UMIACS-TR-91-69
CS-TR-2672

May 1991

A Pattern Recognition Approach for Software Engineering Data Analysis*

L.C. Briand, V.R. Basili and W.M. Thomas

Institute for Advanced Computer Studies and
Department of Computer Science
University of Maryland
College Park, MD 20742

Abstract

In order to understand, evaluate, predict, and control the software development process with regard to such perspectives as productivity, quality, and reusability, one needs to collect meaningful data and analyze them in an effective way. However, software engineering data have several inherent problems associated with them and the classical statistical analysis techniques do not address these problems very well. In this paper, we define a specific pattern recognition approach for analyzing software engineering data, called Optimized Set Reduction (OSR), that overcomes many of the problems associated with statistical techniques. OSR provides mechanisms for building models for prediction that provide accuracy estimates, risk management evaluation and quality assesssment. The construction of the models can be automated and evolve with new data over time to provide an evolutionary learning approach (the Improvement Paradigm) to software modeling and measurement. Experimental results are provided to demonstrate the effectiveness of the approach for the particular application of cost estimation modeling.

*Research for this study was supported in part by NASA grant NSG 5123, Coopers & Lybrand (IAP member) and Vitro Corporation (IAP member).

1 Introduction

Managing a large scale software development requires the use of quantitative models to provide insight and support control based upon historical data from similar projects. This implies the need for a quantitative approach.

- to build models of the software process, product, and other forms of experience (e.g., effort schedule, reliability, ...) based upon common characteristics for the purpose of prediction.
- to recognize and quantify the influential factors (e.g., personnel capability, storage constraints, ...) on various issues of interest (e.g., productivity improvement, effort estimation, ...) for the purpose of understanding and monitoringthe development.
- to evaluate software products and processes from different perspectives (e.g., productivity, fault rate) by comparing them with projects with similar characteristics.
- to identify strengths and weaknesses in the current environment.
- to understand what we can and cannot predict and control so we can monitor it more carefully.

In Section 2 of this paper, we discuss the needs and the constraints in building effective models for the software development environment. In Section 3, we review the failings of the classical statistical approaches with regard to software engineering data. We offer a new approach for analyzing software engineering data in Section 4, called Optimized Set Reduction (OSR), that overcomes many of the problems associated with statistical techniques. The approach is based upon pattern recognition techniques tailored to the software engineering field and offers advantages that overcome many of the problems associated with statistical techniques.

Besides overcoming some of the drawbacks of statistical analysis for software engineering model building, OSR provides mechanisms for building models for prediction that provide estimates of the accuracy of the prediction, models for risk management evaluation for the risk areas of interest, and quality assessment relative to the pertinent quality models.

The construction of the models can be automated and evolve with new data over time to provide an evolutionary learning approach to software modeling and measurement in order to support software development management.

In Section 5, experimental results are provided to demonstrate the effectiveness of the approach for the particular application of cost estimation modeling. The data set used are the COCOMO [BOE81] and Kemerer [KEM87] data sets. Although the particular example of cost estimation is chosen, the approach is also being used for other forms of model building, e.g., the building of models for maintenance.

Section 6 discusses some general issues related to multivariate data analysis in the context of OSR. A paradigm that defines the learning aspects of software development and management (called the Improvement Paradigm) is described in Section 7 and it is shown how OSR performs with the learning and model refinement issues in the IP framework.

2 Requirements for an Effective Data Analysis Procedure

Based upon the constraints associated with the data and the analysis procedures, we generate a set of requirements for model building approaches. In the text that follows, we will refer the variable to be assessed as the "Dependent Variable" (DV)(e.g. productivity, fault rate) and the variable explaining the phenomenon as "Independent Variables" (IV) (e.g. personnel skills, data base size) [BOE81]. We will see later on that all the issues related to prediction, evaluation and risk management may be formalized under this form. The various IVs will form the dimensions of an Euclidian space called the "sample space" in the following text.

2.1 Constraints related to Software Engineering Data

Model building in Software Engineering is faced with the following difficulties:

- C_1: There is no theory proven to be effective in any environment that would give a formal relationship among measured metrics in the development process. Therefore the capabilities of classical statistical approaches seem very limited and statistical simulation (i.e. Monte Carlo approach) appear improbable.
- C_2: The best we can do is make assumptions about the probability density distributions, with respect to the dependent and independent Variables of interest, with very little evidence to support our assumptions.
- C_3: The sample size is usually small relative to the requirements of the classical statistical techniques, the quality of the data collected, and the number of significant independent variables (IV). This is due to the nature of the studied objects in software engineering and it is difficult to avoid (e.g. software system, module, change, defect ...).
- C_4: "Software engineering modelers" have to deal with missing, interdependent and non-relevant independent variables: This is due to a lack of understanding of the software development process.
- C_5: Both data defined on a continuous (i.e. ratio, interval) and a discrete (i.e. nominal, ordinal) range have to be handled. Collecting data in a production environment is a difficult task and discrete data collection is sometimes performed to facilitate the measurement process. Also, the nature of some of the data may be discrete.

2.2 Requirements to alleviate these constraints

Matching the constraints, we can define requirements for effective data analysis procedures by the following list:

- R_1 [matches C_1, C_2]: The data analysis procedure should avoid assumptions about the relationships between the Variables regarding the probability density distribution on the IV and DV ranges.

- R_2 [C_3, C_4]: A mechanism is needed to evaluate accuracy for each performed estimation. The variations of accuracy lie in a large range depending on the object to be assessed. For example, you may want to assess a software project from the point of view of productivity.

  C_3 The amount of available relevant data may differ according to the characteristics of the project to be assessed (i.e. location of the project in the sample space). For example, you may have more data with respect to data processing business applications than with respect to real time systems. In small samples, this phenomenon can have significant consequences.

  C_4 The performed data collection may be more suitable to certain kinds of objects than others. For example, measuring objectively time constraints for real time systems may be difficult and therefore may introduce uncertainty in assessement.

- R_3 [C_4]: The data analysis procedure must be as robust as possible to missing, non-relevant, interdependent IVs and outliers. Then, some procedures must be available in order to detect and alleviate the effects related to these kind of disturbances in the data set.

- R_4 [C_5]: The data analysis procedure must be able to handle easily both discrete and continuous metrics without biasing the results obtained.

3 Statistical Background in the Field

Most of the studies in software engineering have been based on multiple regression and related techniques (e.g. F tests, stepwise selection). These procedures do not seem to match most of the previously defined requirements for the following reasons:

- The adjusted coefficient of correlation is a parameter globally calculated over all the data points. It is the ratio of the variance explained by the calculated correlation versus the variance still unexplained. This does not provide accuracy of individual estimations (see requirement R_2). Calculating confidence intervals does not seem possible with an acceptable accuracy. Their calculations are based on an approximation of the distribution (i.e. normal) and variance of the residuals in the studied population. Too few data are usually available to make a reasonable estimation of the variance. Moreover, we should not expect a constant variance all over the regression space. The disturbance due to some missing independent variables in the regression equation may have a variable intensity in different parts of the regression space. Considering that in the field of Software Engineering we already make approximations on the mathematical shape of the regression function, with weak theories to support our assumptions, this variability may be significant.

- This technique is very sensitive to perturbations related to **missing, interdependent** and **non-relevant** metrics (i.e. independent variables) [DIL84]. Procedures to deal with these problems are complex and errorprone (e.g stepwise selection, F partial

tests, detection of outliers). These problems make difficult any quantitative analysis of the development environment and affect the accuracy of predictions.

- **Continuous** metrics are at times difficult to compare because the calculated regression coefficients are dependent on their corresponding measurement units. The classical way to solve this problem is to work with Beta coefficients (i.e. standardized regression coefficients). The dependent variable and the independent variables are divided by their standard deviation before calculating the regression equation. In other words, IVs and DVs are normalized to their variability. This way, the used regression parameters become unitless numbers and are independent of the magnitude of units. However, the magnitude of the Beta coefficients are dependent on the variability of the independent variables in the particular sample that provide the data. This is another cause of innacuracy in the performed estimations.
- **Discrete** metrics are very difficult to handle because the notion of distance between categories would be required to build Multiple Regression based models. Then, some strong assumptions are necessary to integrate these data in a regression calculation (i.e. assumptions about the relative distances between the defined categories). The usual way to handle categorical data is to create a regression parameter for each category of each categorical metric (i.e. dummy variable). For example, The value of each parameter is either 1 or 0 according to the categories to which the object to be assessed belongs. Two major drawbacks may be identified:
  - The number of regression parameters increases rapidly with the number of categorical metrics.
  - The discrete values (e.g. $0, 1$)assigned to these parameters are arbitrary and then make the calculated regression coefficient meaningless for analyzing the respective influence of the various regression parameters. No beta-coefficient can be calculated.

There have been some significant attempts to develop alternative techniques to regression based procedures to analyze software engineering data. The use of statistical classification techniques has been used to make predictions with respect to errorprone modules in large scale software systems [SEL88]. These experiments showed encouraging and interesting results. However, if we consider the usual data availability in Software Engineering, the used data set was very large (5000 data points). The objective was simply to classify objects among two categories (i.e. non errorprone and errorprone modules) and not to come up with a comprehensive data analysis procedure. These experiments opened a door to a huge research field to be investigated.

