A Systematic Review of Software Maintainability Prediction and Metrics

Mehwish Riaz, Emilia Mendes, Ewan Tempero
Department of Computer Science
The University of Auckland, New Zealand
mria007@aucklanduni.ac.nz, {emilia, e.tempero}@cs.auckland.ac.nz

Abstract

This paper presents the results of a systematic review conducted to collect evidence on software maintainability prediction and metrics. The study was targeted at the software quality attribute of maintainability as opposed to the process of software maintenance. The evidence was gathered from the selected studies against a set of meaningful and focused questions. 710 studies were initially retrieved; however of these only 15 studies were selected; their quality was assessed; data extraction was performed; and data was synthesized against the research questions. Our results suggest that there is little evidence on the effectiveness of software maintainability prediction techniques and models.

1. Introduction

Software maintainability, the ease with which a software system can be modified [13], is an important software quality attribute. Intrinsically associated with this quality attribute is the maintenance process, which has long been known to represent the majority of the costs of a Software Development Life-Cycle (SDLC) [3], [33]. Therefore, the maintainability of a software system can significantly impact software costs. This means that it is important to be able to forecast a software system's maintainability so to effectively manage costs. Research into software maintainability prediction includes proposing and validating maintainability predictors based on measurable factors that have a bearing on the software maintenance activity [3].

Prediction, commonly known as estimation [15], is an important part of project planning [36]. Estimates can be made for projects or processes as well as products. When these are made for projects, these are called effort estimates and the process is called effort estimation [22] or software cost estimation [15], [20]. When estimates are made to a maintenance process, the means of obtaining such estimates is called maintenance cost prediction [11] or maintenance project effort estimation [1]. In addition, estimates of quality attributes give a measurable value of the quality of the attribute that a software product possesses. The focus of this study is on the measurement and prediction of the quality attribute of maintainability.

In order to differentiate between maintenance and maintainability we consider the following definitions:

Software maintenance is defined as "the process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment" [13].

Software maintainability is defined as "the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment" [13].

From the definitions it is clear that maintenance is the process performed as part of the SDLC whereas maintainability is the quality attribute associated with the software product. These are two inherently different but interlocked concepts. Maintenance vs. maintainability represent process vs. quality attribute and their predictions are called process cost prediction vs. quality attribute measurement, respectively.

A software maintainability prediction model enables organizations to predict the maintainability of their software systems, thus providing a means to better manage their maintenance resources in addition to adopting a defensive design [24]. This can then help in reducing the maintenance effort and therefore, reducing the overall cost and time spent on a software project.

Given how an improved understanding of software maintainability can help organizations create better systems, it is important to understand this process in order to make significant contributions to this area. Therefore this paper presents a Systematic Review (SR) of software maintainability prediction and metrics

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

in order to understand its current state of the art and to identify opportunities for future research.

The remainder of the paper is organized as follows: Section 2 describes the steps followed for the SR. Section 3 discusses the SR findings. A discussion of the results, gaps in the research area, threats to validity, and road blocks are given in Section 4 followed by conclusions and suggestions on possible future work in Section 5.

2. Systematic review

An SR provides the means to identify, evaluate and interpret all available research relevant to a particular research question, topic area, and phenomenon of interest [17], [25].

This section provides the details of the steps that were followed for conducting this SR. For complete guidelines for conducting SRs, refer to Kitchenham [16], [17].

2.1. Research questions

Specifying the right research question(s) is very important for an SR in order to present detailed and suitable findings of a given domain, and to help identify and scope future research activities [17]. The research questions for this SR were structured with the help of the Population, Intervention, Comparison, Outcome, Context (PICOC) criteria [25]. However, as the objective of this SR was not to find evidence about the comparison of approaches, methods or models; the 'Comparison' part of the PICOC was excluded. The population considered is that of software applications; our intervention was methods/techniques to predict maintainability and maintainability predictors; the outcomes of interest were accuracy of maintainability prediction methods/techniques and successful maintainability predictors; and all kinds of empirical studies and study settings encompassed the context.

The research questions addressed by our SR are:

Question 1: What evidence is there for maintainability forecasting techniques/methods for software applications?

Question 2: What forecasting techniques/methods have been used to predict maintainability for software applications?

Question 2 (a): What measures have been used for measuring accuracy of the maintainability prediction models/techniques for software applications?

Question 2 (b): What were the numeric values obtained for the prediction accuracy of the maintainability prediction models/techniques for software applications?

Question 2 (c): Which cross-validation method was used to find the accuracy of the prediction techniques for maintainability in software applications?

Question 2 (d): Was there a forecasting technique/method reported to be significantly superior for predicting maintainability for software applications, and if so which one?

Question 3: What factors and metrics have been investigated as maintainability predictors for software applications? Which of these are considered successful predictors?

Question 3 (a): At what stage of the project are these predictors gathered?

Question 3 (b): If previous studies predict various types of maintenance and facets of maintainability, what were the different types of predictors used for each of them?

Question 4: How is maintainability understood and measured in the context of software applications and which quality model(s) does it comply with?

The first research question is our primary research question whereas the last three and their associated sub-questions are the secondary research questions.

