Software Defect Association Mining and Defect Correction Effort Prediction

Qinbao Song, Martin Shepperd, Michelle Cartwright, and Carolyn Mair

Abstract—Much current software defect prediction work focuses on the number of defects remaining in a software system. In this paper, we present association rule mining based methods to predict defect associations and defect correction effort. This is to help developers detect software defects and assist project managers in allocating testing resources more effectively. We applied the proposed methods to the SEL defect data consisting of more than 200 projects over more than 15 years. The results show that, for defect association prediction, the accuracy is very high and the false-negative rate is very low. Likewise, for the defect correction effort prediction, the accuracy for both defect isolation effort prediction and defect correction effort prediction are also high. We compared the defect correction effort prediction method with other types of methods—PART, C4.5, and Naïve Bayes—and show that accuracy has been improved by at least 23 percent. We also evaluated the impact of support and confidence levels on prediction accuracy, false-negative rate, false-positive rate, and the number of rules. We found that higher support and confidence levels may not result in higher prediction accuracy, and a sufficient number of rules is a precondition for high prediction accuracy.

Index Terms—Software defect prediction, defect association, defect isolation effort, defect correction effort.

1 Introduction

The success of a software system depends not only on cost and schedule, but also on quality. Among many software quality characteristics, residual defects has become the *de facto* industry standard [12]. Therefore, the prediction of software defects, i.e., deviations from specifications or expectations which might lead to failures in operation [11], has been an important research topic in the field of software engineering for more than 30 years. Clearly, they are a proxy for reliability, but, unfortunately, reliability is extremely difficult to assess prior to full deployment.

Current defect prediction work focuses on estimating the number of defects remaining in software systems with code metrics, inspection data, and process-quality data by statistical approaches [7], [18], [9], capture-recapture (CR) models [27], [21], [6], [10], and detection profile methods (DPM) [28].

The prediction result, which is the number of defects remaining in a software system, can be used as an important measure for the software developer [16], and can be used to control the software process (i.e., decide whether to schedule further inspections or pass the software artifacts to the next development step [19]) and gauge the *likely* delivered quality of a software system [11]. In contrast, Bhandari et al. [4], [5] propose that the defects found during production are a manifestation of process deficiencies, so they present a case study of the use of a defect based method for software in-process improvement. In particular, they use an attribute-focusing (AF) method [3] to discover associations among defect attributes such as defect type, source, phase introduced, phase found, component, impact, etc. By finding out the event that could have led to the associations, they identify a process problem and implement a corrective action. This can lead a project team to improve its process during development. We restrict our work to the predictions of defect (type) associations and corresponding defect correction effort. This is to help answer the following questions:

1. For the given defect(s), what other defect(s) may co-occur?
2. In order to correct the defect(s), how much effort will be consumed?

We use defect type data to predict software defect associations that are the relations among different defect types such as: If defects a and b occur, then defect c also will occur. This is formally written as $a \wedge b \Rightarrow c$. The defect associations can be used for three purposes:

First, find as many related defects as possible to the detected defect(s) and, consequently, make more-effective corrections to the software. For example, consider the situation where we have classes of defect a, b, and c and suppose the rule $a \wedge b \Rightarrow c$ has been obtained from a historical data set, and the defects of class a and b have been detected occurring together, but no defect of class c has yet been discovered. The rule indicates that a defect of class c is likely to have occurred as well and indicates that we should check the corresponding software artifact to see whether or not such a defect really exists. If the result is positive, the search can be continued so, if rule $a \wedge b \wedge c \Rightarrow d$ holds as well, we can do the same thing for defect d. We

• *Q. Song is with the Department of Computer Science and Technology, Xi'an Jiaotong University, 28 Xian-Ning West Rd., Xi'an, Shaanxi, 710049 China. E-mail: qbsong@mail.xjtu.edu.cn.*
• *M. Shepperd, M. Cartwright, and C. Mair are with the School of Information Science, Computing, and Mathematics, Brunel University, Uxbridge, UB8 3PH UK.*
E-mail: {martin.shepperd, michelle.cartwright, carolyn.mair}@brunel.ac.uk.

Manuscript received 9 Dec. 2004, revised 10 Oct. 2005; accepted 2 Dec. 2005; published online 15 Feb. 2006.
Recommended for acceptance by R. Lutz.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-0276-1204.

believe this may be useful as it permits more-directed testing and more-effective use of limited testing resources.

Second, help evaluate reviewers' results during an inspection. For example, if rule $a \wedge b \Rightarrow c$ holds but a reviewer has only found defects a and b, it is possible that (s)he missed defect c. Thus, a recommendation might be that his/her work should be reinspected for completeness.

Third, to assist managers in improving the software process through analysis of the reasons some defects frequently occur together. If the analysis leads to the identification of a process problem, managers have to come up with a corrective action.

At the same time, for each of the associated defects, we also predict the likely effort required to isolate and correct it. This can be used to help project managers improve control of project schedules.

Both of our defect association prediction and defect correction effort prediction methods are based on the association rule mining method which was first explored by Agrawal et al. [1]. Association rule mining aims to discover the patterns of co-occurrences of the attributes in a database. However, it must be stressed that associations do not imply causality. An association rule is an expression $\mathcal{A} \Rightarrow \mathcal{C}$, where $\mathcal{A}$ (Antecedent) and $\mathcal{C}$ (Consequent) are sets of items. The meaning of such rules is quite intuitive: Given a database $\mathcal{D}$ of transactions, where each transaction $\mathcal{T} \in \mathcal{D}$ is a set of items, $\mathcal{A} \Rightarrow \mathcal{C}$ expresses that whenever a transaction $\mathcal{T}$ contains $\mathcal{A}$, then $\mathcal{T}$ also contains $\mathcal{C}$ with a specified confidence. The rule confidence is defined as the percentage of transactions containing $\mathcal{C}$ in addition to $\mathcal{A}$ with regard to the overall number of transactions containing $\mathcal{A}$.

The idea of mining association rules originates from the analysis of market-basket data where rules like "customers who buy products p_1 and p_2 will also buy product p_3 with probability c percent" are extracted. Their direct applicability to business problems together with their inherent understandability make association rule mining a popular data mining method. However, it is clear that association rule mining is not restricted to dependency analysis in the context of retail applications, but can successfully be applied to a wide range of business and science problems.

Although association rule mining aims to discover the co-occurring patterns of the attributes in databases, it has been shown that association rule mining based classification—associative classification—frequently has higher classification accuracy than other classification methods. The underlying reason is that these methods use heuristic/greedy search techniques to build classifiers; they induce a representative subset of rules. However, as association rule mining explores high-confidence associations among multiple variables, it may overcome some constraints introduced by other techniques, e.g., decision-tree induction methods that examine one variable at a time. Thus, associative classification takes the most effective rule(s) from among all the rules mined for classification.

Extensive performance studies have also shown that associative classification frequently generates better accuracy than state-of-the-art classification methods. Ali et al. [2] use association rule mining to do a partial classification in the context of very large numbers of class attributes, when most attribute values are missing, or the class distribution is highly skewed and the user is interested in understanding the low-frequency classes. Liu et al. [17] and Yin and Han [30] integrate classification and association rule mining to build a classifier; the former method prunes rules using both minimum support and pessimistic estimation. Wang et al. [26] proposed a general method for turning an arbitrary set of association rules into a classifier. Dong et al. [8] combined several association rules to classify a new case, which partially addressed the low support of classification rules because a combined rule has a lower support. Wang et al. [25] used multilevel association rules to build hierarchical classifiers where both the class space and the feature space are organized into a taxonomy. Association rule mining based methods have also been used in the prediction of WWW caching and prefetching [29], outer membrane proteins [23], and software source-code changes [31], [32]. The successful use of association rule mining in various fields motivates us to apply it to the software defect data set.

The rest of the paper is organized as follows: In Section 2, we describe the approach used by the study. In Section 3, we present the specific defect association and defect-correction effort prediction methods. In Section 4, we present our experimental results. Finally, in Section 5, we summarize our work and findings.

2 Research Method

2.1 General Method

The objective of the study is to discover software defect associations from historical software engineering data sets, and help determine whether or not a defect(s) is accompanied by other defect(s). If so, we attempt to determine what these defects are and how much effort might be expected to be used when we correct them. Finally, we aim to help detect software defects and effectively improve software control.

For this purpose, first, we preprocessed the NASA SEL data set (see the following section for details), and obtained three data sets: the defect data set, the defect isolation effort data set, and the defect correction effort data set. Then, for each of these data sets, we randomly extracted five pairs of training and test data sets as the basis of the research. After that, we used association rule mining based methods to discover defect associations and the patterns between a defect and the corresponding defect isolation/correction effort. Finally, we predicted the defect(s) attached to the given defect(s) and the effort used to correct each of these defects. We also compared the results with alternative methods where applicable.

