Empirical Software Engineering, 10, 183–218, 2005.

Assessment of a New Three-Group Software Quality Classification Technique: An Empirical Case Study

TAGHI M. KHOSHGOFTAAR taghi@cse.fau.edu
Florida Atlantic University, Boca Raton, FL 33431, USA

NAEEM SELIYA nseliya@cse.fau.edu
Florida Atlantic University, Boca Raton, FL 33431, USA

KEHAN GAO kgao@cse.fau.edu
Florida Atlantic University, Boca Raton, FL 33431, USA

Editor: Claes Wohlin

Abstract. The primary aim of risk-based software quality classification models is to detect, prior to testing or operations, components that are most-likely to be of high-risk. Their practical usage as quality assurance tools is gauged by the prediction-accuracy and cost-effective aspects of the models. Classifying modules into two risk groups is the more commonly practiced trend. Such models assume that all modules predicted as high-risk will be subjected to quality improvements. Due to the always-limited reliability improvement resources and the variability of the quality risk-factor, a more focused classification model may be desired to achieve cost-effective software quality assurance goals. In such cases, calibrating a three-group (high-risk, medium-risk, and low-risk) classification model is more rewarding.

We present an innovative method that circumvents the complexities, computational overhead, and difficulties involved in calibrating pure or direct three-group classification models. With the application of the proposed method, practitioners can utilize an existing two-group classification algorithm thrice in order to yield the three risk-based classes. An empirical approach is taken to investigate the effectiveness and validity of the proposed technique. Some commonly used classification techniques are studied to demonstrate the proposed methodology. They include, the C4.5 decision tree algorithm, discriminant analysis, and case-based reasoning. For the first two, we compare the three-group model calibrated using the respective techniques with the one built by applying the proposed method. Any two-group classification technique can be employed by the proposed method, including those that do not provide a direct three-group classification model, e.x., logistic regression and certain binary classification trees, such as CART. Based on a case study of a large-scale industrial software system, it is observed that the proposed method yielded promising results. For a given classification technique, the expected cost of misclassification of the proposed three-group models were significantly better (generally) when compared to the technique's direct three-group model. In addition, the proposed method is also evaluated against an alternate indirect three-group classification method.

Keywords: Software quality prediction, three-group classification, discriminant analysis, decision trees, case-based reasoning, expected cost of misclassification.

1. Introduction

A software quality assurance (SQA) team strives for achieving high software reliability and minimal software maintenance (Beizer, 1990), especially in the case of high-assurance systems (Xu and Khoshgoftaar, 2001). The extent of applying software testing and reliability improvement endeavors is usually dependent on the available (project-specific) software testing and quality improvement resources. Subsequently, if timely predictions of the likely-risk of software modules were available, a more focussed quality control plan can be devised. Specifically, the quality improvement of high-risk modules can be given the highest priority.

Software metrics-based quality classification models can achieve such a timely risk-based quality estimation (Ebert, 1996; Fenton and Pfleeger, 1997). Software metrics and risk-factor (e.x., number of faults) data collected during a previous system release (or similar project) are used to calibrate a classification model. Subsequently, the model uses the known software metrics of the current project to estimate the risk-factor for its software modules (Gray and MacDonell, 1999). A few of such classification techniques include, optimized set reduction (Briand et al., 1993), classification trees (Porter and Selby, 1990; Takahashi et al., 1997), case-based reasoning (Khoshgoftaar and Seliya, 2003), logistic regression (Schneidewind, 2001), neural networks (Hochman et al., 1997), and fuzzy logic (Ebert, 1996).

Classifying software modules into two risk-based groups, i.e., fault-prone (*fp*) and not fault-prone (*nfp*), is the more commonly practiced trend (Khoshgoftaar and Allen, 2001; Khoshgoftaar et al., 2002; Ohlsson et al., 1996; Ohlsson and Runeson, 2002). Based on the prediction of such a model, the available reliability improvement resources can be allocated to the low-quality software modules, allowing a cost-effective SQA approach. Classification models are built on the premise that quality improvement efforts will be applied to all the (predicted) *fp* modules.

Development teams have time-and-again realized that testing and quality improvement resources are almost-always limited and finite. Consequently, all *fp* modules may not be inspected and improved. In addition, when the variation of the risk-factor among the predicted *fp* modules is too large, it can significantly limit the effectiveness of a two-group classification model. For example, with regards to improving return-on-investment (ROI), which of the *fp* modules should be targeted first? However, we note that the application of a three-group model may not be beneficial in all software engineering situations. For example, we have modeled systems where the number of faults ranged from 0 to 4 (Lanning and Khoshgoftaar, 1995). In systems where the faults range is small (i.e., faults detected by clients), the application of a two-group model may be sufficient for the desired quality needs.

In situations where an improved cost-effective resource utilization (better ROI) and a more focused SQA plan is desired, a three-group classification would be more rewarding. For instance, such a model can identify modules as either Red (high-risk), Yellow (medium-risk), or Green (low-risk) (Ohlsson et al., 1999; Ohlsson and Wohlin, 1998; Szabo and Khoshgoftaar, 2000) based on software metrics collected prior to system tests. Since the primary goal of a three-group classification model is to improve the cost-effectiveness and ROI of the reliability improvement efforts, it is important that the model identifies all the high-risk modules correctly. Therefore, all modules predicted as Red must be targeted for additional testing. In contrast, a Yellow module may be inspected using a fraction of the time and resources allocated to a Red module, depending on resource availability. Furthermore, modules predicted as Green will not be reviewed at all.

The application of a software quality classification model is usually determined where in the software life cycle such a model is needed. We have already discussed how a three-group classification model can be used to attain a cost-effective software quality improvement. As another application example of a three-group software quality classification, a model calibrated using software design-level metrics can be used to delegate implementation of the Red modules to experienced programmers. In addition, less experienced programmers can be assigned to implement the modules predicted as Yellow or Green. Similarly, other software metrics may be used for modeling purposes depending on their availability and where in the life cycle a software quality classification model is desired.

This study introduces a new three-group software quality classification modeling technique. The method does not involve a new mathematical model. Instead, it proposes an innovative use of any existing two-group technique three times, in order to calibrate a three-group model. One may argue, why bother with the proposed approach?, and why not just use an existing three-group modeling technique? The use of three-groups as compared to two-groups is not very common. If the development and SQA teams desire a three-group classification, they may have to resort to a three-group technique they are not familiar with. Investing time, effort, and resources toward learning an unfamiliar method may be a deterring factor in calibrating a three-group model. The organization will be more inclined to use a two-group modeling technique with which it has had more experience. By utilizing any existing two-group technique, the proposed three-group method avoids the need for an organization to expend resources in learning a new (unfamiliar) three-group classification technique.

As compared to a two-group classification problem, a three-group classification involves some impractical and often difficult to address issues, such as complexity, comprehensibility, and computational overhead. In addition, not all existing quality classification techniques can provide a three-group model, such as logistic regression (Khoshgoftaar and Allen, 1999), Boolean discriminant functions (Khoshgoftaar and Seliya, 2002; Schneidewind, 1997), and binary decision trees (Khoshgoftaar and Allen, 2001; Khoshgoftaar et al., 2002).

The complexities of calibrating three-group models can be very taxing when certain classification techniques are used. For example, Boolean discriminant functions (BDFs) are used to classify modules into two classes (Khoshgoftaar and Seliya, 2002; Schneidewind, 1997), depending on whether a module satisfies (or not) the associated BDF. If a similar methodology were to be adopted for a three-group model, a *trinary* discriminant function would be needed, which inherently involves more complex computations and is also difficult to comprehend. Consequently, for an organization that is familiar with such techniques, the simplicity of a two-group model may be preferred, even if a three-group model can yield more rewarding SQA.

As mentioned earlier, not all classification techniques facilitate a three-group model. In such cases, since the practitioner cannot ascertain if those methods can provide better models, s/he may be missing on the opportunity to calibrate a better three-group model. In addition, for techniques that are capable of providing a direct[1] three-group model, such as neural networks (Khoshgoftaar and Lanning, 1995), discriminant analysis (Szabo and Khoshgoftaar, 2000), and non-binary decision trees (Quinlan, 1993), computational overhead, modeling complexity and model-comprehensibility may be of practical concern.

The proposed three-group modeling approach circumvents the difficulties and complexities associated with calibrating pure three-group classification models. In cases where such a model is desired, practitioners can simply apply the proposed methodology to any existing[2] two-group technique (which they are familiar with), including those that take the machine learning and computational intelligence approaches (Michalski et al., 1998). However, does the proposed methodology yield useful classification models as compared to those calibrated directly by some techniques? A logical solution to this question would be to evaluate the performance of a model built using a direct three-group classification technique with the one (based on the same technique) using the proposed method.

This paper presents such a comparative study for two techniques that facilitate a direct three-group classification model: discriminant analysis (Szabo and Khoshgoftaar, 2000) and the C4.5 decision tree algorithm (Ponnuswamy, 2001; Quinlan, 1993). The goal of the comparative study is to inspect how does the performance of the proposed model stand up against that of the model calibrated using the technique's direct approach. In addition to these two techniques, we also demonstrate the application of the proposed methodology with our previously developed two-group technique using case-based reasoning (CBR) (Khoshgoftaar and Seliya, 2003). Though a direct three-group model using CBR can be implemented, to our knowledge there has been no published work demonstrating the same. This is likely because of the computational complexity and modeling difficulty involved.

A basic measurement of accuracy for a classification model is its ability to classify modules correctly, i.e., a lower misclassification errors. In a three-group (Red, Yellow, and Green) model, there are six different types of misclassifications. Since the costs and effort needed to rectify the different misclassifications are (practically speaking) disparate, evaluating such models based solely on their individual misclassification error rates is impractical, and may lead to dubious model-selection.[3] Consequently, in our studies we use the *expected cost of misclassification* (see next section) as a singular unified measure that incorporates the error rates as well as their respective costs.

An empirical software engineering approach is followed in our investigations (Wohlin et al., 2000). The case study presented consists of software metrics and fault data collected from embedded software applications of a wireless telecommunication configurations system. It is observed that the proposed three-group classification approach provides simplicity, flexibility, and comprehensibility in model-calibration and model-interpretation. In the context of the discriminant analysis and C4.5 decision tree techniques, our empirical studies concluded that the proposed method yielded significantly better classification models than those built by the techniques directly. The comparative tests are based on the p-values obtained from a pairwise Z-test. Furthermore, the usefulness of a computational intelligence technique, such as CBR, can be explored for calibrating better three-group classification models.

The paper continues, in Section 2, with our discussion on the expected cost of misclassification as a model-evaluation measure. Section 3 discusses our proposed modeling approach, whereas, in Section 4 we present a brief overview of the three classification techniques. Sections 5 and 6 discuss the empirical case-study and its results. In Section 7, we compare the performances of the proposed indirect three-group classification method with an alternate indirect three-group classification method suggested by an anonymous reviewer. Finally, we summarize with conclusions and directions for future work.

2. Evaluating Classification Models

Classification models are associated with their different types of misclassification error rates. For example, let Type_AB be a misclassification that misclassifies a module of the A group as a module of the B group. The Type_AB misclassification error rate, denoted as $Pr(B|A)$, is defined as the fraction of the number of modules in group A that are misclassified as belonging to group B, i.e.,

$$Pr(B|A) = \frac{n_{AB}}{n_A} \tag{1}$$

where, n_{AB} represents the number of modules in group A that are misclassified as belonging to group B, and n_A represents the total number of modules in group A.

Though the basic measurement of the accuracy of a classification model is its misclassification error rates, evaluating a model based solely on its error rates is inappropriate. This is primarily because the different types of misclassifications are associated with disparate corrective costs and efforts. For example, when a high-risk module is predicted as low-risk, it may not be subjected to quality improvements: implying a lost opportunity of improving the module prior to operations. Consequently, expensive on-site repairs and system-downtime may be needed to fix the fault. On the other hand, when a low-risk module is flagged as high-risk, it implies that it will be subjected to inspection and reviews which may later be determined as unnecessary.

Comparing three-group classification models based on their misclassification rates can be a difficult task, because each of their six error rates could vary (lower or higher) as compared to their respective competing counterparts. A solution to such a problem has usually been obtained by evaluating models based on their overall error rates. However, such a strategy implies that the error rates have equal costs of misclassification: an impractical approach. This study utilizes a singular unified measure, expected cost of misclassification (ECM), that incorporates the respective costs of the error types and the prior probabilities of class memberships.

Our research team first investigated the use of ECM in the context of controlling overfitting in two-group classification tree models (Khoshgoftaar et al., 2000c). A software quality assurance team can determine which classification model yields the best (lowest) ECM for the given software project. The ECM model-evaluation measure used in our studies is defined as (Seber, 1984),

$$\text{ECM} = \sum_{i=1}^{k} \left(\pi_i \sum_{j=1}^{k} C(j|i) Pr(j|i) \right) \tag{2}$$

where, k is the number of classes or groups; π_i is the prior probability of the class i membership; $C(j|i)$ and $Pr(j|i)$ are respectively, the misclassification cost and probability

Table 1. Misclassifications for a three-group model.

