An application of Bayesian network for predicting object-oriented software maintainability

C. van Koten [1] and A.R. Gray

Department of Information Science,University of Otago, P.O.Box 56, Dunedin, New Zealand

Abstract

As the number of object-oriented software systems increases, it becomes more important for organizations to maintain those systems effectively. However, currently only a small number of maintainability prediction models are available for object-oriented systems. This paper presents a Bayesian network maintainability prediction model for an object-oriented software system. The model is constructed using object-oriented metric data in Li and Henry's datasets, which were collected from two different object-oriented systems. Prediction accuracy of the model is evaluated and compared with commonly used regression-based models. The results suggest that the Bayesian network model can predict maintainability more accurately than the regression-based models for one system, and almost as accurately as the best regression-based model for the other system.

Key words: Object-oriented systems, Maintainability, Bayesian network, Regression tree, Regression

1 Introduction

It is arguable that many object-oriented (OO) software systems are currently in use. It is also arguable that the growing popularity of OO programming languages, such as Java, as well as the increasing number of software development tools supporting the Unified Modelling Language (UML), encourages more OO systems to be developed at present and in the future. Hence it is important

[1] Corresponding author. Tel.: +64-3-479-8142; fax: +64-3-479-8311.
E-mail address: ckoten@infoscience.otago.ac.nz

that those systems are maintained effectively and efficiently. A software maintainability prediction model enables organizations to predict maintainability of a software system and assists them with managing maintenance resource. In addition, if an accurate maintainability prediction model is available for a software system, a defensive design can be adopted. This would minimize, or at least reduce future maintenance effort of the system. Maintainability of a software system can be measured in different ways. In this paper, maintainability is measured as the number of changes made to the code during a maintenance period. Alternatively, maintainability may be measured as effort to make those changes. When maintainability is measured as effort, the predictive model is called a maintenance effort prediction model. It is unfortunate that the number of software maintainability prediction models including maintenance effort prediction models, is currently very small in the literature.

Programming an OO software system is different from programming a non-OO system due to the concepts that are specific to the OO paradigm, for example, *objects*, *inheritance* and *encapsulation*. This difference limits the applicability of well-known non-OO software effort prediction models, such as COCOMO [3], to OO software effort prediction, as well as non-OO software metrics, such as Function Points [1], to measuring the characteristics of OO software systems [23]. Hence a number of new software metrics were proposed specifically for OO systems. Some of those OO metrics were used to predict maintainability of OO systems. Examples of the OO metrics are Chidamber and Kemerer (C&K) metrics and Li and Henry (L&H) metrics [10,25]. It was shown that the L&H metrics had a correlation with the number of changes made to the code of the OO software system [25]. It was also shown that multiple linear regression models consisting of the C&K, L&H and other OO metrics were able to predict software maintenance effort for some OO systems [17].

This paper constructs an OO software maintainability prediction model using a technique known as Bayesian network [14,20,22]. This technique allows a user to construct a predictive model based on Bayesian probability theory [12]. An application of Bayesian network to Software Engineering is currently limited to a small number of studies of development effort prediction [2,11,31,34] and defect prediction [15,28]. However, Bayesian network can also be a promising new technique for OO software maintainability prediction. This is due to the ability to explicitly represent uncertainty using probabilities, the ability to incorporate existing human expert's knowledge into empirical data, and the ability to update the model when new information becomes available. Hence this paper investigates a research problem of what prediction accuracy a Bayesian network OO software maintainability prediction model can achieve. The term *prediction accuracy* in this paper means how well a predictive model constructed using known data can predict the outcomes of unknown data. The Bayesian network model's prediction accuracy is evaluated using some accu-

racy measures, which are commonly found in the software effort prediction literature [16,24]. Those measures are absolute residuals, the magnitude of relative error (MRE) and pred measures. Then, the Bayesian network model's prediction accuracy is compared with regression-based models, namely, a regression tree [4] model and two different types of multiple linear regression models.

The structure of the reminder of this paper is as follows. Section 2 describes the OO software datasets and the sampling method used. Section 3 describes the Bayesian network OO software maintainability prediction model. This is followed by Section 4, which describes the regression tree model and the multiple linear regression models. Section 5 describes the prediction accuracy measures used. Section 6 evaluates the Bayesian network model's prediction accuracy using those accuracy measures and compares it with the regression tree model and multiple linear regression models. Finally Section 7 presents conclusions and discussions about a direction of future studies.

