A Classification Scheme for Studies on Fault-Prone Components

Per Runeson[1], Magnus C. Ohlsson[1] and Claes Wohlin[2]

[1] Dept. of Communication Systems, Lund University, Box 118, SE-221 00 Lund, Sweden
{Per.Runeson, Magnus_C.Ohlsson}@telecom.lth.se

[2] Dept. of Software Engineering & Computer Science, Blekinge Institute of Technology, SE-372 25 Ronneby, Sweden, Claes.Wohlin@bth.se

Abstract. Various approaches are presented in the literature to identify fault-prone components. The approaches represent a wide range of characteristics and capabilities, but they are not comparable, since different aspects are compared and different data sets are used. In order to enable a consistent and fair comparison, we propose a classification scheme, with two parts, 1) a characterisation scheme which captures information on input, output and model characteristics, and 2) an evaluation scheme which is designed for comparing different models' capabilities. The schemes and the rationale for the elements of the schemes are presented in the paper. Important capabilities to evaluate when comparing different models are rate of misclassification, classification efficiency and total classification cost. Further, the schemes are applied in an example study to illustrate the use of the schemes. It is expected that applying these schemes would help researchers to compare different approaches and thereby enable building of a more consistent knowledge base in software engineering. In addition it is expected to help practitioners to choose a suitable prediction approach for a specific environment by filling out the characterisation scheme and making an evaluation in their own environment.

1 Introduction

In the software engineering research community, various approaches for identification of fault-prone components or modules are published, see for example [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. However, the studies are conducted in many different ways and hence the results are not comparable. The studies compare different aspects using different data sets, leading to inconsistent results and confusion. To enable building of knowledge and experience in software engineering, it is necessary to have methods to support comparison and replication of results. The objective here is to provide such support for predicting fault-prone components.

Most studies are conducted using the same principles. First, metrics are collected to be used as predictors. The data are divided into two sets, *fit data* and *test data*. Second, the prediction model is built using the *fit data* set and some statistical technique. The model is evaluated using the *test data* set, and possibly re-built and re-evaluated. Third, the same metrics are collected in a new project and the prediction model is applied in a development project. The output from this step is a prediction of fault-proneness that can be used for quality management purposes.

Even though almost all the published studies are conducted according to the same principles, there are numerous variants of the studies. Different metrics are collected, different statistical techniques are used in building the models, different aspects of fault-proneness are output from the models and they are evaluated in different ways. This makes it hard to make cross-study and meta analyses about prediction of fault-proneness. Furthermore, it is hard for a potential user of the prediction approaches to judge which one to choose. E.g. what does it mean to compare a prediction model which classifies a module as non fault-prone if they are fault-prone in 73% of the cases [4], to another model which has a misclassification rate of 25% [8] on another data set? Which is the better in general, and which is the better for a certain context?

In this paper we propose a classification scheme as a first attempt to overcome those hurdles. The scheme has two constituents, 1) a characterisation scheme which summarises general characteristics, characteristics of the input, the output and the prediction approach, and 2) an evaluation scheme which involves quantitative criteria for evaluation of the ability of predicting fault-proneness for the approach. The classification scheme is an initial attempt to address two different purposes. The first purpose is to characterise new approaches being published, thus bringing more consistent knowledge to the software engineering community, e.g. by meta-analyses. The second purpose is for intended users of the prediction approaches, for example software project managers, to help selecting a suitable prediction approach and evaluate it in a specific context.

In this paper, the techniques applied to build a prediction model are referred to as *method*. The *model* itself is the mathematical formula defining the prediction calculations. The combination of a method used for model building and a selected set of metrics are referred to as a *prediction approach*. A specific instance of an approach is referred to as a *study*.

The paper is organised as follows. In Section 2 the characterisation scheme is presented. The different characteristics and example criteria are given. The evaluation scheme is presented in Section 3. In Section 4 the schemes are used in an example evaluation of three different approaches to illustrate the use of the schemes. Finally, the contributions are summarised in Section 5.

2 Characterisation Scheme

In the strive for managing software projects and in particular the quality of software being developed, different approaches have been defined to estimate properties of software, related to faults resided in the product. The approaches use different metrics, different methods and provide different outputs. In order enable comparison between different approaches, we propose a characterisation scheme.

