Fuzzy model for an early estimation of software development effort

S. Elyassami and A. Idri

***Abstract*—**Estimating the cost of software is one of the most challenging tasks in project management. Accurate cost estimates of software projects and avoiding the overestimates and the underestimates are critical to reach the project's success. Considerable research attention is now directed at gaining a better software cost estimating models. In this paper, we investigate the use of Fuzzy C5 decision tree for software cost estimation; it is designed by integrating the principles of C5 decision tree and the fuzzy set-theoretic concepts. The fuzzy logic is used to provide a formal description of concepts representing data by linguistic values and fuzzy variables in order to handle imprecision and uncertainties in software cost estimation. We have utilized the two statistics that are often used to measure the accuracy of predictive models: the mean magnitude relative error, MMRE, and the number of predictions within 25% of the actuals, pred(25). An empirical validation of our model is reported using the International Software Benchmarking Standards Group data repository. Empirical results approve the deductions gotten in previous studies: incorporate fuzzy logic and decision tree in cost estimation models improve greatly the accuracy of produced estimates. The results are compared with those produced by the fuzzy version of ID3 decision tree.

***Keywords*—**Decision Tree, Effort Estimation, Fuzzy Logic, Fuzzy ID3, Fuzzy C5, Software project.

I. Introduction

PREDICT the amounts of effort that will be required to develop system are crucial for a number of purposes. These include budgeting, tradeoff, risk analysis, project planning, project control and software improvement investment analysis. However, produce an accurate estimate of software development costs at the beginning of the project life cycle is a very complex task. Molokken and Jorgensen report that software projects expend between thirty percent and forty percent more effort than is estimated [17]. Improving the estimation accuracy and the techniques to produce better estimates continues to attract considerable research attention. In order to achieve accurate estimates and avoid the overestimates and the underestimates, several techniques have been developed and validated in the last few decades. The software cost estimation methods available including algorithmic methods, estimating by analogy, expert judgment method, price to win method, top-down method, and bottom-up method. The modeling technique used by many software cost estimation models is globally based on a mathematical function such as effort=α×sizeβ, where α represents a productivity coefficient, and β indicates the economies/diseconomies scale coefficient factor. However, other cost estimation models are based on computational intelligence techniques [16] [19] such as case based reasoning [30], decision trees [31], artificial neural networks [9] [30], fuzzy logic based models [18] [29] and decision trees.

The decision tree method is widely used for inductive learning and has been demonstrating its superiority in terms of predictive accuracy in many fields [4] [23]. The most important benefits when using estimation by decision trees are: First, decision trees approach may be considered as "white boxes", it is simple to understand and easy to explain its process to the users, contrary to other learning methods. Second, decision trees may be used to feature subset selection to exceed the problem of cost driver selection in software.

In the real world, data are often imprecise or uncertain. To analyses and process such data, fuzzy decision trees have been proposed [1], [2], [5], [13], [14], [20] as a combination of classical decision trees with fuzzy sets and fuzzy logic introduced by Zadeh [33]. The most important feature of fuzzy sets is value in the range of [1, 0] instead of {1, 0} [6], [21], [22]. The fuzzy logic can be used to provide a formal description of concepts representing data by linguistic values and fuzzy variables. Fuzzy Logic is determined as a set of mathematical principles for knowledge representation based on degrees of membership rather than on crisp membership of classical binary logic.

Several attempts have been made to revitalize some of the existing models, by introducing fuzzy logic, in order to handle imprecision and uncertainties in software cost estimation field. The first work which appears the theory of fuzzy sets in decision trees is attributed to Chang and Palvlidis in 1977 [27], which defined a binary tree using a branch-bound-backtrack algorithm, but limited instruction on fuzzy decision tree construction. In 1980, Adamo proposed to extend the method to decision trees where the data involved appear as words belonging to the common language whose semantics representations are fuzzy sets [1]. The proposed fuzzy decision trees have been achieved through the modification of the ID3 algorithm. Pedrycz et al. [13] investigate a fuzzy set approach to estimate software project efforts. They propose an augmentation of the well-known class of COCOMO cost estimation models by admitting a granular form of the

S. Elyassami is with the ENSIAS, Mohammed Vth –Souissi University, Department of Software Engineering, BP. 713, Madinat Al Irfane, Rabat MOROCCO.

A. Idri is with the ENSIAS, Mohammed Vth –Souissi University, Department of Software Engineering, BP. 713, Madinat Al Irfane, Rabat MOROCCO.

estimates of the variables used there and found that the concept of information granularity and fuzzy sets, in particular, plays an important role in making software cost estimation models more users friendly. Idri et al. [7] studied the application of fuzzy logic to the cost drivers of the COCOMO'81 model, specifically its intermediate version. Each cost driver in the intermediate COCOMO'81 model is measured using a rating scale of six linguistic values. The proposed model tolerates imprecision in inputs and generates more gradual outputs that is less sensitive to the changes in the inputs. An approach using fuzzy logic was also proposed by Idri et al. [6] to handle projects attributes described by categorical variables instead of numerical variables.

We are concerned, in our researches, with fuzzy decision trees models that allow exploiting the tolerance for imprecision, uncertainty and approximate reasoning offered by the fuzzy logic theory. Experiments [3] have shown that combining fuzzy logic and the decision tree models providing more realistic estimates for software development effort. In an earlier work [4] we have empirically studied the use of fuzzy ID3 decision tree technique for software cost estimation based on ISBSG dataset. The impact of thresholds and especially the fuzziness control threshold that controlling the growth of the tree, on the accuracy of fuzzy ID3 was investigated.