2 OO software datasets

2.1 Characteristics of datasets

This paper uses OO software datasets published by Li and Henry [25]. The datasets consist of five C&K metrics: DIT, NOC, RFC, LCOM and WMC, and four L&H metrics: MPC, DAC, NOM and SIZE2, as well as SIZE1, which is a traditional lines of code size metric. Those metric data were collected from a total of 110 classes in two OO software systems: User Interface Management System (UIMS) and Quality Evaluation System (QUES). The code was written in $Classical - Ada^{TM}$. The UIMS and QUES datasets contain 39 classes and 71 classes, respectively. Maintainability was measured in CHANGE metric by counting the number of lines in the code, which were changed during a three-year maintenance period. Neither UIMS nor QUES datasets contain actual maintenance effort data. The description of each metric is given in Table 1.

The descriptive statistics of the UIMS and QUES datasets are shown in Table 2.

The Pearson's correlation coefficients between CHANGE and each of the OO metrics are shown in Table 3.

Table 3 shows that there is a significant correlation between CHANGE and the OO metrics. However, Table 3 also shows that the correlations in the UIMS

dataset are different from the correlations in the QUES dataset. In addition, Tables 2 shows that the characteristics of the UIMS dataset are different from the QUES dataset. Thus, this paper regards the UIMS and QUES datasets as being heterogeneous and constructs a separate maintainability prediction model for each dataset.

2.2 Sampling method

Approximately a two-third of the cases in each dataset is chosen by random sampling without replacement using a function provided in a statistical software package, SPSS 11.0. This subset forms a learning subset, which is used to construct a maintainability prediction model. The remaining one-third of the cases forms a test subset, which is used to evaluate the model's prediction accuracy for unknown data. This results in the UIMS having 26 cases in a learning subset and the QUES 48 cases. Consequently, the UIMS has 13 cases in a test subset and the QUES 23 cases. Although there are many different ways to split a given dataset, this paper has chosen the described split for both datasets. The decision was made in order to have a small number of learning cases and at the same time, to maintain the comparability of prediction accuracy of the two datasets by using the same proportion of the cases for learning and testing. This is because the number of known cases is usually small for predicting maintainability of an OO software system and thus, using a small number of learning cases is considered to be more realistic for the model.

The above sampling is repeated 10 times from each dataset in order to create 10 different subsets for learning and testing for each dataset. Those subsets are used to perform a 10-cross validation when prediction accuracy of the models are evaluated.

3 Bayesian network model

3.1 Bayesian network

A Bayesian network (also known as Bayes net, causal probabilistic network, Bayesian belief network, or simply belief network) is a directed acyclic graph (DAG) whose nodes represent events in a domain [22]. These events are connected with directed links, which represent an association or a causal relationship between them. When a link represents an association, the direction is defined according to the order of time in which the events happen, that is, the link starts from the preceding event. When a link represents a causal

relationship, the link starts from the causal event. Figure 1 shows an example Bayesian network consisting of three events, X_1, X_2, and Y. In Figure 1, an event is shown as an ellipse and a directed link is shown as an arrow. This example Bayesian network shows that X_1 and X_2 have an association or a causal relationship with Y, that is, the outcomes of the events X_1 and X_2 have an effect on the outcome of the event Y.

In a Bayesian network, a relationship between events is defined as a *conditional probability*, $P(Y \mid X)$, which is the probability of the event Y conditional on a given outcome of event X. The *conditional probability* is calculated using Bayes' Theorem [22]:

$$P(Y \mid X) = \frac{P(X \mid Y)P(Y)}{P(X)} \tag{1}$$

where $P(X \mid Y)$ is the *conditional probability* of the event X given the event Y, and P(X) and P(Y) are the probabilities of events X and Y respectively. From this point of view, Bayesian networks can be considered as a network of events connected by the probabilistic dependencies between them. The probabilistic dependency is maintained by the conditional probability table (CPT), which is attached to the corresponding event. The CPT shows all possible outcomes of the event and the conditional probabilities corresponding to each outcome given an outcome of an associated or a causal event. If an event has no associated or causal event, the event is given an unconditional (also called a marginal) probability distribution instead of a CPT. Once either a CPT or an unconditional probability distribution is assigned to all events, the network can calculate the *joint probability* distribution P(θ,x) over the network, which is defined as

$$P(\theta, x) = P(x \mid \theta)P(\theta) \tag{2}$$

In Equation 2, θ denotes a vector whose components are parameters that describe the probability distributions over the network, and x denotes a piece of data. In Equation 2, P(θ) is called the *prior* probability distribution of the network and P(x $\mid$ θ) is called the *likelihood* of the data, given the *prior* probability distribution.

