A Hybrid Approach to Analyze Empirical Software Engineering Data and its Application Predict Module Fault-proneness in Maintenance

Authors:
Sandro Morasca
Günther Ruhe

Accepted for publication in
Journal of Systems and
Software

IESE-Report No. 026.99/E
Version 1.0
May 10, 1999

Abstract

Knowledge discovery from software engineering measurement data is essential in deriving the right conclusions from experiments. Various data analysis techniques may provide data analysts with different and complementary insights into the studied phenomena. In this paper, two data analysis techniques —Rough Sets and Logistic Regression— are compared, from both the theoretical and the experimental point of view. In particular, the empirical study was performed as a part of the ESPRIT/ESSI project CEMP on a real-life maintenance project, the DATATRIEVE™ project carried out at Digital Engineering Italy. We have applied both techniques to the same data set. The goal of the experimental study was to predict module fault-proneness and to determine the major factors affecting software reliability in the application context. The results obtained with either analysis technique are discussed and compared. Then, a hybrid approach is built, by integrating different and complementary knowledge obtained from either approach on the fault-proneness of modules. This knowledge can be reused in the organizational framework of a company-wide experience factory.

Keywords: Knowledge Discovery, Hybrid Approach, Rough Sets, Logistic Regression, Empirical Case Studies, Experience Factory, Fault-proneness, Software Maintenance.

Table of Contents

1 Introduction

Discovery, representation, and reuse of software engineering development know-how are crucial for improvement of software quality in an organization. The Quality Improvement Paradigm (QIP) (Basili et al., 1994) offers a general evolutionary framework for performing systematic quality improvement by means of reuse of experience. Software measurement and analysis of measurement data from carefully designed experiments are essential parts of the QIP.

Careful data analysis and knowledge discovery are necessary for deriving the right conclusions from data. In a field like software engineering, it is important to make the most out of the available data sets, which may be smaller than in other application fields. No "silver bullet" technique exists answering all questions for different situations and circumstances in an optimal way. On the contrary, it is reasonable to expect that different techniques will have context-sensitive strengths and weaknesses. Furthermore, different insights will likely be gained from the different assumptions underlying them and from the use of different analysis processes. Therefore, we believe that it is important to study, assess, and compare several different techniques.

In this paper, we will study the application of two techniques, Logistic Regression (Hosmer and Lemeshow, 1989) and Rough Sets (Pawlak, 1991) to the analysis of measurement data of a real-life maintenance project, the DATATRIEVE™ project carried out at Digital Engineering Italy. The experiment was performed as a part of the CEMP project (CEMP, 1995), an ESPRIT/ESSI project funded by the Commission of the European Communities. The overall project was devoted to the customized establishment of goal-oriented measurement programs which were based on the *Goal/Question/Metric* (GQM) paradigm (Basili and Weiss, 1984), (Basili and Rombach, 1988). The GQM paradigm was applied and evaluated in a case study replicated across three major companies. In all participating projects, goals related to reliability and reusability were studied, to increase consistency and comparability of results across the CEMP project.

Logistic Regression is a statistical classification technique based on maximum likelihood estimation that was originally developed for biomedical applications and has already been successfully used in a few software engineering applications (Briand et al., 1994). Based on the values of a given set of explanatory variables for the attributes of an object, Logistic Regression estimates the probability that the object may be classified as belonging to one of the possible categories. Therefore, Logistic Regression is quite different from other regression techniques (e.g., linear regression), which aim at finding out to what extent a relationship of a specified kind (e.g., linear) exists between explanatory and dependent variables.

Rough Set theory is a new and promising machine learning data analysis approach which has been successfully applied in many real-life problems of various areas, e.g. medicine, pharmacology, business, banking. Although no additional information such as independence of explanatory variables or assumptions on data distribution is required, conclusions can be drawn even for small sized data sets. As a result, computation of cause-effect relationships in a minimal knowledge representation is performed. Rough Sets have been used in knowledge engineering applications in the past. So far, few applications exist in software engineering (Ruhe, 1996).

The main objectives of the paper are to

- demonstrate the contributions of Logistic Regression and Rough Sets based analysis from experimental software engineering. The data were from a real-world case study devoted to predict fault-proneness of modules (i.e., the likelihood that they are faulty) under maintenance constraints;
- investigate the quality of predictions of the different approaches for a series of data sets of varying size;
- compare the performance of both approaches and suggest a hybrid approach combining their strengths;
- evaluate the results of the hybrid approach when applied to predict fault-proneness of modules.

We also discuss and compare the empirical results of the application of Logistic Regression, Rough Sets, and the new hybrid approach. We believe that these results can be helpful to practitioners in their own applications, but we also think that caution should be used before reusing these results "as-is." Our viewpoint —consistent with the use of the GQM paradigm and the objectives of the CEMP project, which advocate the *customized* introduction of measurement programs— is that, at this stage of technology, there are no models that are applicable unchanged to all software projects. The identification of such general models, which can be applied across all environments, is a long-term goal, which will be achieved only after a number of experiments are carried out and successfully replicated.

The remainder of this paper is organized as follows. In Section 2, the practical problem is described, including the experimental design and other context information. The analysis techniques —Logistic Regression and Rough Sets— are concisely described and compared in Section 3. In Section 4, we describe the analysis results we have obtained by applying Logistic Regression and Rough

Sets to the entire maintenance data set obtained in the DATATRIEVE™ project and eight randomly generated subsets of the entire data set, for comparison purposes. Based on these results, we introduce a hybrid approach (Section 5) that combines the strengths of either approach and therefore results in improved classification accuracy. Summary and concluding remarks follow in Section 6.