In the present paper we are concerned with studying the fuzzy C5 decision tree model for software cost estimation and the impact of the pruning confidence factor on the accuracy of fuzzy C5 decision tree estimates. An overview on the use of fuzzy decision trees in software cost estimation is also given. This paper is organized as follows: In Section 2, we briefly describe the fuzzy C5 decision tree and we also show how a fuzzy C5 decision tree can be used to estimate software development effort. In Section 3, we present the description of dataset used to perform our empirical studies and we also describe the data pre-processing carried out in our case study. Section 4 focuses on the experimental design and the evaluation criteria adopted to measure the predictive accuracy of the model. In section 5, we discuss the results obtained when the fuzzy C5 decision tree is used to estimate software development effort. In section 6, we provide a comparison of the estimation results produced by means of the fuzzy C5 model and the fuzzy ID3 model.

II. Fuzzy decision tree for software cost estimation

Decision tree based methods are widely used in data mining and decision support applications. Decision tree is fast and easy to use for rule generation and classification problems, moreover it is an excellent representation tool of decisions. The popularity of decision tree is based on the algorithm ID3 by Quinlan [14]. The most widely used algorithms for building decision trees are ID3 [14], C4.5 [15] and CART [1]. These algorithms generate a tree structure through recursively partitioning the feature space until the whole decision space is completely partitioned into a set of non-overlapping subspaces. The selection of features for the partitioning process is another important characteristic of decision trees because only relevant features are selected, improving the time taken to classify new examples as well as the interpretability of the model. This type of feature selection is usually known as embedded. Overfitting can be avoided by a stopping criterion that prevents some sets of training examples from being subdivided, or by removing some of the structure of the decision tree.

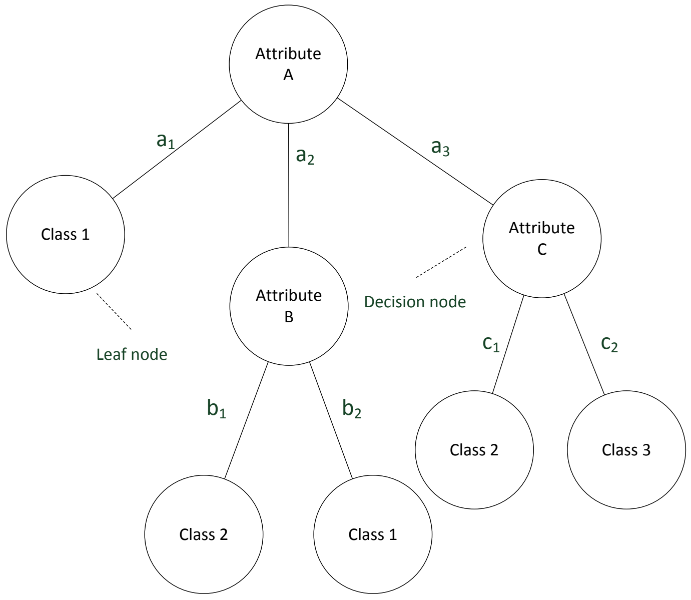

Fig. 1 An example of a sample decision tree

Decision tree is a classifier in the form of a tree structure, where each node is either a leaf node that indicates the value of the target attribute (class) of examples, or a decision node that specifies some test to be carried out on a single attribute-value, with one branch and sub-tree for each possible outcome of the test.

More specifically, decision trees classify instances by sorting them down the tree from the root node to some leaf node, which provides the classification of the instance. Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute.

A decision tree can be used to classify an example by starting at the root of the tree and moving through it until a leaf node, which provides the classification of the instance.

Decision tree for software cost estimation is formed of one root node, which is the tree top, and a series of other nodes. Each node corresponds to a split on the values of one input variable represented by a cost driver and the terminal nodes (or leaves) are represented by the work effort.

C5.0 [16] is an extension of the C4.5 algorithm designed by Ross Quinlan. Quinlan has created C5.0 and See5 (See5 for Windows and C5.0 for Unix/Linux) to give a number of improvements that were not supported by C4.5. In general, C5.0 builds smaller decision trees than C4.5, supports boosting, uses less memory while being more accurate and is significantly faster than C4.5.

Fuzzy C5 decision tree is based on a fuzzy implementation of the C5.0 algorithm. The major characteristic of fuzzy C5 is that support fuzzy thresholds; each example belongs to a node to a certain degree. For example, the attribute "FP" (Function point) in a crisp decision tree two branches associated with it, one for FP < TH and another for FP > TH for some threshold TH. When the value of FP is near TH, small changes in the value can generate quite different classifications. With fuzzy thresholds, both branches of the tree are investigated if the value of FP is close to TH; the results are then combined to give a classification that change more slowly with the value of FP.

Fuzzy C5 decision tree algorithm builds decision trees from a set of training data based on information gain heuristic and entropy measures to decide on the importance of the features. However, the features are all defined in terms of fuzzy sets before the induction of the tree. The main steps of the induction process of the fuzzy C5 tree are:

1. Define the fuzzy representation of the training set attributes
2. Calculate the entropy and information gain of each attribute to split the training set and generate rules until all attribute are used or all training examples are classified;
3. Once the tree is induced, prune it using a defined confidence limit to estimate the real error (25% is the default value).