On the other hand, Bayes' Theorem 1 shows that

$$P(\theta \mid x) = \frac{P(x \mid \theta)P(\theta)}{P(x)} \tag{3}$$

In Equation 3, P(θ $\mid$ x) is called the *posterior* probability distribution of the network. Considering that P(x) is a constant for a given x, Equations 2 and

Name	Description
DIT	Depth of the inheritance tree (= inheritance level number of the class, 0 for the root class)
NOC	Number of children (= number of direct sub-classes that the class has)
MPC	Message-passing coupling (= number of send statements defined in the class)
RFC	Response for a class (= total of the number of local methods and the number of methods called by local methods in the class)
LCOM	Lack of cohesion of methods (= number of disjoint sets of local methods, i.e. number of sets of local methods that do not interact with each other, in the class)
DAC	Data abstraction coupling (= number of abstract data types defined in the class)
WMC	Weighted method per class (= sum of McCabe's cyclomatic complexity of all local methods in the class)
NOM	Number of methods (= number of local methods in the class)
SIZE1	Lines of cod (= number of semicolons in the class)
SIZE2	Number of properties (= total of the number of attributes and the number of local methods in the class)
Change	Number of lines changed in the class (insertion and deletion are independently counted as 1, change of the contents is counted as 2)

Table 1
Software Metrics [25]

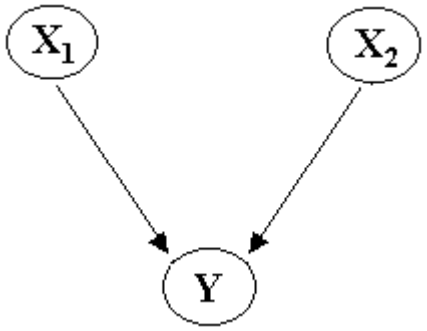

Fig. 1. An Example Bayesian Network

	Mean	Median	Std Dev	Min	Max	Skewness	Kurtosis
DIT	2.15	2	0.90	0	4	- 0.54	0.09
NOC	0.95	0	2.01	0	8	2.24	4.28
MPC	4.33	3	3.41	1	12	0.73	- 0.70
RFC	23.21	17	20.19	2	101	2.00	4.94
LCOM	7.49	6	6.11	1	31	2.49	6.86
DAC	2.41	1	3.40	0	21	3.33	12.87
WMC	11.38	5	15.90	0	69	2.03	3.98
NOM	11.38	7	10.21	1	40	1.67	1.94
SIZE1	106.44	74	114.65	4	439	1.71	2.04
SIZE2	13.97	9	13.47	1	61	1.89	3.44
Change	46.82	18	71.89	2	289	2.29	4.35

(1) UIMS dataset

	Mean	Median	Std Dev	Min	Max	Skewness	Kurtosis
DIT	1.92	2	0.53	0	4	- 0.10	5.46
NOC	0	0	NA	0	0	NA	NA
MPC	17.75	17	8.33	2	42	0.88	1.17
RFC	54.44	40	32.62	17	156	1.62	1.96
LCOM	9.18	5	7.31	3	33	1.35	1.10
DAC	3.44	2	3.91	0	25	2.99	12.82
WMC	14.96	9	17.06	1	83	1.77	3.33
NOM	13.41	6	12.00	4	57	1.39	1.40
SIZE1	275.58	211	171.60	115	1009	2.11	5.23
SIZE2	18.03	10	15.21	4	82	1.71	3.42
Change	64.23	52	43.13	6	217	1.36	2.17

(2) QUES dataset

Table 2
Descriptive Statistics

3 show that

$$posterior\ distribution \propto likelihood \times prior\ distribution \tag{4}$$

as well as

$$posterior\ distribution \propto joint\ probability\ distribution \tag{5}$$

The above relationship 4 provides the principle of Bayesian updating, where the *prior* probability distribution is updated using the likelihood of observed data. That is, a Bayesian network can be recalibrated every time new information becomes available. The effect of new information propagates both for-

	Pearson's Correlation Coefficient	
	UIMS dataset	**QUES dataset**
DIT	- 0.433**	- 0.090
NOC	0.559**	NA
MPC	0.454**	0.461**
RFC	0.643**	0.388**
LCOM	0.568**	0.050
DAC	0.629**	0.083
WMC	0.646**	0.425**
NOM	0.635**	0.142
SIZE1	0.626**	0.635**
SIZE2	0.666**	0.149

(** indicates a significant correlation at the 0.01 level)

Table 3
Correlations between CHANGE and OO Metrics

ward and backward on the network and it enables the network to update the *joint probability* distribution. In addition, the *prior* probability distribution of a Bayesian network can be specified *subjectively*, that is, specified initially based on only human expert's knowledge. Then, the network is updated using empirical data. This enables expert's knowledge to be incorporated into empirical data. Alternatively, when no expert's knowledge is available for specifying the *prior* probability distribution, the network can perform *batch learning*. In *batch learning* the structure of the network and all the CPTs and unconditional probability distributions are learned from data. There are a number of different algorithms that enable a Bayesian network to perform *batch learning* [9].

