Estimating Software Project Effort by Analogy Based on Linguistic Values

Ali Idri
ENSIAS, BP. 713, Agdal
Université Mohamed V Souissi
Rabat, Morocco
E-mail : idri@ensias.ma

Alain Abran
École de Technologie Supérieure
1180 Notre-Dame Ouest,
Montréal, Canada H3C 1K3
E-mail : abran.alain@uqam.ca

Taghi M. Khoshgoftaar
Empirical Software Engineering Laboratory
Florida Atlantic University
E-mail : taghi@cse.fau.edu

Abstract

Estimation models in software engineering are used to predict some important attributes of future entities such as software development effort, software reliability and programmers productivity. Among these models, those estimating software effort have motivated considerable research in recent years. The prediction procedure used by these software-effort models can be based on a mathematical function or other techniques such as analogy based reasoning, neural networks, regression trees, and rule induction models. Estimation by analogy is one of the most attractive techniques in the software effort estimation field. However, the procedure used in estimation by analogy is not yet able to handle correctly linguistic values (categorical data) such as 'very low', 'low' and 'high'. In this paper, we propose a new approach based on reasoning by analogy, fuzzy logic and linguistic quantifiers to estimate software project effort when it is described either by numerical or linguistic values; this approach is referred to as Fuzzy Analogy. This paper also presents an empirical validation of our approach based on the COCOMO'81 dataset.

1. Introduction

Accurate estimation of the effort and the schedule required to develop and/or maintain a software system is one of the most critical activities in managing software projects. This task is known as Software Cost Estimation. In order to make accurate estimates and avoid gross misestimations, several cost estimation techniques have been developed. These techniques may be grouped into two major categories [25]:

- algorithmic models, and
- non-algorithmic models.

The first category holds the most popular technique (at least in the literature), and is illustrated by estimation models such as COCOMO [5, 6, 8], PUTNAM-SLIM[23], and function points analysis [2, 20]. Algorithmic models are derived from the statistical or numerical analysis of historical projects data (simple/multiple/stepwise regression, Bayesian approach, polynomial interpolation, etc). There are two main disadvantages to these models. First, they make an assumption on the form of the prediction function, represented by: $Effort = a \times size^{b}$ where α represents a productivity coefficient and β an economies (or diseconomies) of scale coefficient. Second, they need to be adjusted or calibrated to local circumstances (an example of calibrating and reformulating COCOMO'81 model is in [10]).

The non-algorithmic models have been developed to avoid the above mentioned shortcomings. Recently, many researchers have begun to turn their attention to this alternative, and in particular to a set of approaches based on neural networks, regression trees, rule induction, and case-based reasoning. This alternative has two significant advantages: First, the capability to model the complex set of relationship between the dependent variable to predict (cost, effort) and the independent variables (cost drivers) collected earlier in the lifecycle. Second, the capability to learn from historical projects data (especially for neural networks).

Experience has shown that there does not exist a ‘best’ prediction technique outperforming all the others in every situation. Indeed, Shepperd et al., Niessink and Van Vliet found that estimation by analogy generated better results than stepwise regression [22, 24, 25]. However, Briand et al., Stensrud and Myrtveit reported opposite results [7, 21]. Recent research has been initiated to explain the relationship between different properties of historical projects dataset (size, number of attributes, presence of outliers…) and the accuracy of a prediction system [26]. Our work deals with an important limitation of all estimation techniques, which arises when software projects are described using categorical data (nominal or ordinal scale) such as ‘very low’, ‘low’ and ‘high’. These qualifications are called linguistic values in fuzzy logic terminology. Building cost estimation models based on linguistic values is a serious challenge for the software cost estimation community. Recently, Angelis et al. [4] were the first to propose the use of categorical regression procedure (CATREG) to build cost estimation models when software projects are described by categorical data. This procedure quantifies categorical attributes by assigning numerical values to their categories in order to produce an optimal linear regression equation for the transformed variables. This approach has the following limitations:

- It replaces each linguistic value by one numerical value. This is based on the assumption that a linguistic value can always be defined without vagueness, imprecision and uncertainty. Unfortunately, this is not often the case. Indeed, linguistic values come from human judgements that are always vague, imprecise and uncertain. For example, let us assume that the experience of programmers is measured by three linguistic values: 'low', 'nominal' and 'high'. Most often the meaning of these values are not defined precisely and consequently we cannot represent them by individual numerical values.
- There is no natural interpretation of the numerical values assigned by this approach
- It assigns numerical quantities to linguistic values in order to produce an optimal linear regression equation whereas the initial relation between effort and cost drivers may be non-linear.

A more comprehensive approach to deal with linguistic values is by using fuzzy set theory. Consequently, the purpose of this paper is to provide a new approach based on analogy and fuzzy logic to estimate effort when software projects are described either by numerical or linguistic values.

This paper is organized as follows: A discussion why categorical data should be considered as a particular case of linguistic values is presented in Section 2. In Section 3, we present the classical procedure of estimation by analogy; which procedure cannot handle linguistic values. To overcome this limitation, we propose, in Section 4, a new approach that can be seen as a fuzzification of the classical approach of estimation by analogy. Section 5, presents an empirical validation of our technique by means of the COCOMO'81 dataset. The obtained results are compared to those of three other techniques. Finally, Section 6 discusses our findings and suggests future work in this area.

2. Categorical data and linguistic values

Here we discuss why categorical data, as software measurement researchers define them, are only a particular case of linguistic values. The two terminologies come from two different fields. The first is used in measurement theory whereas the second is used in fuzzy sets theory.

In 2001, measurement theory became over a hundred years old. According to Zuse, it began with Helmholtz pioneering paper: 'Counting and measuring from epistemological point of view' and lead to modern axiomatic representational theory of measurement as presented by Krantz et al. [19, 33].