2.2. Search strategy used for primary studies

The search terms used in our SR were developed using the following steps [16], [17]:

1. Derive major search terms from the research questions by identifying Population, Intervention, Outcome, and Context.
2. Identify keywords in the relevant papers.
3. Identify alternative spellings and synonyms for the search terms with the help of a thesaurus.
4. Use Boolean OR to construct search strings from the search terms with similar meanings.
5. Use Boolean AND to concatenate the search terms and restrict the research.

The resulting search string is given below:

(software application OR software OR system OR software system OR application) AND (method OR Process OR technique OR system OR practice OR procedure) AND (factor OR feature) AND (metric OR measure OR measurement) AND (predict OR forecast OR calculate OR prediction OR estimation) AND (maintainability OR analysability OR analyzability OR changeability OR stability OR testability OR compliance OR maintenance OR maintenance process OR enhancive maintenance OR adaptive maintenance OR corrective maintenance OR preventive maintenance) AND (quality model) AND (successful)

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

2.3. Search process

Our search process comprised two phases: *primary search* and *secondary search*.

2.3.1. Primary search process. The primary search process was directed towards searching online databases, search engines, electronic journals, conference proceedings, and grey literature using the derived search string. The resources used were chosen because (1) they were used by previous SRs in software engineering, (2) they were suggested by the University of Auckland Library website as relevant to the subject, and (3) the venues contained publications that were relevant to our areas of interest. Only the online databases that publish peer-reviewed articles were considered. As a result, Agile Alliance was removed from the list after a confirmation from Agile Alliance that it contains publications that are not peer reviewed. Also, IEEE Computer Society Digital Library and Lecture Notes in Computer Science were removed from the list after confirmation from the respective publishers that these provide the same results as IEEE Xplore and SpringerLink, respectively.

In order to make the search process comprehensive, our searches included literature published from (and inclusive of) year 1985 to Nov. 11th, 2008. The year 1985 was chosen as a preliminary search found no results previous to it.

The original comprehensive search string given above did not yield any results. To address this, we broadened the search criteria and tailored the strings to each database on the recommendations of the subject librarian of the University of Auckland. Using broader search criteria on full text searches lead to thousands of results, most of which were irrelevant. In the end, searches were restricted to titles and abstracts only. Table 1 summarizes the results of the searches and the sources on which they were ran.

Table 1. Summary of search results

No	Database Name	No. of Search Results	No. of Duplicates found	No. of Relevant Articles
1	Scopus	334		8
2	IEEE Xplore	11	7	0
3	Current Contents	6	6	0
4	Computer Database	40	3	4
5	Science Direct	19	17	0
6	Springer Link	17	3	1
7	Inspec	27	21	1
8	ACM Digital Library	255	16	0
9	ProQuest Computing	1	1	0
	Total	**710**		**14**

Note that the sequence of the databases in table 1 represents the same sequence in which they were searched and the number of duplicates in the databases is also due to this sequence.

For details of the search strings and fields searched for each database, refer to Riaz [32].

The search process applied on other resources did not yield any new results for the electronic journals and conference proceedings, and also did not yield any relevant results for search engines and grey literature.

2.3.2. Secondary search process. The secondary search process consisted of three sub-processes: 1) review references for the selected primary studies to find any relevant articles, 2) review citations for the selected primary studies to find any relevant articles, also known as a backward pass [30], 3) identify and contact selected primary studies' authors for any unpublished work, if the need arises.

Note that the secondary search process and the study selection process (Section 2.5) were performed iteratively until no new studies were identified.

2.4. Inclusion and exclusion criteria for study selection

The criteria to include studies in our SR were to include any study that was either:

1. performing maintainability (any facet/sub-characteristic and/or type of maintainability as defined in numerous quality models) prediction for software applications AND providing empirical validation of its findings, OR
2. discussing maintainability (any facet/sub-characteristic and/or type of maintainability as defined in numerous quality models) measures or metrics for software applications AND providing empirical validation of its findings.

Any study that discussed prediction or metrics of quality attributes other than maintainability e.g., fault-proneness etc., or discussed prediction or metrics of maintenance instead of maintainability, was excluded.

2.5. Study selection

The study selection phase comprised two processes: *initial selection* and *final selection*. During the initial selection phase, screening of titles and abstracts for the potential primary studies was performed by all the three authors against the inclusion/exclusion criteria. For almost 30%, no decision could be made just on title and abstract, as these papers did not make a clear distinction between maintenance prediction and maintainability prediction. For these cases, final

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

selection involved examining full papers to make the decision.

After performing this step, a total of 14 studies were selected. This number was obtained after the primary search process, as indicated in table 1. During the secondary search process, references and citations for the 14 selected studies were reviewed, and led to the inclusion of another 5 studies. Out of these, one study did not meet the inclusion criteria and was excluded. Out of the remaining four, three did not obtain a good score on the quality assessment (see section 2.6) and were also excluded. As a result, a total of 15 studies were selected. Table 2 gives the list of selected studies.