2 The Experimental Environment

Our experiment concerned the transition of the DATATRIEVE™ product from version 6.0 to version 6.1. The DATATRIEVE™ product was undergoing both adaptive (DATATRIEVE™ was being transferred from platform OpenVMS/VAX to platform OpenVMS/Alpha) and corrective maintenance (failures reported from customers were being fixed) at the Gallarate (Italy) site of Digital Engineering. At the time of the experiment, the DATATRIEVE™ team was composed of six people. The DATATRIEVE™ product was originally developed in the BLISS language. BLISS is an expression language. It is block-structured, with exception handling facilities, co-routines, and a macro system. It was one of the first non-assembly languages for operating system implementation.. Some parts were later added or rewritten in the C language. Therefore, the overall structure of DATATRIEVE™ is composed of C functions and BLISS subroutines. The empirical study of this paper reports only the BLISS part, by far the bigger one. In what follows, we will use the term "module" to refer to a BLISS module, i.e., a set of declarations and subroutines usually belonging to one file. More than 100 BLISS modules have been studied. It was important to the DATATRIEVE™ team to better understand how the characteristics of the modules and transition process were correlated with the code quality.

We followed the QIP-based measurement process used for all application experiments within the CEMP project. More details, along with guidelines for the application of the process, may be found at (CEMPwww) and in (CEMP, 1995). In particular, the Goal/Question/Metric (GQM) paradigm was used as a part of this QIP-based process to iteratively define metrics for those attributes which are important in the context of the measurement goals.

The approach described in (CEMPwww) and (CEMP, 1995) requires a close contact between the measurement team and the project team. Software measurement goals should be identified by the project team. To be effective, measurement must be carried out towards objectives that are relevant to the software organization under study. Also, the quality aspect (e.g., software reliability) along with its influencing factors (e.g., software size) must be identified by the project team, who has a sensible understanding of the important characteristics of the processes and products studied. We believe that this approach helps software organization better focus on the important objectives and characteristics.

This is the GQM goal that was established by the project team:

Analyze version 6.1 of DATATRIEVE™	*(object of study)*
for the purpose of understanding	(*purpose*)
with respect to the impact of modifications from version 6.0 to version 6.1 on reliability	(*quality focus*)
from the viewpoint of the project leader	(*viewpoint*)
in the following environment: Digital Italy – Gallarate	(*environment*)

The five dimensions of measurement goals help precisely identify what is investigated (*object of study*), why (*purpose*), with respect to which specific attribute (*quality focus*), for whose benefit (*viewpoint*), and in what context (*environment*). GQM goals are later refined into questions and metrics, to capture all relevant factors[1] in a quantitative fashion. In the above goal, the purpose "understanding" was chosen since our measurement study was carried out as the beginning of a measurement program. It was considered too ambitious to start a measurement program with a prediction or improvement goal, which can be established only after a clear understanding of the processes and projects has been obtained. At any rate, we also obtained results that can be used as a first kernel around which prediction models can be built.

The objective of our data analysis was to study whether it was possible to classify modules as non-faulty or faulty, based on a set of measures collected on the project. Therefore, the dependent variable of our study is

- **Faulty6.1**: its value is 0 for all those modules in which no faults were found; its value is 1 for all other modules.

Here, we will list the relevant explanatory variables that quantify factors deemed important by the developers and that will also be used in the experimental comparison between Rough Sets and Logistic Regression. We limited our study only to explanatory variables that measured factors suggested by the project team. Also, our study encompassed other factors indicated by the project team and the corresponding explanatory variables, but our experimental analysis showed that they were not relevant with respect to the quality focus of interest. For each module *m*, we found that the following explanatory variables were relevant:

- **LOC6.0:** number of lines of code of module *m* in version 6.0.
- **LOC6.1:** number of lines of code of module *m* in version 6.1.
- **AddedLOC**: number of lines of code that were added to module *m* in version 6.1, i.e., they were not present in module *m* in version 6.0.
- **DeletedLOC:** number of lines of code that were deleted from module *m* in version 6.0, i.e., they were no longer present in module *m* in version 6.1.

[1] An intermediate document, the abstraction sheet, was used to refine the GQM goal, and as a basis to derive the questions. Details about this augmentation to the GQM paradigm can be found in (Latum et al., 1998).

- **DifferentBlocks**: number of different blocks module *m* in between versions 6.0 and 6.1.
- **ModificationRate**: rate of modification of module *m*, i.e.,
- (AddedLOC + DeletedLOC) / (LOC6.0 + AddedLOC).
- **ModuleKnowledge**: subjective variable that expresses the project team's knowledge on module *m* (low or high)
- **ReusedLOC**: number of lines of code of module *m* in version 6.0 reused in module *m* in version 6.1.

3 A Concise Introduction to Logistic Regression and Rough Sets

Here, we briefly illustrate the basic concepts of Logistic Regression (Section 3.1) and Rough Sets (Section 3.2). For a comprehensive introduction, the reader may refer to (Hosmer and Lemeshow, 1989) and (Pawlak, 1991), respectively. The two data analysis techniques are then compared in Section 3.3, based on a set of desirable requirements.

3.1 Logistic Regression

Logistic Regression is a technique for estimating the probability that an object belongs to a specific class, based on the values of the explanatory variables that quantify the object's attributes. As such, Logistic Regression is different from other regression techniques (e.g., linear regression), whose goal is to determine whether there is some form of functional dependency (e.g., linear) between explanatory variables and dependent variable. Logistic Regression does not assume any strict functional form to link explanatory variables and the probability function. Instead, this functional correspondence has a flexible shape that can adjust itself to several different cases. Logistic Regression is based on maximum likelihood and assumes that all observations are independent.