As in other sciences (physics, medicine, civil, etc.), measurement has been discussed in software engineering for over thirty years. The objective of software measurement is to improve the software process and consequently the quality of its various deliverables. This can be achieved by evaluating, controlling and predicting some important attributes of software entities such as development effort, software reliability, and programmers productivity. However, measurement in software engineering is quite different from other (classical) fields and is due to two main reasons. First, software engineering is a 'young' science and still needs further maturing. Second, most of the software attributes are qualitative rather than quantitative such as portability, maintainability, and reliability. They currently depend on human views. The qualitative issue is related to the scale type on which the attributes are measured. Often, researchers in software measurement identify the scale type of a measure as being one of the five types defined by Stevens in 1946 [27]: Nominal, Ordinal, Interval, Ratio or Absolute.

Categorical attributes are those with nominal or ordinal scale. The Nominal scale is the lowest scale level and it only allows the classification of software entities in different classes or categories. Examples of this scale can be found in literature, such as the language used in the implementation phase (C, C++, Java, Cobol, etc) and the application type (Business, Control, Finance, etc). The Ordinal scale provides us, in addition to the classification of software entities, information about an ordering of the categories. Examples of ordinal attributes are complexity of software (simple, nominal, complex) and software comprehensibility (very low, low, nominal, high).

In software engineering, it seems that the most interesting software attributes are measured either on a Nominal or Ordinal scale. The problem of categorical attributes is caused by the fact that humans are directly involved in the measurement process. Consequently, the results of the measurements may be highly influenced by their own judgement. Thus, humans often use linguistic values such as 'very low', 'complex', 'important' and 'essential' rather than numerical data to evaluate software attributes. When using linguistic values, imprecision, uncertainty and partial truth are unavoidable. However, until now, software measurements community often use numbers or classical interval to represent these linguistic values. This representation does not mimic the way in which humans interpret linguistic values and consequently cannot deal with imprecision and uncertainty. To overcome this limitation, we have suggested the use of fuzzy sets rather than classical interval (or numbers) to represent categorical data [11,12,13,14,15]. The main motivation of fuzzy sets theory, founded by Zadeh in 1965, is apparently the desire to build a formal quantitative framework that captures the vagueness of humans knowledge since it is expressed via natural language. Consequently, in this work we use fuzzy sets theory to deal with linguistic values in the estimation by analogy procedure.

3. Estimation by analogy

Estimation by analogy is essentially a form of Case-Based Reasoning which has four steps [1]:

1-Retrieve the most similar case or cases
2-Reuse the information and knowledge in that case to solve the problem
3-Revise the proposed solution
4-Retain the parts of this experience likely to be useful for future problem solving

For effort estimation, CBR is based on the following affirmation: *similar software projects have similar costs*. It has been deployed as follows. First, each project must be described by a set of attributes that must be relevant and independent. Second, we must determine the similarity between the candidate project and each project in the historical database. Third, we use the known effort values from the historical projects to derive an estimate for the new project. This later step is known as case adaptation.

Vacninanza et al. have suggested that CBR might be usefully adapted to make accurate software effort predictions [28]. Ever since, estimation by analogy has been the subject of studies aimed at evaluating, enhancing, reformulating and adapting the CBR life cycle according to the features of the software effort prediction problem. Shepperd et al. have been involved in the development of CBR techniques and tools to build software effort prediction systems for the past five years [24, 25]. In their recent work, they tried to explain why different research teams have reported widely different results when using CBR technology. In addition to the characteristics of the projects data being used, Shepperd et al. examined the impact of the choice of the number of analogies and adaptation strategies. They used a dataset of software projects collected by a Canadian software house to validate their findings. They found that choosing the number of analogies is important. Specifically, three analogies seemed to be optimal. However, a fixed value for k (number of analogies) was more effective for the large datasets while distance based analogies selection appeared more effective for the smaller datasets. They also found that case adaptation strategies seemed to have little impact on the accuracy of the estimation by analogy [17].

Angelis and Stamelos [3], while studying the estimation by analogy method for Albrecht's software projects, explored the problem of determining the parameters for configuration of the analogy procedure before its application to a new software project. They studied three parameters, which are the choice of distance measure that will be used to evaluate the similarity between software projects, the number of analogies to take into account in the effort estimation, and the statistic that will be used in calculating the unknown effort from the efforts of the similar projects. They suggested that the bootstrap method should be used to configure these three parameters. Bootstrapping consists of drawing news samples of software projects from the original dataset and testing parameter performances on the generated data. This allows the user to identify which parameter values give accurate estimates often. These values can be used to generate prediction for a new software project. This kind of search for optimal parameters is called calibration of the estimation procedure.

However, even though it is well recognized that estimation by analogy is a promising technique for software cost and effort estimation, there are certain limitations that prevent it from being more popular. The most important is that until now it cannot handle linguistic values such as 'very low', 'low', and 'high' whereas many other factors, such as experience of programmers, complexity of modules and software reliability are measured on an ordinal or nominal scale composed of linguistic values. For example, the well-known COCOMO'81 model has 15 attributes out of 17 (22 out of 24 in the COCOMOII) which are measured with six linguistics values: 'very low', 'low', 'nominal', 'high', 'very high', and 'extra-high' [5, 6, 8]. To overcome this limitation, in the next section we present a new method that can be seen as a fuzzification of the classical analogy to deal with linguistic values.

4. Estimation by Fuzzy Analogy

Fuzzy Analogy is a fuzzification of the classical analogy procedure [15]. It is also composed of three steps: identification of cases, retrieval of similar cases and cases adaptation. Each step is a fuzzification of its equivalent in the classical analogy procedure. In the following sub-sections, each step will be further detailed.