2.2 Data Source and Data Extraction

The data we used is SEL Data [22] which is a subset of the online database created by the NASA/GSFC Software Engineering Laboratory (SEL) for storage and retrieval of software engineering data for NASA Goddard Space Flight Center. The subset includes defect data of more than 200 projects completed over more than 15 years.

TABLE 1
Defect Type Description

Defect type	Description
Computational defect	Computational defects are those that cause a computation to erroneously evaluate a variable's value. These defects could be equations that are incorrect not because of the incorrect use of a data structure within the statement but by miscalculation.
Data value defect	Data value defects are those that are a result of the incorrect use of a data structure. Examples of this type of defects errors are the use of incorrect subscripts for an array, the use of the wrong variable in an equation, the use of the wrong unit of measurement, or the inclusion of an incorrect declaration of a variable local to the module.
Internal interface defect	Internal interface defects are those that were associated with internal structures of a module.
External interface defect	External interface defects are those that were associated with structures existing outside the module's local environment but which the module used.
Initialization defect	Initialization defects are those that result from an incorrectly initialized variable, failure to re-initialize a variable, or because a necessary initialization was missing; failure to initialize or re-initialize a data structure properly upon a module's entry/exit is also considered an initialization defect.
Logic/control structure defect	Logic/control structure defects are those that cause an "incorrect path" in a module to be taken. Such a control defect might be a conditional statement causing control to be passed to an incorrect path.

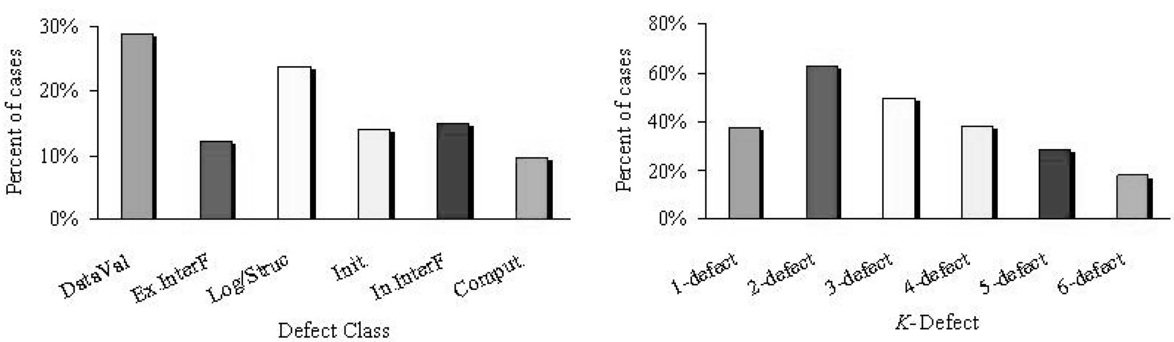

Fig. 1. Distribution of cases by defect type and k-defect for the defect data set.

The SEL Data is a database consisting of 15 tables that provide data on the projects' software characteristics (overall and at a component level), changes and errors during all phases of development, and effort and computer resources used. In the SEL Data, the defects are divided into six types, Table 1 contains the details. See [15] for further information.

In addition, the effort used to correct defects falls into four categories: One Hour or Less, One Hour to One Day, One Day to Three Days, and More Than Three Days.

For the purpose of defect association prediction, we used SQL to extract defect data from different tables of the SEL data and obtained the basic defect data set. The defect data is very simple, and consists of defect types and the corresponding dates on which the need for change was determined. Then, we followed the sliding window approach, that is, two subsequent defects a and b are part of one transaction if they are at most one day apart and belong to the same project, to infer the transactions needed for the association rule mining. Fig. 1[1] contains summary information. We would like to clarify that while we set the size of the sliding window to one day in this analysis, it does not imply that we are limited to this value and it could be set to any other time interval thought appropriate. Moreover, placing defects into one transaction according to the selected sliding window just means they co-occur during the given time window, it does not imply they

1. The notation k-defect means that k defects occurred together in a transaction.

TABLE 2
Defect Effort Data Description

Attribute	Description
Defect type	Defect type is the type of a defect as introduced in Table 1.
Effort	Effort used to isolate or correct the defect.
Typo	Typo is the flag that indicates a typographical error.
Code_Left_Out	Code_Left_Out is the flag that indicates an omission error that is a result of forgetting to include some entity within a module.
Bad_Code_Added	Bad_Code_Added is the flag that indicates a commission error that is a result of an incorrect executable statement.

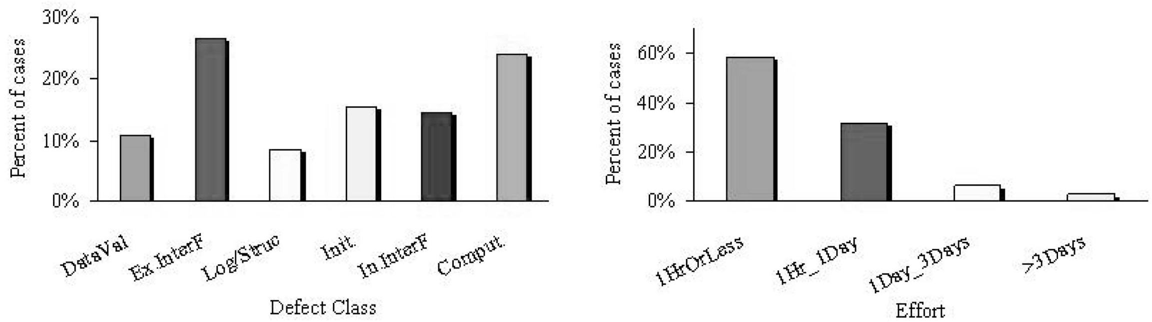

Fig. 2. Distribution of cases by defect type and effort for the isolation effort data set.

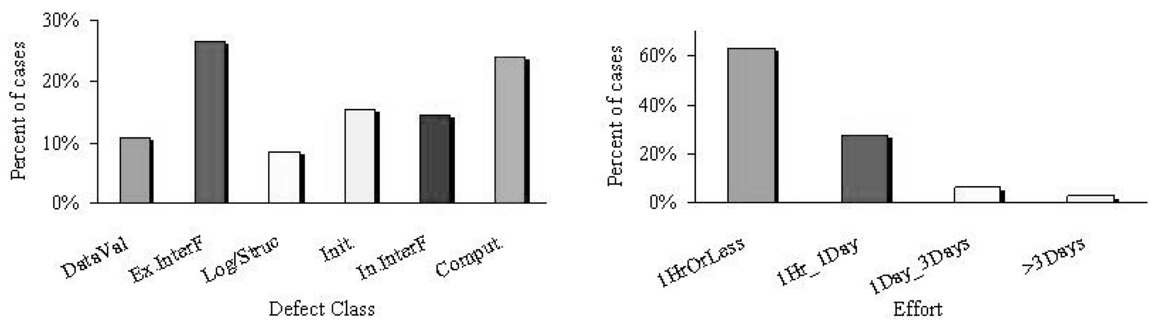

Fig. 3. Distribution of cases by defect type and effort for the correction effort data set.

must be dependent. However, if the placement of defects is coincidental they tend not to form association rules.

For the purpose of defect correction effort prediction, we also used SQL to extract defect data and the corresponding isolation and correction effort data from SEL data and obtained two further data sets: the defect isolation effort data set and the defect correction effort data set. Both consist of five attributes (see Table 2 for details). Fig. 2 and Fig. 3 provide the corresponding summary information.

For both the defect association and defect correction effort predictions, the data set $\mathcal{D}$ is randomly split into five mutually exclusive subsets $\mathcal{D}_1$, $\mathcal{D}_2$, ..., and $\mathcal{D}_5$ of equal size, and $\cup_{i=1}^{5}\mathcal{D}_i = \mathcal{D}$. We used $\mathcal{D} \ominus \mathcal{D}_t$ [2] $(t = \{1, 2, \ldots, 5\})$, and $\mathcal{D}_t$ as training sets and test sets, respectively.

2.3 Analysis Approach

We use the five-fold cross-validation method as the overall analysis approach. That is, for each $\mathcal{D}$ of the defect data set, the defect isolation effort data set, and the defect correction effort data set, the inducer is trained and tested a total of five times. Each time $t \in \{1, 2, \ldots, 5\}$, it is trained on $\mathcal{D} \ominus \mathcal{D}_t$ and tested on $\mathcal{D}_t$.

