# Software cost prediction with predefined interval estimates

Stamatia Bibi, Ioannis Stamelos, Lefteris Angelis

## Abstract

*Defining the required productivity in order to complete successfully and within time and budget constraints a software development project is actually a reasoning problem that should be modelled under uncertainty. One way of achieving this is to estimate an interval accompanied by a probability instead of a particular value. In this paper we compare traditional methods that focus on point estimates, methods that focus both on point and interval estimates and methods that produce only predefined interval estimates. In the case of predefined intervals, software cost estimation becomes a classification problem. All the above methods are applied on two different data sets, namely the COCOMO81 dataset and the Maxwell dataset. Also the ability of classification techniques to resolve one classification problem in cost estimation, namely to determine the software development mode based on project attributes, is assessed and compared to reported results.*

## 1. Introduction

Estimating the cost required for a software development project is one of the crucial aspects of project planning and management but still remains an open issue, due to the diversity of cost factors, their unclear contribution to productivity, the high degree of uncertainty and the lack of information in the early stages of software development. For these reasons, low accuracy and unsuccessful estimations seem to have been inevitable so far.

Software cost estimation actually involves the estimation of productivity or effort needed to complete a project. Most methods proposed produce point estimates of these attributes, along with prediction intervals, in an attempt to consider the uncertainty or risk associated to the estimation process. However, because software cost data sets are small (counting tenths of projects in most cases), estimate intervals are often too large to be useful for practical cost estimation. Therefore, researchers tend to neglect estimate intervals, focusing on the point estimates. This is controversial to the fact that the development of a software artifact is a human driven procedure, where unexpected problems may arise.

For a more realistic approach, it is necessary to consider both uncertainty and risk, weighing the chance of events occurring and the impact they might have. It is useful to reflect the level of uncertainty in the estimate and interval estimates can lead to that direction. Usually interval estimates are created during the estimation process by firstly making a point estimate and then assessing prediction intervals. However, there is also the possibility to pre-define the intervals of productivity before the estimate generation. This can be done in order to control the estimation procedure, distribute the projects in the training data set as uniformly as possible into the various productivity intervals and ensure that estimate intervals will not be too large.

The target of this study is to identify techniques that are capable to produce predefined intervals and to provide some evidence of the prediction accuracy of those techniques, comparing them also to techniques that have traditionally focused on point estimates. Classification and Regression Trees (CART) is a technique that may produce both point and interval estimates. Two techniques that produce interval estimates only are Bayesian Belief Networks (BBN) and Association Rules (AR). Known techniques focusing on point estimates are Ordinary Least Squares (OLS) and Forward Pass Residual Analysis (FPRA) and Analogy Based Estimation (ABE). Also the comparison between learning-oriented techniques (OLS, FPRA), machine learning techniques (CART, BBN, A.R) and expertise-based techniques

(ABE) is inevitable. All the above methods are applied on two different data sets, namely the COCOMO81 dataset and the Maxwell dataset. The differences in estimation accuracy between the techniques are presented and discussed in both cases.

An additional research target is to compare the classification techniques in classification problems that occur within software cost estimation. A typical example is the determination of the software development mode in the COCOMO81 dataset. The ability of the classification techniques to determine the software development mode based on project attributes is assessed and compared to reported results obtained through two other methods, namely hierarchical clustering and discriminant analysis [23].

Various studies have been conducted so far concerning the comparison and evaluation of different cost estimation techniques [3], [9], [17], [1]. In particular, some of them suggest the estimation of intervals [10], [12], [22] and one the estimation of predefined intervals [21]. Machine learning techniques such as Neural Networks [8], CART [20] have also been implemented but all producing point estimates. Especially for association rules only two studies have been found in the literature involving effort estimation [14], [15]. In these studies rules are extracted from decision trees, not from exploring frequently appearing item sets. This study examines the applicability of machine learning methods in estimating predefined productivity intervals.

This paper starts with the description of the data sets and data preparation in Section 2. Section 3 presents briefly the estimation techniques. Section 4 provides the results of each technique. In Section 5, the perceived advantages and the disadvantages of each technique as well as future work are discussed.