The relationship 5 is the principle of Bayesian inference, where computation of the *joint probability* distribution is performed in order to summarize the *posterior* probability distribution. On a Bayesian network, the computation of the *joint probability* distribution is simplified using the chain rule. The chain rule enables the *joint probability* distribution to be calculated as the product of CPTs and unconditional probability distributions over the network. When a Bayesian network is a predictive model, the resulting *posterior* probability distribution provides the probability distribution of the predicted variable, that is, a list of all possible outcomes of the variable and the associated probabilities. In other words, a Bayesian network model provides an interval estimate of the predicted variable with the associated probabilities. When a point estimate is preferable, the outcome with the highest probability value would be chosen. More detailed information about Bayesian network can be found in the literature [14,20,22,12].

3.2 Model construction

There are a number of software tools that assist with constructing a Bayesian network. This paper uses a tool called Bayda, a software package developed in the University of Helsinki. Bayda allows users to construct a special type of Bayesian networks called Naïve-Bayes classifier. A Naïve-Bayes classifier is a Bayesian network, in which a single node representing a classification variable is connected to all other nodes that represent predictor variables. This special Bayesian network assumes no expert's knowledge about the *prior* probability distribution, but performs *batch learning* to learn it from data. This paper chooses a Naïve-Bayes classifier instead of a general Bayesian network. This is because for the Bayesian network maintainability prediction model, a single node CHANGE is connected to the 10 OO metric predictor variables in Table 1, but no sufficient expert's knowledge is available to quantitatively specify the *prior* probability distribution of this network. Thus, the network needs to perform *batch learning* using a learning subset.

After the *batch learning*, the network predicts the *posterior* probability distribution of CHANGE for each case in the corresponding test subset, by computing the *joint probability* distribution. As was mentioned previously, a Bayesian network model's prediction output is an interval estimate. However, this paper uses a point estimate. This is because the interval estimate is difficult to compare with the other models. The point estimate is chosen from the interval estimate as the value that has the highest probability. When more than one value have the same highest probability, the mean of those values is chosen.

4 Regression-based models

4.1 Regression tree model

Regression tree is a tree-structured regression technique, which recursively partitions the data space of a given dataset with a number of regression surfaces, on each of which a constant estimate of the response variable is given according to a chosen regression method [4]. Figure 2 shows an example regression tree. In Figure 2, four sequential binary splits partition all cases in the dataset into five terminal nodes $T_1, \ldots, T_5$, which are shown as five squares. Each terminal node consists of only the cases that satisfy the corresponding sequential splitting rules to reach it. An ellipse represents an intermediate node, where cases are split according to the splitting rule. A regression tree gives an estimate of the response variable at each intermediate node and each terminal node. The estimate is calculated from the cases in the node, but the type of

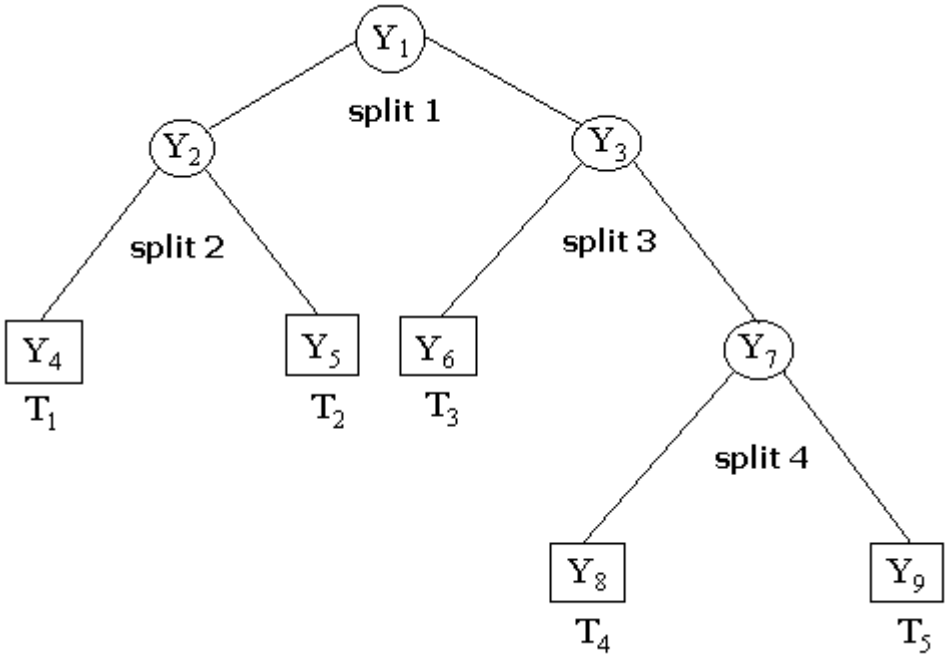

Fig. 2. An Example Regression Tree

the estimate depends on the splitting rule. For example, the splitting rule used is to minimize the mean within node sum of squares, mean of the cases in the node is used as the estimate for the node. In Figure 2, the estimates are shown as $Y_1, \ldots, Y_9$.

