Comparative Analysis of Object-Oriented Software Maintainability Prediction Models

Narimane Zighed, Nora Bounour, Abdelhak-Djamel Seriai

FOUNDATIONS OF COMPUTING AND DECISION SCIENCES
Vol. 43 (2018) No. 4
ISSN 0867-6356
DOI: 10.1515/fcds-2018-0018 e-ISSN 2300-3405

Comparative Analysis of Object-Oriented Software Maintainability Prediction Models

Narimane Zighed[*], Nora Bounour [*], Abdelhak-Djamel Seriai +

Abstract. Software maintainability is one of the most important aspects when evaluating the quality of a software product. It is defined as the ease with which the existing software can be modified. In the literature, several researchers have proposed a large number of models to measure and predict maintainability throughout different phases of the Software Development Life Cycle. However, only a few attempts have been made for conducting a comparative study of the existent proposed prediction models. In this paper, we present a detailed classification and conduct a comparative analysis of Object-Oriented software maintainability prediction models. Furthermore, we considered the aforementioned proposed models from three perspectives, which are architecture, design and code levels. To the best of our knowledge, such an analysis that comprises the three levels has not been conducted in previous research. Moreover, this study hints at certain fundamental basics concerning the way of how measure the maintainability knowing that at each level the maintainability will be measured differently. In addition, we will focus on the strengths and weaknesses of these models. Consequently, the comparative study yields that several statistical and machine learning techniques have been employed for software maintainability prediction at code level during the last decade, and each technique possesses its specific characteristic to develop an accurate prediction model. At the design level, the majority of the prediction models measured maintainability according to the characteristics of the quality models. Whereas at the architectural level, the techniques adopted are still limited and only a few of studies have been conducted in this regard.

Keywords: Metrics; Maintainability Prediction; Object-Oriented Software; Quality Model, Prediction Model.

1. Introduction

Software maintenance is one of the essential phases of Software Development Life Cycle process “SDLC”. Defects introduced at this stage are the most dangerous compared to those

[*] Badji Mokhtar University, Annaba, Algeria, narimanezighed@gmail.com
[*] Badji Mokhtar University, Annaba, Algeria, nora_bounour@yahoo.com
+ Montpellier University, Montpellier, France, abdelhak.seriai@lirmm.fr

which could be introduced at the other phases of software development cycle [26]. Maintenance activities start from the moment when a system comes into operation and continues for the remainder of the product's life. Thus, according to the past studies, they have found that for some products this can last twenty years on average, unlike the development phase which can last from one to two years [19]. In addition, the time spent and effort required to correct defects in this phase consumes about 40 to 70% of the cost of the entire life cycle [21].

Therefore, McCall model which is one of the important and oldest software quality models has cited maintainability as one of a total of eleven factors, these factors are broken down by the 3 perspectives: product revision (maintainability, testability and flexibility), product transition (portability, reusability, and interoperability), product operations (correctness, reliability, efficiency, integrity, usability) [1]. Where the quality factor maintainability would have criteria of simplicity, conciseness, and modularity as sub-characteristics [33].

Several definitions for "software maintainability" have been considered. According to Standard Glossary of Software Engineering IEEE it is defined as “the ease with which a software system or component can be modified” [15]. In some studies, it has been defined as “number of lines of code changed” [22], [24], [30], [31], [35], [36], [12], [37]. In other works, maintainability has been defined as the time required to make changes, and time to understand, develop and implement a modification [29].

A maintainability prediction model is used to estimate maintenance effort of systems using some information about it and a preselected technique for developing this model. When we talk about making prediction at code level often, we measure the maintainability by number of lines of code changed. Although at design level maintainability prediction is based on quality models’ factors like understandability, modifiability, extendibility and flexibility...Etc. But for the architecture level, the maintainability is measured by estimating the effort of change.

A good predictive model of software maintainability allows organizations to efficiently manage their maintenance resources and also guide decision-making related to software maintenance. Which can help further to reduce maintenance effort and hence they can minimize the overall effort and cost of the software project [23].

In the literature, many researchers have experimented different models to predict software maintainability, whether at the code level or at the more abstract level as design and architecture levels. Most of these models were proposed in the code level while a few models were proposed at design and architecture levels (see Figure.1).

In this paper, we have considered the most important techniques proposed to predict Object-Oriented software maintainability. We focus on a classification of these models and we highlight the main difference between them. Then a comparative analysis is conducted to detail all the result obtained. Unlike [11], [25] which discuss the use of prediction techniques and compared the proposed models. In this paper we focus on the way of how the maintainability was predicted at different levels. However, no comparative analysis has been published previously which including code, design and architecture levels.