Regarding the pruning process, C5 employs post-pruning, the pruning takes place after the tree is completely induced. The pruning process basically assesses the error rates of the tree and its components directly on the set of training examples.

To understand the process of pruning, assume T training cases are covered by a leaf, E of them incorrectly. This way, the error rate for this leaf is E/T. If we take this set of T training cases as a sample, it can tell us something about the probability of error over the entire population of examples covered by this leaf. This probability cannot be determined exactly, but it has a probability distribution that is usually summarized by a pair of confidence limits. For a given confidence factor level CF, the upper limit on this probability can be found from the confidence limits for the binomial distribution; this upper limit is here written as UCF(E, T), and since its upper and lower limits are symmetrical, the probability that the real error rate exceeds UCF(E, T) is CF/2. As pointed out by Quinlan, although one might argue that this heuristic is questionable, it frequently yields acceptable results [12]. The default confidence limits used by C5 is 25%. It is important to notice that the smaller the confidence limit, the higher the chances of pruning, and the higher the confidence limit, the smaller the chances of pruning.

The fuzzy decision tree generation process consists on the decomposition of selected attributes into fuzzy sets, building of the fuzzy decision tree from the ISBSG dataset and measure of the estimates accuracy generated by the fuzzy C5 decision tree using the actual effort and the estimated effort. Where actual effort represents real effort derived from historical data on software projects already have been finished and estimated effort represents the estimated value of project effort by using our model based on fuzzy C5 decision tree.

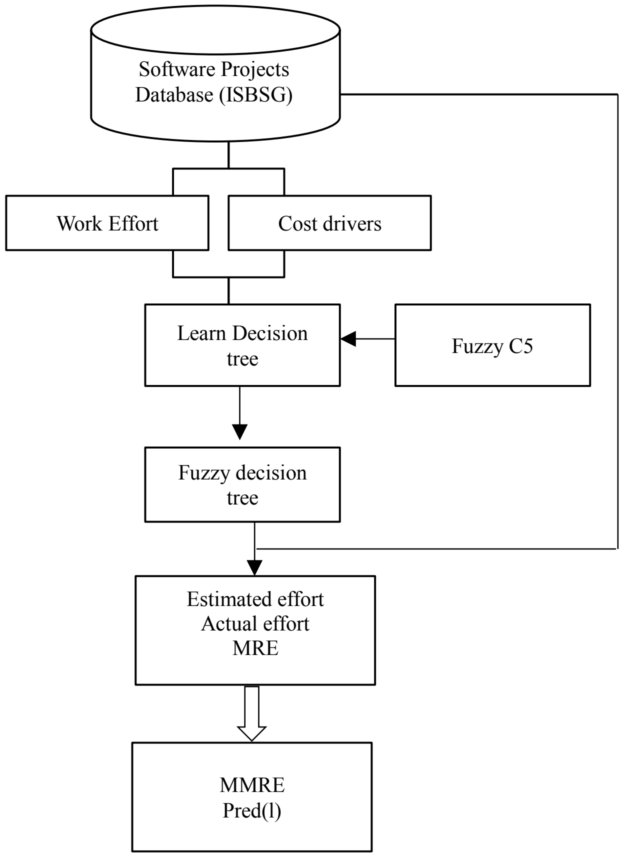

Fig. 2 Fuzzy C5 decision tree generation process

III. Data pre-processing

We evaluated our approach on the ISBSG (International Software Benchmarking Standards Group) data repository. The ISBSG repository (release 8) consists of 2027 projects collected from twenty countries around the world. Major contributors are Australia (21%), Japan (20%) and the United States (18%).

The reduction of dimensionality in ISBSG dataset is primordial to operate faster and improve classification accuracy. This section describes the preprocessing carried out on the ISBSG dataset. To do so, we performed data transformation and data selection.

A. Data transformation

Data transformation enables algorithms to be applied without difficulty and improves their performance and their effectiveness. Data transformation operations contribute to get the required information from incomplete, noisy and incoherent set of data. We have chosen in this study to applied two data transformation operations: Handle missing values and data normalization.

1) Missing Data

Missing Data is the common problem that comes up through the data preparation stage. There are many approaches to deal with missing values, for instance: Avoid Missing data and Data Imputation.

The first approach generally lost too much useful information. In this study, the second method is chosen to handle missing values.

Imputation methods use information available in the dataset to predict missing values. These methods may be grouped into two categories: single and multiple imputation methods [20]. In single imputation methods, the missing value is replaced with one imputed value, and in multiple imputation methods, several values are used.

To handle missing data in ISBSG dataset, the Mean or Mode Single Imputation (MMSI) method is used. For continuous attributes, MMSI method substitutes the missing value with the mean. For nominal attributes, MMSI method substitutes the missing value with the mode (the value that is repeated more often than any other). For example, for the language type attribute, the dominant value in the dataset is 3GL (67%). So, in the case of language type attribute, the missing values are replaced by 3GL.

2) *Normalizations*

Due to the nature of software attributes, some of continuous features show a larger range of values than others which may make the effect of this feature too important or easily neglected. The solution is to scale continuous features into the same range. To achieve this, all continuous features are normalized applying the min max normalization formula as show in eq. (1) such that all numeric variables are scaled in the range [0, 1].

$$x_i(k) = \frac{x_i(k) - \min(x(k))}{\max(x(k)) - \min(x(k))} \tag{1}$$

B. *Data Selection*

To increase the efficiency of prediction accuracy by FDT we have to apply some data selection techniques to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. In this study, two operations in data reduction are used: Feature Selection and Case Selection.