Table 2. List of selected studies

Study ID	Authors	Year
S1 [24]	Oman, P., and Hagemeister, J.	1994
S2 [4]	Coleman, D., Ash, D., Lowther, B., and Oman, P.	1994
S3 [5]	Coleman, D., Lowther, B., and Oman,P.	1995
S4 [31]	Welker, K.D., Oman, P.W.	1997
S5 [27]	Schneberger, S. L.	1997
S6 [8]	Ferneley, E.H.	1999
S7 [10]	Genero, M., Olivas, J., Piattini, M., and Romero, F.	2001
S8 [7]	Di Lucca, G.A., Fasolino, A.R., Tramontana, P., Visaggio, C.A.	2004
S9 [21]	Misra, S.C.	2005
S10 [19]	Lim, J. S., Jeong, S. R.,and Schach, S.R.	2005
S11 [26]	Pizzi, N.J., and Pedrycz, W.	2006
S12 [29]	Van Koten, C., Gray, A.R.	2006
S13 [28]	Shibata, K., Rinsaka, K., Dohi, T., and Okamura, H.	2007
S14 [34]	Zhou, Y., and Leung, H.	2007
S15 [35]	Zhou, Y., and Xu, B.	2008

2.6. Study quality assessment

For study quality assessment, quantitative and qualitative checklists were developed. The quality checklists are a means to assess the quality of the selected studies and therefore their importance as evidence to answer the research questions. These checklists can also provide a further criterion for inclusion of studies [17] as those scoring very low on quality are candidates to be excluded because they do not provide reliable evidence. None of our selected studies were qualitative and so the studies were assessed using only the quantitative checklist. The checklists were compiled from questions suggested by Crombie [6], Fink [9], Petticrew [25], and Kitchenham [17]; and rephrased according to our needs. This resulted in 19 questions (refer to Riaz [32] for a complete list), and each question assessed a quality facet of a study using a three point scale: 'Yes', 'No', or 'Partially', with associated scores of 1, 0, and 0.5, respectively. Therefore, a study could score between values 0 and 19. We selected 7 as a cutoff point and therefore, any study that scored 7 or below was excluded from the list of selected studies. Table 3 gives a summary of the scores obtained for each selected study.

2.7. Data extraction

Data extraction from all selected studies was performed by the first author, and then compared to data extracted of a subset of selected studies by the third author to check for consistency. A sample of the extracted data is available from Riaz [32].

When the necessary data was not available in the study, the authors were contacted where possible. If contact was not possible, or no reply was received, a conservative answer was recorded. For example, one quality related question "Are potential confounders adequately controlled for the analysis?" was not answered in any of the selected studies. Only one contactable author responded, and so a 'Partially' answer was recorded for the remaining studies.

3. Results

The summary data were generated by tabulating results against each research question. The results against each research question are discussed in the subsequent sections. Due to space constraints, complete summary tables for each question could not be included in this paper. For detailed summary tables refer to Riaz [32].

3.1. Question 1

Question 1 of the SR was directed towards finding evidence on software maintainability prediction techniques/methods. The results of the data synthesis are briefly summarized in Table 4.

Table 3 Study Quality Scores

Study ID	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Total Score	10.5	7.5	7.5	14.5	13.5	15	9	11	14.5	14	7.5	16.5	9	16	12.5

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

Table 4 Summary of Evidence for Software Maintainability Prediction Techniques

Study ID	Maintainability Facet/Maintenance Type	Forecasting Technique
S1		3 regression based models 1. Single metric model based on Halstead's E: 2. A four-metric polynomial model based on Halstead's E , McCabe's V(G), LOC, and CMT 3. A five-metric linear regression model based on Halstead's E , McCabe's V(G), LOC, CMT, and a subjective evaluation of the external documentation and build complexity
S2		1. HPMAS 2. Polynomial maintainability assessment model a. Four-metric polynomial based on Halstead's effort metric and on metrics measuring extended cyclomatic complexity, lines of code, and number of comments b. Revised four-metric polynomial based on Halstead's volume metric and on metrics measuring extended cyclomatic complexity, lines of code, and percentage of comments
S3		Four-metric polynomial maintainability assessment model based on Halstead's E, V(G'), LOC, and percentage of comments
S4		1. Improved, three-metric MI model 2. Improved, four-metric MI model
S7	Understandability, Analysability, and Modifiability	Fuzzy Prototypical Knowledge Discovery (FPKD) used for prediction based on Fuzzy Deformable Prototypes
S8		Web Application Maintainability Model (WAMM)
S9		6 models based on multivariate regression analysis
S11		Stochastic Feature Selection (SFS) 1. The best single result generated by SFS (SFSs) 2. Aggregated result using the fuzzy integration approach (SFSf)
S12		1. Bayesian Network - Naïve-Bayes Classifier 2. Regression Models a. Regression Tree b. Multiple linear regression model using backward elimination c. Multiple linear regression model using stepwise selection
S13		3 Non-Homogeneous Poisson Process based Software Reliability Models (NHPP-based SRMs): 1. Exponential SRM 2. S-Shaped SRM 3. Rayleigh SRM
S14		Multivariate Linear Regression (MLR), Artificial Neural Network (ANN), Regression Tree (RT), Support Vector Regression (SVR), & Multivariate Adaptive Regression Splines (MARS)
S15		1. Univariate Linear Regression Analysis 2. Multivariate Linear Regression model (MLR) by stepwise variable selection procedure

The collected evidence shows the use of various types of prediction models and techniques, where the most common models observed from the synthesized data (S1, S2, S3, S4, S9, S12, S14, S15) are models that use algorithmic techniques for prediction: regression trees (S14), linear regression (S1 to S4, S9, S12), univariate linear regression (S15), variants of multivariate linear regression (S14, S15), and support vector regression (S14). The other techniques/models used are hierarchical dimensional assessment model - Hewlett Packard's software Maintainability Assessment System (HPMAS) (S2), Web Application Maintainability Model (WAMM) (S8); and those based on fuzzy deformable types (S7), stochastic feature selection (S11), Bayesian Networks (S12), and software reliability models (S13).

