Estimating the Development Cost for Intelligent Systems

Stamatia Bibi and Ioannis Stamelos

Department of Informatics, Aristotle University of Thessaloniki, Aristotle University Campus, Thessaloniki 54 124, Greece, sbibi@csd.auth.gr, stamelos@csd.auth.gr; http://sweng.csd.auth.gr

Summary. In this chapter we suggest several estimation techniques for the prediction of the functionality and productivity required to develop an Intelligent System application. The techniques considered are Analogy Based Estimation, Classification trees, Rule Induction and Bayesian Belief Networks. Estimation results of each technique are discussed and several conclusions are drawn regarding the methods, the platform and the languages used for the development of Intelligent Systems. The data set used in the analysis is the publicly available ISBSG data set.

1 Introduction

Software cost estimation is the process of predicting the amount of effort or the productivity required for the completion of a software artifact. Typically software cost estimation involves initially an assessment of the project attributes and then the application of a method for the generation of an estimate. This estimate is used for a number of purposes including budgeting, trade off and risk analysis, project planning and control, and investment analysis. The potential advantages that arise from this procedure can explain the vast amount of literature that exists on this area covering a wide range of methods and practices for software cost estimation.

With the growing importance of Intelligent Systems (IS) in various application domains [10,28,36], many practitioners see the measurement of intelligent applications [22] as a particularly interesting area of research. IS support the decision-making process in organizations and public sectors. As a consequence, enterprises and organizations adopt IS in order to discover hidden information and improve their performance in various areas such as public relationships, customer satisfaction and services innovation.

Successful completion of a software project requires appropriate software development and project management processes. For project managers, cost estimation is one of the crucial steps at the beginning of a new software project [27]. In the field of software cost estimation there is a trend to calibrate

S. Bibi and I. Stamelos: *Estimating the Development Cost for Intelligent Systems*, Studies in Computational Intelligence (SCI) **104**, 25–45 (2008)
www.springerlink.com

estimation models with the needs of particular applications, e.g. Web applications [32]. IS development differs substantially from traditional software development because in such projects there are additional constraints due to the different development approach, the cost of computation, the hardware demands and the different programming languages, platforms and algorithms used. Additionally it is known that cost estimation models are sensitive to the data used for their construction. Estimation models present increased accuracy when the projects used to predict future projects are developed in similar environment and under the same constraints with the projects under estimation. One method has been proposed for estimating the cost of IS development [26] based on COCOMO II model [7] that demands a wealth of data regarding the projects variables. This model is very analytical as it requires the values of 23 cost variables and then calculates a parametric cost equation. Considering that often in the initialization of a project there is no available knowledge regarding the factors discussed in [26] we suggest in this chapter the exploitation of knowledge of previously completed projects. A low cost method for productivity analysis is learning from past projects to predict future ones. In this study we extract knowledge from Isbsg data set [15] regarding two types of IS, namely Decisions Support Systems (DSS) [35] and Knowledge Based Systems (KBS) [30]. Using this knowledge one should be able to: (a) Identify a productivity interval for the development of IS projects, (b) Discover possible influence of project attribute values on productivity (the utilization of certain tools, languages, databases used may increase or decrease productivity).

This chapter provides an in depth analysis of the productivity required to complete an IS application. We apply several machine learning techniques, namely Analogy based estimation (ABE), Classification trees, Rule induction and Bayesian analysis in order to estimate size, productivity and extract useful knowledge regarding the development of IS projects. It should be mentioned that the applications studied were implemented with non AI languages. To our knowledge there is no publicly available data set with specialized data on the development of IS projects, such as language, mining algorithm, computational costs and database size. As a consequence we limit our analysis on a sample of IS projects coming from a large multi-organizational data set containing several other types of projects as a well. Due to the different type of projects that appear in the data set, the data considered in the study are general and they do not provide more specific knowledge involving issues, mentioned earlier concerning purely IS development.

2 Related Work

There are many studies in literature regarding the implementation of data mining techniques [9,12,22] for the development of IS projects. Various methods, techniques and algorithms for implementing IS have been a research issue for several studies [14, 39]. In these studies, certain methods for extracting

knowledge from data bases have been analyzed, implemented and evaluated in terms of computational time, predictive accuracy, descriptive accuracy and misclassification costs. Comparative studies for different kind of development techniques and algorithms address issues regarding the mining efficiency of the methods but they do not deal with issues involving the time needed for the implementation of the applications, the tools and the productivity required for such projects when implemented with different kind of data mining techniques.

There are only few studies in literature dealing with software engineering issues in the development of IS. In [13] a software engineering environment for developing IS is suggested, that enables the cooperation of various technologies for that purpose. The reusability of database components utilized in the development of IS is discussed in [38]. Organizing IS development process with Crisp DM methodology is suggested in [9] that provides a framework that contains the corresponding phases of a project, their respective tasks, and relationships between these tasks. These studies propose methods for simplifying the development of IS projects. Another approach to simplify the development of IS projects is to effectively schedule the whole software engineering process of IS systems. One part of scheduling involves the estimation of costs. An estimation of the time needed to complete an IS project and of the way the selection of software development language, platform or methodology affects this time could provide useful knowledge especially in the initialization of an IS project. One method has been proposed for estimating the cost of IS development [26] based on COCOMO II model [7] that utilizes data regarding the projects variables. This model is very analytical as it requires the values of 23 cost variables, such as number of tables and tuples, type of data sources, type and number of models extracted, and then calculates a parametric cost equation. To our knowledge no other study is found in literature regarding specifically IS development costs.

