University of Alberta

Semantic-based Analysis of IF-THEN Rules

by

Aashima Kapoor ©

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering

Edmonton, Alberta
Spring 2008

Your file Votre référence
ISBN: 978-0-494-45828-0
Our file Notre référence
ISBN: 978-0-494-45828-0

ABSTRACT

THESIS TITLE – "Semantic-based Analysis of IF-THEN Rules"

Software quality is an important phase in the software life cycle. It focuses on keeping the software fully functional and up to date. Engineers use different approaches and methods to gain understanding of various software quality factors so that software systems can be maintained effectively and sustain to high performance. A lot of efforts have been put into finding a way to measure quality factors viz. maintainability, complexity, reliability of software. It is a common opinion that software quality should be described using a set of measurable software attributes.

This thesis looks at the issue of rule-based description of attributes of software with different levels of software quality factors. Varieties of rules are extracted from a data set that represents human evaluation of software quality of software objects. The usefulness of rules is measured and interpreted using semantic aspect of rules. Rule similarity and rule inclusion measures are used to identify the most diverse sets of rules representing human evaluation criteria. Additionally, an automated system is developed to represent the rules and analyze them using a semantic-based rule similarity and inclusion concept in order to obtain the most interesting and much generalized rules that best define the human evaluation criteria.

Acknowledgements

I would like to express my sincere thanks and gratitude to my thesis supervisor, Dr. Marek Reformat. It was only with his eternal guidance and support that I was able to materialize and complete this presented research work. And many thanks to all those who helped me in this work.

TABLE OF CONTENTS

List of Tables

List of Figures

1. INTRODUCTION – "Software quality matters"

1.1 Context and Motivation

Software quality assessment is becoming increasingly important. These days software spreads from computers to the engines of automobiles to robots in factories to X-ray machines in hospitals. This is where software quality matters that applications must be foolproof, and - whether embedded in the engine of a car, a robotic arm in a factory or a healing device in a hospital – poorly deployed code can be lead to lost of lives.

In addition to this, the complexity caused by the emergence of new software design and development paradigms makes the task more critical and more difficult. In this respect, it has become important to develop tools that allow assessing the software quality way early in the software life cycle.

Nowadays, data mining models are being used to estimate the quality of software. This thesis, proposes a method for analysis of association rules built on software data. Knowledge from these association rules is essential for effective decision making in software development life cycle. The proposed method, deals with generating the association rules on large set of data representing software attributes, and finding interesting and manageable number of rules among these discovered association rules. The system will facilitate the understanding of large amounts of data, gathered from various software packages, by discovering interesting regularities. The interesting rules identified will help software engineers or domain experts to predict the quality factors viz. maintainability, complexity, reliability or any such behavior of the software.

The biggest stimulus to carry with this work was the power and easiness of the "rules". There are two approaches to analyze a given set of attributes and classify them into a class hierarchy – black-box and white-box. Rule induction is an example of white-box approach where the relationship between the input attributes and the output class is very comprehensible. On the other hand other methods like Decision Trees, Neural Networks use black-box approach. They simply predict the class of the attributes without showing any details on how these predictions were made. In general, a rule-based approach is relatively easy to generate, easy to understand as it gives us insight into how exactly the rules are used to make classifications. The next major motivation was to conquer the biggest problem faced by domain experts while dealing with rule induction methods i.e. it is very difficult for domain experts to analyze large number of rules generated from large datasets, and find the ones which are unique and the most meaningful. Although several solutions have been proposed in the studies on data mining and knowledge discovery, these studies are not focused on similarities between rules obtained. The syntactic and semantic comparisons of the generated rules result in information that is helpful in increasing the efficacy of decisions made by software experts.

1.2 <u>Goals</u>

- Generation of rules

  Generation of rules can be considered as a vital goal in rule induction approaches. The first goal to be achieved was to understand the various rule-generating tools and generate abundant rules to analyze and interpret further. It is important to know exactly how various tools generate rules, how these tools evaluate rules and whether the result is what we desire.

- Analyzing and Interpretation of Rules

Understanding the importance and goodness of rules is very imperative. As described earlier it is possible to generate large number of rules on a given set of data but the real challenge is to generalize those rules and to come up with a set of best rules. The goal was to identify a set of most suitable approaches for rule analysis.

- Rule Evaluation Method

  Several methods of evaluating the rules were implemented. Finally the goal was to develop an automated system to find the best set of rules out of large sets of discovered rules. This was accomplished by following a very rigorous approach and was the most crucial step of my research work. Application of this system to a set of rules derived from a given dataset will result into a small set of best rules able to classify new data points into a class hierarchy.

1.3 Contributions

Each work performed in the software research industry is incomplete without its contributions. The following list holds the key contributions of my research:

- The main contribution is to propose and implement an automated system that searches for rules that not only possess high accuracy but are also unique and interesting in a sense to a user or application. It is a very adaptable system as engineers could take independent modules from it and merge them with their own code with ease.
- Another important contribution comes in the form of methodology developed for finding similar and highly inclusive rules. The method implemented in this research work will eliminate all those rules that may repeat themselves in other

rules and surpass more promising and interesting rules. It is a flexible approach that has been validated on the outcome of two rule-generating tools viz. See5 (initially C 4.5) and 4C Rule Builder.

- Much work has been conducted to find interesting patters or rules in medical domains or critical care databases. But evidently, not much work could be seen to evaluate association or classification rules in Software Quality domain. Hence, my work offers a state-of-art approach to appraise software quality using rule-induction.
- The approach has been applied to real-world data collected on software systems from National Research Council (NRC), Canada, and NASA, USA.
- Last but not the least, a part of completed research work was published in 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06) pp. 723-731. Marek Reformat, Aashima Kapoor, Nicolino J. Pizzi. "Software Maintenance: Similarity and Inclusion of Rules in Knowledge Extraction," http://doi.ieeecomputersociety.org/10.1109/ICTAI.2006.106

1.4 Thesis Overview

The presented research work aims at developing methods that contribute towards the ongoing research of finding interesting rules, and at addressing the issues of evaluating software quality using rules. The thesis is organized in the following way. Section 2 is dedicated to background and review of related literature and work done in the area of rule comparison. The concepts of generation of rules and similarity and inclusion of rules are presented in this section. Section 3 represents the basic methodology of the research work to obtain the set of interesting rules. Section 4 discusses the feasibility study in detail with the data description, methodology and experimental evaluations. Next, Section 5 gives a detailed description of building the automated system for finding the most relevant and important rules. This is subsequently followed by Section 6 that shows the results of experiments conducted on real-world software engineering datasets. The summary and conclusions constitute Section 7.

2. BACKGROUND AND RELATED WORK

Association rule is an efficient data mining technique that focuses on discovering patterns and knowledge from large databases. An association rule [18] is a rule of the form $\alpha \rightarrow \beta$, where α and β represent itemsets which do not share common items. The association rule $\alpha \rightarrow \beta$ holds in the transaction set L with confidence c, $c = |\alpha \cup \beta| / |\alpha|$, if c% of transactions in L that contain α also contain β. The rule $\alpha \rightarrow \beta$ has support s, $s = |\alpha \cup \beta|/|L|$, if s% of transactions in L contain $\alpha \cup \beta$. Here, we call α antecedent, and β consequent. Confidence gives a ratio of the number of transactions that the antecedent and the consequent appear together to the number of transactions the antecedent appears. Support measures how often the antecedent and the consequent appear together in the transaction set.

2.1 Generation of Association Rules

The importance of discovering association rules as a tool for knowledge discovery in databases has recently been recognized. Association rules represent a promising technique to find hidden patterns in a given data-set.

Generating rules in Data Mining is all about extracting patterns from an organization's stored or warehoused data. These patterns can be used to gain insight into the aspects of an organization's operations, and to predict the outcomes for future situations as an aid to decision-making.

Following the easiness to understand rules, two different data mining tools for generating rules were used, namely **SEE 5** and **4C Rule Builder.** See5 and 4C are sophisticated data mining tools for discovering patterns that delineate categories, assembling them into classifiers, and using them to make predictions.

SEE 5 or C5

The See5 algorithm (Quinlan, 1997) is the latest version of the ID3 and C4.5 algorithms developed in the last two decades. The criterion employed in See5 algorithm to carry out the partitions is based on some concepts from Information Theory and has been improved significantly over time. The main idea shared with similar algorithms is to choose that variable that provides more information to realize the appropriate partition in each branch in order to classify the training set.

See5 generates a text file containing a rule-set, used for classifying (predicting) each record in a dataset, into a discrete set of pre-determined classifications viz. {Good, Bad}, {High, Medium, Low}, etc.

See5 uniquely addresses the issue of comprehensibility. It finds patterns that provide insight in addition to supporting accurate predictions. In line with this approach, See5 emphasizes rule-based classifiers because they are easier to understand -- each rule can be examined and validated separately, without having to consider it in the context of the classifier as a whole.

An important feature of See5 is its ability to generate classifiers called rulesets that consist of unordered collections of (relatively) simple if-then rules. The Rulesets option causes classifiers to be expressed as rulesets rather than decision trees, giving the following rules:

```
Rule 1: (31, lift 42.7)
        thyroid surgery = f
        TSH > 6
        TT4 <= 37
        ->  class primary  [0.970]

Rule 2: (63/6, lift 39.3)
        TSH > 6
        FTI <= 65
        ->  class primary  [0.892]
```

4C Rule Builder

4C Rule Builder is an advanced tool that performs tasks like pattern recognition, prediction, decision making etc. It has a user friendly interface and is designed very intuitively with a windows-based interface providing easy to use options. 4C is capable of generating models for large datasets consisting of thousands of data points described by hundreds of features. The data models generated by 4C are in the form of simple IF-THEN rules that are compact, can be verified and comprehended. One of the important characteristic of 4C is that it provides deep feature and selector rankings that can be used for importance analysis and feature selection. Like See5 it works with any kind of data be it numeric discrete, numeric continuous and nominal features. This data mining tool has built in features to deal with missing values, noisy data and difficult datasets that describe only a small part of feature space.

Here is an example of the output generated by 4C to explain how the rules generated look like:

Rules Generated:

RULE1

IF **job = yes** and **area = OK** and **25.5<age<66.5** and **deposited != (44.50;49.50)** and **1.50<monthly loan payment<55.0** and **payment months != 30,4** THEN ***credit granted***

RULE2

IF **item purchased = (pc,stereo,jewelry,bike)** and **area = ok** and **17.5<age<66.5** and **deposited !=(44.50,49.50)** and **monthly loan payment= (0.5; 3.5) or (4.5;6.5)** THEN ***credit granted***

Usefulness ranking of the data:

Useful attributes: deposited, monthly loan payment, payment months, area, age, job, item purchased

Useless attributes: years with company, married, sex

Most useful selectors: age = (25.50,66.50), deposited = (0.5,44.5) and (49.5; 550), monthly loan payment= (1.50,3.50) and (4.50,6.50), area = ok

2.2 <u>Concept of Semantic & Syntactic Similarities</u>

A rule generation process can be performed using a number of approaches and methods. Each of them leads to generation of different rules. The rules differ in a number of ways. If we assume that an atomic component is built in the following way: value of attribute '*a*' is larger then number n, then we can identify a variety of differences among rules: rules can have different number of atomic components in antecedents, different attributes and their values, and different values of conditions. All these differences are related to the syntax of rules. We would like to focus on a different kind of comparison among rules – the semantic similarity. This similarity "looks inside" rules and compares them based on a number and identity of points that satisfy the antecedent parts of rules.

2.2.1 <u>Tversky's Similarity Model</u>

The two main approaches for assessing similarity are content models and distance models. Content models conceptualize the characteristics with respect to which objects are similar "as more or less discrete and common elements" [3]. Distance models conceptualize these characteristics "as dimensions on which the objects have some degree of proximity" [3]. Many set theoretic measures in the content model category are generalized by Tversky's parameterized ratio model of similarity [17]. It expresses similarity between objects as a ratio of the measures of their common and distinctive features:

$$S(X,Y)=\frac{f(X\cap Y)}{f(X\cap Y)+\alpha*f(X-Y)+\beta*f(Y-X)} \qquad (1)$$

X and Y represent sets describing respective objects x and y. (X ∩ Y) represents the features that objects x and y have in common. (X - Y) represents the features that x has but y does not. (Y - X) represents the features that y has, but x does not. The function f measures the contribution of any specific feature to the similarity between objects. The value f (X) for object x is considered a measure of the overall salience of that object. Factors adding to an object's salience include "intensity, frequency, familiarity, good form, and informational content" [17]. The function f is additive on disjoint sets, for example, set cardinality. The factors α and β are nonnegative valued unbounded parameters specifying the importance of these two components. This measure is normalized, $0 \le S(X, Y) \le 1$. For α=β=1, S becomes the Jaccard index:

$$S-jaccard(X,Y)=\frac{f(X\cap Y)}{f(X\cup Y)} \qquad (2)$$

With α=1, β=0, S becomes the degree of inclusion for X, the proportion of X overlapping with Y.

$$S-inclusion(X,Y)=\frac{f(X\cap Y)}{f(X)} \qquad (3)$$

Similarly with α=0, β=1, S becomes the degree of inclusion for Y, the proportion of Y overlapping with X. This parameterization is not necessary, however, since equation (4) can be formulated as S-inclusion (Y, X).

2.2.2 Tversky's Model for Similarity & Inclusion of Rules

Presented Tversky's model of similarity can be easily applied for comparison of rules. In order to compare the rules semantically the contingency table is used. An example of such a table is presented below, Table 1.

Table 1. A contingency table for rules R1 and R2

	R1	**no R1**
R2	a	b
no R2	c	d

The number a represents a number of data points that are satisfied by antecedents of both rules R1 and R2. The number b identifies a number of data points that satisfy the rule R2 but do not the rule R1, while the number c opposite – a number of points that satisfy the rule R1 but do not R2. The numbers a, b and c can be directly plug in into the equation (2). The equation is presented below

$$S-jacarrd(R1,R2)=\frac{a}{a+b+c} \qquad (4)$$

Tversky's model allows us to calculate inclusions of rules. The formulas can be presented in the following forms:

$$S-inclusion(R1 \text{ in } R2)=\frac{a}{a+c} \qquad (5)$$

$$S-inclusion(R2 \text{ in } R1)=\frac{a}{a+b} \qquad (6)$$

2.3 Related Work

There is an ongoing research focused on finding a single metric that estimates the importance, interestingness or quality of association rules. The biggest problem remains that association rule algorithms generate large number of rules to be analyzed and ranked for their importance. Researchers have been finding different ways to ignore and eliminate redundant and unimportant rules. This is a generic problem in data mining [19]. [19] discusses the subject of interestingness of discovered knowledge in general.

In this study of finding such important and useful association rules, we propose a new semantic approach strictly based on Teversky's Similarity Model (Section 2.2). Large number of rules are generated and then compared based on their semantic similarities and inclusion of rules is used to eliminate redundant rules.

Similarity measures other than Teversky's (Jaccard coefficient of similarity) have also been looked upon and applied. [23] uses a combination of accuracy and coverage to measure the semantic similarity between two rules.

Kulczynski	$\frac{1}{2}\left(\frac{a}{a+b}+\frac{a}{a+c}\right)$
Ochiai	$\frac{a}{\sqrt{(a+b)(a+c)}}$
Simpson	$\frac{a}{\min\{(a+b),(a+c)\}}$

where, a/(a + b) is the accuracy and a/(a + c) is the coverage.

[23] emphasizes on comparing the rules based on any of these similarity measures and then grouping the similar rules. It uses multidimensional scale and assigns a point for each rule with its distance information. The main focus of this research work is to find and visualize groups of similar rules. They claim that the final visualization can provide some insight to domain experts about the rules. However, the work does not distinguish any rules as weak or strong rules. They propose visualization as the way of managing large number of rules.

Another work by Jiye Li [24] introduces a rough set based rule importance measure to select most important rules. Rules that are generated more frequently than others among the total rule set are considered to be more important. Importance of a rule is defined as the frequency of the association rule generated across all the rule sets. In one of the other works, Alex A. Freitas [27] discusses several factors related to rule-interestingness. He introduces a new rule interestingness measures "attribute surprisingness" alongside talking about other factors like coverage, confidence, completeness of rules, attribute costs, misclassification costs etc. Unfortunately, these papers do not consider semantic similarity between two rules as an important rule-interestingness measure.

A different approach towards finding interesting rules from large sets of discovered association rules has been discussed in [28]. They use *rule templates* (regular expressions) to describe the structure of interesting rules. This is a syntactic based approach where templates describe a set of rules by specifying what attributes occur in the antecedent and what attribute is the consequent. The discovered rules that tend to match these templates are termed as interesting rules. One of the major uncovered problem this paper mentions in its work is redundancy. They talk about how eliminating redundant rules can benefit the whole working of finding interesting rules. To remove redundant rules is one of the accomplished tasks of my presented work. Thus, the completed research work will definitely prove beneficial to the research industry in this direction.

From all these works it is also clear that there are no specific ways and terms to define the interestingness or importance of rules. These definitions may vary depending on the application, domain expert analyzing these rules or even the user. Although there exist various ways of finding interesting rules, but very little effort has been put forward to apply '*Similarity Models*' to uncover useful and important rules. Also, no research work could be found where these similarity models have been applied to software data.

The proposed study aims at using these similarity models on software quality data to measure or predict SQ factors. One such software quality factor is Software Maintainability. Engineers are still on their progress path of finding a single technique that could estimate the level of maintainability of software. The most popular one seems to be the Maintenance Index [15]. However, in many cases, the way to find out about software maintainability level is to ask programmers or maintenance engineers to estimate maintainability of objects via visual inspection. Such an approach means that person's individual experience and background have impact on results of maintainability evaluations.

3 METHODOLOGY – SELECTING INTERESTING RULES

The process of find "interesting" rules highly depends on the application or the user. Each researcher can have his own way of defining and calculating the interestingness of association rules. The Semantic based similarity model that we use to achieve the most interesting rules is based on two important characteristics of rules viz. how "important" a rule is; and how "unique" it is out of the large set of rules discovered. These two factors form the base of the interestingness measure used in this research work to find the best set of rules. The Importance and Uniqueness of rules along with how they define interesting rules is described below:

Importance of Rules. One of important ways of evaluating a rule is to numerically represent its classification capability. In other words, this means calculating "goodness" of rules. The "goodness" can be calculated in a number of ways using variety of measures. Some of these measures are directly obtained from the classification process: a number of correctly classified data points, a number of misclassified points; while some could be obtained via simple arithmetic calculations, for example: accuracy, recall, and precision[1], as well as confidence and support[2]. There are a number of ways how correctness of the rule can be evaluated.

[1] The values of accuracy, recall and precisions are calculated based on the confusion matrix. In the confusion matrix for the case of two-class classification process, let *a* represents *TruePositive*, *b* – *FalsePositive*, c – *FalseNegative*, and d – *TrueNegative*. In such case the following formulas can be used:

$$accuracy = \frac{a+b}{a+b+c+d},\ recall = \frac{a}{a+c},\ precision = \frac{a}{a+b}$$

[2] The support of the rule is the relative frequency of rules that contain A and C:

$$support(A \rightarrow C) = support(A+C),$$

$$support(A \rightarrow C) = \frac{support(A+C)}{support(A)}$$

The method used in this research is based on application of the Laplace ratio. This ratio is very easily applicable to evaluate "goodness" of a single rule. In such case, the direct measures: number of properly classified data points and number of misclassified data points, are used. The formula used for calculating value of the ratio is presented below:

$$LaplaceRatio = \frac{Properly\_Classified\_Points + 1}{Properly\_Classified\_Points + Misclassified\_Points + 2} \quad (7)$$

This formula provides a balance between rule's ability to properly classify data points, and its popularity. The popularity relates to a number of data points that satisfy the rule. It is assumed that such behavior of the Laplace ratio is very much related to human's perception of "goodness" of the rule that is a combination of classification ability of the rule and its generality.

<u>Uniqueness of Rules.</u> A "unique" rule can be defined as the one which is least similar to other rules of the same class hierarchy. In order to find the uniqueness of a rule, we use the Teversky's similarity model (Section 2.2.2). The equations (4), (5) and (6) from section 2.2.2 are used for selecting a set of diversified rules.