We use the association rule mining method to learn rules from the training data sets. For defect association prediction, the rule learning is straightforward, while for defect correction effort prediction, it is more complicated because the consequent of a rule has to be defect correction effort. Considering the target of association rule mining is not predetermined and classification rule mining has only one predetermined target, the class, we integrate these two techniques to learn defect correction effort prediction rules by focusing on a special subset of association rules whose consequents are restricted to the special attribute, the effort.

Once we obtain the rules, we rank them, and use them to predict the defect associations and defect correction effort with the corresponding test data sets. The predictions of defect associations and defect correction effort are both

2. The notation $\mathcal{D} \ominus \mathcal{D}_t$ means set $\mathcal{D}$ minus set $\mathcal{D}_t$.

based on the length-first (in terms of the number of items in a rule) strategy (see Section 3.2 for details).

To our knowledge, there is no work on software defect association prediction and there is no other method that can be used for this purpose. Therefore, we are unable to directly compare our defect association prediction method with other studies. As the defect correction effort is represented in the form of categorical values and there are some attributes to characterize it, it can be viewed as a classification problem. This allows us to compare our defect correction prediction method with three different types of methods. These methods are the Bayesian rule of conditional probability-based method, Naïve Bayes [13], the well-known trees-based method, C4.5 [20], and the simple and effective rules-based method, PART [14].

3 Rule Discovery and Defect/Effort Prediction

In this section, we first introduce the basic concepts of association rule mining. Then, we present the rule-ranking strategy used for the purpose of defect association and defect correction effort predictions. After that, we respectively give the methods of defect association prediction and defect correction effort prediction based on the association rule mining method.

3.1 Association Rule Discovery

Association rule mining searches for interesting relationships, e.g., frequent patterns, associations, correlations, or potential causal structures, among sets of objects in databases or other information repositories. The approach is data rather than hypothesis driven. The interestingness of an association rule is measured by both support and confidence, which respectively reflect the usefulness and certainty of the rule. It must be stressed that even rules that discover with high levels of support (or relevance) and high confidence do not necessarily imply causality. However, such rules would obviously stimulate further research through the postulation of models that can be empirically evaluated.

Let $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_m\}$ be a set of attribute values, called items. A set $\mathcal{A} \subseteq \mathcal{I}$ is called an item set. Let a database $\mathcal{D}$ be a multiset of $\mathcal{I}$. Each $\mathcal{T} \in \mathcal{D}$ is called a transaction. An association rule is an expression $\mathcal{A} \Rightarrow \mathcal{C}$, where $\mathcal{A} \subset \mathcal{I}$, $\mathcal{C} \subset \mathcal{I}$, and $\mathcal{A} \bigcap \mathcal{C} = \phi$. We refer to $\mathcal{A}$ as the antecedent of the rule, and $\mathcal{C}$ as the consequent of the rule. The rule $\mathcal{A} \Rightarrow \mathcal{C}$ has support $Supp(\mathcal{A} \Rightarrow \mathcal{C})$ in $\mathcal{D}$, where the ***support*** is defined as $Supp(\mathcal{A} \Rightarrow \mathcal{C}) = Supp(\mathcal{A} \bigcup \mathcal{C})$. That means $Supp(\mathcal{A} \Rightarrow \mathcal{C})$ percent of the transactions in $\mathcal{D}$ contain $\mathcal{A} \bigcup \mathcal{C}$, and $Supp(\mathcal{A}) = |\{\mathcal{T} \in \mathcal{D} | \mathcal{A} \subseteq \mathcal{T}\}| / |\mathcal{D}|$ is the support of $\mathcal{A}$ that is the fraction of transactions $\mathcal{T}$ supporting an item set $\mathcal{A}$ with respect to database $\mathcal{D}$. The number of transactions required for an item set to satisfy minimum support is referred to as the *minimum support count*. A transaction $\mathcal{T} \in \mathcal{D}$ supports an item set $\mathcal{A} \subseteq \mathcal{I}$ if $\mathcal{A} \subseteq \mathcal{T}$ holds. The rule $\mathcal{A} \Rightarrow \mathcal{C}$ holds in $\mathcal{D}$ with confidence $Conf(\mathcal{A} \Rightarrow \mathcal{C})$, where the ***confidence*** is defined as $Conf(\mathcal{A} \Rightarrow \mathcal{C}) = Supp(\mathcal{A} \bigcup \mathcal{C}) / Supp(\mathcal{A})$. That means $Conf(\mathcal{A} \Rightarrow \mathcal{C})$ percent of the transactions in $\mathcal{D}$ that contain $\mathcal{A}$ also contain $\mathcal{C}$. The confidence is a measure of the rule's strength or certainty while the support corresponds to statistical significance or usefulness.

Association rule mining generates all association rules that have a support greater than minimum support $min.Supp(\mathcal{A} \Rightarrow \mathcal{C})$, in the database, i.e., the rules are frequent. The rules must also have confidence greater than minimum confidence $min.Conf(\mathcal{A} \Rightarrow \mathcal{C})$, i.e., the rules are strong. The process of association rule mining consists of these two steps: 1) Find all frequent item sets, where each $\mathcal{A} \bigcup \mathcal{C}$ of these item sets must be at least as frequently supported as the minimum support count. 2) Generate strong rules from the discovered frequent item sets, where each $\mathcal{A} \Rightarrow \mathcal{C}$ of these rules must satisfy $min.Supp(\mathcal{A} \Rightarrow \mathcal{C})$ and $min.Conf(\mathcal{A} \Rightarrow \mathcal{C})$.

3.2 Rule-Ranking Strategy

Before prediction, we rank the discovered rules according to the length-first strategy. The length-first strategy was used for two reasons. First, for the defect association prediction, the length-first strategy enables us to find out as many defects as possible that coincide with known defect(s), thus preventing errors due to incomplete discovery of defect associations. Second, for the defect correction effort prediction, the length-first strategy enables us to obtain more-accurate rules, thus improving the effort-prediction accuracy.

Specifically, the length-first rule-ranking strategy is as follows:

1. Rank rules according to their length. The longer the rules, the higher the priority.
2. If two rules have the same length, rank them according to their confidence values. The greater the confidence values, the higher the priority. The more-confident rules have more predictive power in terms of accuracy; thus, they should have higher priority.
3. If two rules have the same confidence values, rank them according to their support values. The higher the support values, the higher the priority. The rules with higher support value are more statistically significant, so they should have higher priority.
4. If two rules have the same support value, rank them in alphabetical order.

The algorithmic description of the strategy is shown in Fig. 4.

3.3 Defect Association Prediction

As the first step of defect association prediction, we used the association rule mining method to find defect association rules from the defect data set. Although the discovery of defect association rules is straightforward, the implementation requires certain modifications to the data set. As Fig. 1 shows, 37.36 percent of cases in the defect data set contain only one defect. This means at least 37.36 percent of cases will not be correctly predicted. The reason is that the association rule mining method can only discover the rules with two or more defects. In order to predict the defects that occurred independently, we added a NULL to the transactions with only one defect. With this modification, the

```
Input: Rules – the defect association rules generated by the association rule mining
       Apriori algorithm.
Output: Rules – the ranked defect association rules by applying the proposed rule
       ranking length-first strategy, the most priori first.
for each rule r_i ∈ Rules do
  for each rule r_{j>i} ∈ Rules do
    if Length(r_i) < Length(r_j) then
      r_i ↔ r_j; // make more priori first
    else if Length(r_i) ≡ Length(r_j) then
      if Conf(r_i) < Conf(r_j) then
      // the priority depends on confidence values for rules with the same length
        r_i ↔ r_j;
      else if Conf(r_i) ≡ Conf(r_j) then
        if Supp(r_i) < Supp(r_j) then
        // the priority depends on support values for rules with the same length
        // and the same confidence value
          r_i ↔ r_j;
        else if Supp(r_i) ≡ Supp(r_j) then
          if AlphabetOrder(r_i) > AlphabetOrder(r_j) then
            r_i ↔ r_j;
          end if
        end if
      end if
    end if
    r_i ≺ r_j; // r_i has higher priority
  end for
end for
```

Fig. 4. Rule-ranking (RR) strategy.

association rule mining method is able to find rules like $Defect\{a\} \Rightarrow Defect\{NULL\}$, which means defect a occurred independently.

The next step is to predict whether or not a k-defect will occur with others. The prediction begins by ranking the discovered rules according to the strategy presented in Section 3.2. Then, for the k-defect, we scan the rules one by one and identify the rule whose antecedent contains the k-defect. After that, we merge the consequent of the corresponding rule with the k-defect and generate a $(k+1)$-defect. For the $(k+1)$-defect, we repeat the process until there are no rules available, the $\{(k+n)\text{-defect}\} \ominus \{k\text{-defect}\}$ is the defect(s) which occurred with the k-defect. Fig. 5 contains an algorithmic description of the procedure.