1) *Feature Selection*

The aim is to identify the features that have the highest potential to provide good effort estimates. A function point is used as a measure of software size and was chosen in this study, firstly. The second variable is team size that is a potential cost factor. Thirdly, development platform was chosen, where each project is classified as either, a PC, Mid-Range or Main Frame. Other important criterion for selecting projects is user base. User base enclose 3 criteria: the first one is the number of business units that the system services, the second one is the number of physical locations being serviced by the system and the last one is the number of users using the system concurrently. Finally, the project effort is used as the dependent feature that provides the total effort for all phases of the project development life cycle.

Seven criteria are chosen in estimation data set, these criteria are also suggested by ISBSG. Table I includes the project metrics that have been used in this study.

Table I. Project metrics used

Metric	**Definition**
Function Points	adjusted function point count number
Max Team Size	maximum number of people on the project
User Base - Business Units	number of business units that the system services
User Base - Locations	number of physical locations being serviced by the system
User Base - Concurrent Users	number of users using the system concurrently
Development Platform	primary platform (PC, Mid-Range or Mainframe)
Normalized Work Effort	total effort in hours recorded against the project for all phases of the development life cycle

2) *Case Selection*

In order to obtain a reduced data set, some projects had to be excluded. The raw data was filtered by several criteria. Four filtering criteria are used in this study:

- ➢ Data Quality Rating
- ➢ Resource Levels
- ➢ Rating for Unadjusted Function Points
- ➢ Development Type

a) *Data Quality Rating*

We have to ensure that the model is built on the basis of a reliable data set. According to ISBSG, only projects with data quality ratings A or B were included. Projects rated C and D offer valuable data, but uncertainty about some of their size or effort values. Hence, projects whose quality ratings are C and D were not included in the estimation data set.

b) *Resource Levels*

Four resource levels are identified in the ISBSG data collection instrument:

- ➢ 1 = development team effort
- ➢ 2 = development team support
- ➢ 3 = computer operations involvement
- ➢ 4 = end users or clients

Generally, the cost estimation models take into account only development team effort and support (resource level 1 and resource level 2) rather than considering the other costs like computer operations involvement and effort expended by end user (resource level 3 and resource level 4).

Therefore, only projects recorded at the first resource level or at the second resource level were considered. So, the work effort for the development team and support is included in the work effort number. Projects with level 3 or level 4 were discarded. The goal is to make the results as generalizable as possible.

c) Rating for Unadjusted Function Points

Rating for Unadjusted Function Points (UFP) is an ISBSG rating code applied to the unadjusted FP data by the ISBSG quality reviewer to measure the UFP integrity. Projects of UFP A or B are included in the subset. Projects whose UFP ratings are C and D were left.

d) Development Type

Development type is the final criterion on which we based our investigation. This measure describes whether the development is a new development, enhancement or redevelopment. Only the new development projects are considered in our study.

Table II. The criteria used to reduce the dataset

Criteria	**Selected Data**	**Discarded Data**
Development Type	New Development	Enhancement and Redevelopment
UFP	A or B	C and D
Resource Levels	1 or 2	3 and 4
Data Quality Rating	A or B	C and D

The original ISBSG data set was reduced as follows:

- Remove projects if they were not assigned a high data quality rating (A or B) by ISBSG.
- Remove projects with resource levels different from 1 or 2 (development team effort and development team support only).
- Remove projects if their rating for unadjusted function points different from A or B.
- Remove projects with development type enhancement or redevelopment and keep only new developments projects.

IV. Experiment Design

This section describes the experiment design of the fuzzy C5 decision tree. Fuzzy C5 algorithm was applied for the induction of the decision trees using the latest release of See5 on the ISBSG project data. Each ISBSG project is described by a set of attributes; the fuzzy partitions were automatically created for each attribute. The classification is determined as follows: If the attribute value in question is under lower bound or above upper bound, classification is determined using the single branch corresponding to the studied case. If the attribute value is between the lower and the upper bound, both branches of the tree are explored and the results combined probabilistically.

A series of experiments is conducted and the MC that represents the number of minimum instances per node was held constant at 2. Fuzzy C5 model was evaluated with CF values ranging from 0.1 to 1 by an increment of 0.1.

A. Model building

Fuzzy C5 decision tree is a top–down and recursive decision inducer. It consists of two conceptual phases: growing and pruning. In each iteration, the algorithm considers the partition of the training set using the outcome of a fuzzy membership function of the input attributes. The selection of the most appropriate function is made according to the splitting measure. After the selection of an appropriate split, each node further subdivides the training set into smaller subsets, until no split gains sufficient splitting measure or a stopping criteria is satisfied. After the growing stage, decision tree is pruned by removing insignificant nodes using a pruning method.

B. Splitting Criterion

The algorithms that are used for building decision trees habitually work top-down by selecting a variable at each step that is the next best variable to use in splitting the data set. The best variable is defined by how well it splits the data set into homogeneous data subsets that have the same value of the target variable. Different algorithms use different formulae for splitting the data set.