Let us suppose that the dependent variable Y can take only the two values 0 and 1, like in our case, where Faulty6.1 will be the dependent variable Y. Let us suppose that there are a number of explanatory variables X_i. The measures we have collected on the DATATRIEVE™ modules (see the list at the end of Section 2) will be the explanatory variables in our case. The multivariate Logistic Regression model is defined by the following equation (if it contains only one independent variable, then we have a univariate Logistic Regression model):

$$p(Y=1) = \pi(X_1, X_2, \ldots, X_n) = \frac{e^{(C_0 + C_1 \bullet X_1 + \ldots + C_n \bullet X_n)}}{1 + e^{(C_0 + C_1 \bullet X_1 + \ldots + C_n \bullet X_n)}} ,$$

where p(Y=1) is the probability that Y = 1. The curve describing the relationship between p(Y=1) and any single X_i —i.e., under the assumption that all other X_j's are constant— has a flexible S shape which ranges between the following two extreme cases:

- A horizontal line, when variable X_i is not significant (probability p(Y=1) is a constant with respect to X_i);

- A vertical line, when variable X_i alone is able to differentiate between the case Y = 0 and the case Y = 1, i.e., based on the value of X_i alone, one can perfectly predict whether Y = 0 or Y = 1.

We will use the following three statistics to illustrate and evaluate the experimental results obtained with Logistic Regression:

- *C_i, the regression coefficients*, estimated via the optimization of a likelihood function. The likelihood function is built in the usual fashion, i.e., as the product of the probabilities of the single observations, which are functions of the explanatory and dependent variables (whose values are known in the observations) and the coefficients (which are the unknowns). The coefficients of the Logistic Regression equation show the extent of the impact of each explanatory variable on the estimated probability, and, therefore, the importance of each explanatory variable.
- Logistic Regression also provides an alternative way to assess the extent of this impact, via the variation in the odds ratio due to the variation of a value of an explanatory variable. The odds ratio is defined by $\frac{\pi}{1-\pi}$, and is an indicator of the likelihood of an event that is commonly used in several real-life situations. (for instance, when one provides the odds of a certain event, e.g., 7:2). It can be shown that the variation in the odds ratio due to a variation ΔX_i in the value of explanatory variable X_i is given by $e^{C_i \Delta X_i}$. Consequently, the variation due to a change of one unit in X_i is given by e^{C_i}.
- p, the statistical significance of the logistic regression coefficients, which provides an insight into the accuracy of the coefficient estimates. The level of significance of a Logistic Regression coefficient C_i provides the probability that C_i is different from zero by chance. In other words, the level of significance of C_i provides the probability that the corresponding variable X_i has an impact on □ by chance. Historically, a significance threshold (α) of α= 0.05 (i.e., 5% probability) has often been used in univariate analysis to determine whether a variable is a significant predictor. The larger the level of significance, the larger the standard deviation of the estimated coefficients, the less believable the calculated impact of the coefficient. The significance test is based on a likelihood ratio test, commonly used in the framework of Logistic Regression.
- *R^2, the goodness of fit*, not to be confused with least-square regression R^2 —they are built upon very different formulae, even though they both range between 0 and 1. The higher R^2, the higher the effect of the model's explanatory variables, the more accurate the model. However, as opposed to the R^2 of least-square regression, a high value for R^2 is rare for logistic regression. R^2 may be described as a measure of the *proportion of total uncertainty* that is attributed to the model fit.

3.2 Basic Concepts of Rough Set Theory

Rough Set theory assumes that attributes of objects (i.e., BLISS modules in our application case) are measured on a nominal or ordinal scale. For software engineering measurement, this requirement is often met quite naturally (e.g., programming language, experience level of project team, degree of communication). Otherwise, it is necessary to subdivide each attribute's domain into subclasses (nominal scale) or subintervals (ordinal scale). The specific discretization used may affect the quality of results in terms of accuracy of prediction. Four automatic discretization techniques were applied to ten real-word data sets with equal interval width, equal frequency per interval, minimal class entropy (Chmielewski and Grzymala-Busse, 1994). Accuracy of the techniques was compared but no clear preference could be found.

Knowledge obtained from Rough Set-based analysis of experimental data is represented by means of production rules, which describe the relationships between premises and conclusions. Background and algorithms to generate production rules are described in (Pawlak, 1991). The premise of a production rule is the conjunction of predicates of the form "explanatory variable = value". The conclusion of a production rule is of the form " dependent variable = value".

The objects in the data set are partitioned into subsets such all the objects of one subset have the same values for the explanatory variables. It might be the case that these subsets do not change even after a few independent variables are discarded. Some explanatory variables may shown to be redundant and thus can be removed from the set of explanatory variables without affecting the classification of the objects in the data set. On the other side, some of the explanatory variables can be shows to be core attributes which means that their elimination would decrease the accuracy of classification. For details we refer to (Pawlak 1991).

Rules generated from Rough Set analysis may be of two kinds:

- Deterministic rules: the objects in the data set that satisfy their premise belong to exactly one category of the dependent variable. Therefore, any new module to be classified that satisfies the premise of a deterministic rule is classified according to the value of the dependent variable in the conclusion of the rule.
- Non-deterministic rules: the objects in the data set that satisfy their premise belong to more than one category of the dependent variable. In this case, any new object to be classified that satisfies the premise of a non-deterministic rule is not classified.

Each rule is associated with its *absolute strength*, defined as the number of objects of the data set that satisfy its premise. To reflect different frequencies of occurrence of the different values for the dependent variable, we introduce the measure of *relative strength*. It is defined as the ratio of the number of objects (modules) that satisfy the premise of the rule to the total number of objects that have, for the dependent variable, the value of the consequence of the rule. Absolute and relative strengths provide an idea of the importance of the considered rule in explaining the behavior of the dependent variable based on the explanatory variables.

3.3 Conceptual Criteria for Comparison

We list a set of desirable requirements that data analysis approaches should satisfy and compare Logistic Regression (LR) and Rough Sets (RS) according to these assessment criteria. The list includes the criteria suggested in (Briand et al., 1992).

