Towards Building Method Level Maintainability Models Based on Expert Evaluations

Péter Hegedűs[1], Gergely Ladányi[1], István Siket[2], and Rudolf Ferenc[1]

[1] University of Szeged, Department of Software Engineering
Árpád tér 2. H-6720 Szeged, Hungary
{hpeter,lgergely,ferenc}@inf.u-szeged.hu
[2] Hungarian Academy of Sciences, Research Group on Artificial Intelligence
Tisza Lajos krt. 103. H-6720 Szeged, Hungary
siket@inf.u-szeged.hu

Abstract. The maintainability of software systems is getting more and more attention both from researchers and industrial experts. This is due to its direct impact on development costs and reliability of the software. Many models exist for estimating maintainability by aggregating low level source code metrics. However, very few of them are able to predict the maintainability on method level; even fewer take subjective human opinions into consideration. In this paper we present a new approach to create method level maintainability prediction models based on human surveys using regression techniques.
We performed three different surveys and compared the derived prediction models. Our regression models were built based on approximately 150 000 answers of 268 persons. These models were able to estimate the maintainability of methods with a 0.72 correlation and a 0.83 mean absolute error on a continuous [0,10].

Keywords: Software maintainability, Regression analysis, ISO/IEC 9126, Comparative study

1 Introduction

Analyzing the maintainability of software systems is one of the core research topics in the field of software engineering. This is due to its direct impact on development costs and reliability of the software [3]. The development costs of a system with poor maintainability are significantly higher and unexpected errors are more likely to occur. This might be critical in many software domains, e.g. air traffic control, banking systems or energetics.

In our previous work [11] we built a maintainability model based on the ISO/IEC 9126 [13] standard by applying classification algorithms (using source code metrics as predictors) on manually labeled methods. The labeling of 350 Java methods was performed by 35 IT experts in such a way that each expert evaluated 10 different methods. Although the classification models worked well in classifying the maintainability of methods using 3 classes: *good*, *average*, *bad*; using a finer scale the precision of the models decreased. We have found out that this is due to the deviation of the experts' votes and the classification performs badly in an unbalanced training set (almost 70% of the labeled methods fell

into the *good* category). To overcome this problem we improved our surveys in two different ways: first, only one expert was asked to evaluate lots of methods while in the other case one method was evaluated by more participants and the evaluation score of the methods were calculated as the averages of the votes.

Moreover, instead of using classification we applied regression techniques to assess the tendency of maintainability on a much finer scale. As it defined in [18] predicting the values of numeric or continuous attributes is known as *regression* in the statistical literature. Regression differs from classification in that the output or predicted feature in regression problems is continuous. However, many standard classification techniques (e.g. neural networks, decision trees) can be adapted for regression. In this paper we present and compare the results of the regression models based on the following three surveys:

- **Experts' evaluation.** More experts evaluated the methods; every method was evaluated by only one expert.
- **One person's evaluation.** One expert evaluated all the methods; every method was evaluated by this expert.
- **Students' evaluation.** A large number of students evaluated the methods; every method was evaluated by at least 7 students.

Our regression models were built based on approximately 150 000 answers of 268 persons. These models were able to estimate the maintainability of methods with a 0.72 Pearson-correlation and a 0.83 mean absolute error on a continuous [0,10] scale where 0 means the absolutely not maintainable and 10 means the perfectly maintainable source code. With the improved surveys we tried to answer the following research questions:

Research question 1: *How effectively can we apply regression techniques to predict maintainability sub-characteristic on a continuous scale?*

Research question 2: *How is the prediction of regression models affected by the underlying surveys?*

The rest of the paper is structured as follows. In Section 2 we overview the related work. Then, in Section 3 we introduce the improved surveys and technical details about the performed evaluations. Section 4 presents the results of the surveys and the comparison of the different regression models. Afterwards, Section 5 discusses the known threats to the validity of our work. Finally, we conclude the paper and present future work in Section 6.