Fuzzy C5.0 builds decision trees from a set of training data using the information gain based on the concept of entropy. At each node of the tree, fuzzy C5.0 chooses the attribute of the data that most successfully splits the data set into subsets improved in one class or the other. The splitting criterion is the normalized information gain (gain ratio). The attribute with the highest gain ratio is chosen to make the decision. The gain ratio is defined as follows:

$$GainRatio(a_{i,}S) = \frac{InformationGain(a_{i,}S)}{Entropy(a_{i,}S)} \quad (2)$$

where S is the training data and $A = a_1, a_2, a_3, \dots, a_i$ is the input attribute.

C. Stopping Criterion

Fuzzy C5 growing phase continues until a stopping criterion is activated. The following conditions are common stopping rules:

1. All instances in the training set belong to a single value.
2. The maximum tree depth has been reached.
3. The best splitting criteria is not greater than a certain threshold.
4. If the node were split, the number of cases in one or more child nodes would be less than the minimum number of cases for child nodes.
5. The number of cases in the terminal node is less than the minimum number of cases for parent nodes.

Fuzzy C5 algorithm used the last condition as stopping criterion; The Minimum cases parameter that constrains the degree to which the decision tree can grow up. Throughout the induction of the decision tree, the dataset is divided on the attributes that provide the most information gain. When a split is made generating a child leaf that represents less than a minimum number of examples from the dataset, the parent node and its children are replaced by a single node.

D. Pruning Method

Using tightly stopping criteria tends to create small and under–fitted decision trees. On the other hand, using loosely stopping criteria tends to generate large decision trees that are over–fitted to the training data. Pruning methods were developed for resolving this problem. In the present study, a loosely stopping criterion is used, allowing the decision tree to overfit the training data. Then the over-fitted tree is reduced into a smaller tree by removing sub–branches that are not contributing to the generalization accuracy.

The pruning method used in C5.0 is the Error-Based pruning. The degree of pruning is controlled by the confidence factor parameter CF. This parameter affects the way that error rates are calculated during the post-pruning stage. The higher the CF, the more likely the current error rate is accepted and no pruning will be done.

Working from the bottom up, the post-pruning requires a fully induced decision tree and attempts to remove insignificant nodes by calculating the error rate for each child node, and then to derive the total error of the parent node.

E. Evaluation criteria

To measure the accuracy of the estimates generated by the fuzzy C5, we employ two statistics that are often used to measure the accuracy of predictive models: the mean magnitude relative error (MMRE) and the number of predictions within 25% of the actuals (pred(25)).

The magnitude of relative error (MRE) is defined as follows:

$$MRE = \left| \frac{Effort_{actual} - Effort_{estimated}}{Effort_{actual}} \right| \tag{2}$$

where $Effort_{actual}$ is the actual effort of a project in the dataset, and $Effort_{estimated}$ is the estimated effort that was obtained using a model or a technique.

The MRE values are calculated for each project in the datasets, while mean magnitude of relative error (MMRE) computes the average over N projects.

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} \left| \frac{Effort_{actual,i} - Effort_{estimated,i}}{Effort_{actual,i}} \right| \times 100 \tag{3}$$

The acceptable target values for MMRE are $MMRE \leq 25$. This indicates that on the average, the accuracy of the established estimation model would be less than 25%.

Another widely used criterion is the prediction Pred(p) which represents the percentage of MRE that is less than or equal to the value p among all projects. This measure is often used in the literature and is the proportion of the projects for a given level accuracy [19]. The definition of Pred(p) is given as follows:

$$Pred(p) = \frac{k}{N} \tag{4}$$

Where N is the total number of observations and k is the number of observations whose MRE is less or equal to p. A common value for p is 25, which also used in the present study. The prediction at 25%, Pred(25), represents the percentage of projects whose MRE is less or equal to 25%. The acceptable values for Pred(25) are $Pred(25) \geq 75$.

This indicator reveals what proportion of estimates is within a tolerance of 25%. Pred(p) is clearly insensitive to the degree of inaccuracy of estimates outside the specified tolerance level. For example, a pred(25) indicator will not distinguish between a prediction system for which predictions deviate by 27% and one for which predictions deviate by 270%.

V. Overview of the Experimental Results

This section presents and discusses the results obtained when applying the fuzzy C5 decision tree to the ISBSG dataset. We conducted several experiments using several values for the pruning confidence factor. The aim is to find the most appropriate configuration that improves the estimates.

The results for the different configurations have been reported. Figure 2, 3 and 4 show the tree size and the accuracy of the generated fuzzy C5 decision trees, measured in terms of MMRE and Pred(25), on ISBSG dataset.

Table III. Fuzzy C5 model results

CF	Size	MMRE	PRED
0.1	37	15	83,12
0.2	49	5	94,8
0.3	49	5	94,8
0.4	52	1,56	97,4
0.5	54	0,45	98,7
0.6	54	0,45	98,7
0.7	57	0	100
0.8	57	0	100
0.9	57	0	100

Figure 4 compares the accuracy of the model, in terms of MMRE, when varying the pruning CF. When setting the pruning CF at 0.2 the model produces a prediction error equal to 5 (MMRE=5) and when setting the pruning CF at 0.4 the model produces a prediction error equal to 1.56 (MMRE = 1.56). We note that the performance of fuzzy C5 model increased as the pruning confidence factor increased.