On the other hand there is a vast literature in the area of software cost estimation (SCE). An extensive review of software cost estimation methods can be found in [6]. Expert judgment [19–21, 29], Model based techniques [1, 7, 16, 17, 33] and Learning oriented techniques [3, 4, 24, 25] are the main categories of software cost estimation methods. The wide application of the techniques in several development environments for the estimation of software costs has indicated that the combination of methods provides improved results. Cost estimation techniques can present several advantages and disadvantages in terms of accuracy, comprehensibility, causality, applicability and risk management. The combination of methods can provide a more complete estimation framework than one method alone. In this study taking into consideration the findings of [23] we suggest the combination of several learning oriented methods in order to improve the estimation and descriptive efficiency of predictive models for an IS project.

Learning oriented techniques, are recently applied in SCE and use data from past historical projects to estimate and control the new ones. Learning oriented techniques are very useful when a company stores its own data

and wants to utilize this knowledge in order to guide and control the development of future projects. Literature has indicated several advantages and disadvantages of these methods when applied alone.

ABE is a widely used method to identify historical projects, with attributes similar to the project under estimation, that will provide an estimation [2]. ABE is a very accurate method when the historical data set includes similar projects to the one under estimation but can be very inaccurate in other situations.

Neural networks [25] can be very accurate but difficult to apply and interpret, while rule induction and decision trees [24] are easy to understand but of lower accuracy. Association rules as a descriptive modeling method [4, 5] can address causality, are easily interpretable but they cannot always provide an estimate. Bayesian Networks [3, 34] identify the underlying relationships among all project attributes, can be easily combined with expert judgment but usually they demand large data sets in terms of accuracy.

In this study we experiment with several learning methods regarding the SCE of IS in order to provide a complete estimation framework. In particular we apply:

- Rule induction and Classification And Regression Trees (CART): to estimate size and productivity with probability values that assess uncertainty.
- ABE: to identify important attributes and distance metrics for productivity and size estimation.
- Association rules: to explore patterns that associate various project attributes.
- Bayesian Belief Networks (BBN): to identify the dependencies among the project attributes.

3 Estimation Methods

In this section we present analytically the methods used to analyze and model data regarding the implementation of IS systems.

3.1 Classification and Regression Trees

CART is a widely used statistical procedure for producing classification and regression models with a tree-based structure in predictive modeling [8]. The CART 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 one of several different splitting criteria, all producing 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 tree is produced and specific

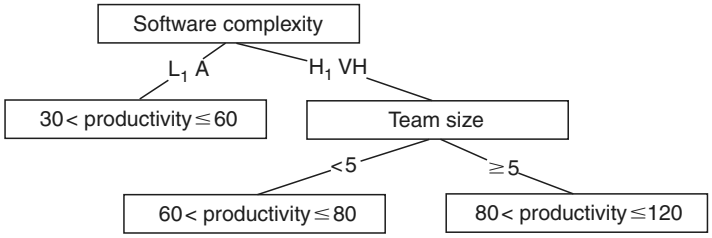

Fig. 1. A CART for software productivity estimation

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.

CART are able to classify not only all the projects in the training data set, but unknown projects from a wider group of projects of which the training projects is presumed to provide a representative example. In our study a serial algorithm is applied. The decision tree classifier consists of two phases: a *growth* phase and a *prune* phase. In the growth phase, the tree is built by recursively partitioning the data until each partition is either 'pure' or sufficiently small. The form of the split used to partition the data depends on the type of the attribute used in the split. Only binary splits are considered. Once the tree is fully grown, it is 'pruned' in the second phase to generalize the tree by removing dependence on statistical noise. Pruning a branch of a tree consists of deleting all descendants of the branch except from the root node. The pruning algorithm used is based on the Minimum Description Length principle [31].

A simple CART for software productivity estimation is presented in Fig. 1. In order to estimate the productivity category of a new project, starting from the root node, the right branch is selected according to the project's attributes at each level, moving down until a terminal node is reached. In particular the tree of Fig. 1 can be interpreted as following:

If software complexity is average or low productivity value is estimated to be greater than $30\,\mathrm{fp\ h^{-1}}$ and equal or less than $60\,\mathrm{fp\ h^{-1}}$. If software complexity is high or very high and the team size is less than five people productivity value is estimated from 60 to $80\,\mathrm{fp\ h^{-1}}$. Otherwise if the team size includes five or more people productivity will be between 80 and $120\,\mathrm{fp\ h^{-1}}$.

We use classification trees for the analysis because they handle variables with nominal and ordinal scales, they provide results easy to interpret, and they can handle data sets with few cases.

3.2 Association Rules

Association rules [14] are among the most popular representations of local pattern recognition. They are a form of descriptive modeling and have as a target to describe the data and their underlying relationships with a set

of rules that jointly define the target variables [39]. 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-occurrence of certain events in a database.

Each rule consists of two parts. The left part is the Rule Body (antecedent) and is the necessary condition in order to validate the right part, Rule Head (consequent). Each rule states that if the rule body is true then the rule head is also true with probability p. It is obvious that the A.R are Boolean propositions with true or false values. In the rule head, any Boolean expression can be used, but usually conjunction is preferred for simplicity purposes.

Given a set of observations over attributes $\mathrm{A}_1,\ \mathrm{A}_2, \ldots, \mathrm{A}_\mathrm{n}$ in a data set D a simple association rule has the following form:

$$(\mathrm{A}_1 = \mathrm{X} \wedge \mathrm{A}_2 = \mathrm{Y}) \Rightarrow \mathrm{A}_3 = \mathrm{Z}$$
$$\text{confidence} = p(A_3 = Z | A_1 = X, A_2 = Y),\ \text{support} = \text{freq}\ (X \cup Y \cup Z, D)$$

This rule is interpreted as following: when the attribute A_1 has the value X and attribute A_2 has the value Y then there is a probability p (confidence) that attribute A_3 has the value Z. For this rule two major statistics are computed, confidence and support values. Confidence is the probability p defined as the percentage of the records containing X, Y and Z with regard to the overall number of records containing X and Y only. The other statistic is support value which is a measure that expresses the frequency of the rule and is the ratio between the number of records that present X, Y and Z to the total number of records in the dataset.