The scheme contains characteristics that are subject to choices when selecting an approach to apply in a particular environment. The items in the scheme are not orthogonal: a choice of a certain value of a characteristic may limit the choices for other characteristics, i.e. the choices are nested. However, the characteristics included in the scheme are those on which decisions are to be taken by the users of an approach. The characteristics are:

- General – application domain and purpose of approach.
- Output – output type and output metrics.
- Input – artefacts and input metrics.
- Model – model class, optimization criterion and method.

The characteristics chosen are presented in the sections below and summarised in a scheme in Table 1.

Table 1. Characterisation scheme.

	Item	Value
General	Application domain	Telecommunications, Aeronautics, Automotive, Information systems
	Purpose of study	Identify all/certain share of fault-prone components
Output	Output type	Estimation | Ranking | Classification
	Output metric	Number of faults (per unit) | Order of fault-proneness | Fault-prone / non fault-prone
Input	Artefact	Requirements specification | Design specification | Code | Test specification
	Input metrics	List of metrics (requirements, design, code, test)
Model	Model class	1: Input (Product) –> Output (Product) 2: Input (Product, Resource) –> Output (Product, Resource) 3: Input (Product) –> Output (Process) 4: Input (Process) –> Output (Process)
	Optimization criterion	Misclassification | Cost of misclassification | Total classification cost
	Method	Principal Component Analysis | Boolean Discriminant Analysis | Spearman Rank Correlation | Optimized Set Reduction | Neural Network | Regression Analysis | …

2.1 General

The general characteristics present which application domain the models is applied to and the purpose of the study. The purpose is in most cases to identify fault-prone components to enable quality improvement activities. The differences may be in the quality requirements, whether all faults should be removed which is typical for an aeronautics application, or if some level of fault density is acceptable, as the case in telecommunications.

2.2 Output Type

In this characterisation, we are primarily focused on the applicability of the approaches; thus we begin with the output. There are three principal types of output:

1. Approaches for *estimation* of fault content. These approaches try to estimate the number of faults resided in a software component or other artifact. An example of an approach is presented by Ming et al. [9].
2. Approaches for *ranking* fault-prone components. These approaches rank the components with respect to their predicted fault-proneness, and enable classifying components as fault-prone according to a threshold. For example, they may pick out the 10% most fault-prone components. An example of an approach is presented by Ohlsson and Alberg [6] and further refined by Ohlsson and Wohlin [7].
3. Approaches for *classifying* fault-prone components. These approaches classify components into either fault-prone or non fault-prone based on some fixed threshold criterion, for example pointing out all components which have more than five faults. Examples are presented by Khoshgoftaar et. al. [4] and Schneidewind [8].

The output type is hence *estimation*, *ranking* or *classification*. The classification is the weakest form of model. A ranking model can be transformed into a classification model by cutting the ranking list into two, based on some criterion, e.g. percentage of components. An estimation model can be turned into a ranking model by sorting the components according to their estimated value, and turned into a classification model by comparing a threshold on the estimated value.

2.3 Output Metrics

The metric, which is produced as an output of the estimation model, is mainly limited by the output type, as defined above. However, minor variants exist:

- Estimation approaches may predict the number of faults per component, the number of fault per line of code or faults per some other length or size metric.
- The output metric from classification approaches is a boolean; fault-prone or non fault-prone or three-level [12]
- Ranking approaches provide an order number for each component with respect to its fault-proneness.

The metrics are all product metrics, i.e. characteristics of the product itself [11].

2.4 Artefacts

The next characteristic concerns the artefacts to which the approach is applicable. A typical use for a fault-proneness prediction approach is to predict, based on data from one type of artefacts, which artefacts are likely to be fault-prone in a subsequent phase. A quality improvement effort is then planned in the subsequent phase based on the prediction result. For example, more testing time may be planned to fault-prone components than to other components based on data from a design document.

The artefact characteristic tells to which artefacts the approach may be applied. Here we have chosen to illustrate the artefacts through a standard setup of documents: Requirements specification, Design specification, Code, Test specification. The published approaches are primarily applied to the design and code artefacts.