A contingency table for rules R1 and R2

	R1	**no R1**
R2	a	b
no R2	c	d

$S-jacarrd(R1,R2) = \frac{a}{a+b+c}$ (4) // Calculates similarity between two rules

$S-inclusion(R1 \text{ in } R2) = \frac{a}{a+c}$ (5) // Calculates Inclusion of R1 in R2

$S-inclusion(R2 \text{ in } R1) = \frac{a}{a+b}$ (6) // Calculates Inclusion of R2 in R1

The procedure consists of two steps:

- The first part is dedicated to identifying rules that are semantically similar to each other (equation (4)); a calculation of similarity measures for all pairs of rules is performed – the pairs with highest values of similarity measure are considered for removal from the set of rules as we need to keep rules that are least similar (unique);
- The second part is focused on selecting rules that should be removed, this process is performed based on inclusion measures (equations (5) and (6)) calculated for the rules that are semantically similar (above); for each rule from pairs of similar rules a max value of inclusion of this rule in other rules is calculated; a rule with higher value of the average from each pair is removed.

The general approach is to find more generalized and important rules by comparing rules based on their similarities and inclusion of rules. The rules with higher degree of similarity will be considered for elimination. Inclusion of rules will help decide which rule to be eliminated from within the pair of two highly similar rules. The rules that are completely included or tend to repeat themselves in others are considered as weak and hence can be ignored without major loss of coverage and accuracy.

Figure 1 below shows a general framework for the methodology to be followed and is explained in a pseudo code below.

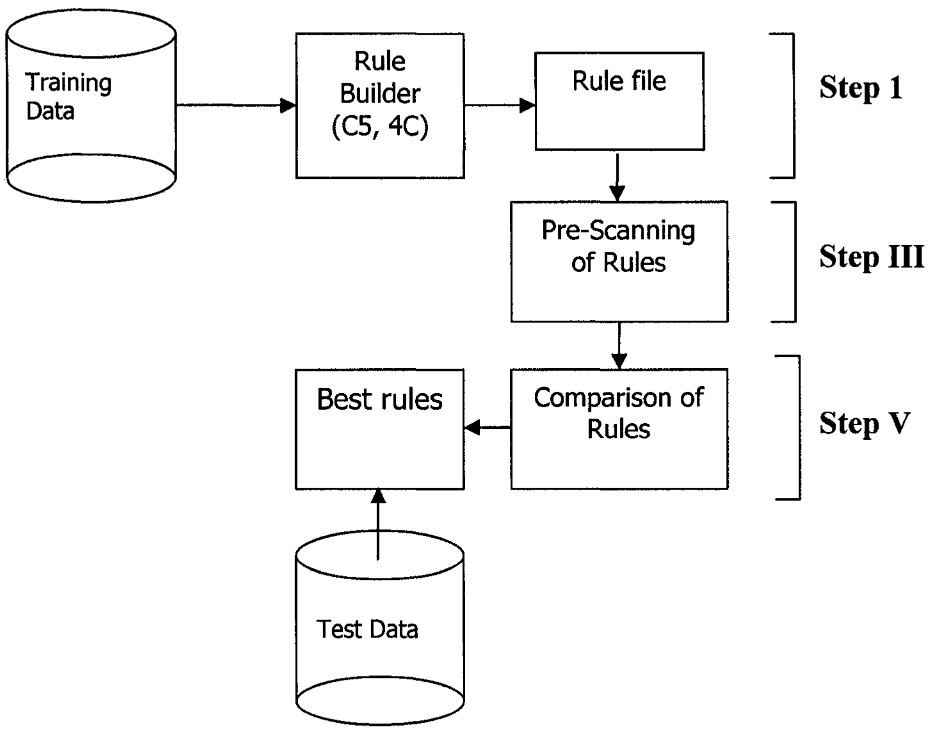


Figure 1. Process Flow

In the following are explained the major steps for discovering useful rules:

I. **Generation of Rules** – Use different tools to generate rules on the same dataset. Make separate Rules Files for each tool.

For all the rules for each Class follow the steps in the pseudo code below:

II. **Compute Laplace Ratio** – Compute Laplace ratio (equation 7) for the rules in a Rules File.

III. **Pre-Scanning of Rules** – Ignore the rules with Laplace value less than the minimum threshold Laplace value.

Note: The threshold to be set for the Laplace ratio can be different for different rule-generating softwares. For example: for See5 it was set to .70 and on the other hand for

4C Rule Builder .60 was found as the appropriate threshold value. It may also vary depending on the type of dataset used to generate rules.

IV. Enough rules should be found after ignoring the ones below threshold. If enough rules have been identified then STEP V else if there are not enough rules left to analyze, increase the threshold of the Laplace ratio to a greater value or vice-versa and repeat STEP III.

V. **Rule Analysis Procedure (Comparison of Rules)** – Two ways are used to determine the rules that best qualify for their interestingness. Interestingness of a rule is measured by its Importance and Uniqueness as detailed in the beginning of this section.

This methodology was implemented and validated on 2 different datasets with wide range of attributes and data points. The methodology was implemented in 2 phases, first as a feasibility study and second, as an advanced automated system to select the most interesting rules out of the entire lot. The feasibility study or the preliminary study of the proposed method proved to be very helpful in determining the way towards more advanced approach of finding interesting rules.

EXAMPLE

To understand the flow of the algorithm here is a simple example. The rules shown here were randomly picked and hence do not show the exact results from any of the experimented datasets.

Let, D be the dataset under consideration. D has 366 data points, 64 attributes {a1, a2, a3, a4...............a64} and two classes namely Class 0 and Class 1 with specific significance to their names or labels. See5 was used to generate rules over this dataset. The dataset has not been divided into training and test to ease the flow and understandability of the methodology. The rules generated by See5 are as follows:

RULES:

Rule 1 read as: 3 data points were misclassified out of total 38 that were classified by this rule.

If attribute a41 > 963 Then Class predicted is 1. The Laplace ratio is shown at the end of each rule in square brackets [0.900].

```
Rule 1: (38/3, lift 5.7)
    a41 > 963
      ➔   class 0  [0.900]

Rule 2: (25/2, lift 5.6)
    a2 > 26
    a14 > 11
    a52 <= 1
        ➔  class 0  [0.889]

Rule 3: (29, lift 6.1)
    a3 > 819
    a39 > 11253
        ➔  class 0  [0.968]

Rule 4: (21, lift 6.0)
    a31 > 52
    a39 > 11253
        ➔  class 0  [0.957]


Rule 5: (68/4, lift 2.4)
    a2 <= 3
    a14 <= 11
    a27 <= 21
    a64 <= 5
        ➔  class 1  [0.929]

Rule 6: (160/28, lift 2.2)
    a37 <= 142
        ➔  class 1  [0.821]

Rule 7: (75/2, lift 2.5)
    a4 <= 2
    a33 <= 1
    a37 <= 142
        ➔  class 1  [0.961]

Rule 8: (135/10, lift 2.4)
    a27 <= 24
    a37 <= 142
    a43 <= 11
        ➔  class 1  [0.920]
```

```
Rule 9: (26/2, lift 1.4)
    a20 <= 6
    a21 > 0
    a27 <= 20
    a36 <= 23440
        → class 1  [0.649]
```

Total of 9 rules were generated on the given dataset.

In the next step first we need to set up a threshold value for the Laplace ratio. For this set of rules .70 is assumed to be the Laplace threshold value. In order to qualify the pre-scanning of rules a rule must hold a minimum Laplace of .70. One may notice here that .80 could also be a good choice for Laplace threshold. But we must not forget that there are hundreds of thousands of rules generated at times on the actual datasets. Thus, we must set up a generalized cut off value for each tool by careful examinations of the ranges of Laplace value given by each tool.

All Rules for Class 0 (R1, R2, R3 and R4) posses a good Laplace value and thus all qualify for further analysis and interpretation. For Class 1, last rule has a Laplace of .649 which is below the threshold and thus Rule 9 needs to be eliminated at this step. We are left with 4 rules for Class 0 and 4 rules for Class 1, a total of 8 rules.

Now we need to compare rules within each class. In other words we need to select rules on the basis of their importance and uniqueness for each class. In the end we will be left with best rules for Class 0 and Class 1 respectively.

We start by building contingency tables for each pair of rules to be compared. How to build this table to calculate similarity and inclusion of rules is explained in Section 2.2.

Similarity and Inclusion of Rules for Class 0

For the pair R1 and R2

	R2	R2 (N)	
	17	21	R1
R1	8	366-(17+21+8) = 320	(N)

S (R1, R2) = 0.369

Inc (R1 in R2) = 0.447

Inc (R2 in R1) = 0.68

For the pair R1 and R3

	R1	R1 (N)
R3	29	0
R3 (N)	9	366-(29+0+9) = 328

S (R1, R3) = 0.763

Inc (R1 in R3) = 0.763

Inc (R3 in R1) = 1.0

For the pair R1 and R4

	R1	R1 (N)
R4	19	2
R4 (N)	19	366-(19+2+19) = 406

S (R1, R4) = 0.475

Inc (R1 in R4) = 0.5

Inc (R4 in R1) = 0.90

For the pair R2 and R3

	R2	R2 (N)
R3	17	12
R3 (N)	8	366-(17+12+8) = 329

S (R2, R3) = 0.459

Inc (R2 in R3) = 0.68

Inc (R3 in R2) = 0.58

For the pair R2 and R4

	R2	R2 (N)
R4	8	13
R4 (N)	17	366-(8+13+17) = 328

S (R2, R4) = 0.21

Inc (R2 in R4) = 0.38

Inc (R4 in R2) = 0.32

For each of these pairs plug in the values in equation 4, 5 and 6 to calculate the similarity and inclusion between each pair. The whole purpose is to keep rules which are important (in terms of accuracy, Laplace, goodness) and unique. The pairs with high semantic similarity value are the ones that will be considered for further analysis. For instance, for the above pair R1 and R3 they show a semantic similarity of about 76% with each other. It means that the chances of one rule being redundant or inclusive in the other are higher. The values for their inclusion clearly shows that R3 is 100% included in R1. In other words all the data points that are predicted by R3 are also predicted by R1. Thus, there will be no coverage degradation if R3 was eliminated because all data points covered by R3 are being covered by R1. In next scenario we notice 90% inclusion of R4 in R1. But the similarity between the two rules is not so high. In such cases it largely depends on the knowledge expert if he is willing to take the loss of 10% coverage if he chooses to remove R4. Next the two pairs of comparison (R2, R3) and (R2, R4) show medium and low ranges of similarity and inclusion, respectively.

This is the basic foundation on which my research work is based. However, implementing this methodology over real and large datasets was a challenge. In order to test and authenticate this whole methodology first step taken was to conduct a feasibility study of the process on real software dataset. This whole process was applied on the first dataset (Maintainability Dataset) that revealed some real facts and limitations in this methodology. Those were eradicated in the second step of building an automated, reliable system to perform the same task which is also the final result of my thesis.

4. SIMILARITY & INCLUSION OF RULES IN SOFTWARE MAINTENANCE DATA – A FEASIBILITY STUDY

4.1 Overview

This study looks at the issue of rule-based description of attributes of software with different levels of maintainability. Varieties of rules are extracted from a data set that represents human evaluation of maintainability of software objects. Rule similarity and rule inclusion measures are used to identify the most diverse sets of rules representing human evaluation criteria. Additionally, the rules representing all evaluators are analyzed using a rule similarity concept in order to learn more about common evaluation criteria.

The cost of maintenance activities accounts for more than half of the typical software budget [7][16]. This fact alone means that any effort leading to increase of maintenance effectiveness is essential for any software developing company. Application of different tools that support maintenance tasks is important, but a better understanding of what makes software more maintainable is of profound significance.

In the study, we analyze rules extracted from Software Maintenance data in order to find out what each evaluator "means" as bad/good maintainability. Additionally, we compare these findings, identify software attributes that evaluators find as important, and check if there is an agreement among them in identifying objects of good and bad maintainability. The approach we used to accomplish that is based on extraction of rules representing evaluators' views and comparing them. The comparison of rules is performed using the concepts of semantic similarity and inclusion of rules.

4.2 Software Maintenance Data

The data used in this method was collected during an experiment conducted in National Research Council (NRC), Canada. In the experiment, three programmers have independently analyzed software objects of the system EvIdent®. The aim of this evaluation was estimation of easiness of performing maintenance tasks on the objects.

For each of the 366 software objects, three programmers, called 'A', 'D' and 'V', were asked to independently rank objects' maintainability in the scale from 1 to 5 (where 1 means BAD maintainability, 2 means BAD-MEDIUM maintainability, 3 means MEDIUM maintainability, 4 means MEDIUM-GOOD maintainability, and 5 means GOOD maintainability). The programmers determined maintainability of objects based on their own judgment.

At the same time, a set of 64 software metrics was calculated for each object. As the result, the collected data set consists of 366 data points represented by a set of 64 software metrics and three values assigned to each point by three programmers. Some of the extracted metrics are shown in Table 2.

The graph in Figure 2 shows the distribution of data points identified as 1, 2, 3, 4 or 5. The graph clearly shows that there are very few data points classified as 1 (bad maintainability). Consequently, it would be difficult and inadequate to generate rules for such a small data subset. In other words, rules generated for 1 will be specific to these data points and will lack generality.

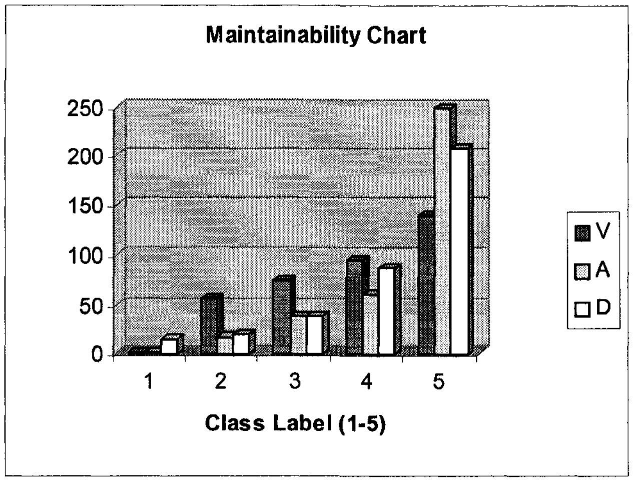


Figure 2. Distribution of Maintainability Data

In order to create rules that are more general to any kind of data with such measures, we modified the data. The data points were grouped together in a way to increase uniform distribution of data points over classes (scale values). The data points classified as 2 earlier were added to the group 1, and data points classified as 4 earlier were added to the group 3.

In all, maintainability of data points will now be classified as 1, 3 or 5 (where 1 means BAD maintainability, 3 means MEDIUM maintainability, 5 means GOOD maintainability).

Table 2. Software Attributes (Software Maintainability Data)

ATTRIBUTE	DESCRIPTION
TYPE	Type: GUI (=1), Data Model (2), Algorithm (3), Other (4).
METH	Number of Methods
LOC ALOC	Number of lines of code. Mean LOC per method.
RCC1	Ratio of comment lines of code to lines of code including whitespace & comments.
MTOK	Median TOK (# tokens) per method
DEC	Number of decisions.
WDC	Weighted measure of the number of nested decisions for an object (the greater the nesting the greater the associated weight).
MWDC	Median WDC per method.
INCL	# inner classes.
DINH	Depth of inheritance.
CHLD	Number of children.
SIBL	Number of siblings.
RCR	Code reuse: ratio of overloaded inherited methods to those that are not.
CBO	Coupling between objects – Number of other objects to which the corresponding object is coupled.
RFO	Response for an object. Response set contains the methods that can be executed in response to a message being received by the object.
MNL1 MNL3	Maximum method name length. Mean method name length.
ATCO	Attribute Complexity.
DAC	Data Abstraction Coupling - Number of reference types used in the attribute declaration.
HLDF,	Halstead difficulty.
HLPL	Halstead program Length.
HLVC	Halstead Program Vocabulary.
HLVL	Halstead Program Volume.
HLOR	Halstead # operators.
HLUN	Halstead # unique operands.
HLUR	Halstead # unique operators.
MIC	Method Invocation Coupling.
MAXQ	Maximum size operations.
ATTR	Number of attributes.
CONS	Number of constructors.
IMST	Number of Import statements.
OVRM	Number of overridden methods.
REMM	Number of various remote method calls.
PKGM	% Package members in a class.
WMC2	Weighted methods per class.

4.3 Methodology

4.3.1 Rule-based Maintenance Measures

The main goal here is to represent levels of maintainability in the form of IF-THEN rules generated from the data collected during the maintenance evaluation experiment conducted in NRC (Section 4.2).

In order to select best rules "representing" each programmer, and then find out commonalities among these rules, a special methodology described in Section 3 was applied:

- Sets of rules are generated for each class and each programmer, two different approaches are used to obtained diversified sets of rules (Section 4.3.2);
- Each set of rules is analyzed based on importance and uniqueness of rules (Section 3);
- All sets of rules describing objects with *bad maintainability* are compared using the similarity measure;
- All sets of rules describing objects with *good maintainability* are compared using the similarity measure.

4.3.2 Rule Generation Procedure

After deciding on the class labels as 1, 3 and 5, the next step is to generate rules for each class for each programmer.

Two experiment setups are followed to generate rules for each class:

a. all-class approach:

rules for all the classes – 1, 3, 5 – are generated in one run of the rule generation algorithm[3]; the input file is a single data file consisting of data points identified as 1, 3 or 5; the rules for each Class 1, 3 and 5 are generated;

b. *two-class approach:*
The original data file is used to create three data files: each new file consists of data points that "belong" to two groups – a group representing one of the Class 1, 3 or 5, and a group of points that do not belong to the Class 1, 3, or 5; for example a data file for Class 1 consists two Classes – 1 and 0 (the data points identified as 3 or 5 in original file are now as 0).

Once the rules are generated for each class, for each programmer, we move ahead to select the best rules. A number of rules for each class are generated, but we restrict ourselves to selecting only 2 best rules for each class by each programmer. This restriction is imposed in order to keep the process computationally less expensive.

4.4 FEASIBILITY STUDY – EXPERIMENTAL EVALUATION AND RESULTS

4.4.1 Bad Maintainability

At the beginning let's look at the bad maintainability of software object. The main aim is to find out which attributes are important for each programmer during his/her estimation of bad maintainability, as well as to determine a small and diverse set of rules "representing" each programmer.

Programmer V
The programmer V indicated that 58 objects (15.8 % out of 366) are of bad maintainability. The application of both approaches for generating rules resulted in a total

[3] The rule generation algorithm used here is C5.0/See 5.0.

of 2 rules for the all-class approach (Section 4.3.2.a), and 2 rules for the two-class approach (Section 4.3.2.b). The values of Laplace ratios for the rules are in the range of 0.88 to 0.90 for rules generated by all-class approach and 0.95 to 0.97 for rules generated by two-class approach. The initial selection of rules is done solemnly on the basis of "goodness" of rules. From all these rules, the rules with the Laplace ratio above 0.9 have been selected: a single rule V-L-1A[4] from the first set, and two rules V-L-1 and V-L-2 from the second set. All three rules are shown below.

Rule$_{V\text{-}L\text{-}1A}$ (35, 5, 0.900[5]):
IF HLOR > 963
THEN class: **bad_maintainability**

Rule$_{V\text{-}L\text{-}1}$ (29, 0, 0.968):
IF LOC > 819 & HLVL > 11253
THEN class: **bad_maintainability**

Rule$_{V\text{-}L\text{-}2}$ (19, 2, 0.957):
IF ATCO > 52 & HLVL > 11253
THEN class: **bad_maintainability**

The inspection of these three rules leads to conclusion that the rule V-L-1 is the best. It properly classifies 29 points and does not misclassify any points.

A contingency table for all three rules is built, Table 3. It can be observed what the rules V-L-1 and V-L-1A are the most similar ones.

High similarity of rules V-L-1 and V-L-1A means that one of them is redundant and should be removed from the set of rules. In order to do this, a table of inclusions is built, Table 4. The inclusion measures are calculated for each rule, V-L-1 and V-L-1A, in reference to the other two rules.

[4] The letter A indicates that the rules with the letter were generated using all-class approach.

[5] The triple (35, 5, 0.900) means that the rule properly classified 35 data points, that it misclassified 5 data points, and its value of Laplace ratio is 0.900.