Error	Cost	Number	Description
Type_GR	C_{GR}	n_{GR}	Green misclassified as Red
Type_GY	C_{GY}	n_{GY}	Green misclassified as Yellow
Type_YR	C_{YR}	n_{YR}	Yellow misclassified as Red
Type_YG	C_{YG}	n_{YG}	Yellow misclassified as Green
Type_RY	C_{RY}	n_{RY}	Red misclassified as Yellow
Type_RG	C_{RG}	n_{RG}	Red misclassified as Green

of assigning an observation of class i to class j. For a correct classification the cost, i.e., $C(i|i)$, is zero. From a Bayesian viewpoint, the class proportions of the population are information which is known prior to applying a model. In our empirical studies, π_i is approximated (for a given data set) as the ratio of the number of modules of class i to the total number modules in the data set. For a three-group classification problem, Equation (2) is expanded to yield,

$$\mathrm{ECM} = \frac{1}{N}(n_{GR}C_{GR} + n_{GY}C_{GY} + n_{YR}C_{YR} + n_{YG}C_{YG} + n_{RY}C_{RY} + n_{RG}C_{RG}) \tag{3}$$

where, N is the total number of observations in the data set. The other notations are summarized in Table 1.

3. Proposed Three-Group Classification Method

A three-group classification model requires the estimation of two threshold (or critical) values in order to delineate the three classes. For a given organizational environment, these threshold values are usually dependent on past experience with previous system releases or similar projects. It should be noted that the computational complexity of a classification problem increases dramatically as the number of groups in a classification problem increases. Therefore, generally speaking, the simplicity and comprehensibility of a two-group model is preferred over a direct three-group model.

The proposed technique can be applied by utilizing any existing two-group classification technique three times, yielding the three risk-based classes. Therefore, it can be used with classification methods that do not provide a three-group classification, such as logistic regression and certain binary classification trees, i.e., CART. Furthermore, the attractiveness and simplicity of a two-group classification is maintained reasonably. We now present the details of the proposed three-group classification technique.

The modules of the training data set, i.e., *fit* data set, are partitioned into the three risk-based classes: Red, Yellow, and Green. The threshold values of the risk factor that delineate the three classes are determined according to the quality control needs of a given project. In our studies the number of faults in a module is used to assess its quality factor or risk class. Let the two threshold cut-off values for the number of faults (denoted as *Fault*) be thd_{low} and thd_{high} where, $thd_{low} < thd_{high}$. A module i in the fit data set can then be classified into a group, denoted as $Class_i$, based on the following rules:

$$Class_i = \begin{cases} \text{Green}, & \text{if } Fault \le thd_{low} \\ \text{Yellow}, & \text{if } thd_{low} < Fault \le thd_{high} \\ \text{Red}, & \text{otherwise.} \end{cases} \tag{4}$$

The following steps illustrate the application of the proposed three-group classification technique. These steps can be repeated as needed by varying the model-parameters specific to the classification technique that is being used.

1. Using all the Green and Red modules in the fit data set, calibrate a (first) two-group classification model, and denote it as Model_GR. Apply this fitted model to parse the test data set into two classes, *Green** and *Red**, and denote these two parsed groups as Test-I and Test-II, respectively. The *'s indicates that the two groups obtained by parsing the test data set may not be purely Green or Red. Some of the modules in the Test-I and Test-II data sets may be Yellow, i.e., medium-risk.

2. Using all the Green and Yellow cases in the fit data set, calibrate another (second) two-group classification model, and denote it as Model_GY. Apply this fitted model to parse the Test-I group (defined in Step 1) into two sub-groups. Denote these two sub-groups as Test-Green and Test-Yellow_I. All the modules in the Test-I group are classified as either Green or Yellow. At this point, all the Green modules of the test data set have either been classified correctly or have been misclassified.

3. Using all the Yellow and Red cases in the fit data set, calibrate another (third) two-group classification model, and denote it as Model_YR. Apply this fitted model to parse the Test-II group (defined in Step 1) into two sub-groups and denote them as Test-Yellow_II and Test-Red. Hence, all the modules in the Test-II group are classified as either Yellow or Red. At this point, all the Red modules of the test data set have either been classified correctly or have been misclassified.

4. Merge the Test-Yellow_I and Test-Yellow_II groups to yield the Test-Yellow group. At this stage of the process, the three risk-based classes of the test data set have been identified, i.e., predicted. The model is now ready of performance evaluation (see Section 2).

5. Depending on the project's model-selection criteria for obtaining the preferred three-group classification model, certain parameters of the underlying classification technique can be varied to yield different three-group models. For example, in the case of discriminant analysis (Berenson et al., 1983), the smoothing parameter (λ) can be adjusted to satisfy model-selection criteria.

A diagrammatic representation of the described three-group modeling technique is presented in Figure 1, in which Model_GR, Model_GY, and Model_YR represent the

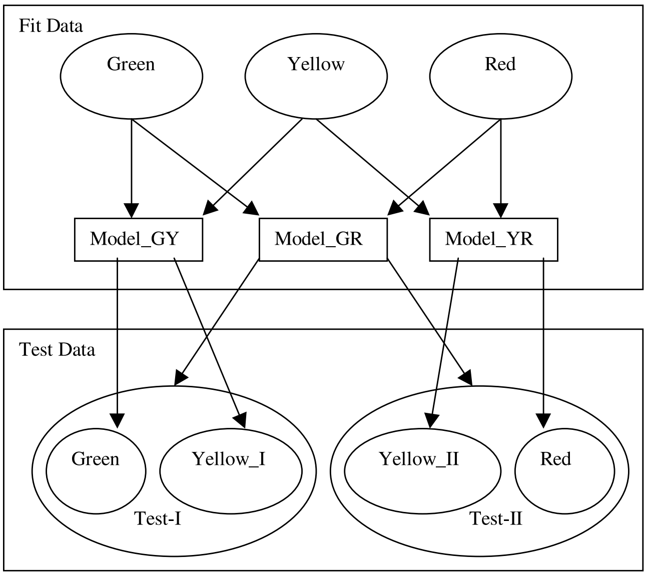


Figure 1. Proposed three-group classification technique.

models built (above) in Steps 1, 2, and 3, respectively. For each two-group model calibrated above, we employed a classification rule that is similar to our previously proposed generalized classification rule (Khoshgoftaar and Allen, 2000).

4. Classification Techniques Studied

In our empirical studies regarding the validation of the proposed algorithm, we investigated several classification techniques, including discriminant analysis (Szabo and Khoshgoftaar, 2000), the C4.5 decision tree algorithm (Ponnuswamy, 2001; Quinlan, 1993), case-based reasoning (Khoshgoftaar and Seliya, 2003), and logistic regression (Khoshgoftaar and Allen, 1999). However, to avoid a large sized paper, we presented our findings of only the first three classification methods.

The aim of this section is not to provide a detailed description of the three classification techniques presented. Instead, a brief overview to illustrate the key concepts of each technique is provided. For further algorithmic and modeling details of these classification methods, please refer to the respective references provided.

4.1. Discriminant Analysis

Discriminant analysis (DA) is a multivariate statistical technique for determining the optimum assignment, i.e., minimum misclassification, of observations into two or more classes based on quantitative predictors of a fit data set (Berenson et al., 1983). The resulting discriminant model can then be used to classify the observations with known predictors but unknown class. As a statistical technique, DA is available in many modern statistical packages, such as the SAS and SPSS (Seber, 1984).

Classification can be preformed by using a parametric approach or a nonparametric approach. A parametric classification is suitable only when the within-class distributions of the independent variables approximate a multivariate normal distribution. However, when the distribution within each group does not have any specific distribution, or is assumed to have a distribution other than the multivariate normal distribution, nonparametric methods can be used to derive the needed classification criteria (Szabo, 1995). We used the nonparametric analysis approach in our empirical studies with DA (Berenson et al., 1983).

Estimated density functions for the mutually exclusive groups are computed using a *normal kernel* function on the vectors of the independent variables, Bayesian posterior probabilities of membership in a particular group, and the prior probability distributions of the groups in the fit data set from which the discriminant model is built (Szabo and Khoshgoftaar, 2000). The DA technique was an appropriate candidate to test and validate the effectiveness of the proposed technique, because it provides a direct three-group classification model. Consequently, we were able to compare the two models calibrated using DA, i.e., the three-group model built using DA directly versus the three-group model built using the proposed methodology.

4.2. C4.5 Classification Tree Algorithm

The *C4.5* algorithm is an inductive supervised learning system which employs decision trees to represent a quality model (Quinlan, 1993). As a descendent of another induction program, ID3, it consists of four principal programs: decision tree generator, production rule generator, decision tree interpreter, and production rule interpreter. The algorithm uses these four programs when constructing and evaluating the different classification trees.

Certain pre-processing of data is needed by the algorithm in order for it to build decision trees. Some of these include attribute value description type, predefined discrete classes, and sufficient number of observations for supervised learning. Different tree models can be calibrated by varying certain modeling parameters: minimum node size before further splitting and pruning percentage (Ponnuswamy, 2001; Quinlan, 1993). The classification tree is initially empty and the algorithm begins adding decision and leaf nodes, starting with the root node. This decision tree algorithm was another appropriate candidate to test and validate the proposed technique. Similar to DA, the C4.5 algorithm facilitates a three-group classification.

4.3. Case-Based Reasoning

Case-Based Reasoning (CBR) (Kolodner, 1993), is a modeling technique that finds solutions to new problems based on past experiences, represented by “cases” in a “case library." The case library and the associated retrieval and decision rules constitute a CBR model. Each case in the case library has known attributes and class membership. A CBR system can take advantage of the availability of new or revised information by adding new cases or by removing obsolete cases from the case library. Due to its good scalability, CBR provides fast retrieval even as the size of the case library scales up. CBR systems can be designed to alert users when a new case is outside the bounds of current experience. A theoretical approach and description of CBR is presented in (Kolodner, 1993; Leake, 1996).

In the context of software quality estimation, the working hypothesis of CBR is that a module currently under development is probably of high-risk if a module with similar attributes in an earlier system release (or similar project) was of high-risk. A CBR software quality estimation system utilizes a similarity function to determine the most similar cases (to the current or target case) from the case library. The distances obtained are then evaluated by a solution process algorithm to determine the quality factor (e.x., risk class) for the current case.

In a previous study (Khoshgoftaar and Seliya, 2003), our research group introduced a CBR classification technique to calibrate software metrics-based quality estimation models. The CBR modeling technology has been implemented in our empirical software quality estimation tool, SMART: the Software Measurement Analysis and Reliability Toolkit (Khoshgoftaar et al., 2000a), at the Empirical Software Engineering Laboratory, Florida Atlantic University.

It should be noted that the CBR technology presented in our previous studies does not provide a three-group classification. However, as a computational intelligence technique it provided promising results for calibrating two-group classification models. The application of the proposed three-group technique can be applied to the CBR technology, thus utilizing its effective classification technique.

5. Case Study: Wireless Configuration Systems

The software metrics and fault data for the case study were collected from the initial releases of two Windows©-based embedded software systems. The applications were used primarily for customizing the configuration of wireless telecommunications products. Written in C++, these embedded software applications provide similar functionalities, and contained common source code. The primary difference between the two applications is the type of wireless product that each supports. Both systems comprised of over 1400 source code files and contained more than 27 million lines of code each.

The set of available software metrics is usually determined by pragmatic considerations. A data mining approach is preferred in exploiting software metrics data (Fayyad, 1996), by which a broad set of metrics are analyzed rather than limiting data collection according to predetermined research questions. However, due to practical software engineering issues the number and selection of software metrics for a given software system often depends on the available data collection tools.

The software metrics for this case study were obtained by observing the configuration management systems of the applications. The problem reporting system tracked and recorded problem statuses. Information such as, how many times a source file was inspected prior to system tests, were logged in its database. The software metrics obtained reflected aspects of source files, and consequently, a module in this case study comprised of a source file. Moreover, the software metrics used for modeling purposes was governed by their availability. It is not advocated that other software metrics are not useful. Another software project might collect (depending on availability) and use a different set of software metrics (Basili et al., 1996; Runeson et al., 2001).

The fault data collected for this case study, represent the faults (quality factor) discovered during system tests. Therefore, the classification models predict the class for software modules prior to system tests. This implies that unit testing for the individual program modules has already been performed. Upon preprocessing and cleaning the software data, 1211 modules remained and were used for model calibration. Data preprocessing and cleaning primarily included the removal of observations with missing information or incomplete data. The decision to remove certain modules from the data set was based on our discussion with the development team. Among the 1211 modules considered form modeling, over 66% (809) were observed to have no faults, and the remaining 402 modules had one or more faults.

The five software metrics used for quality prediction are presented in Table 2. The product metrics used are statement metrics for the source files. They primarily indicated the number of lines of source code prior to the coding phase (i.e., auto-generated code) and just before system tests. The process metric, *Ins*, was obtained from problem reporting systems of the embedded software applications, and was obtained by observing the inspections logged in the problem reporting database. The *Ins* metric considered all kinds of inspections for a given program module.

Prior to classification modeling the threshold values that delineate software modules into the appropriate risk-based groups, are selected according to the SQA goals of a given project. Threshold-selection is an important decision since it can affect the usefulness of the classification model. For the software system considered, it was decided (based on our discussions with the development team) that modules with 5 or more faults are considered as Red (high-risk); modules with no faults are considered Green (low-risk); and all others (1 to 4 faults) are considered as Yellow (medium-risk).

Table 2. Software product and process metrics.

Notation	Description
Ins	Number of times the source file was inspected prior to system test release.
BCode	Lines of code for the source file prior to the coding phase. This represents auto-generated code.
SCode	Lines of code for the source file prior to system test release.
Bcomm	Commented lines of code for the source file prior to the coding phase. This represents auto-generated code.
Scomm	Commented lines of code for the source file prior to system test release.

Table 3. Defect density of modules: split one.