4 A Pattern Recognition Approach for Analyzing Data

Based upon pattern recognition principles [TOU74], we propose a data analysis procedure that is intended to fulfill, to a certain extent, the previously described requirements for effective data analysis. This procedure and the main principles supporting are described in this section.

4.1 Description of the Main Underlying Technique

The technique has as its goal the recognition of patterns in a data set. These patterns are used as a basis for understanding and assessing the development process, product and environment.

4.1.1 The Basic Concepts and Terminology

- A **learning sample** consists of N vectors $(DV_i, IV_{1,i}, ..., IV_{n,i})$, $i \in (1, ..., N)$, containing one dependent and n independent variables. These vectors are defined in an Euclidian space called the "sample space". These vectors, which we will call **pattern vectors**, represent measurements taken in the environment.

- A **measurement vector** is defined as the set of independent variable values representing a particular object whose dependent variable value is to be predicted. That is, it is a pattern vector without the dependent variable.

- To be able to make predictions on the **dependent variable**, its range has to be sub-divided or grouped into what we will call **DV classes**. These classes correspond to natural situations that can be encountered in the measurement environment, with respect to the dependent variable, e.g. productivity. If the Dependent Variable is either "ratio", "interval" or "ordinal", the dependent variable range is sub-divided into intervals, if the Dependent Variable is "nominal", categories may be grouped into a smaller set of classes. They are called "states of nature" in decision theory and "pattern classes" in the pattern recognition field [TOU74]. We have chosen the name DV classes in order to make the connection with a classical statistical approach for multivariate analysis.

- To be able to use the **independent variables** as a basis for predicting the dependent variable, they, like the Dependent variables, must be mapped into **IV classes** by sub-dividing or grouping.

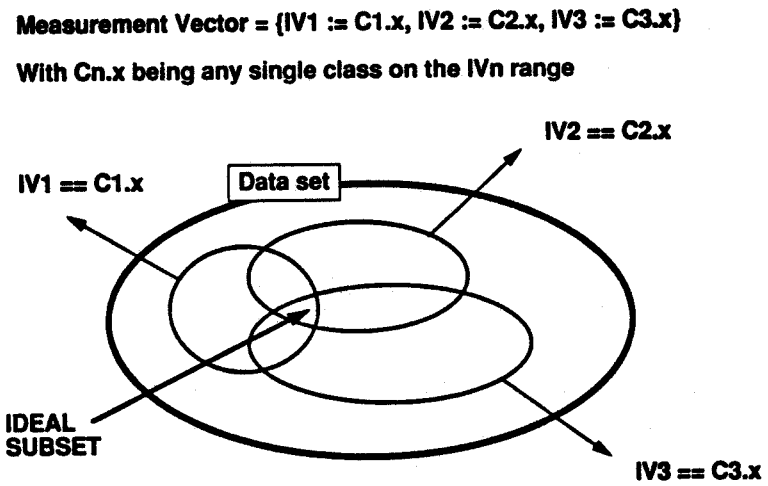
- A **pattern** is defined as a non-uniform distribution of probabilities across the DV classes.The further a distribution is from uniformity, the more the pattern is considered as significant (measurable metric developed below).

4.1.2 A Particular Pattern Recognition Process

The problem of predicting the dependent variable for a particular project can be stated as follows: Given a particular measurement vector (MV), determine the probability that the actual dependent variable value lies in each of the DV classes. The shape of the probability density function on the DV class range associated with MV is unknown. The goal and the basic principle of this process is to find a subset of pattern vectors in the data set, whose values for the independent variable are similar to the values for the independent variable of MV and show a significant pattern among the DV classes.

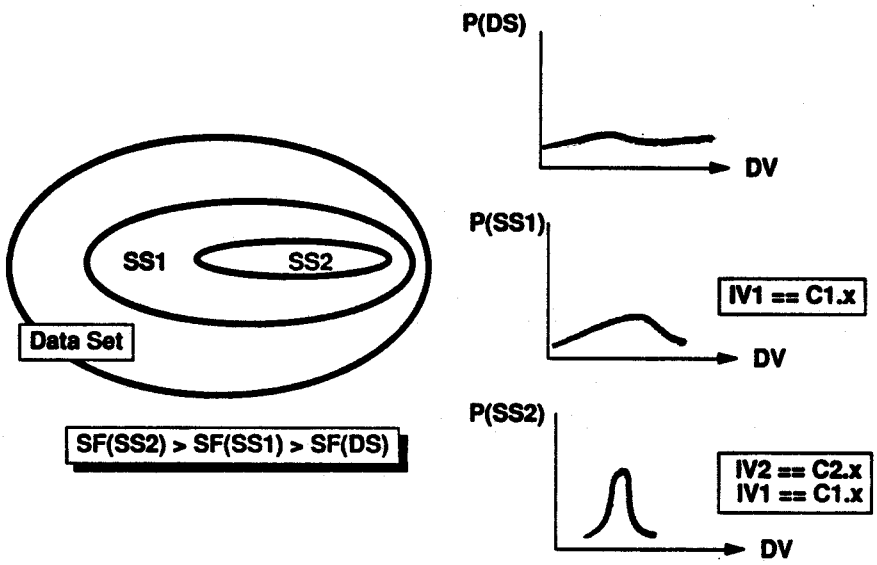
Taking this approach in the ideal, we can assume that given a learning sample, LS, and a measurement vector, MV, we could select an ideal subset of all the pattern vectors in LS having the exact same IV instances as MV (Figure 1). However, since we are usually working with small samples and numerous independent variables the ideal subset is typically too small to be useful, so this ideal approach is not applicable.

Figure 1 - Ideal approach



Nevertheless, we need to find a subset, SS, of LS that contains pattern vectors similar to MV with respect to some IVs and that yields significant patterns. SS must be large enough to be usable (notion defined below) and must contain sufficiently homogeneous pattern vectors to yield significant patterns. To extract SS from LS, we have to select a subset of IVs among those available, that will be used to select the pattern vectors that will form SS. The pattern vectors matching the MV instances with respect to the selected IVs will be extracted from LS. This IV selection will be performed in a stepwise manner and based upon a **selection function SF** (Figure 2). In other words, each pattern resulting from a potential IV selection will be evaluated using SF which provides information on the degree of significance of the pattern.

Figure 2 - Pattern recognition process



Two reasons justify a stepwise process:

- We wish to stop the subsetting whenever the resulting subset SS is too small.
- If we increase the pattern significance progressively, set reduction after set reduction, we increase the probability of not getting a good pattern purely by chance due to the small size of the sample on which the pattern is based. A constant trend is more convincing than a single pattern based on a small sample.

In the Software Engineering Laboratory at the University of Maryland, a set of experiments have led us to develop the following pattern recognition process (called Optimized Set Reduction) applied for any MV:

- Step 1: DV classes are formed either by dividing the DV range into intervals or by grouping the defined DV categories. For optimal results, a similar number of pattern vectors should be in each class. The mechanism for creating classes is further described below.
- Step 2: IV classes are formed in a similar way.
- Step 3: The learning sample is successively decomposed into subsets. At each step, an IV is selected (according to a **selection function** described below) and the objects having the same instance for the IV as the object to be assessed are extracted to

form the reduced subset. This is done recursively on the reduced subsets. We call the performed process an **Optimized Set Reduction.**

- Step 4: When a predefined condition is reached, the reduction stops. This condition will be referred to as the **termination criteria** and will be discussed below. The subsets resulting from this criteria are called the **terminal subsets.**

- Step 5: The pattern vectors in the terminal subset(s) are then used to calculate **probability** that the actual dependent variable value lies in each of the DV classes. Several alternatives may be considered for calculating these probabilities and two of them are described below.

The resulting probabilities (that form the obtained pattern) may be used either for **DV predictions, risk management** or **quality evaluation** in a way that will be described in Section 4.3.

Despite an apparent simplicity, this approach opens up a set of research questions associated with each of the steps, that need to be further investigated. The details of each of the steps, as well as the open questions are discussed here:

- Which IV **selection function** should be used as a reduction mechanism (Step 3)?

  The best we have found so far is Entropy (F). The measure of entropy generalized for m classes from information theory can be used as the impurity evaluation function [BRE84, SEL88]:

$$F = \sum_{i=1}^{m} -P(C_i/x) \log_m P(C_i/x)$$

  where $P(C_i/x)$ is the conditional probability of x of belonging to

  the DV class C_i, $i \in (1, ..., m)$.

  We assume that the lower the entropy the more likely we are to have found a significant pattern. This assumption has been supported by the performed experiments: a good correlation between Entropy and MRE (i.e. Magnitude of Relative Error) has been observed. The selected IV is the one, that minimizes the defined selection function.

- How is the **termination criteria** determined (Step 4)?