2.5 Input Metrics

The characteristics defined so far, define which prediction we want – the output – and to which type of documents we want to apply the prediction – the artefact. Now we have to study which type of information is available as input to the approach.

The input metrics are the list of metrics collected and used to build the prediction model and to predict fault-proneness of components. The choice of metrics is primarily restricted by the artefact to which the approach is applied. For example, lines of code metrics are not available before the code is available and can hence not be used for prediction purposes if the approach is to be used in the design phase. Most approaches are based on product metrics, either from design or code documents.

2.6 Model Class

Given the input and output metrics, the resulting prediction model can be classified according to the scheme proposed by Fenton and Pfleeger [11]. The classes do not cover all possible combinations of inputs and outputs, but the feasible ones, according to Fenton and Pfleeger.

- Class 1: Input (Product) –> Output (Product)
- Class 2: Input (Product, Resource) –> Output (Product, Resource)
- Class 3: Input (Product) –> Output (Process)
- Class 4: Input (Process) –> Output (Process)

As most models use product metrics from artefacts in earlier phases and predicts products metrics of artefacts in later phases, we generally have models of class 1.

2.7 Optimization Criterion

When fitting a model to the fit data, we may optimize the model according to a specific criterion. This far, we have seen three different criteria published:

- *Misclassification*. This means that the model is Optimized to produce as few misclassifications as possible. A misclassification means that a fault-prone component is classified as non fault-prone or vice versa.
- *Cost of misclassification*. It is generally accepted that some misclassifications cost more than others do. It is more costly to classify a fault-prone component as non fault-prone, which means that faults are slipping through into later phases implying more expensive identification and repair. Classifying non fault-prone components as fault-prone means wasting effort in the project.
- *Total classification cost*. This means that the model is Optimized to give as low cost as possible. The cost included is the cost of the quality actions spent on the components correctly classified as fault-prone, and the cost of misclassification.

The optimization criterion may be one of the three described above.

2.8 Method

Finally a statistical or another method is applied to build the prediction model. Methods utilised include, for example, Principal Component Analysis (PCA) [3], Boolean Discriminant Analysis [8], Spearman Rank Correlation [5], Optimized Set Reduction [1], Neural Network Techniques [2] and Regression Analysis [9]. There are possibly other methods that can be used.

2.9 Use of the Classification Scheme

The classification scheme is intended to bring clarity to differences and similarities between different approaches. To the practitioner, the input and output sections are of primary interest, while for the researcher, the model section is of interest as well.

3 Evaluation Scheme

Among the approaches published in the literature for predicting fault-proneness, many are evaluated with respect to their ability to predict properly. However, there are almost as many evaluation criteria as there are prediction approaches. In order to enable clearer comparisons and cross-study evaluation, we propose an evaluation scheme to be used when evaluating the performance of prediction approaches. Furthermore, detailed enough descriptions of the approaches should be published to enable true evaluation and replication.

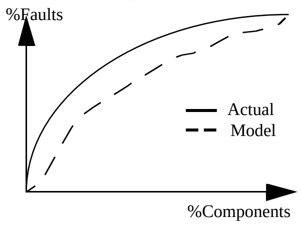
The evaluation criteria are put together in a scheme, in Table 2. Part a) contains a contingency table to show an overall picture of the estimation ability of the approach. Part b) is applicable only to the ranking type of approaches, showing an Alberg diagram [6]. The concept of the Alberg diagram is that if the components are placed in decreasing order according to the number of faults, then the accumulated number of faults for different percentages of the components would be the best any model could achieve. By comparing the actual and the model curves the quality of the model is quantifiable. In the evaluation scheme, we propose criteria with respect to *overall misclassification*, *classification efficiency* and *total classification cost* instead of focusing only on one of them. Part c) is a definition of estimation accuracy for estimation type models.

3.1 Misclassification

The evaluation criterion relates to the approaches' ability to point out components as fault-prone and non fault-prone. Here we define two types of errors that may be conducted when trying to predict fault-prone components[1]:

1. Sometimes the opposite definitions are used, see for example [10], which depends on which hypothesis is tested. We define the null hypothesis, H_0: The component *is* fault-prone, and the alternative hypothesis, H_1: The component is *not* fault-prone.