Figure 5 shows the results of the model, in terms of Pred(25), when varying the pruning CF. From this figure, we note that the accuracy of fuzzy C5 model performs much better when increasing the pruning CF. For example, when setting the pruning CF at 0.1 the number of predictions within 25% of the actuals is equal to 83.12 (pred(25)= 83.12) and when setting the pruning CF at 0.5 the number of predictions within 25% of the actuals equal to 98.7 (pred(25) = 98.7). The results obtained using different configurations of fuzzy C5 for ISBSG dataset shows that lowering the pruning confidence factor is useful for reducing the tree size, and also helps in filtering out inappropriate nodes that would otherwise lead to classification errors.

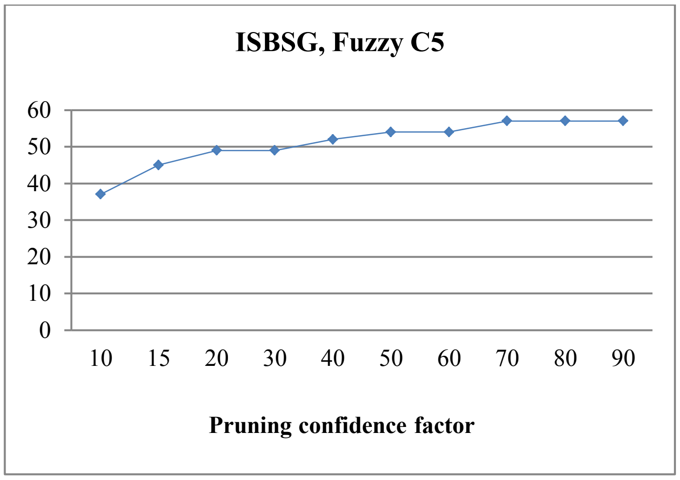

Fig. 3 Fuzzy C5 tree size

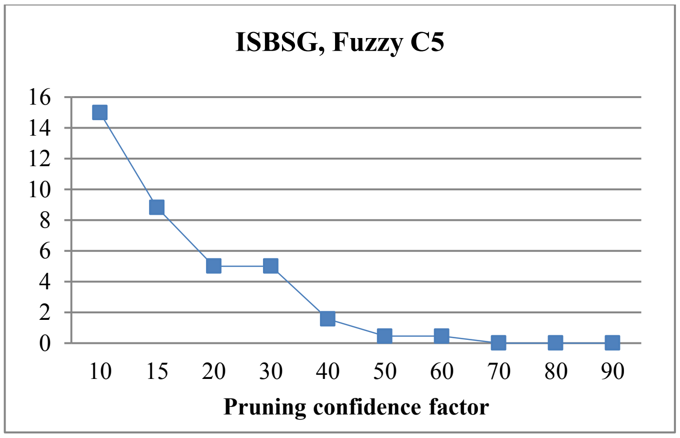

Fig. 4 Estimation accuracy on fuzzy C5 decision tree in term of MMRE

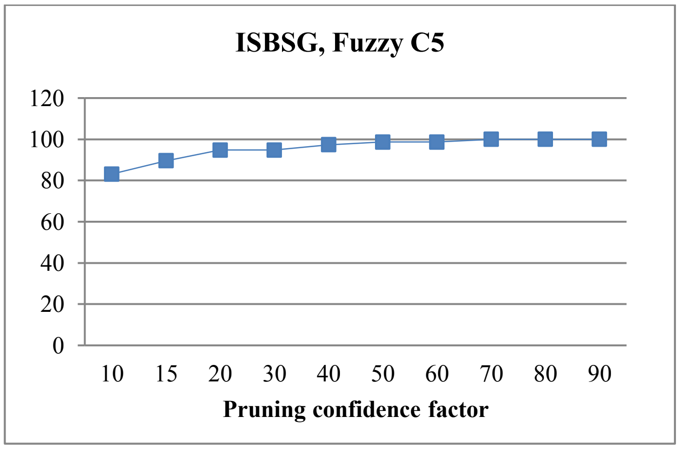

Fig. 5 Estimation accuracy on fuzzy C5 decision tree in term of Pred(25)

VI. Comparison Results

In an earlier work [1] we have studied the use of fuzzy ID3 decision tree for software cost estimation on ISBSG dataset. The fuzzy ID3 models are grown using different values for the fuzziness control threshold that permit controlling the growth of the generated fuzzy trees. The fuzziness control threshold verifies if the ratio of membership of a class at tree node is higher than a given threshold. The value for the fuzziness control threshold was varied between 0.1 and 0.9.

A comparison, in terms of MMRE and Pred(25), between fuzzy ID3 model results and the results produced by the fuzzy C5 model has been investigated.

Figures 5 and 6 show the estimation accuracy, in term of Pred(25), generated respectively by fuzzy C5 and fuzzy ID3 models. Figures 4 and 7 show the estimations accuracy in term of MMRE generated respectively by fuzzy C5 and fuzzy ID3 models. Comparing the two models results, cost estimates produced by the fuzzy C5 model are all acceptable even if varying the pruning confidence factor value. However, the fuzzy ID3 model produces acceptable cost estimates when decreasing the fuzziness control threshold under 0.5. In general, the fuzzy C5 model shows better estimation accuracy than the fuzzy ID3 model in terms of MMRE and Pred(25).

The fuzzy C5 model performs much better when increasing the pruning confidence factor value while the fuzzy ID3 model performs much better when decreasing the fuzziness control threshold value. We can explain this divergence by the facts follows:

- A lower fuzziness control threshold value produces a larger tree and the classification accuracy will be higher,
- A lower pruning confidence factor value reduces the generated tree while the classification accuracy will be lower.