The remainder of this paper is organized as follows: section 2 gives a categorization of models used for predicting Object-Oriented software maintainability. The comparative analysis of the different models and discussion are detailed in Section 3. Finally, Section 4 concludes the work and states the possible future work.

2. Maintainability prediction models

Various studies have been conducted and proposed in the literature for predicting software maintainability. We categorize these works according to the abstraction level exploited by the model. We distinguish three levels: architecture, design and code level. In Figure.1, we represent the number of papers submitted in each stage of SDLC within a period of time. The papers that are classified in figure 1, they are published in journals, conferences and others which include book chapters, technical reports, white papers, symposium. The important journals were selected depending to the impact factor while conferences were identified according to the international repute that address issues in the field of software maintenance.

We restricted our search to the period from 1993 to 2017 and the search strategy were developed according to the following steps:
1) Derive major search strings from the research questions
2) Use Boolean OR to construct search strings from the search terms with similar meanings. Use Boolean AND to concatenate the search terms and restrict the research.
3) The resulting were the keywords given bellow: (maintainability prediction OR effort maintainability prediction OR software maintainability estimation) AND (machine learning techniques OR regression techniques OR methods) AND (Object Oriented metrics) AND (code and design and architecture)

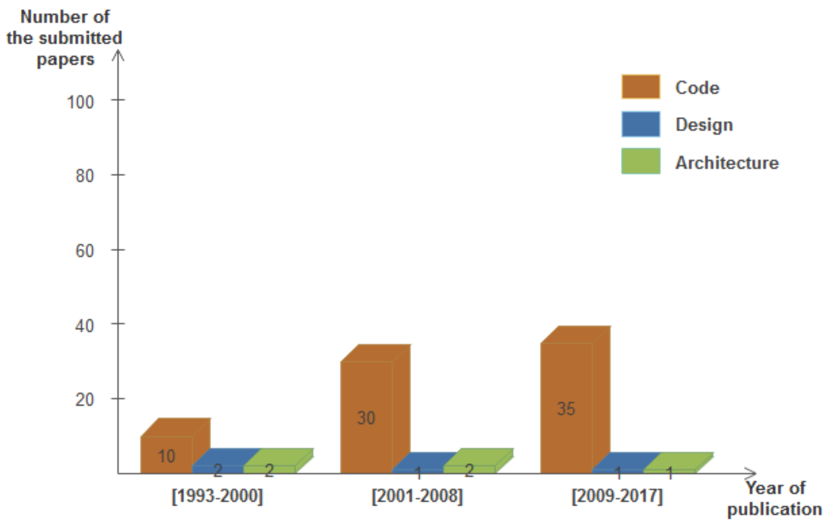

Figure 1. Distribution of papers submitted for maintainability prediction from different perspective

The most obvious from the graph is that few works have been proposed in early stages: architecture and design, also we noticed that most studies have treated the prediction of maintainability at code level. In the next sections, we will represent the most important models in each level.

2.1. Prediction models at architectural level

Software architecture is one such a key artifact which can be used in early maintenance prediction. According to Bass [7] “the software architecture of a program or computing system is the structure of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them”. Among the important recent works that dealt with maintainability prediction using software architecture the studies that were led by Anwar and al in [5], [6] and Bengeston and al in [9], the proposed approach has several inputs: architecture specification, engineer expertise, maintenance data history. The authors developed a probability-based approach. They estimated " maintenance effort " at the software architecture level using the scenario profiles.

The first step was to identify the different change scenarios that can be encountered during the maintenance phase. Once the growth scenario profile is developed, the next step was to classify scenarios based upon their complexity levels (Simple, Average, Complex), then each scenario is assigned a certain probabilistic weight. The weight measure of the scenario is defined as relative probability during a specific time interval; the weights are assigned based upon historical maintenance data. If no historical maintenance data is available then domain expert or software architect estimates the scenario weight. Once the change impact analysis has been performed, it is possible to predict the maintenance effort according to a proposed mathematic formula. We clearly perceive that the approach proposed in [5, 6] is inspired by the one that was presented in [9].

2.2. Prediction models at design level

The software design of an application has a considerable effect on quality factors such as maintainability. Using software design to quantify some quality factors will help organizations to plan resources accordingly. In terms of prediction, some techniques are available for the software engineer’s community to predict system maintainability at the design level.

In a series of studies [14-15], Marcela Genero investigates the possibility of the use structural complexity and size metrics as good predictors of maintainability by constructing maintainability prediction models based on metrics of UML class diagrams. In 2003 she developed four models which relate the size and structural complexity metrics of UML class diagrams with maintainability measures like understandability time, modifiability time, modifiability completeness, modifiability correctness. In these models, she defined modifiability and understandability as maintainability’s sub-characteristics. The metrics from this model can be used for software maintainability prediction resolutions. In order to test her hypothesis, she used Multivariate Linear technique which is commonly used, it allows to figure out the relationship between dependent and independent variables. The results achieved have an accurate prediction.