Table 2. Evaluation scheme for a) classification models, b) ranking models and c) estimation models.

Classification	a)	Model non fault-prone	Model fault-prone	Total	Ranking	b) 
	Actual non fault-prone	A	B Type II	A+B		
	Actual fault-prone	C Type I	D	C+D		
	Total	A+C	B+D	A+B+C+D		
	Type I rate C/(C+D)% Type II rate B/(A+B)% Overall misclassification (B+C)/(A+B+C+D)% Classification efficiency (B+D)/(C+D) Total classification cost: C*Cost$_{\text{Type I}}$+(B+D)*Cost$_{\text{Type II}}$				Estimation	c) $Accuracy = \frac{Faults_{Estimated} - Faults_{Actual}}{Faults_{Actual}}$

- Conducting a type I error means classifying actually fault-prone components as non fault-prone.
- Conducting a type II error means classifying actually non fault-prone components as fault-prone.

The rate of the two error types as such has been used as an evaluation criterion [3, 8]. The approach, which minimises the misclassifications, is then considered the best. The overall misclassification is the total number of the misclassifications of type I and II, normalised by the total number of components. The criteria are defined as follows:

$$\text{Type I rate} = C/(C+D)\% \tag{1}$$

$$\text{Type II rate} = B/(A+B)\% \tag{2}$$

$$\text{Overall misclassification} = (B+C)/(A+B+C+D)\% \tag{3}$$

3.2 Cost of Misclassification

The number of errors does not capture the cost aspects of the classification. Type I errors are, for example, in general more expensive to commit than type II errors are. A fault that remains in a product in the operational phase, costs more than the effort spent on, for example, reviewing an additional component which is misclassified as fault-prone but it is actually not. Therefore the misclassification cost (MCC) is introduced as an evaluation criterion [10], here presented with our notations:

$$MCC = C * C_{Type\ I} + B * C_{Type\ II} \tag{4}$$

It shall be noted that all costs used in this paper are relative costs. In [10] the ratio between $C_{\text{Type I}}$ and $C_{\text{Type II}}$ is exemplified with a factor 10.

The problem with this evaluation criterion is that it, in many cases, has its optimum driven by the type I error. This implies that too many components are pointed out as fault-prone to minimise the risk for type I errors. This provides low type I error rate but a high type II error rate, since the cost of type II errors is lower that the cost of type I errors.

The main purpose of classification approaches is to optimize the use of resources, i.e. the primary concern is the overall cost spent on identifying and improving the fault-prone components. In other words, we would like to minimise the effort spent in development and in maintenance due to the faults in the software. We refer to this a the Total Classification Cost (TCC), which is defined as:

$$TCC = C * C_{Type\ I} + (B+D) * C_{Type\ II} \tag{5}$$

The difference between equations (4) and (5) is the last term which adds the cost of effort spent on components which are correctly classified as fault-prone, thus enabling comparison to the cost of spending effort on all components. The TCC shall be as low as possible. In any case, it shall be less than spending additional effort on all components, i.e. $(A+B+C+D)*C_{Type\ II}$.

For ranking approaches, the evaluation criteria presented above have to be calculated for each threshold value chosen. Instead of tabulating all these data, we propose the Alberg diagram to be used [6]. The Alberg diagram is a graphical representation of the approaches' capability of pointing out fault-prone components, see Table 2b.

For estimation approaches, they can be turned into ranking approaches by sorting the components according to their estimated value, and into classification approaches by applying a threshold. Then the Alberg diagram and the contingency table can be used. In addition, the estimation accuracy can be used as an evaluation criterion, specifically for estimation approaches, see Table 2c.

3.3 Classification Efficiency

The classification efficiency is an evaluation criterion related to cost and misclassifications. However, it provides a viewpoint that is of importance from a practical project management perspective. We define the classification efficiency as the number of components classified as fault-prone divided by the actual number of fault-prone components.

$$\textit{Classification efficiency} = (B+D)/(C+D) \tag{6}$$

The optimum value of classification efficiency is hence 1 (B and C equal 0). A value less than 1 means that too few components are classified as fault-prone while a value of more than one means that too many components are classified as fault-prone.