## 2. Data Set Description

The first dataset that was used in this analysis is the COCOMO81 dataset coming from TRW defense systems. This dataset consists of 63 projects. Apart from the general attributes, a set of 17 attributes, called cost drivers, is used in order to extract useful patterns. These attributes, based on the analysis of Kitchenham in [11], can take the following values: Super Low (SL), Extra Low (EL),Very Low (VL), Low (L), Nominal (N), High (H),Very High (VH), Extra high (EH), Super High (SH). Also KDSI is included, the delivered source instructions and adjusted KDSI. The data set is analytically presented in [11].

The second dataset that was used in order to extract models and evaluate them is a dataset coming from bank applications published in [16]. This database contains 63 projects from which 60 were used, the ones with no missing values. 22 variables are taken into consideration in order to establish their relationship with productivity. The following variables are contained: application type (customer service, MIS, transaction processing, production control, Information/On-line service), the hardware platform (networked, mainframe, pc, mini computer and multi-platform), the database architecture (relational, sequential, other), the user interface (graphical/gui, text user interface/tui), source (insource, outsourced). Another 15 variables influencing productivity are also present: customer participation (custpart), development environment adequacy (devenvad), staff availability (staffav), standards use (standuse), methods use (methuse), tools use (tooluse), software's logical complexity (cplx), requirements volatility (reqvol), quality requirements (qualreq), efficiency requirements (effreq), installation requirements (installation_req), staff analysis skills (staffanal), staff application knowledge (stappknow), staff tool skills (staff_tool_skills), staff team skills (stateam). The values that these variables take are Very Low(1), Low(2), Nominal(3), High(4), Very High(5).

For applying classification techniques and for extracting association rules the values of productivity were transformed from continuous to categorical by being classified into predefined intervals. One problem that arises is the choice of the number of intervals and the width of each interval. The number of intervals may be selected automatically according to various rules that are proposed in the statistical literature, or may be determined manually, keeping in mind that the narrower the interval, the more useful the estimate is. In this study we took the second approach, although the former definitely deserves further attention.

We preferred to consider intervals that may be appealing to software managers: relatively few intervals were chosen (because of the low number of projects in the datasets) with rounded lower and upper limits, so as to be easily identifiable by a human. This is best seen in the case of COCOMO81 dataset. Another issue is the width of the intervals. When possible, we chose intervals that grew progressively larger, so as to allow larger magnitude errors for higher productivity values. This approach had an additional important benefit: the projects were as uniformly as possible divided among the intervals, since in both datasets many projects are concentrated in small productivity values. The intervals chosen for the two datasets are presented in table 1.

*Table 1: Productivity Intervals*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| COCOMO DSI/MM | <30 | 30-60 | 60-100 | 100-160 | 160-270 | 270-450 | 450-750 | >750 |
| MAXWELL FP/TotalHours | <0.043 | 0.043-0.058 | 0.058-0.085 | 0.085-0.12 | 0.12-0.145 | 0.145-0.17 | >0.17 | |

## 3. Methods used for Cost Estimation

This section briefly describes the techniques applied by the authors in software cost estimation data. The selected techniques are the ones that are able to produce interval estimates taking into consideration the uncertainty that characterize our problem. Also these techniques produce results that are easily interpreted, understood and reproduced.

### 3.1. Classification And Regression Trees (CART)

CART is a widely used statistical procedure for producing classification and regression models with a tree-based structure in predictive modeling [4]. The CART tree model consists of an hierarchy of univariate binary decisions. The algorithm used operates by choosing the best variable for splitting data into two groups at the root node. It can use any of several different splitting criteria, all produce the effect of partitioning the data at an internal node into two disjoint subsets in such way that the class labels are as homogeneous as possible. This splitting procedure is then applied recursively to the data in each of the child nodes. A greedy local search method to identify good candidate tree structures is used. Finally, a large, high depth tree is produced and specific branches of this tree are pruned according to the stopping criteria, so as to avoid overfitting of the data and over-specialization of the model.

### 3.2. Association Rules (AR)

Association rules [7] are among the most popular representations for local pattern detection. Their target is to find frequent combinations of attribute values that lay in databases. An association rule is a simple probabilistic statement about the co-occurence of certain events in a database, and has the following form:

IF A1=X AND A2=Y THEN A3=Z

$$A1\_X+A2\_Y \rightarrow A3\_Z$$

where A1_X+A2_Y is considered to be the rule body and A3_Z is considered to be the rule head. This rule is interpreted as following: when the attribute A1 has the value X and attribute A2 has the value Y then there is a probability p that attribute A3 has the value Z, where $p=p(A3=Z|A1=X,A2=Y)$. This probability is called *confidence.* Another probability that defines each rule is *support* = p(A1=X, A2=Y, A3=Z). Certain constraints on confidence and support are used in order to select the most representative rules.