	Number of modules	Total faults	Lines of code (*SCode*)	Defect density
Fit Data				
Green	539	0	26635	0.00000
Yellow	211	437	24779	0.01764
Red	57	813	44390	0.01831
Total	807	1250	95804	0.01305
Test Data				
Green	270	0	13130	0.00000
Yellow	106	221	14667	0.01507
Red	28	386	18399	0.02098
Total	404	607	46196	0.01314

An ideal software quality modeling situation would involve software metrics data from multiple software releases or several similar projects, as it provides a more realistic picture. This is because, in such a scenario the test data used for evaluating the calibrated quality estimation model, is independent of the training data set (Khoshgoftaar et al., 2000b). Such an approach simulates the application of the model to the currently under-development software release. However, for the embedded software applications software data from only the initial release was available.

The training and test data sets were defined by applying an impartial data-splitting on the original data set. Two-thirds (807) of the modules were used as fit data, whereas the remaining one-third (404) were used as test data. According to the selected class threshold values, Table 3 summarizes the defect densities of modules for the Red, Yellow, and Green classes. To avoid biased results due to a lucky (or unlucky) data-split, fifty random combinations of the fit (807) and test (404) data sets were obtained from the original data set. Subsequently, models were calibrated and evaluated for each of these 50 combinations. Table 3 presents the fault distributions of the fit and test data sets for only one data split.

6. Results and Analysis

The selection of the preferred classification model is usually dependent on the given software project. For example, in the case of two-group (*fp* and *nfp*) classification models of high-assurance and safety-critical systems, the disparate costs of the misclassifications and the distributions of the two classes may play an important role. Furthermore, since the actual misclassification costs are unknown at the time of modeling, the selected classification model may be sensitive to the misclassification costs. Due to such factors the preferred balance between the two misclassifications is influenced by the needs of a project. For example, in our previous studies with two-group classification models, a preferred balance of equality between the two misclassification error rates has been used (Khoshgoftaar et al., 2000c).

Model selection for a three-group classification problem is affected by similar issues as mentioned above. However, since the main goal of a three-group model is to provide a more focused software quality improvement, additional factors may take priority. For example, in our case studies we utilized the following model-selection strategy when calibrating three-group models using both the direct and indirect modeling approaches.

1. The Type_RG error rate should be zero or as low as possible. This is important because, the management team aims to detect and rectify all the high-risk (Red) modules since they are likely to have the most faults. Among the models with the preferred Type_RG error rate, the one with the lowest Type_GR is selected.

2. The remaining four error rates are preferred as follows:

   - A preferred balance of equality between the Type_GY and Type_YG error rates, with Type_YG as low as possible.

   - A preferred balance of equality between the Type_YR and Type_RY error rates, with Type_RY as low as possible.

However, the model-selection strategy is dependent on the system being modeled and its SQA goals. A balance of equality among the error rates (except Type_GR and Type_RG) was selected because the number of Green modules were disproportionately greater than the number of Yellow. Along the same lines, the number of Yellow modules were disproportionately greater than the number of Red modules. In addition, the misclassification costs, C_{YG} and C_{RY} are (practically speaking) greater than C_{GY} and C_{YR}, respectively.

3. Model-selection for all classification methods (direct and indirect) are based on the *leave-one-out* (LOO) cross-validation technique, i.e., class prediction for the *i*th module (x_i), in the fit data set is based on a classification model calibrated using all, except the *i*th, observations in the fit data set.

As a guidance to practitioners in replicating our empirical case studies, we present a detailed illustration of the model-calibration and model-selection process of the proposed method when applied to the discriminant analysis classification technique. The details of the other two classification techniques are summarized due to paper-size concerns.

6.1. Discriminant Analysis Models

The quality of the classification model obtained by DA is affected by the smoothing parameter used, i.e., λ. When calibrating a direct three-group classification model using DA, we varied λ from 0.001 to 0.1 with an increment of 0.001. The selection of the preferred classification model is based on the error rates obtained from the LOO cross-validation technique. The calibrated direct model was then applied to the test data set.

Table 4. Direct discriminant analysis model number of modules and percentage.

Actual risk group	Predicted risk group Green	 Yellow	 Red	Total
Green	230 (85.19%)	16 (5.93%)	24 (8.89%)	270 (100.0%)
Yellow	20 (18.87%)	52 (49.06%)	34 (32.08%)	106 (100.0%)
Red	0 (0.0%)	6 (21.43%)	22 (78.57%)	28 (100.0%)
Total	250 (61.88%)	74 (18.32%)	80 (19.80%)	404 (100.0%)

The predictive capability of the direct model (using one of the fifty data splits) in terms of its misclassification rates on the test data set, is summarized in Table 4.

We now discuss the application of the proposed algorithm using DA, i.e., calibrating an indirect three-group model. As outlined in Section 3 and Figure 1, three two-group classification models are calibrated, i.e., Model_GR, Model_GY, and Model_YR. The smoothing parameter for these three models were adjusted independently, yielding different models. The λ values for the three two-group models, i.e., Model_GR, Model_GY, & Model_YR, were 0.03, 0.04 and 0.10, respectively. The individual models were selected as discussed below.

For comparative purposes, we present the model calibrated by the indirect method using the same data split representing the direct DA model in Table 4. The predictive capability (404 modules of the test data) of Model_GR is summarized in Table 5. The two middle columns indicate the parsed groups, Test-I and Test-II (see Figure 1). The first column segregates the modules of the Test-I and Test-II groups according to the their actual risk class, i.e., Green, Yellow, and Red. Therefore, the number of modules and percentages presented in the table, indicate the actual module-quality distribution of the two groups.

As per Step 2 of the proposed method, Model_GY is calibrated and applied to the modules of the Test-I group. As shown in Table 6, the model predicts the quality of the Test-I modules as either Green or Yellow_I, i.e., the two middle columns. The first column of the table segregates the modules of the (two) parsed groups of Test-I with respect to their actual risk class. Note that the total number of modules of the Green (243) and Yellow_I (13) groups must be the same as of the Test-I (256) group.

Table 5. Model_GR-Step 1 of indirect method number of modules and percentage.

Actual risk group	Predicted risk group Test_I	 Test_II	Total
Green	230 (85.19%)	40 (14.81%)	270 (100.00%)
Yellow	26 (17.92%)	80 (66.04%)	106 (100.00%)
Red	0 (0.00%)	28 (100.00%)	28 (100.00%)
Total	256 (63.37%)	148 (36.63%)	404 (100.00%)

Table 6. Model_GY-Step 2 of indirect method number of modules and percentage.

	Predicted risk group		
Actual risk group	Green	Yellow_I	Total
Green	224 (97.39%)	6 (2.61%)	230 (100.00%)
Yellow	19 (73.08%)	7 (26.92%)	26 (100.00%)
Red	0 (0.00%)	0 (0.00%)	0 (0.00%)
Total	243 (94.92%)	13 (5.08%)	256 (100.00%)

Following the Step 3 of our algorithm, Model_YR is calibrated and applied to the modules of the Test-II group. As shown in Table 7, the model estimates the quality of the Test-II modules (148) as either Yellow_II (93 modules) or Red (55 modules). Finally, upon combining the models, Model_GY and Model_YR, the indirect three-group DA model is obtained as summarized in Table 8. The predicted classifications, either right or wrong, can be read from the table which can be viewed as a 3×3 matrix. For example, the Type_GR and Type_RG misclassifications are 5.93% and 0.00%, respectively.

During the model selection process for Model_GR, the Type_RG error rate is kept as low as possible, preferably zero. Among the models with the lowest Type_RG, the ones with lower Type_GR are preferred. This is because it is important to keep the ineffective inspections and quality testing to the minimum, i.e., inspecting the Green modules when the resources could be used to improve the Red and Yellow modules. When selecting an appropriate candidate for Model_GY, it was desired that its misclassifications, i.e., Type_GY and Type_YG be balanced with the latter being as low as possible. On the same token, for Model_YR it was desired that the Type_YR and Type_RY be balanced with the latter being as low as possible.

The classification results of the direct and indirect DA models shown above in Tables 4 and 8, represent the first data split of Tables 9 and 10, respectively, which summarize the direct and indirect models calibrated for all the fifty data splits.[4] The relative performances of the direct and indirect models with regards to their misclassification rates is discussed below.

The Type_RG misclassifications of both the models were similar for many split combinations, and they both demonstrated near-perfect quality classifications for most of the data splits. The average Type_RG across the fifty splits for the direct and indirect (proposed) DA models are 0.21% and 0.07%, respectively. In the context of the high-

Table 7. Model_YR-Step 3 of indirect method number of modules and percentage.

	Predicted risk group		
Actual risk group	Yellow_II	Red	Total
Green	24 (60.00%)	16 (40.00%)	40 (100.00%)
Yellow	63 (78.75%)	17 (21.25%)	80 (100.00%)
Red	6 (21.43%)	22 (78.57%)	28 (100.00%)
Total	93 (62.84%)	55 (37.16%)	148 (100.00%)

Table 8. Indirect discriminant analysis model number of modules and percentage.

	Predicted risk group			
Actual risk group	Green	Yellow	Red	Total
Green	224 (82.96%)	30 (11.11%)	16 (5.93%)	270 (100.0%)
Yellow	19 (17.92%)	70 (66.04%)	17 (16.04%)	106 (100.0%)
Red	0 (0.0%)	6 (21.43%)	22 (78.57%)	28 (100.0%)
Total	250 (61.88%)	74 (18.32%)	80 (19.80%)	404 (100.0%)

and low-risk modules, the indirect models have a better classification for the Red modules, as compared to the direct models. The same averages for Type_GR are 10.35% and 6.04%, respectively, i.e., better resource usage by the indirect model. However, the indirect models generally have a higher Type_RY error rate than the direct models. This indicates that in the context of the high- and medium-risk modules, the direct models yield better resource utilization.

As compared to the direct DA models, the indirect (proposed technique) models had fewer numbers of Type_YR and Type_GR misclassifications, indicating lower resource wastage in reviewing the lower risk, i.e., Green and Yellow, modules. In addition, the average Type_YG error rate for the indirect DA models was lower than that of the direct DA models. On the other hand, the Type_GY misclassification rates for the direct models were lower than those of the indirect models, indicating that the proposed technique may tend to increase the inspection effort slightly, i.e., Yellow modules inspected are actually Green.

In the previous two paragraphs and Tables 9 and 10, we observe that with respect to all the error rates, neither modeling method consistently yields better three-group models. The Type_GR, Type_RG, Type_YG, and Type_YR of the indirect models are generally better than those of the direct models. However, its Type_GY and Type_RY is generally higher. So, how does a practitioner select the best model? or what criteria should be observed when choosing the appropriate model?

In addition to the above mentioned difficulty associated in choosing a model based solely on misclassification error rates, we have already pointed out that the cost associated with each of the six errors are different in the real world. To overcome these issues, we use the ECM value of a three-group model as a model-evaluation measure. It is a singular measure, i.e., one can select a model with the lowest ECM, and it incorporates the different costs of misclassifications, i.e., it takes a more realistic and practical approach. Subsequently, by providing the different misclassification costs appropriate to the given organization and/or project, a three-group model that provides the lowest ECM can be selected as the preferred model.

6.2. C4.5 Algorithm and CBR Models

The preferred three-group models calibrated using the C4.5 classification tree algorithm directly and indirectly (using the proposed indirect method) are presented in

Table 9. Discriminant analysis:—Direct method.