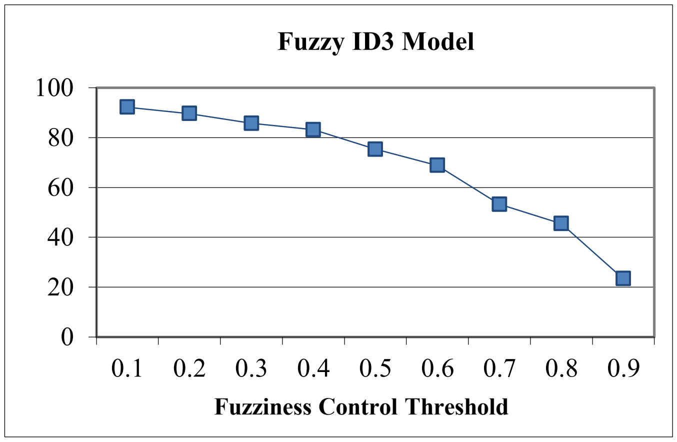

Fig. 6 Estimation accuracy on fuzzy ID3 decision tree in term of Pred(25)

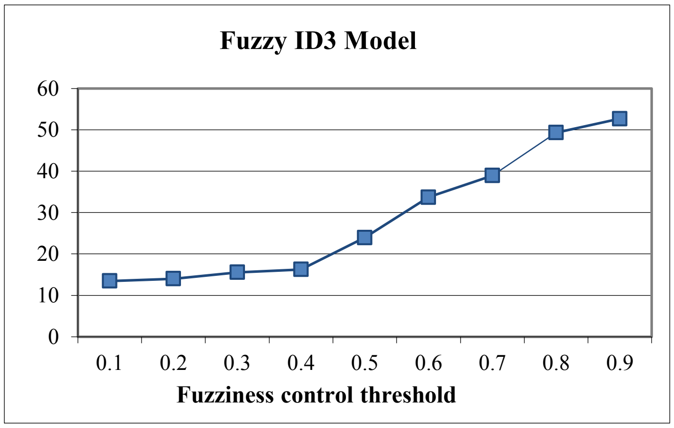

Fig. 7 Estimation accuracy on fuzzy ID3 decision tree in term of MMRE

VII. Conclusion

A conclusion section is not required. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions.

In this paper, we have studied the impact of the pruning confidence factor of on the accuracy of the fuzzy C5 decision tree model for software cost estimation. We have conducted several experiments using different values for the pruning confidence factor to fix the most appropriate configuration that generate the optimal model and improves the estimates. We have evaluated our model on the ISBSG data repository.

The mean magnitude relative error (MMRE) and the number of predictions within 25% of the actuals (pred(25)) have been used to measure the accuracy of the estimates generated by the fuzzy C5 model. These indicators are often used to measure the accuracy of predictive models

The results show that combining fuzzy logic and the decision tree improves greatly the accuracy of estimates. The experiments show the superiority of models cost estimation software development based on fuzzy decision trees. In our testing, we found that proper utilization of pruning confidence factor and of fuzziness control threshold has shown an increase of estimation accuracy. Therefore, several values for the models parameters must be evaluated when building fuzzy decision trees for software cost estimation to fix the most appropriate value for the study dataset.