The values of inclusion measures indicate that the rule V-L-1 is fully included in the rule V-L-1A. This means that the rule V-L-1 can be removed from the set of rule without any degradation of coverage.

Table 3. Contingency table for **bad_maintainability** rules representing the programmer V

Pair of rules (R1, R2)	**Contingency table entries**				**Similarity measure**
	a (R1 & R2)	**b (R1 & nR2)**	**c (nR1 & R2)**	**d (nR1 & nR2)**	
(**V-L-1**, **V-L-1A**)	29	0	9	328	**0.7632**
(**V-L-2**, **V-L-1A**)	19	2	19	326	0.4750
(**V-L-1**, **V-L-2**)	13	16	8	329	0.3514

Overall, two rules V-L-1A and V-L-2 (rules which are underlined) are used to represent the programmer V. The first rule indicates that if the number of operators in a class (HLOR) is above 963 the class is of bad maintainability. The second rule points to the case that a class is of bad maintainability if it has the sum of each attribute's value higher than 52 and its Halstead Program Volume is higher than 11,253.

Table 4. Inclusion table for rules V-L-1 and V-L-1A

Inclusion pair	**Inclusion measure**	**Max of inclusion measure**
V-L-1 in **V-L-2**	0.4483	**1.0000**
V-L-1 in **V-L-1A**	1.0000	
V-L-1A in **V-L-1**	0.7632	**0.7632**
V-L-1A in **V-L-2**	0.5000	

Programmer A

A total of 3 rules were generated using both the approaches (Section 4.3.2).

Rule$_{A-L-1A}$ (5, 0, 0.857):
IF HLDF > 135
THEN class: **bad_maintainability**

Rule$_{A-L-2A}$ (4, 0, 0.833):
IF WDC <= 228 & ATCO <= 36 & HLVC > 162 & ATTR > 14
THEN class: **bad_maintainability**

Rule$_{A-L-1}$ (9, 1, 0.833):
IF TYPE = 1 & LOC > 695 & MWDC <= 1 & CHLD <= 0 & OVRM > 1
THEN class: **bad_maintainability**

Table 5. Contingency table for **bad _maintainability** rules representing the programmer A

Pair of rules (R1, R2)	**Contingency table entries**				**Similarity measure**
	a (R1& R2)	**b (R1 & nR2)**	**c (nR1 & R2)**	**d (nR1 & nR2)**	
(A-L-1, A-L-1A)	2	8	3	353	0.1538
(A-L-1, A-L-2A)	3	7	1	355	**0.2727**
(A-L-1A, A-L-2A)	1	4	3	358	0.1250

The calculations of inclusion measures for these rules resulted in the elimination of the rule A-L-2A. The two remaining rules (rules those are underlined) defined bad maintainability, from the point of view of the programmer A, in the following way: class is of bad maintainability if its Halstead Difficulty is above 135, or if it is a GUI class with more than 695 lines of code, median number of decisions less or equal one, no children, and more than 1 overridden methods.

Programmer D

The programmer D evaluated 35 (9.6 % out of 366) classes as of bad maintainability. Both rule generation approaches resulted in 5 rules for the all-class approach, and 2 for the two-class approach. The best rules are shown:

Rule$_{D\text{-}L\text{-}1A}$ (13, 0, 0.933):
IF REMM > 101
THEN class: **bad _maintainability**

Rule$_{D\text{-}L\text{-}2A}$ (10, 0, 0.917):
IF LOC > 401 & DINH <= 2 & SIBL > 1 & RFO > 80 & MNL3 > 13
THEN class: **bad _maintainability**

Rule$_{D\text{-}L\text{-}1}$ (12, 0, 0.929):
IF MTOK <= 154 & HLVC > 198 & MIC > 2
THEN class: **bad _maintainability**

According to the proposed analysis, similarity measures were calculated as shown in Table 6. Following this, inclusion measures of rules D-L-1 and D-L-1A in reference to other rules were obtained. As the result, the rule D-L-1 has been removed.

Table 6. Contingency table for **bad _maintainability** rules representing the programmer D

Pair of rules (R1, R2)	**Contingency table entries**				**Similarity measure**
	a (R1 & R2)	**b (R1 & nR2)**	**c (nR1 & R2)**	**d (nR1 & nR2)**	
(**D-L-1**, D-L-**1A**)	10	2	3	351	**0.6667**
(**D-L-1**, D-L-**2A**)	0	12	10	344	0.0000
(**D-L-1A**, D-L-**2A**)	2	11	8	345	0.0952

The simplest of rules that represent the programmer D indicates that bad maintainability classes are ones with number of remote methods exceeded 101. The second rule is more complex: the bad maintainability classes have more than 401 lines of code, less or equal to 2 depth of inheritance, more than 1 sibling, more than 80 methods executed in response to a message received by the class, and a mean method name length over 13 characters.

Comparison of Rules

The data processing and analysis of the generated rules have lead to a three pairs of rules. Each pair "represents" a single programmer. The next step in analysis of Maintenance data is comparison of the three pairs of rules. This comparison should provide us with an indication if there was any common ground used to identify bad maintainability objects. This task is accomplished by looking at the similarity among rules.

The similarity measure is calculated for eight triples. The approach taken here is an extension of the idea presented in Section 2.2.2. This time, two tables, similar to Table 1, are built. The first table is for true R1, while the second is for false R1. The number of software objects that satisfy all three rules (column R1, R2 & R3) and any combination of two rules (other columns) are in Table 7. The similarity measures are presented in Table 8.

Table 7. Number of software objects satisfying three & two rules simultaneously for **bad_maintainability**

ID	R1				not R1			
	R2 & R3	R2 & nR3	nR2 & R3	nR2 & nR3	R2 & R3	R2 & nR3	nR2 & R3	nR2 & nR3
1*	0	5	7	26	0	0	3	340
2	0	1	6	14	2	2	5	352
3	0	4	2	15	2	4	6	351
4	0	1	2	18	0	4	8	348
5	2	8	0	28	0	0	8	338
6	2	3	10	23	0	0	1	343
7	3	1	3	14	2	4	5	352
8	5	5	7	21	0	0	1	345

* 1: (V-L-1A A-L-1A D-L-2A) 2: (V-L-2 A-L-1A D-L-1A)
3: (V-L-2 A-L-1 D-L-2A) 4: (V-L-2 A-L-1A D-L-2A)
5: (V-L-1A A-L-1 D-L-2A) 6: (V-L-1A A-L-1A D-L-1A)
7: (V-L-2 A-L-1 D-L-1A) 8: (V-L-1A A-L-1 D-L-1A)

Table 8. Similarity measures[6] calculated based on Table 7

ID	Sim(R1, R2, R3)	Sim(R2, R3)	Sim(R1, R3)	Sim(R1, R2)
1*	0.0000	0.0000	0.1707	0.1316
2	0.0000	0.1250	0.2143	0.0400
3	0.0000	0.1111	0.0690	0.1481
4	0.0000	-	-	-
5	0.0435	-	-	-
6	0.0513	-	-	-
7	0.0938	-	-	-
8	0.1282	0.2778	0.3077	0.2632

*as in Table 7

We can conclude based on inspection of Table 8 that in the case of identifying bad maintainability software objects there was a very little "agreement" among all three programmers. The values of similarity measure (second column of Table 8) are very small. There are a few pairs of rules (columns 2, 3 and 4) that have similarity measures around 0.25. It seems that there is some similarity between rules representing programmers A and D.

4.4.2 <u>Good Maintainability</u>

In analogy to Section 4.4.1, the same process of generating rules and their analysis has been performed for good maintainability. Due to space limitation, we briefly describe

[6] The empty entries in Table 8 are due to repetition of pairs.

rules for each programmer and discuss an issue of similarity among them. We dedicate more space for the comparison of all rules.

Programmer V

The number of objects identified by the programmer V as of good maintainability is 139 (38.0% out of 366). Such a large number of objects led to generation of rule with large coverage. The best three rules are shown below:

Rule$_{\text{V-H-1A}}$ (64, 4, 0.929):
IF METH <= 3 & WDC <= 11 & MNL1 <= 21 & WMC2 <= 5
THEN class: **good_maintainability**

Rule$_{\text{V-H-1}}$ (73, 2, 0.961):
IF ALOC <= 2 & DAC <= 1 & HLPL <= 142
THEN class: **good_maintainability**

Rule$_{\text{V-H-2}}$ (125, 10, 0.920):
IF MNL1 <= 24 & HLPL <= 142 & HLUR <= 11
THEN class: **good_maintainability**

The similarity measures for all three pairs are close to each other. The rules of the last pair, Table 9, are the most similar to each other. After calculating inclusion measures it becomes obvious that the rule V-H-1 can be removed (its inclusion measure with the rule V-H-1A is 0.92).

The rules describing what does it mean good maintainability for programmer V indicate that these classes have: a number of methods less than 3, a number of decision less than or equal 11, a maximum method name length less than 22, and a number of methods is less or equal to 5; or a maximum method name length less than 25, Halstead Program Length less than 143, and Halstead number of unique operators less than 12.

Table 9. Contingency table for **good_maintainability** rules representing the programmer V

Pair of rules (R1, R2)	**Contingency table entries**				**Similarity measure**
	a (R1 & R2)	**b (R1 & nR2)**	**c (nR1 & R2)**	**d (nR1 & nR2)**	
(V-H-1, V-H-1A)	45	30	23	268	0.4592
(V-H-2, V-H-1A)	64	71	4	227	0.4604
(V-H-1, V-H-2)	69	6	66	225	**0.4849**

Programmer A

The programmer A identified 248 objects (67.8% out of 366) as of good maintainability.

The best rules that represent his decisions are shown:

Rule$_{A\text{-}H\text{-}1}$ (34, 0, 0.972):
IF TYPE = 4 & HLDF <= 28
THEN class: **good_maintainability**

Rule$_{A\text{-}H\text{-}2}$ (163, 5, 0.965):
IF CBO <= 5 & HLDF <= 28
THEN class: **good_maintainability**

Rule$_{A\text{-}H\text{-}3}$ (70, 2, 0.959):
IF TYPE = 2 & MAXQ <= 3 & CONS <= 2 & OVRM <= 13
THEN class: **good_maintainability**

The similarities measures are quite low indicating that the rules cover quite different objects, Table 10. In order to eliminate one rule we pick the first pair and after calculating the inclusion measure we remove the rule A-H-1 (the measure for this rule and the rule A-H-2 is 0.8529).

Table 10. Contingency table for **good_maintainability** rules representing the programmer A

Pair of rules (R1, R2)	**Contingency table entries**				**Similarity measure**
	a (R1 & R2)	**b (R1 & nR2)**	**c (nR1 & R2)**	**d (nR1 & nR2)**	
(A-H-1, A-H-2)	29	5	139	189	**0.1676**
(A-H-1, A-H-3)	0	34	72	260	0.0000
(A-H-2, A-H-3)	13	155	59	139	0.0573

Programmer D

The programmer D recognized 207 objects (56.6% out of 366) as of good maintainability. The rules are below:

Rule$_{D\text{-}H\text{-}1}$ (135, 7, 0.944):
IF DEC <= 7 & RCR <= 0 & HLUN <= 63
THEN class: **good_maintainability**

Rule$_{D\text{-}H\text{-}2}$ (116, 7, 0.936):
IF RCC1 <= 9.7 & AWDC <= 7 & IMST <= 12 & REMM <= 35 & PKGM <= 1
THEN class: **good_maintainability**

Rule$_{D\text{-}H\text{-}3}$ (160, 20, 0.885):
IF DAC <= 2 & HLUN <= 63
THEN class: **good_maintainability**

The pair D-H-1 and D-H-2 has the highest value of similarity measure (Table 11), and the rule D-H-1 has a high value of inclusion measure (with rule D-H-3 – 0.9437).

Table 11. Contingency table for **good_maintainability** rules representing the programmer D

Pair of rules (R1, R2)	***Contingency table entries***				***Similarity* measure**
	a (R1 & R2)	**b (R1 & nR2)**	**c (nR1 & R2)**	**d (nR1 & nR2)**	
(D-H-1, D-H-2)	92	50	31	193	**0.5318**
(D-H-1, D-H-3)	134	8	172	52	0.4268
(D-H-2, D-H-3)	101	22	158	85	0.3594

Comparison of Rules

The comparison of all rules generated for all programmers shows more similarities, Tables 12 and 13.

Table 12. Number of software objects satisfying three and two rules simultaneously good_maintainability

ID	**R1**				**not R1**			
	R2 & R3	**R2 & nR3**	**nR2 & R3**	**nR2 & nR3**	**R2 & R3**	**R2 & nR3**	**nR2 & R3**	**nR2 & nR3**
1*	12	6	46	2	8	46	57	364
2	16	37	66	16	4	15	37	350
3	18	0	48	2	41	13	73	364
4	52	2	77	4	7	11	44	362
5	56	8	2	2	41	63	24	364
6	64	0	2	2	93	11	21	364
7	82	46	1	6	15	25	25	360
8	126	2	3	4	31	9	20	362

* 1: (V-H-1A A-H-3 D-H-2) 2: (V-H-2 A-H-3 D-H-2)
3: (V-H-1A A-H-3 D-H-3) 4: (V-H-2 A-H-3 D-H-3)
5: (V-H-1A A-H-2 D-H-2) 6: (V-H-1A A-H-2 D-H-3)
7: (V-H-2 A-H-2 D-H-2) 8: (V-H-2 A-H-2 D-H-3

Table 13. Similarity measures[7] calculated based on Table 12

ID	**Sim(R1, R2, R3)**	**Sim(R2, R3)**	**Sim(R1, R3)**	**Sim(R1, R2)**
1*	0.0678	0.1143	0.4427	0.1500
2	0.0838	-	0.4659	0.3442
3	0.0923	0.3057	0.3626	-
4	0.2640	-	-	-
5	0.2857	0.5000	-	0.3721
6	0.3316	-	-	-
7	0.4100	-	0.4743	-
8	0.6462	0.8220	0.6935	0.7314

[7] The empty entries in Table 13 are due to repetition of pairs.

* as in Table 12

The similarity value calculated for the three rules V-H-2, A-H-2, D-H-3 (0.6462) indicates that these rules are very similar. All three programmers identified the same objects as of good maintainability. These objects are described by rules that look quite different. This means that each programmer "saw" different aspects of software objects, and that these three rules describe practically the same objects. A better description of good maintainability objects can be obtained when all three rules can be combined.

```
IF MNL1 <= 24  & HLPL <= 142  &  HLUR <= 11
&  CBO <= 5  & HLDF <= 28
&  DAC <= 2  & HLUN <= 63
THEN class: good_maintainability
```

Another high value of similarity measure (0.5000) is obtained for a pairs of rules: A-H-2, D-H-2

```
IF CBO <= 5  & HLDF <= 28
&  RCC1 <= 9.7  & AWDC <= 7  & IMST <= 12  &
   REMM <= 35  & PKGM <= 1
THEN class: good_maintainability
```

To conclude, the feasibility study conducted on Maintainability data successfully predicted different levels of maintainability. The best rules identified for each class are highly accurate and unique.

5. AUTOMATED TOOL FOR GENERALIZATION OF RULES

5.1 System Overview

The work done in Feasibility study proved out to be very beneficial to predict the level of maintainability of the software attributes under consideration. The underlying reason of conducting this preliminary study (Section 4) was to gain insight into the working of semantic and syntactic comparisons of rules. After realizing the significance of discovering IF-THEN rules to predict the quality of a software, we needed to develop a system that could perform the whole task automatically, was far more accurate than a manual approach, was less time consuming and could easily handle large datasets. This section describes the major contribution of my research work aiming at developing and implementing an automated system to find the best rules for a given dataset. The approach and methodology used here are the same as in feasibility study shown in Section 4. The only difference being that the feasibility study experiments were conducted manually whereas the system developed here is a fully functional automated approach to find the best set of rules. This entire process is implemented in Python 2.4 and all the data and results are stored using Extensible Markup Language (XML).

Different data splits can be used in the system. Here the complete dataset was divided into 3 sets: Training, Evaluation and Testing. The idea of creating three separate datasets was to check the importance of rules at three different levels and to come up with more generalized rules as far as possible. Figure 3 below explains the purpose of each of these datasets.

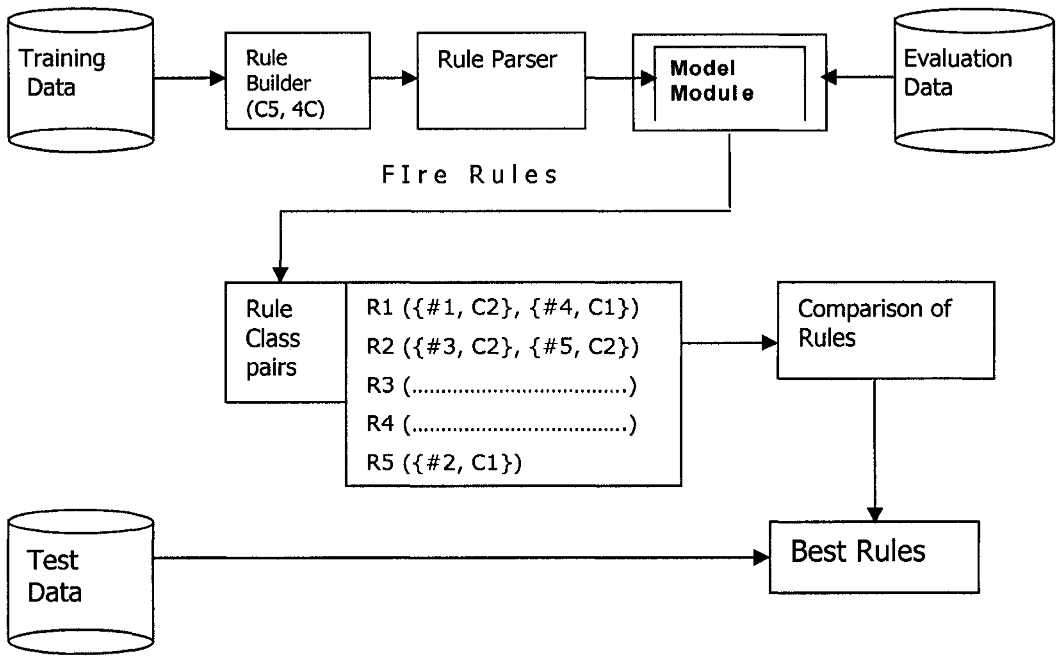


Figure 3. Structural Design of the automated tool

The process in Figure 3 begins with generating the Modules or Executables (next Section 5.2). First of all Rule Builders (See5 & 4C) are used to generate initial rule files on training data. This is the only use of training data in the process that it leads to generation of initial set of rules. These rules without any processing are passed on to the subsystem that generates the Modules. One Module will contain all the rules generated by one Rule builder. Now, once we have all the rules in the executable modules, these rules are fired on the evaluation dataset. This is an intermediate state to check the excellence of rules generated. The output of firing rules on evaluation data is in the form of pairs (data point, class) for each rule that was fired. For example: firing rule R1 on evaluation data results in 2 pairs of (data point, class): ({#1, C2}, {#4, C1}). This means rule R1 covered data points 1 and 4 and predicted them as Class 2 and Class 1 respectively. The next step in the process is to compare these rules. The entire process of selection of rules described in Section 3 (importance and uniqueness of rules) is followed. Rules are compared based on their goodness and uniqueness. Unimportant or uninteresting rules are ignored and the result is the set of rules that best predict the evaluation data without much loss of coverage and accuracy due to ignoring some rules. So we see that actual processing is done on evaluation data. The system, when fires rule on evaluation data, calculates the new Laplace ratios and replaces the initial Laplace. The initial Laplace of the rules was determined based on training data and the new Laplace helps us know how the rule performs on evaluation data. Next, the system will ignore the rules that have Laplace below threshold. Then the system finds the similarity and inclusion measures of the qualified rules and outputs the best set of rules.

Finally these best rules are used to predict the classes of data points from the test dataset which will finally prove the authenticity of the system.

This automated tool will also help in comparing rules generated by two different tools. There are many different Rule Generating software tools available to us. All of them work in different ways to generate rules on the same dataset. For example, we may notice the amount of difference in the rules generated by 4C and See5. The rules are not only generated in different ways but are also different syntactically. There are different terms and measures a rule-generating tool uses to characterize its rules. For example, support, confidence, accuracy, Laplace ratio, lift and so on. Thus, there is a need to compare rules generated by different tools to find any commonalities between these rules and to come up with best set of rules among those generated by different tools.