  The termination criteria needs to be tuned to the environment, i.e. the available data set. Logically, if measuring the significance of a pattern by calculating its entropy is reasonable, then the entropy should be strongly correlated to the observed prediction accuracy (i.e. Magnitude of Relative Error for continuous DVs and Misclassification Rate for discrete DVs). Therefore, an estimation of the prediction MRE/MR is possible by looking at the decomposition entropy. There are two bounds on the calculation. If there were no termination criteria, the reduction could decompose to a subset of a single pattern vector, yielding the meaningless minimum entropy of zero. On the other hand, if we stop the reduction too soon, we have not sufficiently decomposed the data set to provide an the most accurate

characterization of the object to be assessed. Thus we are interested in achieving an accurate approximation of the selection function based upon some minimum number of pattern vectors in the terminal subsets. To find this minimum number, we must experiment with the learning sample by examining the correlation between the MRE/MR and the selection function (e.g., entropy). If this correlation becomes too weak, then the acceptable minimal number of pattern vectors in a subset should be increased. The goal is to find a compromise between a good correlation and a sufficient number of decompositions to provide a reasonable accuracy for predicting the value of the DV. This determines the number of pattern vectors used as our termination criteria (Figure 3).

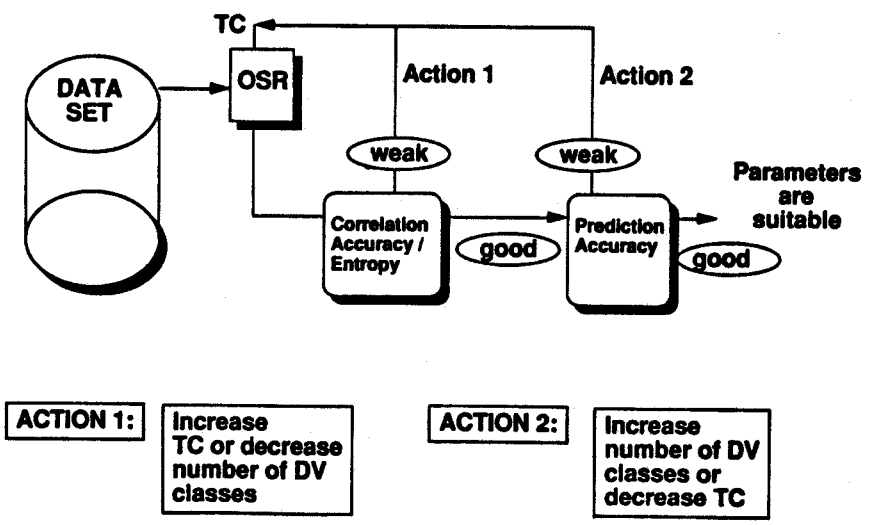
- How to create suitable DV and IV classes (Steps 1 and 2)?

  Whenever the variable is either continuous or ordinal, the range is divided in a fixed number of classes. This number may be fixed through a trial and refinement procedure (Figure 3).

  At least two concurrent factors have to be taken into account:

  - The amount of data available: Increasing the number IV classes decreases the average number of reductions during the OSR process, because the average set reduction rate (ratio # of pattern vectors before red./ # of pattern vectors after red.) decreases. This tends to decrease the DV prediction accuracy. Whenever the number of DV classes increases, the ratio # pattern vectors / # of DV classes decreases and the calculation of the conditional probabilities is less accurate. Thus the strength of the correlation *Accuracy/Entropy* is also very dependent on the number of defined DV classes. So decreasing the number of DV classes is also a way of improving the correlation. One has to determine, based on the learning sample, which of the two solutions: decreasing the number of DV classes or increasing the termination criteria, is affecting the DV prediction accuracy the most.

Figure 3 - Tuning the OSR parameters



- The granularity of the classes: Decreasing the number of DV or IV classes may make predictions more inaccurate because you may be missing real distinctions in the values of the data. Granularity is clearly dependent upon the ability to collect the data accurately. The required accuracy of the DV prediction is a major criterion for DV granularity.

Whenever the variable is nominal, categories may be grouped to form IV or DV classes. With respect to DVs, the grouping will depend on the classification to be performed and also the size of the data set , as above. For IVs, grouping may be subjective and depends upon the goals of the analysis. For example, assume one wishes to predict productivity and one of the available IVs is "Programming language used". If the possible instances for the variable are "COBOL, FORTRAN, ADA, C++," one could create a class "high level language" containing ADA, C++ (because they allow better rates of code reuse and easier testing procedures) and a second class "Low level languages" containing COBOL, FORTRAN. If the amount of data makes it possible, four classes with the four programming languages may be used.

- How does one estimate the **conditional probabilities** that the object to be assessed falls into the various DV classes (Step 5)?

  A simple rule would be to calculate the probabilities as the ratios of pattern vectors falling into the various categories versus the total number of pattern vectors. This is the only solution for discrete DVs because there is no notion of distance in the range of values. A more refined approach for continuous DVs might be to sum the distances between the subset pattern vectors and the class mean for each of the DV classes. Call this TD_n, where n represents the class index. Note that TD_n is inversely proportional to the concentration of pattern vectors around the class mean for class n. Then calculate:

$$P(C_n/x) \approx \frac{1 - (\frac{TD_n}{\sum_{i=1}^{m} TD_i})}{m-1}$$

  where m is the number of DV classes.

  This formula assumes that the probability is inversely related to the total distance (TD_n) of the pattern vectors to the class mean. This approach refines the probability calculation since it takes into account the distances between the subset of pattern vectors and the class means, no just their membership in a specific class. We can further refine this probability calculation by defining TD as the sum of the exponentials of the distances in order to decrease the weight of the pattern vectors furthest from some particular class mean.

The OSR process could be improved by investigating the following issues:

- Could non-parametric statistical classification techniques (e.g. K-nearest neighbors) [COV67] be used in order to improve the accuracy of the results ?

These techniques could be used, for example, in the terminal subsets to help refine the results in cases where the entropy is high. The K-nearest neighbors technique yielded interesting results in experiments on data sets other than the one used as illustration in this paper. This approach needs to be further investigated in order to come up with effective empirical procedures.

- Could some decomposition heuristics be used to improve the pattern recognition process effectiveness?

  The simplest way of building an optimized set is to select in a stepwise manner the IVs yielding the best decomposition entropy until the Termination Criteria is reached as described above. Although this is certainly the fastest way of generating an optimized set, it is not the most efficient in terms of pattern recognition. One can imagined more sophisticated heuristics for example to get better entropies and therefore more accurate estimations. The user could select, for the S first stages of reduction, the A IVs yielding the best decomposition entropies. Then, in order to make an estimation of the studied dependent variable, the user would have to consider S^A terminal subsets (each of them yielding an independent result). These results can be weighted and averaged. The weights can be calculated using, for each terminal subset, the respective entropy and/or number of pattern vectors. Moreover, the variance of the obtained results is another insight into the reliability of the prediction.

4.2 Optimized Set Reduction and Decision Trees

The reader may notice similarities with decision tree techniques (e.g. the use of entropy as a selection function). However, the approach differs in several critical points, the two most significant discussed below. Moreover, unlike decision trees whose the goal is classification [BRE84], the goal of OSR is the estimation of the conditional probabilities on DV ranges associated with specific pattern vectors. These probabilities are intended to be used for prediction, risk management and quality evaluation as described in Section 4.3.

Decision trees and OSR use different decomposition strategies. For example, Selby and Porter identified two different selection functions (linear and logarithmic) [SEL88]. Since the two functions showed similar behaviors and only the logarithmic function is generalizable to N classes where $N > 2$ (note that decision trees usually deal with binary decisions), we have been using the logarithmic function, i.e., the information theory entropy function. Even with this function, some heuristics still need to be developed in order to improve the IV selection and the pattern recognition process. In the decision tree approach, one set of pattern vectors is decomposed by selecting the IV yielding the best average weighted entropy across the various formed subsets. This decomposition is repeated in each branch of the formed tree until the predefined termination criterion is met. The Optimized Set Reduction approach tends to be more effective in recognizing significant patterns. The following example illustrates a typical case where the difference is noticable:

Let us assume that Complexity ($CPLX$) and Personnel capability ($PERS$) are among several IVs available for the decomposing the available data set, S, to predict the dependent variable, productivity. S can be decomposed into two pairs of subsets $SS1, SS2$ and

$SS3, SS4$ using $CPLX$ and $PERS$, respectively. Assume that $SS1$ and $SS2$ contain respectively low and high complexity projects. In a similar way, assume that $SS3$ and $SS4$ contain projects with low and high experience personnel, respectively.

Let the average entropies for each of the two decompositions be $E1$ and $E2$ where $E1 < E2$. The IV $CPLX$ is thus selected for the first decomposition step of the decision tree. Now suppose the $entropy(SS1) > entropy(SS3)$ because for a low complexity system, $PERS$ is the most influential parameter. However, suppose $entropy(SS2) < entropy(SS4)$ because $CPLX$ is still the most significant IV for complex systems. If we want to assess an object (e.g. project) $o1$ that falls in either $SS1$ or $SS4$ (according to the selected decomposition), the result will be different according to which fo the two decomposition techniques is used. For the decision tree and OSR techniques, the subsets used for estimating productivity will be $SS1$ and $SS3$, respectively. To assess an object $o2$ that falls in either $SS2$ or $SS4$, the result with both techniques will be the same: $SS2$ will be used.

