decision making; evaluating cost across different projects and performing maintenance processes [10].

This prediction can assist in gaining insights on likely future maintenance, and can help in decreasing the total cost and overall effort of the software project [11]. However, building an accurate prediction model is a difficult task to achieve.

Several empirical studies have been performed to investigate various types of individual machine learning models, including: Neural networks [8]; Bayesian networks [6]; Linear regression [12]; Multiple additive regression trees [7]; K-means clustering [13]; Support vector regression [14]; and Multilayer perceptron [15]. However, the prediction accuracy of these individual models is disappointing and does not meet the criteria suggested by MacDonnell and Kitchenham et al. [16, 17]. These criteria will be explained in section 2.4.

In order to resolve the lack of consistent results in individual models, an ensemble machine learning model is introduced to investigate the potential for improving the accuracy prediction of individual models. The ensemble machine learning model is created from individual models in a heterogeneous (which integrates various types of individual models) or homogenous (which integrates the same types of individual model) fashion. A number of ensemble models have been investigated in relation to the problem of software defect prediction to improve their accuracy performance over individual models: voting feature intervals [18]; combined defect predictor [19]; average probability ensemble [20]; bagging and boosting [21]; stacking [22]; and adaptive selection of classifiers in bug prediction [23]. Heterogeneous ensemble models have been proposed to increase accuracy prediction over individual models, such as a software maintainability evaluation model based on combining multiple classifiers [24]. In addition, homogeneous ensembles have been used for predicting software maintainability, such as weighted voting, majority voting, and hard instance [25]. However, it is noteworthy that the application and evaluation of ensemble models to predict software maintainability is very limited.

The rest of this paper is organised as follows. Section 2 presents a brief description of the research background. Section 3 highlights the research methodology. Section 4 provides expected results. Finally, the paper summary of the current status concludes in Section 5.

## 2 Research background

The research background includes six primary sections: OO systems, metrics, maintainability, datasets, building prediction models and evaluating prediction models. Each section presents an overview of related studies and describes the basic concepts of this research.

### 2.1 Metrics

Metrics are independent variables that capture features of software systems that can be used to build predictive models of software maintainability. Prior software maintainability studies utilized a wide variety of OO metrics: Oman and Hagemeister metrics [26]; Coleman et al metrics [27]; Genero et al metrics [28]; Welker et al metrics [29]; Misra metrics [30]. These studies confirmed a relationship between OO metrics and software maintainability. However, this relationship is considered nonlinear, complicated and of low accuracy [8]. OO metrics, which primarily include the C&K metrics [5] and L&H metrics [2], have been used widely due to their strong relationship with software maintainability [6-8, 15, 25]. C&K includes six OO metrics: number of children (NOC), coupling between objects (CBO), response for a class (RFC), depth of the inheritance tree (DIT), weighted method per class (WMC) and lack of cohesion of methods (LCOM). L&H metrics [2] involve all C&K metrics except CBO and also include further metrics: abstract data type (ADT), message-passing coupling (MPC), lines of code (LOC), number of methods (NOM), number of semicolons in a class (SIZE1) and number of properties (SIZE2). In addition, Li and Henry introduced the CHANGE metric, which is a dependent variable to capture software maintainability prediction by computing the number of lines changed in the class during the maintenance process.

### 2.2 Maintainability

Maintainability is a dependent variable that may be determined by a wide variety of independent variables. The ISO/IEC 25010 standard [31] defined a software quality model as a collection of attributes that include: efficiency, usability, suitability, compatibility, security, reliability, portability and maintainability. Therefore, maintainability is one essential attribute of software quality and is recognised as one of the most challenging measurements due to the problem of predicting activity in the future [32]. Software maintainability is described as the ability of a software system to be modified in order to develop, correct, adapt to changes in the environment or meet particular requirements. This description indicates that software maintainability relies on various aspects of software modification (i.e. adaptation, correction, improvement or prevention). Furthermore, the ISO/IEC 25010 standard categorized software maintainability into five major sub-characteristics: reusability to identify the level of the assets can be used to construct other systems, modularity to identify the level of component independence and the extent to which changes to one component impact on the rest of the system, analyzability to identify the ease with which the software may be analysed to investigate (for example) the consequence of changes or diagnose problems, testability to identify the degree to which test criteria for a system can be established and tests to meet the criteria developed, and modifiability to identify the degree to which it is possible to modify the software product without degrading its quality.

Studies have acknowledged various types of software maintenance measurements: adaptive maintenance effort [33], corrective maintenance effort [10] and maintenance time [34]. In several studies [6-8, 15], change maintenance effort is frequently chosen to calculate the number of changes made in the class during the maintenance period. This measurement is based on the CHANGE metric proposed in [2] and discussed in the introduction.