5.2 System Architecture and Design

A Blackboard architecture has been used to design the system. The blackboard architecture addresses the issue of information sharing and is used as a central repository of data by all the subsystems. In this architecture different data splits can be used, the generated rules can be fired on any given dataset and then based on rule comparison measures, system will compare all the rules that are fired and generate output files.

Blackboard Architecture

Blackboard systems were designed to resolve complex Artificial Intelligence (AI) problems. The blackboard system works based on the following metaphor [25]:

"Imagine a group of human specialists seated next to a large blackboard. The specialists are working cooperatively to solve a problem, using the blackboard as the workplace for developing the solution. Problem solving begins when the problem and initial data are written onto the blackboard. The specialists watch the blackboard, looking for an opportunity to apply their expertise to the developing solution. When a specialist finds sufficient information to make a contribution, she records the contribution on the blackboard, hopefully enabling other specialists to apply their expertise. This process of adding contributions to the blackboard continues until the problem has been solved".

There are a number of features that make such architectural models to be the best in their ability to solve complex problems. For example, the blackboard systems are flexible in their representation. There is no specific format in which the contributions should be depicted on the blackboard. They may vary per expert. Such kind a system allows easy interaction among the experts who have diverse opinions and thinking.

In all, a blackboard system can be defined as a repository of solutions and contributions to the current problem which are repeatedly updated.

The blackboard architecture consists of three major components:

1. The software specialist modules, which are build by the experts to provide specific expertise needed by the application.
2. The blackboards, which contain all the data, problem statements, updated solutions and any contributions towards solving the problem. The blackboards are being updated and watched continuously.
3. Control unit, which controls the flow of the problem-solving activity. This can be seen as a way to organize the use of data in other working units in effective and coherent manner.

Figure 4 below represents the blackboard architecture of our application and the program flow. Though it is possible to maintain a single blackboard for all the data, results and experimental calculations, a number of blackboards have been constructed to ease the readability of the solution. The blackboards are split on the basis of the type of information they will contain. For example: in the data_blackboard, as the name suggests, we write the dataset under consideration. Every time we run the program for a different dataset, it will be copied into the data_blackboard replacing the existing dataset. Similarly, the results_blackboard will contain the results of running the program on a given dataset.

Next, we may notice the 'Control' unit of our architecture model. It is the control unit here that organizes the flow of the application. It is coded in Python and is simply like a function call mechanism where we just call the functions in a particular sequence. The control is responsible for calling different specialized working modules in an ordered fashion. It may also be defined as a central control component that evaluates the current state of processing and coordinates the specialized programs [26].

The specialized modules or the functions form the main structure of our blackboard model to select the best rules out of large number of rules generated by rule-generating tools. These modules are all written using Python 2.4 and all major modules are shown in Figure 4 below.

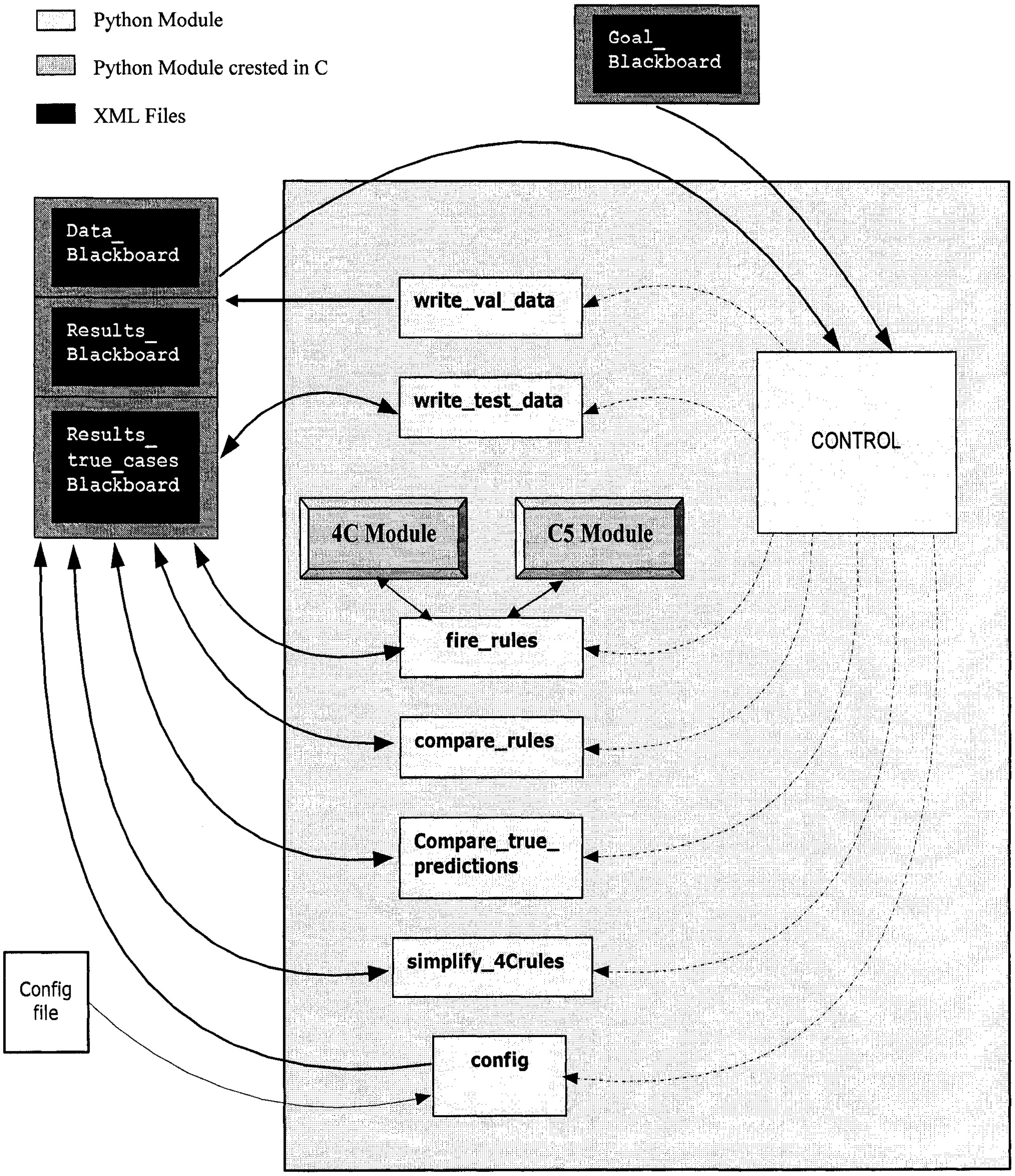


Figure 4. Blackboard Architecture

These specialized units are usually independent of each other. Following is the description of all the major modules of this application.

1. write_xxxx_data – This function is responsible for reading the datasets - training, evaluation or test dataset – which should be used and writing them in desired order to the data_blackboard. A version of this function – with a different name – exists for each type of dataset. For example: write_eval_data will read the evaluation data text file and write it to data_blackboard and similarly the write_test_data will write the test dataset to the blackboard.
2. 4C and See5 Module Creation - Every rule generating software generates rules in a specific format. These formats are basically constructed in a way that they can be easily interpreted by design experts or users. Thus, we needed a way to convert these specific outputs from say See5 and 4C into generic formats that will be used by our system. These generic formats are in the form of a python dynamic module (pyd). In other words, these python modules are just another way of expressing the rule-generating tool's outputs in the form of an executable program. The main reason for creating these modules was to find an efficient way to use the rules generated by rule-generated tools for classification of any number of datasets. Figure 5 shows the flow of creation of a module.

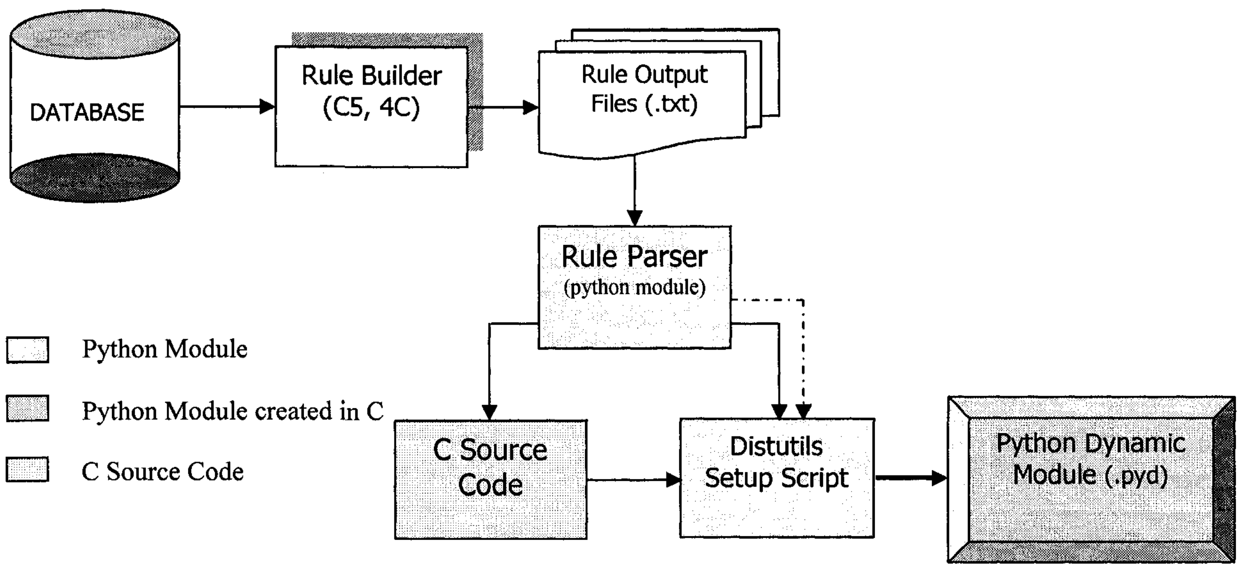


Figure 5. Module Creation

A rule parser is written for each outputted format of rule-generating tools. These parsers are capable of reading the output files of rule-generating tools and translate these rules into conditional statements. For example, in our case two parse files will be created, one to parse the results of See5 and the other to parse and read the output of 4C rule builder. The See5 parser file will read the output from See5 and write a C file which contains all the rules from the See5 output file in the form of if-else statements. After creating this C file the parser generates a module file i.e. in our case a .pyd file. A pyd is a dll built by a C compiler. They are renamed to .pyd files simply to differentiate them from "regular" windows DLL's.

Let us consider an example rule file. The rules below are the output of See5 saved as a text file. Running the See5 parser to read and translate these rules into meaningful and readable if-then rules, will give us the output as shown on next page.

```
See5 [Release 2.04]    Thu Mar 01 18:18:35 2007
-------------------
Options:  Rule-based classifiers
Class specified by attribute `class'
Read 245 cases (65 attributes) from Main_V.data

Rules:

Rule 1: (22, lift 5.9)
  MNL2 <= 8
  HLOR > 933
  -> class 1  [0.958]

Rule 2: (12, lift 5.7)
  INCL <= 1
  HLOR > 933
  -> class 1  [0.929]

Rule 3: (12/2, lift 4.8)
  MNOC <= 4
  WDC > 16
  -> class 1  [0.786]

Default class: 3

Evaluation on training data (245 cases):

        Rules
   ----------------
     No      Errors
     11   15( 6.1%)   <<

    (a)  (b)  (c)    <-classified as
   ---- ---- ----
     37    3          (a): class 1
      4  100    5     (b): class 3
           3   93     (c): class 5
```

Partial result after parsing the above text file (output of See5) using the Rule Parser:

```
if ((vars[27] <= 8) && (vars[40] > 933) )
    Fire_Rule(1, 1); // (0.958)


if ((vars[16] <= 1) && (vars[40] > 933) )
```

```
    Fire_Rule(2, 1); // (0.929)


if ((vars[4] <= 4) && (vars[13] > 16) )
    Fire_Rule(3, 1); // (0.786)
```

These parser files, thus, transform the input text into simple if-else statements, which are suitable for later processing and which capture the implied hierarchy of the input.

3. fire_rules – This function is one of the important steps in processing the evaluation data. This fire_rules module picks each data point from the evaluation dataset, matches it to the rules in the respective .pyd modules and predicts the class of each data point on the basis of the rule fired. The result is (rule, class) pair for each data point. It means that for each data point we obtain, the information about which rule has been fired and the class predicted by that rule. As a trivial example, let us say we have 10 data points in evaluation dataset. Fire_rules runs and predicts class for each of these 10 points on the basis of the rules present in the modules and finally gives us the (rule, class) pair for each data point. Further, this (rule, class) pairs are changed to (data point, class) pairs for each rule to show how many data points were covered by each rule along with the class predicted for each data point by that rule [Figure 3].
4. comapare_rules – This is the most significant step in the process of discovering relevant rules. The comparison of rules is based on the semantic comparison of the rules. Each rule is automatically compared with all the other rules of the same class. They are compared for their goodness (Laplace ratio), similarity (jaccard's

measure) and uniqueness (inclusion measures) as discussed in Section 3. Rules which tend to be similar to other rules and the rules that are included in others are ignored.

5. compare_true_predictions – Initially while comparing rules, rules were compared for any predictions they made, including misclassifications made by rules. This factor remained uncovered during the feasibility study and later it was realized that rules should be compared on the true positives predicted by the rules. This step is performed by the program in addition to the above comparison. Here rules are compared for their similarity and inclusion only on the basis of correct prediction made by them and the results are stored separately in results_true_cases_blackboard.

6. simplify_4CRules – This function of the program can be considered as an additional exercise to simplify the rules generated by 4C. The rules generated by the 4C rule builder are complex and very hard to understand and interpret. It uses negation to represent the rules. For example:

   Rule 1:

```
If [ <RCC1
!=[7.05,8.80),[10.30,11.90),[22.90,34.40),[34.40,57.00),[57.00,434.00),[-
0.85,3.45),[11.90,14.50),[3.45,4.65),[8.80,10.30)><LCOM !=[-
0.50,10.50),[10.50,28.50),[28.50,49.00),[84.50,119.00),[325.00,601.00)><HLVL
!=[81.00,161.00),[2E3,2.25E3),[549.00,895.00),[2.25E3,2.7E3),[3.74E3,4.9E3),[6.02
E3,7.61E3),[1.57E3,2E3),[161.00,270.00),[-
1.00,37.00),[3.01E3,3.74E3),[1.07E3,1.57E3),[2.7E3,3.01E3),[895.00,1.07E3),[37.00
,81.00),[387.00,549.00),[270.00,387.00),[1.94E4,2.99E4),[1.58E4,1.94E4)>] Then
Class = 1
```

covers 17% (7 out of 40) examples from Pos Matrix

All the rules generated by 4C are represented by negation and they also use exponent to express some large values of the attributes (highlighted value). Thus, the module simplify_4Crules makes the rules readable by expressing them without use of any negation and also takes away the scientific expression E and represents values in their actual decimal form. The above rule is simplified as follows:

```
( RCC1 < -0.85 || ( RCC1 => 4.65 && RCC1 < 7.05 ) || (RCC1 => 14.5 && RCC1 <
22.9 ) || RCC1 => 434.0 )
(LCOM < -0.5 || (LCOM => 49.0 && LCOM < 84.5 ) || (LCOM => 119.0 && LCOM <
325.0 ) || LCOM => 601.0 )
( HLVL < -1.0 || (HLVL => 1570.0 && HLVL < 2250.0 ) || (HLVL => 4900.0 &&
HLVL < 6020.0 ) || (HLVL => 7610.0 && HLVL < 15800.0 ) || HLVL => 29900.0 )
     Fire_Rule(1, 1); // 1
```

5.3 Format of Output files

All the output files of the system are generated in XML and stored in appropriate blackboards. There are two main output formats that are revealed here. One is the result of firing a rule on a given dataset and the second is the result of comparison of rules. Starting with what we see as output once a rule is fired on a dataset:

```
<Rule_C5-M-1>
  <rule_result>10:5-T 11:5-T 14:5-T 34:5-T 36:5-T 38:5-T 44:5-T 49:5-T
    55:5-T   58:5-T 71:5-T 80:5-T 85:5-T 92:5-T 94:5-T 97:5-T 100:5-
    T</rule_result>
  <rule_count>17 17 0</rule_count>
  <rule_laplace>0.947</rule_laplace>
  <rule_class>5</rule_class>
</Rule_C5-M-1>
```

Table 14. Format of output files

Naming Convention	{Rule_C5-M-1} -> {Rule_tool-Data-#}. Tool is See5 or 4C. Data type is M (Maintainability Data) or N (NASA Data) and # represents the Rule No.
Rule_result	{10:5-T} -> {data point no.: class predicted – True/False}
Rule_count	{17 17 0} -> {Total predictions; correct predictions, misclassified}
Rule_laplace	Laplace for the rule based on the above count
Rule_class	Class predicted by this rule

Next we see the output when rules are compared to each other. For instance, the rule C5-M-1 is compared to other See5 rules: C5-M-2 and C5-M-3. In order to compare these rules based on their similarity and inclusion we need to find the values of a, b, c and d as shown in table below. Once we have values of a, b, c, d from the contingency table we calculate e (Similarity between rules R1 and R2); f (Inclusion of R1 in R2) and g (Inclusion of R2 in R1).

```
<Rule_C5-M-1>
  <compared_rule_C5-M-2>17 0 26 78 Similarity: 0.395; Inclusion: 1.0, 0.395
                        a b c d              e                  f, g
  <compared_rule_C5-M-3>0 17 15 89 Similarity: 0.0; Inclusion: 0.0, 0.0

</Rule_C5-M-1>
```

A contingency table for rules (C5-M-1)R1 and (C5-M-2)R2

	R2	no R2
R1	a	b
no R1	c	d

e = Similarity (R1, R2) = a/ (a+b+c)
f = Inclusion (R1 in R2) = a/(a+b)
g = Inclusion (R2 in R1) = a/(a+c)

Thus, Section 5 details the architecture, design and working of the system developed to find the most interesting rules from among the large number of rules generated.

6. EXPERIMENTAL EVALUATION OF THE AUTOMATED TOOL

The entire approach of extracting important and relevant rules using the automated system described in section 5 was empirically tested on two datasets namely, Software Maintainability data and the NASA data.

6.1 Data Description and Preparation

The developed method was implemented and tested on two different datasets: - Software Quality Data or Maintainability Data, the one used in feasibility study (Section 4.2), and second dataset was a bigger dataset provided by National Aeronautics and Space Administration (NASA). This NASA data was collected from the various software packages running in the live production environment of NASA and has been used for various kinds of research works. Each dataset was split in various ways to help examine each phase of the system critically.

Maintainability Data – This dataset is made up of 366 data points and 64 attributes. The data points can be classified into 3 classes: Class 1 (Bad Maintainability), Class 3 (Medium level of Maintainability) and Class 5 (Good Maintainability). The details of Maintainability data are in Section 4.2. In order to use this dataset on the automated approach it was divided into two sets: Training Dataset (245 data points) and Test Dataset (121 data points). The evaluation dataset was not created due to the small size of the dataset.

NASA Data - This dataset is composed of 9510 data points and 21 different attributes. These software attributes are shown in Table 15. The number of classes to be predicted is 2 viz. Faulty (Class1) and Non-faulty (Class 0).

Table 15. Software Attributes (NASA Data)

#	ATTRIBUTE NAME
1	LOC_BLANK
2	BRANCH_COUNT
3	LOC_CODE_AND_COMMENT
4	LOC_COMMENTS
5	CYCLOMATIC_COMPLEXITY
6	DESIGN_COMPLEXITY
7	ESSENTIAL_COMPLEXITY
8	LOC_EXECUTABLE
9	HALSTEAD_CONTENT
10	HALSTEAD_DIFFICULTY
11	HALSTEAD_EFFORT
12	HALSTEAD_ERROR_EST
13	HALSTEAD_LENGTH
14	HALSTEAD_LEVEL
15	HALSTEAD_PROG_TIME
16	HALSTEAD_VOLUME
17	NUM_OPERANDS
18	NUM_OPERATORS
19	NUM_UNIQUE_OPERANDS
20	NUM_UNIQUE_OPERATORS
21	LOC_TOTAL

The NASA dataset has been used in two different splits on the automated approach to maximize and improve the productivity and overall result of the system.