In another study Kiewkanya and al [20] proposed a methodology for constructing maintainability model of Object-Oriented System by using two concepts understandability and modifiability. They carried out a controlled experiment with undergraduate students, with the purpose of building models for the maintenance level (easy, medium and difficult). These models are developed by using three different techniques and were based on measurement values calculated from UML class diagrams and sequence diagrams. The first maintainability model used the metrics-based discriminant technique which analyzed the pattern of correlation between maintainability levels and structural complexity design metrics. The second model was built using the weighted-score-level technique by taking a weighted sum of understandability and modifiability scores. The third model was proposed using the weighted-predicted-level technique that uses a weighted sum of predicted understandability and modifiability levels, obtained by applying understandability and modifiability models. However, a comprehensive set of examination questions were required to capture understandability and modifiability score for each software design model. Further, those scores might suffer from subjectivity due to different levels of understanding and of subjects in the experiments.

In 2010, in a research conducted by Rizvi and Khan [29], the authors carried out an empirical study which investigated the relation between the software maintainability of the class diagram and his Understandability and Modifiability. They found that Understandability and modifiability are strongly correlated with maintainability and can therefore be used as good predictors of maintainabilityof software. Then a prediction model was developed to predict the maintainability of a class diagram in terms of the Understandability and the modifiability of these classes by using a multiple (multivariate) regression method. The study involved values of understandability, modifiability, maintainability and eleven measures of size and structural complexity previously collected by controlled experiments on 28 class diagrams. They applied a multivariate linear regression to construct models to estimate the comprehensibility and modifiability of class diagrams using the eleven measurements and to estimate the maintainability of the class diagram using Understandability and modifiability as attributes.

Further, in the year 2013 Alshayeb [2] performed an empirical study to evaluate the relationship between four stability metrics and indices of maintenance effort, they found that classes with higher values of Class Stability Metrics (CSM) are associated with lower values of perfective maintenance effort measured by hours while none of the stability metrics is correlated with maintainability measured by the number of changed lines.

From the point of view of Kumar and Dhanda [21], they saw that maintainability of an Object-Oriented software design is affected by several factors, in which extendibility and Flexibility that are taken as a major and key factor. The data used during the study is the same as [29]. They developed three models to compute flexibility, extendibility and maintainability of the class diagrams, the prediction model measures the maintainability of Object-Oriented design in terms of their flexibility and extendibility. All the models have been developed using the process of multiple linear regressions.

Whereas Soni and al [34], had developed three models to compute maintainability prediction for class diagram using Extendibility and Reusability of the class diagrams. The maintainability is measured in terms of their extendibility and reusability. All the three models have been developed using the method of multiple linear regressions.

Lu and al in [32] proposed a methodology for assessing software maintainability at design level and more precisely the maintainability of class diagram. Considering a set of metrics for class diagram measurement, the authors have made a comprehensive study on maintainability assessment from the defect-correction perspectives. Using a defect repository and corrective maintenance history of Apache Tomcat (maintained from 2006 to 2014), they have concluded that software maintainability can be accurately estimated in terms of time span, number of modified lines of code and impact span for a defect correction using size, coupling and inheritance metrics.

2.3. Prediction models at code level

Several techniques and approaches have been proposed in the literature to predict the maintainability at code level, these methods vary from simple statistical models such as regression analysis to complex automatic learning algorithms, such as "neural networks, genetic algorithms, etc.". Various methods proposed in the literature for the prediction of maintainability are summarized in Table 1.

Table 1. List of shortlisted studies

ID	Authors and year	Techniques used
[22]	[Li and al, 93]	Multiple Linear Regression
[13]	[Fioravanti and al, 01]	Multilinear regression analysis
[10]	[Dagpinar and al, 03]	Multiple Linear Regression
[27]	[Misra, 05]	Multivariate Regression Analysis
[30]	[Van koten and al, 06]	Bayesian Network
[35]	[Zhou and al, 07]	Multivariate Adaptive Regression Splines
[3]	[Aggarwal and al, 08]	Artificial Neural Network (Multilayer Feed Forward)
[12]	[Elish and al, 09]	TreeNet « or also known as Multiple Additive Regression Trees »
[31]	[Li-jin and al, 09]	Projection Pursuit Regression
[18]	[Jin, and Liu, 10]	Support Vector Machine (SVM)
[23]	[Malhotra and al, 12]	Genetic Algorithms Probabilistic Neural Network Group Method of Data Handling
[8]	[Baqais and al, 13]	Neural Network Genetic Algorithms
[24]	[Malhotra and al, 14]	Group Method of Data Handling
[19]	[Jindal and al, 15]	Neural Network (Radial Basis Function NN)
[17]	[Jain and al, 16]	Evolutionary Algorithm

2.3.1 Statistical models

In order to monitor maintenance and reengineering, quantitative metrics to assess and predict system characteristics should be used. There are many predictive models of maintainability of software published in the literature that suggest a way to establish the relationship between metrics and software maintenance.