AR mining is a two stage process. The first stage involves the identification of all frequent set of attributes contained in the given data set. A set of attributes is frequent if its associated support exceeds a certain support threshold defined by the user. The second stage is generating all pertinent ARs from these itemsets. An AR is pertinent if its associated confidence exceeds a certain confidence threshold specified by the user.

A simple example of an Association Rule coming from software data sets is presented in Table 1.

The association rule of Table 1 states that 80% of the cases that involve the development of a *DSS* project *Inhouse* also use a *Methodology*. If the data set used contains 39 instances then 12 out of 39 projects (support = 12/39 = 30.8%) of the data set satisfy the particular rule and 12 out of 14 projects (confidence = 12/14 = 80%) that involve the development of a *DSS* project *Inhouse* use a *Methodology*.

Table 1. Association rule describing whether methodology is used or not

Support	Confidence	Rule Body	Rule Head
30.8	80.0	DSS + DevelopedInhouse	Methodology_Yes

AR was selected because it is a method for descriptive modeling that identifies specific relationships often ignored by predictive models. Frequent and pertinent relationships among the project attributes and the development productivity are discovered. The knowledge extracted from the data set is in the form of *if-then* rules that are easily understood, giving the chance to experts to analyze, validate and combine known facts about the domain.

3.3 Bayesian Belief Networks

Bayesian Belief Networks are Directed Acyclic Graphs (DAGs), which are causal networks that consist of a set of nodes and a set of directed links between them, in a way that they do not form a cycle [18]. Each node represents a random variable that can take discrete or continuous finite, mutually exclusive 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 node) 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 some times 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 [11]:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{1}$$

For each node A with parents B1, B2, ..., Bn there is attached an NxM Node Probability Table (NPT), where N is the number of node states and M is the product of its cause-nodes states. In this table, each column represents a conditional probability distribution and its values sum up to 1.

A simple BBN estimating software effort is the one presented in Fig. 2. Attached to the node of effort there is a node probability table that provides possible values of the effort needed to develop the intelligent system, based on the combination of values that the programming language and the development platform nodes take.

In this study the extraction of BBN is achieved with an award winning algorithm [11]. BBN were selected because of their ability to represent all the dependencies among the variables participating in the study.

3.4 Rule Induction

RI, as CART method, is a particular aspect of inductive learning. Inductive learning is then the process of acquiring general concepts from specific

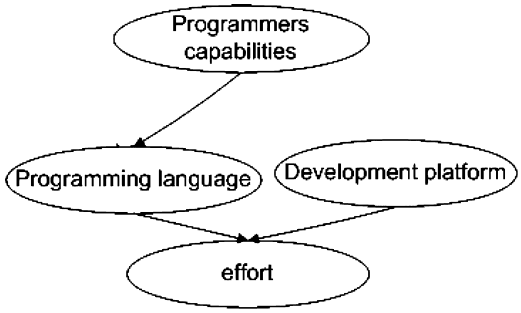

Fig. 2. A BBN for software effort estimation

Table 2. Rule induction for software productivity estimation

If language used = java and development type = enhancement then $40 <$ productivity ≤ 60 total no of projects = 10 wrong estimates = 2

examples. By analyzing many examples, it may be possible to derive a general concept that defines the production conditions.

Rule induction is an alternative approach to CART that takes each class separately, and try to cover all examples in that class, at the same time excluding examples not in the class. This is a so called, *covering approach*, because at each stage a rule is determined that covers some of the examples. Covering algorithms operate by adding tests to the rule that is under construction, always trying to create a rule with maximum accuracy. Whereas CART algorithm chooses an attribute to maximize the separation between the classes (using information gain criterion), the covering algorithm chooses an attribute-value pair to maximize the probability of the desired classification.

In this chapter, we apply the PART algorithm that is based on extracting partial decision trees. The method also handles missing values by assigning an instance with missing value to each of the tree branches with a weight proportional to the number of training instances descending that branch, normalized by the total number of training instances with known values. In this study for the induction of classification trees and rules we utilize the Weka machine learning library [37].

A simple rule coming from the domain of software cost estimation will have as Rule Body certain software project attribute values and as a Rule Head a productivity (or cost, or effort) value. A simple example of a rule is presented in Table 2.

This rule is interpreted as following: If the language that will be used for the development of new project is java and the development type of the project is enhancement then there is (10-2)/10 = 80% (confidence value) probability that the productivity value of the project will be between 40 and 60 lines of code per hour.

One advantage of inductive learning over other machine learning methods is that the rules are transparent and therefore can be read and understood. Proponents of RI argue that this helps the estimator understand the predictions made by systems of this type.

3.5 Analogy Based Estimation

ABE is essentially a form of case based reasoning. The main aspect of the method is the utilization of historical information from completed projects with known size, effort or productivity. The most appropriate attributes are selected according to which the new project is compared with the old ones in the historical dataset. The attribute values are standardized (between 0 and 1) so that they have the same degree of influence and the method is immune to the choice of units.

The next step is to calculate how much the new project differs from the other projects in the available database. This can be done by using a 'distance' metric between two projects, based on the values of the selected attributes for these projects. The most known such distance metric is the *Euclidean* or *straight-line distance* which has a straightforward geometrical meaning as the distance of two points in the k-dimensional Euclidean space:

$$d_{new,i} = \left\{ \sum_{j=1}^{k} (Y_j - X_{ij})^2 \right\}^{1/2}, \quad i = 1, 2, ..., n \tag{2}$$

Other possible distance metrics are the Minkowski distance, the Canberra distance, the Czekanowski coefficient and the Chebychev or 'Maximum' distance (see [2] for definitions).