NASA Data Split I

Training Dataset – 3340 data points

Evaluation Dataset – 3000 data points

Test Dataset – 3170 data points

NASA Data Split II

The dataset was divided into 3 training sets (T#1, T#2, T#3), 3 evaluation datasets (V#1, V#2, V#3) and 1 test dataset (TEST).

Each Training Dataset – 1700 data points

Each Evaluation Dataset – 700 data points

Test Dataset – 2310 data points

The idea behind this split was to generate as many as possible initial rules from training datasets and to rigorously test the effectiveness of the system. This split aimed at generating a large number of rules from the 3 different training datasets. Then all these rules were analyzed on three different evaluation data sets. For example, if there is a rule that was initially generated on training dataset T#1, and then fired on evaluation datasets V#1, V#2 and V#3 respectively, we had 3 evidences of goodness of this rule, and this would lead to a better decision if that rule should be kept or ignored. In this case, we would gain a strong evidence and confidence to do it.

Next, in the following section we present the results obtained for Maintainability data and the subsequent section will discuss the results on NASA data.

6.2 "Automated Generalization of Rules" on Maintainability data

This section discusses the experimental evaluation and results of implementing our program and methodology on Software Maintainability data. The Maintainability data was collected over various software applications to define three major levels or classes of maintainability; Good, Medium and Bad (Section 4.2).

Two independent rule generating tools were used: See5 and 4C Builder.

6.2.1 See5 Rule Generation

Table 16. Specification of See5 experiments with Software Maintainability data

Rule Builder: **See5**
Data: **Software Maintainability Data** (366 data points, 64 attributes, Classes: 1, 3, 5)
Training Dataset: **245** Data points
Evaluation Dataset: None
Testing Dataset: **121** Data points
Laplace Threshold: **0.70**
Initial Rules generated: **11**

As we see the dataset was divided into two sets: training and test dataset. The training dataset was used to generate the initial 11 rules by the See5 algorithm. Among these 11 rules generated, first five rules predict the class of the data points as Class 1 (Bad Maintainability), three rules for Class 3 (Medium Maintainability) and remaining three rules are for Class 5 (Good Maintainability). From the output file of See5, python module was generated. After generating the executable python module, the rules were used on the test dataset giving us {Rule, Class} pairs (Following the approach in Figure 3).

For example:

Initial rules generated by See5 for Class 5 (Good Maintainability) on training dataset are as follows:

Rule$_{C5\text{-}M\text{-}1}$: (17, 0, 0.947)[8] //Rule$_{C5\text{-}M\text{-}1}$ -> Rule generated by C5 on Maintainability data (M).

```
	WDC <= 16
	FACE <= 0
	MNL1 <= 21
```

[8] The triple (17, 0, 0.947) means that the rule properly classified 17 data points, that it misclassified 0 data point, and its value of Laplace ratio is 0.947.

```
        OPER <= 1
            ➔  class 5

Rule_C5-M-2: (42, 1, 0.956)
        MNL1 <= 25
        DAC <= 3
        HLOR <= 55
        WMC2 <= 27
            ➔  class 5

Rule_C5-M-3: (14, 1, 0.882)
        WDC <= 16
        FACE > 0
        MNL1 <= 21
        PROM <= 43
            ➔  class 5
```

When these rules are fired on the test dataset through the python module the result is as follows:

```
<Rule_C5-M-1>
  <rule_result>10:5-T 11:5-T 14:5-T 34:5-T 36:5-T 38:5-T 44:5-T 49:5-T
    55:5-T   58:5-T 71:5-T 80:5-T 85:5-T 92:5-T 94:5-T 97:5-T 100:5-
    T</rule_result>
  <rule_count>17 17 0</rule_count>
  <rule_laplace>0.947</rule_laplace>
  <rule_class>5</rule_class>
</Rule_C5-M-1>
```

This result is in the form of {data point: class predicted – True/False}. Example – {10:5 – T} -> the data point with id 10 of test dataset is predicted as Class 5 and the prediction is correct. The rule_count gives us the number of total data points classified; number of correct predictions; number of misclassifications. As shown above for the rule C5-M-1, total predictions made are 17; correct classification is 17 and misclassified are zero. New Laplace ratio of the rule on test data is calculated based on these numbers. Details about the format of output files can be found in Section 5.3.

```
<Rule_C5-M-2>
  <rule_result>3:5-T 8:5-T 10:5-T 11:5-T 14:5-T 15:5-T 21:5-T 28:5-T 29:5-T
    34:5-T 35:5-T 36:5-T 37:5-T 38:5-T 42:5-T 44:5-T 49:5-T 50:5-T 53:5-T
    55:5-T 58:5-T 61:5-T 62:5-T 71:5-T 73:5-T 79:5-T 80:5-T 81:5-T 84:5-T
    85:5-T 91:5-T 92:5-T 93:5-T 94:5-T 96:5-T 97:5-T 99:5-T 100:5-T 102:5-
    T 108:5-F 109:5-T 113:5-T 119:5-T</rule_result>
  <rule_count>43 42 1</rule_count>
  <rule_laplace>0.956</rule_laplace>
  <rule_class>5</rule_class>
</Rule_C5-M-2>
```

```
<Rule_C5-M-3>
  <rule_result>8:5-T 15:5-T 20:5-T 35:5-T 50:5-T 53:5-T 62:5-T 84:5-T 88:5-
    T 91:5-T 93:5-T 99:5-T 108:5-F 109:5-T 119:5-T</rule_result>
  <rule_count>15 14 1</rule_count>
  <rule_laplace>0.882</rule_laplace>
  <rule_class>5</rule_class>
</Rule_C5-M-3>
```

The new values of Laplace ratio (Laplace ratio calculated on test data in this case) for all the three rules range from .88 to .96. All these 3 rules for Class 5 have Laplace ratio values above the threshold value. Thus, these rules are considered further for comparison to derive the best rules for prediction of good maintainability. Comparing these 3 rules with each other the result is as follows (details of the format of rule comparison file are given in Section 5.3.):

```
<Rule_C5-M-1>

   <compared_rule_C5-M-2>17 0 26 78 Similarity: 0.395; Inclusion: 1.0, 0.395

   <compared_rule_C5-M-3>0 17 15 89 Similarity: 0.0; Inclusion: 0.0, 0.0

<Rule_C5-M-2>

    <compared_rule_C5-M-3>13 30 2 76 Similarity: 0.289; Inclusion: 0.302,
0.867
```

The above comparison of rules for Class 5 clearly shows that rule C5-M-2 is dominating other 2 rules. Rule C5-M-1 is completely included in rule C5-M-2 (Figure 6) and hence can be ignored without any loss of coverage and accuracy. Rule C5-M-3 is approximately 86 % included in the Rule C5-M-2 which is a good percentage of inclusion without much loss of coverage.

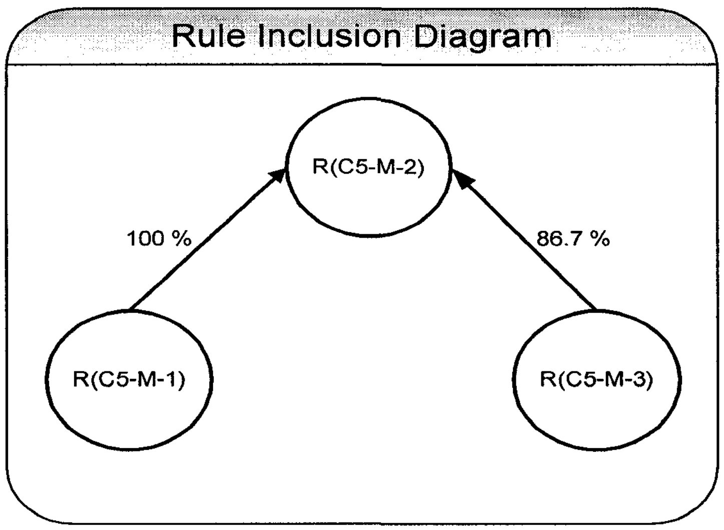


Figure 6. Rule Inclusion Diagram for See5 Rules on Maintainability Data (Class 5)

The fact that rule C5-M-2 appears to be more important and relevant is supported by the high Laplace ratio (0.956) it has among all the 3 rules. Eliminating Rule C5-M-3 highly depends on the type of application being studied and the requirements of the domain experts. There is a slight loss of coverage if Rule C5-M-3 is ignored, thus domain experts will have to critically analyze this loss of coverage before making any decision to eliminate this rule.

Another interesting observation was made during these experiments when we looked at these rules from the point of view of their syntax.

```
Rule_C5-M-1: (17, 0, 0.947)
      WDC <= 16
      FACE <= 0
      MNL1 <= 21
      OPER <= 1
          → class 5

Rule_C5-M-2: (42, 1, 0.957)
      MNL1 <= 25
      DAC <= 3
      HLOR <= 55
      WMC2 <= 27
          → class 5

Rule_C5-M-3: (14, 1, 0.882)
      WDC <= 16
      FACE > 0
      MNL1 <= 21
      PROM <= 43
         ➔  class 5
```

These rules are very different syntactically. For example, the first 2 rules above hardly use the same attributes. They have only one attribute in common i.e. MNL1 (Maximum method name length) but if we look at the results of comparison of these rules in Figure 7, we find that rule C5-M-1 is 100 percent included in rule C5-M-2. All the data points that are covered by rule C5-M-1 are also covered by rule C5-M-2. This proves the redundancy of these rules - they cover the same points. Thus, the important information to be mentioned here is that two rules might look so different syntactically but if we look at them based on semantic measures they both might cover the same data. In other words, even two syntactically different rules can provide us with the same knowledge. So, we aim at extracting such knowledge, and we can remove the rules which provide redundant information.

Next, for Class 1 (Bad maintainability) initially 5 rules were generated on training dataset. Firing these rules on the test dataset gave similar results as shown above for Class 5. Only 2 rules out of 5 qualified for further analysis with Laplace ratio values above the established threshold.

Rule$_{C5-M-4}$: (22, 0, 0.958) // On training data

```
      MNL2 <= 8
      HLOR > 933
         ➔  class 1
```

```
<Rule_C5-M-4>                          // On test data
  <rule_result>4:1-T 17:1-T 22:1-T 40:1-T 59:1-T 63:1-T 66:1-T 68:1-T 70:1-
   F 90:1-T 106:1-T 116:1-T</rule_result>
  <rule_count>12 11 1</rule_count>
  <rule_laplace>0.857</rule_laplace>
  <rule_class>1</rule_class>
</rule_C5-M-4>
```

Rule$_{C5-M-5}$: (12, 0, 0.929) // On training data

```
      INCL <= 1
      HLOR > 933
```

```
        → class 1

<Rule_C5-M-5>                    // On test data
  <rule_result>17:1-T 22:1-T 68:1-T 70:1-F 86:1-T 90:1-T 106:1-
    T</rule_result>
  <rule_count>7 6 1</rule_count>
  <rule_laplace>0.778</rule_laplace>
  <rule_class>1</rule_class>
</Rule_C5-M-5>
```

Comparing these rules:

```
<Rule_C5-M-4>
      <compared_rule_C5-M-5>6 6 1 108 Similarity: 0.462; Inclusion: 0.5,
0.857
```

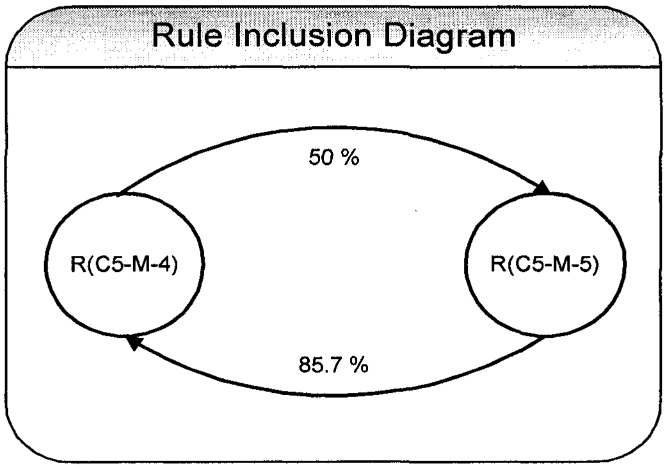


Figure 7. Rule Inclusion Diagram for See5 Rules on Maintainability Data (Class 1)

Comparison of these 2 rules (Figure 7) illustrates that the two rules are quite different based on their similarity and coverage. None of them is highly inclusive in the other. Thus, both the rules are used to predict bad maintainability.

Next, we evaluate the 3 initial rules generated for Class 3 by See5 on training dataset. Firing these rules on test dataset and ignoring the ones with Laplace ratio values less than 0.70 (threshold), 2 rules were left and they are as follows:

Rule$_{C5\text{-}M\text{-}6}$: (12, 4, 0.722)

```
      MNL1 > 25
      DAC <= 3
            → class 3
```

```
<Rule_C5-M-6>
  <rule_result>1:3-T 6:3-T 24:3-T 26:3-T 41:3-F 43:3-F 45:3-T 52:3-T 54:3-F
    64:3-T 65:3-T 70:3-T 82:3-F 83:3-T 103:3-T 111:3-T</rule_result>
  <rule_count>16 12 4</rule_count>
  <rule_laplace>0.722</rule_laplace>
  <rule_class>3</rule_class>
</Rule_C5-M-6>
```

Rule$_{C5-M-7}$: (52, 19, 0.726)
    HLOR > 55
        ➔ class 3

```
<Rule_C5-M-7>
  <rule_result>1:3-T 2:3-T 4:3-F 5:3-T 6:3-T 7:3-T 9:3-T 12:3-T 13:3-T 16:3-T
    17:3-F 18:3-F 19:3-T 20:3-F 22:3-F 23:3-F 25:3-F 26:3-T 31:3-T 32:3-T
    33:3-T 39:3-T 40:3-F 43:3-F 45:3-T 46:3-T 47:3-T 48:3-T 51:3-T 52:3-T
    56:3-T 57:3-T 59:3-F 60:3-T 63:3-F 65:3-T 66:3-F 67:3-T 68:3-F 69:3-T
    70:3-T 72:3-T 74:3-T 75:3-T 76:3-T 77:3-T 78:3-T 82:3-F 83:3-T 86:3-F
    87:3-T 88:3-F 90:3-F 95:3-T 98:3-T 101:3-T 103:3-T 104:3-T 105:3-T
    106:3-F 107:3-T 110:3-T 111:3-T 112:3-T 114:3-T 115:3-T 116:3-F
    117:3-T 118:3-T 120:3-T 121:3-T</rule_result>
  <rule_count>71 52 19</rule_count>
  <rule_laplace>0.726</rule_laplace>
  <rule_class>3</rule_class>
</Rule_C5-M-7>
```

Comparing Rules for Class 3 (Figure 8):

```
<Rule_C5-M-6>

      <compared_rule_C5-M-7>12 4 59 46 Similarity: 0.16 Inclusion: 0.75,
0.169
```

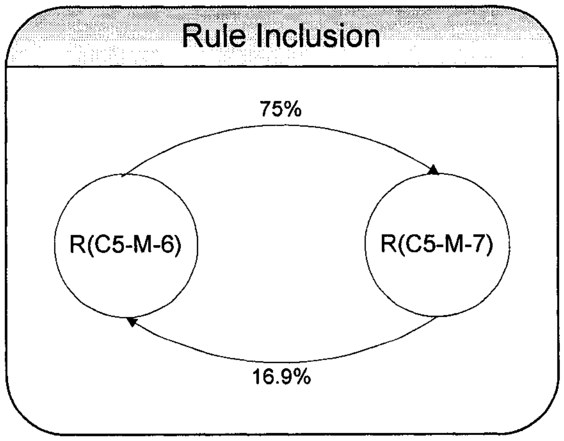


Figure 8. Rule Inclusion Diagram for See5 Rules on Maintainability Data (Class 3)

Rules C5-M-6 and C5-M-7 for Class 3 have low similarity and inclusion values. Based on the above comparison of these two rules none of them can be ignored. Both rules are used to predict the Class 3 on Maintainability data.

6.2.2 4C Rule Generation

Table 17. Specification of 4Cexperiments with Software Maintainability data

Rule Builder: **4C Rule Builder**
Data: **Software Maintainability Data** (366 data points, 64 attributes, Classes: 1, 3, 5)
Training Dataset: **245** Data points
Evaluation Dataset: None
Testing Dataset: **121** Data points
Laplace Threshold: **0.60**
Initial Rules generated: **9**

Among the initial rules generated on Maintainability training dataset, one rule is for Class 1, five rules for Class 3 and three rules are for Class 5. Given below are the three rules for Class 5 (Good Maintainability) generated on training data by 4C rule builder.

Rule$_{4C\text{-}M\text{-}1}$ (24, 0, 0.962):
```
If [ <ATOK != [64.50,81.00),[50.50,64.50),[96.50,118.00),[118.00,144.00),
              [201.00,322.00),[81.00,96.50),[144.00,169.00)>
<MTOK!= [52.50,68.00),[42.50,52.50),[68.00,103.00)>
<DEC ! = [9.50,20.50),[20.50,32.50), [32.50,48.50),[48.50,62.00),[81.00,124.00),
         [62.00,81.00),[124.00,925.00)>
<FACE != [3.50,4.50),[0.50,2.50),[2.50,3.50)><RCR !=[1.20,2.75)>
<RFC ! = [385.00,401.00),[37.50,52.50),[483.00,1.16E3),[441.00,483.00),
         [70.50,111.00)>
<MNL3 !=[21.50,38.00)>
<MNL4 !=[23.50,41.50)>
<ATCO !=[83.00,128.00),[49.00,64.50),[128.00,887.00)>
<HLPL !=[310.00,350.00),[350.00,412.00),[1.96E3,2.43E3),[541.00,680.00),
         [810.00,1.06E3),[680.00,810.00),[1.06E3,1.43E3),[1.43E3,1.96E3),[2.43
         E3,3.43E3),[3.43E3,1.99E4),[456.00,541.00),[412.00,456.00)>
<CHCL !=[1.50,5.50)><CONS !=[3.50,5.50),[5.50,9.00)>
<OPER !=[21.50,34.50),[34.50,58.00),[58.00,212.00)>]
Then Class = 5
```

Rule$_{4C\text{-}M\text{-}2}$ (20, 0, 0.955):

```
If [ <TYPE !=3,4>
<LOC !=[98.50,122.00),[869.00,1.22E3),[203.00,250.00),[364.00,466.00),
        [250.00,303.00),[579.00,713.00),[1.22E3,8.16E3),[303.00,364.00),[170.0
        0,203.00),[466.00,579.00),[11.50,25.00),[148.00,170.00),
        [713.00,869.00)>
<ATCO !=[36.50,49.00),[7.50,17.50),[17.50,26.50),[83.00,128.00),[64.50,83.00),
        [49.00,64.50),[128.00,887.00)>
<HLPL !=[310.00,350.00),[130.00,172.00),[350.00,412.00),[1.96E3,2.43E3),
        [810.00,1.06E3),[680.00,810.00),[1.06E3,1.43E3),[1.43E3,1.96E3),[2.43E
        3,3.43E3),[3.43E3,1.99E4),[456.00,541.00),[172.00,235.00),[412.00,456.
        00),[58.50,87.50),[235.00,310.00),[87.50,130.00)>
<PUBM !=[12.50,25.50),[25.50,35.50),[66.50,76.50),[-0.50,12.50)>]
Then Class = 5
```

Rule$_{4C-M-3}$ (12, 2, 0.813):

```
If [ <LOC !=[-1.00,11.50),[122.00,148.00),[65.00,80.50),[98.50,122.00),
     [869.00,1.22E3), [203.00,250.00),[364.00,466.00),[250.00,303.00),
     [579.00,713.00),[1.22E3,8.16E3),[303.00,364.00),[43.50,65.00),
     [25.00,43.50), [170.00,203.00),[80.50,98.50),[466.00,579.00),
     [148.00,170.00), [713.00,869.00)>
<MNL3 !=[8.50,15.50),[15.50,21.50)>]
Then Class = 5
```