Several studies have used linear regression techniques to construct models for predicting the maintainability of software. For example, Li and Henry [22] they used these techniques to predict maintenance effort, with a combination of metrics collected from the software source code can be used as input to predict maintenance effort using multivariate regression analysis model (MLR).

In [13] a study was carried out by Fioravanti and Nesi with the aim of building and evaluating a prediction model and the measurements for estimating the adaptive maintenance effort of Object-Oriented systems.

According to Misra [27], the MI "maintainability index" have been used as an indicator of the maintainability of Object-Oriented systems. It has studied the effect of the twenty different metrics at the design / code level on maintainability using statistical techniques of linear regression to predict the effort of maintainability.

Zhou and Leung in [35] also used a multivariate adaptive regression splines technique, which allows to model the relationship between a desired value (also called a target variable) and several predictive variables to construct a prediction model of the maintenance effort using the metric data collected from two different Object-Oriented systems presented in [22]. These metrics represent cohesion, coupling, inheritance and size.

In a study conducted by Li-jin and al [31], a predictive model of maintainability was developed using the static technique "Projection Pursuit Regression" and this by means of a set of Object-Oriented metrics which is constituted as maintainability predictors. The values of these metrics were collected from two software products UIMS (User Interface Management System) and QUES (Quality Evaluation System).

Jun and Liu [18] validated their prediction model using the datasets collected from the software systems developed by graduate students. Their results show that when Support Vector Machine (SVM) is combined with clustering for the purpose of maintenance effort predictions, correlation between Chidamber and Kemerer “C&K” metric suite and maintainability was found to be as high as 0.769 which is statistically quite significant.

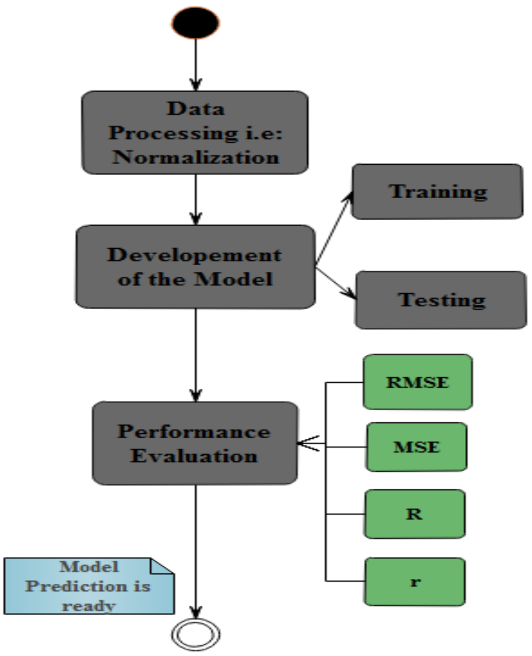
Data Processing i.e: Normalization
Training
Developement of the Model
Testing
RMSE
Performance Evaluation
MSE
R
r
Model Prediction is ready

Figure 2. The general process of using ML in maintainability prediction

2.3.2. Machine learning techniques-based models

Machine learning (ML) is used for the study of information which helps in the mechanization of the development of a prediction model. A great variety of experimental work has been carried out which recommends the use of automatic learning techniques [28].

However, the link between Object-Oriented metrics and maintainability is often complex and non-linear, limiting the accuracy of classical approaches. Different prediction models- based ML techniques and Object-Oriented metrics have been proposed for the prediction of software maintenance effort.

I- Using a single technique

Van and Gray in [30] have constructed a model for predicting the software maintainability of OO systems using Bayesian networks. The authors used the Object-Oriented metric data set presented by Li and Henry [22]. In this study, maintainability is measured as the number of code's changes during a maintenance period. To build the Bayesian network the Bayda tool was used. Bayda allows users to build a special type of Bayesian networks called Bayesian naive classifier. In which a single node representing a classification variable (change) is connected to all the other nodes that represent the predictor variables (ten of OO metrics). Then the precision of the model predictions is evaluated and compared with models-based regression. The results suggest that the Bayesian network model can predict maintainability more accurately than regression-based models for one system (UIMS) and almost as accurately as the best regression-based model for the other system (QUES).

Elish [12] used TreeNet to construct a software maintainability prediction model using the same data collected by Li and Henry [22]. They explored their experience on two popular data sets in the field of maintainability known as UIMS and QUES. The authors have proved that it also provides competitive results compared to other models.