Data	Type_RG		Type_RY		Type_YG		Type_YR		Type_GY		Type_GR	
split	Num	%	Num	%	Num	%	Num	%	Num	%	Num	%
1	0	0.00	6	21.43	20	18.87	34	32.08	16	5.93	24	8.89
2	0	0.00	5	17.86	32	30.19	36	33.96	24	8.89	26	9.63
3	0	0.00	4	14.29	24	22.64	31	29.25	23	8.52	33	12.22
4	1	3.45	6	20.69	29	27.10	33	30.84	16	5.97	25	9.33
5	0	0.00	4	14.29	21	20.00	35	33.33	14	5.17	25	9.23
6	0	0.00	7	24.14	28	26.67	33	31.43	16	5.93	25	9.26
7	0	0.00	5	17.24	23	21.90	33	31.43	23	8.52	25	9.26
8	0	0.00	4	14.29	23	21.50	35	32.71	21	7.81	26	9.67
9	0	0.00	5	17.24	23	21.90	38	36.19	30	11.11	25	9.26
10	1	3.57	5	17.86	23	21.30	37	34.26	18	6.72	29	10.82
11	0	0.00	8	27.59	32	31.37	28	27.45	28	10.26	26	9.52
12	1	3.45	2	6.90	29	26.85	38	35.19	18	6.74	30	11.24
13	0	0.00	4	14.29	26	24.53	37	34.91	16	5.93	26	9.63
14	0	0.00	5	17.86	29	27.62	37	35.24	29	10.70	32	11.81
15	0	0.00	6	20.69	17	16.67	34	33.33	22	8.06	30	10.99
16	0	0.00	8	25.81	30	28.57	29	27.62	21	7.84	25	9.33
17	0	0.00	10	34.48	24	21.82	41	37.27	25	9.43	27	10.19
18	0	0.00	6	22.22	25	23.36	29	27.10	21	7.78	20	7.41
19	0	0.00	7	24.14	25	24.04	36	34.62	23	8.49	37	13.65
20	0	0.00	5	17.24	15	14.71	36	35.29	14	5.13	36	13.19
21	0	0.00	5	17.86	25	24.04	35	33.65	15	5.51	27	9.93
22	0	0.00	3	9.68	33	29.46	34	30.36	17	6.51	24	9.20
23	0	0.00	7	22.58	22	20.95	43	40.95	13	4.85	32	11.94
24	0	0.00	4	14.29	23	22.12	34	32.69	21	7.72	37	13.60
25	0	0.00	6	23.08	29	28.43	33	32.35	15	5.43	26	9.42
26	0	0.00	3	10.00	17	15.32	44	39.64	18	6.84	35	13.31
27	0	0.00	7	22.58	29	26.36	33	30.00	24	9.13	20	7.60
28	0	0.00	6	20.00	18	17.31	38	36.54	15	5.56	30	11.11
29	0	0.00	4	13.79	28	26.92	36	34.62	26	9.59	21	7.75
30	0	0.00	10	32.26	20	18.87	35	33.02	27	10.11	26	9.74
31	0	0.00	2	7.69	28	26.92	32	30.77	18	6.57	34	12.41
32	0	0.00	5	17.86	25	23.15	32	29.63	21	7.84	25	9.33
33	0	0.00	6	20.69	21	19.44	35	32.41	17	6.37	32	11.99
34	0	0.00	3	10.71	24	24.00	35	35.00	22	7.97	25	9.06
35	0	0.00	6	21.43	25	23.58	35	33.02	17	6.30	22	8.15
36	0	0.00	5	17.86	23	21.50	34	31.78	21	7.81	32	11.90
37	0	0.00	4	13.79	19	16.96	44	39.29	20	7.60	37	14.07
38	0	0.00	3	11.54	29	28.43	34	33.33	17	6.16	24	8.70
39	0	0.00	6	21.43	18	17.31	34	32.69	19	6.99	33	12.13
40	0	0.00	3	11.11	30	28.04	39	36.45	16	5.93	31	11.48
41	0	0.00	9	33.33	14	14.29	37	37.76	30	10.75	26	9.32
42	0	0.00	4	15.38	26	22.22	44	37.61	16	6.13	27	10.34
43	0	0.00	5	20.00	21	20.39	30	29.13	19	6.88	21	7.61
44	0	0.00	4	13.79	30	27.78	31	28.70	11	4.12	34	12.73
45	0	0.00	8	29.63	23	20.91	32	29.09	21	7.87	28	10.49
46	0	0.00	4	13.79	25	24.04	31	29.81	24	8.86	31	11.44
47	0	0.00	7	20.59	25	23.81	37	35.24	21	7.92	19	7.17
48	0	0.00	5	18.52	13	12.26	39	36.79	25	9.23	29	10.70
49	0	0.00	9	27.27	19	20.21	32	34.04	25	9.03	26	9.39
50	0	0.00	8	28.57	25	22.52	39	35.14	22	8.30	29	10.94
Avg.	0.06	0.21	5.46	19.03	24.1	22.78	35.2	33.30	20.2	7.50	27.9	10.35

Table 10. Discriminant analysis:— Indirect method.

Data	Type_RG		Type_RY		Type_YG		Type_YR		Type_GY		Type_GR	
split	Num	%	Num	%	Num	%	Num	%	Num	%	Num	%
1	0	0.00	6	21.43	19	17.92	17	16.04	30	11.11	16	5.93
2	0	0.00	8	28.57	27	25.47	22	20.75	49	18.15	10	3.70
3	0	0.00	4	14.29	22	20.75	23	21.70	35	12.96	27	10.00
4	0	0.00	7	24.14	17	15.89	21	19.63	45	16.79	12	4.48
5	0	0.00	5	17.86	21	20.00	22	20.95	33	12.18	17	6.27
6	0	0.00	7	24.14	22	20.95	22	20.95	35	12.96	18	6.67
7	0	0.00	7	24.14	20	19.05	25	23.81	48	17.78	13	4.81
8	0	0.00	6	21.43	19	17.76	23	21.50	42	15.61	17	6.32
9	0	0.00	5	17.24	15	14.29	30	28.57	48	17.78	23	8.52
10	0	0.00	7	25.00	22	20.37	24	22.22	40	14.93	15	5.60
11	0	0.00	10	34.48	28	27.45	23	22.55	39	14.29	19	6.96
12	0	0.00	4	13.79	19	17.59	26	24.07	38	14.23	16	5.99
13	0	0.00	11	39.29	23	21.70	16	15.09	47	17.41	8	2.96
14	0	0.00	6	21.43	23	21.90	29	27.62	52	19.19	21	7.75
15	0	0.00	7	24.14	12	11.76	28	27.45	42	15.38	17	6.23
16	0	0.00	8	25.81	27	25.71	18	17.14	43	16.04	15	5.60
17	1	3.45	12	41.38	22	20.00	26	23.64	39	14.72	16	6.04
18	0	0.00	8	29.63	22	20.56	13	12.15	42	15.56	7	2.59
19	0	0.00	10	34.48	22	21.15	27	25.96	54	19.93	16	5.90
20	0	0.00	4	13.79	15	14.71	17	16.67	50	18.32	13	4.76
21	0	0.00	5	17.86	18	17.31	26	25.00	43	15.81	12	4.41
22	0	0.00	8	25.81	29	25.89	16	14.29	39	14.94	14	5.36
23	0	0.00	7	22.58	15	14.29	26	24.76	52	19.40	13	4.85
24	0	0.00	5	17.86	21	20.19	27	25.96	40	14.71	25	9.19
25	0	0.00	6	23.08	26	25.49	25	24.51	40	14.49	14	5.07
26	0	0.00	4	13.33	17	15.32	26	23.42	42	15.97	17	6.46
27	0	0.00	9	29.03	25	22.73	22	20.00	41	15.59	11	4.18
28	0	0.00	6	20.00	13	12.50	26	25.00	36	13.33	22	8.15
29	0	0.00	4	13.79	28	26.92	27	25.96	38	14.02	13	4.80
30	0	0.00	13	41.94	14	13.21	30	28.30	44	16.48	20	7.49
31	0	0.00	6	23.08	24	23.08	19	18.27	50	18.25	13	4.74
32	0	0.00	5	17.86	18	16.67	26	24.07	45	16.79	14	5.22
33	0	0.00	7	24.14	20	18.52	25	23.15	31	11.61	19	7.12
34	0	0.00	3	10.71	21	21.00	25	25.00	43	15.58	17	6.16
35	0	0.00	7	25.00	15	14.15	24	22.64	42	15.56	11	4.07
36	0	0.00	7	25.00	24	22.43	21	19.63	37	13.75	16	5.95
37	0	0.00	7	24.14	12	10.71	29	25.89	50	19.01	19	7.22
38	0	0.00	5	19.23	27	26.47	22	21.57	30	10.87	17	6.16
39	0	0.00	7	25.00	18	17.31	30	28.85	43	15.81	17	6.25
40	0	0.00	3	11.11	24	22.43	25	23.36	44	16.30	19	7.04
41	0	0.00	10	37.04	13	13.27	22	22.45	48	17.20	12	4.30
42	0	0.00	4	15.38	23	19.66	27	23.08	38	14.56	15	5.75
43	0	0.00	7	28.00	15	14.56	23	22.33	40	14.49	14	5.07
44	0	0.00	4	13.79	21	19.44	16	14.81	41	15.36	20	7.49
45	0	0.00	9	33.33	20	18.18	25	22.73	47	17.60	15	5.62
46	0	0.00	7	24.14	23	22.12	16	15.38	40	14.76	19	7.01
47	0	0.00	7	20.59	22	20.95	34	32.38	32	12.08	18	6.79
48	0	0.00	7	25.93	13	12.26	24	22.64	42	15.50	22	8.12
49	0	0.00	9	27.27	17	18.09	27	28.72	39	14.08	26	9.39
50	0	0.00	9	32.14	18	16.22	25	22.52	57	21.51	15	5.66
Avg.	0.02	0.07	6.78	23.7	20.22	19.1	23.76	22.50	42.1	15.61	16.3	6.04

Table 11. C4.5 algorithm:— Direct method.

Data split	Type_RG		Type_RY		Type_YG		Type_YR		Type_GY		Type_GR	
	Num	%	Num	%	Num	%	Num	%	Num	%	Num	%
1	0	0.00	5	17.86	12	11.32	20	18.87	10	3.70	76	28.15
2	1	3.57	5	17.86	22	20.75	34	32.08	13	4.81	50	18.52
3	0	0.00	5	17.86	41	38.68	28	26.42	21	7.78	42	15.56
4	0	0.00	10	34.48	13	12.15	16	14.95	17	6.34	63	23.51
5	0	0.00	7	25.00	21	20.00	19	18.10	8	2.95	78	28.78
6	0	0.00	5	17.24	22	20.95	20	19.05	15	5.56	51	18.89
7	1	3.45	6	20.69	8	7.62	26	24.76	8	2.96	110	40.74
8	1	3.57	5	17.86	16	14.95	23	21.50	14	5.20	53	19.70
9	0	0.00	3	10.34	10	9.52	35	33.33	21	7.78	107	39.63
10	1	3.57	2	7.14	24	22.22	27	25.00	11	4.10	58	21.64
11	0	0.00	7	24.14	3	2.94	16	15.69	13	4.76	150	54.95
12	1	3.45	7	24.14	13	12.04	21	19.44	6	2.25	71	26.59
13	0	0.00	6	21.43	15	14.15	23	21.70	25	9.26	60	22.22
14	0	0.00	3	10.71	16	15.24	23	21.90	10	3.69	76	28.04
15	0	0.00	7	24.14	12	11.76	20	19.61	12	4.40	135	49.45
16	0	0.00	6	19.35	10	9.52	25	23.81	4	1.49	83	30.97
17	0	0.00	13	44.83	7	6.36	12	10.91	6	2.26	103	38.87
18	0	0.00	5	18.52	13	12.15	29	27.10	22	8.15	70	25.93
19	1	3.45	8	27.59	23	22.12	27	25.96	12	4.43	47	17.34
20	0	0.00	7	24.14	12	11.76	18	17.65	10	3.66	68	24.91
21	0	0.00	6	21.43	15	14.42	50	48.08	30	11.03	39	14.34
22	0	0.00	8	25.81	16	14.29	19	16.96	14	5.36	61	23.37
23	0	0.00	10	32.26	25	23.81	35	33.33	23	8.58	37	13.81
24	0	0.00	7	25.00	11	10.58	18	17.31	16	5.88	95	34.93
25	0	0.00	9	34.62	20	19.61	13	12.75	10	3.62	54	19.57
26	1	3.33	6	20.00	14	12.61	24	21.62	11	4.18	64	24.33
27	5	16.13	6	19.35	18	16.36	14	12.73	2	0.76	62	23.57
28	0	0.00	10	33.33	17	16.35	19	18.27	14	5.19	58	21.48
29	0	0.00	7	24.14	14	13.46	18	17.31	15	5.54	71	26.20
30	1	3.23	7	22.58	24	22.64	28	26.42	13	4.87	55	20.60
31	0	0.00	3	11.54	17	16.35	18	17.31	4	1.46	63	22.99
32	0	0.00	4	14.29	12	11.11	31	28.70	21	7.84	81	30.22
33	0	0.00	7	24.14	5	4.63	17	15.74	9	3.37	144	53.93
34	0	0.00	10	35.71	7	7.00	17	17.00	8	2.90	98	35.51
35	1	3.57	10	35.71	22	20.75	16	15.09	9	3.33	58	21.48
36	0	0.00	5	17.86	18	16.82	25	23.36	13	4.83	58	21.56
37	1	3.45	4	13.79	17	15.18	28	25.00	7	2.66	68	25.86
38	1	3.85	6	23.08	9	8.82	12	11.76	12	4.35	76	27.54
39	0	0.00	8	28.57	11	10.58	26	25.00	11	4.04	125	45.96
40	0	0.00	1	3.70	17	15.89	35	32.71	35	12.96	84	31.11
41	0	0.00	5	18.52	17	17.35	19	19.39	9	3.23	90	32.26
42	0	0.00	3	11.54	24	20.51	23	19.66	11	4.21	41	15.71
43	0	0.00	5	20.00	6	5.83	22	21.36	6	2.17	124	44.93
44	0	0.00	4	13.79	21	19.44	18	16.67	18	6.74	76	28.46
45	1	3.70	1	3.70	35	31.82	27	24.55	13	4.87	29	10.86
46	0	0.00	5	17.24	17	16.35	19	18.27	9	3.32	90	33.21
47	0	0.00	6	17.65	27	25.71	24	22.86	9	3.40	57	21.51
48	1	3.70	2	7.41	7	6.60	27	25.47	13	4.80	115	42.44
49	2	6.06	3	9.09	23	24.47	34	36.17	16	5.78	51	18.41
50	2	7.14	2	7.14	32	28.83	28	25.23	13	4.91	34	12.83
Avg.	0.44	1.50	5.84	20.37	16.62	15.69	23.3	22.08	13.04	4.83	74.2	27.47

Table 12. C4.5 algorithm:— Indirect method.