The measures used to evaluate the defect association prediction method are prediction accuracy, false-negative rate, and false-positive rate, which are defined as follows: Let $\mathcal{G}$ be a given original defect set, $\mathcal{R}$ be the real defect set, and $\mathcal{P}$ be the predicted defect set. The prediction *accuracy* of $\mathcal{P}$ is defined as:

$$Accuracy(\mathcal{P}) = \frac{|(\mathcal{R} \ominus \mathcal{G}) \cap (\mathcal{P} \ominus \mathcal{G})|}{|\mathcal{R} \ominus \mathcal{G}|}, \tag{1}$$

where, if $\mathcal{G} \equiv \mathcal{R} \equiv \mathcal{P}$ or $\mathcal{R} \subset \mathcal{P}$, $Accuracy(\mathcal{P}) = 1$.

We use the false-negative rate and false-positive rate to present the prediction error of defect associations. The *false-negative rate* $\mathcal{FN}$ denotes how many defects that are not predicted to occur along with the given set $\mathcal{G}$ but actually do, and is defined as follows:

$$\mathcal{FN}(\mathcal{P}) = \frac{|\mathcal{R} \ominus \mathcal{R} \cap \mathcal{P}|}{|\mathcal{R} \ominus \mathcal{G}|}, \tag{2}$$

where, if $\mathcal{G} \equiv \mathcal{R} \equiv \mathcal{P}$ or $\mathcal{R} \subset \mathcal{P}$, $\mathcal{FN}(\mathcal{P}) = 0$.

The *false-positive rate* $\mathcal{FP}$ denotes how many defects are predicted to occur along with the given set $\mathcal{G}$ but actually do not. It is defined as follows:

$$\mathcal{FP}(\mathcal{P}) = \frac{|\mathcal{P} \ominus \mathcal{R} \cap \mathcal{P}|}{|\mathcal{R} \ominus \mathcal{G}|}, \tag{3}$$

where, if $\mathcal{G} \equiv \mathcal{R} \equiv \mathcal{P}$ or $\mathcal{R} \supset \mathcal{P}$, $\mathcal{FP}(\mathcal{P}) = 0$.

3.4 Defect Correction Effort Prediction

Because the intent of defect correction effort[3] prediction is predetermined and the association rule mining has no special goal, the association rule mining based discovery of defect correction effort prediction rules is not straightforward. For

3. Here, the defect correction effort includes both defect isolation effort and defect correction effort.

Input: $Rules$ – the output of rule ranking RR;
$\mathcal{G}$ – a given k-defect set.
Output: $Associations$ – the predicted defect set that can be co-occurred with $\mathcal{G}$.
$PreAssoc \leftarrow \mathcal{G}$; // the candidate associations
for each rule $r \in Rules$ do
  if $\exists PreAssoc \in Antecedent(r)$ then
    $PreAssoc \leftarrow PreAssoc \cup Consequent(r)$; // combining
  end if
end for
$Associations \leftarrow Longest(PreAssoc) \ominus \mathcal{G}$;

Fig. 5. Defect association prediction procedure.

the purpose of defect correction effort prediction, we used the constraint-based association rule mining method [24]. Specifically, the procedure of association rule mining was adapted as follows:

1. Compute the frequent item sets that occur together in the training data set at least as frequently as a predetermined $min.Supp$. The item sets mined must also contain the effort labels.
2. Generate association rules from the frequent item sets, where the consequent of the rules is the effort. In addition to the $min.Supp$ threshold, these rules must also satisfy a minimum confidence $min.Conf$.

For the algorithmic description of the constraint-based association rule mining procedure, we refer readers to [24].

Once the rules are obtained, we rank them according to the approach presented in Section 3.2. Then, for each element of the given defect and its attributes, we scan the rules one by one and identify the rule whose antecedent contains the element. After that, we merge the antecedent of the corresponding rule with the element, generate an element set, and obtain the corresponding effort. For the element set, we continue the scan until no more rules fit. At this point, we have obtained the most likely candidate effort and the corresponding similarity for the element. We repeat the process until all the elements of the given defect and its attributes have been checked. Finally, we compare the similarities of all the candidates; the most similar candidate's effort is the defect correction effort for the given defect and its attributes. Fig. 6 is the algorithmic description which contains the details of the procedure.

As the defect correction effort is represented in categorical values and there are some attributes characterizing it, it can be viewed as a classification problem. Hence, we use the normal classification accuracy measure to evaluate the defect correction effort prediction method and the other three methods.

4 Experimental Results

In this section, we present the experimental results for the defect data set, the defect isolation effort data set, and the defect correction effort data set with different minimum-support thresholds and different minimum-confidence thresholds for association rule mining based defect association and defect correction effort predictions. We also

Input: $Rules$ – the output of rule ranking RR;
$dAttri$ – a defect and its attributes.
Output: Eft – the predicted effort for correcting defect $dAttri$.
$Sim \leftarrow 0$; $Eft' \leftarrow \emptyset$; //candidates
for each element $e \in dAttri$ do
  $Def_0 \leftarrow e$;
  for each rule $r \in Rules$
    if $\exists Def_0 \in Antecedent(r) \wedge Length(Def_0) < Length(dAttri)$ then
      $Def_0 \leftarrow Def_0 \cup Antecedent(r)$;
      $Eft_0 \leftarrow Consequent(r)$;
    end if
  end for
  $Eft' \leftarrow Eft' \cup Eft_0$;
  $Sim \leftarrow Sim \cup \frac{\|dAttri \cap Def_0\|}{\|dAttri\|}$;
end for
$Eft \leftarrow \{Eft' : Eft' \text{ with the } max\{Sim\}\}$;

Fig. 6. Effort prediction procedure.

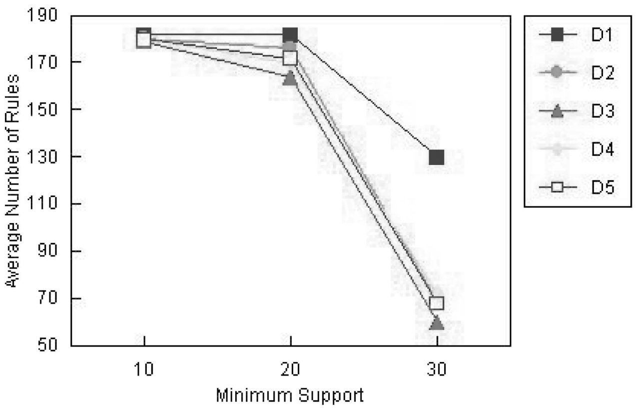

Fig. 7. Distribution of defect association rules by $Min.Supp$ for the five defect data sets.

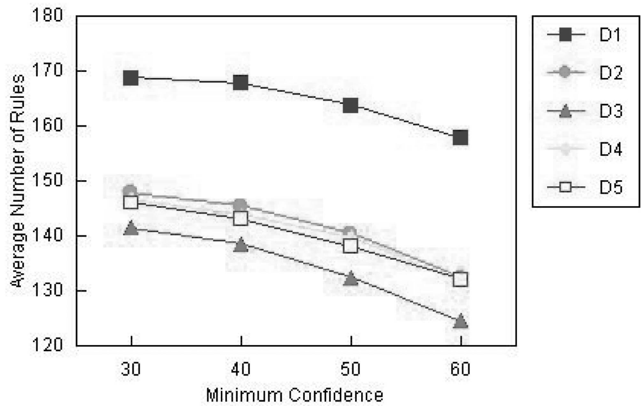

Fig. 8. Distribution of defect association rules by $Min.Conf$ for the five defect data sets.

compared the proposed methods with the other three methods if applicable. As five training and test sets were used, for each of the defect data set, the defect isolation effort data set, and the defect correction effort data set, we let the average accuracy of the corresponding five test sets be the final result.

4.1 Defect Association Prediction

4.1.1 Defect Association Rules

When mining defect association rules, we applied three values for minimum support (10, 20, and 30 percent) and four minimum confidence values (30, 40, 50, and 60 percent). This means a total of 12 cases were considered. At the same time, these 12 cases were applied to the five training data sets, which were derived from the defect data set by using the method presented in Section 2.2. This resulted in a total of 60 sets of rules, which consist of more than 1,000 rules. For this reason, we do not list them all (!), but instead present their typical forms and the statistical analysis of the results.