1. Avoidance of restrictive assumptions (probability density distribution on the independent and dependent variable ranges, independence among explanatory variables).

   LR: No strict assumption is necessary on the functional relationship between independent and dependent variables, as Logistic Regression curves are able to approximate several different functional forms. Independence of observations is necessary to apply maximum likelihood techniques. However, independence of observations is also assumed true for the vast majority of statistical techniques. Data sets of larger size lead to higher confidence in the results.

   RS: No assumption is necessary with respect to the probability distribution of variables and independence between variables. No assumption is necessary even on the size of the data set. Rough Sets analysis can be applied even to small-size data sets.

2. The modeling process needs to capture the impact of integrating all explanatory variables.

   LR: Univariate analysis is first carried out. Based on the results obtained, multivariate models are built by means of a multistage process.

   RS: All explanatory variables are taken into account at the same time. Therefore, the Rough Sets approach may be considered as a kind of multivariate analysis. Rough Sets analysis allows for detection of core and redundant variables.

3. Robustness to outliers.

   LR: This technique is less sensitive to outliers than other statistical techniques, though some kind of preliminary outlier analysis must be carried out.

   RS: Outliers are reflected as "outlier rules," i.e., rules with low strength, which do not influence other rules in any form.

4. Ability to handle interdependencies among explanatory variables.

   LR: Interdependencies are detected by means of statistical techniques that allow data analysts to check for correlations or associations between variables.

   RS: Computation of reducts (see Section 4) provides information on the core and redundant explanatory variables. Therefore, Rough Sets analysis allows the identification of a set of essential explanatory variables (core) and a set of variables that depend on those in the core.

5. Reliability of each individual prediction.

   LR: Given the values for the explanatory variables of an object, Logistic Regression provides an estimate for the probability of that object to be, say, faulty. In addition, Logistic Regression provides an estimate for the variance of this probability. Therefore, the reliability of each prediction can be assessed.

   RS: There is no reliability measure from the original approach. Quality of prediction is measured by using different techniques such as ten-fold cross validation. Each individual rule is accompanied by its absolute and relative strength.

6. Management of inconsistent pieces of information

   LR: Inconsistent pieces of information are handled in a probabilistic way. Logistic Regression provides an estimate of the probability of an object to be classified in each of the possible classes of the explanatory variable. Therefore, Logistic Regression also tells how likely it is for the value of the explanatory variable of an object to be due to random.

   RS: Inconsistencies are handled by nondeterministic rules.

7. No need for discretization

   LR: Logistic Regression may be applied to interval or ratio level data—in addition to nominal and ordinal data—so there is no need for discretization.

   RS: Rough Sets based analysis uses explanatory variables defined on a nominal or ordinal scale. Granularity and discretization techniques may influence analysis results.

8. Manipulation of different levels of granularity

   LR: Although not needed, discretization may be carried out in Logistic Regression. A reason for this may be the way the data has been collected, to make the results less sensitive to possible data collection problems, because the actual data collected may be imprecise.

   RS: The degree of accuracy of results is determined by the number of intervals. However, Rough Sets-based computations assume a low (about five) number of intervals. This often reflects degree of accuracy available from software engineering experimental data.

9. Scale invariance

   LR: The analysis can be carried out regardless of the type of scale.

   RS: There is no dependency on type of scale, after discretization has been carried out.

4 Experimental Results

We present the results we obtained with Logistic Regression (Section 4.1) and Rough Sets (Section 4.2) on the entire data set, to show what kind of results the two approaches can provide. In Section 4.3, to better assess similarities and differences, Rough Sets and Logistic Regression are applied to randomly generated subsets of the original complete data set. This will lead to the introduction of the hybrid approach in Section 5.

4.1 Logistic Regression

Here, we summarize the results we have obtained with Logistic Regression on the entire data set obtained in the DATATRIEVE™ project. We show the multivariate model we obtained, which is based on the explanatory variables that were found significant and able to explain a relevant part of uncertainty on the dependent variable in the univariate analysis. More details can be found in (Hosmer and Lemeshow, 1989). Here is the multivariate model that we identified using $p(\text{Faulty6.1}=1) = \pi(\text{AddedLOC, ModificationRate, ModuleKnowledge})$[2] for short:

$$\log(\pi/1-\pi) = -11.65 + 0.0286\ \text{AddedLOC} + 17.11\ \text{ModificationRate} \\ + 3.53\ (\text{ModuleKnowledge} - 1) - 0.06\ \text{AddedLOC*ModificationRate}$$

The term +3.53 (ModuleKnowledge - 1) shows how Logistic Regression deals with ordinal explanatory variables. The difference between the actual value (ModuleKnowledge) and a reference value (the numeric value 1, which is assumed to represent "high" knowledge of a module) is used as the actual covariate.

AddedLOC*ModificationRate, being the product of two explanatory variables, is called an interaction term. Interaction terms are used to check whether the combined effect of two variables has an impact on the dependent variable that is not captured by a purely linear model. The R^2 we obtained is 0.46, which is quite high for Logistic Regression. In column "Estimate (std dev)," Table 1 reports on the estimates for the Logistic Regression coefficient and their their standard deviation (in parentheses). Column "p" reports on the significance of the coefficients of the multivariate model. All of the explanatory variables of the multivariate model we have identified have a very high significance. For in-

[2]This formula shows the Logistic Regression equation of Section 3 in an equivalent form, to highlight the expression containing the explanatory variables. The right-hand side of the equality is the exponent of *e* in the original formula.

stance, the probability that the impact of AddedLOC on the dependent variable is due to chance is 0.0006, i.e., 0.06%.