The models proposed in S2 and S8 are similar as they both calculate maintainability based on hierarchical models that have metrics at the leaf nodes. The metrics' values are averaged along with their respective weights (S2) or summed up after multiplying their weights to obtain a quantifiable value of maintainability (S8). However, it should be noted that the model proposed in S8 is specific to Web applications and therefore, may not be generalized for all software applications.

The models in studies S1 to S4 were proposed by the same group of authors who refined them over a period of time. Therefore, given these models were refined further, and were also applied in different studies to broader categories of software applications, these were considered as separate studies.

Note that S6 used software design level metrics to predict maintainability and S5 and S10 provided a list of factors that contribute to maintainability; as such, they do not present any prediction models and therefore were not included in Table 4.

In addition to the evidence on software maintainability prediction techniques & models, we also gathered evidence on which technique was used

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

with which maintainability sub-characteristic (e.g., analyzability, changeability, stability, testability, or compliance) or maintenance type (e.g., corrective, preventive, enhancive, or adaptive). Our results show that none of the prediction models proposed by any of the selected studies dealt with any specific type of maintenance, except for S7, which dealt with the sub-characteristics of understandability, analyzability, and modifiability.

To summarize, the most commonly used technique observed for software maintainability prediction was algorithmic, and maintainability prediction techniques seemed to be equally applied to any sub-characteristic of maintainability, or any type of maintenance.

3.2. Question 2

Question 2 was directed towards finding details on the use of software maintainability prediction techniques, where sub-questions 2(a), 2(b), and 2(c) investigated accuracy measures, prediction accuracy values, and cross-validation methods, respectively. Sub-question 2(d) investigated techniques & models that were superior in terms of their prediction accuracy.

The important findings against questions 2(a), 2(b), 2(c) and 2(d) are given in Table 5. The answer to Question 2 can be found in Table 4, which gives the list of prediction techniques used by the selected studies for maintainability prediction.

Table 5 Summary of Evidence on Measures of Accuracy and Models Predicted to be Superior

Study ID	Forecasting Technique/Method	Accuracy Measure	Value of Accuracy Measure	Model reported to be superior
S1	1. Single metric model 2. Four metric model 3. Five metric model			Five metric model
S2	1. HPMAS 2. Polynomial maintainability assessment model a. Four-metric polynomial b. Revised four-metric polynomial	Average residual for comparison between two polynomial based models.	Less than 1.4	Revised four-metric polynomial
S4	1. Improved 3-metric model 2. Improved 4- metric model			If comments are meaningful, then 4-metric model, otherwise 3-metric model
S9	6 models based on multiple regression	Standard estimation error	4.994387	Model 6
S11	Stochastic Feature Selection (SFS) 1. Best single result (SFSs) 2. Aggregated using the fuzzy integration (SFSf)	Proportion representing the number of times the model was correct	0.58 for SFSs and 0.75 for SFSf 10-fold random validation	Aggregated using the fuzzy integration (SFSf)
S12	1. Naïve-Bayes Classifier 2. Regression Tree 3. Multiple linear regression model using backward elimination 4. Multiple linear regression model using stepwise selection	1. *Absolute Residual Error (ARE)*: Sum, Median and Standard deviation of ARE 2. *Magnitude of Relative Error (MRE)*: Maximum value of MRE 3. *Pred*: pred(0.25) & pred(0.30)	Refer to Riaz [32] for details 10-fold cross-validation	Bayesian Network model for first data set (UIMS). Multiple linear regression models using both backward and stepwise selection are both best for second dataset (QUES)
S14	1. Multivariate Linear Regression (MLR) 2. Artificial Neural Network (ANN) 3. Regression Tree (RT) 4. Support Vector Regression (SVR) 5. Multivariate Adaptive Regression Splines (MARS)	Same as above	Refer to Riaz [32] for details Leave-one-out (LOO) cross validation	For first data set (UIMS), both MARS and SVR were reported best. For second data set (QUES), MARS is reported to be the best.
S15	1. Univariate Linear Regression Analysis 2. Multivariate Linear Regression (MLR) model by using stepwise variable selection procedure	1. Coefficient of determination of the regression model (R^2 between actual and predicted maintainability) 2. Absolute Relative Error (ARE) 3. Magnitude of Relative Error (MRE)	1. $R^2 = 0.752$ (filtered model) and $R^2 = 0.471$ (all observations) 2. Mean ARE = 18.563 3. MRE = 1.005 or 100.5% Leave-one-out (LOO) cross-validation	Multivariate Linear Regression model (MLR model)

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

3.3. Question 3

Question 3 investigated metrics and factors used for maintainability prediction, and which of these were successful predictors. Sub-questions 3(a) and 3(b) were to inform the project stage at which metrics were measured and the type of maintenance and sub-characteristic of maintainability for which these metrics were used as maintainability predictors. We list only the successful maintainability predictors.