4.1 Identification of cases

The goal of this step is the characterization of all software projects by a set of attributes. Selecting attributes describing accurately software projects is a complex task in the analogy procedure. Indeed, the selection of attributes depends on the objective of the CBR system. In our case, the objective is to estimate the software project effort. Consequently, the attributes must be relevant for the effort estimation task. The problem is to detect the attributes exhibiting a significant relationship with the effort in a given environment. The solution adopted by cost estimation researchers and practitioners is to test the correlation between the effort and all the attributes for which data in the studied environment are available. This solution does not take into account attributes that can affect largely the effort, if they have not yet recorded data. Another interesting criterion is that each relevant attribute must be independent from the other attributes. In the ANGEL tool, Shepperd et al.[24, 25] propose to resolve the attributes selection problem by applying a brute force search of all possible attributes subsets. They acknowledge that this is an NP-hard search problem and consequently is not a feasible solution when the number of the candidate attributes is large. Briand et al. propose to use a t-test procedure to select the set of attributes [7]. Shepperd claimed that this is not a good solution because this procedure is not efficient to model the potential interactions between the software project attributes [17]. We strongly agree with this critic and don't believe that statistical methods can solve the selection problem in the software cost estimation field. There are two other criteria every relevant and independent attribute must obey. They are, the attribute must be comprehensive which implies that it must be well defined and the attribute must be operational which implies that it must be easy to measure. These criteria have yet not been the subject of an in-depth study in the cost estimation literature. We believe that the best way to solve the attributes selection problem is by integrating a learning procedure in the analogy approach. We discuss in Section 6 how Fuzzy Analogy can satisfy the learning criterion. Before learning, during the training phase of our approach, we adopt a variation of Shepperd's solution by allowing the estimators to use the attributes that they believe best characterize their projects and are more appropriate in their environment.

The objective of our Fuzzy Analogy approach is to deal with linguistic values. In the identification step, each software project is described by a set of selected attributes that can be measured by numerical or linguistic values. These values will be represented by fuzzy sets. In the case of a numerical value x_0 (no uncertainty), its fuzzification will be done by the membership function that takes the value of 1 when x is equal to x_0 and 0 otherwise. Let us suppose for linguistic values that we have M attributes and for each attribute V_j, a measure with linguistic values is defined (A_k^j). Each linguistic value, A_k^j , is represented by a fuzzy set with a membership function ($m_{A_k^j}$). It is preferable that these fuzzy sets satisfy the *normal condition*, i.e., they form a fuzzy partition and each of them is convex and normal [13]. The use of fuzzy sets to represent categorical data, such as 'very low' and 'low' mimics the way in which humans interpret these values and consequently it allows us to deal with vagueness, imprecision and uncertainty in the cases identification step. Another advantage of our Fuzzy Analogy approach is that it takes into account the importance of each selected attribute in the cases identification step. It is obvious that all selected attributes do not necessarily have the same influence on the software project effort. Hence, we are required to indicate the weights, u_k, associated with all selected attributes in the cases identification step.

4.2 Retrieval of similar cases

This step is based on the choice of a software project similarity measure. This choice is very important since it will influence which analogies are found. In literature, most researchers have used the Euclidean distance when projects are described by numerical data and the equality distance when they are described by linguistic values (categorical data) [24]. These two measures are not suitable when linguistic values are represented by fuzzy sets. Consequently, we have proposed a set of candidate measures for software project similarity to avoid this limitation [12]. These measures evaluate the overall similarity of two projects P_1 and P_2, $d(P_1,P_2)$, by combining the individual similarities of P_1 and P_2 associated with the various linguistic variables (attributes) (V_j) describing P_1 and P_2, $d_{v_j}(P_1,P_2)$. After an axiomatic validation of some proposed candidate measures for the individual distances $d_{v_j}(P_1,P_2)$, we have retained two measures [13]:

$$d_{v_j}(P_1,P_2)=\begin{cases}\left.\begin{array}{l}\max\limits_k \min(\grave{\imath}_{A_k^j}(P_1),\grave{\imath}_{A_k^j}(P_2))\\ max-min\ \ aggregation\end{array}\right\}(1.1)\\ \left.\begin{array}{l}\sum\limits_k \grave{\imath}_{A_k^j}(P_1)\times\grave{\imath}_{A_k^j}(P_2)\\ sum-product\ \ aggregation\end{array}\right\}(1.2)\end{cases}$$

where V_j are the linguistic variables describing projects P_1 and P_2, A_k^j are the fuzzy sets associated with V_j, and $m_{A_k^j}$ are the membership functions representing fuzzy sets A_k^j .

To evaluate the overall distance of P_1 and P_2, the individual distances $d_{v_j}(P_1,P_2)$ are aggregated using Regular Increasing Monotone (RIM) linguistic quantifiers such as '*all*', '*most*', '*many*', '*at most a*' or '*there exists*'. The choice of the appropriate RIM linguistic quantifier, *Q*, depends on the characteristics and the needs of each environment. *Q* indicates the proportion of individual distances that we feel is necessary for a good evaluation of the overall distance. The overall similarity of P_1 and P_2 , $d(P_1 , P_2)$ is given by one of the following formulas [14]:

$$d(P_1, P_2) = \begin{cases} all\ of\ (d_{v_j}(P_1, P_2)) \\ most\ of\ (d_{v_j}(P_1, P_2)) \\ many\ of\ (d_{v_j}(P_1, P_2)) \\ .... \\ there\ exists\ of\ (d_{v_j}(P_1, P_2)) \end{cases}$$

When choosing the appropriate RIM linguistic quantifier to guide the aggregation of the individual distances, its implementation is realized by an Ordered Weight Averaging operator [14, 29, 30]. The overall distance, *d(P₁ , P₂)*, is calculated by means of the following formula:

$$d(P_1,P_2) = \sum_{j=1}^{M}\left(Q\left(\frac{\sum_{k=1}^{j}u_k}{T}\right) - Q\left(\frac{\sum_{k=1}^{j-1}u_k}{T}\right)\right)\times d_{v_j}(P_1,P_2) \qquad (2)$$

where $d_{v_j}(P_1,P_2)$ is the jth largest individual distance, u_k is the importance weight associated with the kth variable describing the software project, and *T* is the total sum of all importance weights u_k which are provided in the cases identification step.