The performance of these initial three rules for Class 5 on test dataset is as follows:

```
<Rule_4C-M-1>
  <rule_result>10:5-T 11:5-T 14:5-T 23:5-T 28:5-T 29:5-T 34:5-T 36:5-T
   38:5-T 44:5-T 49:5-T 58:5-T 61:5-T 71:5-T 73:5-T 79:5-T 80:5-T 85:5-T
   92:5-T 94:5-T 96:5-T 97:5-T 100:5-T 102:5-T</rule_result>
  <rule_count>24 24 0</rule_count>
  <rule_laplace>0.962</rule_laplace>
  <rule_class>5</rule_class>
</Rule_4C-M-1>

<Rule_4C-M-2>
  <rule_result>10:5-T 15:5-T 34:5-T 35:5-T 36:5-T 41:5-T 42:5-T 44:5-T
   49:5-T 58:5-T 62:5-T 79:5-T 80:5-T 91:5-T 92:5-T 93:5-T 96:5-T 97:5-T
   109:5-T 113:5-T</rule_result>
  <rule_count>20 20 0</rule_count>
  <rule_laplace>0.955</rule_laplace>
  <rule_class>5</rule_class>
</Rule_4C-M-2>

<Rule_4C-M-3>
  <rule_result>10:5-T 14:5-T 17:5-F 34:5-T 36:5-T 38:5-T 51:5-F 58:5-T
   71:5-T 79:5-T 80:5-T 92:5-T 97:5-T 102:5-T</rule_result>
  <rule_count>14 12 2</rule_count>
  <rule_laplace>0.813</rule_laplace>
  <rule_class>5</rule_class>
</Rule_4C-M-3>
```

All the rules above for Class 5 possess high Laplace ratio (threshold Laplace ratio for 4C is .60) and hence will be compared with each other.

```
<Rule_4C-M-1>
  <compared_rule_4C-M-2>11 13 9 88 Similarity: 0.333; Inclusion: 0.458, 0.55
  <compared_rule_4C-M-3>12 12 2 95 Similarity: 0.462; Inclusion: 0.5, 0.857

<Rule_4C-M-2>
  <compared_rule_4C-M-3>8 12 6 95 Similarity: 0.308; Inclusion: 0.4, 0.571
```

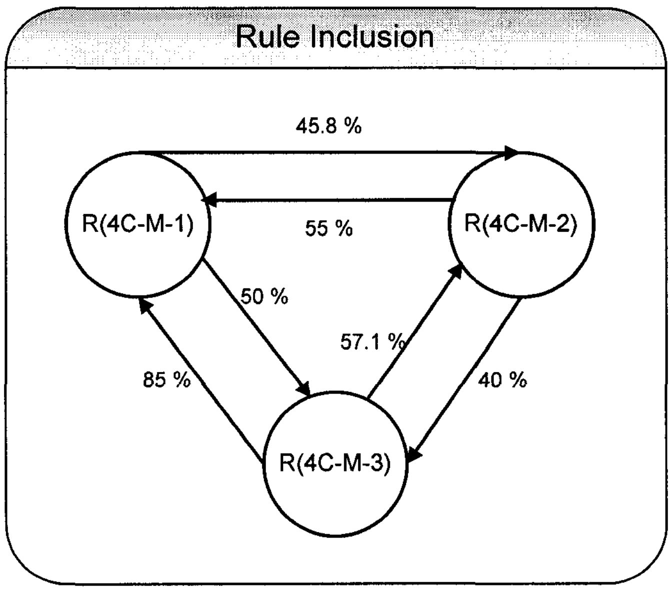


Figure 9. Rule Inclusion Diagram for 4C Rules on Maintainability Data (Class 5)

Comparison of these rules does not give any dominating rules. But, definitely it will give domain experts some insight to the semantics of the rules. Thus, rules 4C-M-1, 4C-M-2 and 4C-M-3 are selected as best rules by 4C to predict good maintainability (Class 5) on Maintainability data.

Now, we look at initial rules generated by 4C for Class 3 (Medium Maintainability). 5 rules were generated by 4C on training dataset. These rules were fired on test dataset. 3 rules were left after ignoring rules with low Laplace ratio values.

Rule$_{4C-M-4}$ (40, 21, 0.65):
If [<TYPE !=4>
<LOC !=[869.00,1.22E3),[25.00,43.50),[1.22E3,8.16E3),[43.50,65.00),[122.00,148.00)>

```
<TOK !=[3.57E3,4.25E3),[52.00,86.00),[6.23E3,3.65E4),[121.00,173.00)>
<DEC !=[62.00,81.00),[124.00,925.00)>
<INCL !=[2.50,5.50),[9.50,16.00)>
<LCOM !=[119.00,170.00)>
<HLEF !=[82.50,323.00),[6.08E5,9.72E5),[ 0.50,82.50),[2.04E6,1.23E7),
         [323.00,694.00),[3.94E3,8.13E3),[1.73E3,2.34E3),[1.25E4,2.05E4)>
<HLOR !=[-0.50,12.50),[1.72E3,1.01E4),[26.50,39.50)>
<MEMB !=[40.50,67.00)><PKGM !=[14.50,35.50)>]
Then Class = 3
```

Rule$_{4C-M-5}$ (10, 5, 0.647):

```
If [ <TYPE !=2,3>
<LOC !=[250.00,303.00),[579.00,713.00),[25.00,43.50),[1.22E3,8.16E3),
        [43.50,65.00),[11.50,25.00),[303.00,364.00),[203.00,250.00),[98.50,122.
        00)>
<TOK !=[1.25E3,1.41E3),[4.25E3,6.23E3),[52.00,86.00),
        [-2.00,24.50),[24.50,52.00),[781.00,1.02E3),[1.02E3,1.25E3),
        [583.00,663.00)>
<ATOK !=[96.50,118.00),[144.00,169.00),[322.00,771.00),[22.50,37.50)>
<MTOK !=[179.00,736.00),[68.00,103.00)>
<FACE !=[3.50,4.50),[2.50,3.50),[0.50,2.50)>
<LCOM !=[601.00,1.37E3),[1.37E3,1.43E4),[49.00,84.50)>
<RFC !=[12.50,22.50),[336.00,358.00),[0.50,12.50),[111.00,297.00)>
<HLEF !=[82.50,323.00),[1.73E5,2.33E5),[694.00,1.73E3),
         [-0.50,82.50),[2.04E6,1.23E7),[2.34E3,3.94E3),[8.23E4,1.23E5),
         [2.05E4,2.96E4)>]
Then Class = 3
```

Rule$_{4C-M-6}$ (3,1, 0.667):

```
If [ <LOC !=[-1.00,11.50),[869.00,1.22E3),[1.22E3,8.16E3),[43.50,65.00),
             [65.00,80.50),
             [364.00,466.00),[466.00,579.00),[713.00,869.00),[203.00,250.00)>
<RCC1 !=[7.05,8.80),[8.80,10.30),[22.90,34.40),[11.90,14.50),[10.30,11.90),
         [14.50,16.60),[4.65,5.75),[57.00,434.00)>
<RCC2 !=[17.50,26.50),[90.00,112.00),[73.50,90.00),[112.00,136.00),
         [26.50,35.50), [61.00,73.50)>
<TOK !=[3.57E3,4.25E3),[3.05E3,3.57E3),[4.25E3,6.23E3),[6.23E3,3.65E4),
        [24.50,52.00),[1.41E3,1.84E3),[781.00,1.02E3),[263.00,374.00),[1.02E3,1
        .25E3),[1.84E3,3.05E3),[374.00,523.00)>
<DINH !=[3.50,4.50),[0.50,2.50)>
<RFC !=[323.00,336.00),[401.00,441.00),[483.00,1.16E3),[52.50,70.50),
        [0.50,12.50),[111.00,297.00),[385.00,401.00),[297.00,323.00),[358.00,38
        5.00)>]
Then Class = 3
```

The coverage and performance of these rules on test data is as follows:

```
<Rule_4C-M-4>
  <rule_result>1:3-T 2:3-T 3:3-F 4:3-F 5:3-T 6:3-T 9:3-T 10:3-F 12:3-T 13:3-T
    17:3-F 19:3-T 22:3-F 23:3-F 24:3-T 26:3-T 29:3-F 33:3-T 34:3-F 36:3-F
    39:3-T 43:3-F 44:3-F 45:3-T 48:3-T 49:3-F 51:3-T 52:3-T 57:3-T 58:3-F
    60:3-T 67:3-T 70:3-T 72:3-T 74:3-T 76:3-T 77:3-T 78:3-T 79:3-F 80:3-F
    82:3-F 83:3-T 85:3-F 87:3-T 92:3-F 94:3-F 95:3-T 97:3-F 101:3-T 102:3-F
    104:3-T 105:3-T 107:3-T 108:3-T 110:3-T 111:3-T 114:3-T 115:3-T
    117:3-T 118:3-T 120:3-T</rule_result>
  <rule_count>61 40 21</rule_count>
  <rule_laplace>0.651</rule_laplace>
  <rule_class>3</rule_class>
</ Rule_4C-M-4>


< Rule_4C-M-5>
  <rule_result>6:3-T 7:3-T 26:3-T 32:3-T 49:3-F 51:3-T 58:3-F 63:3-F 80:3-F
    87:3-T 89:3-T 98:3-T 105:3-T 116:3-F 121:3-T</rule_result>
  <rule_count>15 10 5</rule_count>
  <rule_laplace>0.647</rule_laplace>
  <rule_class>3</rule_class>
</ Rule_4C-M-5>

< Rule_4C-M-6>
  <rule_result>16:3-T 24:3-T 40:3-F 46:3-T</rule_result>
  <rule_count>4 3 1</rule_count>
  <rule_laplace>0.667</rule_laplace>
  <rule_class>3</rule_class>
</ Rule_4C-M-6>
```

They are compared for similarity and inclusion measures and result is as follows:

```
<Rule_4C-M-4>
  <compared_rule_4C-M-5>5 56 10 50 Similarity: 0.111; Inclusion: 0.125, 0.5
  <compared_rule_4C-M-6>1 60 3 57 Similarity: 0.024 Inclusion: 0.025, 0.333
</Rule_4C-M-4>

<Rule_4C-M-5>
  <compared_rule_4C-M-6>0 15 4 102 Similarity: 0.0 Inclusion: 0.0, 0.0
</Rule_4C-M-5>
```

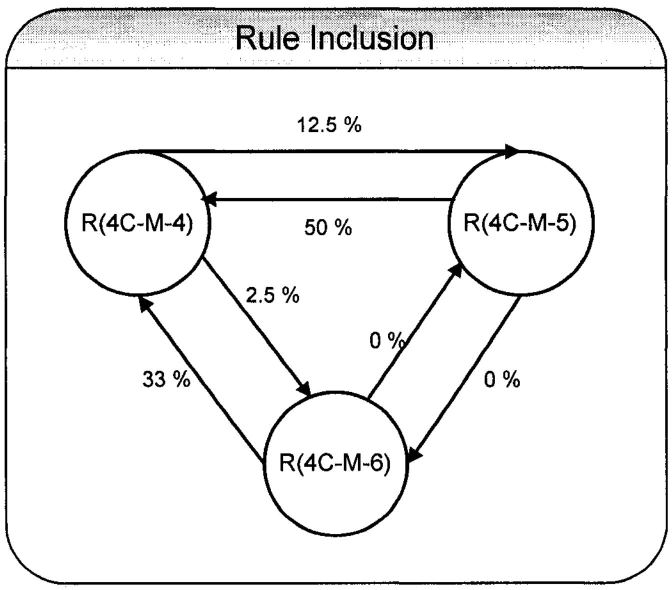


Figure 10. Rule Inclusion Diagram for 4C Rules on Maintainability Data (Class 3)

The comparison numbers clearly show none of the rules can be ignored. They are very different semantically and show very low redundancy.

Next, there is only 1 rule generated for Class 1 by 4C on Maintainability data. The rule is shown below and has a Laplace ratio value below threshold. The rule is ignored due to low Laplace ratio on test dataset.

```
Rule4C-M-7 (6, 25, 0.212):
If [<RCC1 !=[7.05,8.80),[10.30,11.90),[22.90,34.40),[34.40,57.00),
          [57.00,434.00),[-0.85,3.45),[11.90,14.50),[3.45,4.65),[8.80,10.30)>
<LCOM !=[-0.50,10.50),[10.50,28.50),[28.50,49.00),[84.50,119.00),
          [325.00,601.00)>
<HLVL != [81.00,161.00),[2E3,2.25E3),[549.00,895.00),[2.25E3,2.7E3),
          [3.74E3,4.9E3),[6.02E3,7.61E3),[1.57E3,2E3),[161.00,270.00),
          [- 1.00,37.00),[3.01E3,3.74E3),[1.07E3,1.57E3),[2.7E3,3.01E3),
          [895.00,1.07E3),[37.00,81.00),[387.00,549.00),[270.00,387.00),
          [1.94E4,2.99E4),[1.58E4,1.94E4)>]
Then Class = 1
```

Table 18 gives the summary of which rules have been found unique and important by both the rule-generating tools independently on Maintainability data.

Table 18. Summary of Best Rules on Maintainability Data

	Class 5	Class 3	Class 1

See5	**R(C5-M-2)**
	R(C5-M-3)
		R(C5-M-6)
		R(C5-M-7)
			R(C5-M-4)
			R(C5-M-5)

4C Rule Builder	**R(4C-M-1)**
	R(4C-M-2)
	R(4C-M-3)
		R(4C-M-4)
		R(4C-M-5)
		R(4C-M-6)
			--

6.2.3 Combined Analysis of Rules generated by See5 and 4C

Before experimenting with the NASA data, another major task was to compare the rules generated by See5 and 4C Builder. Till now we have been comparing rules according to the class they belong to and the tool that generated them. After finding the relevant and important rules generated by one tool, they could be compared with the relevant rules generated by the other tool. This will further generalize the rules making them more universal and powerful over the entire set of rules generated by both tools.

For example, let us take a look at the rules for Class 5 both for See5 and 4C rule builder. From Table 18, R(C5-M-2) and R(C5-M-3) generated from See5 and R(4C-M-1), R(4C-M-2) and R(4C-M-3) generated by 4C are found to have high Laplace ratio values and to be quite unique (all highly inclusive rules were ignored) for Class 5 Maintainability data. Now, let us compare[9] these rules with each other

```
<R(C5-M-2)>
  <compared_rule_R(4C-M-1)>23 20 1 77 Similarity: 0.523 Inclusion: 0.535, 0.958
  <compared_rule_R(4C-M-2)>19 24 1 77 Similarity: 0.432 Inclusion: 0.442, 0.95
  <compared_rule_R(4C-M-3)>12 31 2 76 Similarity: 0.267 Inclusion: 0.279, 0.857
</R(C5-M-2)>

<R(C5-M-3)>
  <compared_rule_R(4C-M-1)> 0 15 24 82 Similarity:0.0 Inclusion:0.0,0.0
  <compared_rule_R(4C-M-2)> 6 9 14 92 Similarity:0.207 Inclusion:0.4,0.3
```

[9] Comparisons shown here are on Test Dataset (121 data point)

```
<compared_rule_R(4C-M-3)> 0 15 14 92 Similarity:0.0 Inclusion:0.0,0.0
</R(C5-M-3)>
```

Figure 11 shows the comparison chart of inclusions of See5 and 4C rules.

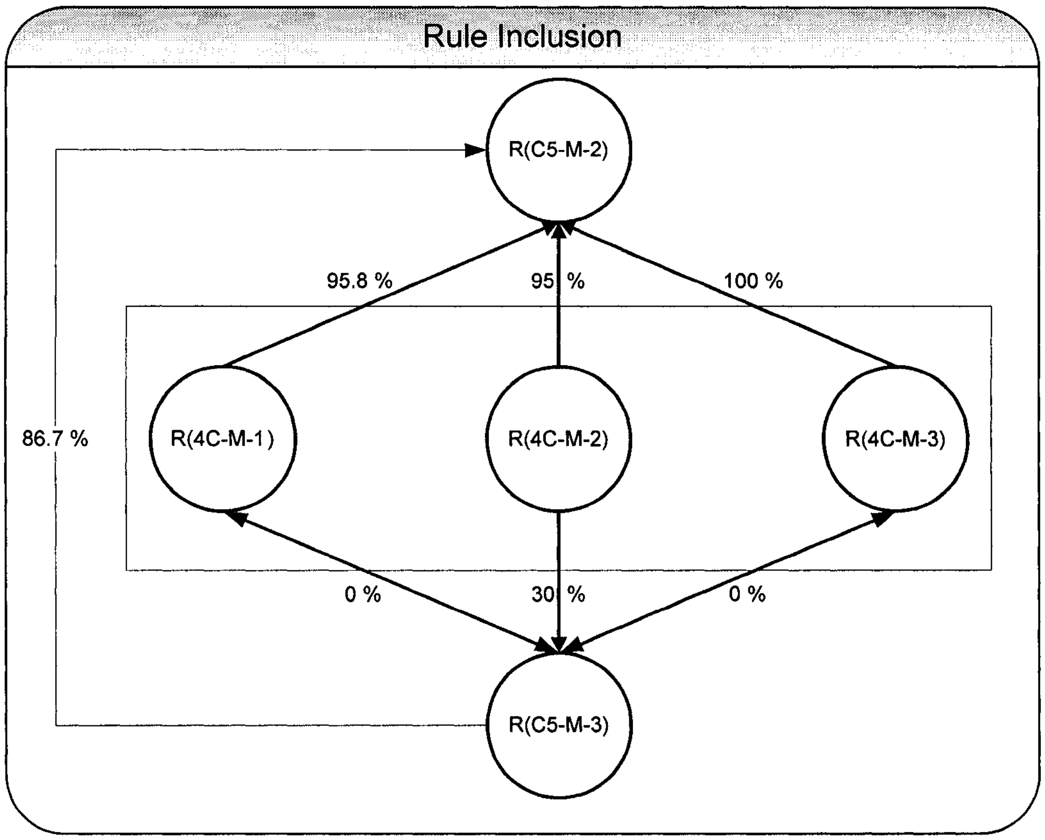


Figure 11. See5 and 4C Rule Comparison

Inclusions among all the rules have not been shown to ease readability. Clearly, we see that R(C5-M-2) i.e. Rule 2 generated by See5 on Maintainability data for Class 5 is more dominating as it covers a good percentage of all the other rules. This rule defines good maintainability in the following way: class is of good maintainability if maximum method name length is less than 26; data abstraction coupling i.e. Number of reference types used in the attribute declaration is less than or equal to 3; number of halstead operators is less than 56 and weighted methods per class is less than or equal to 27.

There are two possible ways to analyze this kind of a scenario. We already know that rule C5-M-2 is the most dominating among all the rules generated by both tools and thus, can be used alone to predict Class 5 (good maintainability) on Maintainability data without much loss of coverage.

Alternative option for the domain experts or user is to use the second set of rules i.e. R(C5-M-3), R(4C-M-1), R(4C-M-2) and R(4C-M-3). Comparisons of these rules along with Figure 11, clearly illustrates that this set of rules is very distinctive and dissimilar to each other. Thus, this set of 4 rules can be used instead of using a single rule C5-M-2 to predict Class 5 for Maintenance data.

Next, we show comparison of See5 and 4C rules for Class 3 on Maintainability data. From Table 18, R(C5-M-6) and R(C5-M-7) best predict Class 3 by See5. And R(4C-M-4), R(4C-M-5), R(4C-M-6) best define Class 3 by 4C rule builder. Comparing these rules:

```
<Rule_C5-M-6>
<compared_rule_4C-M-4>9 7 52 53 Similarity: 0.209 Inclusion: 0.75, 0.225
<compared_rule_4C-M-5>2 14 13 92 Similarity: 0.1 Inclusion: 0.167, 0.2
<compared_rule_4C-M-6>1 15 3 102 Similarity: 0.071 Inclusion: 0.083, 0.333
</Rule_C5-M-6>

<Rule_C5-M-7>
<compared_rule_4C-M-4>38 33 23 27 Similarity: 0.704 Inclusion: 0.731, 0.95
<compared_rule_4C-M-5>9 62 6 44 Similarity: 0.17 Inclusion: 0.173, 0.9
<compared_rule_4C-M-6>2 69 2 48 Similarity: 0.038 Inclusion: 0.038, 0.667
</Rule_C5-M-7>
```

The comparison results of See5 and 4C rules for Class 3, clearly distinguish all these rules. Only 1 case of high similarity and inclusion is found here. Rule C5-M-7 and 4C-M-4 are 70% similar and show high inclusions. Rule 4C-M-4 is 95% covered in Rule C5-M-7 and thus can be ignored leaving behind 4 best rules – R(C5-M-6), R(C5-M-7), R(4C-M-5) and R(4C-M-6).