Aggarwal and al [3] used ANN as a technique to construct a prediction model to estimate the maintenance effort of OO software at the class level by estimating the number of rows modified per class, On the other hand, the authors of [3] aim to explore empirically the relationship between OO metrics and maintainability estimation. The values of the metrics studied were collected from a total of 110 classes from two software systems: UIMS and QUES [22]. A common problem that assumes when using metrics is called correlation. This problem arises when the dependent variables are strongly correlated with each other and this is the case with the OO metrics to remedy this problem Principal Component Analysis is a statistical technique that has been used in this study to transform the data in uncorrelated variables and reduce the correlation between the independent variables. The constructed ANN model belongs to the whole of the multilayer perceptron [4] and the results of the latter's evaluation showed that the relative absolute mean error (MARE) was 0.265 of the model. The results show that the ANN is capable of providing an adequate model for predicting maintenance effort and that OO metrics can be useful in guiding prediction.

Jindal and al [19] also used neural networks to develop a prediction model, but by using another type of neural network "radial basis function network. Malhotra and Chug [23] constructed a model using automatic learning algorithms to predict the maintainability of Object-Oriented software such as Genetic Algorithms (GA) and Group Method of Data Handling (GMDH). They evaluated the execution of this model using UIMS and QUES systems, and found that the GMDH network model is one of the better methods of demonstration to anticipate the maintainability of the product.

In 2016 a recent study was conducted by Jain, Tarwani and Chug [17] to propose the use of genetic algorithms for predicting the software maintainability, and to compare its performance with various automatic learning techniques. They extracted the OO metrics from four open source projects jTDS , jWebUnit, jXLS and SoundHelix, using the tool (Chidamber and Kemerer Java Metrics). The Weka tool was also used to construct the prediction model. In their study, maintainability was measured by counting the number of changes at the code level, which was calculated by comparing each class of the two versions using the Beyond Compare tool. To evaluate the accuracy of predictions found they used Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) as precision measures of prediction which were proposed by Kitchenham. According to the results, the genetic algorithm gives more accurate predictions compared to other models of automatic learning- based prediction.

II- Using hybrid techniques

From 2012 onwards, hybrid methods proved that their results are even better in predicting maintainability.

Baqais and al [8] also used neural networks to propose a model of maintenance effort prediction. This model is classified as a hybrid model because they used the ANN to construct it while the genetic algorithm was used to speed up the ANN process by adjusting the parameters of its design in order to achieve an optimized topology. In this study, four groups of metrics were evaluated to conclude their direct influence on maintainability, which was measured by the use of the maintainability index, the LOC (Line of Codes), NOA (Number of Attributes), NLM (Number of Local Methods) and WMC (Weighted Methods per Class) that represent: size, cohesion, coupling and inheritance, they were collected from the Android project and they were analyzed empirically to understand their relationship to maintainability. The Metamata tool was used to calculate the metrics while another tool called JHawk was also used to calculate the maintainability index. After this, the data was provided to another tool called DTREG to build the predictive model based on AI techniques. The results show that the size of the code and the coupling metrics are a good indicator to provide an accurate prediction of the maintainability measure.

Malhotra and Chug [24] deployed static measurements and found that the performance of the Group Data Management Method (GMDH) was concise in terms of prediction accuracy and can be viewed as a prediction of maintainability.

3. Discussion and comparison

In this part, we compare the different works presented previously. The comparative study is elaborated by answering the following research questions:

Q1: How the software maintainability was measured using different artifacts of the aforementioned abstraction levels?

Q2: Have the OO metrics been used at all the considered levels? What is the set of OO metrics used for the software maintainability prediction? And which of these were the most influential on maintainability prediction?

Q3: What kinds of datasets used for empirical validations?

Q4: How can we judge the performance of prediction models?

Q5: What are the most accurate techniques to use for predicting maintainability?

3.1. Measurement and definition of software maintainability during the considered SDLC phases.

Table 2. Different measurements of maintainability

Studies		**Measurement of software Maintainability**
Architectural level	[5], [6], [9]	The maintainability is measured by estimating change effort (hours effort) required for dealing with changes in maintenance phase by using mathematical model.
Design level	[15]	Understandability: The ease with which a class diagram can be understood.
		Analyzability: The capability of a class diagram to be diagnosed for deficiencies or to identify parts to be modified. Modifiability: The capability of a class diagram to enable a specified modification to be implemented.
	[20], [29]	Two sub-characteristics of maintainability are considered: Understandability: is the degree to which the software design model can provide its clear meaning to evaluator. Modifiability: is the degree to which the software design model can be changed.
	[2]	The authors correlates class stability with maintainability effort measured by the number of hours spent on maintenance activities and by the line of code changes
	[34]	The authors estimate the maintainability of class diagrams in respect of their Extendibility.
Code level	[27], [8]	Maintainability was measured using the widely accepted Maintainability Index (MI).
	[30], [35], [31], [12], [3], [19], [17], [24]	The number of changes made to the code during a maintenance period.