4.3 Case adaptation

The objective of this step is to derive an estimate for the new project by using the known effort values of similar projects. There are two problems here. First, the choice of how many similar projects should be used in the adaptation. Second, how to adapt the chosen analogies in order to generate an estimate for the new project. In the literature, one can notice that there is no clear rule to guide the choice of the number of analogies, *k*. Shepperd et al. have tested two strategies to calculate the number *k*, by setting it to a constant value (they explore values between 1 and 5), or by determining it dynamically as the number of projects that fall within distance (d) of the new project [17]. Briand et al. have used a single analogy [7]. Angelis and Stamelos have tested a number of analogies in the range of 1 to 10 when studying the calibration of the analogy procedure for the Albrecht's dataset [3]. The results obtained from these experimentations seemed to favour the case where *k* is lower than 3. We are not convinced by the approach of fixing the number of analogies to be considered in the case adaptation step. The principle of this approach is to take only the first *k* projects that are similar to the new project. Let us suppose that the distances between the first three projects of one dataset (P_1, P_2, P_3) and the new project (P) are respectively: 3.30, 4.00 and 4.01. When we consider *k* equal to 2, we use only the two projects P_1 and P_2 in the calculation of an estimate of P. Project P_3 is not considered in this case although there is no clear difference between d(P_2, P)=4.00, and d(P_3, P)=4.01! We believe that the use of the number *k* relies on the use of the classical logic principle: the transition from one situation (contribution in the estimated cost) to the following (no contribution in the estimated cost) is abrupt rather than gradual. In Fuzzy Analogy, we propose a new strategy to select projects that will be used in the adaptation step. This strategy is based on the distances *d(P, P$_i$)* and the definition adopted in the studied environment for the proposition '*P_i* is closely similar project to *P*'. Intuitively, *P_i* is closely similar to *P* if *d(P,P$_i$)* is in the vicinity of 1 (0 in the case of Euclidean distance). The only way to represent correctly the value 'vicinity of 1' is by using a fuzzy set defined in the unit interval [0, 1]. Indeed, this fuzzy set defines the 'closely similar' qualification adopted in the environment. Figure 1 shows a possible representation for the value 'vicinity of 1'. In this example all projects that have *d(P,P$_i$)* higher than 0.5 contribute to the estimated cost of *P*; the contribution of each *P_i* is weighted by $\mu_{\text{vicinityof 1}}(d(P,P_i))$.

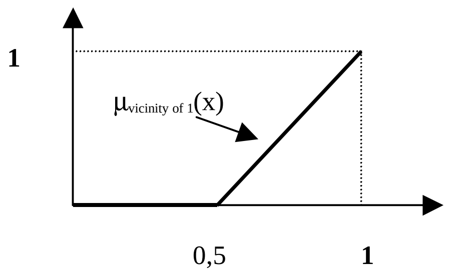

Figure 1: A possible definition of the value 'vicinity of 1'

The second problem in this step is to adapt the chosen analogies in order to generate an estimate for the new project. The most common solutions use the (weighted) mean or the median of the *k* chosen analogies. In the case of weighted mean, the weights can be the similarity distances or the ranks of the projects. For our Fuzzy Analogy approach, we use the weighted mean of all known effort projects in the dataset. The weights are the values of the membership function defining the fuzzy set 'vicinity of 1'. The formula is then:

$$Effort(P) = \frac{\sum_{i=1}^{N} m_{vicinity\ of\ 1}(d(P,P_i))\times Effort(P_i)}{\sum_{i=1}^{N} m_{vicinityof\ 1}(d(P,P_i))} \qquad (3)$$

The main advantage of our adaptation approach is that it can be easily configured by defining the value 'vicinity of 1' according to the needs of each environment. An interesting case arises when $m_{vicinityof1}(x) = x$ in Formula 3 since it gives exactly the ordinary weighted average. This property will be used in the validation of our approach on the COCOMO'81 dataset.

5. Empirical results

The following section presents and discusses the results obtained when applying the Fuzzy Analogy approach on the COCOMO'81 dataset. The calculations are made by using the F_ANGEL tool. F_ANGEL is a software prototype that we have developed to automate the Fuzzy Analogy approach. It can be seen as a fuzzification of the software ANGEL developed by Shepperd et al. for the classical analogy procedure. The results were compared with those of three other models: classical analogy, the original intermediate COCOMO'81, and 'fuzzy' intermediate COCOMO'81 [5, 11]. The accuracy of the estimates is evaluated by using the magnitude of relative error MRE defined as:

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

The *MRE* is calculated for each project in the dataset. In addition, we use the measure prediction level *Pred*. This measure is often used in the literature. It is defined by:

$$\Pr ed(p) = \frac{k}{N}$$

where *N* is the total number of observations, *k* is the number of observations with an *MRE* less than or equal to *p*. A common value for p is 0.25; in our evaluation, we use *p* equal to 0.20 as it was used for evaluation of the original version of the intermediate COCOMO'81 model. The *Pred*(0.20) gives the percentage of projects that were predicted with an *MRE* equal or less than 0.20. Other four quantities are used in this evaluation: min of *MRE*, max of *MRE*, standard deviation of *MRE* (SDMRE), and mean *MRE* (MMRE).