Data	Type_RG		Type_RY		Type_YG		Type_YR		Type_GY		Type_GR	
Split	Num	%	Num	%	Num	%	Num	%	Num	%	Num	%
1	0	0.00	5	17.86	14	13.21	19	17.92	10	3.70	52	19.26
2	1	3.57	8	28.57	10	9.43	18	16.98	6	2.22	70	25.93
3	0	0.00	7	25.00	14	13.21	8	7.55	8	2.96	57	21.11
4	0	0.00	8	27.59	9	8.41	17	15.89	5	1.87	67	25.00
5	1	3.57	2	7.14	47	44.76	17	16.19	7	2.58	37	13.65
6	0	0.00	4	13.79	23	21.90	18	17.14	13	4.81	57	21.11
7	1	3.45	4	13.79	18	17.14	20	19.05	13	4.81	53	19.63
8	0	0.00	8	28.57	18	16.82	24	22.43	10	3.72	56	20.82
9	0	0.00	4	13.79	14	13.33	27	25.71	15	5.56	51	18.89
10	0	0.00	6	21.43	22	20.37	25	23.15	11	4.10	34	12.69
11	0	0.00	8	27.59	16	15.69	13	12.75	13	4.76	61	22.34
12	1	3.45	6	20.69	16	14.81	21	19.44	8	3.00	54	20.22
13	0	0.00	8	28.57	12	11.32	18	16.98	15	5.56	45	16.67
14	0	0.00	4	14.29	24	22.86	21	20.00	8	2.95	53	19.56
15	1	3.45	6	20.69	13	12.75	15	14.71	11	4.03	50	18.32
16	0	0.00	5	16.13	16	15.24	21	20.00	8	2.99	50	18.66
17	1	3.45	8	27.59	19	17.27	19	17.27	6	2.26	42	15.85
18	0	0.00	2	7.41	7	6.54	28	26.17	22	8.15	79	29.26
19	2	6.90	7	24.14	19	18.27	24	23.08	14	5.17	46	16.97
20	0	0.00	7	24.14	7	6.86	13	12.75	8	2.93	105	38.46
21	0	0.00	6	21.43	6	5.77	34	32.69	26	9.56	87	31.99
22	0	0.00	4	12.90	24	21.43	32	28.57	19	7.28	29	11.11
23	0	0.00	6	19.35	15	14.29	33	31.43	20	7.46	43	16.04
24	1	3.57	7	25.00	16	15.38	17	16.35	12	4.41	55	20.22
25	0	0.00	6	23.08	16	15.69	21	20.59	13	4.71	44	15.94
26	0	0.00	10	33.33	11	9.91	18	16.22	13	4.94	54	20.53
27	2	6.45	7	22.58	26	23.64	19	17.27	9	3.42	43	16.35
28	0	0.00	8	26.67	8	7.69	24	23.08	16	5.93	57	21.11
29	0	0.00	2	6.90	17	16.35	18	17.31	7	2.58	40	14.76
30	0	0.00	7	22.58	16	15.09	24	22.64	15	5.62	56	20.97
31	0	0.00	3	11.54	9	8.65	21	20.19	8	2.92	62	22.63
32	1	3.57	4	14.29	17	15.74	23	21.30	9	3.36	46	17.16
33	0	0.00	8	27.59	22	20.37	20	18.52	14	5.24	51	19.10
34	1	3.57	3	10.71	24	24.00	19	19.00	13	4.71	42	15.22
35	0	0.00	11	39.29	12	11.32	15	14.15	12	4.44	57	21.11
36	1	3.57	3	10.71	20	18.69	19	17.76	10	3.72	45	16.73
37	0	0.00	6	20.69	16	14.29	19	16.96	7	2.66	59	22.43
38	0	0.00	7	26.92	12	11.76	10	9.80	11	3.99	45	16.30
39	0	0.00	7	25.00	16	15.38	19	18.27	5	1.84	58	21.32
40	0	0.00	4	14.81	11	10.28	17	15.89	22	8.15	58	21.48
41	0	0.00	10	37.04	14	14.29	25	25.51	9	3.23	47	16.85
42	0	0.00	2	7.69	9	7.69	34	29.06	15	5.75	93	35.63
43	0	0.00	4	16.00	7	6.80	21	20.39	6	2.17	86	31.16
44	0	0.00	6	20.69	17	15.74	19	17.59	15	5.62	46	17.23
45	1	3.70	3	11.11	14	12.73	21	19.09	7	2.62	51	19.10
46	0	0.00	8	27.59	9	8.65	17	16.35	9	3.32	76	28.04
47	0	0.00	4	11.76	13	12.38	28	26.67	11	4.15	59	22.26
48	0	0.00	3	11.11	9	8.49	27	25.47	30	11.07	49	18.08
49	0	0.00	7	21.21	6	6.38	21	22.34	11	3.97	65	23.47
50	0	0.00	5	17.86	22	19.82	20	18.02	12	4.53	51	19.25
Avg.	0.3	1.05	5.8	20.12	15.4	14.58	20.8	19.67	11.9	4.43	55.5	20.56

Tables 11 and 12, respectively. These tables indicate the predictive capabilities of the models, i.e., when applied to the test data set. Observing the misclassification error rates of all the fifty data splits as a whole, we observe that the indirect models provide lower and more consistent Type_RG error rates. The average Type_RG error across the fifty data splits for the direct and indirect models are 1.5% and 1.05%, respectively, i.e., indirect models are better. The average Type_GR error rate of the two respective models are 27.47% and 20.56%, indicating that the proposed technique will result in less resources being used to inspect the Green modules.

The application of the proposed method on our previously developed two-group classification technique using CBR (Khoshgoftaar and Seliya, 2003), yielded models as shown in Table 13. Similar to that shown for the C4.5 technique, the table provides the predictive capabilities of the classification models. We only calibrate the indirect three-group models using CBR, because a direct three-group model using CBR is practically not feasible, i.e., practically incomprehensible and requires great computational effort. The average Type_RG and Type_GR error rates across the fifty data splits for the three-group CBR models are 0.14% and 6.04%, respectively.

6.3. Expected Costs of Misclassifications

The costs associated with the six misclassification errors are clustered into two groups. The first group consists of $\{C_{YR}, C_{GY}, C_{GR}\}$, i.e., a lower-risk module being flagged as a higher-risk module. This group implies ineffective program reviews and wasted software testing efforts. The second group consists of $\{C_{RG}, C_{RY}, C_{YG}\}$, indicating the corrective cost (and effort) needed due to the missed opportunities for fixing problems prior to operations. The costs of the first group are practically speaking, lower than those of the second group. From a software project management point of view, among the errors of the first group the Type_GY error is the least expensive, while the Type_GR error is the most taxing. Among the errors of the second group, the Type_RG is the most expensive, whereas the Type_YG is the least expensive.

Practical usefulness of a risk-based classification model is measured by its cost-effective quality improvement aspect (see Section 2). The models calibrated in this study are evaluated based on their ECM values with respect to different likely cost values. Since the actual cost values are unknown at the time of modeling, such an empirical approach provides evidence regarding a no-surprise software quality and reliability engineering. In the case of the lower cost group, we selected three realistically-proportional cost values: $\{C_{YR}, C_{GY}, C_{GR}\} = \{2,1,3\}$, $\{3,1,4\}$, or $\{4,1,5\}$. For each of these combination patterns, the costs of the other group were varied to encompass a wide spectrum of possible values. However, due to space considerations, we present the results for only the $\{C_{YR}, C_{GY}, C_{GR}\} = \{2,1,3\}$ combination pattern of the lower cost group. The empirical results of the other two cost combination patterns can be accessed through the internet at http://www.cse.fau.edu/~taghi/esej_threeclass.html, which presents the necessary tabulated results.

The ECM values of the direct and indirect three-group models calibrated using DA are presented in Table 14. A row in the tables presents, for a given cost combination, the

Table 13. Case-based reasoning:— Indirect method.

Data	Type_RG		Type_RY		Type_YG		Type_YR		Type_GY		Type_GR	
split	Num	%	Num	%	Num	%	Num	%	Num	%	Num	%
1	0	0.00	8	28.57	9	8.49	23	21.70	39	14.44	15	5.56
2	0	0.00	7	25.00	12	11.32	23	21.70	52	19.26	17	6.30
3	0	0.00	4	14.29	15	14.15	27	25.47	43	15.93	19	7.04
4	0	0.00	10	34.48	12	11.21	19	17.76	46	17.16	10	3.73
5	0	0.00	7	25.00	17	16.19	20	19.05	36	13.28	14	5.17
6	0	0.00	7	24.14	13	12.38	23	21.90	47	17.41	21	7.78
7	0	0.00	9	31.03	10	9.52	24	22.86	56	20.74	18	6.67
8	0	0.00	8	28.57	12	11.21	22	20.56	52	19.33	19	7.06
9	0	0.00	6	20.69	12	11.43	31	29.52	55	20.37	21	7.78
10	0	0.00	8	28.57	18	16.67	28	25.93	51	19.03	15	5.60
11	0	0.00	9	31.03	9	8.82	18	17.65	62	22.71	11	4.03
12	0	0.00	6	20.69	13	12.04	26	24.07	61	22.85	10	3.75
13	0	0.00	6	21.43	9	8.49	23	21.70	34	12.59	20	7.41
14	0	0.00	4	14.29	17	16.19	25	23.81	52	19.19	21	7.75
15	0	0.00	7	24.14	9	8.82	24	23.53	52	19.05	16	5.86
16	0	0.00	8	25.81	20	19.05	14	13.33	42	15.67	6	2.24
17	0	0.00	12	41.38	17	15.45	20	18.18	46	17.36	16	6.04
18	0	0.00	4	14.81	8	7.48	21	19.63	42	15.56	14	5.19
19	0	0.00	6	20.69	9	8.65	31	29.81	48	17.71	21	7.75
20	0	0.00	9	31.03	10	9.80	25	24.51	43	15.75	20	7.33
21	0	0.00	2	7.14	12	11.54	28	26.92	44	16.18	24	8.82
22	1	3.23	7	22.58	16	14.29	29	25.89	45	17.24	10	3.83
23	0	0.00	8	25.81	12	11.43	24	22.86	47	17.54	10	3.73
24	0	0.00	7	25.00	11	10.58	28	26.92	48	17.65	25	9.19
25	0	0.00	6	23.08	9	8.82	20	19.61	40	14.49	14	5.07
26	0	0.00	8	26.67	13	11.71	28	25.23	38	14.45	24	9.13
27	0	0.00	7	22.58	20	18.18	23	20.91	44	16.73	10	3.80
28	0	0.00	6	20.00	8	7.69	22	21.15	54	20.00	14	5.19
29	0	0.00	6	20.69	15	14.42	24	23.08	54	19.93	13	4.80
30	0	0.00	14	45.16	9	8.49	24	22.64	49	18.35	16	5.99
31	0	0.00	4	15.38	9	8.65	21	20.19	57	20.80	14	5.11
32	0	0.00	4	14.29	16	14.81	19	17.59	34	12.69	15	5.60
33	0	0.00	6	20.69	13	12.04	24	22.22	45	16.85	20	7.49
34	0	0.00	5	17.86	11	11.00	25	25.00	49	17.75	16	5.80
35	0	0.00	7	25.00	12	11.32	22	20.75	38	14.07	16	5.93
36	0	0.00	7	25.00	19	17.76	29	27.10	44	16.36	20	7.43
37	0	0.00	5	17.24	12	10.71	27	24.11	57	21.67	12	4.56
38	0	0.00	5	19.23	11	10.78	21	20.59	35	12.68	20	7.25
39	0	0.00	6	21.43	9	8.65	31	29.81	59	21.69	12	4.41
40	0	0.00	3	11.11	16	14.95	26	24.30	49	18.15	25	9.26
41	0	0.00	9	33.33	7	7.14	22	22.45	60	21.51	8	2.87
42	1	3.85	2	7.69	18	15.38	24	20.51	37	14.18	17	6.51
43	0	0.00	5	20.00	8	7.77	26	25.24	54	19.57	17	6.16
44	0	0.00	8	27.59	18	16.67	18	16.67	47	17.60	12	4.49
45	0	0.00	8	29.63	9	8.18	25	22.73	48	17.98	22	8.24
46	0	0.00	6	20.69	17	16.35	18	17.31	42	15.50	26	9.59
47	0	0.00	7	20.59	11	10.48	30	28.57	30	11.32	14	5.28
48	0	0.00	6	22.22	4	3.77	22	20.75	62	22.88	12	4.43
49	0	0.00	8	24.24	8	8.51	23	24.47	57	20.58	18	6.50
50	0	0.00	6	21.43	16	14.41	31	27.93	40	15.09	15	5.66
Avg.	0.04	0.14	6.7	23.18	12.4	11.68	24.02	22.72	47.32	17.54	16.3	6.04

Table 14. Discriminant analysis: ECM values.