### 3.3. Bayesian Belief Networks (BBN)

Bayesian Belief Networks (BBN) [6] are graphical models and have become attractive because of their ability to represent uncertainty and to efficiently perform reasoning tasks. The knowledge that they manage is in the form of dependence and independence relationships, two basic notions in human reasoning. Both relationships are coded by means of the qualitative component of the model, the Directed Acyclic Graph (DAG). Each node represents a random variable that can take discrete or continuous values according to a probability distribution, which can be different for each node. Each link expresses probabilistic cause-effect relations among the linked variables and is depicted by an arc starting from the influencing variable (parent) and terminating on the influenced variable (child node). The presence of links in the graph may represent the existence of direct dependency relationships between the linked variables (that sometimes may be interpreted as causal influence or temporal precedence). The absence of some links means the existence of certain conditional independency relationships between the variables.

The strength of the dependencies is measured by means of numerical parameters such as conditional probabilities. Formally, the relation between the two nodes is based on Bayes' Rule: P(X|Y) = P(Y|X)P(X)/P(Y).

The rest of the methods considered for comparison are well known: COCOMO [2] is a widely known software cost model. OLS is also a statistical technique [5], applied often in cost estimation. Analogy-based estimation is based on the similarity of the target project to historical ones [1].

The methods that have been directly applied by the authors for this paper are CART, Association Rules, BBN and Analogy based estimation, with the help of several open source tools that can be found in [13], [18], [19]. For the rest of the methods the results are coming from other studies on this topic. In particular, for FPRA and CART we consulted [11], for OLS the results can be found in [16].

## 4. Results

Before discussing the results it is useful to define the accuracy metrics that will be used in order to compare and evaluate the results of each model. In particular, Mean Magnitude Relative Error will be used, defined as: $MMRE = \frac{100}{n}\sum_{i=1}^{n}\frac{|P_i - E_i|}{P_i}$ where $P_i$ is the actual productivity and $E_i$ is the estimate and $n$ is the number of projects.

Also, $PRED(Y)$ will be used, i.e. the percentage of projects (k) for which the prediction falls within the Y% of the actual value. We employ Pred(25) for both datasets used. In COCOMO81 data set, PRED(20) is also presented in some methods. In the case of interval estimates, relative errors are calculated by considering the mean of the interval.

Also *hitrate* will be used for the estimation of mode[10], i.e. the percentage of projects for which mode has been successfully estimated. Usually the validation of a model is done by removing one data point at the a time from the data set, recalculating the model and estimating the value of the project that was left out (a method known as JackKnifing).

### 4.1. COCOMO81 data set

First, the results for the COCOMO81 data set are presented. The techniques that have been applied are CART, BBN, AR, ABE. The model derived from the COCOMO81 data set by applying FPRA [11] included the following variables: TIME, PCAP, RVOL, with TIME contributing to productivity at 45.3%, PCAP at 18.3%, RVOL at 11.6% and duration at 1,8%. The CART technique (the ordinal scales have been taken from [11]) suggested the structures shown in figures 1,2 :