This example shows that the OSR technique is more effective since it always selects the decomposition that yields the best entropy for the object to be assessed and is therefore more likely to achieve better entropy values with fewer levels of decomposition. This property is crucial if you consider that the number of possible decompositions is very limited for small samples.

In conclusion, the decision tree technique may be seen as an optimized partition of the data set. The OSR approach is a set reduction process for recognizing the most significant (with respect to its pattern) subset for the specific object being assessed. This makes the OSR technique more calculation intensive, but all automatable.

Another important difference between the OSR and decision tree technique is the definition of the decomposition termination criterion. In the decision tree approach, the termination criteria is some minimal class conditional probability. If one of the two classes of the DV range is above this probability, it is selected as the one to which the object to be classified belongs. This selection appears to be arbitrary. In the OSR approach, the termination criteria is based upon the need to evaluate the accuracy of the prediction, i.e., it is determined by the correlation between accuracy, e.g., MRE and the selection function, e.g., entropy.

4.3 Prediction, evaluation and risk management

The three processes, prediction, quality evaluation and risk assessment, are all based on the recognized patterns and follow a similar quantitative approach even though they apparently deal with three different purposes.

- Prediction:

  In this case, one is interested in estimating only one Dependent Variable based on the set of available Independent Variables. The dependent variable is a measurable object characteristic that is not known or accurately estimatable at the time it is needed. For example, one may wish to predict the error rate expected for a particular project development process in order to determine whether to apply a particular code inspection intensive development process. So, one tries to estimate the error rate

based on other characteristics (IVs) that may be measured, evaluated subjectively with a reasonable accuracy, or estimated through other models.

If the dependent variable is defined on a continuous range (i.e. the notion of distance between two values on the range is meaningful), the following approach may be used: by dividing the DV range into m successive intervals (C_i classes: $i \in (1...m)$) and calculating $P(C_i/x)$ for each of them, we have in fact approximated the actual density function $P(DependentVariable/x)$ by assuming it to be uniform in each class C_i.

Therefore, the following expected μ_i value can be calculated on C_i:

$$\mu_i = E[Productivity/C_i, x] = \frac{lower\_boundary\_C_i + upper\_boundary\_C_i}{2}$$

In other words, the actual density function is approximated by a histogram, where each column represents the conditional probability of a particular pattern vector x that lies in a particular DV class C_i. No assumption has been made with respect to the form of this probability density function.

The expected value on the total DV range can be approximated as follows:

$$E[Prod/x] = \mu \approx \sum_{i=1}^{m} P(C_i/x) \times \mu_i$$

This expected value can be used as an estimate of the Dependent Variable. The average error interval that can be expected may be estimated quite accurately by using the correlation of accuracy to entropy. This correlation will be confirmed by the experiments described in Section 5.

If the Dependent Variable is defined on a discrete range, then prediction becomes a classification problem:

Given a set of probabilities that a particular pattern vector x belongs to each DV class C_i, the decision maker must decide to which class to assign x. The class with the highest probability may not always be chosen. Rather, we may choose the class based upon the loss associated with incorrect classifications. This is the Bayesian approach. A risk (or loss) matrix L has to be defined by the decision maker where Lij represents the loss of having chosen the strategy appropriate for C_j when the DV class (or state of nature) is actually C_i. A Bayesian classifier [TOU74] will try to minimize the conditional average risk or loss $Rj(x)$ (j = 1 ... m) considering the m defined DV classes.

$$Rj(x) = \sum_{i=1}^{m} LijP(C_i/x)$$

$P(C_i/x)$ represents the probability that pattern vector x comes from the pattern class C_i. The bayesian classifier assigns a pattern vector x to the class j with the lowest R value.

- Risk management:

  Software development organizations are interested in assessing the risk associated with management and technical decisions in order to guide and improve the development processes. Referencing [CHA89], the risk associated with an action (e.g. software development) may be described through three dimensions:

  - $D1$: The various possible outcomes
  - $D2$: The potential loss associated with them
  - $D3$: The chance of occurrence for each outcome

  One encounters multiple kinds of interdependent risks during software development (e.g. technical, schedule, cost) [CHA89] and this make risk management and models complex. Also, the notion of risk is by definition subjective because the associated loss strongly depends upon one's point of view. Charette in [CHA89] writes:" One individual may view a situation in one context, and another may view the exact same situation from a completely different one". According to his/her goals and responsibilities, one will define the risk in different ways, in the form of various models.

  If we try to make the link between the above description of risk and OSR, the following straightforward associations may be established:

  - Outcomes (i.e. dimension $D1$) and DV classes.
  - Potential loss (i.e. dimension $D2$) and distance on the DV range between the DV class mean and the planned DV value.
  - Chance of occurrence (i.e. dimension $D3$) and conditional probability for each DV class.

  In order to analyze risk during software development, we propose the following approach based upon OSR:

  First, based on the three previously described risk dimensions, we calculate the expected difference (distance on the range) between planned and actual values for each DV representing a potential risk (e.g. schedule, effort, ...). Let us call these distances "DV expected deviations". From a decision maker's perspective, the potential loss resulting from his/her decisions is intrinsically a function of several DV expected deviations that may be seen as a specific and subjective risk model. Therefore, a "loss function" is used as a risk analysis model and may be defined as a function that combines several DV expected deviations, parameters (e.g. reflecting management constraints) and constants (e.g. weights). The calculation details are illustrated in the example below.

  Decision making will result in various sets of IV instances and therefore will have an impact on the numerical results of an OSR. Through this mechanism, management choices will have an impact directly on the calculated DV plan/actual deviations and indirectly (i.e. through the selected loss function) on the calculated potential loss.

  Consider the following example with the two continuous DVs, productivity and fault rate. A budget and schedule have been imposed on the project manager by upper

management. Therefore a specified productivity P_r will be required to reach the management goals. From the point of view of the project manager, the risk of failure may be represented as a simple function calculating the Productivity Expected Deviation (PED):

$$PED = \sum_{i=1}^{m} P(C_i/x) \times (P_r - \mu_i)$$

where μ_i is the mean of C_i

According to the result of this estimation, the project manager will be able assess the difficulty of the job and make a decision with respect to the development process in order to alleviate the staff loading and make a suitable trade-off between quality and productivity. Some analysis can be performed by the manager to see how the risk evolves according to controllable project parameters (i.e. some of the Independent variables). If the project manager wants to make a risk/effort trade-off, in order to improve competitiveness on a commercial proposal for example, he/she can calculate how the risk evolves according to the productivity required. Based on these observations, a suitable risk/effort tradeoff can be selected to maximize chances of success.

One's perspective of risk may be more complex than the previously defined function, PED. For example, assume that a contractor wishes to define risk of financial loss if the system is delivered late and/or there are effort overruns. One can define the Schedule Expected Deviation (SED) as the expected delay, i.e., the difference between the planned and predicted schedule and the Effort Expected Deviation (EED) as the expected effort overrun, i.e., the difference between the planned and predicted effort expenditures. Then

- $SED = \frac{Estimated\_Size}{PED \times Avg\_Team\_Size}$
- $EED = \frac{Estimated\_Size}{PED}$

where $Estimated\_Size$ is either a parameter, like Avg_Team_Size (i.e.

provided as an input by the manager), or another Dependent Variable (i.e. result of an OSR using some Function Point-like metrics, for example, as IVs). So the financial loss function can be defined as a function of both variables SED and EED.

Now suppose that the cost of delay on a particular contract is exponential to the delay itself. This exponential assumption is based upon predictions with respect to the delay of other projects dependent upon the completion of this project and the resulting compensations to be given to the customer. Thus, the SED needs to be weighted by some Cost per Delay Unit that is an exponential function of SED, call this CDU. Also suppose that CEU is the average Cost per Effort Unit, i.e., the average cost per staff hour for the specific project development team.

Then we can define

$$Financial\_loss = SED \times CDU + EED \times CEU$$

- Quality Evaluation:

  In any quality model, you need a baseline in order to be able to make sensible comparisons. For example, let us assume that the quality perspectives of interest (i.e. quality drivers) are *productivity* and *fault_rate* since management views quality as reliable and inexpensive software.

  Assume that using some project features as IVs, the OSR approach yields clear patterns (i.e. low entropy) with respect to productivity in the available data set. These patterns represent the expected productivity distributions in the current development environment for the project under study. The relationship between the actual productivity for the project under study and the expected value of the predicted patterns provides the basis for quality evaluation, from the productivity perspective.

  For example, suppose the actual productivity for the project under study falls far below the expected value of the predicted patterns. This implies that the quality of the project with respect to productivity is low. Using the pattern as a basis of comparison, we may ask where the difference comes from.

  Several causes may be investigated: incomplete or inadequate data collection, some possible new features or variables affecting the development process, or, of course, the process quality (e.g. process conformance) is quite low.

  In order to quantify quality, from the perspective of productivity, according to some quality model, a quality value could be defined as a function the distance between the actual productivity and predicted value(s) of productivity based on the recognized pattern(s).

  This distance may be defined as:

$$Prod\_deviation = AP - \sum_{i=1}^{m} P(C_i/x) \times \mu_i$$

  with AP the actual measured productivity.