The maintainability of a software system can be measured in different ways. At code level most of studies have measured maintainability by estimating the number of changes made to the code during a maintenance period, and in few of studies, the maintainability has been quantified by the Maintainability Index (MI) [27-8]. On other hand at design level the maintainability can be estimated by measuring some of the sub-characteristics of maintainability such as understandability, analyzability, modifiability and testability. In some studies, they have measured it by measuring two or even three sub-characteristics but the understandability still an important sub-characteristic of maintainability, since professionals spend at least half of their time analyzing software to understand it [15-20,29]. But at architectural level they estimated the effort of change for measuring the maintainability. Table 2 shows different measurements of software maintainability.

3.2. Metrics suite used in software maintainability prediction.

The link between OO design/code metrics and software maintainability has been proposed by many researchers. These studies [3], [10], [22], [15], [27], [30] have found that a strong link between these metrics and software maintainability exist. Due to the help of many empirical studies, it has been established that the quality of software design, as well as code, is very important to improve the maintainability of software. Numerous measures havebeen proposed in the literature to capture the structural quality of code and design of Object-Oriented programs, when at architecture level the OO metrics were not used for maintainability prediction. Such measures are aimed at providing means of evaluating the quality of the software, among which the most well-known metrics are those of Chidamber and Kemerer "CK", Lee and Henry [22].

Table 3. List of OO metrics

Studies		**Software metrics used**
Code level	[22], [27], [30], [3], [12], [23], [24]	CK metrics, Li and Henry metrics, and Size Metrics
	[35]	CK metrics, and Li and Henry
Design level	[15], [29]	Set Metrics for UML Class Diagrams

As can be concluded, the performance of the maintainability prediction models depends on choosing the right set of Object-Oriented design/code metrics.

In [8] the results suggested that coupling metrics are a good indicator to provide an accurate prediction of the maintainability measure. In [10] the results indicate that size and import direct coupling metrics serve as the important predictors measuring maintainability of classes while inheritance, cohesion, and indirect/export coupling measures are not. Zhou [33] found that the average control flow complexity per method (OSAVG) appears to be the most important maintainability factor.

While studies [15], [30], [12], [3] and others did not provide any explicit decision for successful predictors of maintainability.

Overall it was observed that metrics related to size, complexity and coupling were to date the most successful maintainability predictors employed.

3.3. Kinds of dataset used to elaborate software maintainability models

Many researchers and studies have conducted empirical studies in part to prove and show that the values of OO design metrics had a considerable effect on maintainability. All of these studies are based on small projects, proprietary software databases, open source software, etc.

Building and evaluating software maintainability prediction techniques rely mainly on datasets, some research studies have taken real-life data whereas some studies have used the dataset proposed by Li and Henry [22] from two commercial software packages namely user interface management system (UIMS) and quality evaluation system (QUES). Maintenance efforts are generally calculated by counting the number of lines added, deleted or modified during operations. The source code of old and new versions were collected and analyzed against modifications made in every class. Values of OO software design metrics suite were calculated and combined with corresponding changes made into that class so as to generate datasets which were further divided into 3:1:1 for training, testing and validation, respectively, during model implementations [24].

Table 4. List of datasets used and its characteristics

Datasets	Referred in studies	Code characteristic
UIMS and QUES	[30], [35], [3], [12], [31], [22], [23], [24].	- UIMS (User Interface Managemen System, 39 classes) - QUES (Quality Evaluation System, 71 classes) Both systems were implemented in ADA language.
Open Source Datasets: Androidprojet, jTDS, jWe- BUnit, JXLS, Sound Helix, Apache Tomcat server.	[8], [17], [32].	- jTDS (64 classes) [17] - jWebUnit (22 classes) [17] - jXLS (78 classes) [17] - SoundHelix (67 classes) [17] - Android project (78 classes) [8] - Apache Tomcat [32] All systems are written in Java
Propriety software	[10], [27].	- 50 projects written in C++ total of 15637 classes [27] -Two systems written in Java Fujaba-UML (FUML) and DynamicOb- jec Browser (dobs) [10]
Design diagrams	[21], [29], [15]	Twenty-eight UML class diagrams

3.4. Measures to judge the performance of prediction models.

When we develop a maintainability prediction model, the results obtained must be tested and compared with the actual values of maintainability. There is a number of statistical measures that have been proposed by several authors to measure the accuracy and precision of the prediction and to ensure that a precise prediction is not due to a simple coincidence rather, it exists in fact by proposing formulas with their corresponding interpretations, among the measures most used by the authors of papers presented in above sections: magnitude of relative error (MRE)[30-12-18-35-3-24-23], mean magnitude of relative error (MMRE)[30-12-35-23], residual error (RE), absolute residual error (ARE)[35], mean of ARE (MARE)[18-3],, Standard deviation of ARE (StdevARE), Pred (q)[35-23], Mean Absolut Error [19], R-square[23], root mean square error (RMSE)[19] and Normalized mean square error [8].