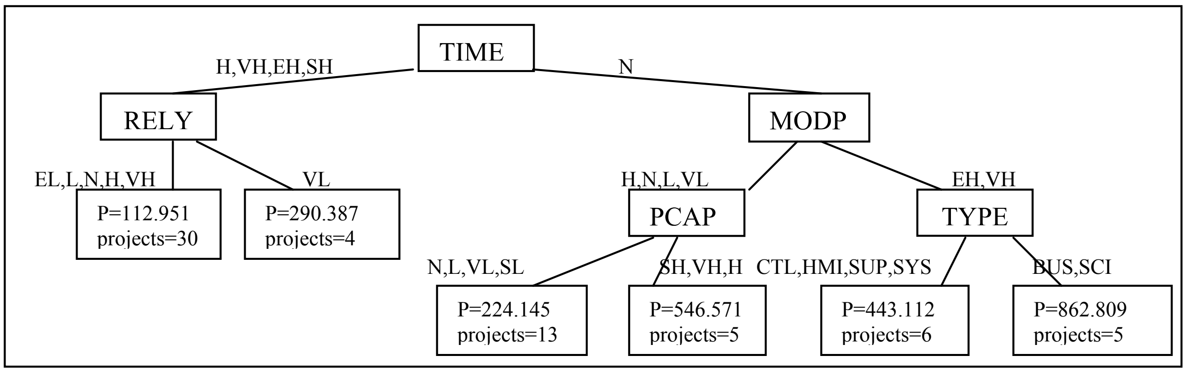

*Figure 1: CART(1) tree for COCOMO81 data set*

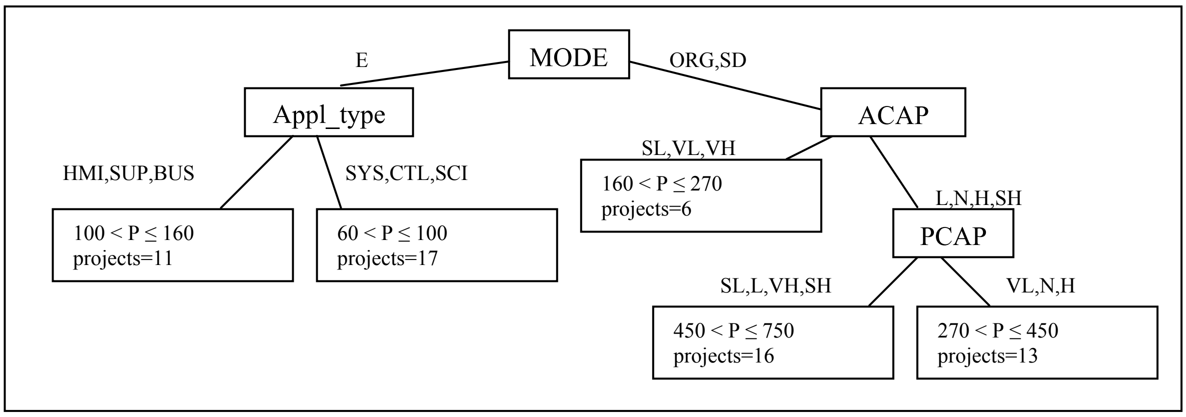

*Figure 2: Another CART(2) tree for COCOMO81 data set*

In the attempt to extract association rules from COCOMO81 data set various problems have been encountered. First of all the suitable, i.e. most representative and powerful, rules should be selected. In addition, the selected rules should be able to provide estimates for all of the 63 projects of COCOMO81 and be as general as possible, so as, given the attributes of a new unknown project, to be able to provide an estimate. For that purpose, rules with high confidence and as high support as possible have been preferred. In order to satisfy the second constraint, rules with few attributes in the rule head were selected so as to avoid over specialization and in cases where the existing preferred rules were not able to provide an estimate, the support or confidence threshold had to be reduced. Eventually, 36 rules were selected with support threshold 4,7% (3 projects), which were used for the evaluation of the model. For instance some rules concerning two categories of productivity are the following in priority order:

| Support | Confidence | Rule Body | | Rule Head |
|---|---|---|---|---|
| 6.3 | 80.0 | [NOM]+[H_VH_EH_SH_RVOL]+[H_VH_TURN] | ==> | [PROD_3] |
| 6.3 | 66.6 | [ACAP_H]+[DATA_N] | ==> | [PROD_4] |

The second rule can be interpreted as follows: When the programmers' analysis capability is high and the database size is nominal then the productivity is likely to be in the fourth category (100<PROD<160). This pattern is presented in 6.3% of the dataset projects (4 projects) and 66.7% of the projects that present ACAP high and DATA nominal fall into the fourth category of productivity (4 out of 6 projects).