	Cost of misclassifications						Indirect DA		Direct DA		Statistics	
#	C_{RG}	C_{RY}	C_{YG}	C_{YR}	C_{GY}	C_{GR}	ECM	SD	ECM	SD	Z	p
1	10	6	5	2	1	3	0.6943	0.0650	0.8124	0.0581	16.80	0.00
2	10	7	5	2	1	3	0.7111	0.0676	0.8259	0.0591	16.08	0.00
3	10	8	6	2	1	3	0.7779	0.0774	0.8991	0.0682	15.23	0.00
4	12	7	5	2	1	3	0.7112	0.0678	0.8262	0.0593	15.94	0.00
5	12	8	6	2	1	3	0.7780	0.0776	0.8994	0.0684	15.10	0.00
6	12	10	6	2	1	3	0.8116	0.0839	0.9264	0.0713	13.68	0.00
7	15	7	5	2	1	3	0.7113	0.0682	0.8267	0.0596	15.70	0.00
8	15	8	6	2	1	3	0.7782	0.0779	0.8999	0.0687	14.91	0.00
9	15	10	8	2	1	3	0.9118	0.0991	1.0462	0.0888	13.41	0.00
10	20	10	6	2	1	3	0.8120	0.0848	0.9276	0.0722	13.25	0.00
11	20	12	8	2	1	3	0.9456	0.1058	1.0740	0.0921	12.07	0.00
12	20	15	10	2	1	3	1.0961	0.1314	1.2338	0.1150	10.66	0.00
13	25	15	8	2	1	3	0.9962	0.1171	1.1152	0.0983	10.28	0.00
14	25	15	10	2	1	3	1.0963	0.1320	1.2346	0.1155	10.53	0.00
15	30	10	5	2	1	3	0.7624	0.0801	0.8695	0.0666	12.48	0.00
16	30	15	10	2	1	3	1.0966	0.1326	1.2353	0.1161	10.39	0.00
17	30	20	15	2	1	3	1.4307	0.1908	1.6011	0.1718	9.07	0.00
18	40	25	15	2	1	3	1.5151	0.2086	1.6702	0.1814	7.54	0.00
19	40	30	10	2	1	3	1.3488	0.1957	1.4395	0.1564	4.87	0.00
20	50	25	10	2	1	3	1.2654	0.1745	1.3734	0.1417	6.27	0.00
21	50	30	15	2	1	3	1.5996	0.2287	1.7393	0.1934	6.20	0.00
22	80	20	15	2	1	3	1.4332	0.1968	1.6086	0.1795	8.21	0.00
23	80	40	20	2	1	3	2.0191	0.3090	2.1771	0.2629	5.10	0.00
24	80	50	20	2	1	3	2.1869	0.3508	2.3123	0.2897	3.64	0.00
25	100	25	10	2	1	3	1.2679	0.1823	1.3808	0.1507	5.63	0.00
26	100	50	25	2	1	3	2.4382	0.3883	2.6135	0.3317	4.50	0.00
27	100	60	40	2	1	3	3.3567	0.5472	3.6435	0.4912	5.27	0.00
28	100	75	50	2	1	3	4.1090	0.6844	4.4427	0.6153	4.95	0.00
29	120	40	20	2	1	3	2.0211	0.3145	2.1831	0.2689	4.91	0.00
30	120	70	50	2	1	3	4.0260	0.6704	4.3781	0.6092	5.27	0.00
31	150	25	10	2	1	3	1.2703	0.1913	1.3883	0.1647	4.99	0.00
32	150	50	10	2	1	3	1.6899	0.3120	1.7261	0.2446	1.14	0.13
33	150	80	50	2	1	3	4.1953	0.7068	4.5177	0.6296	4.54	0.00
34	150	100	50	2	1	3	4.5310	0.7807	4.7880	0.6724	3.32	0.00
35	200	50	15	2	1	3	1.9426	0.3411	2.0318	0.2773	2.37	0.01
36	200	70	30	2	1	3	3.0290	0.5201	3.1969	0.4375	3.09	0.00
37	200	90	50	2	1	3	4.3656	0.7487	4.6603	0.6555	3.86	0.00
38	200	100	80	2	1	3	6.0350	1.0451	6.5850	0.9683	5.25	0.00
39	200	150	50	2	1	3	5.3726	1.0027	5.4712	0.8230	1.00	0.16
40	200	150	100	2	1	3	7.8750	1.3820	8.4539	1.2473	4.24	0.00

mean ECM and its standard deviation (SD) across the models calibrated for the fifty data splits. A quick inspection indicates that the indirect method yielded lower ECM values than the direct method. To investigate if the improvement was of any significance, we performed a pairwise Z-test with the null hypothesis that, for a given technique, the direct three-group method yields “lower than or equal to” ECM values than the indirect method.[5] The p-values indicated that for all 120 (except three cases) cost combinations, the proposed method yielded better DA models at a significance of 1% or lower. In the cases of the three exceptions, the proposed method was better at a significance of less than 20%.

Table 15. C4.5 decision tree and CBR models: ECM values.

	C4.5 Decision tree							
	Indirect C4.5		Direct C4.5		Statistics		CBR model	
#	ECM	SD	ECM	SD	Z	p	ECM	SD
1	0.6131	0.0755	0.6992	0.0875	5.75	0.00	0.6104	0.0611
2	0.6273	0.0756	0.7137	0.0872	5.86	0.00	0.6269	0.0629
3	0.6798	0.0877	0.7693	0.0989	5.23	0.00	0.6741	0.0707
4	0.6288	0.0764	0.7158	0.0874	5.90	0.00	0.6271	0.0630
5	0.6813	0.0886	0.7714	0.0993	5.26	0.00	0.6743	0.0708
6	0.7098	0.0900	0.8003	0.0999	5.39	0.00	0.7073	0.0756
7	0.6310	0.0778	0.7191	0.0881	5.94	0.00	0.6274	0.0631
8	0.6835	0.0901	0.7747	0.1001	5.30	0.00	0.6746	0.0709
9	0.7885	0.1181	0.8859	0.1284	4.31	0.00	0.7690	0.0882
10	0.7157	0.0942	0.8091	0.1026	5.50	0.00	0.7081	0.0759
11	0.8207	0.1221	0.9202	0.1311	4.45	0.00	0.8024	0.0935
12	0.9399	0.1533	1.0459	0.1633	3.78	0.00	0.9133	0.1150
13	0.8672	0.1287	0.9691	0.1365	4.60	0.00	0.8524	0.1028
14	0.9436	0.1563	1.0513	0.1659	3.81	0.00	0.9138	0.1153
15	0.6850	0.0893	0.7788	0.0983	6.28	0.00	0.6784	0.0713
16	0.9473	0.1596	1.0568	0.1692	3.83	0.00	0.9143	0.1156
17	1.2097	0.2388	1.3348	0.2532	2.85	0.00	1.1501	0.1647
18	1.2884	0.2506	1.4179	0.2638	2.96	0.00	1.2336	0.1806
19	1.1686	0.1973	1.2845	0.2066	3.97	0.00	1.1625	0.1733
20	1.1048	0.1904	1.2231	0.2004	4.07	0.00	1.0811	0.1522
21	1.3671	0.2653	1.5011	0.2785	3.05	0.00	1.3170	0.1988
22	1.2468	0.2766	1.3892	0.3008	2.93	0.00	1.1551	0.1698
23	1.7231	0.3735	1.8840	0.3947	2.63	0.00	1.6383	0.2678
24	1.8656	0.3949	2.0286	0.4155	2.66	0.00	1.8031	0.3079
25	1.1419	0.2336	1.2776	0.2648	3.84	0.00	1.0860	0.1565
26	2.0716	0.4754	2.2560	0.5035	2.37	0.01	1.9586	0.3367
27	2.7874	0.7097	3.0177	0.7531	1.84	0.03	2.5838	0.4702
28	3.3835	0.8800	3.6459	0.9341	1.68	0.05	3.1380	0.5885
29	1.7528	0.4054	1.9276	0.4390	2.68	0.00	1.6422	0.2712
30	3.3270	0.8876	3.5954	0.9442	1.70	0.04	3.0576	0.5744
31	1.1790	0.2863	1.3320	0.3504	3.38	0.00	1.0910	0.1645
32	1.5354	0.3474	1.6934	0.3979	3.58	0.00	1.5031	0.2724
33	3.4919	0.9200	3.7726	0.9782	1.76	0.04	3.2254	0.6085
34	3.7770	0.9521	4.0617	1.0077	1.80	0.04	3.5551	0.6805
35	1.7637	0.4402	1.9536	0.5152	3.01	0.00	1.6615	0.2945
36	2.6221	0.6606	2.8598	0.7206	2.32	0.01	2.4516	0.4513
37	3.6716	0.9708	3.9716	1.0372	1.85	0.03	3.3952	0.6469
38	4.9607	1.4379	5.3503	1.5370	1.50	0.07	4.4808	0.8919
39	4.5270	1.1094	4.8390	1.1757	1.88	0.03	4.3843	0.8957
40	6.4379	1.7987	6.8959	1.9140	1.44	0.08	5.9190	1.1887

The mean ECM values of the direct and indirect models calibrated using the C4.5 decision-tree algorithm are presented in Table 15. Each of the 40 rows in Table 15 respectively correspond to the rows in Table 14, i.e., the cost combinations for a particular row in Table 15 can be obtained by referring to the corresponding (same #) row of Table 14. For example, the cost combination of row 20 in Table 15 is $\{C_{YR} = 50, C_{GY} = 25, C_{GR} = 10, C_{YR} = 2, C_{GY} = 1, C_{GR} = 3\}$, which is obtained by observing the cost combination of row 20 in Table 14. We recall that for each model, the ECM values are computed for the three cost combination patterns discussed earlier; however, the results of only one combination are presented in this paper. As noted earlier, the results of the other two cost combinations are available at http://www.cse.fau.edu/~taghi/esej_threeclass.html.

In the context of Table 15, we observe that the proposed method (using C4.5) yielded significantly better (p-values of 8% or lower) ECM values than the corresponding direct method for all cost combinations. This observation is similar to the one made for DA. Therefore, for both the DA and C4.5 techniques the proposed method yielded significantly improved three-group models. The table also shows the mean ECM values of the three-group models calibrated using CBR, i.e., the models obtained by applying the proposed technique. As mentioned earlier, a direct three-group model using CBR though possible, is practically not feasible. The incomprehensibility associated with calibrating, selecting, and evaluating a direct model may require an exceptional amount of computational effort. Furthermore, to our best knowledge there has not been any published work on direct three-group models with CBR.

6.4. Ranking the Three-Group Models

In this study, we have presented the results of three-group classification models calibrated with five different approaches, i.e., DA (direct and indirect), C4.5 (direct and indirect), and CBR (indirect). Due to the multiple techniques explored, we were interested in answering practical questions such as, "how are these models ranked relative to each other?" and "are their mean ECM values significantly different from each other?" Consequently, we performed a pairwise Z-test between the different models, and the obtained p-values were recorded.

Table 16 presents the p-values of the pairwise significance tests performed (the row index (#) in the table respectively corresponds to the cost combinations of Table 14). Comparisons shown in the last five columns can be read as "A vs. B" with the null hypothesis that A's ECM value is "greater than or equal to" B's ECM. The letters 'I' and 'D' shown next to the techniques indicate respectively, the model calibrated using the proposed indirect method and that obtained by applying the classification technique directly (if available). It has already been discussed earlier that in the case of discriminant analysis and C4.5, the proposed indirect method yielded significantly lower ECM values, i.e., better cost-effective software quality control, as compared to the direct methods of the respective techniques. Therefore, we can conclude that C4.5 (I) > C4.5 (D) and DA (I) > DA (D), where '>' implies that the left hand side is always significantly better than the right hand side.

Table 16. Pairwise comparisons: *p*-values.

#	C4.5 (I) vs. C4.5 (D)	DA (I) vs. DA (D)	C4.5 (I) vs. CBR (I)	CBR (I) vs. C4.5 (D)	C4.5 (D) vs. DA (I)
1	0.00	0.00	0.41	0.00	0.63
2	0.00	0.00	0.49	0.00	0.57
3	0.00	0.00	0.34	0.00	0.30
4	0.00	0.00	0.45	0.00	0.62
5	0.00	0.00	0.30	0.00	0.35
6	0.00	0.00	0.43	0.00	0.26
7	0.00	0.00	0.38	0.00	0.70
8	0.00	0.00	0.26	0.00	0.42
9	0.00	0.00	0.12	0.00	0.12
10	0.00	0.00	0.29	0.00	0.43
11	0.00	0.00	0.14	0.00	0.13
12	0.00	0.00	0.10	0.00	0.04
13	0.00	0.00	0.21	0.00	0.12
14	0.00	0.00	0.08	0.00	0.06
15	0.00	0.00	0.31	0.00	0.85
16	0.00	0.00	0.06	0.00	0.08
17	0.00	0.00	0.03	0.00	0.01
18	0.00	0.00	0.04	0.00	0.01
19	0.00	0.00	0.41	0.00	0.04
20	0.00	0.00	0.18	0.00	0.10
21	0.00	0.00	0.07	0.00	0.02
22	0.00	0.00	0.01	0.00	0.18
23	0.00	0.00	0.04	0.00	0.02
24	0.00	0.00	0.11	0.00	0.01
25	0.00	0.00	0.04	0.00	0.60
26	0.01	0.00	0.03	0.00	0.01
27	0.03	0.00	0.01	0.00	0.00
28	0.05	0.00	0.01	0.00	0.00
29	0.00	0.00	0.02	0.00	0.09
30	0.04	0.00	0.01	0.00	0.00
31	0.00	0.00	0.01	0.00	0.89
32	0.00	0.13	0.26	0.00	0.52
33	0.04	0.00	0.01	0.00	0.00
34	0.04	0.00	0.03	0.00	0.00
35	0.00	0.01	0.05	0.00	0.56
36	0.01	0.00	0.02	0.00	0.07
37	0.03	0.00	0.01	0.00	0.01
38	0.07	0.00	0.00	0.00	0.00
39	0.03	0.16	0.16	0.00	0.00
40	0.08	0.00	0.01	0.00	0.00

Upon examining the 4th column of Table 16, i.e., C4.5(I) vs. CBR(I), we observe that for many cost combinations, C4.5(I) performed significantly better, whereas, for some (noticeable) other combinations CBR(I) performed better. Hence, we conclude that C4.5(I) $\simeq$ CBR(I), indicating that though C4.5(I) had more cases in its favor, CBR(I) provided better results in some other cases. The 5th column of the table, i.e., CBR(I) vs. C4.5(D), suggests that the CBR (I) method is significantly better than C4.5 (D) for all cases, i.e., “CBR (I) > C4.5 (D).” Combining the last two observations we obtain the partial order, “C4.5 (I) $\simeq$ CBR (I) > C4.5 (D).”