Term	Estimate (std dev)	p
Intercept	-11.65 (2.78)	0.0000
AddedLOC	0.0286 (0.0084)	0.0006
ModificationRate	17.11 (5.72)	0.0028
ModuleKnowledge	3.53 (1.13)	0.0018
AddedLOC*ModificationRate	-0.06 (0.024)	0.0123

Table 1. Logistic Regression: multivariate model.

In our context, an observation is the detection/non detection of a fault in a module. Each detection/non detection of a fault is assumed to be an event independent from the other fault detections/nondetections. These observations were used to build the likelihood equation to estimate the coefficients of the Logistic Regression model.

4.1.1 Interpretation of the Model

All explanatory variables are sure to have an impact on the dependent variable, since their p-values are very small, i.e., there is a small probability that we have detected this impact just by chance. Therefore, they are potential risk factors and should be monitored and used when deciding whether a module should undergo further inspections and testing.

The impact is mostly in the "expected" directions, as shown by the signs of the coefficients. On the one hand, the estimated probability that there is a fault in a module increases when either AddedLOC or ModificationRate increases, or when the knowledge of a module is not good (ModuleKnowledge = 1 means good knowledge and ModuleKnowledge = 2 means bad knowledge). On the other hand, the interaction term is a correction term that mitigates the increase in the estimated probability in the multivariate model when both AddedLOC and ModificationRate increase.

Just knowing that p(Faulty6.1=1) basically increases when AddedLOC or ModificationRate increases, or when ModuleKnowledge gets worse is not enough. We also need to know which of the explanatory variable has the greatest impact on the dependent variable, i.e., which explanatory variable is the most important one to keep monitored and controlled. The numerical values of the coefficients of the Logistic Regression equation show the extent to which each explanatory variable affects the estimated probability. The variation in the odds ratio may also be used to this end. Some caution must be used, however, before drawing the conclusion that the explanatory variables with the biggest coefficients in absolute value are also the ones with the biggest impact. In our case, this would for instance mean that ModificationRate (whose coefficient is

17.11) is more important than AddedLOC (whose coefficient is 0.0286). Variable ModificationRate can only range between 0 and 1, and is rarely close to 1. Variable AddedLOC has 0 as its lower bound, and no upper bound. One should bear in mind that the variation in the odds ratio depends on both the coefficients *and* the variations of the explanatory variables. For instance, if a typical value of AddedLOC is 100 lines of code and a typical value in ModificationRate is 0.1 (which are variations with respect to the initial situation of the software product), the impact of AddedLOC is greater than the impact of ModificationRate, since the variation on the odds ratio due to AddedLOC would be $e^{2.86}$, while the variation in the odds ratio due to ModificationRate would be $e^{1.711}$. In our case, since the average value of AddedLOC is 117.95 (with standard deviation 118.63), that of ModificationRate is 0.2437 (with standard deviation 0.1280), that of the interaction term AddedLOC*ModificationRate is 33.52 (with standard deviation 45.87), and the only permissible variation in ModuleKnowledge is 1, we can conclude that the four terms of the Logistic Regression equation can be ranked as follows:
(1) Module-Knowledge, (2) ModificationRate, (3) AddedLOC, (4) AddedLOC*ModificationRate.

4.1.2 Classification Results

The classification results obtained with Logistic Regression are summarized in Table 2. We have assumed a threshold of $p = 0.085$ to predict a module as faulty, which is the actual proportion of non-faulty modules found in our sample, i.e., a module was predicted to be faulty only if the estimated probability for it to be faulty exceeded 0.085. Therefore, we classified as faulty those modules for which

$$\log(\pi/1-\pi) > \log(0.085/0.915)$$

i.e.,

$$-11.65 + 0.0286\ \text{AddedLOC} + 17.11\ \text{ModificationRate} + 3.53\ (\text{ModuleKnowledge} - 1) - 0.06\ \text{AddedLOC*ModificationRate} > \log(0.085/0.915)$$

However, the choice of a threshold is to some extent a subjective decision, and other thresholds may be used.

LR classification	Predicted non-faulty 6.1	Predicted faulty 6.1	Total
Actual non-faulty 6.1	63.1%	28.4%	91.5%
Actual faulty 6.1	0.8%	7.7%	8.5%
Total	63.9%	36.1%	100%

Table 2. Logistic Regression: classification results.

4.2 Rough Sets: Results

Experimental data were analyzed by the Rough Sets based data analysis system PROFIT, developed at Institute of Computing Science of Technical University Poznan (Slowinski and Susmaga, 1996). Cluster analysis techniques were used to obtain discrete variables from the experimental data in an automated way.

Fifteen deterministic rules were generated. Among them, three rules were non-deterministic. In Table 3, we report the ones with the greatest relative strength. In more detail, we show the three (deterministic) rules with the greatest relative strength for non-faulty modules (Faulty6.1 = 0) and the two rules with the greatest relative strength for faulty modules (Faulty6.1 = 1).

Premise	Conclusion	Relative Strength
(DifferentBlocks = low) & (ModuleKnowledge = high)	Faulty6.1 = 0	56%
(LOC6.0 =medium) & (AddedLOC = low) & (ReusedLOC = low)	Faulty6.1 = 0	43%
(DifferentBlocks = low) & (ReusedLOC = very high)	Faulty6.1 = 0	37%
(DifferentBlocks = medium) & (ModuleKnowledge = high)	Faulty6.1 = 1	27%
(AddedLOC = medium) & (ReusedLOC = high)	Faulty6.1 = 1	18%

Table 3. Rules with greatest relative strength generated by the Rough Sets data analysis.

Deterministic and non-deterministic rules aggregating knowledge gained from the experiments are an essential input for feedback sessions (van Latum et al., 1998). *Feedback sessions* are organized meetings integrating members of the project and the GQM experts. They are an essential device for analysis and interpretation of measurement data. There are four principal cases which my occur with respect to hypotheses formulated at the beginning of the experiment:

- Confirmation of hypothesis
- Update of hypothesis
- Rejection of hypothesis
- Formulation of new hypothesis.