  If we try to include in the quality model both the *Fault_rate* and *Productivity* quality drivers and we assume an approach similar to the *Prod_deviation* evaluation for calculating a *Fault_deviation*, then a global quality evaluation may be formalized by the following quality model.

  Let us define NFD as *Fault_deviation* (i.e. fault rate deviation) normalized by the fault rate standard deviation in the available data set and NPD as the equivalent variable for *Prod_deviation*. Based upon these unitless deviations, we define the following quality model:

  - If $NFD < 0$, $NPD > 0$, the larger $|NFD \times NPD|$ is, the better the quality.
  - If $NFD > 0$, $NPD < 0$, the larger $|NFD \times NPD|$ is, the worse the quality.
  - If both NFD and NPD are negative, the larger $\frac{NFD}{NPD}$ is, the better the quality.
  - If both NFD and NPD are positive, the smaller $\frac{NFD}{NPD}$ is, the worse the quality.

- If both NFD and NPD have the same sign and $\frac{NFD}{NPD}$ has a value close to 1, then quality may be assessed as average or nominal.

This particular quality model takes into account two dependent variables and illustrates that a quality model may be a subjective function of several distances on the rerspective dependent variable ranges. This model might be modified, according to the user perspective of quality, to change the weighting of the various DVs or factors, e.g., doubling the effect of fault rate in the evaluation of quality.

4.4 Development Environment Analysis

The user may wish to analyze the collected historical data in order to get some intuition about the development environment (e.g. what significant features affect productivity?). This could help an organization improve its development process(es) and management techniques by tailoring them to their environment.

To obtain experimental results, one may perform an OSR on each of the pattern vectors in the available data set, using all other pattern vectors as the learning sample. Based on the N Optimized Set Reductions (N being the number of pattern vectors in the test sample), the occurrences of each IV in the N terminal subsets are counted and weighted.

Several possibilities are available to weight the counts :

- according to the ranks where they appear in the terminal subsets
- according to the number of data points on which they are selected.
- according to the entropy variations they create

This count should give an insight into the significance ranking (according to the dependent variable) of the available IVs in the development environment where the data has been collected. Obviously, these counts can be performed on subsets of the data set and compared (e.g. real time versus business applications).

If an IV is known from experience as very influential but it does not appear very often in the terminal subsets, several reasons are possible :

- the IV represents a factor that is very influential with respect to the dependent variable but is quite constant in the development environment. This may due to inappropriate interval boundaries on the range of the independent variable that do not sufficiently differentiate the objects of study in the learning sample. This ability to differentiate may change over time with the evolution of technology.
- the IV is not selected because it is highly interdependent with other available IVs selected during the set reductions (this issue is explained more in detail in Section 6).
- an innacurate data collection

5 Experimental Results

In this section we demonstrate the effectiveness of the approach by applying the OSR modeling process to the problem of cost estimation and showing that OSR is able to recognize meaningful patterns on the available data sets. Although we will only be dealing with the prediction capability, this experiment can be viewed as demonstrating the effectiveness of the risk assessment and quality evaluation capabilities since the three processes have been shown in Section 4.3 to all be based on the conditional probabilities estimated on the dependent variable range. Therefore the accuracy of the three formulas are all dependent on the significance of the recognized patterns.

We will also determine the most influential factors affecting productivity, regardless of the development environment. We will do this by assessing the significance of each factor in predicting productivity. We will then analyze the results to determine if they are reasonable.

The largest part of our data set comes from the COCOMO database published in 1981 (63 data points/pattern vectors) [BOE81]. A second source of data is provided by Kemerer (15 data points), collected in a COCOMO format and published in 1987 [KEM87].

The first data set is a mix of business, system, control, high level interface and scientific applications. A significant percentage of these projects has been developed in FORTRAN (38%) and a very small number in Cobol (8%). The second set of projects reflects a variety of data processing applications most of them developed in Cobol (87%). The following sections describe an experimental evaluation of the OSR technique as applied to effort estimation based upon these two data sets. Results obtained with the OSR approach are compared to results from other standard approaches in order to provide some basis for comparison.

In the following sections, we use three different data analysis procedures to make cost predictions for the fifteen Kemerer's projects based on their COCOMO cost drivers:

- the OSR approach.
- a calibrated intermediate COCOMO model.
- a pure regression approach.

In what follows, we will use the term data set to refer to the COCOMO and Kemmerer data sets, the test sample refers to the Kemerer data set which is the sample on which we are going to assess the OSR capabilities, and the learning sample for each optimized set reduction is the data set minus the project we are trying to assess.

Thus, 15 optimized set reductions will be performed, one for each of the test sample pattern vectors. Each time, the pattern vector to be assessed will be removed from the whole data set to form the learning sample (77 projects) in order to get realistic results. This would be similar to a situation where an organization has collected data about 77 projects and wants to assess a new project.

However, the reader must consider that various subsets of the 78 projects have been developed in various environments, at different points in time and collected by different people

according to different procedures, in different organizational structures. The difficulties in tailoring the COCOMO cost drivers to various environments causes a loss of consistency in the data collection regardless of the analysis technique. Moreover, it is important to notice that the project productivities lie over a very large range (i.e. from 20 to 2491 LOC/MM). The 78 pattern vector data set is small enough to assess the capability of the approach to deal with small samples. The number of independent variables used (15) compared to the available data set and the nature of these IVs (i.e. ordinal, nominal) make any pure regression analysis based approach difficult to apply.

In order to evaluate the performance of the OSR technique, we will first reference the results obtained by Kemerer in using some of the main models available:

- SLIM: MRE=772%
- Intermediate COCOMO: MRE=583%
- FP: MRE=103%
- ESTIMACS: MRE=85%

According to the author, one of the reasons for which the last two models yielded substantially better results is that they are built on business data processing application projects. Since the data used to develop the FP and ESTIMACS models were either not available or not provided in a COCOMO format, we cannot include them in our data set even though they may be more suitable as a learning sample for the fifteen projects of the test sample.

5.1 Understanding and assessing the cost of development using OSR

As the Dependent Variable, we use project productivity (i.e. size/effort). The size metric used is the "Adjusted Delivered Source Instruction" as defined in COCOMO, and the effort unit is staff-months. The Independent variables are the COCOMO cost drivers. The ranges for all but one (SCED) of the IVs have been decomposed into two intervals (i.e. the boundary being located either just below or above nominal depending on the IV) and the DV range into five intervals containing an equivalent number of pattern vectors, to the extent possible.

Using the termination approach defined in Section 4.1, the termination criterion was set to 8 projects, after being tuned based upon the learning sample. No more sophisticated decomposition heuristic was used.

Table 1 gives the results for each of the fifteen data points of the test sample. The five columns contain the project number, the actual productivity, the predicted productivity, the actual effort, the predicted effort and the entropy yielded by the OSR processes.

Table 1

PROJECT	Actual prod	Predicted prod	Actual effort	Predicted effort	ENTROPY
1	884	299	287	846	0.63
2	491	935	82	44	0.24
3	580	674	1107	668	0.45
4	2467	643	87	333	0.06
5	1338	952	336	473	0.27
6	595	1196	84	42	0.47
7	1853	1016	23	42	0.47
8	1535	1006	130	199	0.52
9	2491	431	116	670	0.56
10	542	1028	72	38	0.06
11	983	1028	258	247	0.06
12	557	1025	231	125	0.06
13	1028	1035	157	155	0.06
14	667	1070	247	154	0.27
15	861	964	70	62	0.06

Despite an encouraging average prediction accuracy which is discussed below, the three data points with highest productivity (project 4, 7 and 9 in Table 1) yield large effort overestimation. These three projects have a productivity far above the other 75 projects of the learning sample and these productivity values might be the result of several problems:

- the size evaluation (i.e. adjusted KDSI) has not been performed according to the rules described in [BOE81]. This lead to a size overestimation.
- something occurred that was not captured by the 15 COCOMO cost drivers.

More information on these particular projects would be necessary in order to make a more thorough analysis. However, in order to keep these three projects from introducing noise in our analysis, we will also analyze the results obtained by removing them from the learning sample.

Tables 2.1 and 2.2 summarize the previous results by giving, for three entropy intervals, the average Magnitude of Relative Error of the effort estimation in the respective intervals for

each modeling technique, (columns $MRE-OSR$, $-CC$ for calibrated COCOMO and $-SR$ for stepwise regression), and the percent of the test sample falling in that interval, (column $\%TS$). Table 2.1 takes into account the projects $4, 7, 9$ and Table 2.2 ignores them. This provides some insight into the correlation between the accuracy of the effort estimation and the entropy.

Table 2.1

SUBSET	MRE-OSR	MRE-CC	MRE-SR	% TS
F <= 0.06	65%	34%	78%	40%
0.06 < F<= 0.47	50%	168%	80%	40%
0.47< F<= 0.63	246%	622%	261%	20%

Table 2.2

SUBSET	MRE-OSR	MRE-CC	MRE-SR	% TS
F <= 0.06	21%	33%	47%	42%
0.06 < F<= 0.47	43%	81%	60%	42%
0.47< F<= 0.63	124%	76%	105%	16%

The reader should notice from Table 2.2 that as the entropy gets lower, the accuracy (i.e. MRE) gets higher. Therefore, whenever one makes an estimate, the entropy of the pattern

on which the estimate is based is calculable and provides an assessment of the accuracy of the estimate. For example, if the obtained entropy is around 0.06, you know that the expected accuracy should be around 21%, according to the results obtained in Table 2.2. In Table 2.1, the results seem disturbed by the three high productivity projects. They create large standard deviations with respect to the accuracies in each entropy interval. Table 2.3 gives the standard deviations calculated in each interval. The first column takes into account the three high productivities projects. The second one ignores them.