In conclusion the estimation of the productivity using analogies is based on the completed projects that are similar to the new one. The user of the method has to calculate the distances of the new project from all the database projects and identify few 'neighbour' projects, i.e. those with relatively small distance value. The estimation of the productivity is eventually obtained by some combination of the productivities/efforts of the neighbor projects. Typically, the statistic used is the mean (weighted or simple) or the median of these effort values. The calibration of the analogy-based method requires the detection of the best configuration of the available method options. The options that may be adjusted are:

(a) The distance metric by which the projects of the database will be sorted according to their similarity to the one under estimation (e.g. Euclidean distance, Manhattan distance)
(b) The number of closest projects (analogies)
(c) The set of attributes for judging analogy

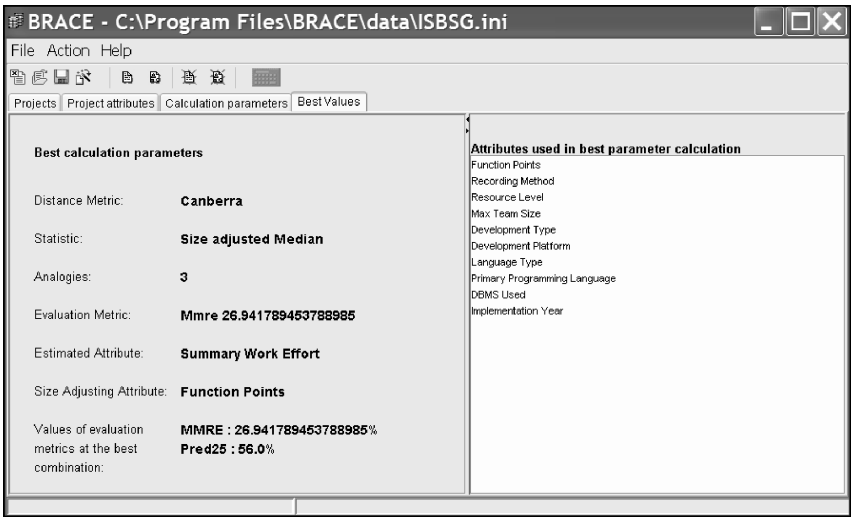

Fig. 3. Best calculation parameters for effort estimation

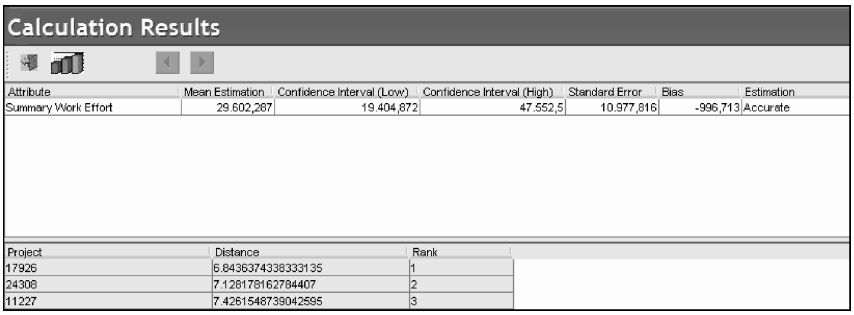

Fig. 4. Estimation of the effort value of a new project based on the best parameters of Fig. 3

In this study we apply and calibrate ABE for size and productivity prediction with the help of Brace tool [2]. A simple example that demonstrates the calibration of the method and the selection of best parameters is presented in Fig. 3. In this example the most appropriate distance metric to be used for the calculation of distances among similar projects for the particular data set is Canberra distance, the size adjusted median of the predicted interval is selected as a point estimate and the number of analogies is set to three. The Mean Magnitude Relative Error (MMRE) is 26.9% and the prediction within 25% of actual value is 56%. The size adjusting attribute is function points, a choice made by the user.

The results of the application of ABE in the estimation are presented in Fig. 4. Similar projects from the training set to the project under estimation are presented in descending order. The effort values of the first three similar

projects are used for the calculation of an effort interval of the new project. Apart from a low and a high effort value, a point estimate is calculated along with a bias value.

4 Data Description and Preparation

The data set used for this analysis is the widely known ISBSG7 data set, a publicly available multi-organizational data set. ISBSG is a repository maintained by the International Software Benchmarking Standards Group to help developers with project estimation and benchmarking. ISBSG data repository release 7 [15] contains 1,239 projects that cover the software development industry from 1989 to 2001. The data set contains over 50 fields involving the projects origin, age, context, the type of the product and the project and the development environment, the methods and tools utilised. We selected the particular data set because it contained 39 projects involving the development of Decision Support Systems and Knowledge Based Systems that could form a sample data set for our analysis. All variables were taken into account in the study regardless of the missing values observed. The variables are presented in Tables 3 and 4.

Table 3. Variables in ISBSG data set

Attribute	Values
DT: Development type	Enhancement, NewDevelopment, Re-development
LT: Language type	3GL,4GL
PPL: Programming language	4GL, ACCESS, C, COBOL, CSP, JAVA, NATURAL, OTHER 4GL, VISUAL BASIC
OT: Organization type	Banking, Communication, Community Services, Electricity & Gas & Water, Insurance, Manufacturing, Professional Services, Public Administration, Computers
AT: Application type	DSS, Knowledge based
UCT: Use of case tools	Yes, No
PC: Package customization	Yes, No
BAT: Business area type	Banking, Engineering, Financial, Insurance, Manufacturing, Personnel, Sales & Marketing, Social Services, Hardware/Software
DBMS: Database system	ADABAS, ACCESS, IMS, SQL, ORACLE, RDB, M204
Used Methodology: whether methodology was used to build the software	Yes, No
DP: Development platform	MF, MR, PC
HMA: How methodology acquired	Developed/purchased, Developed Inhouse, Purchased