3.3.1. Successful predictors gathered at source code level. Table 6 presents a list of successful software maintainability predictors reported in studies S1 to S4, S11, S12, S14 and S15, which are measured at source code level or project's post-implementation stage.

3.3.2. Successful predictors gathered at source code level specific to Web-based Software Applications. In addition to the predictors shown in Table 6, study S8 lists successful software maintainability predictors measured at the post-implementation stage or source code level that are specific to Web-based software applications. These predictors are listed below, arranged in categories:

Size metrics: TotalWebPage#, TotalLOC#, WebObject#, ServerScripts#, ClientScripts#, and TotalLanguages#

Complexity metrics: Total Cyclomatic Complexity

Control Coupling metrics: Total WebPageRelationships# / TotalWebPage#

Data Coupling metrics: Total WebPageDataCoupling / TotalWebPage#

3.3.3. Successful predictors gathered at the system design stage. Study S6 reports the following successful software maintainability predictors gathered at the system design stage: Control flow complexity of each unique information flow terminating (or emanating) 'CF', spinal complexity (control flow history) 'SC', Fan-out 'FO', Fan-out complexity total 'FO-CF', and Fan-out multiplicity complexity total 'FO-MCF'.

3.3.4. Factors that impact software maintainability. Study S5 provided a list of factors observed to either increase or decrease software maintainability: PC developments tools, PC graphical interfaces, PC packaged software, and PC programming languages increased maintainability; variety of processors, number (variety) of system software programs, rate of technological change, variety of telecommunication protocols, and variety of data storage formats decreased maintainability.

Table 6. Successful software maintainability predictors gathered at source code level

Study ID	Metrics	Study ID	Metrics
S1,S2 S3,S4	Halstead's Effort 'E'	S11	Mean tokens per method
S1	Avg. effort per module 'ave-E'	S11	Mean no. of decisions per method
S1,S2 S3, S4 S11	Halstead's volume 'V'	S11 S12	Lack of Cohesion of Methods 'LCOM'
S1,S4 S12,S 14	Lines of code 'LOC', 'SIZE1'	S11	Halstead's difficulty 'D'
S1	Halstead's predicted length 'N^'	S11 S12 S14	Data Abstraction Coupling 'DAC'
S1	Halstead's length 'N'	S11	Halstead's program vocabulary 'n'
S1	Avg. volume per module 'ave-V'	S11	Halstead's unique operands count'n2'
S1	Avg. no. of comment lines per module 'ave-CMT'	S11	Halstead's unique operators count 'n1'
S1	Avg. LOC per module 'ave-LOC'	S11	Maximum number of operations
S1	Avg. extended cyclomatic complexity per module 'ave-V(g')'	S11	No. of overridden methods
S1	Avg. predicted length per module 'ave-N^'	S11	Percentage of private member
S1	Avg. length per module 'ave-N'	S12	Depth of Inheritance Tree 'DIT'
S1,S4	extended cyclomatic complexity 'V(G')'	S12 S14	NO. of Children 'NOC'
S1	Total purity ratio 'PR'	S12	Response For a Class 'RFC'
S1	Avg. purity ratio per module 'ave-PR'	S12 S14	Weighted Method per Class 'WMC'
S1	Sum of the avg. variable spans 'SPN'	S12 S14	Message-Passing Couple 'MPC'
S1	Total no. of executable semicolons 'NES'	S12	No. Of Methods 'NOM'
S1,S4	Number of comment lines 'CMT'	S12	Number of properties 'SIZE2'
S1	average variable span per module 'ave-SPN'	S15	Avg. complexity per method 'OSAVG'
S1	Avg. no. of executable semicolons 'ave-NES'	S15	Avg. no. of methods per class 'CSO'
S4	cyclomatic complexity	S15	Avg. attributes per class 'CSA'
S11	Median lines of code per object method	S15	Avg. no. of children per class 'SNOC'
S11	Type of object		

Studies S7 and S9 suggested several metrics for maintainability prediction; however, they did not

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

document successful maintainability predictors. S5 and S10 discussed factors impacting upon maintainability; however did not provide any metrics to be used to measure such factors; S5 provided a list of factors that increased/decreased maintainability; and study S13 discussed no predictors or metrics for the created maintainability prediction models.

Of the abovementioned successful predictors, most are gathered at the source code level. Only those given by S6 are gathered at the design level.

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

Out of the 15 studies in the SR, 10 measured successful maintainability predictors at a project's source code level whereas two measured at the design level. Two studies focused on Maintainability factors, as opposed to metrics: one discussed these factors taking into account the entire SDLC, whereas the other did not explicitly mention the project stage at which those factors should be considered. One study - S13, suggested no metrics and employed prediction models at the maintenance stage. However, it was not clear as to what stage of the maintenance process the model was most accurate.

The metrics gathered at the source code level were all applied to maintainability in general and to no type of maintenance or maintainability facet in particular.

The metrics gathered at the design level applied to corrective and preventative maintenance and to no particular maintainability facet.