The typical forms of the defect association rules are listed below:

- $Defect\{DataValue\}$ $\Rightarrow Defect\{Null\}@(32.5\%, 79.9\%)$.
- $Defect\{Ex.Interface\}$ $\Rightarrow Defect\{Comput.\}@(39.0\%, 69.5\%)$.
- $Defect\{Comput.\} \wedge Defect\{Ini.\}$ $\Rightarrow Defect\{Ex.Interface\}@(34.3\%, 75.1\%)$.
- $Defect\{In.Interface\} \wedge Defect\{Ini.\}$ $\wedge\ Defect\{DataValue\}$ $\Rightarrow Defect\{Logi.Struc\}@(35.4\%, 94.8\%)$.
- $Defect\{Comput.\} \wedge Defect\{Ini.\}$ $\wedge\ Defect\{Logi.Struc\}$ $\wedge\ Defect\{DataValue\}$ $\Rightarrow Defect\{In.Interface\}@(31.4\%, 88.1\%)$.

The first is a one-defect rule, which means that defect $DataValue$ occurred independently in 32.5 percent of cases in the defect data set and, when occurring, it is independent with a probability of 79.9 percent. The second is a two-defect rule, which means these two defects can co-occur. This rule shows that 39 percent of the cases in the defect data set contain both defects $Ex.Interface$ and $Comput.$, and 69.5 percent of the cases in the defect data set that contain defects $Ex.Interface$ also contain defect $Comput.$ It reveals that defect $Comput.$ can co-occur with defect $Ex.Interface$ with a significance of 39 percent and a certainty of 69.5 percent. The third is a three-defect rule, which means these three defects can co-occur. This rule shows that 34.3 percent of the cases in the defect data set contain defects $Comput.$, $Ini.$, and $Ex.Interface$, and 75.1 percent of the cases in the defect data set that contain defects $Comput.$ and $Ini.$ also contain defect $Ex.Interface$. It reveals that defect $Ex.Interface$ can co-occur with defects $Comput.$ and $Ini.$ with a significance of 34.3 percent and a certainty of 75.1 percent. The fourth is a four-defect rule, which reveals that defect $Logi.Struc$ can co-occur with defects $In.Interface$, $Ini.$, and $DataValue$ with a significance of 35.4 percent and a certainty of 94.8 percent. The fifth is a five-defect rule, which reveals that defect $In.Interfacec$ can co-occur with defects $Comput.$, $Ini.$, $Logi.Struc$, and $DataValue$ with a significance of 31.4 percent and a certainty of 88.1 percent.

Fig. 7 is the distribution of defect association rules by minimum support ($min.Supp$) for all five training data sets. We observe that the overall behaviors of the training data sets are very similar except for the data set $D1$. Specifically, for each data set, the average number of rules decreases as the $min.Supp$ increases from 10 percent through 20 to 30 percent, and the decrease is very sharp when $min.Supp$ exceeds 20 percent.

Fig. 8 is the distribution of defect association rules by minimum confidence ($min.Conf$) again for all five training data sets. We observe that for each data set, the average number of rules decreases as $min.Conf$ increases from 30 percent through 40 percent and 50 to 60 percent, but the average number of rules for data set $D1$ is greater than other data sets at each confidence point.

Fig. 9 is the distribution of defect association rules by rule length. Comparing it with Fig. 1, we observe that these two distributions are dissimilar, especially for one-defect and two-defect. This is because there are only six types of defects. Thus, for one-defect, six rules can cover all cases and, for two-defect, the combinations of any two of six defects are also relative smaller, moreover, some defects may never occur together.

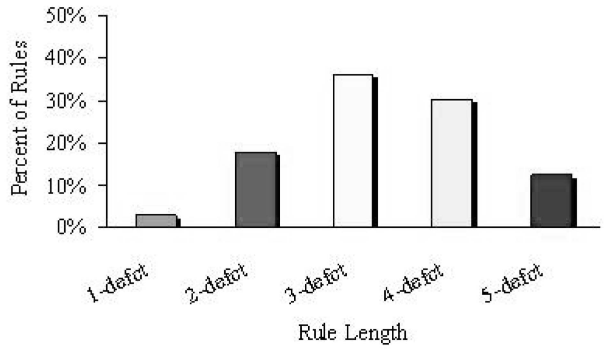

Fig. 9. Distribution of defect association rules by rule length for the defect data set.

TABLE 3
Defect Association Prediction Accuracy

Constrain	Accuracy	Error
min. Supp = 20% min. Conf = 30%	Mean = 96.59%	$\mathcal{FP}$ = 13.17% $\mathcal{FN}$ = 2.84%
	Median = 96.38%	
	Min = 95.83%	
	Max = 98.06%	

4.1.2 *Defect Association Prediction Results*

Table 3 contains the defect association prediction results with $min.Supp = 20\%$ and $min.Conf = 30\%$. We observe that the defect association prediction accuracy is very high and the false negative rate $\mathcal{FN}$ is very low. These are both desirable. The smaller the false negative rate $\mathcal{FN}$, the smaller the possibility that the defect associations are not identified. On the other hand, the false positive rate $\mathcal{FP}$ is rather high, but this guarantees that we can find more defects. However, the cost is increased testing effort.

Table 4 and Table 5 show the results of the impact of $min.Supp$ and $min.Conf$ on the defect association prediction accuracy, respectively. Table 4 shows that both the defect association prediction accuracy and the false-positive rate $\mathcal{FP}$ decrease, but the false-negative rate $\mathcal{FN}$ increases as $min.Supp$ increases from 10 to 30 percent. The defect association prediction accuracy with $min.Supp = 10\%$ is the same as $min.Supp = 20\%$ except the $\mathcal{FP}$ of the former is greater than the latter. This means decreasing $min.Supp$ has no further impact upon the prediction accuracy. In contrast, it will increase the $\mathcal{FP}$, which is something we wish to avoid.

Table 5 reveals that the defect association prediction accuracy decreases, but both $\mathcal{FP}$ and $\mathcal{FN}$ increase as $min.Conf$ increases from 30 to 60 percent. This means that increasing $min.Conf$ cannot improve the prediction accuracy further. In contrast, it will increase both $\mathcal{FP}$ and $\mathcal{FP}$.

The quality of defect association prediction depends on the rules, thus we also explored the impact of $min.Supp$ and $min.Conf$ on the number of rules and further on the prediction accuracy. Table 6 and Table 7 show the results. We observe that the total number of rules with different length decreases as $min.Supp$ increases from 10 to 30 percent. It seems that $min.Supp$ affects long rules more than short ones but $min.Conf$ affects short rules more than long rules, although the total number of rules with different length decreases as $min.Conf$ increases from 30 to 60 percent. In comparison with Table 3, we find that in order to obtain high prediction accuracy, a sufficient number of rules is a precondition.

TABLE 4
Defect Association Prediction Accuracy with Different Minimum Supports

min.	Accuracy			Error	
Support	Mean	Median	Std. Deviation	$\mathcal{FP}$	$\mathcal{FN}$
10%	91.90%	92.73%	4.43%	17.92%	2.84%
20%	91.90%	92.73%	4.43%	17.86%	2.84%
30%	85.35%	86.40%	6.12%	12.43%	8.11%

TABLE 5
Defect Association Prediction Accuracy with Different Minimum Confidences

min.	Accuracy			Error	
Confidence	Mean	Median	Std. Deviation	$\mathcal{FP}$	$\mathcal{FN}$
30%	94.42%	95.98%	4.03%	11.37%	4.59%
40%	92.44%	94.34%	3.74%	13.35%	4.59%
50%	88.17%	89.68%	4.15%	17.62%	4.59%
60%	83.83%	84.07%	5.10%	21.93%	4.63%

TABLE 6
Number of Defect Association Prediction Rules by Rule Length with Different Minimum Supports

min.	Rule Length					
Support	1	2	3	4	5	Total
10%	4.45	25.7	59.85	60	30	180
20%	4.45	25.7	59.85	59.4	23.2	172.6
30%	4.45	25.7	36.65	11.4	1.2	79.45

4.2 Defect Isolation Effort Prediction

4.2.1 *Defect Isolation Effort Prediction Rules*

Again, we have 12 cases to consider (minimum support values of 10, 20, and 30 percent combined with minimum confidence values of 10, 20, 30, and 40 percent). These 12 cases are applied to the five training data sets, which were derived from the defect isolation data set by using the method presented in Section 2.2. Again, this yields a total of 60 sets of rules, which total more than 1,000 rules.