Table 4. Descriptive statistics for the attributes of IS systems

Variable	Min	Max	Mean	St. deviation
Function points	18	1,474	413.846	403.827
Effort	220	16,000	3,093.154	3,874.975
Productivity	0.024	1.147	0.28	0.306
Max team size	0	18	4.235	4.562
Date	1993	2000	1995.703	1.869

Table 5. Descriptive statistics for the attributes of MIS systems

Variable	Min	Max	Mean	St. deviation
Function points	25	17,518	557.276	1,097.789
Effort	17	99,088	6,412.665	11,107.226
Productivity	0.01	5.35	0.204	0.402
Max team size	0	468	8.184	36.071
Date	1989	2001	1996.201	2.871

In the data set there are two variables indicative of the projects cost, i.e. *size* measured in function points and *effort* measured in hours. Function Points (FP) is a standard metric for the relative size and complexity of a software system, originally developed by Alan Albrecht of IBM [1]. The size is determined by identifying the components of the system as seen by the end-user: the inputs, outputs, inquiries, interfaces to other systems, and logical internal files. The components are classified as simple, average, or complex. All of these attributes are then assigned a score by the analyst and the total is expressed in Unadjusted FPs (UFPs). Complexity factors described by 14 general systems characteristics, such as reusability, performance, and complexity of processing can be used to weight the UFP. Factors are also weighted on a scale of 0–5. The result of these computations is a number that correlates to system functional size. We decided to estimate the productivity value of the projects as well defined as the ratio between the function points of a project and the effort measured in hours.

In the same data set there are data regarding the implementation of other application type projects. It is interesting to compare such data with those of the IS data set. In Table 5, descriptive statistics regarding the function points, effort, productivity, max team size and date values of Management Information Systems are presented. Maximum values of function points between the two application types are quite different but the min and mean values are close for both types. Regarding the effort we can conclude that in the particular data set a simple MIS project requires less effort than an IS application but in total the mean effort required for a MIS project is more than the mean effort required for an IS application. Mean productivity values for both application types are quite similar although MIS projects require larger development teams, almost double in size, than IS projects. Finally, as expected MIS

Table 6. Productivity and function points intervals considered in the study

Variable	Low	Average	High	Very high
Productivity	0.024–0.09	0.09–0.168	0.168–0.6075	0.6075–1.147
Function points	18–157	157–254.5	254.5–916	914–474

application had earlier implementation dates than IS applications. We must mention that this comparison is limited to the specific information included in ISBSG data set. Also the observations regarding IS projects are only 39 and the ones regarding MIS projects are 352.

In order to apply the estimation methods we had to discretize [5] the field of productivity and assign productivity values into intervals. Discretization of productivity raised two issues: the number of intervals and the discretization method. Due to the few projects contained in the data set we intuitively decided to define four intervals of productivity each one related to approximately equal number of projects. The productivity (measured in function points per hour), and function point intervals are presented in Table 6.

Especially for Bayesian analysis data preparation was slightly different as the method cannot handle missing values. We had to dismiss records with missing values in several variables so as to leave as many projects and variables as possible in the analysis. The final set of projects was 20, the number of productivity intervals 3 0.02–0.1, 0.1–0.4, 0.4–1.15 and the variables that participated are the ones in Tables 4 and 5 apart from *dbms, hma* and *bat* that presented many missing values. Also for variables like organization type and language that presented many different values we grouped together values that had the same effect on productivity.

5 Results and Discussion Regarding Functional Size

In this section the results regarding the estimation of the *size* of IS applications will be presented in the form of classification tree and rules. The classification tree suggested is presented in Fig. 5. It seems that the *business area type* in which the system will be applied and the *data base* used can affect the *size* of the application. Applications implemented for financial and engineering business area types are relatively large in size. For the rest of the applications, the ones that used Adabas, Access, Oracle and Relational Data Bases as a data base management system presented average size while the rest could be considered as small applications. The classification and regression tree of figure can classify correctly 53.85% of the data set.

PART decision rules regarding the estimation of size are presented in Table 7. The rules comply with the CART of Fig. 5 and additionally provide two hints: Applications developed in *Access* with the help of *Case tools* present average size. *Decision support systems* also are of average size. The

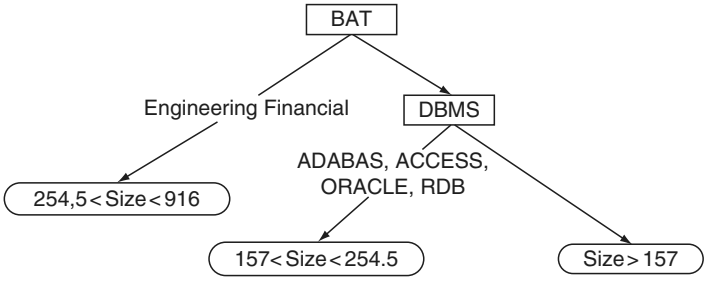

Fig. 5. CART for the estimation of functionality of an IS

Table 7. Rule set for the estimation of function points

Rules
BAT = Financial: 254.5 < size < 916 (6.78/2.26)
Dbms = Access AND CaseTool = Yes: 157 < size < 254.5 (4.33/1.41)
BAT = Engineering: 254.5 < size < 916 (3.52/1.23)
AT = DSS: > 157 < 254.5 (14.33/8.12)
else: size < 157 (10.04/5.42)

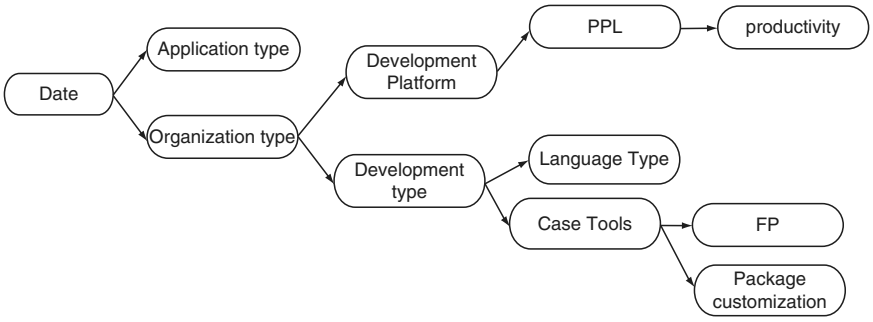

Fig. 6. BBN for predicting the productivity of IS

rule set of Table 7 is also able to classify correctly 53.85% of the projects into a size interval.