The class level metrics (S7) are applicable to maintainability facets of understandability, analyzability, and modifiability but to no specific maintenance type. As S7 did not provide any successful maintainability predictors, they are not provided in the above list.

3.4. Question 4

Question 4 was directed towards investigating how maintainability was understood and measured in the context of software systems and which quality model it complied to. Table 7 gives a summary of the definitions used in different studies.

The definitions provided by studies S2 to S5, S7 and S10 conform to the software maintainability definition used herein. The definitions provided by studies S6, S12 to S14 varied but were related to either errors in the software and their correction, or to changes made to the software. Studies S1, S8, S9, and S15 did not provide any explicit definition for maintainability.

Table 7. Definitions for maintainability from selected studies

Study ID	Maintainability/Maintainability Facet and Definition
S2, S3, S4	The ease with which a software application or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment
S5	Software maintenance difficulty is the relative degree to which software maintenance is hard to understand, perform, and manage.
S6	The article takes *error rate* as independent variable and suggests measures that can be used to highlight features in software design that may require future corrective or preventative maintenance activity.
S7	*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.
S10	The ease with which a software system can be modified to correct a fault or conform to changed requirements.
S11	How difficult it is to make small or incremental changes to an existing software object without introducing errors in logic or design.
S12, S14	The number of changes made to the code during a maintenance period.
S13	The probability that the detected fault can be corrected and removed by an arbitrary time *t*.

The selected studies also provided evidence on the use of metrics to quantify software maintainability. Table 8 presents a summary of our findings.

The SR results suggested that ordinal scale maintainability metrics were the means most commonly used to measure maintainability, followed by change metrics and Maintainability Index (MI).

The results of our SR also showed that most studies did not consider any type of maintenance in particular while discussing maintainability. Only S6 considered corrective (to correct errors) and preventive (to update the software in anticipation of future problems) maintenance [8]. S2 and S3 provided definitions for corrective, adaptive, and perfective maintenance but did not use these concepts further.

Finally, only study S10 mentioned a quality model in which maintainability was based upon (ISO/IEC 12207). No other study discussed maintainability in light of any quality model.

In summary, the understanding of the term maintainability is in conformance to that provided by IEEE [13]; expert judgment (ordinal scale metric) was the most commonly used means to measure/assess maintainability; maintainability was not commonly discussed in relation to any particular maintenance

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

type; and the only evidence of conformance of maintainability to a quality model favored the ISO/IEC standard 12207.

Table 8. Summary of Maintainability Metrics

Study ID	Measures/Metrics for Maintainability
S1	*Subjective assessment* (ordinal scale metric) by using the U.S Air Force Operational Test and Evaluation Center's AFOTEC software maintainability evaluation instrument, which provides a rating as well as categorizes maintainability as low, medium or high.
S2,S3	Same as S1 but call it HPMAS Maintainability Index
S4	Same as S1, S2 and S3
S5, S7	Expert opinion using an ordinal scale
S9,S15	Maintainability Index (MI)
S10	1. Total effort in person-minutes to comprehend, modify, and test the artifacts related to the system 2. Volume of changes made to software artifacts. Change volume was measured as: a. The number of pages changed in a document b. The number of modified executable LOC c. The number of test cases constructed d. The LOC written to construct test scripts e. The number of files needed to be compiled f. The number of files needed to be deployed
S12 S14	CHANGE metric: count of LOC changed during a 3-year maintenance period
S13	$G(u) = 1-\exp(-v^{\wedge}u)$, according to $M_I/M/\infty$ framework

4. Discussion

The results of the SR suggest that there are no obvious choices for maintainability prediction models. Of the models proposed in the literature, few are supported with accuracy measures, few use any form of cross-validation, and few have evidence for external validity. The kinds of models proposed are also quite limited. Out of the 15 selected studies, 12 proposed models for software maintainability prediction and of these only six used accuracy measures. Of the 12 studies that have proposed models for software maintainability prediction, 8 used algorithmic and more specifically regression analysis techniques and only four mentioned the use of cross-validation techniques for measuring prediction accuracy. The models proposed in S1 were specific to HP's data set and those proposed in S12 and S14 were based on Li & Henry's datasets [18] making their external validity limited. It was also observed from the results of the SR that out of a total of 8 studies providing software maintainability prediction models based on algorithmic techniques, only 3 compared the algorithmic techniques with other techniques e.g., S2 compares regression analysis based prediction models with HPMAS, S12 with Bayesian Network (BN) based prediction model and S14 with Artificial Neural Network (ANN) based model, respectively. All these comparisons, except for S12, suggested the use of regression analysis-based models. S12 reported BN-based model to have better prediction accuracy than regression analysis based model for one out of two data sets. To summarize, 2/12 studies using prediction models compared algorithmic techniques with other techniques – BN and ANN – where the dataset used [18] comprised only two data points, thus hampering the result's external validity.

Therefore, further work is needed to (1) develop models for software maintainability prediction having a certain level of external validity and (2) to compare different prediction techniques such as expert opinion, different algorithmic and artificial intelligence techniques [23] for software maintainability prediction.