Table 2.3

SUBSET	SD-15	SD-12
F <= 0.06	102	20
0.06 < F<= 0.47	9	4
0.47< F<= 0.63	323	58

According to the second column of Table 2.3, the standard deviations of the average accuracy in the two lower entropy intervals are relatively small showing that the prediction accuracy may be estimated with a reasonable precision. If the entropy is below 0.06, the accuracy has a high probability of being under 41% (average MRE plus the standard deviation in this specific entropy category). For example, if the distribution of the residuals around the average MRE is assumed Normal, then this probability value lies around 85%. Since the highest entropy interval contains only two projects, it is difficult to draw any conclusion from its high standard deviation. However, a high standard deviation of the MRE values should be expected in high entropy categories since patterns are not very significant and therefore yield inaccurate predictions.

A problem that may affect the analysis is that the estimation accuracy may be disturbed significantly when the actual productivities of projects are close to the extreme boundaries of the productivity ranges. This is because the density of projects in these parts of the range may be much lower than projects found in the middle of interval ranges. Only intuition based upon understanding the particular situation can help the manager detect an unusual, extremely low/high productivity so the effort estimate may be increased/decreased. Obviously, something that has never or rarely occurred is difficult to predict.

In summary, based on Table 2.2, the results are very encouraging, especially if we consider that we are using only fifteen new projects and with different profiles. That is, despite the constraints, impressive results ($averageMRE = 32\%$) was achieved in 84% of the cases

(projects $4, 7, 9$ are ignored). We have clearly satisfied requirement R_2 from Section 2.2. Another important point is that no assumptions have been made of the kind discussed in requirement R_1.

Thus, someone with only 15 projects specific to his/her environment may use the COCOMO data set to form a learning sample on which to construct an OSR model and get some useful predictions.

In order to determine the most influential factors affecting productivity, counts of occurrences of the various independent variables have been performed according to the description given in Section 4.4. Table 3 has three different columns. The first gives the raw counts of occurrences, i.e., the number of occurrences of each IV used to create some decomposition for the given test sample. The second gives the same counts of occurrences weighted according to the level of decomposition where the independent variables appears in the OSR processes (e.g. if an IV appears at the level two of a decomposition then its occurrence count increases by $1/level = 0.5$). The third column shows the total variation of the productivity mean due to each cost driver in the fifteen OSRs performed. This table gives several insights into the influence of each of the fifteen cost drivers on the productivity of the fifteen business applications studied. The three columns give three different perspectives of the impact of each cost driver on the predictive ability of the model.

Table 3

C. DRIV	OCC	W. OCC	MEAN VAR.
RELY	2	1.33	720
DATA	3	2.33	594
CPLX	1	0.25	270
TIME	1	0.33	280
STOR	11	10.5	2770
VIRT	4	2.66	65
TURN	4	1.83	85
ACAP	7	3.5	706
AEXP	1	0.33	350
PCAP	7	2.66	2075
LEXP	0	0	0
VEXP	1	0.5	268
MODP	1	0.5	270
TOOL	0	0	0
SCED	0	0	0

When a factor yields a low number of occurrences, several causes may be considered:

- The factor is quite constant in the learning sample and therefore will not help in differentiating projects. It should be noted that when a factor is constant in an environment, there is a loss of information about the effects of that variable.
- The factor has not much influence on the Dependent variable studied.
- There is an interdependence between the IV and some more influential IV.

Table 4 provides an assessment of the influence of the cost drivers on productivity relative to only one of the three columns from Table 3, the unwieghted occurrences. Table 4 has 2 columns: the first one gives non-weighted counts of occurrences and the second one shows the cost drivers matching them. Two distinct cost driver subsets may be observed that yield very different counts of occurrences (≤ 4, ≥ 7). Considering the test sample size, we may only say that there is clearly a set of three very influential cost drivers for which the data collection must be as accurate and consistent as possible, (ACAP, PCAP, STOR).

Table 4

# OCC	Cost Drivers
0	Tool, Sced, Lexp
1	Cplx, Time, Aexp,Vexp,Modp
2	Rely
3	Data
4	Virt, Turn
7	Acap, Pcap
11	Stor

Storage constraints appear to be very significant. This makes sense for systems focusing on data processing and dealing with very large amounts of data. The data set shows that most of these fifteen projects fall in the categories "high" and "very high" with respect to the cost driver DATA which makes the previous statement reasonable. The cost driver

STOR appeared at the top level of decomposition in ten of the fifteen OSRs performed. The average entropy for the ten projects is much lower (i.e. 0.17) than the average entropy for the five projects where STOR was not used (i.e. 0.52). This furthers the argument that STOR is very significant for the prediction ability of the OSR model for this particular test sample. STOR was not used in any case where storage constraints were rated above nominal. As a consequence, high storage constraints make productivity difficult to predict for this particular data set.

Of course, as shown in Software Engineering Economics [BOE81], the staff capabilities (Analyst, Programmers) play a crucial role in achieving optimal productivity. When STOR, ACAP and PCAP were all included in the decompositions, the entropies were the best obtained (i.e. 0.06). Therefore, ACAP and PCAP seem to significantly improve the average entropy of the recognized patterns (i.e. 0.06 instead of 0.17). Moreover, all the projects where ACAP and PCAP were used had high capability teams. This shows that productivity predictability is more accurrate at higher ranges of capabilities.

Table 4 shows that the reliability and complexity factors have a weaker influence on productivity than expected (according to the results published in [BOE81]). However, the fact that most of the fifteen projects of the test sample fall in the "nominal" category and there are no extreme complexities or reliabilities present may explain the lack of significance of these parameters for this particular test sample.

Therefore, based on the above examples, it should be noted that the significance of the IVs (i.e. cost drivers) in predicting productivity seems to be very dependent on the localization of the test sample pattern vectors on the IV ranges themselves. STOR does not seem to help predict productivity when the storage constraints are high in the range (i.e. above nominal). In this case, the variability (and the entropy) in the produced patterns is larger and therefore STOR is not selected in the decompositions. The same phenomenon may be observed for low ranges of PCAP and ACAP. With respect to CPLX and RELY, results are more difficult to interpret because of the lack of variation in the test sample. However, their low significance in the test sample might be explained as a consequence of the nominal reliability and complexity of the projects included in the test sample. Considering that most of these projects are business data processing applications, this result seems to make sense.

Neither "Virtual machine volatility" (VIRT) nor "Virtual machine experience" (VEXP) show much influence here on productivity. There is nothing in the data that explains this result and therefore we may conclude that these factors are not significant. The factors MODP (i.e. modern programming practices) and TOOL (i.e. use of software tools) show almost no significance. This result contradicts the results shown in COCOMO. No interdependence with other factors was detected in the data set. This latter result may be due to a weak variability in the level of technology involved in the system developments or inconsistent data collection between the two data sets used. For example, the definitions of the MODP categories are quite fuzzy and leave room for interpretation and inconsistencies.

AEXP also has minimal impact. This may be understandable in this context where the application domain of these 15 projects does not involve high-technology or mathematically intensive problems. LEXP shows no influence on productivity. Considering the low com-

plexity of the programming language COBOL, the learning process for this language may be estimated as very fast for any experienced programmer. SCED (i.e. schedule constraints) shows no impact on productivity and therefore confirms the conclusions of the COCOMO model.

According to the counts of occurrences in Table 4, the three most influential cost drivers (i.e. ACAP, PCAP, STOR) belong to the category of the seven most influential cost drivers in the COCOMO's cost driver ranking, despite the different nature of the considered fifteen projects.

5.2 Effort Estimation Using a Regression Tailored COCOMO

To allow for an evaluation of value of the OSR technique for the prediction of productivity and effort, a comparison with more conventional techniques is provided. In Section 5.1, we referenced an evaluation by Kemerer [KEM87], where the average MRE was 583%. However, the model overpredicted on every project, so it seems that COCOMO may be calibrated too high for this environment. It seems unfair to directly compare uncalibrated results to the results obtained through the OSR technique. In this section we describe an experiment in tailoring the intermediate COCOMO model to the data supplied by Kemmerer using regression based techniques. The calibration technique chosen was the one that most closely resembled the previously described OSR experiment. A new model was developed from project data from both COCOMO and Kemerer, using the intermediate COCOMO project information. The base data set for the experiment was the data points of the COCOMO model, along with all but one of the data points from Kemerer's data set. The mode of the removed project was determined, and a model was developed from all of the projects of that mode from the base data set, with the local data points being weighted three times the COCOMO data points. A regression was performed on the equation $ln(E/\pi) = A + bln(S)$ (where π is the product of the effort multipliers) to determine the best values for the constant term a (where $a = e^A$) and the scale factor b. The values of the effort multipliers were not adjusted in any way. The model for the prediction of effort then becomes $E = \pi a S^b$.