Thus the automated system presented in the thesis offers a mechanism to find generalized rules based on their semantic comparison. This mechanism provides deep insight into the performance of rules and thus enables the domain experts and software engineers to choose rules that interest them the most. The system presented here considers distinguishable measures of rules such as Laplace ratio, accuracy, and coverage to find the unique and important rules, as well as identifies a set of rules that can be considered for removal. It will be in hands of the domain analysts and engineers to study these figures and results and then ignore any rule.

6.2.4 True-positives Comparison

The above comparisons were made irrespective to the correct predictions made by rules. In other words, the system only looked at the data points that were covered by a given rule and considered those points as coverage of rule without checking if those data points have been classified correctly or not. Thus, all rules were compared again considering the true positives predictions made by the rules. Now onwards, all the comparisons will be made taking into consideration only the correct predictions of the rules.

The rules compared in Figure 11 were again compared based on only the true-predictions made by these rules. Not much of a difference was noted when rules were compared over their true-positive predictions. Rule C5-M-2 was still found dominating over the other rules. The reason behind not much difference being found when considering rules only with true-predictions is due to the fact that the rules selected for comparisons have high Laplace ratio and very little misclassified predictions.

6.3 "Automated Generalization of Rules" on NASA data

This section discusses the Experimental evaluation and results of implementing our program and methodology on NASA data. Two independent rule generating software programs were used: See5 and 4C Builder.

6.3.1 See5 Rule Generation

Table 19. Specification of See5 experiments on NASA data

Rule Builder: **See5**
Data: **NASA Data** (9510 data points, 21 attributes, Classes: 0, 1)
Training Dataset: **3340** Data points
Evaluation Dataset: **3000** Data points
Testing Dataset: **3170** Data points
Laplace Threshold: **0.70**
Initial Rules generated: **8**

As compared to the Maintainability data, NASA data is a large dataset with 9510 data points in total. Initially 8 rules were generated by See5: four rules for Class 0 (Non-Faulty) and next four for Class 1 (Faulty). Unlike the Maintainability dataset, this dataset has been divided into training, evaluation and testing. Training dataset is used to generate the initial rules using the rule-generating software. Evaluation dataset is used to analyze these rules for their Laplace ratio values, goodness and uniqueness. The best rules found are finally fired on test dataset to see if these selected best rules give the best results.

Beginning with rules for Class 0 generated by See 5 on NASA training data, the rules look as follows:

R$_{(C5\text{-}N\text{-}1)}$: (0, 0, 0.5)
IF LOC_BLANK > 1 & CYCLOMATIC_COMPLEXITY <= 8 & DESIGN_COMPLEXITY > 3 & LOC_EXECUTABLE <= 14
THEN Class: 0

R$_{(C5\text{-}N\text{-}2)}$: (1465, 37, 0.975)
IF CYCLOMATIC_COMPLEXITY <= 1 & NUM_OPERATORS > 8

THEN Class: 0

R(C5-N-3): (2720, 59, 0.978)
IF LOC_BLANK <= 1 & CYCLOMATIC_COMPLEXITY <= 8
THEN Class: 0

R(C5-N-4): (1582, 26, 0.983)
IF CYCLOMATIC_COMPLEXITY <= 8 & DESIGN_COMPLEXITY <= 3 & NUM_UNIQUE_OPERANDS <= 68
THEN Class: 0

These rules are fired on the evaluation dataset. According to the Laplace ratio values calculated for the evaluation data, only 3 rules out of 4 are left for further analysis.

Table 20. Performance of Rules for Class 0 (Non-faulty) on evaluation data

Training Data Rules	Laplace on Evaluation Data	Comment
R(C5-N-1)	-	Ignored due to low Laplace ratio
R(C5-N-2)	97.5	Qualify
R(C5-N-3)	97.8	Qualify
R(C5-N-4)	98.3	Qualify

Comparing the rules with high Laplace ratio (Table 20):

```
<Rule_C5-N-2>
  <compared_rule_C5-N-3>1465 0 1255 285 Similarity: 0.539
                                        Inclusion: 1.0, 0.54
  <compared_rule_C5-N-4>706 759 876 659 Similarity: 0.302
                                        Inclusion: 0.482, 0.44

<Rule_C5-N-3>
  <compared_rule_C5-N-4>1582 1138 0 280 Similarity: 0.582
                                        Inclusion: 0.582, 1.0
```

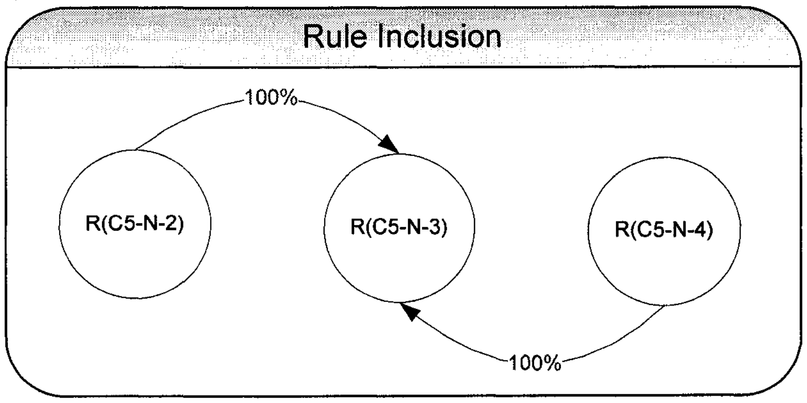


Figure 12. Rule Inclusion Diagram for See5 Rules (Data: NASA; Class: 0)

From Figure 12, we see that R(C5-N-3) appears to cover all the other rules for Class 0 generated by See5 on NASA Data (Figure 10). After ignoring rules R(C5-N-2) and R(C5-N-4) only 1 rule for Class 0 on NASA data is left. This discovered best rule is fired on the test dataset of 3170 data points and the result is promising.

```
R(C5-N-3): (2720, 59, 0.978)                      // On Evaluation Dataset
        LOC_BLANK <= 1
        CYCLOMATIC_COMPLEXITY <= 8
      → class 0

R(C5-N-3): (2879, 96, 0.967)                      // On Test Dataset
    LOC_BLANK <= 1
    CYCLOMATIC_COMPLEXITY <= 8
      → class 0
```

Next, we move to Rules generated by See5 for Class 1. Initially on training dataset 4 rules were generated to predict Class 1. These rules when fired on evaluation data have Laplace ratio as follows:

Table 21. Performance of Rules for Class 1 (Faulty) on evaluation data

Training Data Rules	Laplace on Evaluation Data	
R(C5-N-5)	0.5	Ignored due to low Laplace ratio
R(C5-N-6)	0.5	-- do --
R(C5-N-7)	0.112	-- do --
R(C5-N-8)	0.5	-- do --

Unfortunately none of the rules from Table 21 qualified for further analysis. All of them had a Laplace ratio values below the threshold when they were fired on the evaluation dataset. Thus, no rule generated by See5 was found good enough to predict Class 1 (faulty) on NASA data.

6.3.2 4C Rule Generation

Table 22. Specification of 4C experiments with NASA data

Rule Builder: **4C Rule Builder**
Data: **NASA Data** (9510 data points, 21 attributes, Classes: 0, 1)
Training Dataset: **3340** Data points
Evaluation Dataset: **3000** Data points
Testing Dataset: **3170** Data points
Laplace Threshold: **0.60**
Initial Rules generated: **216**

4C Rule Builder on NASA data gives a total of 216 rules on training dataset. Out of these 105 rules are for Class 0 (Non-Faulty) and rest 111 rules are for Class 1 (Faulty). Finding useful rules from such a large set of rules was a big challenge for a person. However, our automated system very aptly, was able to reduce the number of rules generated by removing redundant and low accuracy rules.

Firstly, 105 rules for Class 0 are fired on evaluation dataset. Only 42 rules are found to have Laplace ratio values greater than the threshold set for Laplace ratio for 4C builder. These 42 rules are compared and the most inclusive or rules with redundant coverage are ignored by the system. This leaves a total of 29 4C generated rules that are important and unique and are best to predict the Class 0 on NASA data. These 29 rules when tested on the Test Dataset gave the most promising results with most rules having a high Laplace ratio ranging between 0.92 to 0.99 with an exception of only 3 rules that range between 0.75 to 0.80. Coverage and Laplace ratio for all 29 rules is shown in Table 23.

Table 23. Performance of Best Rules by See5 on NASA Data on Test Data

Rule #	**# data points correctly classified**	**# data points misclassified**	**Laplace Ratio on Test Data**
1	500	14	0.971
2	302	4	0.984
3	336	8	0.974
4	53	0	0.982
5	268	1	0.993
6	313	5	0.981
7	514	10	0.979
8	31	0	0.97
9	24	0	0.962
10	12	0	0.929
11	120	0	0.992
12	311	4	0.984
13	201	2	0.985
14	101	2	0.971
15	21	0	0.957
16	216	3	0.982
17	129	4	0.963
18	87	1	0.978
19	216	2	0.986
20	2	0	0.75
21	41	1	0.955
22	198	3	0.98
23	107	0	0.991
24	73	1	0.974
25	7	1	0.80
26	71	3	0.947
27	17	1	0.9
28	444	3	0.991
29	129	0	0.992

Only 1 rule out of the 29 best rules discovered is shown here due to space constraints.

Rule$_{4C\text{-}N\text{-}1}$ (120, 0, 0.992):
If [< BRANCH_COUNT !=[341.00,988.00),[27.50,33.50),[227.00,341.00),[166.00,227.00),
[40.00,47.50),[63.50,73.50),[134.00,148.00),[33.50,40.00),[15.50,
21.50),[47.50,56.00), [21.50,27.50),[92.50,100.00),[8.50,15.50),
[56.00,63.50),[114.00,134.00),[100.00,114.00), [73.50,84.00),
[84.00,92.50),[148.00,166.00)>
< HALSTEAD_CONTENT !=[150.00,571.00),[104.00,150.00),[83.10,104.00),[68.00,83.10),
[52.20,57.90), [39.00,42.80),[47.00,52.20),[57.90,68.00),
[27.00,29.70),[42.80,47.00), [22.20,24.60), [32.60,36.10),
[36.10,39.00),[14.80,17.50),[19.90,22.20),[2.58,12.00),
[17.50,19.90),[12.00,14.80)>
< HALSTEAD_DIFFICULTY !=[68.50,408.00),[44.80,52.30),[52.30,68.50), [28.90,31.80),
[26.70,28.90), [24.10,26.70), [35.10,39.50),[39.50,44.80),

```
                      [31.80,35.10),[19.90,21.80), [21.80,24.10),[15.80,17.80),
                      [17.80,19.90),[13.90,15.80),[12.20,13.90), [0.75,5.11),
                      [7.24,8.97)>
< HALSTEAD_EFFORT !=[2.41E5,4.06E7),[1.19E5,2.41E5),[7.25E4,1.19E5), [4.94E4,7.25E4),
                      [3.61E4,4.94E4),[2.61E4,3.61E4),[1.43E4,1.9E4), [1.12E4,1.43E4),
                      [1.9E4,2.61E4), [8.61E3,1.12E4),[6.37E3,8.61E3), [3.86E3,5.09E3),
                      [1.04E3,1.56E3),[5.09E3,6.37E3),[275.00,633.00),[2.98E3,3.86E3),
                      [6.00,275.00)>
< LOC_TOTAL !=[582.00,4.22E3),[200.00,224.00),[464.00,582.00), [97.50,114.00),
                      [320.00,362.00),[114.00,130.00),[81.50,97.50),[224.00,246.00),
                      [277.00,320.00),[146.00,163.00),[65.50,81.50),[130.00,146.00),
                      [163.00,180.00),[49.50,65.50),[33.50,49.50),[180.00,200.00),
                      [246.00,277.00),[362.00,464.00),[17.50,33.50)>]
Then Class = 0
```

On the other hand, out of 111 rules generated for Class 1 by 4C on NASA training data, only 2 rules qualify with high values of Laplace ratio on evaluation data. These rules are R(4C-N-2) and R(4C-N-3) with Laplace ratios 0.667 and 0.75, respectively. They are compared for similarity and inclusion.

```
<Rule_4C-N-2>
  <compared_rule_4C-N-3>1 0 1 2998 Similarity: 0.5 Inclusion: 1.0, 0.5
</Rule_4C-N-2>
```

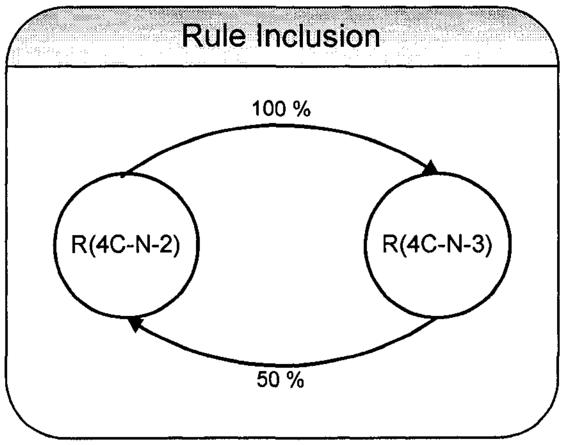


Figure 13. Rule Inclusion Diagram for 4C Rules on NASA Data (Class 1)

From Figure 13, rule 4C-N-2 is fully inclusive in 4C-N-3. They are 50% similar to each other. Thus, rule 4C-N-2 can be ignored as all the data points covered by it are also covered by 4C-N-3. The dominating rule 4C-N-3 is fired on test dataset.

Rule$_{\mathbf{4C\text{-}N\text{-}3}}$ (60, 5, 0.91):

```
If [ < LOC_EXECUTABLE !=[503.00,3.44E3),[384.00,503.00),[82.50,95.50),[109.00,122.00),
                        [95.50,109.00),[239.00,269.00),[27.50,41.50),[13.50,27.50),[41.
                        50,55.50), [136.00,150.00), [69.50,82.50),[181.00,209.00),
                        [55.50,69.50),[122.00,136.00), [150.00,164.00),[164.00,181.00),
                        [209.00,239.00),[269.00,328.00),[328.00,384.00)>
<HALSTEAD_CONTENT !=    [104.00,150.00), [42.80,47.00),[150.00,571.00),[83.10,104.00),
                        [52.20,57.90),[32.60,36.10),[57.90,68.00), [39.00,42.80),
                        [24.60,27.00), [2.58,12.00),[68.00,83.10), [14.80,17.50),
                        [22.20,24.60),[19.90,22.20),[36.10,39.00),[17.50,19.90),
                        [29.70,32.60),[47.00,52.20)>
< HALSTEAD_PROG_TIME !=[1.35E4,2.25E6),[36.00,59.30),[2.83E3,4.1E3),
                        [2.05E3,2.83E3),[6.72E3,1.35E4),[1.48E3,2.05E3),
                        [364.00,494.00),[15.70,36.00),[4.1E3,6.72E3),[809.00,1.09E3),
                        [170.00,221.00),[638.00,809.00),[90.40,126.00),[1.09E3,1.48E3)
                        ,[494.00,638.00),[59.30,90.40),[0.32,15.70),[296.00,364.00)>
< HALSTEAD_VOLUME !=[6.08E3,9.37E4),[158.00,223.00), [2.14E3,2.71E3),
                        [1.74E3,2.14E3), [3.73E3,6.08E3), [1.03E3,1.22E3),
                        [2.71E3,3.73E3),[278.00,337.00),[885.00,1.03E3),[478.00,557.0
                        0),[6.20,95.50),[656.00,760.00),[1.22E3,1.42E3),[337.00,406.00
                        ),[1.42E3,1.74E3),[406.00,478.00),[557.00,656.00),
                        [760.00,885.00),[95.50,158.00)>
< NUM_UNIQUE_OPERANDS!=[110.00,120.00), [189.00,272.00),[56.50,64.50),
                        [48.50,56.50),[98.50,110.00),[32.50,40.50),[144.00,163.00),
                        [16.50,24.50),[40.50,48.50),[0.50,8.50),[24.50,32.50),
                        [64.50,72.50),[120.00,134.00),[72.50,80.50),[80.50,88.50),
                        [88.50,98.50),[134.00,144.00),[163.00,189.00)>]
Then Class = 1
```

After thorough analysis it was found that the possible reason for a low number of rules found useful for Class 1 could be the ratio between data points that belong to Class 1 and Class 0. In NASA dataset, out of 9510 data points there are only 1765 data points of Class 1 and rest 7745 are for Class 0. We assumed that the uneven split of the Class 1 data points in training, evaluation and test datasets might be the cause of not finding many good rules for Class1.

Table 24. Summary of Best Rules on NASA Data

	Class 0	**Class 1**
See5	**R(C5-N-3)**	**None**
4C Rule Builder	**29 distinct rules**	**R(4C-N-3)**

6.3.3 Combined Analysis of Rules generated by See5 and 4C rules

From Table 24 above, there is only 1 rule generated by See5 that is found interesting for Class 0. On the other side for 4C rule builder, 29 rules are discovered as important and unique to predict Class 0 on NASA data. Comparing rule C5-N-3 with 29 rules of 4C:

```
<Rule_C5-N-3>[10]
<compared_NASA_4Crules>
<compared_rule_1>494 2385 6 285 Similarity: 0.171 Inclusion: 0.172, 0.988
<compared_rule_2>293 2586 9 282 Similarity: 0.101 Inclusion: 0.102, 0.97
<compared_rule_3>322 2557 14 277 Similarity: 0.111 Inclusion: 0.112, 0.958
<compared_rule_4>53 2826 0 291 Similarity: 0.018 Inclusion: 0.018, 1.0
<compared_rule_5>268 2611 0 291 Similarity: 0.093 Inclusion: 0.093, 1.0
<compared_rule_6>313 2566 0 291 Similarity: 0.109 Inclusion: 0.109, 1.0
<compared_rule_7>514 2365 0 291 Similarity: 0.179 Inclusion: 0.179, 1.0
<compared_rule_8>29 2850 2 289 Similarity: 0.01 Inclusion: 0.01, 0.935
<compared_rule_9>24 2855 0 291 Similarity: 0.008 Inclusion: 0.008, 1.0
<compared_rule_10>12 2867 0 291 Similarity: 0.004 Inclusion: 0.004, 1.0
<compared_rule_11>120 2759 0 291 Similarity: 0.042 Inclusion: 0.042, 1.0
<compared_rule_12>311 2568 0 291 Similarity: 0.108 Inclusion: 0.108, 1.0
<compared_rule_13>201 2678 0 291 Similarity: 0.07 Inclusion: 0.07, 1.0
<compared_rule_14>101 2778 0 291 Similarity: 0.035 Inclusion: 0.035, 1.0
<compared_rule_15>21 2858 0 291 Similarity: 0.007 Inclusion: 0.007, 1.0
<compared_rule_16>216 2663 0 291 Similarity: 0.075 Inclusion: 0.075, 1.0
<compared_rule_17>128 2751 1 290 Similarity: 0.044 Inclusion: 0.044, 0.992
<compared_rule_18>86 2793 1 290 Similarity: 0.03 Inclusion: 0.03, 0.989
<compared_rule_19>216 2663 0 291 Similarity: 0.075 Inclusion: 0.075, 1.0
<compared_rule_20>2 2877 0 291 Similarity: 0.001 Inclusion: 0.001, 1.0
<compared_rule_21>41 2838 0 291 Similarity: 0.014 Inclusion: 0.014, 1.0
<compared_rule_22>198 2681 0 291 Similarity: 0.069 Inclusion: 0.069, 1.0
<compared_rule_23>107 2772 0 291 Similarity: 0.037 Inclusion: 0.037, 1.0
<compared_rule_24>73 2806 0 291 Similarity: 0.025 Inclusion: 0.025, 1.0
<compared_rule_25>5 2874 2 289 Similarity: 0.002 Inclusion: 0.002, 0.714
<compared_rule_26>71 2808 0 291 Similarity: 0.025 Inclusion: 0.025, 1.0
<compared_rule_27>16 2863 1 290 Similarity: 0.006 Inclusion: 0.006, 0.941
<compared_rule_28>444 2435 0 291 Similarity: 0.154 Inclusion: 0.154, 1.0
<compared_rule_29>129 2750 0 291 Similarity: 0.045 Inclusion: 0.045, 1.0
</compared_NASA_4Crules>
</Rule_C5-N-3>
```

The above comparison clears shows that Rule C5-N-3 dominates all the rules generated by 4C rule builder.

