Towards A Fuzzy Logic Based Measures for Software Projects Similarity

Ali IDRI and Alain ABRAN

Research Lab. In Software Engineering Management
Department of computer Science
UQÀM, P.O Box. 8888, Succ. Centre-Ville, Montréal,
Canada
E-mail: idri@ensias.um5souissi.ac.ma
alain.abran@uqam.ca

Abstract

The attribute of similarity of software projects has not been the subject of in-depth studies even tough it is often used when estimating software development effort by analogy. Most of the proposed measures of projects attribute are described by numerical variables (interval, ratio or absolute scale). However, in practice many factors which describe software projects, such as the experience of programmers and the complexity of modules, are measured on the basis of an ordinal scale composed of qualifications such as 'very low' and 'low'. Many of these qualifications (linguistic values in fuzzy logic) of these attributes can also be represented by fuzzy sets. This will enable us to measure the similarity between software projects which are described by linguistic values (ordinal scale). Furthermore, the proposed measures can, of course, be also used when projects are described by numerical values.

Keywords*: Software Metrics, , Similarity, Fuzzy Logic.*

INTRODUCTION

As in other sciences, the similarity between two software projects, which are described and characterised by a set of attributes, is evaluated by measuring the distance between these two projects through their sets of attributes. Thus, two projects are not similar if the differences between their set of attributes are obvious. It is important to note that the similarity between two projects depends also on their environment: projects which are similar in a specific type of environment may not necessarily be similar in other environments. So, according to Fenton's definitions [5], similarity will be considered as an external product attribute and consequently it can be measured only indirectly.

In the software measurement literature, similarity is often used when estimating effort by analogy; various authors have put forward various proposal for means of deriving similarity as input to the estimation process, and in particular [12] who has proposed a variety of approaches including a number of preference heuristics. The nearest neighbour algorithms is one such approach; it is based upon straightforward Euclidean distance, where the variables are numeric, or the sum of squares of the differences for each variables, where the variables are categorical. One such type of algorithm is described in [13,14, 15]:

$$d(P_1, P_2, V) = \frac{1}{\sum_{v_j} d_{v_j}(P_1, P_2)} \qquad (1)$$

where P_1 and P_2 are the software projects, V is the set of the variables Vj which describe the projects P_1 and P_2,

$$d_{v_j}(P_1, P_2) = \begin{cases} (V_j(P_1) - V_j(P_2))^2 & (2.1) \\ \begin{cases} 0 & (2.2) \\ 1 & (2.3) \end{cases} \end{cases}$$

where 2.1) the variables are numeric; therfore, the d(P_1, P_2, V) will be the Euclidean distance, 2.2) if the variables are categorical and $V_j(P_1) = V_j(P_2)$ or 2.3) if the variables are categorical and $V_j(P_1) \neq V_j(P_2)$; so the d(P_1, P_2, V) will be the equality distance.

Shepperd et al.[14,15], while exa*min*ing a simple similarity measures, found three major inadequacies. First, they are computationally intensive. Consequently, many Case-Based Reasoning systems have been developed such as ESTOR[16] and ANGEL[14,15]. Second, the algorithm are intolerant of noise and of irrelevant features. To overcome this problem, they propose that the algorithms must learn from successful and unsuccessful predictions. Third, and most critical, they cannot handle categorical data others than binary valued variables. However, in software metrics, specifically in software cost estimation models, many factors (linguistic variables in fuzzy logic), such as the experience of programmers and the complexity of modules, are measured on an ordinal scale composed of qualifications such as 'very low' and 'low' (linguistic values in fuzzy logic). For example, in the COCOMO'81 model (respectively the COCOMO II model) 15 attributes among 17 (respectively 22 among 24) are measured with six linguistic values ('very low', 'low', 'nominal', 'high', 'very high', 'extra-high') [2,3,4]. As another example, in the size measurement method of Function Points, the assignment of the level of complexity for each item (input, output, inquiry, logical file, or interface) uses three qualifications ('simple', 'average', 'complex') and the calculation of the General System Characteristics is based on 14 attributes measured on an ordinal scale (from 'irrelevant' to 'essential') [8].
In this work, an alternative approach is proposed to deal with this limitation of categorical variables. For example, the similarity between projects will be measured using a new measure based on fuzzy logic when some of the attributes of projects are described by categorical values such as 'low', 'very low' and 'high'.