Our results also highlight that the prediction techniques employed by the selected studies may be equally applied to any sub-characteristic of maintainability or any type of maintenance. One possible explanation for this is that maintainability and its sub-characteristics (as defined by ISO/IEC [14]) are closely related to complexity [2], [12] and software size [31], [34], and are also used to compute Maintainability Index [31]. Therefore, models using complexity and size metrics as predictors may be likely to be equally applicable to any maintainability sub-characteristic. A counter argument is that the use of these models has not been validated against different maintainability sub-characteristics and maintenance types. Whichever way, it is clear that there is a need to further explore whether the maintainability sub-characteristics and maintenance types can be treated uniformly in relation to maintainability prediction.

Our results also suggest that metrics related to application size, complexity and coupling were the most commonly used maintainability predictors. The most commonly used maintainability metric was an ordinal scale metric based on expert opinion which, in addition to suggesting the importance of intuitive evaluation of software quality attributes, highlights the fact that metrics using more robust scale types are not commonly employed for quantifying maintainability. By covering as many online databases and broadening the search strings, we made an attempt to ensure broad coverage, yet the number of maintainability metrics and predictors was not large. Therefore, there is scope for the proposal of other metrics that may be more applicable for maintainability prediction.

Several issues worth noting arose during the SR process. As already mentioned, many studies did not clearly differentiate between maintenance and maintainability and so the search query was modified with this in mind. We also had to tailor our query for each database. Because we made conservative changes

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

to our query, many irrelevant results were reported. Despite the quality checklists commonly used for SRs being a good mechanism for determining the studies' quality, they sometimes did not, in our view, depict the true quality of the study. For example, there were questions that sometimes could not be directly answered which resulted in giving average scores to such questions.

While extracting data, we refrained from making assumptions and reported only the results that were clearly stated by the studies. This approach, on one hand, helped to avoid bias in understanding the studies' findings and making assumptions that were not made by the author(s); while on the other hand, it prevented us from deriving certain facts, indirectly mentioned in the primary studies, which would be worth recording.

5. Conclusions

This paper presents the results of an SR on software maintainability prediction and metrics. A total of 710 results were returned, of which 18 were initially selected and three discarded later on due to their low quality. As a result, 15 studies were selected for the SR. Data were extracted from these studies and then synthesized against the defined research questions.

Our results showed that maintainability, as understood in the context of software systems, was in conformance to the definition provided by IEEE [13]; the commonly used maintainability prediction models were based on algorithmic techniques and there was no distinction of which models should be applied to which maintainability sub-characteristic or maintenance type; the most commonly used predictors were those based on size, complexity and coupling, and gathered at source code level; the use of prediction techniques and models, accuracy measures and cross-validation methods was found scarce for validating maintainability prediction models; and the most commonly used maintainability metric employed an ordinal scale and was based on expert judgment.

For future work, we suggest the creation of more robust and reliable software maintainability prediction models. We also suggest further exploration of the treatment of maintainability prediction techniques/models and their applicability on different maintainability sub-characteristics and maintenance types. The effective use of accuracy measures for the models has not been observed providing a possible threat to the successful application of these models. There is, therefore, a need to further validate maintainability prediction models and suggest models that are not completely domain specific. In addition, the use of other prediction techniques such as algorithmic and non-algorithmic techniques and their applicability for maintainability prediction may be investigated.

6. References

[1] Y. Ahn, J. Suh, S. Kim, and H. Kim, 2003, "The Software Maintenance Project Effort Estimation Model Based on Function Points", *J Softw Maint Evol, 15, 2,* 2003, pp. 71 – 85.

[2] R. K. Bandi, V. K. Vaishnavi, and D. E. Turk, "Predicting Maintenance Performance Using Object-Oriented Design Complexity Metrics", *IEEE T Software Eng, 29, 1*, Jan. 2003, pp. 77 – 87.

[3] P. Bhatt, K. Williams, G. Shroff, and A. K. Misra, "Influencing Factors in Outsources Software Maintenance", *ACM SIGSOFT Software Engineering Notes, 31, 3*, May 2006, pp. 1 – 6.

[4] D. Coleman, D. Ash, B. Lowther, and P. Oman, "Using Metrics to Evaluate Software System Maintainability". *IEEE Computer*, Aug. 1994, pp. 44 – 49.

[5] D. Coleman, B. Lowther, and P. Oman, "The Application of Software Maintainability Models in Industrial Software Systems. *J Syst Software, 29 (1995)*, pp. 3 – 16.

[6] I.K. Crombie, *The Pocket Guide to Appraisal*, BMJ Books, 1996.

[7] G.A. Di Lucca, A.R. Fasolino, P. Tramontana, and C.A. Visaggio, "Towards the Definition of a Maintainability Model for Web Applications, In Proceedings of the CSMR'04, 2004, pp. 279.

[8] E.H. Ferneley, "Design Metrics as an Aid to Software Maintenance: An Empirical Study", *J Softw Maint Evol - R, 11 (1999)*, pp. 55 – 72.

[9] A. Fink, *Conducting Research Literature Reviews. From the Internet to Paper,* Sage Publication Inc., 2005.

[10] M. Genero, J. Olivas, M. Piattini, and F. Romero, "Using metrics to predict OO information systems maintainability", *Lecture Notes in Computer Science, 2068 (2001)*, pp. 388 – 401.