The typical forms of the defect isolation effort prediction rules are listed as follows:

TABLE 7
Number of Defect Association Prediction Rules by Rule Length with Different Minimum Confidences

min.	Rule Length					
Confidence	1	2	3	4	5	Total
30%	7.0	29.0	52.27	43.6	18.13	150.07
40%	5.2	28.4	52.27	43.6	18.13	147.60
50%	3.2	25.6	52.27	43.6	18.13	142.80
60%	2.4	19.8	51.67	43.6	18.13	135.60

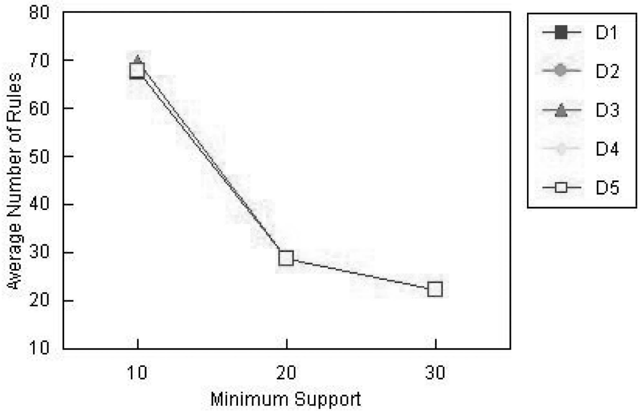

Fig. 10. Distribution of defect isolation effort prediction rules by $Min.Supp$ for the five defect isolation effort data sets.

- $Defect\{Ini.\} \Rightarrow Effort\{One\ Hour\ or\ Less\}$ $@(15.3\%, 64.1\%)$
- $Defect\{Comput.\} \wedge Attri\{TypoErr = N\}$ $\Rightarrow Effort\{One\ Day\ to\ Three\ Days\}$ $@(10.2\%, 10.5\%)$.
- $Defect\{Logic.Stru\} \wedge Attri\{TypoErr = N\}$ $\wedge\ Attri\{OmisErr = Y\}$ $\Rightarrow Effort\{One\ Hour\ to\ One\ Day\}$ $@(15.4\%, 40.1\%)$.
- $Defect\{DataValue\} \wedge Attri\{TypoErr = N\}$ $\wedge\ Attri\{OmisErr = N\} \wedge\ Attri\{ComisErr = Y\}$ $\Rightarrow Effort\{One\ Hour\ or\ Less\}@(10.6\%, 64.5\%)$.

The first rule contains only one antecedent. It shows that for an initialization defect, no matter what attributes are attached, the effort used to isolate it is One Hour or Less with a significance of 15.3 percent and a certainty of 64.1 percent. The second rule shows that 10.2 percent of the cases in the defect isolation effort data set contain defect $Comput.$, defect attribute $TypoErr = N$, and defect isolation effort One Day to Three Days, and 10.5 percent of the cases in the defect isolation effort data set that contain defects $Comput.$ and $TypoErr = N$ also contain defect isolation effort One Day to Three Days. This means that if a computational defect was not caused by a typographical error, we can say the effort used to isolate it is One Day to Three Days with a significance of 10.2 percent and a certainty of 10.5 percent. The third rule reveals that if a logical-structure defect was caused not by a typographical error but by an omission error, we can say the effort used to isolate it is One Hour to One Day with a significance of 15.4 percent and a certainty of 40.1 percent. The fourth rule reveals that if a data-value defect was caused neither by a typographical error nor by an omission error but by a commission error, we can say the effort used to isolate it is One Hour or Less with a significance of 10.6 percent and a certainty of 64.5 percent.

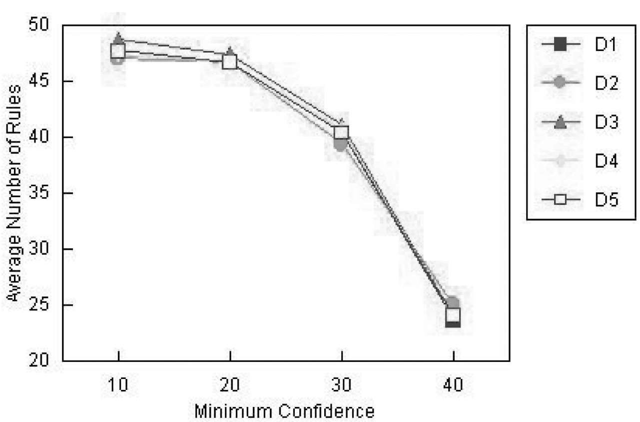

Fig. 11. Distribution of defect isolation effort prediction rules by $Min.Conf$ for the five defect isolation effort data sets.

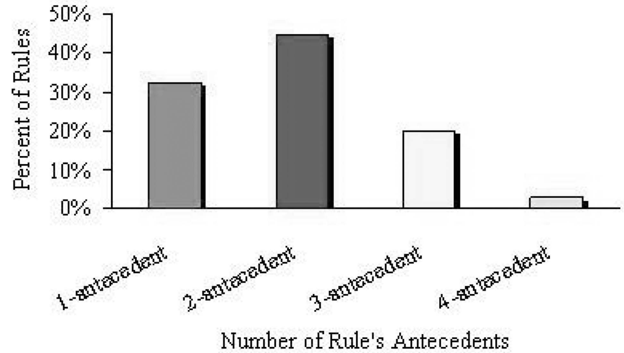

Fig. 12. Distribution of defect isolation effort prediction rules by rule length for the defect isolation effort data set.

Fig. 10 is the distribution of defect isolation effort prediction rules by minimum support $min.Supp$ for all five training data sets. We observe that, for each data set, the average number of rules decreases as the minimum support $min.Supp$ increases from 10 percent through 20 to 30 percent, and the decrease is very sharp when $min.Supp$ exceeds 10 percent but is less than 20 percent.

Fig. 11 is the distribution of defect isolation effort prediction rules by minimum confidence $min.Conf$ for all the five training data sets. We observe that, for each data set, the average number of rules decreases as the minimum confidence $min.Conf$ increases from 10 percent through 20 and 30 percent to 40 percent, and the decrease is sharp when $min.Conf$ exceeds 30 percent.

Fig. 12 is the distribution of defect isolation effort prediction rules by rule length. We observe that nearly half of the rules have two antecedents and more than 30 percent of rules have one antecedent. Thus, these rules constitute the main part of the rule set.

4.2.2 Defect Isolation Effort Prediction Results

Table 8 contains the defect isolation effort prediction results of four methods: association rule mining based method

TABLE 8
Defect Isolation Effort Prediction Accuracy

Method	Accuracy		
	Mean	Median	Std. Deviation
Apriori@(10,10)	93.80%	93.86%	0.35%
PART	68.55%	68.57%	0.20%
C4.5	68.26%	68.19%	0.16%
Naïve Bayes	67.98%	68.18%	0.49%

TABLE 9
Defect Isolation Effort Prediction Accuracy with Different Supports

min.	Accuracy		
Support	Mean	Median	Std. Deviation
10%	87.77%	92.34%	9.11%
20%	86.33%	92.26%	10.70%
30%	86.33%	92.26%	10.70%

(with $min.Supp = 10\%$ and $min.Conf = 10\%$), PART, C4.5, and Naïve Bayes. We observe that the accuracy of the association rule mining based method is higher than the other three methods by at least 25 percent. This means for the purpose of predicting defect isolation effort, association rule mining substantially outperforms the Bayesian probability, tree, and rule-based methods. The reason association rule mining based prediction performs so much better than other methods is that it explores high confidence associations among multiple variables and discovers interesting rules, i.e., rules that are useful, strong, and significant. By contrast, the other methods use domain independent biases and heuristics to generate a small set of rules, which results in many interesting and useful rules remaining undiscovered.

Table 9 and Table 10 are the results of the impact of $min.Supp$ and $min.Conf$ on the defect isolation effort prediction accuracy, respectively. Table 9 shows that the defect isolation effort prediction accuracy decreases as $min.Supp$ increases from 10 to 30 percent, and the accuracy with $min.Supp = 10\%$ is just a little better than $min.Supp = 20\%$. This means decreasing $min.Supp$ cannot greatly improve the prediction accuracy.

Table 10 shows that the defect isolation effort prediction accuracy decreases as $min.Conf$ increases from 9 to 40 percent, and the accuracy with $min.Conf = 10\%$ slightly higher than $min.Conf = 20\%$. This means decreasing $min.Conf$ doesn't greatly improve prediction accuracy.

Table 11 and Table 12 are the results of the impact of $min.Supp$ and $min.Conf$ on the number of defect isolation effort prediction rules, respectively. We observe that the total number of rules with different length decreases as both $min.Supp$ and $min.Conf$ increase. It seems that the conclusion that a sufficient number of rules is a precondition for high prediction accuracy also works in the context of defect isolation effort prediction.