ABE pointed out as the best configuration to use the Euclidean distance metric with one analogy considering as best point estimate the mean value of the size interval. The best attribute subset for providing an estimation is considered the *development type*, the *development platform*, the *language type* the *data base* used and the *use of case tools*.

Bayesian analysis presented in Fig. 6 indicated the *use of case tools* as the only variable that directly affects software size. Also the number of function points is conditionally independent from *organization type* and *development type* when there is knowledge regarding the use of case tools. Since the values of case tools (no, yes) are uniformly distributed the only conclusion we can

Table 8. Node probability table for the estimation of function points

Use of case tools size	No	Yes
Size < 225	0.244	0.445
225–800	0.378	0.311
Size > 800	0.378	0.244

reach regarding the number of function points is that when there is absence of case tools the functionality that has to be implemented is increased which is a strange finding. The use of case tools enables the implementation of the same projects with less functionality implemented by the developers, a fact that reduces complexity and therefore the possibility of fault existence. The node probability table for the estimation of the number of function points is presented in Table 8. The last column of Table 8 can be interpreted as following: The functional size of an IS project when case tools are used to aid the development is 44.5% possible to be less than 225 fp, 31.1% is likely to be between 225 and 800 fp and there is only 24.4% possibility to be more than 800 fp.

6 Results and Discussion Regarding Productivity

This section contains the details of a formal analysis of the data set, comprising a classification tree analysis, rule induction and Bayesian analysis.

The tree extracted from the data is presented in Fig. 7. In the tree, the variables *Business Area Type, Function Points* and *Primary Programming Language* are able to classify correctly 69.23% of the projects. Low productivity values are observed in *Banking, Insurance, Personnel* and *Social Services* applications. High productivity value is presented in large applications developed mainly with *Cobol, Access, C* and other not mentioned *4GL* languages. The intermediate classes of productivity are obtained in relatively small application (<800 fp) or in applications developed with *Natural, Java, SQL* and *Csp*.

Table 9 presents the rule extracted with PART algorithm. The first number in brackets is the total number of projects that comply with the rule body while the second number is the number of the projects being misclassified by the rule in the subset of the examples the rule is applied.

The rules are applied in the order discovered and are able to classify correctly 71.8% of the projects participating in the study. Important attributes that appear often in the rules are *Organization and Application type. Function points* also seam to be a critical estimation variable. Low productivity values are observed in the development of projects designated to *Insurance* and *Public administration* organizations. Projects designed for *Energy* organizations are mainly assigned to high productivity values while projects applied

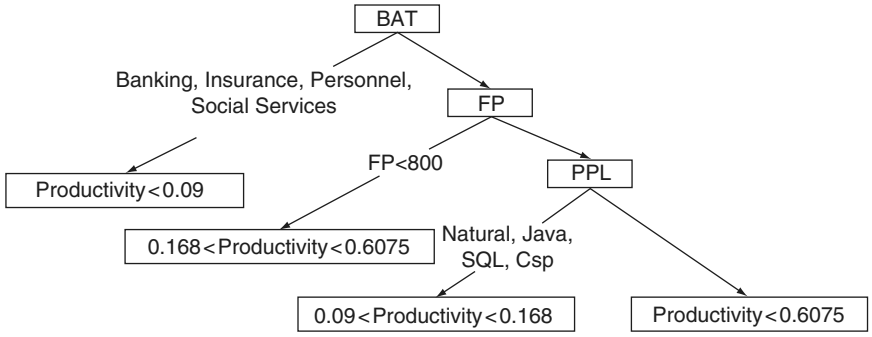

Fig. 7. Classification tree for predicting the productivity of IS

Table 9. List of rules for predicting the productivity of IS

Rules
OT = Communication AND FP $\leq$ 937: 0.168 < productivity < 0.6075 (10.0/4.33)
OT = Insurance: productivity < 0.09 (7.52/3.52)
AT = DSS AND OT = PublicAdministration: productivity < 0.09 (5.44/1.44)
OT = Energy: productivity > 0.6075 (3.21/0.21)
AT = DSS AND FP <= 722: 0.168 < productivity < 0.6075 (6.84/2.28)
AT = DSS: 0.09 < productivity < 0.168 (3.0/0)
else: productivity > 0.6075 (3.0/0)

Table 10. Node probability table for the estimation of the productivity of IS

ppl productivity	Access	Other 4GL, cobol, natural	sql,c,csp,java
Productivity < 0.1	0.185185	0.547619	0.138889
0.1–0.4	0.185185	0.261905	0.638889
Productivity > 0.4	0.62963	0.190476	0.222222

to *Communication* organizations tend to fall in the intermediate classes of productivity.

The results observed from the Bayesian analysis, presented in Fig. 6 are also very interesting. A strange absence of dependency in the BBN of Fig. 6 is the one between FP and productivity. This may be explained due to the small training set, the few intervals defined in each variable and the multi organizational data set that prevented the homogeneity of the data. Interesting dependency is the one between *programming language* used and *productivity* value. *Language type* can classify correctly 80% of the projects into a productivity interval with the help of Table 10.