In each case, additional insight is gained from the empirical investigation including Rough Set analysis of data. Either predictions for fault-pronenenss of modules become more reliable, or new explanations are found for modules to be critical.

4.2.1 Interpretation of the Rule

The first rule above is the most powerful one. It states that number of faults in version 6.1 is expected to be zero for those modules where few blocks have been modified from version 6.0 to version 6.1 and the knowledge of the module is high. The premise of the rule is satisfied by 56% of modules for which Faulty6.1 = 0. This is a deterministic rule, so all the modules in the data set for which (DifferentBlocks = low) & (ModuleKnowledge = high) have Faulty6.1 = 0. We also report two other rules for modules with Faulty6.1 = 0, which are self-explanatory. The other two rules refer to those modules for which Faulty6.1 = 1. The stronger of them says that when the number of blocks that are different between the two version is medium and the knowledge of the module is high, then we have a faulty module. The premise of this rule is satisfied by 27% of the modules with Faulty6.1 = 1.

These rules only use the elementary measures on which all the explanatory variables are based: LOC6.0, AddedLOC, DeletedLOC, ReusedLOC, DifferentBlocks, ModuleKnowledge. The core attributes (as defined above) are: LOC6.0, ReusedLOC, DifferentBlocks, and ModuleKnowledge.

4.2.2 Classification Results

Leaving-one-out test was performed to validate the quality of the prediction results. This test consists in repeating N times the iteration of classifying objects, where N is the number of objects in the input data. In each iteration a different object (module) is chosen to be the testing (one-element) sample, while all the other objects constitute the learning sample from which the rules are generated. Accuracy of classification can then be assessed, since the actual classification of the object is known. The results are summarized in Table 4.

RS classification	Not Classified	Predicted non-faulty 6.1	Predicted faulty 6.1	Total
Actual non-faulty 6.1	0%	89.9%	1.6%	91.5%
Actual faulty 6.1	0.8%	3.85%	3.85%	8.5%
Total	0.8%	93.75%	5.45%	100%

Table 4. Rough Set classification results.

The percentages in the "Not Classified" column correspond to the modules that satisfy the premise of some non-deterministic rule and cannot be classified by Rough Sets.

4.3 Comparison of Results on Partitions

In addition to the results for either technique outlined in Sections 4.1 and 4.2, we have also carried out further analysis, by partitioning the original data set into smaller sets, to allow for better and more detailed comparison of Rough Sets and Logistic Regression depending on the data set size. Table 5 describes the characteristics of the 9 data sets we used (data set #1 is the entire data set). For instance, data set #4 contained 50% of the modules of the original data set; 9.2% of the modules in data set #4 were faulty.

#	1	2	3	4	5	6	7	8	9
% modules	100%	50%	50%	50%	50%	24.6%	24.6%	25.4%	25.4%
%faulty modules	8.5%	16.9%	16.9%	9.2%	7.7%	15.7%	15.7%	18.2%	18.2%

Table 5: Characteristics of the data sets.

The 9 data sets were assigned to four classes, based on the similarities in the proportion modules from the original data set contained and the proportion of faulty modules they contained:

Class I	=	{1}
Class II	=	{2,3}
Class III	=	{4,5}
Class IV	=	{6,7,8,9}

The quality of prediction was evaluated by the following three parameters:

1. *Overall completeness*: Proportion of modules that have been classified correctly. This parameter provides information on how well a technique classifies modules, regardless of the category in which the modules have been classified.
2. *Faulty module completeness*: Proportion of faulty modules that have been classified as faulty. This parameter provides information on how many of the faulty modules have been correctly identified by the data analysis technique. Conversely, this parameter provides information on the risk of not having identified faulty modules, and, therefore, not having tested or inspected them more carefully.
3. *Faulty module correctness*: Proportion of modules that have been classified as faulty and were faulty indeed. This parameter provides information on the efficiency of a technique, i.e., on the proportion of modules that are potential candidates for further verification. Conversely, this parameter provides information on the proportion of modules that are not faulty and have undergone further verification anyway.

The idea underlying the use of *faulty module completeness* and *faulty module correctness* is that faulty modules may be considered more relevant than non-faulty ones in this application context and in software engineering in general. Faulty modules may cause serious failures, i.e., if faults are not removed. Therefore, faulty modules should undergo additional verification activities, besides those carried out on all modules.

For each of the four classes, we computed the average values of the three classification indices above. In what follows, LR(i) and RS(i) denote the analysis results of the i-th class from application of Logistic Regression and Rough Sets, respectively. The results are summarized in Table 6. In classes I, III, and IV, the Rough Set approach provides non-deterministic rules in addition to the deterministic ones. As a matter of fact, all of the non-faulty modules were classified and only a few of the faulty ones were not classified. In Table 6, in addition to the three evaluation parameters, we provide the percentages of the faulty modules that were not classified by the Rough Sets. We also show the difference in results related to the three evaluation parameters due to the unclassified modules. The first percentage in each entry corresponds to classifying the unclassified modules as non-faulty; the second one (i.e., the one in parentheses) corresponds to classifying the unclassified modules as faulty. For instance, average overall completeness of class III is 93.85% if the unclassified modules are considered as non-faulty and 95.4% if the unclassified modules are considered as faulty. At any rate, due to the small number of faulty modules, it is very likely that a software manager will consider the unclassified modules faulty, i.e., he or she will take into account the results in parentheses as the useful ones. In each column RS(*) we present the classification results obtained with the leaving-one-out mechanism.