3.5. The most accurate techniques to use for predicting maintainability

The aim of prediction models is to estimate Object-Oriented software maintainability but getting an accurate result remains the primary goal of each study that was carried out in literature. At code level the use of OO metric is inevitable in order to quantify the characteristics of the code and also it provides ways to evaluate the maintainability of software. The link between these metrics and maintainability is often complex and non-linear which limit the accuracy of statistical approaches. The use of ML techniques to develop prediction model gives more accurate results.

4. Threats to validity

In this section, we discuss the main threats to validity which need to be considered when interpreting the results from our study and the way we attempted to alleviate them. The identified threats are given as under:

The first threat to validity in our study is biased in our selection of the works to be included and the research databases. Therefore, in order to ensure that we have chosen the most important studies, we have adopted a selection process and it was unbiased, in which different databases were identified like: Google Scholar, Science Direct, Springer, ACM Digital Library, IEEE Xplore.

The second threat to validity is related to the search strings used for selecting relevant studies. The search strings were derived from the research questions which we have formulated for conducting and guiding our analysis.

5. Conclusion

In this paper, we have presented the most important techniques applied in the literature that were used for the purpose of software maintainability estimation and constructing prediction models. In contrast to other studies that took mostly one level and they analyzed the proposed models, in our study we considered the code, design, and architecture level of SDLC. We performed a comparative analysis by answering the research questions which are presented in section 3. We conclude that only few works have been done at the design and architectural level. For future work, we could explore the use of machine learning techniques at the design and architectural level.

References

[1] AL-Badareen A.B., Selamat M.H., Jabar M.A., Din J., Turaev S.: Software Quality Models: A Comparative Study. In the International Conference on Software Engineering and Computer Systems, pp.46-55, Springer, Malaysia, (2011).
[2] Alshayeb M.: on the relationship of class stability and maintainability, IET Softw.7, pp.339-347, (2013).
[3] Aggarwal K.K., Singh Y., Kaur A., Malhotra R.: Application of Artificial Neural Network for Predicting Maintainability using Object-Oriented Metrics. In World Academy of Science, Engineering and Technology, pp.285-289, (2008).
[4] Aggarwal K.K., Singh Y., Kaur A., Sangwan O.P.: A Neural Net Based Approach To Test Oracle. In ACM SIGSOFT Software Engineering Notes, pp.1-6, (2004).
[5] Anwar S., Ramzan M., Rauf A., Shahid A.A.: Software Maintenance Prediction Using Weighted Scenarios: An Architecture Perspective. In International Conference on Information Science and Applications (ICISA), pp.1-9, IEEE, Korea (South), (2010).
[6] Anwar S.: Software Maintenance Prediction: An Architecture Perspective. PHD thesis, AST National University of Computer & Emerging Sciences, Islamabad, Pakistan, (2010). [7] Bass L., Clements P., Kazman R.: Software Architecture in Practise. Second edition. Addison Wesley, (2003).
[8] Baqais A. A. B., Alshayeb M., Baig Z.A.: Hybrid Intelligent Model for Software Maintenance Prediction. In World Congress on Engineering, pp.358-362, Springer, Lon- don, U.K, (2013).
[9] Bengtsson P., Bosch J.: Architecture Level Prediction of Software Maintenance. In The 3rd European Conference on Software Maintenance and Reengineering (CSMR), pp.139- 147, IEEE, Netherlands, (1999).
[10] Dagpinar M., Jahnke J.H.: Predicting Maintainability with Object-Oriented Metrics - An Empirical Comparison. In The 10th Working Conference on Reverse Engineering (WCRE), pp.155-164, IEEE, USA, (2003).
[11] Elmidaoui S., Cheikhi L., Idri A.: Accuracy Comparison of Empirical Studies on Software Product Maintainability Prediction. World Conference on Information Systems and Technologies, pp.26-35, (2018).
[12] Elish M., Elish K.: Application of TreeNet in Predicting Object-Oriented Software Maintainability: A Comparative Study. In The 13th European Conference on Software Mainte- nance and Reengineering (CSMR), pp.69-78, IEEE, Germany, (2009).
[13] Fioraventi F. , Nesi P.: Estimation and Prediction Metrics for Adaptive Maintenance Effort of Object-Oriented Systems. In IEEE transactions on software engineering, pp.1062– 1084, (2001).
[14] Genero M., Piattini M., Calero C.: Early measures of UML class diagrams, Herms Science Publication, vol. 6, pp.489-515, January (2000).
[15] Genero M., Olivas J., Piattini M., Romero F.: Using metrics to predict OO information systems maintainability, Lecture Notes in Computer Science, pp.388-401, (2001).