A regression tree also outputs the decision rule at each intermediate node. The decision rule shows how cases are split at the node, and is described as the condition that one or more predictor variables must satisfy. This makes a regression tree similar to a decision tree and makes the structure easy to understand. Many researchers have found regression tree to be a useful technique for software effort prediction, especially due to the good explanatory ability [5–8,19,21,29,30,32].

Regression tree is different from Bayesian network. As a summary, regression tree is a regression-based classification technique, which outputs a tree structure similar to a decision tree. A regression tree model is constructed to classify cases in a learning subset based on a specified splitting rule, which is similar to a rule in regression. For example, a regression tree classifies cases by minimizing the mean within node sum of squares. This splitting rule is similar to ordinary least square in regression. A regression tree model does not calculate the probability of the response variable in prediction. It provides a point estimate of the value calculated from the cases in the terminal node, into which a test case is classified.

A regression tree maintainability prediction model is constructed using the same learning subset as the Bayesian network model. The model uses all 10 OO metrics as predictor variables and predicts the value of the response variable CHANGE. The tool used is CART 5.0. The default settings provided by the tool are used for the regression tree construction. The splitting rule chosen is to minimize the mean within node sum of squares. After learning, the model predicts the value of CHANGE for a test case by providing a point estimate. The estimate is the mean of all the cases in the terminal node, into which the test case is classified according to the decision rules of the tree.

4.2 Multiple linear regression models

Two types of multiple linear regression models are constructed from the same learning subset as the previous models. The software package used is SPSS 11.0. In multiple linear regression, a variable selection procedure enables only important predictor variables to be included in the model. Two commonly used variable selection procedures are backward elimination and stepwise selection. Backward elimination eliminates predictor variables whose contribution is less significant according to a specified criterion, while stepwise selection enters predictor variables whose contribution is more significant. In addition, stepwise selection re-assesses the contribution of the variables that already entered, every time a new variable is entered. The re-assessment is performed in the same way as backward elimination using a specified elimination criterion. Backward elimination and stepwise selection often produce different models. Thus, this paper constructs one multiple linear regression model using backward elimination and the other using stepwise selection. The elimination criterion used in backward elimination is the p-value of the F statistic being larger than or equal to 0.1. The entering criterion used in stepwise selection is the p-value of the F statistic being smaller than or equal to 0.05. The eliminating criterion used in stepwise selection is the same as the one used in backward elimination.

In multiple linear regression, multicollinearity between the predictor variables often causes a problem of an unstable model and a large standard error in prediction. Thus, multicollinearity is monitored by the variance inflation factor (VIF) statistic in this paper. A VIF value larger than 10 usually indicates the presence of multicollinearity. When multicollinearity is detected by the VIF value, the offending predictor variable is removed and the model is rebuilt without the offending variable.

The backward elimination model constructed is different from the stepwise selection model for eight out of the 10 learning subsets in the UIMS dataset, and six out of the 10 learning subsets in the QUES dataset. All of those mul-

tiple linear regression models have strongly significant parameter coefficients, as indicated by the associated p-value that is smaller than 0.01. The residual plots of the models show no sign of violation of the assumptions in linear regression.

5 Prediction accuracy measures

This paper evaluates and compares the OO software maintainability prediction models quantitatively, using the following prediction accuracy measures: absolute residual (Ab.Res.), the magnitude of relative error (MRE) and pred measures.

The Ab.Res. is the absolute value of residual given by:

$$Ab.Res. = \mid actual\ value - predicted\ value \mid \tag{6}$$

In this paper, the sum of the absolute residuals (Sum Ab.Res.), the median of the absolute residuals (Med.Ab.Res.) and the standard deviation of the absolute residuals (SD Ab.Res.) are used. The Sum Ab.Res. measures the total residuals over the dataset. The Med.Ab.Res. measures the central tendency of the residual distribution. The Med.Ab.Res. is chosen to be a measure of the central tendency because the residual distribution is usually skewed in software datasets. The SD Ab.Res. measures the dispersion of the residual distribution.

MRE is a normalized measure of the discrepancy between actual values and predicted values, given by [24]:

$$MRE = \frac{\mid actual\ value - predicted\ value \mid}{actual\ value} \tag{7}$$

In this paper, the maximum value of MRE (Max.MRE) is used. The Max.MRE measures the maximum relative discrepancy, which is equivalent to the maximum error relative to the actual effort in the prediction. The mean of MRE, the mean magnitude of relative error (MMRE):

$$MMRE = \frac{1}{n}\sum_{i=1}^{i=n} MRE_i \tag{8}$$

is also used. MMRE measures the average relative discrepancy, which is equivalent to the average error relative to the actual effort in the prediction. Some-

times MMRE is expressed in %. However, this paper follows the difinition given in Equation 8 and does not express MMRE in %.