This model was then evaluated by comparing its prediction to the actual effort for the remaining project. This experiment was done for each of the 15 projects in the Kemerer data set. Table 5 provides a summary of the experiment:

Table 5

Project	Predicted effort	MRE
1	230.9	.196
2	54.5	.339
3	1641.6	.483
4	120.2	.383
5	849.5	1.526
6	187.5	1.232
7	163.7	6.055
8	303.7	1.33
9	2105.8	17.153
10	68.2	.053
11	294.7	.139
12	134.9	.415
13	264.1	.682
14	383.0	.551
15	41.7	.403

The results are not so good. The average MRE overall is 206%; however, if project 9 (an apparent outlier) is removed from the data set, MRE is reduced to 99%. The results for projects of the semi-detached mode were better than either of the other modes. This was expected, as there are many more projects in this mode than in either of the other modes. For the semi-detached mode, the average MRE is 44%, with 3 of the 9 projects having an MRE of $< 20\%$. For the other modes, there may not be enough data points to accurately adjust the model to local environment. For the 2 projects categorized as organic mode, the average MRE was 104%, and for the 4 embedded mode projects, the average MRE was 623%. Removing project 9 from the data set lowers the embedded mode average MRE to 259%. The model still tends to overpredict, with overestimations on 2/3 of the projects, which furthers the notion that the COCOMO cost multipliers are not representative of the local projects. Further experimentation indicated that the best prediction could be obtained by ignoring the mode and all the effort multipliers, and utilizing only the local effort and size data in the development of the model. The fact that the mode was not useful when tailoring the model is either due to the small number of data points in the organic and embedded modes, or an indication that mode is not significant in this environment. This indicates that the COCOMO effort multipliers are either not significant in this environment, or that they do not have the appropriate values for this environment. Unfortunately, since the data

set is small, and the original values for the COCOMO effort multipliers were developed somewhat heuristically, we can not accurately adjust the values to be useful for prediction in this environment.

5.3 Effort Estimation Using a Pure Regression Approach

We performed a standard regression process having the following characteristics:

The dependent variable of the regression was productivity, as in the OSR experiment previously described. A linear functionnal form was used for the regression equation. The fifteen COCOMO cost drivers were the potential parameters included in the equation. Then a stepwise selection of the regression parameters was run based on the F partial values of the various regression parameters [DIL84].

The results for all the fifteen projects are the following:

- $R^2 = 0.36$
- $MRE = 115\%$

If projects $4, 7, 9$ are removed from the test sample (see previous sections), then the results are improved:

- $R^2 = 0.54$
- $MRE = 62\%$

Detailed results are provided in Table 6.

Table 6

Project	Predicted Productivity	Predicted Effort	MRE
1	553.5	458.2	.596
2	628.9	64.4	.219
3	849.6	529.7	.522
4	740.7	289.5	2.331
5	578.1	778.2	1.314
6	1049.1	47.7	.433
7	659.9	65.2	1.809
8	620.2	322.5	1.475
9	370.2	780.7	5.73
10	489 .0	79.8	.108
11	384.0	662.0	1.559
12	1304.9	98.6	.573
13	1159.9	139.1	.114
14	1292.2	127.5	.483
15	939.6	64.1	.083

5.4 A Comparison of the Three Techniques

Comparing the results of the OSR and regression based techniques leads to several observations. First, for this data set, the OSR technique provides a significantly better prediction than either a tailored COCOMO or a stepwise regression. For 10 of the 15 projects, the prediction of the OSR model was more accurate than that of both regression models. If outliers are not removed, the two regression based models had an average MRE of 206% and 115% respectively, while the OSR model had an average MRE of 94%. If the projects 4, 7 and 9 that showed extremely high productivities are not considered, the MRE for the regression models becomes61% and 62% respectively, while the OSR model is 47%. Moreover, the OSR results were obtained without using the notion of "development mode", making the estimation process easier. The results for OSR are much better that the regression techniques in the two lower entropy categories (32% vs. 57% and 54% respectively for the calibrated COCOMO and the stepwise regression models). This result should have been expected since poor entropy implies that no significant pattern has been found and so the poor entropy projects bias the OSR results negatively. Consequently, in the highest entropy category, regression based techniques can perform better.

A second benefit of the OSR technique its ability to provide an indication of the expected accuracy of the prediction, as demonstrated by the clear correlation of MRE to entropy. Projects with characteristics that have previously shown widely varying productivities (i.e. no clear patterns) are flagged with high entropies, allowing the manager to recognize that

the prediction may be suspect. The regression based models provide no such indication of the accuracy for an individual prediction.

For exmaple, the tailored COCOMO model provided relatively accurate predictions for projects of the semi-detached mode (an average MRE of 32%), except for a poor prediction for project 8, with an MRE of 133%. The prediction for project 8 using the OSR model was more accurate (MRE of 53%); however the prediction was flagged with a high entropy, indicating an unstable prediction. The regression based models provide no indication of the potential of the innacurate prediction, while the OSR technique indicates that no significant pattern has been recognized in the available data set. The worse prediction obtained by applying the OSR technique was for project 1 (if you eliminate projects $4, 7, 9$ whose the bad results are likely to be due to extreme productivities and missing IVs). In agreement with the expectation, this poor prediction is visibly flagged with the worse entropy (i.e. 0.63) among the fifteen projects of the test sample.

Despite this important advantages of the OSR technique with respect to prediction, the regression based approaches may be very useful because their prediction abilities appear better when OSR yields bad entropies (see Table 2.2), i.e., when no significant pattern has been recognized. Therefore, an overall prediction process could be defined as a combination of all the modeling techniques previously cited.

6 Data Analysis Disturbances

The decision maker will have to take into consideration the following issues in order to perform a sensible analysis. These problems will have to be better understood and investigated. Some procedures will have to be defined, from the perspective of the OSR approach, in order to alleviate their effects on the data analysis. This section of the paper does not give definitive solutions but rather poses the issues in clear terms and shows how an OSR approach might deal with the described problems.

6.1 Interdependence among explanatory variables

There are two different kinds of interdependence between two variables IV_1 and IV_2, i.e. positive and negative correlation. The first case is where the two variables have a similar influence on a Dependent Variable defined on a continuous range (e.g. DV decreases when either IV_1 or IV_2 increases). For example, let's suppose the dependent variable is productivity and for the given applications, code complexity is strongly correlated with the size of the database (i.e. number of entities and relationships for a relational model) . Both have a tendency to decrease productivity. The variables do not lose their explanatory power and will be effective for prediction using an "optimized set reduction" approach. However, the count of occurrences will be disturbed and will not represent accurately the independent influences of either "code complexity" or "database size" on productivity. There is a causality relationship between the two independent variables. A large "database size" will systematically create a high "code complexity". Thus, the numbers of occurrences for both the independent variables cannot be considered distinctively. However, the two variables

may be combined in a higher level IV called for example "System Complexity".

In the second case, the two variables have an opposite influence on the dependent variable. For example, the DV being productivity and due to an effective project management organization, experienced programmers are assigned on complex programs. This situation is much more harmful in the sense that the two IVs lose their explanatory power (i.e. the productivity will not seem to decrease with complexity). They are not likely to be selected in the decompositions of the OSRs despite their strong influence on productivity. The accuracy of the results should not be significantly affected. However, the actual impact of these factors on productivity becomes difficult to assess. With respect to the previous example, using the ratio Complexity/Experience as an IV could be a solution to investigate in order to minimize the impact of interdependence.

If a missing IV is interdependent with available IV in the data set, then the previously described problems occur and become even more difficult to detect.

From an OSR perspective, an easy way of measuring the interdependence between two positively correlated IVs IV_p and IV_q is to calculate the variation of occurrences O_p if IV_q is withdrawn from the used IV set.

Let us define:

- IVS_1 (IV set 1) $= IVS_2 - IV_q$
- O_{np} being the counted occurrences with IVS_n for $IV_p, n \in (1,2)$

The Level of Interdependence (LI) between A_p and A_q can be define as follows :

$$LI = \frac{|O_{1p} - O_{2p}|}{O_{1p}}$$

where the higher the value of LI, the greater the effect of interdependence on the analysis of the influence of the factors.

The effects of interdependence are not avoidable independent of the modeling technique. The only solution consists in taking a larger sample, preferably in a way that decreases interdependence between IVs. However, being able to detect such phenomena in the data set in a simple and effective way, is required in order to partially fulfill the requirement R_3 (Section 2.2).

This issue is still to be further investigated in order to come up with a well defined procedure based on OSR, correlation matrices, etc. and refined based upon lessons learned from experiments.

6.2 Outliers and Missing Independent Variables

The detection of outliers based on multiple regression is quite subjective and difficult to perform. Outliers are defined as: "Observations that have a disproportionate influence on the calculated model" [DIL84]. Some techniques are available to evaluate the influence

of a data point (or a group of data points) on the regression coefficients, coefficient of determination, or residuals. However, no objective rule or heuristic exists to determine the level when a (group of) data point has a "disproportionate influence". Also one outlier may mask the effect of another and the detection process may become even less effective. Then, you have to examine the influential effects of subsets of data points. It may be difficult to apply (in N-dimension space with N greater than 2) because there is no simple way (e.g., visually) of selecting the right group of data points in order to avoid the "masking effect".