This paper is organized as follows. In the first section, we briefly outline the principles of the fuzzy logic. In the second section, we explain why the existing measures of similarity cannot be used when the projects attributes are described by categorical values; this leads next to proposal of new measures based on fuzzy logic to overcome such limitations. In the third section, we illustrate the use of our measures by evaluating the similarity between projects in the COCOMO'81 database. This illustration will enable us to deduce many remarks about the validation of our measures. A conclusion and an overview of future work concludes this paper.

1. FUZZY LOGIC

According to Zadeh [26], the term Fuzzy Logic (FL) is currently used in two different senses. In a narrow sense, FL is a logical system that aims at a formalisation of approximate reasoning. In a broad sense, FL is almost synonymous with fuzzy set theory. Fuzzy set theory, as its name suggests, is basically a theory of classes with unsharp boundaries. It is considered as an extension of the classical set theory. The membership function $\mu_A(x)$ of x of a classical set A, as subset of the universe X, is defined by:

$$m_A(x) = \begin{cases} 1 & iff\ x \in A \\ 0 & iff\ x \notin A \end{cases}$$

This means that an element x is either a member of set A ($\mu A(x)=1$) or not ($\mu A(x)=0$). Classical sets are also referred to as crisp sets. For many classifications, however, it is not quite clear whether x belongs to a set A or not. For example in [10], if set A represents PCs which are too expensive for student's budget, then it is obvious that this set has no clear boundary. Of course, it could be said that PC priced at $2500 is too expensive, but what about a PCs priced at $2495 or $2502? Are these PCs too expensive or not? Clearly, a boundary could be determined above which a PC is too expensive for the average student, say $2500, and a boundary which a PC is certainly not too expensive, say $1000. Between those two boundaries, however there remains an interval in which it is not clear whether a PC is too expensive or not. In this interval, a grade could be used to classify the

price as partly too expensive. This is where fuzzy sets come in: sets in which the membership has grades in the interval (0,1). The higher the membership x has in fuzzy set A, the more true it is that x is A.

A fuzzy set, introduced by Zadeh is a set with graded membership in the real interval (0,1). It is denoted by:

$$A = \int_X m_A(x) / x$$

where $\mu_A(x)$ is know as the membership function and X is know as the universe of discourse. The Figure 1 shows two representations of the linguistic value 'too expensive'; the first using a fuzzy set (Figure 1 (a)) and the second using a classical set (Figure 1 (b)). The greater advantage of the fuzzy set representation is that it is gradual function rather than an abrupt step function between the two boundaries of $1000 and $2500.

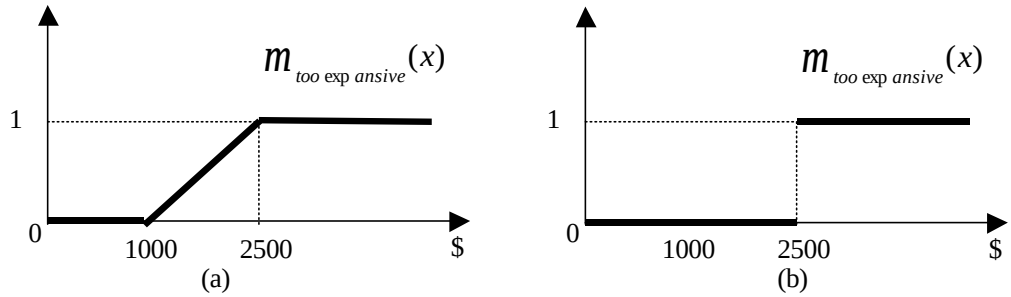

Figure 1. Fuzzy set (a) and Classical set (b) for the linguistic value 'too expensive

Operations on fuzzy sets

As for classical sets, operations are defined on fuzzy sets such as intersection, union, complement, etc. In the literature on fuzzy sets, a large number of possible definitions are proposed to implement intersection, union and complement. For example, general forms of intersection and union are represented by *triangular norms* (T-norms) and *triangular conorms* (T-conorms or S-norms), respectively. A T-norm is a two-place function from $[0,1]\times[0,1]$ to [0,1] satisfying the following criteria [10]:

T-1 T(a, 1) = a
T-2 T(a, b) ≤ T(c, d), whenever a≤c, b≤d
T-3 T(a, b) = T(b, a)
T-4 T(T(a, b),c) = T(a, T(b, c))

The conditions defining a T-conorm (S-norm) are, besides T-2, T3 and T-4:

S-1 S(a, 0) = a

The complement ¬A of fuzzy set A is defined by:

C-1 C(0) = 1
C-2 C(a) < c(b), whenever a>b
C-3 C(C(a)) = a

The following table gives examples of the operators used most often:

T-norm	$\mu_{A\cap B} = \min(\mu_A(x), \mu_B(x))$ $\mu_{A\cap B} = \mu_A(x) \times \mu_B(x)$ $\mu_{A\cap B} = \max(\mu_A(x)+\mu_B(x)-1, 0)$
S-norm	$\mu_{A\cup B} = \max(\mu_A(x),\mu_B(x))$ $\mu_{A\cup B} = \min(\mu_A(x)+\mu_B(x)-1, 0)$
Complement	$\mu_{\neg A}(x) = 1-\mu_A(x)$ $m_{\neg A} = \frac{1-m_{\neg A}(x)}{1+ l m_{\neg A}(x)} \quad l > -1$

Table 1. Examples of T-norm, S-norm and complement operators

Among the others branches of fuzzy set theory, there is fuzzy arithmetic, fuzzy graph theory, and fuzzy data analysis.

2. SIMILARITY MEASURES

The goal is to measure the similarity between two software projects P_1 and P_2 when their attributes are described by categorical values. The classical distance formula (1) should be rejected because:

- The Euclidean distance formula requires values that are numerical rather than sets;
- In software measurement theory, the use of the Euclidean distance in an ordinal scale is insignificant [5,6];
- The equality distance is used when the values are classical intervals rather than fuzzy sets;
- The equality distance is not precise and can hide a great difference between two similar projects [7]

Let projects P_1 and P_2 be described by M linguistic variables (V_j). Then for each linguistic variable 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 ($m_{A_k}^{v_j}$) rather than by a classical interval. The advantages of this representation are:

- It is more general;
- It mimics the way in which humans interpret linguistic values;
- The transition for one linguistic value to a contiguous linguistic value is gradual rather than abrupt.

Using such membership functions, new measures of similarity can be proposed. These measures will operate on two levels:

- Measurement of similarity between two projects according to only one dimension at a time (one variable V_j), $d_{v_j}(P_1, P_2)$.
- Measurement of similarity between two projects according to all dimensions (all variables V_j), $d(P_1, P_2, V)$ or $d(P_1, P_2)$.

Two steps are therefore required for measuring similarity:

First step: Projects similarity according to one dimension, $d_{v_j}(P_1, P_2)$

The first step consists in calculating the similarity between P_1 and P_2 according to each individual attribute with a linguistic variable V_j, $d_{v_j}(P_1, P_2)$. Since each V_j is measured by fuzzy sets, the distance $d_{v_j}(P_1, P_2)$ must express the fuzzy equality according to V_j between P_1 and P_2. This fuzzy equality is a fuzzification of classical equality. So, we must define a fuzzy set which can reflect this. This fuzzy set must have a membership function with two variables ($V_j(P_1)$ and $V_j(P_2)$). This type of fuzzy sets are referred in the fuzzy set theory as a *fuzzy relation*. Such a fuzzy relation can represent an association or correlation between elements of the product space.

In our case, the association that will be represented by this fuzzy relation is the statement *P_1 and P_2 are approximately equal according to Vj*. We note this fuzzy relation by $R_{\approx}^{v_j}$. The problem is how to define the membership function, $\mu_{R_{\approx}^{v_j}}$, associated to the fuzzy relation $R_{\approx}^{v_j}$. $d_{v_j}(P_1, P_2)$ will be exactly $\mu_{R_{\approx}^{v_j}}(P_1, P_2)$.