The original intermediate COCOMO'81 database was chosen as the basis for this validation [5]. It contains 63 software projects. Each project is described by 17 attributes: the software size measured in KDSI (Kilo Delivered Source Instructions), the project mode is defined as either 'organic', 'semi-detached' or 'embedded', and the remaining 15 cost drivers are generally related to the software environment. Each cost driver is measured on a scale composed of six qualifications: 'very low', 'low', 'nominal', 'high', 'very high' and 'extra high'. It seems that this scale is ordinal, but an analysis indicates that one of the 15 cost drivers (SCED attribute) is only assessed to be nominal. This does not cause any problem for the Fuzzy Analogy technique. Indeed, we are dealing with these six qualifications as linguistic values rather than categorical data. In the original intermediate COCOMO'81, the assignment of linguistic values to the 15 cost drivers uses conventional quantization where the values are classical intervals (see [5], pp. 119). Because of the advantages of representation by fuzzy sets over classical intervals, the 15 cost drivers should be represented by fuzzy sets. Among these, we have retained 12 attributes that we had already fuzzified. The other attributes are not studied because these relative descriptions proved insufficient [11]. In this evaluation, we assume that all the COCOMO'81 software projects are described only by these 12 cost drivers (Figure 2).

Because the original COCOMO'81 database contains only the effort multipliers, our evaluation of the Fuzzy Analogy will be made on four 'fuzzy' datasets deduced from the original COCOMO'81 database. Each one of these four 'fuzzy' datasets contains 63 projects with the real values necessary to determine the 12 linguistic values associated to each project. These 12 linguistic values are used to evaluate the similarity between software projects. One of these four fuzzy datasets is considered as an historical dataset, the other three are the current datasets containing the new projects.

Table 1 shows the results obtained using only the max-min aggregation to evaluate the individual distances (Formula (1.1)). We have not used sum-product aggregation (Formula (1.2)) for two reasons [13]:

- We have proved, under what we have called *normal condition*, that max-min and sum-product aggregations give approximately the same results. This is the case for the COCOMO'81 database.
- The sum-product aggregation does not respect all established axioms.

Normally for the overall distances, each environment must define its appropriate quantifier by studying its features and its requirements. Because a lack of knowledge concerning the appropriate quantifier for the environment from which the COCOMO'81 data was collected, we used various quantifiers: 'all', 'there exists', and α-RIM linguistic quantifiers to combine the individual similarities. An α-RIM linguistic quantifier is defined by a fuzzy set in the unit interval with membership function *Q* given by:

$$Q(r) = r^{a} \quad a > 0$$

To calculate the weights w_j's (Formula 2), we must determine the importance weights u_k's associated with the 12 variables describing COCOMO'81 software projects. We use here the productivity ratio, which is the project's productivity ratio (expressed in Delivered Source Instructions by Man-Months) for the best possible variable rating to its worst possible variable rating, assuming that the ratings for all other variables remain constant (Figure 2).

By analyzing the results of the validation of the Fuzzy Analogy technique (Table 1), we noticed that the accuracy of the estimates depends on the linguistic quantifier (a) used in the evaluation of the overall similarity between software projects. So, if we consider the accuracy measured by *Pred*(0.20) as a function of α, we can notice that, in general, it is monotonous increasing according to *a*. This is due to the fact that our similarity measures are monotonous decreasing according to α. Indeed, when α tends towards zero, this implies that the overall similarity will take into account fewer attributes among all those describing software projects. The minimum number of attributes to consider is one. This is the case when using the ‘max’ operator where the selected attribute is the one for which the associated individual distance is the maximum of all individual distances.

Table 1. Results of the evaluation of Fuzzy Analogy

		Datasets								
		Dataset #1			**Dataset #2**			**Dataset #3**		
		Pred(0.20) (%0)	***MMRE (%)***	***SDMRE (%)***	***Pred(0.20) (%)***	***MMRE (%)***	***SDMRE (%)***	***Pred(0.20) (%)***	***MMRE (%)***	***SDMRE (%)***
a-RIM	**Max**	4,76	1801,48	2902,94	4,76	2902,49	1807,17	4,76	1789,64	2865,42
	1/100	4,76	1798,85	2897,77	4,76	2894,28	1803,41	4,76	1786,20	2858,34
	1/30	4,76	1792,70	2885,74	4,76	2875,26	1794,69	4,76	1778,21	2841,96
	1/15	4,76	1783,91	2868,69	4,76	2848,44	1782,32	4,76	1766,86	2818,84
	1/10	4,76	1757,13	2851,77	4,76	2822,64	1770,06	4,76	1755,64	2796,06
	1/7	4,76	1763,86	2830,24	4,76	2788,70	1754,48	4,76	1741,29	2767,29
	1/3	6,34	1714,20	2737,66	6,34	2648,90	1687,68	6,34	1679,67	2646,43
	1	6,34	1550,89	2455,21	3,17	2258,36	1485,66	7,93	1491,93	2306,39
	3	6,34	1168,24	1889,23	9,52	1571,48	1063,19	9,52	1102,40	1694,91
	7	9,52	633,99	1215,81	14,28	830,79	526,57	9,52	603,76	968,01
	10	15,87	371.84	802,30	20,63	525,45	305,98	23,80	385,90	662,02
	15	38,09	143,92	337,76	36,50	284,20	140,68	44,44	206,19	393.30
	30	74,60	20,40	42,06	77,77	160,38	51,67	66,66	62,83	118,44
	100	92,06	4,06	9,05	84,12	30,37	10,49	87,63	23,24	76,68
	Min	92,06	4,03	9,17	87,30	29,12	8,53	88,88	21,99	77,13