In order to compare the AR model with the models of the other methods, we created each time the rules out of 62 projects and tried to estimate the one left. In this case the confidence and the support of each rule were slightly different. For example if we selected to exclude the one project that has ACAP high and DATA nominal, but doesn't belong to the fourth productivity category and also this project satisfies all the criteria of the first rule, then the priority of the two rules changes and the new "unknown" project is misclassified into the fourth category:

| Support | Confidence | Rule Body | Rule Head |
|---|---|---|---|
| 6.4 | 80.0 | [ACAP_H]+[DATA_N] | ==> [PROD_4] |
| 4.8 | 75.0 | [NOM]+[H_VH_EH_SH_RVOL]+[H_VH_TURN] | ==> [PROD_3] |

Bayesian Belief Networks have been also applied on the COCOMO81 data set for the determination of the model structure as well as for the calculation of the conditional probabilities table. In the beginning, a heuristic search algorithm, namely K2 [18] was used to define the conditional independencies for the productivity model.

The above algorithm suggested as parent nodes to productivity node, the programmers capacity and the software development mode, considering that if the values of these two attributes are known, productivity is independent of the other variables. Based on intuition, two other parent nodes were added, the application type of the project and the platform in which the project was developed. So the final BBN that was used for the cost estimation model was the one shown in fig.3.

The calibration phase of the analogy based estimation led to the use of the following attributes in order to make an estimation: database size, programmer's capability, requirements volatility and development mode. The Chebychev distance metric was used with one analogy.

The Jackknifing accuracy of each of the previous methods is presented in table 2. We should point out that there are differences in the accuracy validation methods that were used.

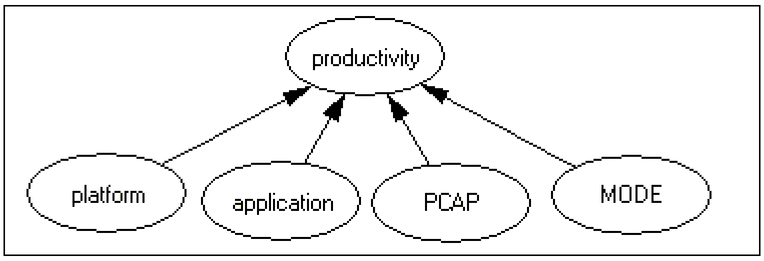

*Figure 3:BBN for the COCOMO81 data set*

In [11] FPRA is reported to have a PRED(20) of 49% and a MMRE of 36% Also CART(1) was found to have a PRED(20) of 29% and MMRE of 62%. The original COCOMO81 model is within 20 percent of the actual effort 68 percent of the time and in that case has a MMRE equal to 18.4%. It is hard to beat COCOMO in its database because it is an ad-hoc model and cannot be validated using JackKnifing and training and test datasets.

*Table 2: Comparison of the models used in the COCOMO81 dataset*

| | CART(2) | AR | BBN | ABE |
|---|---|---|---|---|
| PRED(25)% | 49.2 | 63.4 | 44.8 | 23.8 |
| MMRE% | 105.7 | 29.5 | 59.9 | 49.8 |

## 4.2. Maxwell data set

The techniques that are applied in this data set are CART, AR, BBN and ABE. The procedures that have been used for each of these techniques are the same as the ones described previously for COCOMO81.

OLS results are taken from [16]: two models are described, taking into consideration projects with start date until 1991. The first model includes four variables that explain 79% of the variance in effort. These variables are: interface, ln(size), time and quality requirements. The second model points out ln(size), requirements volatility, and quality requirements as the variables that affect productivity. The results presented here refer to the second one, which is the most effective. CART, AR, BBN and ABE have been applied both to the entire data set and to projects until 1991.

CART suggested the structure presented in figure 4.