2 Related Work

The ISO/IEC 9126 model clearly defines the characteristics of software quality but it does not provide sufficient details about how one should calculate them in practice. Using the results of static source code analyzers is one of the most widespread solutions to calculate an external quality attribute from internal quality attributes [5]. There are several case studies about that the metrics are appropriate indicators for external quality attributes such as code fault proneness [7] [9] [15], maintainability [1] and attractiveness [14].

Dagpinar and Jahnke [8] indicated that size and coupling metrics are significant predictors for measuring maintainability of classes while inheritance,

cohesion, and indirect/export coupling measures are not. Contrary to them we investigated the maintainability on method level rather than class or system level. We used the following method level metrics: method coupling, complexity, size, number of coding rule violations and clone metrics.

Wang, et al. [17] compared different machine learning algorithms to predict software defects. They found that multiple classifiers (e.g. Bagging, Boosting, Random trees, Vote) can effectively improve classification performance of the single classifiers like Naive Bayes. The efficiency of the classification algorithms were tested on 14 datasets and the average accuracy of the best algorithm (Vote) was 88.48%. We also used single (Linear Regression, Neural Network) and multiple (Bagging) machine learning algorithms, but our purpose was not to compare the efficiency of these techniques, but to find out how well these algorithms are applicable to predict the maintainability sub-characteristics.

There are several models for calculating maintainability in a direct way [2] [4] [12] [16]. All of them use some kind of aggregation technique based on various metrics. For example Heitlager et al. [12] proposed an extension of the ISO/IEC 9126 model where metric values are split into five categories, from poor (--) to excellent (++). The evaluation in their model means summing the values for each attribute and then aggregating the values for sub-characteristics. Similarly to them we also tried to provide a bridge between source code metrics and the high level ISO/IEC 9126 quality characteristics, but we approximated maintainability on the level of methods and our estimation model is based on subjective opinions of many IT experts.

3 Applied Surveys

As a sequel of our previous work [11] 268 participants took part in the experiment where the three surveys introduced in Section 1 were performed and more than 150 000 questions were answered (for evaluation statistics see Table 1). The participants had to score the sub-characteristics of *maintainability* defined by the ISO/IEC 9126 standard (*analyzability*, *changeability*, *testability*, *stability*) and a new quality attribute, *comprehensibility*, introduced by us [11]. Besides these quality attributes, the students had to evaluate the maintainability of the methods as well. The evaluation has been performed with the help of our online survey system called Metric Evaluation Framework. For details about the application and technical questions refer to our earlier paper [11].

Experts' evaluation. First, 35 experienced software engineers dealing with software quality at our Software Engineering Department evaluated the 5 sub-characteristics of 350 different methods of jEdit open source text editor (`http://www.jedit.org`). One method was evaluated by only one participant and each participant evaluated 10 methods. The results pointed out that there was a large deviation in the judgments of the sub-characteristics which affected the efficiency of the built prediction models [11]. The cause of the large deviation can be that different experts might have different subjective scales and different interpretation of the same quality concepts. We tried to resolve this problem in two different ways: first, only one expert was asked to evaluate lots of methods

while in the other case one method was evaluated by more participants and the evaluation scores of the methods were calculated as the averages of the votes.

Table 1. Statistics about the evaluations

	Experts	One person	Students
Evaluators	35	1	232
Questions	13 407	11 901	125 097
Methods	350	250	200
System	jEdit	Industrial	jEdit

Table 2. The deviation of the justifications of the properties

Property	Deviation
Analyzability	1.859
Changeability	2.049
Stability	2.222
Testability	2.019
Comprehensibility	1.880
Maintainability	1.975

One person's evaluation. Our first attempt to eliminate the large deviation of the answers was that we asked a software engineer having 2 years experience to evaluate 250 methods of a closed source industrial system. Although we could build a more effective model (see Section 4) this result cannot be treated as a representative one as it might be specific to the given system and evaluator.