Outlier detection in the context of an "Optimized Set Reduction" becomes much easier and more objective and therefore partially fulfills R_3. By performing OSRs, you obtain patterns on the DV range. If some data points are far from the main distribution forming the pattern, then they may be considered as outliers matching the following definition : "Outliers correspond to objects that do not lie in the expected DV intervals because the data collection has not captured the phenomenon responsible for the different behavior of these objects with respect to the DV". According to this definition, outliers are strongly related to the "missing independent variable" issue. Therefore, analyzing the outliers may lead to redefining the data collected.

In Section 5, three among the 78 projects present in the data set had extremely high productivities (i.e. projects 4, 7, 9 in the Kemerer data set). Accordingly, these three extremely high productivities were not explained by the fifteen defined cost drivers and are typical examples of outliers.

7 Learning using an OSR approach

Understanding, evaluating, predicting, and controlling software development requires the ability to build quantitative models of various software processes, products, and other forms of experience, e.g. resource estimation and allocation, defect prediction, reusability. Model development is further complicated by the fact that organizations change and our knowledge evolves over time. Therefore it is necessary that the models evolve over time in an effective way, e.g., valid, cost-effective.

Based upon our experience in trying to evaluate and improve software quality in several organizations [BAS85, BW81, RB87, SB88] a quality-oriented, evolutionary life cycle model has been developed, called the **Improvement Paradigm** (IP) [BAS89, BR88]. The IP is measurement based and requires the development of models that can learn and evolve with an organization.

The IP is defined as a set of six activities:

- $S1$: Characterize the current project and its environment.
- $S2$: Set up goals and refine them into quantifiable questions and metrics for successful project performance and improvement over previous project performances.
- $S3$: Choose the appropriate software project execution model for this project and supporting methods and tools.

- $S4$: Execute the chosen processes and construct the products, collect the prescribed data, validate it, and analyze the data to provide feedback in real-time for corrective action on the current project.

- $S5$: Analyze the data to evaluate the current practices, determine problems, record the findings and make recommendations for improvement for future projects. This is an off-line process which involves the structuring of experience so that it can be reused in the future.

- $S6$: Package the experience in the form of updated and refined models and other forms of structured knowledge gained from this and prior projects and save it in an experience base so it is available for future projects.

In the context of the Improvement Paradigm, two distinct kinds of improvement goals can be seen. First, the manager wants to improve the models that are being used to describe the processes of the environment. Additionally, it is desireable to improve the processes themselves to better meet organizational goals. Unfortunately, these improvement goals conflict. As the processes change, the models that describe these processes may lose there validity. In an environment featuring a great deal of change and experimentation with new methods, techniques and tools, it is essential that the models be refined and assessed continuously.

The OSR approach can be used to address both improvement goals. Analysis of the patterns found in projects with higher or lower productivities can help to focus areas of improvements in the development processes. For example, it may be noticed that projects with tight timing constraints and teams with little prior experience with the target machine typically have low productivities. The manager may choose to ensure that on projects with such timing constraints, the team must have significant target machine experience. Similarly, if a particular project profile yields a high entropy value (indicating widely varying productivities), the manager may make changes to the development process to obtain a more predictable project.

As the processes evolve, the OSR automatically incorporates the new experience into the models. Future predictions will be based on both the new and the old data in a manner transparent to the user. However, some independent variables may become less significant as the technology evolves (e.g. complexity, storage constraints). This may be easily observed by taking several test samples covering the time range and comparing the counts of occurences. If a trend is observed, then one of the two following conclusions may be drawn:

- Ideally, the more one collects data, the better are the chances of improving the process. However, if a metric is expensive to collect and it appears insignificant in the OSR experiments, the team in charge of the data collection might stop collecting the corresponding factors in order to lower the measurement cost.

- The intervals on an IV range, as currently defined, have become unsuitable over time. For instance, let us say the new systems have become more reliable. As a consequence, most of them will fall in the category "very high" and the factor RELY will lose its

explanatory power over time. The way data are collected and/or the way the IV range is divided must be changed in order to improve the prediction process.

8 Conclusions

The Optimized Set Reduction (OSR)has been developed in order to address the specific data analysis issues within the software development process, as defined by the Improvement Paradigm. The procedure has the following positive characteristics that allow prediction, risk assessment and quality evaluation:

- It makes no assumptions with respect to probability density functions on the dependent and independent variable ranges. It does not attempt to fit data to predefined distributions, rather it uses the data to approximate the actual distribution (i.e. patterns). Also, no particular mathematical relationship between the DV and IVs needs to be assumed. Thus OSR seems to fulfill R_1.
- It allows an estimation of accuracy for each analysis so we can answer the question : is the estimation usable? This fulfills R_2.
- It is robust to non-relevant and missing metrics and allows a more objective way of dealing with outliers. First, the OSR process is intended to select the combination of IVs yielding the best patterns and therefore selects automatically the most significant group of IVs. Significance has been evaluated by calculating the entropy of the distributions on the DV range. Non-relevant IVs will not be chosen. Second, OSR detects outlier more easily because it analyzes distributions on the DV range as opposed to influencing analysis of pattern vectors in the multi-dimension sample space. Third, OSR provides an easier way of detecting missing significant IVs by two different means, the recognition of bad entropies and the detection of outliers. The issue of interdependence between IVs requires more investigation. R_3 is therefore partially fulfilled.
- It handles discrete and continuous IVs in a natural way and therefore meets R_4.
- It provides an automated refinement of the model as new data is incorporated into the data set.
- The process for selecting IVs, among those available in the data set, can be automated. Thus, the prediction process may be effectively automated and supported by a tool.
- It provides more objective baselines for comparisons and therefore allows more sensible evaluations (e.g. product and process quality). Then learning procedures, based on more objective quantitative evaluation, may be more accurate and effective.
- The use of the approach supports common sense and intuition as opposed to complex mathematical assumptions. Therefore, project managers are more able to plan, predict, control and make corrective actions supported by intuition.

Finally, the results of the preliminary experiments have been encouraging. They strengthen the idea that predictions may be accurate enough to be usable, despite the inherent constraints of software measurement in a production environment (see 2.1).

A prototype tool supporting the OSR approach is being developed at the University of Maryland as a part of the TAME project.

Future research directions for this work include:

- refine the OSR process by addressing the issues presented in Section 4.1.2.
- analyze other data sets that highlight the issues related to discrete DVs.
- address the unsolved issues related to data interdependence.

9 Acknowledgements

We thank G. Caldiera, L. Kanal, C. Kemerer, B. Pugh and M. Zelkowitz for their excellent suggestions for improving both the structure and the content of this paper.

References

[SB88] R.W. Selby and V. Basili, "Analyzing Error-prone System Coupling and Cohesion", *TR-88-46, Institute for advanced Computer Studies*, University of Maryland, College Park, MD, June, 1988.

[RB87] H.D. Rombach and V. Basili "A Quantitative assessement of Software Maintenance: An Industrial Case Study', *Conference on Software Maintenance*, Austin, TX, September, 1987.

[BW81] V. Basili and D.M. Weiss "Evaluation of a Software Requirement Document by Analysis of Change Data", *Proceedings of the Fifth International Conference on S.E.*, San Diego, CA, March 1981, pp. 314-323.

[BAS85] V. Basili, "Can we Measure Software Technology: Lessons Learned from 8 Years of Trying", *Proceedings of the Tenth Annual Software Engineering Workshop, NASA Goddard Space Flight Center*, Greenbelt, MD, December, 1985.

[BAS89] V. Basili, "Software Development: A Paradigm for the Future (Keynote Address)", *Proceedings COMPSAC '89*, Orlando, FL, September, 1989.

[BR88] V. Basili and H. D. Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments" *IEEE Trans. Software Engineering* 14 (6), June, 1988

[BOE81] B. Boehm, *"Software Engineering Economics"*, Prentice Hall editions, 1981.

[CHA89] R. Charette, *"Software Engineering Risk Analysis and Management"*, Mc Graw-Hill, 1989.

[KEM87] C. Kemerer, "An Empirical Validation of Software Cost Estimation Models", *Communications of the ACM*, 30 (5), May, 1987.

[DIL84] W. R. Dillon, *"Multivariate Analysis: Methods and Applications"*, Wiley and Sons, 1984

[SEL88] Selby and Porter "Learning from examples: Generation and Evaluation of Decision trees for Software Resource Analysis", *IEEE Trans. Software Eng.*, 1988

[BRE84] Breiman and al. "Classification And Regression Trees", *Wadsworth, Brooks/cole advanced books and softwares*, 1984.

[TOU74] Tou and Gonzalez "Pattern Recognition Principles", *Addison-Wesley Publishing Company*, 1974.

[COV67] Cover and Hart "Nearest Neighbor Pattern Classification", *IEEE Trans. Information Theory, IT-13: 21-27*, 1967