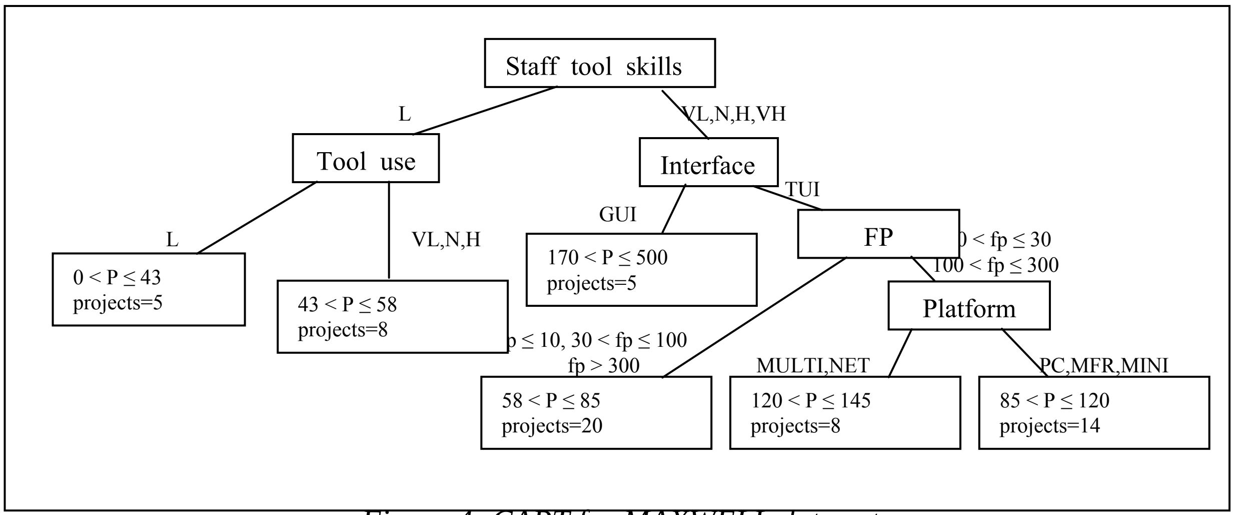

*Figure 4: CART for MAXWELL data set*

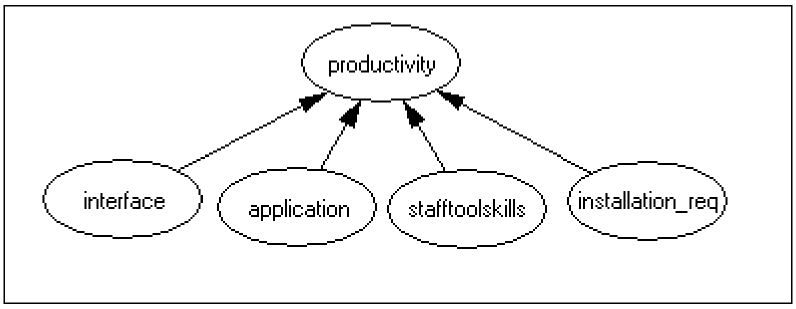

*Figure 5: BBN for MAXWELL data set*

Association rules came up with 36 rules with support threshold 5.0% (3 rules) that are able to estimate all the projects in Maxwell data set. When applying Bayesian Belief Networks the same steps as in the case of COCOMO81 were taken and the suggested network has the structure of fig.5.

Finally the analogy-based method suggested that the more suitable attributes to compare the new project with the historical one's are language, requirements volatility, and size. The distance metric used is the Canberra distance and four analogies were chosen by the calibration phase.

The models described were extracted from the entire data set. The results of these methods by removing one data point at a time are presented, which are more representative. As mentioned above, in order to make the results comparable with OLS, additional models were extracted from projects with starting date 1991 and earlier and the estimations were made for projects with starting dates 1992,1993. The accuracy metrics of each model are presented in table 3. The "*" indicates the evaluation of models estimating projects with starting dates 1992, 1993 , for the other metrics JackKnifing is considered.

*Table 3: Comparison of the models used in MAXWELL dataset*

| | OLS* | A.R* | BBN* | ABE* | CART* | A.R | BBN | ABE | CART |
|---|---|---|---|---|---|---|---|---|---|
| PRED(25) % | 58 | 75 | 45.4 | 41.6 | 41.6 | 60 | 50 | 45 | 53.3 |
| MMRE % | 32 | 23.5 | 45.4 | 56.6 | 119.1 | 42.5 | 54.3 | 42.6 | 46.9 |

## 4.3. Software development Mode estimation.

Our target here is to classify the projects in COCOMO81 database into the three groups defined by Boehm. The user of the COCOMO model has to classify a new project in the appropriate development mode in order to apply the correct COCOMO equation to estimate effort. Failing in this task would produce quite different estimates, since the equations are very sensitive to the coefficients that depend on the development mode. Our target is to propose an automated, systematic approach to the definition of a project's mode. The methods that are applied are CA, DA, CART, AR and BBN.

The results for the first two methods are taken from [23]. Apart from splitting the projects into categories of mode, according to some of their attributes, also confirm the existence of three software development modes.