Pred is a measure of what proportion of the predicted values have MRE less than or equal to a specified value, given by [16]:

$$Pred(q) = \frac{k}{n} \tag{9}$$

where q is the specified value, k is the number of cases whose MRE is less than or equal to q, and n is the total number of cases in the dataset. In this paper, pred(0.25) and pred(0.30) are used because those two pred measures are commonly used in the software effort prediction literature.

In order for an effort prediction model to be considered accurate, $MMRE \leq 0.25$ [13] and/or either $pred(0.25) \geq 0.75$ [13] or $pred(0.30) \geq 0.70$ [27] is suggested in the literature. On the other hand, there is a concern about MRE because MRE is biased [33] and not always reliable as a prediction accuracy measure [18]. However, MRE has been the de facto standard in the software effort prediction literature and no alternative standard exists at present. Thus, this paper still uses the above criteria. However, in addition to that, this paper uses the absolute residual measures because it has shown that the absolute residual measures, in particular the SD Ab.Res., are a better measure than MRE for model comparison [18].

6 Model evaluation and comparison

6.1 Results from UIMS dataset

Table 4 shows the values of the prediction accuracy measures achieved by each of the maintainability prediction models for the UIMS dataset. The values in this table are the mean of the values obtained from the 10 different test subsets, which were created using the sampling method described in Section 2.

Table 4 shows that the Bayesian network model has achieved the MMRE value of 0.972, the pred(0.25) value of 0.446 and the pred(0.30) value of 0.469. Although these values do not satisfy the criteria of an accurate prediction model, which was mentioned in Section 5, those values are the best among all the four models studied. The Wilcoxon signed-rank tests of the MRE values have also confirmed strong evidence that the Bayesian network model's MMRE value is significantly lower and thus, better than those of the other models. In addition, Table 4 shows that the prediction accuracy of the Bayesian network model is

Model	Max. MRE	MMRE	Pred (0.25)	Pred (0.30)	Sum Ab.Res.	Med. Ab.Res.	SD Ab.Res.
Bayesian network	7.039	0.972	0.446	0.469	362.300	10.550	46.652
Regression tree	9.056	1.538	0.200	0.208	532.191	10.988	63.472
Backward elimination	11.890	2.586	0.215	0.223	538.702	20.867	53.298
Stepwise selection	12.631	2.473	0.177	0.215	500.762	15.749	54.114

Table 4
Prediction accuracy for the UIMS dataset

Model	Max. MRE	MMRE	Pred (0.25)	Pred (0.30)	Sum Ab.Res.	Med. Ab.Res.	SD Ab.Res.
Bayesian network	1.592	0.452	0.391	0.430	686.610	17.560	31.506
Regression tree	2.104	0.493	0.352	0.383	615.543	19.809	25.400
Backward elimination	1.418	0.403	0.396	0.461	507.984	17.396	19.696
Stepwise selection	1.471	0.392	0.422	0.500	498.675	16.726	20.267

Table 5
Prediction accuracy for the QUES dataset

also the best in the absolute residual measures. The Wilcoxon signed-rank tests of the absolute residuals have again confirmed strong evidence that the differences of the Bayesian network model from the other models are significant. Thus, it is concluded that the Bayesian network model is able to predict maintainability of the UIMS dataset better than the regression-based models studied.

6.2 Results from QUES dataset

Table 5 shows the values of the prediction accuracy measures achieved by each of the maintainability prediction models for the QUES dataset. The values in this table are the mean of the values obtained from the 10 different test subsets.

Table 5 shows that the Bayesian network model has achieved the MMRE value of 0.452, the pred(0.25) value of 0.391 and the pred(0.30) value of 0.430. Again these values do not satisfy the criteria of an accurate prediction model. In comparison with the UIMS dataset, the MMRE value of 0.452 is better, while the pred(0.25) and pred(0.30) values are poorer. This suggests that the performance of the Bayesian network models may vary depending on the characteristics of dataset and/or depending on what prediction accuracy measure is used. Unfortunately the authors are not able to make any more comment on this finding at present as it requires further investigation.

In comparison with the two multiple linear regression models, the prediction accuracy of the Bayesian network model is lower in all the measures. However, the Wilcoxon signed-rank test of the MRE values has shown no evidence of the difference of the Bayesian network model from the backward elimination model as indicated by the p-value of 0.078. On the other hand, the same test on the stepwise selection model has shown weak evidence of the difference as indicated by the p-value of 0.029. The tests of the absolute residuals have shown strong evidence that the difference of the Bayesian network model from the two multiple linear regression models is significant. Thus, it is concluded that the Bayesian network model's MMRE is not much different from those of the multiple linear regression models, although the Bayesian network model's absolute residual statistics are different.