The last column of Table 16 indicates that though for many cases C4.5 (D) yielded better ECM values than DA (I), there are some noticeable cases in which the converse is true. To indicate the same, we obtain the relation “C4.5 (D) $\simeq$ DA (I),” which implies that among the two methods there is no clear winner. However, since we have already determined that “DA (I) > DA (D),” we can obtain the partial order, “C4.5 (D)$\simeq$DA (I) > DA (D).” The final rank-order of the different methods can be obtained by combining the (above) two partial orders, i.e., C4.5 (I) $\simeq$ CBR (I) > C4.5 (D) $\simeq$ DA (I) > DA (D). Therefore, in the context of the case study presented, the C4.5 (I) models generally yielded best results, whereas the DA (D) models consistently performed poorly. Similar performance results were observed with other case studies as well (Bhupathiraju, 2002; Song, 2001).

7. Alternate Three-Group Classification Method

In this section, we present an alternate three-group classification method as proposed by an anonymous reviewer. Similar to our proposed approach, this method also proposes an indirect approach for grouping modules into three classes with the use of any two-group classification technique. The algorithm involves the calibration of a two-group classification model twice in order to classify modules into three groups. Our aim of presenting this alternate indirect three-group method is to evaluate its performance relative to the proposed indirect three-group classification method. Hence, we only present comparative results for the two indirect methods in the context of the three two-group classification techniques.

7.1. Algorithm of Alternate Method

Assume a fit data set which has its software modules delineated into the Red, Green, and Yellow groups according to Equation (4). The following steps illustrate the application of the alternate three-group method.

1. Using the modules of the fit data set, calibrate a two-group classification model (called Model_I) that classifies the modules as either Red (R) or Not-Red (NR). Apply this fitted model to parse the test data set into two classes, and denote the groups as R_I and NR_I, respectively.

2. Using the modules of the fit data set, calibrate a two-group classification model (called Model_II) that classifies the modules as either Green (G) or Not-Green (NG).

3. Apply Model_II to parse the R_I group obtained in Step 1 into two sub-groups, and denote them as R_I_G and R_I_NG. Similarly, apply Model_II to parse the NR_I

group obtained in Step 1 into two sub-groups, and denote them as NR_I_G and NR_I_NG.

4. Obtain the three-group classification of the test data set as follows. Modules that are classified as R_I_NG will be considered Red. Modules that are classified as NR_I_G will be considered Green. All the remaining modules, i.e., those classified as either R_I_G or NR_I_NG, will be considered Yellow. At this point, all the modules in the test data set have been classified into one of the three risk-based classes.

7.2. Results and Analysis

The model selection criteria for the alternate three-group method is the same (see Section 6) as the one we have used for calibrating the other three-group classification models in this paper. In the context of the DA, C4.5, and CBR classification techniques, we calibrated indirect three-group classification models using both the proposed method (denoted as PrM) and the alternate method (denoted as AlM). Indirect three-group models were calibrated for 10 data splits as compared to building models for 50 data splits. The data splits used for comparison of the PrM and AlM models correspond to rows {5, 10, 15, . . . , 45, 50} of Table 9 through Table 13.

The comparative performances with respect to the mean (of the 10 data splits) ECM values of the two indirect approaches are shown in Table 17, which represents only one of the three cost combination patterns that were considered in our study. The first column of the table indicates the indexes for 40 cost combinations (for the $\{C_{YR}, C_{GY}, C_{GR}\} = \{2, 1, 3\}$ cost combination pattern) used to compute the ECM values. The p-values shown are those of (for a given classification technique) the pairwise t-test performed to investigate whether the ECM values of the PrM models are lower than those of the AlM models. The relative performance comparison for the two indirect methods is evaluated for a significance level of 10%, i.e., $\alpha = 0.1$. Hence: if $p \leq \alpha$, then PrM is significantly better than AlM; and if $(1 - p) \leq \alpha$, then AlM is significantly better than PrM. For all other values of p both methods are considered to have similar performances.

In the context of the DA technique, we observe (p-values) that the PrM models perform significantly better than the AlM models for most cost combinations, except for the rows corresponding to the higher cost values. In the case of most of the higher cost values, the performances of both indirect approaches are similar, i.e., neither method yields better performance than the other. The DA model based on the AlM performs significantly better ($(1 - p) \geq 0.90$) than the one based on PrM for only two cost combination cases: rows 32 and 39 in Table 17. It should be noted that for a software system such as the case study presented, the cost values such as rows 31 and above are (practically speaking) unlikely to occur. However, we presented the comparative results for a wide range of cost combinations to evaluate the models from a sensitivity analysis point of view.

In the context of the C4.5 and CBR techniques, we observe that for most cost (over 80%) combinations up to row 16, the PrM yielded significantly ($p \leq 0.10$) better performance than the AlM. However, for rows pertaining to the higher cost values the

Table 17. Proposed method vs. alternate method: ECM values.

	DA			C4.5			CBR		
	ECM			ECM			ECM		
#	PrM	AlM	*p* value	PrM	AlM	*p* value	PrM	AlM	*p* value
1	0.6812	0.8475	0.00	0.6300	0.7916	0.00	0.6262	0.7527	0.00
2	0.6985	0.8562	0.00	0.6441	0.7963	0.00	0.6448	0.7582	0.00
3	0.7621	0.9191	0.00	0.7027	0.8322	0.00	0.6943	0.7985	0.00
4	0.6985	0.8562	0.00	0.6455	0.7968	0.00	0.6448	0.7587	0.00
5	0.7621	0.9191	0.00	0.7042	0.8327	0.00	0.6943	0.7990	0.00
6	0.7968	0.9364	0.00	0.7324	0.8421	0.01	0.7314	0.8099	0.00
7	0.6985	0.8562	0.00	0.6478	0.7975	0.00	0.6448	0.7594	0.00
8	0.7621	0.9191	0.00	0.7064	0.8334	0.00	0.6943	0.7998	0.00
9	0.8894	1.0448	0.00	0.8238	0.9052	0.05	0.7933	0.8804	0.00
10	0.7968	0.9364	0.00	0.7384	0.8441	0.01	0.7314	0.8119	0.00
11	0.924	1.0621	0.00	0.8557	0.9158	0.11	0.8304	0.8926	0.02
12	1.0686	1.1965	0.00	0.9871	0.9923	0.46	0.9480	0.9787	0.17
13	0.976	1.0881	0.00	0.9017	0.9312	0.27	0.8861	0.9101	0.21
14	1.0686	1.1965	0.00	0.9908	0.9936	0.48	0.9480	0.9800	0.16
15	0.7505	0.8822	0.00	0.7012	0.8153	0.00	0.7005	0.7795	0.00
16	1.0686	1.1965	0.00	0.9946	0.9948	0.50	0.9480	0.9812	0.15
17	1.3866	1.5109	0.02	1.2879	1.1743	0.90	1.1955	1.1829	0.61
18	1.4733	1.5542	0.10	1.3658	1.2002	0.96	1.2884	1.2126	0.92
19	1.3285	1.3265	0.51	1.2136	1.0678	0.97	1.2265	1.0653	0.99
20	1.2418	1.2832	0.23	1.1505	1.0468	0.94	1.1337	1.0406	0.97
21	1.5599	1.5975	0.29	1.4438	1.2262	0.98	1.3812	1.2423	0.98
22	1.3866	1.5109	0.02	1.3250	1.1866	0.92	1.1955	1.1953	0.50
23	1.9646	1.9552	0.54	1.8300	1.4366	0.99	1.7215	1.4787	0.99
24	2.1379	2.0418	0.80	1.9710	1.4837	1.00	1.9072	1.5332	1.00
25	1.2418	1.2832	0.23	1.1876	1.0592	0.96	1.1337	1.0530	0.97
26	2.3693	2.3129	0.69	2.2087	1.6446	1.00	2.0619	1.7126	1.00
27	3.2369	3.2126	0.56	3.0181	2.1594	1.00	2.7116	2.2906	0.99
28	3.9597	3.8847	0.65	3.6752	2.5418	1.00	3.2995	2.7213	0.99
29	1.9646	1.9552	0.54	1.8597	1.4465	1.00	1.7215	1.4886	0.99
30	3.873	3.8413	0.57	3.6196	2.5233	1.00	3.2067	2.6990	0.99
31	1.2418	1.2832	0.23	1.2248	1.0715	0.97	1.1337	1.0653	0.96
32	1.675	1.4998	0.94	1.5775	1.1891	1.00	1.5978	1.2015	1.00
33	4.0463	3.928	0.72	3.7829	2.5777	1.00	3.3923	2.7609	1.00
34	4.3928	4.1012	0.88	4.0651	2.6718	1.00	3.7636	2.8698	1.00
35	1.9064	1.7708	0.89	1.8374	1.3574	1.00	1.7525	1.3884	1.00
36	2.9473	2.7572	0.88	2.7879	1.9193	1.00	2.5879	2.0208	1.00
37	4.2196	4.0146	0.82	3.9611	2.6371	1.00	3.5780	2.8277	1.00
38	5.7814	5.7275	0.58	5.4389	3.6198	1.00	4.6918	3.9292	0.99
39	5.2592	4.5344	0.97	4.8077	2.9193	1.00	4.6918	3.1545	1.00
40	7.5735	7.2448	0.79	7.0354	4.4787	1.00	6.2389	4.8995	1.00

C4.5 and CBR models based on the AlM were generally better ($(1 - p) \geq 0.90$) than the corresponding models based on the PrM. Moreover, for the intermediate cost values, i.e., middle portion of the table, the two indirect methods yielded mixed results. Therefore, we observe that at the lower end of the cost values spectrum the PrM performs better, while at the higher end of the cost values spectrum the AlM performs better. This mixed empirical observation for the indirect models using the C4.5 and CBR techniques, i.e., neither indirect method is clearly better than the other, suggests that further empirical case study investigations are required.

Our future work will evaluate the performance of the indirect methods in the context of other two-group classification techniques. Overall, the performance of both indirect methods confirms the promising use of an indirect approach for calibrating three-group classification models with any existing two-group classification technique.

8. Threats to Validity

In an empirical software engineering task it is important to consider the threats to validity of the obtained results and conclusions (Wohlin et al., 2000). We consider three types of threats to validity for the case study presented earlier. They include: threats to internal validity, threats to external validity, and threats to conclusion validity.

Threats to internal validity are unaccounted influences that may affect case study results. Internal validity is the ability to show that results obtained were due to the manipulation of the experimental treatment variables. In the context of this study, poor classification estimates can be caused by a wide variety of factors, including measurement errors while collecting and recording software metrics; modeling errors due to the unskilled use of software applications; errors in model-selection during the modeling process; and the presence of noise in the training data set. Measurement errors are inherent to the software data collection effort, which is usually specific to the system under consideration. In our study, a common model-building and model-selection approach have been adopted for all the three-group classification techniques. Moreover, the modeling, experimental, and statistical analysis was performed by only one skilled person in order to keep modeling errors to a minimum.

External validity is concerned with generalization of the obtained results outside the experimental setting. Therefore, threats to external validity are conditions that limit generalization of case study results. To be credible, the software engineering community demands that the subject of an empirical study be a system with the following characteristics (Votta and Porter, 1995): (1) developed by a group, rather than an individual; (2) developed by professionals, rather than students; (3) developed in an industrial environment, rather than an artificial setting; and (4) large enough to be comparable to real industry projects. The software system investigated in this study meets all these requirements.

However, it is known in the software engineering field that a given software metrics-based prediction system is likely to be affected by (among other issues) the characteristics of the data and the application domain of the system under consideration (Shepperd and Kadoda, 2001). The results and conclusion of this paper are based on the case study presented, and cannot be generalized for another software system, because it may: have different software data characteristics; utilize different software metrics; and have different software quality improvement objectives. However, the proposed three-group classification method can certainly be applied to other systems. Recall that the proposed method is simpler than other (direct) three-group classification methods, and does not require the analyst to gain expertise in a direct three-group classification method.

Conclusion validity is concerned with the statistical relationship between the treatment and the outcome. It is sometimes also referred to as statistical conclusion validity, i.e., is the relationship between the treatment and the outcome satisfiable for a given significance level. Threats to conclusion validity are issues that affect the ability to draw the correct inferences. In this study the discussions related to performance comparisons between competing three-group classification models were based on observing the *p*-values, as discussed previously.

9. Conclusion

In certain software quality prediction situations, a two-group classification model may not provide the needed cost-effective and focussed risk-based quality models. Two-group models such as fault-prone and not fault-prone, are calibrated on the premise that enough software inspection and quality testing efforts will be applied to all modules flagged as *fp*. However, practically speaking, such project resources are usually pre-assigned and limited. Consequently, all low-quality modules may not be improved. Moreover, among the predicted *fp* modules, the variability of the associated risk-factor (e.x., number of faults) may be high, leading to the difficult issue of which *fp* modules should be targeted for inspection.

When a more focussed software quality classification with regards to better cost-effective resource utilization and ROI, a three-group classification model, i.e., high-risk (Red), medium-risk (Yellow), and low-risk (Green), may be more rewarding. By segregating the high-risk modules from the medium-risk modules, a greater cost-effective quality control can be achieved. However, calibrating such models with existing three-group classification techniques can be computationally expensive, complicated, and difficult to implement and comprehend. Moreover, adopting an unfamiliar classification technique (for direct three-group models) may incur an overhead investment of learning a new technique. In addition, some effective classification methods such as logistic regression and Boolean discriminant functions, currently do not facilitate a three-group model.