Clustering uses an algorithm based on Euclidean's inter-object distance starting considering each object at each own cluster. According to the similarities between the different clusters, their number gradually diminishes, resulting in the final clusters. Complete linkage method was used, based on the maximum linkage between the projects. This method finally classifies correctly 34 of the 63 projects with 17 overlaps and 12 misclassifications.

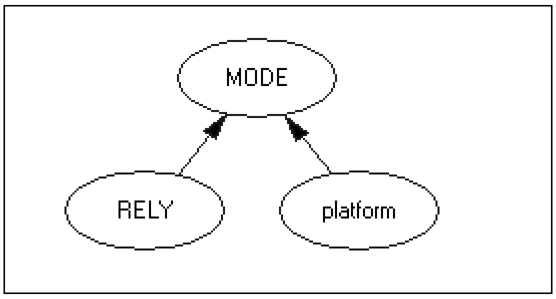

*Figure 6: BBN for MODE*

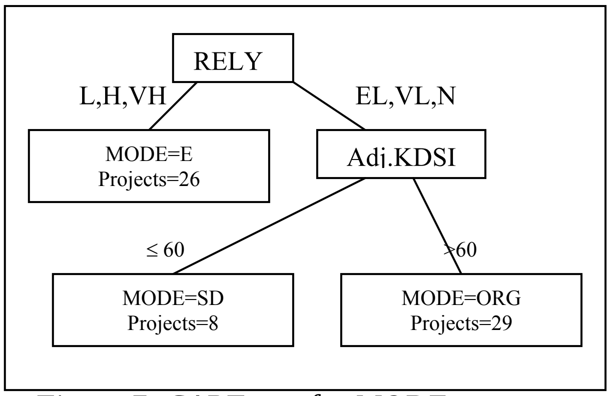

*Figure 7: CART tree for MODE*

Discriminant analysis on the other hand considers a subset of five attributes in order to classify a project into one of the three possible modes. The data base size is used, DATA, the time execution constraint TIME, the use of a virtual machine, VIRT, personnel continuity, CONT, and the use of modern programming practices, MODP.

The structures proposed by CART method and the BBN are presented in figures 6,7.

Eleven association rules were extracted with support threshold 7,9% (5 projects).

The results are presented in table 4:

*Table 4: Comparison of the models used in MODE estimation*

| | CA | DA | CART | AR | BBN |
|---|---|---|---|---|---|
| Hitrate | 54 | 73.2 | 80.9 | 84.1 | 75.8 |

## 5. Conclusions

In this paper, methods producing interval estimates have been identified. Subsequently, the accuracy of these methods has been compared to the accuracy of methods that produce point estimates, by applying them to two public datasets. It must be stressed that these results cannot be used to draw any general conclusions. However, some tentative conclusions may be produced, while sufficient experience from the use of novel techniques has been acquired, indicating certain advantages and drawbacks of each technique.

In COCOMO81 dataset the results show that association rules that produce interval estimates, is a competitive method to FPRA. BBN results are not impressive, but they might be better evaluated in more appropriate estimation conditions, involving larger data sets. On the other hand, CART demonstrate a poor performance. In general, classification techniques are based on combinations of independent variable values. Consequently, their performance suffers when applied on small, unbalanced datasets. CART, due to pruning, tend to reduce the number of eligible intervals, misclassifying the projects that should belong to pruned intervals. In the Maxwell dataset, association rules appear again one of the most accurate methods, along with OLS. There is also a slight improvement in the performance of BBN, since in this case probability combinations cover more projects. This can be explained from the homogeneity of the data set and the fact that the parents can take values from a smaller value range. However, while applying JackKnifing, in 12 cases a particular combination of the parents' values was excluded from the training set, rendering the model incapable of making an estimation. CART still have a poor performance due to the exclusion of some intervals.

In the estimation of software development mode a comparison between the different machine learning methods can be obtained in a pure classification problem. Association Rules and CART have a very good performance classifying correctly the majority of the projects. DA and BBN also produce competitive results, unlike CA that misclassifies a great deal of the projects.