Parameter	LR (I)	RS (I)	LR (II)	RS (II)	LR (III)	RS (III)	LR (IV)	RS (IV)
Non-faulty modules	91.5%		83.1%		91.5%		83.1%	
Faulty modules	8.5%		16.9%		8.5%		16.9%	
Overall completeness	70.8%	93.85% (94.6%)	77.7%	91.5 %	66.2%	93.85% (95.4%)	84.1%	92.1% (93.0%)
Faulty module completeness	94.1%	45.4% (54.5%)	91.1%	77.2 %	83.4%	46.2% (63.3%)	90.9%	66.7% (71.7%)
Faulty module correctness	21.3%	71.4% (75.0%)	42.6%	76.6 %	17.4%	70.8% (77.5%)	58.0%	80.4% (81.7%)
Not classified faulty modules	0%	0.8%	0%	0%	0%	1.4%	0%	0.8%

Table 6. Comparison of classification results for the five classes of examples.

Based on the data analysis, it is possible to draw the following conclusions:

- Rough Sets based analysis performs better for all five classes with respect to overall completeness and faulty module correctness.
- Logistic Regression based analysis performs better for all five classes with respect to faulty module completeness.
- There is no obvious tendency for the dependency between the varied size of the data set and the quality of classification results.

5 Hybrid Approach

Overall, the two techniques look complementary, in that Logistic Regression performs better with respect to faulty module completeness, while Rough Sets analysis performs better with respect to overall completeness and faulty module correctness. The question is: Is it possible to combine the classification strengths of both approaches into a hybrid approach? In what follows, we develop a hybrid approach which answers the above question in a positive way.

The hybrid approach H is described in five steps:

Step1. Let *M* be the set of all modules. The modules of set *M* are classified by Logistic Regression into two disjoint sets: *LRF*, the set of modules classified as faulty, and *LRNF*, the set of modules classified bas non-faulty.

Step2. We apply Rough Set analysis on the entire set *M*. A set of classification rules is obtained.

Step3. Logistic Regression classification with respect to non-faulty modules is considered to be final, since the evaluation results show that the vast majority of modules that belong to *LRNF* are non-faulty indeed.

Step4. On the other hand, *LRF* is likely to include most faulty modules (see the high values for faulty module completeness in Table 6), but also several non-faulty ones (see the low values for faulty module correctness in Table 6). Since Rough Set analysis is likely to eliminate most non-faulty modules, the Rough Set classification rules obtained at Step 2 (based on the entire set *M*) are applied to *LRF*. Modules in *LRF* are classified into two sets: *RSLRF*, the set of modules classified as faulty, and *RSLRNF*, the set of modules classified as non-faulty.

Step5. The final classification of fault modules (*HF*) and non-faulty modules (*HNF*) by means of the hybrid approach H is
HF := RSLRF
HNF := LRNF $\cup$ RSLRNF

For validation purposes, we suggest two heuristics to classify those modules that match a nondeterministic rule when the Rough Sets rules are applied in Step 4:

- Modules matching a non-deterministic rule are classified according to the conclusion which is supported by the maximum number of modules related to *M* (and thus is more likely to occur). The related approach is denoted by **LR*RS.**
- Modules matching a non-deterministic rule are classified as faulty, to be on the "safe side." The related approach is denoted by **LR&RS**.

5.1 Interpretation of the Models and Rules

Classification is carried out by combining the model obtained with Logistic Regression and the rules derived with Rough Sets. The Hybrid approach can be used in a straightforward manner. For instance, suppose that a new module needs to be classified.

If Logistic regression classifies it as non-faulty, since

$$-11.65 + 0.0286\ \text{AddedLOC} + 17.11\ \text{ModificationRate} + 3.53\ (\text{ModuleKnowledge} - 1) \\ -0.06\ \text{AddedLOC}*\text{ModificationRate} \leq \log(0.085/0.915)$$

then the module is classified as non-faulty. Otherwise, the Rough Sets rules (some of which are shown in Section 4.2.1, Table 3) are used to classify the module as either non-faulty or faulty.

5.2 Classification Results

We compare the quality of classification obtained from application of Logistic Regression, Rough Sets analysis, and the hybrid approach with both heuristics. The evaluation is done with respect to the three parameters described above and was applied to all four classes defined in Section 5.4. Corresponding results are given in Tables 7, 8, 9, and 10, respectively.

Parameter	LR(I)	RS(I)	LR*RS(I)	LR&RS(I)
Overall completeness	70.8%	93.85% (94.6%)	96.9%	96.1%
Faulty module completeness	94.1%	45.4% (54.5%)	81.2%	90.9 %
Faulty module correctness	21.3%	71.4% (75.0%)	81.2%	71.4%

Table 7. Comparison of classification results for class I.

Parameter	LR(II)	RS(II)	LR*RS(II)	LR&RS(II)
Overall completeness	77.7%	91.5%	97.7%	97.7%
Faulty module completeness	91.1%	77.2%	86.4%	86.4 %
Faulty module correctness	42.6%	76.6%	100%	100%

Table 8. Comparison of classification results for class II.

Parameter	LR(III)	RS(III)	LR*RS(III)	LR&RS(III)
Overall completeness	66.2%	93.85% (95.4%)	98.5%	98.5%
Faulty module completeness	83.4%	46.2% (63.3%)	81.8%	81.8 %
Faulty module correctness	17.4%	70.8% (77.5%)	100%	100%

Table 9. Comparison of classification results for class III.

Parameter	LR(IV)	RS(IV)	LR*RS(IV)	LR&RS(IV)
Overall completeness	84.1%	92.1% (93.0%)	99.2%	99.2%
Faulty module completeness	90.9%	66.7% (71.7%)	95.5%	95.5%
Faulty module correctness	58.0%	80.4% (81.7%)	100%	100%

Table 10. Comparison of classification results for class IV.