Students' evaluation. The next step was that 232 students having preliminary Java studies evaluated 200 methods. Because of the large number of participants, almost all methods were evaluated at least 7 different students and those methods which had less than 7 evaluations were excluded (about 10%). For each method the averages of the scores were calculated which approximated the student justifications of the given sub-characteristics and the maintainability. Table 2 shows the deviations of the scores given to the different maintainability characteristics. As we can see, the deviation is about 2 in all cases which are surprisingly large taking into account that the scores range from 0 to 10. This points out why it is difficult to build an effective model based on human evaluations. On the other hand, we have to remark that experts usually judge the methods similarly so they would have given more similar scores and the deviation would have been smaller. Unfortunately, we would have to involve lots of experts to prove this hypothesis which would be quite expensive.

4 Results

Applied regression techniques. In order to apply machine learning models effectively in practice, appropriate properties have to be chosen that can be calculated fast and automatically at the same time so we chose method level metrics in our experiment and we calculated them for all methods that were evaluated [11]. The list of considered metrics is the following: *lines of code*[3], *logical lines of code*[4], *number of statements*, *number of parameters*, *number of incoming invocations*, *number of outgoing invocations*, *number of foreign methods accessed*, *number of local methods accessed*, *McCabe's cyclomatic complexity*, *nesting level*, *clone coverage*, and *rule violations*[5]. This way we had all necessary information to build models that could predict maintainability and its sub-characteristics

[3] The end-line of the method minus its begin-line plus 1.
[4] All nonempty, non-comment lines of the method.
[5] Number of PMD (`http://pmd.sourceforge.net/`) rule violations of the method.

based on method level metrics. We used 10-fold cross-validation to evaluate the models. This means that the training data set was split into 10 disjoint parts and 9 of them were used to build the model and its usefulness was tested on the 10th part. Then this process was repeated ten times with different splitting.

In the classical form of machine learning, the unknown value being predicted is nominal, which means that it can have finite possible values and there is neither order nor ratio among them. In our previous work [11] we used three categories (good, average, bad) to classify the methods. In this case, one of the best measures of such learning is the rate of the correctly classified elements. Unfortunately, in that case almost 70% of the methods belonged to the *good* class and therefore the model classified almost all methods into the *good* category so too few bad methods were found what the real purpose of the experiment was.

Regression [18] is another frequently used technique to build models, where the unknown variable can be an arbitrary real number. The Pearson-correlation and the mean absolute error (MAE) are used to measure the usefulness of the model, more precisely, to measure the differences between the expected values and the values given by the model. One of the advantages of regression is that we use continuous scale so we expect more precise results. Furthermore, the correlation tells us how well the model hits the tendency while the MAE indicates how much the model differs from the expected values which are more useful information than those received in the nominal case. This is why we did not use the standard IR measures like precision and recall.

Comparing the different algorithms. In this experiment we applied neural networks, linear regression and decision tree techniques. We used the Weka data mining tool [10] to build appropriate models. First, we examined the efficiency of the three techniques on the results of the students' evaluation, then we compared the results of the three different method evaluations. Weka offers only one option for neural networks and linear regression, but in case of decision trees we chose the one that worked the best for us. This was the REPTree algorithm but it was further improved with a bagging technique [6] which builds more trees based on the learning data set and the prediction is combined by the average of their predictions. Besides the correlation and MAE the efficiency of the results can be measured by comparing to the ZeroR algorithm, whose prediction is always the average of the predicted values in the training set. Without using the metrics as predictors this technique gives the prediction with the smallest average error so we can compare how much the result improves when the metrics are used.

First, we compared the different regression algorithms on the students' evaluation. We calculated the correlation values and the MAEs of all models (see Table 3). As we can see the decision tree has significantly larger average correlation value (0.631) and significantly smaller MAE value (0.87) than the others.

In the following we compared the different evaluations as well. Since the decision tree gave the best results, we applied it in our further investigations.

Comparing the evaluations. We compared the models trained on the three different survey results to see which gives the best results. The results of the model built by decision tree are presented in Table 4.