3.4 Use of the Evaluation Scheme

The evaluation scheme is intended to compare capabilities of the different models. For a practitioner, it can be used to compare different methods in a specific environment, while the researcher may compare methods applied to data from different environments. Depending on which of the characteristics is considered most important, the misclassification, the cost of misclassification or the efficiency is given the highest priority in the comparison.

4 Example Study

In this section we apply the characterisation and evaluation schemes presented in Section 2 and Section 3 to illustrate typical use of the schemes. The example context is an organisation which intends to investigate methods for identification of fault-proneness in their specific settings. They have selected three different candidate approaches:

- Schneidewind (S97) [8]
- Khoshgoftaar et. al. (K96) [4]
- Ohlsson and Wohlin (O98) [7]

In Table 3 the results from the characterisation can be found.

Table 3. Characterisation of three example approaches.

	Item	S97	K96	O98
General	Application domain	Aeronautics	Telecommunications	Telecommunications
	Purpose of study	Identify all fault-prone components	Identify a certain share of fault-prone components	Identify a certain share of fault-prone components
Output	Output type	Classification	Classification	Ranking
	Output metric	Fault-prone / non fault-prone	Fault-prone / non fault-prone	Order of fault-proneness
Input	Artefact	Code	Code	Design
	Input metrics	List of code metrics, see [8].	List of code metrics, see [4].	List of code metrics, see [7]
Model	Model class	Class 1: Input(Product) -> Output(Product)	Class 1: Input(Product) -> Output(Product)	Class 1: Input(Product) -> Output(Product)
	Optimization criterion	Misclassification	Misclassification	Total classification cost
	Method	Kolmogorov-Smirnov and Boolean Discriminant Analysis	PCA and non parametric Stepwise Discriminant Analysis	Spearman Rank Correlation

4.1 Characterisation

General. The approaches represent two different application domains, aeronautics and telecommunications. The domains also reflect the purpose, where the aeronautics approach aims at identifying all fault-prone components, while in the telecommunications domain, only components with more the a certain number of faults are considered fault-prone.

Output. The approaches represent two output types, *classification* and *ranking*. S97 and K96 both produce a binary result, i.e. fault-prone or non fault-prone, while O98 ranks the components in decreasing order according to their fault-proneness. The preferred output type depends on the objective of the study. The objective for project management might, for example, be to priorities available resources like test time and review time and then S97 and K96 could be considered.

Another case is where management, for example, can spend effort on additional reviews on 20 percent of the components. Which components should be considered? To answer this question, approaches like O98, which rank the components in fault-proneness order, are more useful.

Another difficulty with the classification approaches is which threshold value to choose. Depending on the domain the threshold may differ. For example, the S97 approach was developed to be applied to the aeronautics domain, thus quality requirements are very high and the threshold is set to zero faults in the components [8]. The K96 approach sets the threshold based on heuristics [4], i.e. after a discussion with project engineers they decided on a certain threshold.

Input. In order to enable early planning of actions to prevent fault-proneness, the approaches shall be applicable as early as possible. The phase when the approach is applied and the input metrics depend on each other. If the prediction model is based on metrics from the design phase, it can be applied earlier than if it is based on code metrics from the implementation phase. In this case, the O98 approach is based on design metrics while K96 and S97 are based on code metrics.

Depending on the purpose of the application of the approach, it is feasible to use the prediction model with other metrics than described in the original approach. For example, the S97 model could be used with design metrics. In the example study, we have a set of design metrics available, which are to be used, see Table 4.

Model. All three approaches take product metrics as input and predict product metrics; thus they belong to model class 1 according to Fenton and Pfleeger's classification [11]. The optimization criteria are different for the approaches. S97 and K96 use misclassification while O98 uses total classification cost.

Different statistical methods for model building are used. S97 uses Boolean Discriminant Analysis. To choose suitable metrics, S97 use Kolmogorov-Smirnov distances. K96 uses Non-parametric Discriminant Analysis and Principal Components Analysis to extract factor scores to be used in the Discriminant Analysis. Finally, O98 uses Spearman Rank Correlations.

4.2 Evaluation

To be able to compare the different approaches in the example we use the same data set for all candidates even though they were originally created for use in a different phase or with some other input metrics. Hence, actually only the prediction method are used, not the complete approach (see Section 1). This is the typical situation when using the method in a new context where different sets of metrics are collected.