Actually, machine-learning techniques have been compared with model-based techniques. It can be said that a slight superiority of the machine learning techniques has been observed. FPRA and regression models are more reliable at identifying the relevant factors that affect the dependent variable, though their result is often difficult for humans to interpret. On the other hand, models from machine-learning techniques are easily interpreted, allow human interventions, and their results can be used to support expert judgment. However, CART did not produce accurate estimates, while BBN and association rules, although they find powerful and often used patterns that can be confirmed intuitively, suffer from the inability to provide always an estimation. This undesirable situation occurs when the attributes of the new "unknown " projects have not been met in the training data set. In general BBNs can integrate partial knowledge and data concerning a project in the form of 'observed' values of some nodes.

Future research needs to focus on confirming and enriching the results of machine learning methods in larger muti-organizational datasets such as those coming from ISBSG. Of course their accuracy must be compared again to that of model-based techniques. In particular, association rules, seems promising and could be developed further more, as well as BBNs that should be evaluated and judged on more appropriate circumstances, where a large training set will be provided. The choice of estimate intervals is another issue, and further experimentation is needed to understand the implications of the various available approaches.

## 6. References

[1] Angelis, L. and Stamelos, I., “A Simulation Tool for Efficient Analogy Based Cost Estimation”, Empirical Software Engineering 5, 2000, 35-68.

[2] Boehm, B., “Software Engineering Economics”, Prentice-Hall, Englewood Cliffs NJ, 1981.

[3] Boehm, B., Abts, C. and Chulani, S., “Software development cost estimation approaches-a survey”, 2000, Annals of Software Engineering.

[4] Breiman, L., Friedman, J., Oshlen, R. and Stone, C., “Classification and Regression trees”, Wadsworth International Group, Belmont CA, 1984.

[5] Draper, N. and Smith, H., “Applied Regression Analysis”, Wiley, New York, 1981.

[6] Jensen, F., “Bayesian Networks and Decision Graphs”, Springer, Denmark, 2002.

[7] Hand, D., Mannila, H. and Smyth, P., “Principles of Data Mining”, MIT Press, US, 2001.

[8] Heiat, A., “Comparison of artificial neural network and regression models for estimating software development effort”, Information and Software Technology 44, 2002, pp.911-922.

[9] Jeffery, R., Ruhe, M. and Wieczorek, I., “Using Public Domain Metrics to Estimate Software Development Effort”, in IEEE 7th International Software Metrics Symposium proceedings, London UK, 2001.

[10] Jorgensen, M., “An effort prediction interval approach based on the empirical distribution of previous estimation accuracy”, Information and Software Technology 45, 2003, 123-126.

[11] Kitchenham, B., “A procedure for analyzing unbalanced datasets”, IEEE Transactions on Software Engineering 24 (4), 1998, pp278-301.

[12] Kitchenham, B. and Linkman, S., “Estimates, uncertainty and risk”, IEEE Software14 (3), 1997, pp 69-74.

[13] Machine learning software in Java, http://www.cs.waikato.ac.nz/ml/weka/.

[14] Mair, C., Kadoda, G., Lefley, M., Phalp, K., Schofield, C., Shepperd, M. and Webster, S., “An investigation of machine learning based prediction systems”, Journal of Systems and Software 53, 2000, pp. 23–29.

[15] Mair, C and Shepperd, M., “An Investigation of Rule Induction Based Prediction Systems”, in 21st International Conference on Software Engineering, Los Angeles, 1999.

[16] Maxwell, K. Applied Statistics for Software Managers, Prentice-Hall, New Jersey, 2002.

[17] Maxwell, K., Briand L., Emam, K., Surmann, D. and Wieczorek, I. “An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques”, 22ndInternational Conference on Software Engineering ,Limerick, 2000.

[18] Probabilistic Reasoning, Bayesian networks in java, http://www.kddresearch.org/

[19] SERENE Home, http://www.hugin.dk/serene.

[20] Srinivisan, K., Fisher, D., “Machine learning approaches to estimating software development effort”, IEEE Transactions on Software Engineering 21(2), 1995, pp. 126-137.

[21] Stamelos, I., Angelis, L., Dimou, P., and .Sakellaris, E., “On the use of Bayesian belief networks for the prediction of software productivity”, Information and Software Technology 45, 2003, 51-60.

[22] Stamelos, I. and Angelis, L. “Managing uncertainty in project portfolio estimation”, Information and Software Technology 43 (13), 2001, pp 759-768.

[23] Subramanian, G., “An Empirical Examination of Software Development Modes”, Journal of Systems and Software 23 (1), 1993, p.p. 3-7.