A case study of a comprehensive empirical investigation of an innovative three-group classification methodology was presented. The novelty of the proposed technique lies in the fact that it can be applied by using any existing two-group classification technique, in order to yield the three risk-based groups. Hence, the method circumvents the complexities and difficulties faced in calibrating direct three-group models. Moreover, techniques that do not provide a three-group classification can also be exploited.

The main aim of calibrating three-group models as compared to two-group models is to improve (cost-effectiveness) quality improvement resource utilization. In our studies, we evaluated the predictive capabilities of the different models based on their expected cost of misclassification values. We explored the validity of the proposed three-group modeling method by studying models obtained from applying the proposed method to existing two-group classification techniques. The techniques that were considered included, discriminant analysis, the C4.5 decision tree algorithm, case-based reasoning, and logistic regression. However, for paper-size considerations, we presented the empirical results of only the first three techniques.

Since the discriminant analysis and C4.5 decision tree techniques facilitated a direct three-group classification model, a comparative study was performed in which the direct (without proposed method) and indirect (with proposed method) three-group models were studied. For the case study presented and others, it was observed that the indirect three-group models generally yielded better misclassification error rates than those of the direct three-group models, i.e., DA and C4.5. Furthermore, over a wide spectrum of possible misclassification costs, the proposed method yielded consistently better ECM values than those calibrated directly by the respective classification techniques. The proposed indirect method was also compared to an alternate indirect three-group classification method as suggested by an anonymous reviewer. The proposed indirect models based on the DA technique generally yielded better performances. However, the results of the indirect models based on the C4.5 and CBR techniques were not conclusive to indicate which indirect method was better.

The simplicity and applicability of an indirect three-group classification method is very attractive. We prefer an indirect method over a competing direct three-group model even if both techniques provide similar results. Practitioners do not need a great deal of expertise when applying an indirect method for three-group classification. Organizations that are reluctant to use three-group models due to unfamiliarity of existing (direct) techniques, can utilize a familiar two-group classification algorithm in conjunction with the indirect methods presented. Future work may involve comparative empirical validations with other classification techniques, such as artificial neural networks.

Acknowledgments

We are grateful to the coordinating editor, Claes Wohlin, and the three referees for their useful suggestions and comments. We also thank Jayanth Rajeevalochanam for his assistance with reviews. This work was supported in part by the Cooperative Agreement NCC 2-1141 from NASA Ames Research Center, Software Technology Division, and the NASA Grant NAG 5-12129 for the NASA software IV&V Facility at Fairmont, West Virginia.

Notes

1. In this paper, we used the words 'pure' and 'direct' interchangeably to imply a three-group model calibrated directly by the classification technique.
2. This includes those that do not provide a direct three group model. For example, though not presented, we applied the proposed technique using logistic regression.
3. The individual error rates of two competing models may vary inconsistently, making model-selection a difficult and subjective task.

4. All tables including and after Table 9 are presented at the end of the paper.
5. $Z = \overline{D}/(\sigma_D/\sqrt{n})$, where D_j are considered as a sample of difference scores ($D_1, \ldots D_n$) between related (i.e., *matched* or *paired*) observations, σ_D is the standard deviation of D, and $\bar{D}$ is the mean value of a sample of differences (Berenson et al., 1983).

References

Basili, V. R., Briand, L. C., and Melo, W. L. 1996. A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on Software Engineering* 22(10): 751–761.

Beizer, B. 1990. *Software Testing Techniques*. 2nd edition. New York, NY, USA: ITP Van Nostrand Rienhold.

Berenson, M. L., Levine, D. M., and Goldstein, M. 1983. *Intermediate Statistical Methods and Applications: A Computer Package Approach*. Englewood Cliffs, NJ, USA: Prentice Hall.

Bhupathiraju, S. S. 2002. An empirical study of a three-group classification model using case-based reasoning. Master's thesis, Florida Atlantic University, Boca Raton, FL, USA. Advised by Taghi M. Khoshgoftaar.

Briand, L. C., Basili, V. R., and Hetmanski, C. J. 1993. Developing interpretable models with optimized set reduction for identifying high-risk software components. *IEEE Transactions on Software Engineering* 19(11): 1028–1044.

Ebert, C. 1996. Classification techniques for metric-based software development. *Software Quality Journal* 5(4): 255–272.

Fayyad, U. M. 1996. Data mining and knowledge discovery: Making sense out of data. *IEEE Expert* 11(4): 20–25.

Fenton, N. E., and Pfleeger, S. L. 1997. *Software Metrics: A Rigorous and Practical Approach*, 2nd edition. Boston, MA, USA: PWS Publishing Company: ITP.

Gray, A. R., and MacDonell, S. G. 1999. Software metrics data analysis: exploring the relative performance of some commonly used modeling techniques. *Empirical Software Engineering Journal* 4: 297–316.

Hochman, R., Khoshgoftaar, T. M., Allen, E. B., and Hudepohl, J. P. 1997. Evolutionary neural networks: a robust approach to software reliability problems. *Proceedings: 8th International Symposium on Software Reliability Engineering*. Albuquerque, NM, USA, pp. 13–26.

Khoshgoftaar, T. M., and Allen, E. B. 1999. Logistic regression modeling of software quality. *International Journal of Reliability, Quality and Safety Engineering* 6(4): 303–317.

Khoshgoftaar, T. M., and Allen, E. B. 2000. A practical classification rule for software quality models. *IEEE Transactions on Reliability* 49(2): 209–216.

Khoshgoftaar, T. M., and Allen, E. B. 2001. Modeling software quality with classification trees. In H. Pham, (ed.), *Recent Advances in Reliability and Quality Engineering*, Singapore: World Scientific Publishing, pp. 247–270, Chapt. 15.

Khoshgoftaar, T. M., and Lanning, D. L. 1995. A neural network approach for early detection of program modules having high risk in the maintenance phase. *Journal of Systems and Software* 29(1): 85–91.

Khoshgoftaar, T. M., and Seliya, N. 2002. Improving usefulness of software quality classification models based on boolean discriminant functions. *Proceedings: 13th International Symposium on Software Reliability Engineering*. Annapolis, MD, USA, pp. 221–230.

Khoshgoftaar, T. M., and Seliya, N. 2003. Analogy-based practical classification rules for software quality estimation. *Empirical Software Engineering Journal* 8(4): 325–350.

Khoshgoftaar, T. M., Allen, E. B., and Busboom, J. C. 2000a. Modeling software quality: The software measurement analysis and reliability toolkit. *Proceedings: 12th International Conference on Tools with Artificial Intelligence*. Vancouver, BC, Canada, pp. 54–61.

Khoshgoftaar, T. M., Allen, E. B., Jones, W. D., and Hudepohl, J. P. 2000b. Accuracy of software quality models over multiple releases. *Annals of Software Engineering* 9(1–4): 103–116. Kluwer Academic Publishers.

Khoshgoftaar, T. M., Yuan, X., and Allen, E. B. 2000c. Balancing misclassification rates in classification tree models of software quality. *Empirical Software Engineering Journal* 5: 313–330. Kluwer Academic Publishers.

Khoshgoftaar, T. M., Allen, E. B., and Deng, J. 2002. Using regression trees to classify fault-prone software modules. *IEEE Transactions on Reliability* 51(4): 455–462.

Kolodner, J. 1993. *Case-Based Reasoning*. San Mateo, CA, USA: Morgan Kaufmann Publishers Inc.

Lanning, D. L., and Khoshgoftaar, T. M. 1995. The impact of software enhancement on software reliability. *IEEE Transactions on Reliability* 44(4): 677–682.

Leake, D. B. (ed.) 1996. *Case-Based Reasoning: Experience, Lessons, and Future Directions*. Cambridge, MA, USA: MIT Press.

Michalski, R. S., Bratko, I., and Kubat, M. 1998. *Machine Learning and Data Mining: Methods and Applications*, New York, NY: John Wiley and Sons.

Ohlsson, M. C., and Runeson, P. 2002. Experience from replicating empirical studies on prediction models. *Proceedings: 8th International Software Metrics Symposium*. Ottawa, Ontario, Canada, pp. 217–226.

Ohlsson, M. C., and Wohlin, C. 1998. Identification of green, yellow and red legacy components. *Proceedings: International Conference on Software Maintenance*. Bethesda, Washington D.C., USA, pp. 6–15.

Ohlsson, N., Helander, M., and Wohlin, C. 1996. Quality improvement by identification of fault-prone modules using software design metrics. *Proceedings: International Conference on Software Quality*. Ottawa, Ontario, Canada, pp. 1–13.

Ohlsson, M. C., Mayrhauser, A. V., McGuire, B., and Wohlin, C. 1999. Code decay analysis of legacy software through successive releases. *Proceedings: Aerospace Conference (Volume 5)*, Vol. 5. Aspen, CO, USA, pp. 69–81.

Ponnuswamy, V. 2001. Classification of software quality with tree modeling using C4.5 algorithm. Master's thesis, Florida Atlantic University, Boca Raton, FL, USA. Advised by Taghi M. Khoshgoftaar.

Porter, A. A., and Selby, R. W. 1990. Empirically guided software development using metric-based classification trees. *IEEE Software* 7(2): 46–54.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning, Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Runeson, P., Ohlsson, M. C., Wohlin, C. 2001. A classification scheme for studies on fault-prone components. *Lecture Notes in Computer Science* 2188: 341–355. Springer Link.

Schneidewind, N. F. 1997. Software metrics model for integrating quality control and prediction. *Proceedings: 8th International Symposium on Software Reliability Engineering*. Albuquerque, NM, USA, pp. 402–415.

Schneidewind, N. F. 2001. Investigation of logistic regression as a discriminant of software quality, *Proceedings: 7th International Software Metrics Symposium*. London, UK, pp. 328–337.

Seber, G. A. F. 1984. *Multivariate Observations*. New York, NY, USA: John Wiley and Sons.

Shepperd, M., and Kadoda, G. 2001. Comparing software prediction techniques using simulation. *IEEE Transactions on Software Engineering* 27(11): 1014–1022.

Song, H. 2001. Implementation of a three-group classification model using case-based reasoning. Master's thesis, Florida Atlantic University, Boca Raton, FL, USA. Advised by T. M. Khoshgoftaar.

Szabo, R. M. 1995. Improved models of software quality. Ph.D. thesis. Florida Atlantic University, Boca Raton, FL, USA. Advised by Taghi M. Khoshgoftaar.

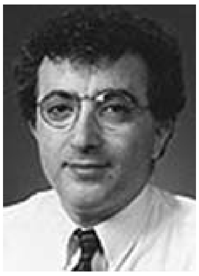
Szabo, R. M., and Khoshgoftaar, T. M. 2000. Classifying software modules into three risk groups. In H. Pham and M.-W. Lu, (eds.), *Proceedings: 6th International Conference on Reliability and Quality in Design*. Orlando, FL, USA, pp. 90–95.

Takahashi, R., Muraoka, Y., and Nakamura, Y. 1997. Building software quality classification trees: Approach, experimentation, evaluation. *Proceedings: 8th International Symposium on Software Reliability Engineering*. Albuquerque, NM, USA, pp. 222–233.

Votta, L. G., and Porter, A. A. 1995. Experimental software engineering: A report on the state of the art, *Proceedings of the 17th International Conference on Software Engineering*. Seattle, WA, USA, pp. 277–279.

Wohlin, C., Runeson, P., Host, M., Ohlsson, M. C., Regnell, B., and Wesslen, A. 2000. *Experimentation in Software Engineering: An Introduction, Kluwer International Series in Software Engineering*. Massachuesetts, USA: Kluwer Academic Publishers.

Xu, Z., and Khoshgoftaar, T. M. 2001. Software quality prediction for high assurance network telecommunications systems. *The Computer Journal* 44(6): 557–568. British Computer Society.



Taghi M. Khoshgoftaar is a professor of the Department of Computer Science and Engineering, Florida Atlantic University and the Director of the Empirical Software Engineering Laboratory. His research interests are in software engineering, software metrics, software reliability and quality engineering, computational intelligence, computer performance evaluation, data mining, and statistical modeling. He has published more than 200 refereed papers in these areas. He has been a principal investigator and project leader in a number of projects with industry, government, and other research-sponsoring agencies. He is a member of the Association for Computing Machinery, the IEEE Computer Society, and IEEE Reliability Society. He served as the general chair of the 1999 International Symposium on Software Reliability Engineering (ISSRE'99), and the general chair of the 2001 International Conference on Engineering of Computer Based Systems. Also, he has served on technical program committees of various international conferences, symposia, and workshops. He has served as North American editor of the *Software Quality Journal*, and is on the editorial boards of the journals *Empirical Software Engineering, Software Quality, and Fuzzy Systems*.

Naeem Seliya received the M.S. degree in Computer Science from Florida Atlantic University, Boca Raton, FL, USA, in 2001. He is currently a Ph.D. candidate in the Department of Computer Science and Engineering at Florida Atlantic University. His research interests include software engineering, computational intelligence, data mining, software measurement, software reliability and quality engineering, software architecture, computer data security, and network intrusion detection. He is a student member of the IEEE Computer Society and the Association for Computing Machinery.

Kehan Gao received the M.S. degree in Mathematics from Florida Atlantic University, Boca Raton, FL, USA, in 1999. She is currently a Ph.D. student in the Department of Computer Science and Engineering at Florida Atlantic University. Her research interests include software engineering, software metrics, software reliability and quality engineering, computer performance modeling, computational intelligence, and data mining.