### 2.3 Datasets

Datasets compose many metrics that include independent and dependent variables. The datasets are split into a testing set to

evaluate a prediction model and a training set to construct the model [35]. Alternatively, N-fold cross-validation is used to compare and evaluate between prediction models by dividing the dataset into ten folds equally. One of these folds is used to test model, and the remaining used to train model, where this process is iterated N times with different folds [36]. Moreover, the datasets include three major types: a public dataset, which is available to use (i.e. User Interface Management System (UIMS) and Quality Evaluation System (QUES) [2]; a partial dataset, which is extracted from available open-source software but unavailable to use, since the researcher does not provide it to the public (i.e. datasets extracted from 148 open-source system [12]); and a private dataset, which is extracted from a private system and unavailable to use (i.e. datasets extracted from private projects [10]). This research plan will initially use the UIMS and QUES datasets that have been widely used for predicting software maintainability, which makes our results comparable and repeatable [2]. Following this, more recent and larger datasets, will be investigated to validate and support the previous result [37].

## 2.4 Building prediction models

Many types of individual machine learning models, such as Neural networks [8], Bayesian networks [6], Linear regression [12], Multiple additive regression trees [7], K-means clustering [13], Support vector regression [14], and Multilayer perceptron [15] have been built to predict software maintainability. However, despite the large number of studies and models created, only a limited number of these have achieved a reasonable level of predictive accuracy, but fail to meet the criteria of an accurate prediction model, which is pred(.30) $\geq$0.70 [16] or pred(.25) $\geq$0.75 or/and MMRE $\leq$ 0.25 [38]. Furthermore, determining the best technique among individual models is difficult because the performance of these techniques relies on the dataset used. Therefore, it is clear that there is a great need to advance the performance of the individual models, and one way of achieving this is by building ensemble modes. The ensemble model may be heterogeneous, which merges various types of individual models (i.e. software maintainability evaluation model based on multiple classifiers combination [24]) or a homogenous, which merges the same types of individual models (i.e. weighted voting, majority voting, and hard instance [25]). This research plan will evaluate and compare the application of bagging and additive regression as an example of the homogenous ensemble model and stacking as an example of the heterogeneous ensemble model.

## 2.5 Evaluating prediction models

Evaluation of prediction models is a vital part of any machine learning problem in order to compare performance between several models and measure the accuracy of the model in predicting software maintainability. Several evaluation measurements have been proposed in the literature to assess prediction models in the software engineering domain [38]. Usually, regression problems used residuals or prediction error [39], while classification problems utilised confusion matrices [40]. Some of the most frequently used evaluation measurements have become de-facto standards to measure prediction accuracy, namely the mean magnitude of relative error (MMRE) and Pred(q), which is the proportion of instances in the dataset where the MRE is less than or equal a defined value (q) [17]. Further baseline measurements can be used to evaluate the performance of the predictors with the dependent variable, e.g. the sample mean [41] or the sample median [42].

# 3 Research Methodology

The fundamental objective of this research is to provide the software project manager with the ability to predict software maintainability accurately using ensemble techniques. To achieve this objective, several research questions are generated, which need to be addressed to face challenges in this research.

**RQ1)** What are the individual prediction models used to predict software maintainability? And what is the best performing individual prediction model?

**RQ2)** What are the ensemble prediction models (heterogeneous and homogeneous) used to predict software maintainability? And which is the best performing ensemble prediction model?

**RQ3)** How much can ensemble models increase or decrease the performance of individual models?

**RQ4)** What are the software maintainability datasets that available to use?

# 4 Expected results

An investigation of software maintainability prediction using ensemble techniques may provide several results. First, it enables the empirical exploration of the positive impact of ensemble models (heterogeneous and homogeneous), and assessment the extent to which these ensemble models provide an improvement in the accuracy prediction over individual models. Second, it enables the comparison between the proposed ensemble models with previous studies conducted on the most popular software maintainability datasets to determine whether these models achieve higher accuracy than the previous studies. Thirdly, it enables the critical validation of the proposed ensemble models by applying these models to numerous and extensive datasets of software maintainability extracted from open-source software projects or gathered from public repositories.

# 5 Summary of the current status

This paper presents a research plan to predict software maintainability of the OO system using ensemble techniques. The basic concept of our research background was provided, then the explanation of the research methodology was demonstrated. Finally, the expected results are determined. The proposed study provides the foundation to identify the main research components and detect further interesting direction studies in the software maintainability prediction field. Furthermore, it can be used as our guide through my PhD study. Our findings will appear in publications of future experiments to investigate the ability of the ensemble models to improve accuracy prediction of software maintainability over individual models.