[16] International Software Testing Qualifications Board: IEEE Standard glossary of terms used in Software Engineering, (2011).
[17] Jain A., Tarwani S., Chug A.: An Empirical Investigation of Evolutionary Algorithm for Software Maintainability Prediction. In Students' Conference on Electrical, Electronics and Computer Science (SCEECS), pp.1-6, IEEE, India, (2016).
[18] Jin C., Liu .J.A.: Applications of Support Vector Machine and Unsupervised Learning for Predicting Maintainability using Object-Oriented Metrics. In The Second International Conference on MultiMedia and Information Technology (MMIT), pp.24 - 27, IEEE, China, (2010).
[19] Jindal R., Malhotra R., Jain A.: Predicting Software Maintenance Effort Using Neural Networks. In The 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), IEEE, India, (2015).
[20] Kiewkanya M., Jindasawat N., MuenchaisriK P.: A Methodology for Constructing Maintainability Model of Object-Oriented Design, the Fourth International Conference on Quality Software (QSIC'04), IEEE, pp.206–213, (2004).
[21] Kumar R., Dhanda N.: Maintainability Measurement Model for Object-Oriented Design. In International Journal of Advanced Research in Computer and Communication Engineering, pp.68-71, (2015).
[22] Li W., Henry S.: Object-Oriented Metrics that Predict Maintainability. Journal of Systems and Software, pp.111–122, (1993).
[23] Malhotra R., Chug A.: Software Maintainability Prediction using Machine Learning Algorithms. In Software Engineering: An International Journal (SEIJ), pp.19-36, (2012).
[24] Malhotra R., Chug A.: Application of Group Method of Data Handling model for software maintainability prediction using Object-Oriented systems. International Journal of System Assurance Engineering and Management, pp.165-173, (2014).
[25] Malhotra R., Chug A.: Software Maintainability: Systematic Literature Review and Current Trends. International Journal of Software Engineering and Knowledge Engineering, Vol. 26, pp.1221–1253, (2016).
[26] Mens T., Serebrenik A., Cleve A.: Evolving Software Systems, Springer, (2014).
[27] Misra S.C.: Modeling Design/Coding Factors That Drive Maintainability of Software Systems. In Software Quality Journal, pp.297-320, (2005).
[28] Nanda S., Saxena S.G., Bala A.G.: Evaluation of Feature Selection Techniques for Software Maintenance Prediction. Thapar University, India, (2017).
[29] Rizvi. S.V, Khan .R.A.: Maintainability Estimation Model for Object-Oriented Software in Design Phase (MEMOOD). In Journal of Computing, pp.26-31, (2010).
[30] Van Koten C., Gray A.R.: An application of Bayesian network for predicting Object-Oriented software maintainability. Information and Software Technology Journal, pp.59- 67, (2006).
[31] Li-jin W., Xin-xin H., Zheng-yuan N., Wen-hua K.: Predicting Object-Oriented Software Maintainability using Projection Pursuit Regression. In The 1st International Conference on Information Science and Engineering, pp.3827-3830, IEEE, China, (2009). [32] Lu Y., Mao X., Li Z.: Assessing Software Maintainability Based on Class Diagram Design: A Preliminary Case Study, (2016).
[33] Saini R., Dubey S.K., Rana A.: Analytical study of Maintainability models for Quality evaluation. In The Indian Journal of Computer Science and Engineering (IJCSE), pp.449- 454, (2011).
[34] Soni N., Khaliq M.: Maintainability Estimation of Object-Oriented Software: Design Phase Perspective", International Journal of Advanced Research in Computer and

Communication Engineering, March (2015).
[35] Zhou Y., Leung H.: Predicting Object-Oriented software maintainability using multivariate adaptive regression splines. The Journal of Systems and Software, pp.1349-1361, (2007).
[36] Zhou Y., XU B.: Predicting the Maintainability of Open Source Software Using Design Metrics. Wuhan University Journal of Natural Sciences, pp.14-20, (2008).
[37] Zhang W., Huang L., Vincent Ng V., Ge J.: SMPLearner: learning to predict software maintainability. The international journal of automated Software Engineering, pp.111-141, (2015).

Received. 5.07.2018, Accepted 16.11.2018