In comparison with the regression tree model, the prediction accuracy of the Bayesian network model is better in most measures, with an exception of the two absolute residual measures: the sum and the standard deviation. However, the Wilcoxon signed-rank tests have shown no evidence of the differences as indicated by the p-value of 0.752 for MRE, and 0.173 for absolute residuals. Thus, it is concluded that there is no difference in prediction accuracy between the Bayesian network model and the regression tree model.

Considering the above findings, the best model for the QUES dataset seems one of the two multiple linear regression models. However, it is difficult to determine exactly which model is the best because the measures in Table 5 present an inconsistent result. That is, the backward elimination model is better when the Max.MRE and the SD Ab.Res. values are taken account of, while the stepwise selection model is better when the MMRE, pred(0.25), pred(0.30), Sum Ab.Res. and Med.Ab.Res. values are taken account of. In addition, the Wilcoxon signed-rank tests of the MRE values and the absolute residuals have shown no evidence of the differences between those two models, as indicated by the p-value of 0.562 for MRE, and 0.694 for absolute residuals. Thus, it is concluded that both multiple linear regression models are the best model for the QUES dataset. However, it should be noted that the MMRE, pred(0.25) and pred(0.30) values of those best models do not satisfy the criteria of an accurate prediction model, either.

6.3 Discussion

Any of the maintainability prediction models constructed in this paper does not satisfy the criteria of an accurate prediction model. However, it is reported that prediction accuracy of software maintenance effort prediction models are often low and thus, it is very difficult to satisfy the criteria [26]. In addition, the prediction accuracy achieved by the maintainability prediction models are certainly higher than the value that would have been if none of the models were used. Thus, it is concluded that the models presented in this paper can predict maintainability of the OO software systems in Li and Henry datasets reasonably well. In particular, the Bayesian network maintainability prediction model has been able to achieve significantly better prediction accuracy than the regression-based models for the UIMS dataset. For the QUES dataset, although the Bayesian network model's prediction accuracy has not been as good as the best regression-based models, that is, the multiple linear regression models, it has been reasonably close. Thus, this could suggest that the Bayesian network model can still be reasonably competitive against the best regression-based models for the QUES dataset.

7 Conclusions

A Bayesian network OO software maintainability prediction model is constructed using the OO software metric data in Li and Henry datasets. The prediction accuracy of the model is evaluated and compared with the regression tree model and the multiple linear regression models using the prediction accuracy measures: the absolute residuals, MRE and pred measures. The results show that the Bayesian network model can predict maintainability of the OO software systems. For the UIMS dataset, the Bayesian network model has achieved significantly better prediction accuracy than the regression tree model and the multiple linear regression models. For the QUES dataset, although the Bayesian network model's prediction accuracy has not been as good as the best models, it has been still reasonably competitive against the best models. Thus, it is concluded that the prediction accuracy of the Bayesian network model is better than, or at least, is competitive against the regression-based models. Those findings have also confirmed that Bayesian network is indeed a useful modelling technique for software maintainability prediction, although further studies are required to realize the full potential as well as the limitation.

The results in this paper also suggest that the prediction accuracy of the Bayesian network model may vary depending on the characteristics of dataset and/or the prediction accuracy measure used. This provides an interesting

direction for future studies. Another interesting direction would be using a Bayesian network with a different structure, for example, a tree augmented Naïve-Bayes classifier (TAN) or a general Bayesian network, for software effort prediction. The authors are currently carrying out a research that applies a general Bayesian network to software development effort prediction. Such studies would allow us to investigate the capability of Bayesian network in Software Engineering further.

Acknowledgements

The authors would like to acknowledge many valuable suggestions made by J. Harraway, Department of Mathematics and Statistics, University of Otago, New Zealand, with regard to the multiple linear regression models presented in this paper.

References

[1] A.J. Albrecht and J.E. Gaffney. Software function, source lines of code, and development effort prediction: a software science validation. *IEEE Transactions on Software Engineering*, SE–9(6):639–648, 1983.

[2] J. Baik, B. Boehm, and B.M. Steece. Disaggregating and calibrating the CASE tool variable in COCOMO II. *IEEE Transactions on Software Engineering*, 28(11):1009–1022, 2002.

[3] B.W. Boehm. *Software Engineering Economics*. Prentice–Hall, 1981.

[4] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Chapman&Hall, 1993.

[5] L.C. Briand, K.E. Emam, D. Surmann, I. Wieczorek, and K.D. Maxwell. An assessment and comparison of common software cost estimation modelling techniques. In *Proceedings of the 21st International Conference on Software Engineering (ICSE'99)*, pages 313–322, 1999.