Interesting assumptions regarding the Bayesian analysis can be summarized as following:

- Applications developed in *mainframe platforms* had mostly as destination *public administration organizations*, were programmed mainly with *4GL* languages and presented low *productivity* values. On the other hand applications developed in *PC* was designated mainly for *communication* and *electricity, gas, water* and *energy* providers, were developed in *Access* and *SQL* and had high *productivity* values. Intermediate productivity values were observed in projects developed in *SQL*.
- Regarding the *size* of the projects measured in Function points it appears that it is negatively correlated with *the use of case tools*.
- It seems that productivity values remain equally distributed into intervals and unaffected by the time. Someone would expect that in most recent projects productivity would be higher but this is not justified by the analysis. It seems that IT personnel becomes more experienced as time passes but still they have to practice in a variety of new technologies.

The results of ABE pointed out that the best configuration to use is Canberra distance metric [2] with one analogy, considering as a point estimate the mean value of the estimated interval. Canberra distance metric is defined as following:

$$d_{new,i} = \left\{ \sum_{j=1}^{k} \frac{|Y_i - X_{ij}|}{|Y_i + X_{ij}|} \right\} \quad i = 1, 2, ..., n \tag{3}$$

The best attribute subset used for estimation is considered the *development platform* and the *development type*.

7 Identification of Associations Between Attributes

In this section we will identify several useful patterns that exist in the IS project data. Frequent and pertinent relationships among the project attributes are discovered and interpreted intuitively. For this purpose AR will be utilized as a method for descriptive modeling. With rules as those presented in Table 11 it is possible to identify often used techniques and methodologies, trends in the development of IS and their evolution over time and finally patterns that a practitioner should avoid.

Certain conclusions that we can reach from the rules are summarized in the following list:

- *Methodology* is used mainly in *Decision Support Systems* projects, developed inhouse, unlike *Knowledge Based* systems where no methodology is mostly used. The use of case tools is observed mainly in applications destined for *personnel business area type*.

Table 11. Association rules detected for attribute estimation

Support	Confidence	Rule Body	Rule Head
30.8	80.0	DSS+DevelopedInhouse	Methodology_Yes
43.6	100.0	4GL	DSS
25.6	90.9	MTS < 2.5	DSS
28.2	91.7	Communication	CaseTool_No
23.1	75.0	Communication	Methodology_No
25.6	83.3	Communication	KB
15.4	100.0	CaseTool_Yes+Methodology_Yes	DevelopedInhouse
10.3	100.0	BAT_Personnel	CaseTool_Yes
17.9	77.8	CaseTool_No+Methodology_No	KB
25.6	90.9	KB	CaseTool_No

- From 1993 to 2000 *communication* organizations demanded mainly knowledge based systems.
- The *4GL*, non AI languages used for the *New development* projects is mainly *Access* and *SQL*.
- The absence of *case tools* and methodology is mainly observed in the development of *knowledge based* systems.
- Projects developed by small *teams* < 2.5 and in *4GL* languages are usually decision support systems.

8 Conclusions and Future Work

In this chapter we examined four machine learning techniques in estimating the productivity for the development of IS. All of the methods were able to classify the majority of projects in the correct productivity interval and provide several assumptions regarding the development environment. CART are able to provide a simple, easily understood estimation framework, rules can provide a more informed estimation and BBN support thorough analysis of the dependencies and independencies among the projects attribute and productivity. Possible advantages of these methods in the estimation of IS systems are:

- The knowledge extracted from the application of the methods is relevant to the data collected. Therefore each organization can specify the data of interest, collect the appropriate information and estimate the target fields. Therefore even if in our study the data analyzed were of multi organizational nature a company could calibrate and adjust the models to its needs collecting the appropriate data.
- The output of the models is easily understood. The models support visual representation of the results, have a strong mathematical background and can be extracted with the use of several tools available free from the internet [9, 37].

It is evident that for more detailed analysis of these issues further research has to be done applying and comparing the previously mentioned methods on data involving IS development with pure AI techniques as well. Data pertaining to the development of IS could involve the mining algorithm used for the implementation of them, the tools, languages and platforms utilized specifically for the development of such projects, the requirements in storage and memory during the application of the systems. When a large cost data base for AI development projects is available, more thorough analysis will be possible and the pros and cons of each development tool, technique or algorithm will be easier to observe. In addition sensitivity analysis will show the robustness of the methods. Finally, collecting such data over long periods will permit trend analysis, providing a picture of AI development evolution over time.