Table 2. Results of the evaluation of classical analogy, ‘fuzzy’ and classical intermediate COCOMO’81[11]

	‘fuzzy’/classical intermediate COCOMO’81			**Classical Analogy (three datasets)**	
	Dataset #1	**Dataset #2**	**Dataset #3**	**K**	**Pred(0.20) %**
Pred(20) (%)	62.14 / 68	46.86 / 68	41.27 / 68	**2**	31,75
Min MRE (%)	0.11 / 0.02	0.40 / 0.02	0.06 / 0.02	**3**	25,40
Max MRE (%)	88.60 / 83.58	3233.03 / 83.58	88.03 / 83.58	**4**	19,05
Mean MRE$_i$(%)	22.50 / 18.52	78.45 / 18.52	30.80 / 18.52	**5**	12,70
Standard deviation MRE	19.69 / 16.97	404.40 / 16.97	22.95 / 16.97	**6**	12,70

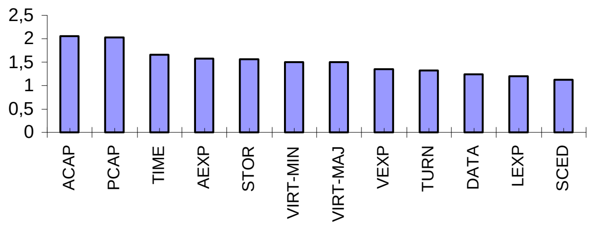

Figure 2: Comparison of the productivity ratios for the 12 variables describing the COCOMO’81 software projects

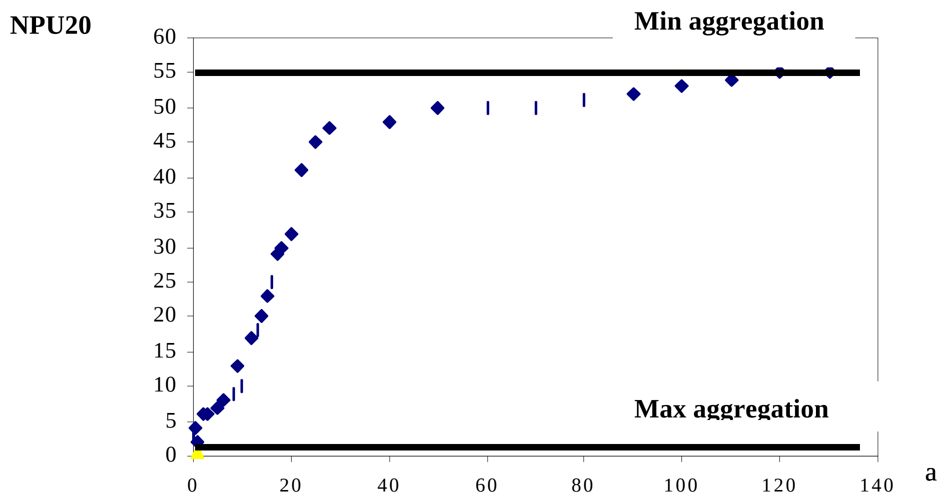

Figure 3: Relationship between a and the number of dataset #2 projects which have an MRE lower or equal than 0.20 (NPU20)

As a consequence, the overall similarity will be higher because we are more likely to find in the COCOMO'81 dataset at least one attribute for which the associated linguistic values are the same for the two projects. By contrast, when α tends towards infinity it implies that the overall similarity will take into account many attributes among all the available ones describing the software projects. As a maximum we may consider all attributes. This is the case when combining by the 'min' operator. As consequence the overall similarity will be minor because we are more likely to find in the COCOMO'81 dataset one attribute for which the associated linguistic values are different for the two projects. Here, it is very important to stress the soft aspect of our Fuzzy Analogy approach. First, we can choose the appropriate weights associated with linguistic variables describing a software project (u_k). These weights represent the importance of the variables in the environment. Second, we can choose the appropriate linguistic quantifier to combine the individual distances; this linguistic quantifier is used to generate the weights w_j. These weights represent the importance associated with the individual distances when evaluating the overall distance. They depend upon the weights u_k and the chosen linguistic quantifier. An interesting case arises if u_k is equal to w_k. This is when α is equal to 1. As a consequence, Formula 2 gives the ordinary weighted average.

Figure 3 shows the relation between α and the number of projects that have an *MRE* smaller than 0.20 (NPU20) for dataset #2. The two bold lines represent respectively the minimum and the maximum accuracy of Fuzzy Analogy when it uses the min and the max aggregation to combine individual similarities. The 'max' ('min') aggregation gives lower (higher) accuracy because it considers only one (all) attribute(s) in the evaluation of the similarity. For the other α-RIM linguistic quantifiers ($0<\alpha<\infty$), the accuracy increases with α because additional attributes will be considered in the evaluation of the overall similarity. For example, a software project P_i which has an overall similarity with P different from zero when α is equal to 10 , may have a null overall similarity when α is equal to 30. Because of that, it is not used in the estimation of the cost for α=30. When α tends towards infinity (this implies that most attributes are considered in the evaluation of the similarity), only projects P_i which are closely similar to P will contribute in the estimate of the cost of P. This is in conformity with common knowledge in the cost estimation field: *evaluation of the similarity between projects is meaningful if they are described by a sufficient number of attributes*. As we can see in Figure 3, the accuracy is a monotonous increasing function of α. However, because Formula 3 used in the adaptation step is not monotonous increasing, we may observe certain anomalies that can lead to misinterpretations of the results. This is the case when α is equal to 1 in dataset #2. It seems that the accuracy when α =1/3 (NPU20=4) is better than that when α=1 (NPU20=2). Indeed, the two additional projects, which have an *MRE* lower than 0,20 (18,74 and 17,68) for α=1/3, have respectively an *MRE* equal to 21,68 and 20,24 when α=1. So, when we have fixed *Pred*(p) at *Pred*(0,20), these two project are not counted. This should not give the impression that the case for α=1/3 generates accurate estimates than the case for α=1. By analysing the results for all projects, we have found the opposite (see the mean and the standard deviation of the *MRE* for α=1/3 and α=1).