[6] L.C. Briand, T. Langley, and I. Wieczorek. A replicated assessment and comparison of common software cost estimation modelling techniques. In *Proceedings of the 22nd International Conference on Software Engineering (ICSE'00)*, pages 377–386, 2000.

[7] L.C. Briand and J. Wüst. The impact of design properties on development cost in object-oriented systems. In *Proceedings of the 7th International Software Metrics Symposium (METRICS'01)*, pages 260–271, 2001.

[8] L.C. Briand and J. Wüst. Modeling development effort in object-oriented systems using design properties. *IEEE Transactions on Software Engineering*, 27(11):963–986, 2001.

[9] W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):195–210, 1996.

[10] S.R. Chidamber and C.F. Kemerer. A metrics suite for object-oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994.

[11] S. Chulani, B. Boehm, and B.M. Steece. Bayesian analysis of empirical software engineering cost models. *IEEE Transactions on Software Engineering*, 25(4):513–583, 1999.

[12] P. Congdon. *Bayesian Statistical Modelling*. John Wiley & Sons., 2001.

[13] S.D. Conte, H.E. Dunsmore, and V.Y. Shen. *Software Engineering Metrics and Models*. Benjamin/Cummings Publishing Company, 1986.

[14] R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer–Verlag New York, 1999.

[15] N. Fenton and M. Neil. A critique of software defect prediction models. *IEEE Transactions on Software Engineering*, 25(5):675–689, 1999.

[16] N.E. Fenton and S.L. Pfleeger. *Software Metrics:A Rigorous & Practical Approach*. PWS Publishing Company, second edition, 1997.

[17] F. Fioravanti and P. Nesi. Estimation and prediction metrics for adaptive maintenance effort of object-oriented systems. *IEEE Transactions on Software Engineering*, 27(12):1062–1084, 2001.

[18] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion mmre. *IEEE Transactions on Software Engineering*, 29(11):985–995, 2003.

[19] A.R. Gray. A simulation-based comparison of empirical modelling techniques for software metrics models of development effort. In *Proceedings of the 6th International Conference on Neural Information processing (ICONIP'99)*, volume II, pages 526–531, 1999.

[20] D. Heckerman and M.P. Wellman. Bayesian networks. *Communications of the ACM*, 38(3):27–30, 1995.

[21] R. Jeffery, M. Ruhe, and I. Wieczorek. Using public domain metrics to estimate software development effort. In *Proceedings of the 7th International Software Metrics Symposium (METRICS'01)*, pages 16–27, 2001.

[22] F.V. Jensen. *Bayesian Networks and Decision Graphs*. Springer–Verlag New York, 2001.

[23] J. Kaczmarek and M. Kucharski. Size and effort estimation for applications written in java. *Information and Software Technology*, 46:589–601, 2004.

[24] B.A. Kitchenham, L.M. Pickard, S.G. MacDonell, and M.J. Shepperd. What accuracy statistics really measure. *IEE Proceedings–Software*, 148(3):81–85, 2001.

[25] W. Li and S. Henry. Object-oriented metrics that predict maintainability. *Journal of Systems and Software*, 23:111–122, 1993.

[26] A. De Lucia, E. Pompella, and S. Stefanucci. Assessing effort estimation models for corrective maintenance through empirical studies. *Information and Software Technology*, 47:3–15, 2005.

[27] S.G. MacDonell. Establishing relationships between specification size and software process effort in case environment. *Information and Software Technology*, 39:35–45, 1997.

[28] M. Neil, N. Fenton, and L. Nielsen. Building large-scale bayesian networks. *The Knowledge Engineering Review*, 15(3):257–284, 2000.

[29] L. Pickard, B. Kitchenham, and S. Linkman. An investigation of analysis techniques for software datasets. In *Proceedings of the 6th International Software Metrics Symposium (METRICS'99)*, pages 130–142, 1999.

[30] K. Srinivasan and D. Fisher. Machine learning approaches to estimating software development effort. *IEEE Transactions on Software Engineering*, 21(2):126–137, 1995.

[31] I. Stamelos, L. Angelis, P. Dimou, and E. Sakellaris. On the use of Bayesian belief networks for the prediction of software productivity. *Information and Software Technology*, 45:51–60, 2003.

[32] E. Stensrud. Alternative approaches to effort prediction of erp projects. *Information and Software Technology*, 43:413–423, 2001.

[33] E. Stensrud, T. Foss, B.A. Kitchenham, and I. Myrtveit. An empirical validation of the relationship between the magnitude of relative error and project size. In *Proceedings of the 8th IEEE Symposium on Software Metrics (METRICS'02)*, pages 3–12, 2002.

[34] B. Stewart. Predicting project delivery rates using the Naive–Bayes classifier. *Journal of Software Maintenance and Evolution: Research and Practice*, 14:161–179, 2002.