Table 3. The MAE and Correlation values of the examined regression techniques

	ZeroR		Neural Network		Linear Reg.		Decision Tree	
	MAE	Corr.	MAE	Corr.	MAE	Corr.	MAE	Corr.
Analyzability	1.201	-0.162	1.076	0.408	1.076	0.466	0.884	0.660
Changeability	1.026	-0.116	1.088	0.362	0.965	0.437	0.861	0.571
Comprehens.	1.574	-0.153	1.387	0.275	1.188	0.491	1.048	0.621
Stability	0.822	-0.239	0.824	0.297	0.833	0.360	0.670	0.572
Testability	1.189	-0.118	1.168	0.427	1.145	0.363	0.926	0.639
Maintainability	1.187	-0.122	1.193	0.587	0.909	0.615	0.831	0.723
Average	**1.166**	**-0.152**	**1.123**	**0.393**	**1.019**	**0.455**	**0.870**	**0.631**

Table 4. Efficency of the decision tree algorithm based on the different surveys

	Experts		One Person		Students	
	MAE	Corr.	MAE	Corr.	MAE	Corr.
Analyzability	1.792	0.479	0.896	0.660	0.884	0.660
Changeability	1.656	0.445	1.011	0.758	0.861	0.571
Comprehensibility	1.867	0.395	1.063	0.712	1.048	0.621
Stability	1.712	0.509	1.154	0.453	0.670	0.572
Testability	1.910	0.520	1.781	0.476	0.926	0.639
Average	**1.787**	**0.469**	**1.181**	**0.612**	**0.878**	**0.612**

The 0.469 average correlation value and the 1.787 average MAE value of the model trained on experts' evaluation show that the decision tree algorithm could not build an effective model. On the other hand, it is interesting that if we consider the correlation only, the model based on experts' evaluation predicts stability and testability better than the model based on the result of the one person's evaluation. The average correlation of the other two models is the same but the average MAE value of the students' evaluation is smaller.

Answering the research questions. RQ1: Neural network and linear regression performed poorly in our experiment so we can say that they are not able to predict the maintainability sub-characteristics efficiently. On the other hand the REPTree decision tree method gave good results in all cases therefore it can be considered an effective regression technique.

RQ2: Because of the preliminary results only decision tree was applied to answer this question. On the students' evaluation it performed uniformly well while on the one person's evaluation it could not predict stability and testability values efficiently. On the other hand, we were not able to predict the results of the experts' evaluation.

5 Threats to Validity

It is common to collect large amount of data with the help of students, but it is always a huge risk as well. The risk in our case is that the opinion of a student is much less reliable than an expert's opinion. To handle this threat we used the averages of lots of student opinions about the quality of the methods. This way we decreased the effect of the unreliable votes.

We compared the efficiency of the regression techniques based on the different surveys, but we left out of consideration that only two of the surveys evaluated the same system. The one person evaluation is based on an industrial system.

Although it is possible that the subject systems have an effect on the efficiency of applied regression techniques, our results are mainly based on the experts' and students' surveys, which use the same subject systems.

The used regression algorithms were trained on 350, 250 and 200 methods, but to accept the achieved results in general a larger amount of data might be needed. However, even these survey results are valuable assets considering the huge number of human evaluators involved.

6 Conclusions and Future Work

In this paper we presented a way to build prediction models for maintainability based on human evaluations. We performed three surveys with different set-ups: experts, one person, and students evaluated a large number of Java methods. By comparing the results of the three evaluations we can conclude that the experts' survey provided the hardest predictable opinions due to the high deviation in the different expert votes (each expert evaluated different methods). On the contrary, the one person and student opinions were equally well predictable by the means of correlation but the mean average error is significantly smaller in the case of student evaluations.

Looking at the different regression techniques we can say that on our training data the decision tree algorithm was the best performing one. The model trained on the students' evaluation predicts the quality attributes with 0.61 correlation and 0.88 mean average error on a [0,10] continuous scale. The maintainability itself is predicted by the model with 0.72 correlation and 0.83 average error. Based on these results we can conclude that efficient maintainability prediction models can be built using regression techniques when we have a large amount of reliable subjective evaluations from one person or less reliable but redundant evaluations from multiple persons.

An interesting question is that how the results of existing maintainability models correlate with the opinions of human evaluators. Our future plan is to improve our probabilistic quality model [2] to enable method level qualifications and compare its results with the human evaluations.

We also plan to extend the questionnaire of the evaluation and apply different kind of predictors for the model building (not just basic metrics).