References

[1] Adamo, J.M.: Fuzzy decision trees. Fuzzy Sets and Systems 4, 207–219 (1980).
[2] Bartczuk, L., Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M.: The new version of Fuzzy-ID3 algorithm. In: ICAISC 2006.LNCS (LNAI), vol. 4029, pp. 1060–1070. Springer, Heidelberg (2006).
[3] Breiman, L. Friedman, J.H. Olsen, R.A. & Stone, C.J. "Classification and Regression Trees", Wadsworth, 1984.
[4] Berger, H. Merkl, D. and Dittenbach, M. "Exploiting Partial Decision Trees for Feature Subset Selection in e-Mail Categorization," in Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 2006), Dijon, France, 2006, pp. 1105-1109.
[5] Crespo, J.J., Sicilia, M.A., Cuadrado, J.J.: On the Use of Fuzzy Regression in Parametric Software Estimation Models: Integrating Imprecision in COCOMO Cost Drivers. WSEAS Transactions on Systems 1(3), (2004) 96-101
[6] Dubois, D., Prade, H.: Fuzzy Sets and Systems: Theory and Applications. Academic Press, San Diego (1980).
[7] Elyassami, S. and Idri, A. "Applying Fuzzy ID3 Decision Tree for Software Effort Estimation", International Journal of Computer Science Issues (IJCSI), Vol. 8, Issue 4, No 1, July 2011, pp. 131-138
[8] Elyassami, S. and Idri, A. "Investigating effort prediction of software projects on the ISBSG dataset" International Journal of Artificial Intelligence & Applications (IJAIA), Vol.3, No.2, March 2012, pp.121-132.
[9] Finnie, G. R. and Witting, G. and Desharnais, J.-M. "A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models", Systems and Software, Vol. 39, No. 3, 1997, pp. 281-289.
[10] Idri, A. and Abran, A. "Evaluating Software Project Similarity by using Linguistic Quantifier Guided Aggregations" Proceedings of IFSA/NAFIPS 2001, Vancouver,Canada, pp. 470 - 475, 2001.
[11] Idri,A. Kjiri, L. and Abran,A. "COCOMO Cost Model Using Fuzzy Logic", 7th International Conference on Fuzzy Theory & Technology, Atlantic City, NJ, February, 2000. pp. 219-223.
[12] Idri, A. Abran, A. and S. Mbarki, "An Experiment on the Design of Radial Basis Function Neural Networks for Software Cost Estimation", in 2nd IEEE International Conference on Information and Communication Technologies: from Theory to Applications, 2006, Vol. 1, pp. 230-235.
[13] Janikow, C.Z.: Fuzzy decision trees: issues and methods. IEEE Transactions onSystems, Man, and Cybernetics 28(3), 1–14 (1998).
[14] Janikow, C.Z.: Exemplar learning in fuzzy decision trees. In: Proc. IEEE InternationalConference on Fuzzy Systems, Piscataway, NJ, pp. 1500–1505 (1996).
[15] Korte M. and Port, D. "Confidence in Software Cost Estimation Results Based on MMRE and PRED", PROMISE'08, May 12-13, 2008, pp . 63-70.
[16] Montequín, V. R. Balsera, J. V. González, C. A. Huerta, G. M. Software project cost estimation using AI techniques. Proceedings of the 5th WSEAS/IASME International Conference on Systems Theory and Scientific Computation, ISTASC'05 Malta, September 15-17, 2005 (pp289-293).
[17] Molokken, K. and Jorgensen, M. "A Review of Surveys on Software Effort Estimation", in International Symposium on Empirical Software Engineering, 2003, pp. 223-231.
[18] Nisar, MW and Wang, Y.-J. and Elahi, M. "Software Development Effort Estimation Using Fuzzy Logic – A Survey", in 5th International Conference on Fuzzy Systems and Knowledge Discovery, 2008, pp. 421-427.
[19] Ortega, F. Villanueva, J. Rodriguez, V. Alvarez, V. Effort Estimation in Information Systems Projects using Data Mining Techniques. Proceedings of the 13th WSEAS International Conference on COMPUTERS, Rodos, Greece, July 22-25, 2009 pages: 652-657.
[20] Ruiz-Gómez, J., Ramos-Jiménez, G., Villalba-Soria, A. Modelling based on rule induction learning. Proceedings of the 3rd WSEAS World Multiconference on: Circuits, Systems, Communications and Computers (CSCC'99), Athens, Greece, July 4-9, 1999, Pages: 5991-5996
[21] Rutkowska, D.: Neuro-Fuzzy Architectures and Hybrid Learning. Physica-Verlag, Springer, New York (2002).
[22] Rutkowski, L.: Methods and Techniques of Artificial Intelligence, PWN, Warsaw,Poland (in Polish) (2005).
[23] Pedrycz, W and Sosnowski, Z. A., "The design of decision trees in the framework of granular data and their application to software quality models," Fuzzy Sets and Systems, Vol. 1234, 2001, pp. 271-290.
[24] Pedrycz, W, Peters, J.F and Ramanna, S, A Fuzzy Set Approach to Cost Estimation of Software Projects, Proceedings of the 1999 IEEE Canadian Conference on Electrical and Computer Engineering Shaw Conference Center , Edmonton Alberta, Canada. 1999, pp. 1068-1073.
[25] Quinlan, JR. "Induction on decision tree," Machine Learning, Vol. 1, 1986, pp. 81-106.
[26] Quinlan, JR. C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, CA, 1993.
[27] Robin L. P. Chang, Pavlidis. "Fuzzy decision tree algorithms," IEEE Transactions on Systems, Man, and Cybernetics, pp. 28-35, January 1977.
[28] Rulequest Research, Data Mining Tools See5 and C5.0, http://www.rulequest.com/see5-info.html.

[29] Sharma, V. and Verma, H. K. "Optimized Fuzzy Logic Based Framework for Effort Estimation in Software Development", Computer Science Issues, Vol. 7, Issue 2, No. 2, 2010, pp. 30-38.
[30] Shepperd M. and Schofield. C. “Estimating Software Project Effort Using Analogies.” Transactions on Software Engineering, vol. 23, no. 12, 1997, pp. 736-747.
[31] Selby, R. W. and Porter, A.A. "Learning from examples: generation and evaluation of decision trees for software resource analysis", IEEE Transactions on Software Engineering, Vol. 14, No. 12, 1988, pp. 1743-1757.
[32] Twala, B. Cartwright M. and Shepperd, M. “Comparison of various methods for handling incomplete data in software engineering databases”, Proc. Intl Symp Empirical Soft Eng 2005, 105-114.
[33] Zadeh, L.A.: Fuzzy sets. Information and Control 8, 338–353 (1965).

A. Idri is a Professor at Computer Science and Systems Analysis School (ENSIAS, Rabat, Morocco). He received DEA (Master) (1994) and Doctorate of 3rd Cycle (1997) degrees in Computer Science, both from the University Mohamed V of Rabat. He has received his Ph.D. (2003) in Cognitive Computer Sciences from ETS, University of Quebec at Montreal. His research interests include software cost estimation, software metrics, fuzzy logic, neural networks, genetic algorithms and information sciences.

S. Elyassami received her engineering degree in Computer Science from the UTBM, Belfort-Montbeliard, France, in 2006. Currently, she is preparing her Ph.D. in computer science in ENSIAS. Her research interests include software cost estimation, software metrics, fuzzy logic and decision trees.