Data Description. The example study is based on an investigation of 28 software components for two releases of a large real time system in the telecommunication domain. The actual releases investigated are considered typical for releases of this particular system, although the choice of system and releases was based on availability. The data set has been used earlier in other studies, for example [12].

The size of the 28 components varies between 270 and 1900 lines of code. The programming language used is a company internal language, targeted at their specific hardware platform. For each of the components, the number of fault reports from test and operation were counted and several design and code metrics were collected.

In total, ten metrics originating from the design documentation were collected. The data collected were primarily a count of the number of symbols of different types. The language used during design is a predecessor to SDL, (ITU-T Specification and Description Language) [13], and it is basically a finite-state-machine with states and signals. A modified version of McCabe's Cyclomatic Complexity [14] was used as well. The modification was simply due to being able to handle signal sending and reception respectively. The design metrics are listed in Table 4. Please note that all metrics are scaled due to confidentiality.

Table 4. Design metrics collected.

Metric	Description
SDL-pages	Number of design pages in terms of SDL diagrams.
Tasks	Number of task symbols in the SDL descriptions.
Outputs	Number of output symbols in the SDL descriptions.
Inputs	Number of input symbols in the SDL descriptions.
If	Number of if-boxes in the SDL descriptions.
States	Number of state symbols in the SDL descriptions.
McCabe	This metric is an adaptation of the original proposal by McCabe. The metric is adapted to enable measurement on design and in particular to include the signalling concept used in SDL.
External inputs	Number of unique external inputs to a component.
External outputs	Number of unique external outputs to a component.
Internal signals	Number of unique internal signals within a component.

We defined the fault-prone group as those components with $Faults \geq 10$ and the non fault-prone ones as $Faults < 10$. We used the data from release *n* as the *fit data* set and the data from release *n+1* as the *test data* set. In the fit set there were six components classified as fault-prone and in the test data set there were eleven.

In the cost model, we have used a cost ratio between type I and type II errors of 10 (see [10]) i.e. type I errors are ten times as expensive as type II errors. However, to highlight the sensitivity of the results to the cost ratio, a cost ratio of 5 has also been used. The total classification cost based on the lower cost ratio is denoted in parentheses in the tables below.

Results. The results from the application of the *S97* model can be found in Table 5. The classification is based on only one variable, External inputs, The results from the Kolmogorov-Smirnov test provides a maximum difference of 0.818 and a significance of 0.004 for External inputs. To add extra variables when creating the model from the fit data set did not improve the results.

Table 5. Results from S97.

Classification		Model non f-p	Model f-p	Total
	Actual non f-p	14	3	17
	Actual f-p	4	7	11
	Total	18	10	28
	Type I rate: 36% Type II rate: 18% Overall misclassification: 25% Classification efficiency: 0.9 Total classification cost: 50 (30)			

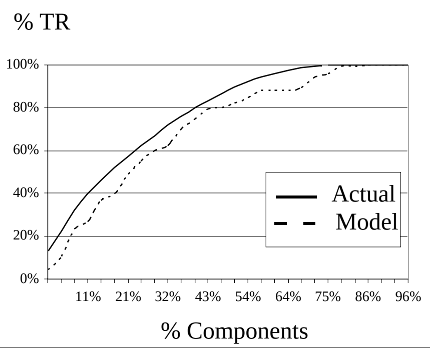
Table 6. Results from K96.

Classification		Model non f-p	Model f-p	Total
	Actual non f-p	15	2	17
	Actual f-p	8	3	11
	Total	24	5	28
	Type I rate: 73% Type II rate: 11% Overall misclassification: 36% Classification efficiency: 0.45 Total classification cost: 85 (45)			

The results from the *K96* model can be found in Table 6. The Principal Components Analysis provides two factors that are reduced to one when the Stepwise Non-parametric Discriminant Analysis is applied with a significance level of 0.05. The prior probabilities for the fault-prone and the non fault-prone components are 0.79 and 0.21 according to the size of the groups and the smoothing parameter for the normal kernel is 0.05. Choosing the smoothing parameter is difficult and in this case we decided on 0.05, because a larger value would not have classified any components as fault-prone.