Acknowledgements

This research was supported by the Hungarian national grant GOP-1.1.1-11-2011-0006 and the European Union co-funded by the European Social Fund, TÁMOP-4.2.2/B-10/1-2010-0012.

References

1. Bagheri, E., Gasevic, D.: Assessing the Maintainability of Software Product Line Feature Models using Structural Metrics. In: Software Quality Journal 19(3):579-612. Springer (2011)
2. Bakota, T., Hegedűs, P., Körtvélyesi, P., Ferenc, R., Gyimóthy, T.: A Probabilistic Software Quality Model. In: Proceedings of the 27th IEEE International Conference on Software Maintenance. pp. 368–377. ICSM 2011, IEEE Computer Society, Williamsburg, VA, USA (2011)

3. Bakota, T., Hegedűs, P., Ladányi, G., Körtvélyesi, P., Ferenc, R., Gyimóthy, T.: A Cost Model Based on Software Maintainability. In: Proceedings of the 28th IEEE International Conference on Software Maintenance. ICSM 2012, IEEE Computer Society, Williamsburg, VA, USA (2012)
4. Bansiya, J., Davis, C.G.: A Hierarchical Model for Object-Oriented Design Quality Assessment. IEEE Transactions on Software Engineering 28, 4–17 (2002)
5. Barbacci, M., Klein, M., Longstaff, T., Weinstock, C.: Quality Attributes. Tech. rep., Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-95-TR-021 (1995)
6. Breiman, L.: Bagging Predictors. Machine Learning 24(2), 123–140 (Aug 1996)
7. Briand, L.C., Wüst, J., Daly, J.W., Porter, D.V.: Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems 51(3), 245–273 (2000)
8. Dagpinar, M., Jahnke, J.H.: Predicting Maintainability with Object-Oriented Metrics - An Empirical Comparison. In: Proceedings of the 10th Working Conference on Reverse Engineering. pp. 155–164. WCRE '03, IEEE Computer Society, Washington, DC, USA (2003)
9. Gyimóthy, T., Ferenc, R., Siket, I.: Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction. IEEE Transactions on Software Engineering 31(10), 897–910 (Oct 2005)
10. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations Newsletter 11(1), 10–18 (Nov 2009)
11. Hegedűs, P., Bakota, T., Illés, L., Ladányi, G., Ferenc, R., Gyimóthy, T.: Source Code Metrics and Maintainability: a Case Study. In: Proceedings of the 2011 International Conference on Advanced Software Engineering And Its Applications (ASEA 2011). pp. 272–284. Springer-Verlag CCIS (Dec 2011)
12. Heitlager, I., Kuipers, T., Visser, J.: A Practical Model for Measuring Maintainability. In: Proceedings of the 6th International Conference on Quality of Information and Communications Technology. pp. 30–39. QUATIC '07, IEEE Computer Society, Washington, DC, USA (2007)
13. ISO/IEC: ISO/IEC 9126. Software Engineering – Product quality. ISO/IEC (2001)
14. Meirelles, P., Santos Jr., C., Miranda, J., Kon, F., Terceiro, A., Chavez, C.: A Study of the Relationships between Source Code Metrics and Attractiveness in Free Software Projects. In: Proc. of the Brazilian Symposium on Software Engineering. pp. 11–20. SBES '10, IEEE Computer Society, Washington, DC, USA (2010)
15. Olague, H.M., Etzkorn, L.H., Gholston, S., Quattlebaum, S.: Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes. IEEE Transactions on Software Engineering 33(6), 402–419 (Jun 2007)
16. Oman, P., Hagemeister, J.: Metrics for Assessing a Software System's Maintainability. In: Proceedings of the Conference on Software Maintenance. vol. 19, pp. 337–344 (Nov 1992)
17. Tao, W., Weihua, L., Haobin, S., Zun, L.: Software Defect Prediction Based on Classifiers Ensemble. Journal of Information & Computational Science 8(16), 4241–4254 (2011)
18. Uysal, I., Güvenir, H.A.: An Overview of Regression Techniques for Knowledge Discovery. Knowledge Engineering Review 14(4), 319–340 (Dec 1999)