Currently, we have completed the systematic literature review to analyse different applied machine learning models; also I have published the first set of the experiment results that evaluated the ability of bagging models to increase accuracy prediction over individual models [44]. In addition, we have extended this

experiment to include more models and use the UIMS dataset. We also have achieved some progress in the second experiment that comprises large datasets, in particular the bug prediction datasets [37]. In the future, we plan to explore other software maintainability measurements and build prediction models using their datasets.

## ACKNOWLEDGMENTS

The author would like to thank her supervisor, Dr. Marc Roper, for his feedback, advice and recommendations to achieve this work. This paper was supported by Princess Nourah bint Abdulrahman University and the University of Strathclyde.

## REFERENCES

[1] N. E. Fenton and M. Neil, "A critique of software defect prediction models," IEEE Transactions on software engineering, vol. 25, no. 5, pp. 675-689, 1999.
[2] W. Li and S. Henry, "Object-oriented metrics that predict maintainability," The Journal of Systems & Software, vol. 23, no. 2, pp. 111-122, 1993.
[3] C. Jones, "The economics of software maintenance in the twenty first century," Unpublished manuscript. http://citeseerx. ist. psu. edu/viewdoc/summary, 2006.
[4] T. DeMarco, Controlling Software Projects: Management, Measurement, and Estimates. Prentice Hall PTR, 1986.
[5] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," IEEE Transactions on software engineering, vol. 20, no. 6, pp. 476-493, 1994.
[6] C. van Koten and A. R. Gray, "An application of Bayesian network for predicting object-oriented software maintainability," Information and Software Technology, vol. 48, no. 1, pp. 59-67, 1// 2006.
[7] M. O. Elish and K. O. Elish, "Application of TreeNet in Predicting Object-Oriented Software Maintainability: A Comparative Study," in 2009 13th European Conference on Software Maintenance and Reengineering, 2009, pp. 69-78.
[8] M. M. T. Thwin and T.-S. Quah, "Application of neural networks for software quality prediction using object-oriented metrics," Journal of Systems and Software, vol. 76, no. 2, pp. 147-156, 5// 2005.
[9] M. M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry, and W. M. Turski, "Metrics and laws of software evolution-the nineties view," in Proceedings Fourth International Software Metrics Symposium, 1997, pp. 20-32.
[10] A. De Lucia, E. Pompella, and S. Stefanucci, "Assessing effort estimation models for corrective maintenance through empirical studies," Information and Software Technology, vol. 47, no. 1, pp. 3-15, 2005.
[11] M. Riaz, E. Mendes, and E. Tempero, "A systematic review of software maintainability prediction and metrics," in Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, 2009, pp. 367-377: IEEE Computer Society.
[12] Y. Zhou and B. Xu, "Predicting the maintainability of open source software using design metrics," Wuhan University Journal of Natural Sciences, journal article vol. 13, no. 1, pp. 14-20, February 01 2008.
[13] J.-C. Chen and S.-J. Huang, "An empirical analysis of the impact of software development problem factors on software maintainability," Journal of Systems and Software, vol. 82, no. 6, pp. 981-992, 2009/06/01/ 2009.
[14] C. Jin and J.-A. Liu, "Applications of support vector mathine and unsupervised learning for predicting maintainability using object-oriented metrics," in Multimedia and Information Technology (MMIT), 2010 Second International Conference on, 2010, vol. 1, pp. 24-27: IEEE.
[15] S. K. Dubey, A. Rana, and Y. Dash, "Maintainability prediction of object-oriented software system by multilayer perceptron model," SIGSOFT Softw. Eng. Notes, vol. 37, no. 5, pp. 1-4, 2012.
[16] S. G. MacDonell, "Establishing relationships between specification size and software process effort in CASE environments," Information and Software Technology, vol. 39, no. 1, pp. 35-45, 1997/01/01/ 1997.
[17] B. A. Kitchenham, L. M. Pickard, S. G. MacDonell, and M. J. Shepperd, "What accuracy statistics really measure [software estimation]," IEE Proceedings - Software, vol. 148, no. 3, pp. 81-85, 2001.
[18] A. T. Mısırlı, A. B. Bener, and B. Turhan, "An industrial case study of classifier ensembles for locating software defects," Software Quality Journal, vol. 19, no. 3, pp. 515-536, 2011.
[19] A. Panichella, R. Oliveto, and A. De Lucia, "Cross-project defect prediction models: L'union fait la force," in Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on, 2014, pp. 164-173: IEEE.