We compared the results of the Fuzzy Analogy with the other three techniques in regard to two criteria: the type of the technique and whether or not the technique uses fuzzy logic in its estimation process. Our findings were the following:

- Fuzzy analogy performs better than the classical analogy in all three datasets when α is higher than a given value. In the classical analogy, we have used the classical equality distance (equal or not) in the evaluation of similarity between projects. All attributes are considered in this evaluation. The best accuracy was obtained when we consider only the two first projects in the adaptation step (*Pred*(0,20)=31,75). The Fuzzy Analogy when using the 'min' aggregation also took into account all attributes in the evaluation of projects similarity. Its accuracy was much higher than that for classical analogy. Two advantages were found when using fuzzy logic with the estimation by analogy. First, it tolerates imprecision and uncertainty in its inputs (cost drivers) and consequently it generates gradual outputs (cost). This is why Fuzzy Analogy gives closer results for the three datasets while classical analogy generates the same or significantly different outputs when the inputs are different (this is the same case between 'fuzzy' and classical intermediate COCOMO'81, see [11] for more details, Table 2). Second, it improves the accuracy of the estimates because our similarity measures are more appropriates than those used in the literature.
- Intermediate COCOMO'81 generates more accurate results than classical analogy but when integrating fuzzy logic in the estimation by analogy procedure, the Fuzzy Analogy performs better than intermediate COCOMO'81. This proves that fuzzy logic is the appropriate tool to deal with linguistic values rather than the classical logic (Aristote logic) used in the original version of the COCOMO'81.

Taking into account these results, we suggest the following ranking of the four techniques in terms of accuracy and adequacy to deal with linguistic values:

1- Fuzzy Analogy
2- Fuzzy intermediate COCOMO'81
3- Classical intermediate COCOMO'81
4- Classical analogy.

6. Discussion and future improvements

In this paper, we have proposed a new approach to estimate the software project effort. This approach is based on reasoning by analogy, fuzzy logic and linguistic quantifiers. Such an approach can be used when the software projects are described by linguistic and/or numerical values. Thus, our approach improves the classical analogy procedure that does not take into account linguistic values. In the Fuzzy Analogy approach, both linguistic and numerical data are represented by fuzzy sets. The advantage of this is to handle correctly the imprecision and the uncertainty when describing a software project. Also, by using RIM linguistic quantifier to guide the aggregation of the individual similarities between two projects, the Fuzzy Analogy approach can be easily adapted and configured according to the specifications of each environment. Also, we have validated the Fuzzy Analogy by using the COCOMO'81 dataset. The results of this validation were compared to those of the classical analogy approach, 'fuzzy' intermediate COCOMO'81 and original intermediate COCOMO'81. We can conclude that fuzzy logic improves the estimation process and consequently generates more accurate estimates.

By using fuzzy logic in its estimation process, our approach satisfies the first criterion of the concept Soft Computing, which is the tolerance of imprecision. As defined by Zadeh [32], Soft Computing is composed of three criteria part of human nature: tolerance of imprecision, learning, and uncertainty.

In this work, we have introduced some learning functionalities in our approach. In the identification step, we can update all information concerning the linguistic variables describing software projects, specifically, theirs linguistic values which depend on humans judgement. For example, the linguistic value 'high' for software reliability may mean that the number of software failures is lower than 6 per month, but in the future, we may require less than 3 software failures per month to evaluate it as 'high'. In the case retrieval step, we can update the definition of the linguistic quantifier used in the environment. Here also, the meaning of a linguistic quantifier depends on human judgement. However, other learning characteristics that are not included in our approach remain. For example, the Fuzzy Analogy must be able to provide its user with a subset of linguistic variables that have always led to accurate estimates in the past. We may then use this sub-set in the identification step. Thus, the selection attributes problem can be solved. Also, Fuzzy Analogy must be able to propose the appropriate linguistic quantifier to be used in retrieval step by using those that have often led to accurate estimates.

In order to satisfy the third criteria of Soft Computing, Fuzzy Analogy must be able to handle the uncertainty when estimating the cost of the new project. Estimate uncertainty occurs because an estimate is a probabilistic assessment of a future condition. Kitchenham and Linkman have examined four sources of estimate uncertainty: model error, measurement error, assumption error and scope error [16]. In our case, we are concerned by the first source of uncertainty. Fuzzy Analogy is based on the affirmation: '*similar projects have similar costs*'. There are two sources of uncertainty in this affirmation. First, the consequence of this affirmation is imprecise. Second, the affirmation '*similar projects have similar* *costs'* is not always deterministic. We can find in some applications of CBR cases that are similar but the outcomes are completely different. It seems that it can be the case in the cost estimation field. Indeed, no cost estimation model can include all the factors that affect the cost required to develop the software. So, the factors that affect cost and are not included explicitly in the evaluation of the similarity between projects contribute to the uncertainty in the predicted cost. In order to take into account the uncertainty of the classical affirmation of CBR, we may replace it by the following *'similar projects have **possibly** similar costs'*. Further research work has been initiated to look at the use of this affirmation as the basics of an improvement of our approach.