[11] J.C. Granja-Alvarez and M.J. Barranco-García, "A Method for Estimating Maintenance Cost in a Software Project: A Case Study". *J Softw Maint Evol, 9, 3*, 1997, pp. 161 – 175.

[12] T. Hirota, M. Tohki, C.M. Overstreet, M. Masaaki, and R. Cherinka, "An Approach to Predict Software Maintenance Cost Based on Ripple Complexity", *In Proceedings of APSEC (Dec. 7-9, 1994)*, pp. 439 – 444.

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

[13] *IEEE Std. 610.12-1990. Standard Glossary of Software Engineering Terminology*, IEEE Computer Society Press, Los Alamitos, CA, 1993.

[14] International Organization for Standarization. 1998. ISO/IEC 9126-1.2 *Information Technology - Software product quality - Part 1: Quality model*.

[15] M. Jorgensen, and M. Shepperd, " A Systematic Review of Software Development Cost Estimation Studies" *IEEE T Software Eng, 33, 1 (Jan. 2007)*, pp. 33 – 53.

[16] B. Kitchenham "Procedures for Performing Systematic Reviews", *Technical Report TR/SE0401* (Keele University) and *Technical Report 0400011T.1* (National ICT Australia), 2004.

[17] B. Kitchenham, "Guidelines for Performing Systematic Literature Review in Software Engineering", *EBSE Technical Report, 2.3,* Keele University, 2007.

[18] W. Li, S. Henry, "Object-Oriented Metrics that Predict Maintainability", *J Syst Software, 23 (1993)*, pp. 111–122.

[19] J.S. Lim, S.R. Jeong, and S.R. Schach, "An empirical Investigation of the Impact of the Object-Oriented Paradigm on the Maintainability of Real-World Mission-Critical Software, *J Syst Software, 77 (2005)*, pp. 131 – 138.

[20] E. Mendes, N. Mosley, and S. Counsell, "The Application of Case-Based Reasoning to Early Web Project Cost Estimation", *In Proceedings of COMPSAC 2002*, pp. 393 – 398.

[21] S.C. Misra, "Modelling Design/Coding Factors that Drive Maintainability of Software Systems", *Software Qual J, 13 (2005)*, pp. 297 – 320.

[22] E. Mendes, "The Use of Bayesian Networks for Web Effort Estimation: Further Investigation", *In Proceedings of ICWE 2008*, pp. 203 – 216.

[23] E. Mendes and N. Mosley, *Web Engineering*. Springer-Verlag Berlin Heidelberg New York. ISBN-10 3-540-28196-7, ISBN-13 978-3-540-28196-2, 2006.

[24] P. Oman and J. Hagemeister, "Construction and Testing of Polynomials Predicting Software Maintainability", *J Syst Software, 24 (1994)*, pp. 251 – 266.

[25] M. Petticrew and H. Roberts, H. *Systematic Reviews in the Social Sciences: A Practical Guide*. Blackwell Publishing, 2006.

[26] N.J. Pizzi and W. Pedrycz, "Predicting Qualitative Assessments using Fuzzy Aggregation" *In Proceedings of NAFIPS 2006*, pp. 267 – 272.

[27] S.L. Schneberger, "Distributed computing environments: effects on software maintenance difficulty", *J Syst Software, 37 (1997)*, pp. 101 – 116.

[28] K. Shibata, K. Rinsaka, T. Dohi, and H. Okamura, "Quantifying Software Maintainability Based on a Fault-Detection/Correction Model", *In Proceedings of the PRDC 2007*, pp. 35 – 42.

[29] C. Van Koten and A.R. Gray, "An Application of Bayesian Network for Predicting Object-Oriented Software Maintainability", *Inform Software Tech, 48, 1 (Jan. 2006)*, pp. 59 – 67.

[30] J. Webster and R.T. Watson, "Analyzing the Past to Prepare for the Future: Writing a Literature Review", *MIS Quarterly, 26, 2, xiii-xxiii*, 2002.

[31] K.D. Welker and P.W. Oman, "Development and Application of an Automated Source Code Maintainability Index", *J Softw Maint Evol - R, 9 (1997)*, pp. 127 – 159.

[32] M. Riaz, Systematic Review of Software Maintainability Prediction and Metrics. http://www.cs.auckland.ac.nz/~mehwish/SLR/, 2009.

[33] M.V. Zelkowitz, "Perspectives in Software Engineering", *ACM Comput Surv (CSUR), 10 (Jun. 1978)*, pp. 197 – 216.

[34] Y. Zhou and H. Leung, "Predicting Object-Oriented Software Maintainability using Multivariate Adaptive Regression Splines, *J Syst Software, 80(2007)*, pp. 1349 – 1361.

[35] Y. Zhou and B. Xu, "Predicting the Maintainability of Open Source Software using Design Metrics", *Wuhan University Journal of Natural Sciences, 13, 1*, pp. 14 – 21, 2008.

[36] K.M. Furulund and K. Moløkken-Østvold, "Increasing Software Effort Estimation Accuracy - Using Experience Data, Estimation Models and Checklists", *In Proceedings of QSIC 2007,* pp. 342 – 347.

978-1-4244-4841-8/09/$25.00 ©2009 IEEE