[10] Comparisons shown here are on Test Dataset (3170 data points)

```
R(C5-N-3): (2879, 96, 0.967)                    // On Test Dataset
        LOC_BLANK <= 1
        CYCLOMATIC_COMPLEXITY <= 8
            ➔  class 0
```

The rule classifies 2975 data points from the total of 3170 data points in test dataset. Out of these 2975 total predictions made; 2879 are correctly classified and 96 data points are misclassified. Very low misclassification with high Laplace ratio shows why the rule is very dominating over all the other rules.

6.3.4 Alternative Approach to Generation of Rules

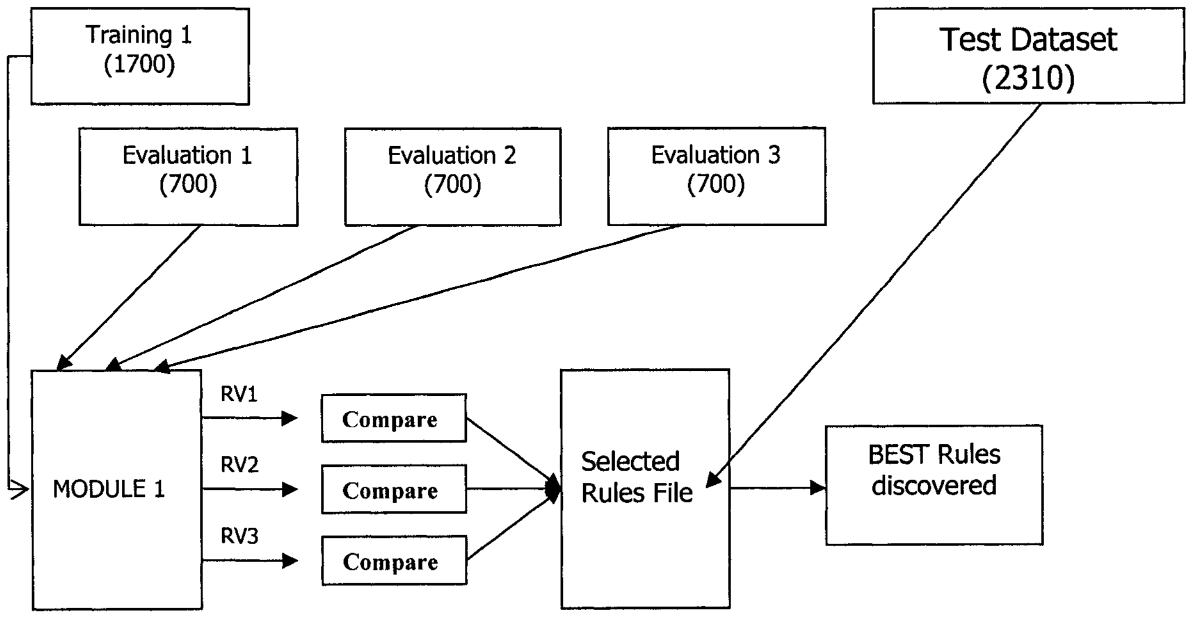
NASA Data was now split in a different way – NASA DATA Split II (Section 6.1). A more careful distribution of the dataset was performed in this split with equal percentage of Class 1 data points in all the slatted datasets. 1765 data points of Class 1 were very carefully divided into training, evaluation and test datasets. Looking at Figure 14, each training dataset (T#1, T#2, T#3) is made up 1700 data points; each evaluation dataset (V#1, V#2, V#3) is made up 700 data points and the rest 2310 data points compile the test dataset.

See5 Rule Generation

Table 25. Specification of See5 experiments on NASA data Split II

Rule Builder: **See5**
Data: **NASA Data** (9510 data points, 21 attributes, Classes: 0, 1)
Training Dataset (T#1, T#2, T#3): **1700** Data points in each dataset
Evaluation Dataset (V#1, V#2, V#3): **700** Data points in each dataset
Testing Dataset: **2310** Data points
Laplace Threshold: **0.70**
Initial Rules generated: **Total 11 Rules – (3 from T#1, 4 from T#2, 4 from T#3)**

From Table 25, we see that 11 rules were generated by See5 on 3 different training datasets. Rule parser will generate 3 python modules (Module 1, Module 2 and Module 3) each for T#1, T#2 and T#3, respectively. For example, in this case Module 1 contains 3 rules from T#1 out of which 1 rule is for Class 0 (non-faulty) and 2 rules are for Class 1 (faulty). These 3 rules are fired on V#1, V#2 and V#3 datasets. The results were put in three separate rules files; RV1, RV2 and RV3. RV1 contains results of how rules from Module 1 performed on V#1 data. Similarly, RV2 and RV3 contain results of firing rules in Module 1 on V#2 and V#3, respectively. Figure 14, shows the data flow and approach followed during the experiments on NASA Data split II.



Note: Module 2 and Module 3 not shown for simplification of the process

Figure 14. Process Flow: NASA Data Split II

The 3 separate sets of rules (RV1, RV2, RV3) are analyzed and compared based on their importance and uniqueness which gives us the set of best rules.

First, rules for Class 0 generated by all three training datasets were evaluated to yield best amongst them.

Table 26. Laplace Ratio of Class 0 rules on Evaluation datasets

Modules	**Rules**	**V#1**	**V#2**	**V#3**
Module 1 (T#1)	R(C5-N2-1)	0.80	0.809	0.777
Module 2 (T#2)	R(C5-N2-2)	0.82	0.841	0.774
	R(C5-N2-3)	0.802	0.812	0.786
Module 3 (T#3)	R(C5-N2-4)	0.742	0.745	0.736

All the rules in Table 26 have a high Laplace ratio when fired on evaluation datasets. The rules are as follows:

```
Rule_C5-N2-1
IF NUM_UNIQUE_OPERANDS <= 63
   THEN Class: 0
```

Rule C5-N2-1 is fired on V#1, V#2 and V#3. Laplace ratio values of this rule on V#1, V#2, V#3 are 0.80, 0.809, 0.777, respectively (Table 26). The rule qualifies with high Laplace ratio and since there is no other rule in Module 1 for Class 0, this rule is passed on to the selected rules file.

Next, we look at rules from Module 2 on three evaluation datasets. 2 rules for Class 0 qualified with high Laplace ratio when fired on all 3 evaluation datasets separately. These rules are as follows:

```
Rule_C5-N2-2:
   IF LOC_CODE_AND_COMMENT <= 3 & HALSTEAD_CONTENT <= 74.07 &
   NUM_UNIQUE_OPERATORS <= 21
   THEN Class: 0

Rule_C5-N2-3:
   IF LOC_BLANK <= 21 & NUM_UNIQUE_OPERATORS <= 21
   THEN Class: 0
```

They are compared and the result is:

```
<Rule_C5-N2-2>
<compared_rule_C5-N2-3>278 60 280 82 Similarity:0.621 Inclusion:1.0,0.621
```

Rule C5-N2-2 is 100% included in rule C5-N-3 and hence can be ignored. Thus, from Module 2 rule C5-N2-3 is passed on to the selected rules file.

Similarly, from Module 3 the following rule is selected:

Rule$_{C5-N2-4}$:
IF HALSTEAD_CONTENT <= 107.18
THEN Class: 0

The selected 3 (C5-N2-1, C5-N2-3 and C5-N2-4) rules are now tested to predict the data points of TEST dataset. All three rules, when fired on test dataset had high Laplace ratio (> 0.91) and were hardly similar or inclusive with each other. The result is as follows:

```
<Rule_C5-N2-1>
<compared_rule_C5-N2-3>71 230 209 1800 Similarity: 0.139
                                       Inclusion: 0.236, 0.254
<compared_rule_C5-N2-4>190 46 219 1855 Similarity: 0.418
                                       Inclusion: 0.805, 0.465
</Rule_C5-N2-1>

<Rule_C5-N2-3>
<compared_rule_C5-N2-4>70 197 109 2624 Similarity: 0.186
                                       Inclusion: 0.262, 0.391
</Rule_C5-N2-3>
```

Thus, 3 rules are found to best predict Class 0 (non-faulty) on NASA data Split II.

Similarly, the rules generated by See 5 for Class 1 on T#1, T#2 and T#3, were evaluated on 3 evaluation datasets. Initially, a total of 7 rules were generated by See 5 on all training datasets (2 on T#1, 2 on T#2 and 3 rules generated on T#3 by See5 for Class1). The performance of these rules on V#1, V#2 and V#3 is given below in Table 27.

Table 27[11]. Laplace Ratio of Class 1 rules on Evaluation datasets

[11] Empty Entries in the Table represent that no data point was classified by that rule.

Modules	**Rules**	**V#1**	**V#2**	**V#3**
Module 1 (T#1)	R(C5-N2-5)	0.519	0.436	0.36
	R(C5-N2-6)	-	.75	-
Module 2 (T#2)	R(C5-N2-7)	-	-	-
	R(C5-N2-8)	-	-	-
Module 3 (T#3)	R(C5-N2-9)	0.654	0.667	0.441
	R(C5-N2-10)	-	-	-
	R(C5-N2-11)	-	-	-

The Laplace ratio threshold value here is 0.70. Only one rule C5-N2-6 on V#2 has a Laplace ratio of .75 which is above the threshold value. But, its performance on V#1 and V#3 is not good. It does not classify even a single data point from V#2 or V#3. Thus, none of the rules in Table 27 qualify for further analysis.

Table 28. Summary of Best See5 Rules on NASA Data split II

	Class 0	**Class 1**

See5	**R(C5-N2-1)**
	R(C5-N2-3)
	R(C5-N2-4)
		None

Table 28 summarizes the number of best rules generated by See5 on NASA Data split II.

<u>4C Rule Generation</u>

Table 29. Specification of 4C experiments on NASA data Split II

Rule Builder: **4C**
Data: **NASA Data** (9510 data points, 21 attributes, Classes: 0, 1)
Training Dataset (T#1, T#2, T#3): **1700** Data points in each dataset
Evaluation Dataset (V#1, V#2, V#3): **700** Data points in each dataset
Testing Dataset: **2310** Data points
Laplace Threshold: **0.60**
Initial Rules generated: **Total 201 Rules – (76 from T#1, 61 from T#2, 64 from T#3)**

The approach in Figure 14, was applied to rules generated by 4C on NASA Data split II, exactly the same way as applied on rules generated by See5. From Table 29, we see that more than 200 rules were generated by 4C rule builder on 3 different training datasets. Rule parser will generate 3 python modules (Module 1, Module 2 and Module 3) each for T#1, T#2 and T#3, respectively. For example, Module 1 contains 76 rules from T#1 out of which 48 rules are for Class 0 (non-faulty) and rest 28 are for Class 1 (faulty).

16 rules are discovered as best rules for Class 0. And for Class 1 only 1 rule is found to be interesting from initial 76 rules generated on T#1.

Similarly, rules from Module 2 and Module 3 will be validated on V#1, V#2 and V#3 and then compared for similarity and inclusion. There are 61 rules in Module 2, 30 for Class 0 and 31 for Class 1. Out of these, only 8 rules are left for Class 0 and unfortunately none of the rules for Class 1 have a good Laplace ratio. Next, from a total of 64 rules in Module 3 only 10 rules are found best for Class 0 and 2 rules for Class 1.

In all, 34 rules for Class 0 and 3 rules for Class 1 were selected from V#1, V#2 and V#3. The selected rules obtained are then tested on TEST dataset to discover the best rules. 31 rules out of 34 for Class 0 were found be the most useful and unique. These rules have Laplace ratio values ranging from .87 to .96 on test dataset.

For Class 1 only 1 rule is left out of 3. Laplace ratio for this rule on test dataset is 0.91.

In conclusion, the developed automated tool was experimentally evaluated on real world data. The best rules discovered by the system will help domain experts to extract knowledge from these rules useful in building high quality softwares.

7. SUMMARY AND CONCLUSION

7.1 Conclusions

In a broader aspect the problem addressed here was to find useful and non-redundant rules from the large sets of discovered association rules. The research essentially aimed at measuring software quality factors using association rules and then finding the best set of rules that can predict the quality or underlying behavior of the software. The research represents a methodology useful for semantic comparison of rules using measures of similarity and inclusion. Application of these measures gives a chance to look "inside" rules and creates possibility of making a selection of interesting rules representing analyzed data. The work describes how certain characteristics of rules such as "importance" and "uniqueness" can help remove redundant and non-similar rules leaving behind the ones that are distinctive and highly accurate.

The proposed methodology has been applied to process Software Maintenance data and NASA data; former representing the levels of maintainability of software objects and later representing the evaluation of faulty or non-faulty behavior of the software attributes. A fully-functional automated system was developed using the proposed methodology. This tool is capable of reading rules from the rule-generating tools and then fires them on the desired dataset. It performs all the calculations based on similarity and inclusion of rules giving us the net result of good and unique rules. The tool was implemented and experimented on real-time software data and proved to be fast, accurate and efficient.

The major contributions of my research work are stated as follows.

- A fully-functional automated system that searches for rules that not only possess high accuracy but are also unique and interesting.
- The system evaluates association or classification rules in Software Quality domain. The best rules discovered by the system will be helpful to domain experts and users in many ways. The representation of these rules is in simple if-then formats and can be used for feature selection, software quality estimations and decision making.
- It is very flexible and adaptable system as engineers can take independent modules from the system and merge them with their own code with ease.

7.2 <u>Future Work</u>

A lot of future work is possible in this direction.

1. Merging Rules: Instead of removing weaker rules altogether we can merge them with strong rules. This will help in eradicating any chances of loss of coverage due to ignoring rules.
2. Comparison of methods that find interesting rules: As stated earlier, this is an ongoing research to find interesting rules from large collection of rules and many researchers have tried to define and express the interestingness of rules. It will be highly recommendable if a comparison of all these studies can be carried out.
3. Combining various methods of finding interesting rules: As we see in [28] the major limitation stated is redundancy in rules which has been covered and taken care of by the presented research work. So, if we combine the work of using templates [28] for

recognizing useful rules with semantic comparison of rules, certainly new and promising developments can be established.

REFERENCES

[1] Aggarwal K. K., Singh Y., and Chhabra J. K.: An Integrated Measure of Software Maintainability. In *Proc. of Annual Reliability and Maintainability Symposium*, IEEE, 2002.

[2] Ash D., Alderete J., Yao L., Oman P. W., and Lowther B.: Using software maintainability models to track code health. In *Proc. of International Conference on Software Maintenance*, IEEE, 1994

[3] Attneave, F.: Dimensions of similarity. *American Journal of Psychology*, 63, (1950) 516-556.

[4] Evanco, W. M.: Analyzing Change Effort in Software During Development. In *Proceedings of Sixth International Software Metrics Symposium* (METRICS'99), (1999), 179-188.

[5] Evanco, W. M.: Prediction Models for Software Fault Correction Effort. In *Proc. of the Fifth Conference on Software Maintenance and Reengineering*, CSMR 2001, 14-16 March 2001, Lisbon, Portugal, (2001) 114-120.

[6] Fenton, N.E. and Neil, M.: A Critique of Software Defect Prediction Models. In *Machine Learning Applications in Software Engineering* (eds: Zhang D, Tsai JJP), World Scientific Publishing Co, (2005) 72-86.

[7] Glass, R.: Software maintenance documentation. *Annual ACM Conference on Systems Documentation*, Pittsburgh, PA (1989).

[8] Halstead M. H., Elements of Software Science, *Operating, and Programming Systems Series Volume 7*, Elsevier, 1977.

[9] Hayes, J. H., Patel, S. C., and Zhao, L.: A Metrics-Based Software Maintenance Effort Model. In *Proc. of the Eighth European Working Conference on Software Maintenance and Reengineering* (CSMR'04), (2004) 254-260.

[10] Jorgensen M.: Experience with the accuracy of Software Maintenance Task Effort Prediction Models. *IEEE Transactions of Software Engineering*, 21(8) (1995) 674-681.

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

[12] Oman P. and Hagemeister J.: Metrics for Assessing a Software System's Maintainability. In *Proc. of Conference on Software Maintenance*, IEEE, 1992.

[13] Reformat M., Pedrycz, W., and Pizzi, N.: Building a Software Experience Factory using Granular-based Models. Fuzzy Sets and Systems, 145(1), (2004) 111-139.

[14] SEI Software Technology Review, *Halstead Complexity Measures*, URL: http://www.sei.cmu.edu/, 2002

[15] SEI Software Technology Review, *Maintainability Index Technique for Measuring Program Maintainability*, URL: http://www.sei.cmu.edu/

[16] Smith, D.: *Designing maintainable software*. New York, Springer, 1999.

[17] Tversky, A.: Features of Similarity. *Psychological Review*, 84, (1977) 327-352.

[18] Rakesh Agarwal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In VLDB '94, September 1994.

[19] Gregory Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, Knowledge Discovery in Databases, pages 229 – 248. AAAI Press/ The MIT Press, Menlo Park, CA, 1991.

[20] Peter Hoschka and Willi Klosgen. A support system for interpreting statistical data. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, Knowledge Discovery in Databases, pages 229 – 248. AAAI Press/ The MIT Press, Menlo Park, CA, 1991.

[21] Jiawei Han, Yandong Cai, and Nick Cercone. Knowledge discovery in databases: an attribute-oriented approach. In Proceedinds of the 18th International Conference on Very Large Databases (VLDB), pages 547 – 559, August 1992.

[22] Marek Reformat, Aashima Kapoor, Nicolino J. Pizzi. "Software Maintenance: Similarity and Inclusion of Rules in Knowledge Extraction," 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06) pp. 723-731

http://doi.ieeecomputersociety.org/10.1109/ICTAI.2006.106

[23] Shusaku Tsumoto, Shoji Hirano, "Visualization of Rule's Similarity using Multidimensional Scaling," *icdm*, p. 339, Third IEEE International Conference on Data Mining (ICDM'03), 2003

[24] Jiye Li, Nick Cercone, "Introducing a Rule Importance Measure," DBLP 2006, pp: 167-189

[25] Blackboard Systems, Daniel D. Corkill, Blackboard Technology Group, Inc.

[26] Buschmann, F., R. Meunier, H. Rohnert, P. Sommerlad, and M.Stal. Pattern Oriented Software Architecture: A System of Patterns. West Sussex, England: John Wiley & Sons Ltd., 1996

[27] Freitas, A.A.: On rule interestingness measures. To appear in Knowledge-Based Systems journal (1999) http://citeseer.ist.psu.edu/freitas99rule.html

[28] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, Hannu Toivonen, and A. Inkeri Verkamo. Finding interesting rules from large sets of discovered association rules. In CIKM-94, 401 -- 407, November 1994

APPENDIX I

Program Requirements:

- Python 2.4 or higher (http://www.python.org)
- PyXML (http://pyxml.sourceforge.net/). Future python distributions may include PyXML extensions natively - try running the program without PyXML first and only install, if necessary.

Rule Parser syntax:

```
python 4C_parse.py <dataset name> <rules filename>
python C5_parse.py <dataset name> <rules filename> <names filename>
```

For example: 'python 4C_parse.py NASA 4C_NASA_Rules' will create a module named NASA_4C.

To run the parsers under windows you must:

- Install MinGW (http://www.mingw.org/)
- Install pexports (http://starship.python.net/crew/kernr/mingw32/pexports-0.42h.zip, copy pexports.exe to your MinGW/bin directory)
- In MSYS, execute the following commands (change the paths and version numbers if necessary):

```
cd /c/Python24/libs
pexports /c/Python24/python24.dll > python24.def
dlltool --dllname python24.dll --def python24.def --output-lib libpython24.a
```

- Run the parser like normal from within MSYS

If you have compilation problems, visit

http://www.mingw.org/MinGWiki/index.php/Python%20extensions

Config file (XML format) must have the following elements:

- training_data (name of the file containing the data used to train the rule builder)
- eval_data or test_data (name of the file containing the evaluation or test data)
- data_delimiter (character or string used to separate the elements of each data point)
- number_of_classes (number of classes in the data set)
- model_list (list of python module names of models you wish to use to model the data)