[20] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," Information and Software Technology, vol. 58, pp. 388-402, 2015.
[21] Y. Zhang, D. Lo, X. Xia, and J. Sun, "An Empirical Study of Classifier Combination for Cross-Project Defect Prediction," in 2015 IEEE 39th Annual Computer Software and Applications Conference, 2015, vol. 2, pp. 264-269.
[22] J. Petri, #263, D. Bowes, T. Hall, B. Christianson, and N. Baddoo, "Building an Ensemble for Software Defect Prediction Based on Diversity Selection," presented at the Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, Ciudad Real, Spain, 2016.
[23] D. D. Nucci, F. Palomba, R. Oliveto, and A. D. Lucia, "Dynamic Selection of Classifiers in Bug Prediction: An Adaptive Method," IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 1, no. 3, pp. 202-212, 2017.
[24] F. Ye, X. Zhu, and Y. Wang, "A new software maintainability evaluation model based on multiple classifiers combination," in 2013 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE), 2013, pp. 1588-1591.
[25] R. Malhotra and M. Khanna, "Particle swarm optimization-based ensemble learning for software change prediction," Information and Software Technology, vol. 102, pp. 65-84, 2018/10/01/ 2018.
[26] P. Oman and J. Hagemeister, "Metrics for assessing a software system's maintainability," in Software Maintenance, 1992. Proceedings., Conference on, 1992, pp. 337-344: IEEE.
[27] D. Coleman, D. Ash, B. Lowther, and P. Oman, "Using metrics to evaluate software system maintainability," Computer, vol. 27, no. 8, pp. 44-49, 1994.
[28] M. Genero, M. Piattini, E. Manso, and G. Cantone, "Building UML class diagram maintainability prediction models based on early metrics," in Software Metrics Symposium, 2003. Proceedings. Ninth International, 2003, pp. 263-275: IEEE.
[29] K. D. Welker, P. W. Oman, and G. G. Atkinson, "Development and application of an automated source code maintainability index," Journal of Software: Evolution and Process, vol. 9, no. 3, pp. 127-159, 1997.
[30] S. C. Misra, "Modeling Design/Coding Factors That Drive Maintainability of Software Systems," Software Quality Journal, vol. 13, no. 3, pp. 297-320, 2005.
[31] O. i. d. normalisation, Systems and Software Engineering: Systems and Software Quality Requirements and Evaluation (SQuaRE): System and Software Quality Models. ISO/IEC, 2011.
[32] M. Dagpinar and J. H. Jahnke, "Predicting maintainability with object-oriented metrics -an empirical comparison," in 10th Working Conference on Reverse Engineering, 2003. WCRE 2003. Proceedings., 2003, pp. 155-164.
[33] F. Fioravanti and P. Nesi, "Estimation and prediction metrics for adaptive maintenance effort of object-oriented systems," IEEE Transactions on Software Engineering, vol. 27, no. 12, pp. 1062-1084, 2001.
[34] R. K. Bandi, V. K. Vaishnavi, and D. E. Turk, "Predicting maintenance performance using object-oriented design complexity metrics," IEEE Transactions on Software Engineering, vol. 29, no. 1, pp. 77-87, 2003.
[35] C. Sammut and G. I. Webb, Encyclopedia of machine learning. Springer Science & Business Media, 2011.
[36] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," presented at the Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2, Montreal, Quebec, Canada, 1995.
[37] M. D. Ambros, M. Lanza, and R. Robbes, "An extensive comparison of bug prediction approaches," in 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), 2010, pp. 31-41.
[38] S. D. Conte, H. E. Dunsmore, and V. Y. Shen, Software engineering metrics and models. Benjamin-Cummings Publishing Co., Inc., 1986, p. 396.
[39] F. Mosteller and J. W. Tukey, "Data analysis and regression: a second course in statistics," Addison-Wesley Series in Behavioral Science: Quantitative Methods, 1977.
[40] T. Fawcett, "An introduction to ROC analysis," Pattern recognition letters, vol. 27, no. 8, pp. 861-874, 2006.
[41] M. Jorgensen, "Experience with the accuracy of software maintenance task effort prediction models," IEEE Transactions on software engineering, vol. 21, no. 8, pp. 674-681, 1995.
[42] E. Mendes and B. Kitchenham, "Further comparison of cross-company and within-company effort estimation models for web applications," in Software Metrics, 2004. Proceedings. 10th International Symposium on, 2004, pp. 348-357: IEEE.
[43] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," Journal of artificial intelligence research, vol. 11, pp. 169-198, 1999.
[44] H. Alsolai, "Predicting Software Maintainability in Object-Oriented Systems Using Ensemble Techniques," in 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2018, pp. 716-721: IEEE.