We observe that the hybrid approach performs better than both LR and RS. There are only small differences between the application of the two heuristics LR*RS and LR&RS. In addition to the 'numerical' superiority, the hybrid approach also integrates other advantages of both basic approaches:

1. Logistic Regression results in a ranking of the most important influence factors among all the attributes considered. This is of value as a guidance for improvement. These results were independently confirmed by Rough Set analysis thus increasing confidence.

2. Rough Sets analysis brings out operational rules by which the modules can be classified. This is of importance because it can be immediately used for decision support.

3. Analysis of Logistic Regression is based on the measures of the GQM plan, which may be in fact derived from more elementary measures. Rough Sets analysis instead used the elementary metrics to which all the measures of the GQM plan could be reduced.

4. The Rough Sets approach offers the strength of the derived rules but does not provide reliable indications on the significance of prediction. The Logistic Regression approach is able to evaluate the confidence with which results have to be considered in the case of larger data sets and in the case of independent attributes.

6 Summary and Conclusions

We have studied and compared Logistic Regression and the Rough Sets approach, two techniques for the analysis and knowledge discovery of software measurement data. Our study shows that both techniques are able to identify essential knowledge to predict module faultiness in maintenance. The results of prediction become even better for the suggested hybrid algorithm exploiting the strengths of both approaches. The main conclusions from the investigations are:

1. The two approaches perform differently with respect to the three chosen criteria of accuracy (overall completeness, faulty module correctness, faulty module completeness). In all five classes of examples, Rough Set based analysis performed better with respect to overall completeness and faulty module correctness, while Logistic Regression was better with respect to faulty module completeness.

2. We suggest a hybrid approach, in which both techniques are applying subsequently. The performance of this approach is demonstrated to be better when compared to separate application of either Logistic Regression or Rough Sets.

3. The knowledge gained from either approach increases confidence of results and is - to some extent - complementary from its nature. It contains independent detection of main influence factors on reliability of modules on the one side and formulation of production rules reflecting the experience from performed experiments on the other side. Both contributions are essential for carrying out similar projects in the future and are part of the knowledge of an organization wide experience factory.

4. Knowledge engineering approaches in software engineering should take into account maturity of the organization; scope and objective of the underlying measurement program; completeness, accuracy, and precision of the collected data. From its methodological background, Rough Sets are more devoted to quantitative analysis based on nominal or ordinal scale while Logistic Regression is more quantitative from its origins.

5. The analysis of the data from a real-world case study offered valuable insights for both the software development process and analysis techniques.

However, more studies of this kind are necessary to further increase confidence of results with respect to both aspects and related to other analysis techniques.

6. The suggested hybrid approach can be applied whenever we have a Boolean dependent variable. Its extension to more general types of variables is an open research topic.

Acknowledgements

We wish to thank the DATATRIEVE™ team for their support and the participants of the CEMP consortium for the useful discussions and suggestions during the project. We also would like to thank Audris Mockus for his comments on an earlier version of this paper.

References

Basili V.R., Caldiera G., and Rombach H.D., Experience Factory, 1994. In Marciniak, J.J., editor, Encyclopedia of Software Engineering, volume 1, 469-476. John Wiley & Sons.

Basili, V.R. and Rombach, H.D., 1988. The TAME Project: Towards Improvement-oriented Software Environments. IEEE Transactions on Software Engineering, SE-14(6), 758-773.

Basili, V.R. and Weiss, D.M., 1984. A Methodology for Collecting valid Software Engineering Data, IEEE Transactions on Software Engineering, SE-10(6), 728-738.

Briand L.C., Morasca S., and Basili V.R., 1994. Defining and Validating High-Level Design Metrics." *Technical Report*, CS-TR-3301, University of Maryland.

Briand L.C., Basili V.R., and Thomas W.M., 1992. A Pattern Recognition Approach for Software Engineering Data Analysis. IEEE Transactions on Software Engineering, 18(11), 931-942.

CEMP (Customized Establishment of Measurement Programs) ESSI Project 10358, Final Report, Kaiserslautern, 1996, p. 109, IESE-Report, 001.96/E

(CEMPwww) http:/www.iese.fhg.de/Services/Projects/Public-Projects/Cemp.html

Chmielewski, M.R.and Grzymala-Busse, J., 1994. Global Discretization of Continuous Attributes as Preprocessing for Machine Learning. Proceedings of Third International Workshop on Rough Sets and Soft Computing, 294-301.

Hosmer, D.W. Jr. and Lemeshow, S., 1989. Applied Logistic Regression. John Wiley & Sons, Inc..

Morasca, S and Ruhe, G, 1997. Knowledge Discovery from Software Engineering Measurement Data: A Comparative Study of Two Analysis Techniques. Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering (SEKE'97), Madrid, Spain, 450-458.

van Latum, F.,van Solingen, R., Hoisl, B, Oivo, M.Rombach, H.D., and Ruhe, G., 1998. Adopting GQM-Based Measurement in an Industrial Environment. IEEE Software, 78-86.

Pawlak, Z., 1991. Rough Sets: Theoretical Aspects of Reasoning about Data, Kluwer Academic, Dordrecht.

Ruhe, G., 1996. Qualitative Analysis of Software Engineering Data Using Rough Sets, Proceedings of the Fourth International Workshop on Rough Sets, Fuzzy Sets, and Machine Discovery, Tokyo, 292-299.

Susmaga R, and Slowinski, R., 1996. Rough Processing of Fuzzy Information Tables (ProFIT). Institute of Computing Science. Poznan University of Technology.

Document Information

Title:	A Hybrid Approach to Analyze Empirical Software Engineering Data and its Application Predict Module Fault-proneness in Maintenance
Date:	May 10, 1999
Report:	IESE-026.99/E
Status:	Final
Distribution:	Public