TABLE 10
Defect Isolation Effort Prediction Accuracy with Different Confidences

min.	Accuracy		
Confidence	Mean	Median	Std. Deviation
10%	92.83%	92.45%	0.73%
20%	92.35%	92.34%	0.12%
30%	92.35%	92.34%	0.12%
40%	67.72%	68.42%	2.60%

TABLE 11
Number of Defect Isolation Effort Prediction Rules by Rule Length with Different Minimum Supports

min.	Rule Length				
Support	1	2	3	4	Total
10%	18.55	29.55	16.95	3.3	68.3
20%	11.75	13.45	3.50	0.0	28.7
30%	8.50	10.20	3.50	0.0	22.2

TABLE 12
Number of Defect Isolation Effort Prediction Rules by Rule Length with Different Minimum Confidences

min.	Rule Length				
Confidence	1	2	3	4	Total
10%	15.47	21.33	9.53	1.33	47.67
20%	15.33	20.93	9.20	1.33	46.80
30%	13.00	18.13	8.20	1.00	40.33
40%	7.93	10.53	5.00	0.73	24.13

4.3 Defect Correction Effort Prediction

4.3.1 Defect Correction Effort Prediction Rules

When mining defect correction effort prediction rules, we used the same configuration as mining defect isolation effort prediction rules. At the same time, the forms of defect correction effort prediction rules are also the same as those of the defect isolation effort prediction rules. Thus, we just present the statistical analysis results of the rules.

Fig. 13 is the distribution of defect correction effort prediction rules by minimum support $min.Supp$ for all the five training data sets. We observe that, for each data set, the average number of rules decreases as the minimum support $min.Supp$ increases from 10 percent through 20 to 30 percent, and the decrease between 10 and 20 percent of $min.Supp$ is very sharp.

Fig. 14 is the distribution of defect correction effort prediction rules by minimum confidence $min.Conf$ for all the five training data sets. We observe that, for each data set, the average number of rules decreases as the minimum confidence $min.Conf$ increases from 10 percent through 20 percent and 30 to 40 percent, and the decrease is quite sharp when $min.Conf$ exceeds 20 percent.

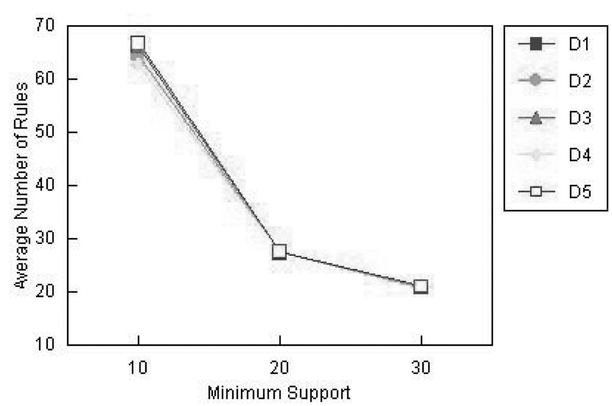

Fig. 13. Distribution of defect correction effort prediction rules by $Min.Supp$ for the five defect correction effort data sets.

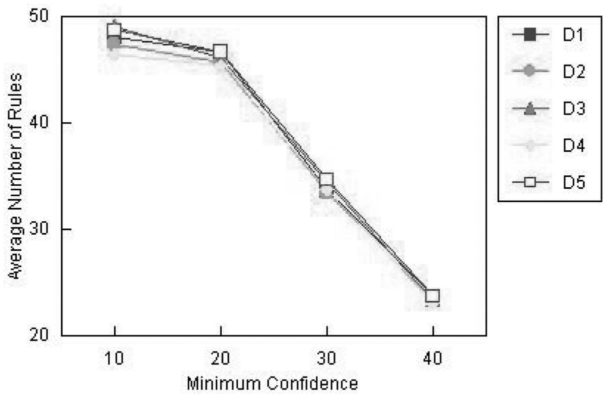

Fig. 14. Distribution of defect correction effort prediction rules by $Min.Conf$ for the five defect correction effort data sets.

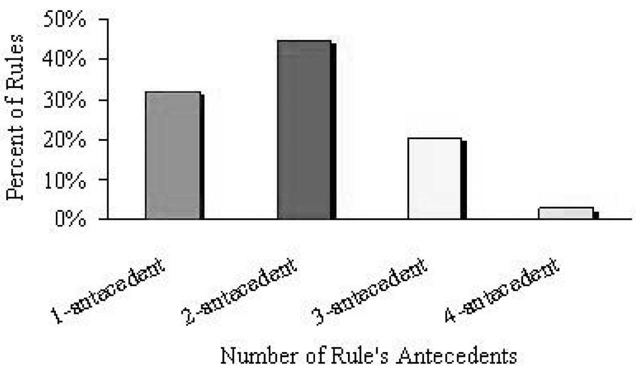

Fig. 15. Distribution of defect correction effort prediction rules by rule length for the defect correction effort data set.

Fig. 15 is the distribution of defect correction effort prediction rules by rule length. We observe that for defect isolation effort prediction, nearly half of the rules have two antecedents and more than 30 percent of rules have one antecedent. These rules constitute the main part of the rule set.

4.3.2 Defect Correction Effort Prediction Results

Table 13 contains the prediction results of four methods: association rule mining based method (with $min.Supp = 10\%$ and $min.Conf = 10\%$), PART, C4.5, and Naïve Bayes. We observe that the accuracy of the association rule mining based method is higher than the other three methods by at least 23 percent. This confirms the findings we obtained from defect isolation effort prediction.

We also have explored the impact of $min.Supp$ and $min.Conf$ on the prediction accuracy, Table 14 and Table 15, respectively, are the results. Table 14 shows that the prediction accuracy decreases as $min.Supp$ increases from 10 to 30 percent, and the accuracy with $min.Supp = 10\%$ is a little higher than with $min.Supp = 20\%$. This means decreasing $min.Supp$ does not greatly improve the prediction accuracy.

Table 15 shows that the defect correction effort prediction accuracy decreases as $min.Conf$ increases from 10 to 40 percent, and the accuracy with $min.Conf = 10\%$ is marginally higher than with $min.Conf = 20\%$. This means deceasing $min.Conf$ does not greatly improve the prediction accuracy.

Table 16 and Table 17, respectively, are the results of the impact of $min.Supp$ and $min.Conf$ on the number of defect correction effort prediction rules. We observe that the total number of rules with different length decreases as both $min.Supp$ and $min.Conf$ increase. It further supports the conclusion that a sufficient number of rules is a precondition for the high prediction accuracy we obtained in the context of defect isolation effort prediction.

TABLE 13
Defect Correction Effort Prediction Accuracy

Method	Accuracy		
	Mean	Median	Std. Deviation
Apriori@(10,10)	94.69%	94.85%	0.45%
PART	71.93%	71.88%	0.51%
C4.5	71.66%	71.40%	0.50%
Naïve Bayes	71.07%	70.98%	0.73%

TABLE 14
Defect Correction Effort Prediction Accuracy with Different Supports

min. Support	Accuracy		
	Mean	Median	Std. Deviation
10%	87.49%	92.67%	9.55%
20%	86.89%	92.58%	9.20%
30%	86.62%	92.43%	9.18%

TABLE 15
Defect Correction Effort Prediction Accuracy with Different Confidences

min. Confidence	Accuracy		
	Mean	Median	Std. Deviation
10%	93.44%	92.94%	0.96%
20%	92.85%	92.83%	0.15%
30%	90.09%	91.84%	2.74%
40%	71.62%	71.37%	0.47%

TABLE 16
Number of Defect Correction Prediction Rules by Rule Length with Different Minimum Supports

min. Support	Rule Length				Total
	1	2	3	4	
10%	17.15	28.10	16.85	3.05	65.15
20%	11.15	13.00	3.25	0.00	27.40
30%	7.90	9.75	3.25	0.00	20.90

TABLE 17
Number of Defect Correction Prediction Rules by Rule Length with Different Minimum Confidences

min. Confidence	Rule Length				Total
	1	2	3	4	
10%	15.33	21.27	10.2	1.07	47.87
20%	15.00	20.60	9.13	1.33	46.07
30%	10.27	15.47	7.20	1.00	33.93
40%	7.67	10.47	4.60	0.67	23.40

5 Conclusions

In this paper, we have presented an application of association rule mining to predict software defect associations and defect correction effort with SEL defect data. This is important in order to help developers detect software defects and project managers improve software control and allocate their testing resources effectively. The ideas have been tested using the NASA SEL defect data set. From this, we extracted defect data and the corresponding defect isolation and correction effort data.

For each of the three data sets, we randomly generated five training data sets and a corresponding five test data sets. After that, we applied the association rule mining method. The results show that for the defect association prediction, the minimum accuracy is 95.38 percent, and the false negative rate is just 2.84 percent; and for the defect correction effort prediction, the accuracy is 93.80 percent for defect isolation effort prediction and 94.69 percent for defect correction effort prediction.

We have also compared the defect correction effort prediction method with three other types of machine learning methods, namely, PART, C4.5, and Naïve Bayes. The results show that for defect isolation effort, our accuracy is higher than for the other three methods by at least 25 percent. Likewise, for defect correction effort prediction, the accuracy is higher than the other three methods by at least 23 percent.