The results from the *O98* model can be found in Table 7. The results from the Spearman Rank Correlations provides correlation between External inputs and faults of 0.645 with a significance of 0.001. Though, the combination of If and External inputs had a correlation of 0.677 with a significance >0.001. Therefore, this combination was used.

Table 7. Results from O98.

Classification		Model non f-p	Model f-p	Total	Ranking
	Actual non f-p	14	3	17	
	Actual f-p	3	8	11	
	Total	17	11	28	
	Type I rate: 27% Type II rate: 18% Overall misclassification: 21% Classification efficiency: 1.0 Total classification cost: 41(26)				

The O98 model has to be evaluated for a given threshold, since it provides a ranking of components in fault-proneness order. In this case we have chosen a threshold of 39 percent, which corresponds to the number of components considered fault-prone for the S97 and K96 models (11/28), thus making the prerequisites the same for all three models. This is shown in the Alberg diagram in Table 7. Noticeable is that the classification efficiency is 1.0 per definition, since the ranking model points out 39 percent of the components, if 39 percent are considered fault-prone. Therefore, the number of type II errors are always equal to the number of type I errors in this model, while the type I and type II *rates* differ.

Interpretation. The scheme contains evaluations with respect to three aspects. They are the following:

- Overall misclassification
- Classification efficiency
- Total classification cost

As mentioned before, we have used a cost ratio between type I and type II errors to be 5 or 10. The different classification results from the previous section are summarised in Table 8, where the reference costs for spending no effort and spending extra effort on all components are provided in the last two rows. Figures from the original studies are presented as a reference in Table 8.

Table 8. Comparison of models in the example study and in the original studies.

	Example study			Original studies		
	S97	K96	O98	S97	K96	O98
Type I rate	36%	73%	27%	2.3%	35.7%	24%
Type II rate	18%	11%	18%	47%	10.5%	58%
Overall misclassification	25%	36%	21%	28%	17.2%	34%
Classification efficiency	0.9	0.45	1.0[a]	1.6	0.97	1.0[a]
Total classification cost	50 (30)	85 (45)	41 (26)	79 (74)	129 (79)	516 (321)
Cost with no effort	110 (55)			430 (215)	280 (140)	670 (335)
Cost with extra effort on all	28 (28)			100 (100)	133 (133)	232 (232)

a. By definition

In this example study, the S97 model classified 18 percent as fault-prone even though they were not and the model missed 36 percent of the fault-prone components. The overall misclassification was 25 percent and the classification efficiency was 0.9, i.e. the model classified 0.9 components as fault-prone for every actual fault-prone component. The total classification cost was 50 (30 if the ratio was set to 5). It should be noted that these numbers should be compared to the cost of spending extra effort on all components, which is 28.

The approach in K96 provides a type I rate of 73 percent and a type II rate of 11 percent. The type I rate is very high and the reason is that the model classifies too few components as fault-prone. The model should have predicted at least twice as many component as fault-prone to be able to perform better. Because of the high type I rate, the total classification cost is as high as 85 (45 if the ratio is set to 5).

The O98 model is, as mentioned earlier, a little bit different since it ranks the components instead of classifying them. By definition it classifies the correct number of components and therefore the classification efficiency is 1.0. The type I rate is 27 percent and the type II rate is 18 percent.

The approaches above show different abilities with respect to the evaluation criteria to illustrate the use of the evaluation scheme. An approach should provide estimates with low misclassification rates. If we are to accept errors, type II errors are preferred, since they are less costly than type I errors. S97 and O98 are fairly equal but K96 have a large type I rate. If we are interested in ranking models, O98 should be selected, otherwise either O98 or S97 can be used.

5 Summary

Numerous approaches to prediction of fault-proneness have been presented in the literature. They are evaluated using various schemes and criteria. In this paper an initial attempt to improve the comparison an use of classification schemes is presented. It has two core constituents is presented: 1) a *characterisation* scheme which provides an overview of the input to the approaches, the output and the prediction model itself; 2) an *evaluation* scheme which enables qualitative evaluation of the approaches' prediction capabilities. If these schemes are used in presenting new approaches and evaluation of existing ones, the research progress can be easier to evaluate and apply.