References

1. Albrecht, A.: Measuring Application Development Productivity. IBM Application Development Symposium, Monterey California, October (1979) 83–92
2. Angelis, L., Stamelos, I.: A Simulation Tool for Efficient Analogy Based Cost Estimation. Empirical Software Engineering, 5(1) (2000) 35–68
3. Bibi, S., Stamelos, I., Angelis, L.: Bayesian Belief Networks as a Software Productivity Estimation Tool. Proc. 1st Balkan Conference in Informatics, Thessaloniki Greece (2003) 585–596
4. Bibi, S., Stamelos, I., Angelis, L.: Software Productivity Estimation Based on Association Rules. Proc. of 11th European Software Process Improvement Conference, Trondheim Norway (2004) 13A1
5. Bibi, S., Stamelos, I., Angelis, L.: Software Cost Prediction with Predefined Interval Estimates. Proc. of 1st Software Measurement European Forum, Rome Italy (2004) 237–246
6. Boehm, B.: Software Engineering Economics. Prentice-Hall, Englewood Cliffs NJ (1981)
7. Boehm, B., Abts, A., Brown, W., Chulani, S., Clark, B., Horowitz, E., Madachy, R., Reifer, D., Steece, B.: Software Cost Estimation with COCOMO II. Prentice Hall, Upper Saddle River NJ USA (2000)
8. Breiman, L., Friedman, J., Oshlen, R., Stone, C.: Classification and Regression Trees. Wadsworth International Group, Pacific Grove CA (1984)
9. Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth. R.: CRISP-DM 1.0 steb-by-step data mining guide. http://www.crisp-dm.org/Process/index.htm CRISP-DM (2000)
10. Chen, S., Liao, C.: Agent-Based Computational Modeling of the Stock Price Volume Relation. Information Sciences, 170(1) (2005) 75–100
11. Cheng, J., Greiner, R., Kelly, J., Bell, DA, Liu, W.: Learning Bayesian Networks from Data: An Information-Theory Based Approach. The Artificial Intelligence Journal, 137 (2002) 43–90
12. Domingos, P.: Metacost: A General Method for Making Classifiers Cost-Sensitive. Proc. Knowledge Discovery in Databases (1999) 155–164
13. Erman, L., Lark, J., Hayes-Roth, F.: ABE: An Environment for Engineering Intelligent Systems. IEEE Transactions on Software Engineering, 14(12) (1988) 1758–1770

14. Hand, D., Mannila, H., Smyth, P., Principles of Data Mining. MIT Press, Cambridge, MA (2001)
15. International Software Benchmarking Group: www.isbsg.org
16. Jeffery, R., Wieczorek, I.: A Comparative Study of Cost Modeling Techniques Using Public Domain Multi-Organizational and Company-Specific Data. Proc. European Software Control and Metrics Conference, Munich Germany (2000) 239–248
17. Jeffery, R., Ruhe, M., Wieczorek, I.: Using Public Domain Metrics to Estimate Software Development Effort. Proc. 7th IEEE Int. Software Metrics Symposium, London UK (2001) 16–27
18. Jensen, F.: An Introduction to Bayesian Networks. Springer, Berlin Heidelberg New York (1996)
19. Jorgensen, M.: A Review of Studies on Expert Estimation of Software Development Effort. Journal of Systems and Software, 70(1–2) (2004) 37–60
20. Jorgensen, M.: Practical Guidelines for Expert-Judgment-Based Software Effort Estimation. IEEE Software, 22(3) (2005) 57–63
21. Jorgensen, M., Sjøberg, D.: An Effort Prediction Interval Approach Based on the Empirical Distribution of Previous Estimation Accuracy. Information and Software Technology, 45(3) (2003) 123–126
22. Kleinberg, J., Papadimitriou, C., Raghavan, P.: A Microeconomic View of Data Mining. Journal of Data Mining and Knowledge Discovery, 2(4) (1999) 311–324
23. MacDonell, S., Shepperd, M.: Combining Techniques to Optimize Effort Predictions in Software Project Management. Journal of Systems and Software, 66(2) (2003) 91–98
24. Mair, C., Shepperd, M.: An Investigation of Rule Induction Based Prediction Systems. Proc. Int. Conf. on Software Engineering, Los Angeles USA (1999)
25. Mair, C., Kadoda, G., Lefley, M., Phalp, K., Schofield, C., Shepperd, M., Webster, S.: An Investigation of Machine Learning Based Prediction Systems. Journal of Systems and Software, 53(1) (2000) 23–29
26. Marbán, O., Amescua Seco, A., Cuadrado, J., García, L.: Cost Drivers of a Parametric Cost Estimation Model for Data Mining Projects (DMCOMO). ADIS Workshop on Decision Support in Software Engineering, Madrid Spain (2002)
27. Maxwell, K., Applied Statistics for Software Managers, Prentice-Hall, Englewood Cliffs (2002)
28. Mirhaji, P., Allemang, D., Coyne, R., Casscells, W.: Improving the Public Health Information Network Through Semantic Modeling. IEEE Intelligent Systems, 22(3) (2007) 13–17
29. Passing, U., Shepperd, M.: An Experiment on Software Project Size and Effort Estimation. Proc. Int. Symp. on Empirical Software Engineering (2003) 120–127
30. Puppe, F.: XPS-99: Knowledge-Based Systems – Survey and Future Directions, Lecture Notes in Computer Science of the 5th Biannual German Conference on Knowledge-Based Systems, Würzburg Germany (1999) 193–200
31. Rissanen, J. Stochastic Complexity in Statistical Inquiry, World Scientific Publication, River Edge (1989)
32. Ruhe, M., Jeffery, R., Wieczorek, I.: Cost Estimation for Web Applications. Proc. 25th Int. Conf. on Software Engineering (2003) 285–294
33. Sentas, P., Angelis, L., Stamelos, I.: Software Productivity and Effort Prediction with Ordinal Regression. Journal of Information and Software Technology, 47(1) (2005) 17–29

34. Stamelos, I., Angelis, L., Dimou, P. Sakellaris, E.: On the Use of Bayesian Belief Networks for the Prediction of Software Productivity. Information and Software Technology, 45(1) (2003) 51–60
35. Turban, E., Aronson, J., Liang, T., Sharda, R.: Decision Support and Business Intelligence Systems. Prentice Hall, Upper Saddle River (2006)
36. Virvou .M: A Cognitive Theory in an Authoring Tool for Intelligent Tutoring Systems. IEEE Int. Conf. on Systems Man and Cybernetics 2002, Vol. 2 Hammamet Tunisia (2002) 410–415
37. WEKA: http://www.cs.waikato.ac.nz/ml/weka
38. Welzer, T., Rozman, I., Druzovec, M., Brumen, B.: Reusability and Requirements Engineering in Intelligent Systems. IEEE Int. Conf. on Systems Man and Cybernetics, Vol. 5 (1999) 796–799
39. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco (2005)