We also have explored the impact of support and confidence levels on prediction accuracy, false negative rate, false positive rate, and the number of rules as well. We found that higher support and confidence levels may not result in higher prediction accuracy, and a sufficient number of rules is a precondition for high prediction accuracy.

While we do not wish to draw strong conclusions from a single data set study, we believe that our results suggest that association rule mining may be an attractive technique to the software engineering community due to its relative simplicity, transparency, and seeming effectiveness in constructing prediction systems.

Acknowledgments

The authors would like to thank the NASA/GSFC Software Engineering Laboratory (SEL) for providing the defect data for this analysis. The authors specially thank the four anonymous reviewers for their insightful and helpful comments, which resulted in substantial improvements to this work. The authors also thank Rahul Premraj for his comments.

References

[1] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," *Proc. ACM SIGMOD Conf. Management of Data,* May 1993.

[2] K. Ali, S. Manganaris, and R. Srikant, "Partial Classification Using Association Rules," *Proc. Third Int'l Conf. Knowledge Discovery and Data Mining,* pp. 115-118, 1997.

[3] I.S. Bhandari, "Attribute Focusing: Machine-Assisted Knowledge Discovery Applied to Software Production Process Control," *Proc. Workshop Knowledge Discovery in Databases,* July 1993.

[4] I.S. Bhandari, M. Halliday, E. Tarver, D. Brown, J. Chaar, and R. Chillarege, "A Case Study of Software Process Improvement During Development," *IEEE Trans. Software Eng.,* vol. 19, no. 12, pp. 1157-1170, 1993.

[5] I.S. Bhandari, M.J. Halliday, J. Chaar, R. Chillarenge, K. Jones, J.S. Atkinson, C. Lepori-Costello, P.Y. Jasper, E.D. Tarver, C.C. Lewis, and M. Yonezawa, "In Process Improvement through Defect Data Interpretation," *IBM System J.,* vol. 33, no. 1, pp. 182-214, 1994.

[6] L.C. Briand, K. El-Emam, B.G. Freimut, and O. Laitenberger, "A Comprehensive Evaluation of Capture-Recapture Models for Estimating Software Defect Content," *IEEE Trans. Software Eng.,* vol. 26, no. 6, pp. 518-540, June 2000.

[7] T. Compton and C. Withrow, "Prediction and Control of Ada Software Defects," *J. Systems and Software,* vol. 12, pp. 199-207, 1990.

[8] G. Dong, X. Zhang, L. Wong, and J. Li, "CAEP: Classification by Aggregating Emerging Patterns," *Proc. Second Int'l Conf. Discovery Science,* pp. 30-42, 1999.

[9] N.B. Ebrahimi, "On the Statistical Analysis of the Number of Errors Remaining in a Software Design Document after Inspection," *IEEE Trans. Software Eng.,* vol. 23, no. 8, pp. 529-532, 1997.

[10] K. El-Emam and O. Laitenberger, "Evaluating Capture-Recapture Models with Two Inspectors," *IEEE Trans. Software Eng.,* vol. 27, no. 9, pp. 851-864, 2001.

[11] N.E. Fenton and M. Neil, "A Critique of Software Defect Prediction Models," *IEEE Trans. Software Eng.,* vol. 25, no. 5, pp. 676-689, 1999.

[12] N.E. Fenton and S.L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach,* second ed. Int'l Thomson Computer Press, 1996.

[13] E. Frank, L. Trigg, G. Holmes, and I.H. Witten, "Naïve Bayes for Regression," *Machine Learning,* vol. 41, no. 1, pp. 5-25, 2000.

[14] E. Frank and I.H. Witten, "Generating Accurate Rule Sets without Global Optimization," *Proc. 15th Int'l Conf. Machine Learning,* pp. 144-151, 1998.

[15] G. Heller, J. Valett, and M. Wild, "Data Collection Procedure for the Software Engineering Laboratory (SEL) Database," Technical Report SEL-92-002, Software Eng. Laboratory, 1992.

[16] G.Q. Kenney, "Estimating Defects in Commercial Software During Operational Use," *IEEE Trans. Reliability,* vol. 42, no. 1, pp. 107-115, 1993.

[17] B. Liu, W. Hsu, and Y. Ma, "Integrating Classification and Association Rule Mining," *Proc. Fourth Int'l Conf. Knowledge Discovery and Data Mining,* pp. 80-86, 1998.

[18] J.C. Munson and T.M. Khoshgoftaar, "Regression Modelling of Software Quality: An Empirical Investigation," *Information and Software Technology,* vol. 32, no. 2, pp. 106-114, 1990.

[19] F. Padberg, T. Ragg, and R. Schoknecht, "Using Machine Learning for Estimating the Defect Content after an Inspection," *IEEE Trans. Software Eng.,* vol. 30, no. 1, pp. 17-28, 2004.

[20] J.R. Quinlan, *C4.5 Programs for Machine Learning.* San Mateo, Calif.: Morgan Kaufmann, 1993.

[21] P. Runeson and C. Wohlin, "An Experimental Evaluation of an Experience-Based Capture-Recapture Method in Software Code Inspections," *Empirical Software Eng.,* vol. 3, no. 3, pp. 381-406, 1998.

[22] SEL Database, http://www.cebase.org/www/frames.html?/www/Resources/SEL/, 2005.

[23] R. She, F. Chen, K. Wang, M. Ester, J.L. Gardy, and F.L. Brinkman, "Frequent-Subsequence-Based Prediction of Outer Membrane Proteins," *Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* 2003.

[24] R. Srikant, Q. Vu, and R. Agrawal, "Mining Association Rules with Item Constraints," *Proc. Third Int'l Conf. Knowledge Discovery and Data Mining (KDD '97),* pp. 67-73, Aug. 1997.

[25] K. Wang, S.Q. Zhou, and S.C. Liew, "Building Hierarchical Classifiers Using Class Proximity," *Proc. 25th Int'l Conf. Very Large Data Bases,* pp. 363-374. 1999.

[26] K. Wang, S. Zhou, and Y. He, "Growing Decision Tree on Support-Less Association Rules," *Proc. Sixth Int'l Conf. Knowledge Discovery and Data Mining,* 2000.

[27] S.V. Wiel and L. Votta, "Assessing Software Designs Using Capture-Recapture Methods," *IEEE Trans. Software Eng.* , vol. 19, no. 11, pp. 1045-1054 1993

[28] C. Wohlin and P. Runeson, "Defect Content Estimations from Review Data," *Proc. 20th Int'l Conf. Software Eng.,* pp. 400-409, 1998.

[29] Q. Yang, H.H. Zhang, and T. Li, "Mining Web Logs for Prediction Models in WWW Caching and Prefetching," *Proc. Seventh ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* Aug. 2001.
[30] X. Yin and J. Han, "CPAR: Classification Based on Predictive Association Rules," *Proc. 2003 SIAM Int'l Conf. Data Mining,* May 2003.
[31] A.T.T. Ying, C.G. Murphy, R. Ng, and M.C. Chu-Carroll, "Predicting Source Code Changes by Mining Revision History," *Proc. First Int'l Workshop Mining Software Repositories,* 2004.
[32] T. Zimmermann, P. Weigerber, S. Diehl, and A. Zeller, "Mining Version Histories to Guide Software Changes," *Proc. 26th Int'l Conf. Software Eng.,* 2004.

Qinbao Song received the PhD degree in computer science from Xi'an Jiaotong University, China, in 2001. He is an associate professor of software technology at Xi'an Jiaotong University, China. He has published more than 50 referred papers in the areas of data mining, machine learning, and software engineering. His research interests include intelligent computing, machine learning for software engineering, and trustworthy software.

Martin Shepperd received the PhD degree in computer science from the Open University in 1991 for his work in measurement theory and its application to software engineering. He is professor of software technology at Brunel University, London, and director of the Brunel Software Engineering Research Centre (B-SERC). He has published more than 90 refereed papers and three books in the areas of empirical software engineering, machine learning and statistics. He is editor-in-chief of the journal *Information and Software Technology* and was associate editor of *IEEE Transactions on Software Engineering* (2000-2004). He has also worked for a number of years as a software developer for a major bank.

Michelle Cartwright was awarded a first-class honors degree in computer science from the University of Wolverhampton in 1992 and a doctorate in computer science from Bournemouth University in 1998. She has published a number of papers in the areas of empirical software engineering, case-based reasoning, and estimation for software projects. She is presently a lecturer in the School of Information Systems, Computing, and Mathematics at Brunel University, London, and a member of the Brunel Software Engineering Research Centre (B-SERC).

Carolyn Mair is completing her PhD in cognitive neuroscience. She is a research assistant in the Brunel Software Engineering Research Centre (B-SERC) at Brunel University. Her current interests are software engineering metrics and software project prediction.