7. References

[1] A. Aamodt, E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches", AI Communications, IOS Press, Vol. 7:1. 1994, pp. 39-59

[2] A. Abran, P.N. Robbillard, "Functions points analysis: an empirical study of its measurement processes", IEEE Trans. On Software Enginnering, 22(12): 1996, pp. 895-909

[3] L. Angelis, I. Stamelos, "A Simulation Tool For Efficient Analogy Based Cost Estimation", Empirical Software Engineering, Vol. 5, no. 1, 2000, pp. 35-68

[4] L. Angelis, I. Stamelos, M. Morisio, "Building a Software Cost Estimation Model Based on Categorical Data", 7th Int, Soft. Metrics Symp., IEEE computer Society, London, April, 2001, pp. 4-15.

[5] B.W. Boehm, *Software Engineering Economics*, Prentice-Hall, 1981.

[6] B.W. Boehm, and *al.*, "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0", *Annals of Software Engineering on Software Process and Product Measurement*, Amsterdam, 1995.

[7] L. Briand, T. Langley, I. Wieczorek., "Using the European Space Agency data set: A replicated assessment and comparison of common software cost modeling", In Proc 22th IEEE International Conference on Software engineering, Limerik, Ireland, 2000, pp. 377-386

[8] D.S. Chulani, "Incorporating Bayesian Analysis to Improve the Accuracy of COCOMO II and Its Quality Model Extension", Ph.D. Qualifying Exam Report, USC, February, 1998.

[9] N. Fenton, and S.L. Pfleeger, *Software metrics: A Rigorous and Practical Approach*, International Computer, Thomson Press, 1997.

[10] Gulezian R., 'Reformulating and Calibrating COCOMO' Journal Systems Software, Vol 16, 1991, pp.235-242

[11] A. Idri, L. Kjiri, and A. Abran, "COCOMO Cost Model Using Fuzzy Logic", *7th Intenational Conference on Fuzzy Theory & Technology*, Atlantic City, NJ, February, 2000. pp. 219-223

[12] A. Idri, and A. Abran, "Towards A Fuzzy Logic Based Measures For Software Project Similarity", *Sixth Maghrebian Conference on Computer Sciences, Fes*, Morroco, November, 2000. pp. 9-18

[13] A. Idri, and A. Abran, "A Fuzzy Logic Based Measures For Software Project Similarity: Validation and Possible Improvements", 7th International Symposium on Software Metrics, *IEEE computer society*,4-6 April, England, 2001. pp. 85-96

[14] A. Idri, and A. Abran, "Evaluating Software Project Similarity by using Linguistic Quantifier Guided Aggregations", 9th IFSA World Congress/20th NAFIPS International Conference, 25-28 July, Vancouver, 2001. pp. 416-421

[15] A. Idri, A. Abran, T. M. Khoshgoftaar, "Fuzzy Analogy: A new Approach for Software Cost Estimation", 11th International Workshop in Software Measurements, 28-29 August, Montreal, 2001, pp. 93-101

[16] B. Kitchenham, S. Linkman, "Estimates, uncertainty and risks", IEEE Software, 14(3), 1997, pp. 69-74

[17] G. Kadoda. M. Cartwright, L. Chen, M. Shepperd, "Experiences Using Case-Based Reasoning to Predict Software Project Effort", EASE, Keele, UK, 2000, p.23

[18] J.L. Kolodner, Case-Based Reasoning, Morgan Kaufmann, 1993

[19] D. H. Krantz, R. D. Luce, P. Suppes, A. Tversky, "Foundations of Measurement: Additive and Polynomial Representations ", Academic Press, Volume 1, 1971

[20] Matson J., E, Barrett B., E, Mellichamp J., M, 'Software Development Cost Estimation Using Function Points', IEEE, Vol. 20, No. 4, Apr., 1994, pp. 275-287

[21] I. Myrtveit, E. Stensrud, "A Controlled Experiemnt to Assess the Benefits of Estimating with Analogy and Regression Models", IEEE Transaction on Software Engineering, 25, 4, July/August, 1999, pp. 510-525

[22] F. Niessink, H. Van Vliet "Predicting Maintenance Effort with Function Points", in Proc Inter. Conf. On Soft. Maintenance, Bari, Italy, IEEE Computer Society, 1997.

[23] Putnam L. H, ' A General Empirical Solution to the Macro Software Sizing and Estmation Problem', IEEE Tronsactions on Soft. Eng., Vol. SE-4, No. 4, July, 1978.

[24] M**.** Shepperd, C. Schofield, and B. Kitchenham, "Effort Estimation using Analogy", ICSE-18, Berlin, 1996, pp. 170-178

[25] M. Shepperd, and C. Schofield, "Estimating Software Project Effort Using Analogies", IEEE Trans. on Software Engineering, Vol. 23, no. 12, November, 1997, pp. 736-743

[26] S. Shepperd, G. Kadoda, "Using simulation to evaluate predictions systems", 7th International Symposium on Software Metrics, IEEE computer society, 4-6 April, England, 2001. pp. 349-358

[27] S. S. Stevens, "On the Theory of scales and Measurement", Science 103, 1946, pp. 677-680

[28] S. Vicinanza, and M.J. Prietolla, "Case Based Reasoning in Software Effort Estimation", Proceedings 11th Int. Conf. on Information Systems, 1990

[29] R.R. Yager, and J. Kacprzyk, The Ordered Weighted Averaging Operators: Theory and Applications" Kluwer: Norwell, MA, 1997.

[30] R.R. Yager, "Quantifier Guided Aggregation using OWA Operators", International Journal of Intelligent Systems, 11, 1996, pp.49-73

[31] L.A. Zadeh, "Fuzzy Set", Information and Control, Vol. 8, 1965, pp. 338-353

[32] L.A. Zadeh, "Fuzzy Logic, Neural Networks, and Soft Computing", Comm. ACM, Vol. 37, no. 3, March, 1994, pp.77-84

[33] H. Zuse, "A Framework of Software Measurement", de Gruyter, 1998.