The characterisation scheme can not only be used for presenting studies, but also selecting prediction approaches based on needs by the users, for example software managers. They can identify their needs based on the different characteristics of the approaches and evaluate the capabilities of different approaches in their environment.

To illustrate the usage of the schemes an example study is presented which evaluates three different approaches with different characteristics. For the evaluation scheme we use data from two releases of a system to illustrate the models' abilities to classify components with respect to misclassification rates and costs. The results from the example study are compared to the original studies.

It is concluded that it is not sufficient to compare the misclassification rate. The classification efficiency and total classification cost provide other aspects of the models abilities, which are related to the intended use of the models in a software development context. Furthermore, the example study and the original studies show different results as they are based on different data sets.

The proposed classification scheme would allow for a better comparison between different prediction approaches. The scheme is expected to support comparison, replication of results and meta-analyses, leading to better and more consistent knowledge and more collected experience in software engineering.

Acknowledgment

We would like to thank Per-Erik Isberg, Department of Statistics, Lund University, for his support on statistical methods. This work was partly funded by The Swedish National Board for Industrial and Technical Development (NUTEK), grant 1K1P-97-09690 and 1K1P-97-09673.

References

1. Briand, L.C., Basili, V.R. and Hetmanski, C.J., "Developing Interpretable Models with Optimized Set Reduction for Identifying High-Risk Software Components", *IEEE Transactions on Software Engineering*, 19(11), 1993, 1028–1044.
2. Khoshgoftaar, T.M and Lanning, D.L, "A Neural Network Approach for Early Detection of Program Modules Having High Risk in the Maintenance Phase", *Journal of Systems and Software*, 29(1), 1995, 85–91.
3. Khoshgoftaar, T.M., Allen, E.B., Kalaichelvan, K.S and Goel, N.,"Early Quality Prediction: A Case Study in Telecommunications", *IEEE Software* (January 1996), 65–71.
4. Khoshgoftaar, T.M, Allen,E.B., Halstead, R. and Trio, G.P., "Detection of Fault-prone Software Modules During a Spiral Life Cycle", *Proceedings of the International Conference on Software Maintenance*, (Monterey, California, USA, November 96) 69-76.
5. Ohlsson, N., Helander, M. and Wohlin, C., "Quality Improvement by Identification of Fault-Prone Modules using Software Design Metrics", In *Proceedings Sixth International Conference on Software Quality*, (Ottawa, Ontario, Canada, 1996), 1–13.
6. Ohlsson, N. and Alberg, H., "Predicting Fault-Prone Software Modules in Telephone Switches", *IEEE Transactions on Software Engineering*, 22(12), 1996, 886–894.
7. Ohlsson, N. and Wohlin, C., "Experiences of Fault Data in a Large Software System", *Information Technology Management: An International Journal*, 2(4), 1998, 163–171.
8. Schneidewind, N. F., "Software Metrics Model for Integrating Quality Control and Prediction", in *Proceedings Fourth International Software Metrics Symposium*, (Albuquerque, NM, USA, November 1997) 402–415.
9. Zhao, Ming, Wohlin, C., Ohlsson, N. and Xie, Min, "A Comparison between Software Design and Code Metrics for the Prediction of Software Fault Content", *Information and Software Technology*, 40(14), 1998, 801–809.
10. Khoshgoftaar, T.M. and Allen, E.B., "The Impact of Cost of Misclassification on Software Quality Modeling", in *Proceedings Fourth International Software Metrics Symposium*, (Albuquerque, NM, USA, November 1997), 54–62.
11. Fenton, N.E. and Pfleeger, S.L., *Software Metrics: A Rigorous and Practical Approach*, Thomson Computer Press, 1996.
12. Ohlsson, M.C. and Wohlin, C., "Identification of Green, Yellow and Red Legacy Components", In *Proceeding of International Conference on Software Maintenance*, (Bethesda, Washington D.C, USA, November 1998), 6–15.
13. ITU, "Recommendation Z.100: SDL – Specification and Description Language", 1988.
14. McCabe, T. J., "A Complexity Measure", *IEEE Transactions on Software Engineering*, 4(2), 1976, 308–320.