Intuitively, the equality between two projects P_1 and P_2 is not null if these two projects have a degree of membership different from 0 to at least one same fuzzy set of Vj :

$$d_{v_j}(P_1, P_2) \neq 0 \quad \textit{iff} \quad \exists A_k / \mu_{A_k}^{V_j}(P_1) \neq 0 \textit{ and } \mu_{A_k}^{V_j}(P_2) \neq 0$$

To illustrate this, we use the following figure:

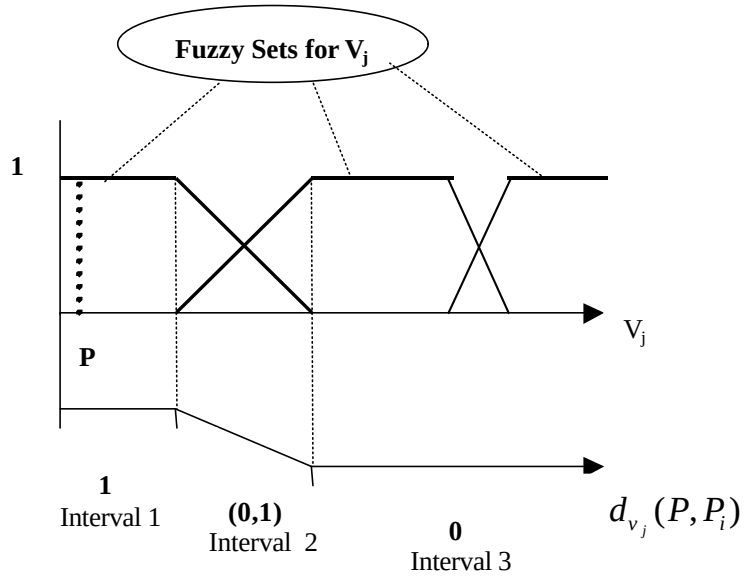

Figure 2. An explanatory example of $d_{v_j}(P, P_i)$

- $d_{v_j}(P, P_i)$ must be equal to 1 if $V_j(P_i)$ is in Interval 1;
- $d_{v_j}(P, P_i)$ must be decreasing strictly from 1 to 0 if $V_j(P_i)$ is in Interval 2;
- $d_{v_j}(P, P_i)$ must be equal to 0 if $V_j(P_i)$ is in Interval 3.

From this, it can be deduced that the fuzzy relation $R_{\approx}^{v_j}$ is a combination of a set of fuzzy relations $R_{\approx,k}^{v_j}$. Each $R_{\approx,k}^{v_j}$ represents the equality of Vj according to one of its linguistic value A_k^j. Indeed, $R_{\approx,k}^{v_j}$ represents the fuzzy *if-then* rule where the premise and the consequence consist of fuzzy propositions:

$$R_{\approx,k}^{v_j} : \textit{if } V_j(P_1) \textit{ is } A_k^j \textit{ then } V_j(P_2) \textit{ is } A_k^{v_j}$$

Hence, for each variable V_j, we have a rule base (RBASE_Vj) that contains the same number of fuzzy *if-then* rules as the number of the fuzzy sets defined for V_j. Each RBASE_Vj express the fuzzy equality between two software projects according to Vj, $d_{v_j}(P_1, P_2)$. When we consider all variables V_j, we obtain a rule base (RBASE) that contain all rules associated to all variables. RBASE expresses the fuzzy equality between two software projects according to all variables Vj, $d(P_1, P_2)$. The number of rules included in RBASE is given by:

$$card(RBASE) = \sum_{j=1}^{M} R(RBASE\_V_j)$$

where R(RBASE_Vj) is the number of rules associated to V_j.

To define $d_{v_j}(P_1, P_2)$, we must combine all fuzzy rules in DBASE_Vj to obtain one fuzzy relation ($R_{\approx}^{v_j}$) that represent DBASE_Vj. This combining of fuzzy if-then rules $R_{\approx,k}^{v_j}$ into a fuzzy relation $R_{\approx}^{v_j}$ is called *aggregation*. The way this is done is different for various types of fuzzy implication functions adopted for the fuzzy rules. These fuzzy implication functions are based on distinguishing between two basic types of implications[10]:

- Fuzzy implication which comply with the classical conjunction (where $R_{\approx,k}^{v_j}$ is defined by $(A_k^j \cap A_k^j)$). In this case, the aggregation operator is a disjunction:

$$R_{\approx}^{v_j} = \cup R_{\approx,k}^{v_j} = \cup (A_k^j \cap A_k^j)$$

- Fuzzy implication which comply with the classical implication (where $R_{\approx,k}^{v_j}$ is defined by $(\neg A_k^j \cup A_k^j)$). In this case, the aggregation operator is a conjunction:

$$R_{\approx}^{v_j} = \cap R_{\approx,k}^{v_j} = \cap (\neg A_k^j \cup A_k^j)$$

Using this basic distinction of two types of fuzzy implications a number of "compositions" can be defined. In our work, we use only the two basic types of fuzzy implications. So, we must choose the operators describing intersection, union and complement. For fuzzy implications based on classical conjunction, we uses an S-norm, such as the *max* or the *sum* operators for the aggregation of the relation $R_{\approx,k}^{v_j}$ and a T-norm, such as the *min* (Mamdani implication) or the *product (Larsen implication)* operators as the implication function for the fuzzy rules. Hence, the membership function of $R_{\approx}^{v_j}$ is:

$$m_{R_{\approx}^{v_j}}(P_1, P_2) = \begin{cases} \max\limits_k \min(m_{A_k}^{v_j}(P_1), m_{A_k}^{v_j}(P_2)) \\ \max - \min \ aggregation \\ ou \\ \sum\limits_k m_{A_k}^{v_j}(P_1) \times m_{A_k}^{v_j}(P_2) \\ sum - product \ aggregation \end{cases} \quad (3)$$

For fuzzy implications based on classical implication, we uses the *min* operator as T-norm for the aggregation of the rules $R_{\approx,k}^{v_j}$ and the *max* operator as S-norm for the implication function for the fuzzy rules (Kleene-Dienes implication). The membership function of $R_{\approx}^{v_j}$ is then:

$$m_{R_{\approx}^{v_j}}(P_1, P_2) = \begin{cases} \min\limits_k \max(1 - m_{A_k}^{v_j}(P_1), m_{A_k}^{v_j}(P_2)) \\ \min \text{ - Kleene - Dienes aggregation} \end{cases} \quad (4)$$

We take $d_{v_j}(P_1, P_2)$ equal to the membership function $m_{R_{=}^{v_j}}(P_1, P_2)$. We can justify this by the definition of the $m_{R_{=}^{v_j}}(P_1, P_2)$: the higher it is, the more true is the proposition that *P_1 and P_2 are approximately equal according to Vj*. When $d_{v_j}(P_1, P_2)$ is equal to 1, this implies a perfect similarity between P_1 and P_2 according to Vj; equal to 0, a total absence of similarity; between 0 and 1, a partial similarity.

Second step: Projects similarity according to all dimensions, d(P_1, P_2)

We calculate the distance d(P_1, P_2) from the various distances $d_{v_j}(P_1, P_2)$:

$$d(P_1, P_2) = F(d_{v_1}(P_1, P_2), ..., d_{v_M}(P_1, P_2))$$

where M is the number of variables describing the projects P_1 and P_2. Because the various $d_{v_j}(P_1, P_2)$ are a membership functions associated to fuzzy relations $R_{=}^{v_j}$, we define F as one of the three operators: *min* as a T-norm, m*ax* as an S-norm and the *i-or* operator as hybrid between both a T-norm and a S-norm. The *i-or* operator was introduced to establish the equality between artificial neural networks (ANN) and the fuzzy rule-based systems (FBRS) [2]. Because, in our case, the formula given in [2] can generate undefined values, we have adopted a modification in order to avoid this. The three formulas obtained are:

$$d(P_1, P_2) = \begin{cases} \min(d_{v_1}(P_1, P_2), ..., d_{v_M}(P_1, P_2)) \\ \max(d_{v_1}(P_1, P_2), ..., d_{v_M}(P_1, P_2)) \\ i-or(d_{v_1}(P_1, P_2), ..., d_{v_M}(P_1, P_2)) = \begin{cases} 0 & \exists k, h / d_{v_k}(P_1, P_2) = 1 \; and \; d_{v_h}(P_1, P_2) = 0 \\ \dfrac{\prod_{j=1}^{M} d_{v_j}(P_1, P_2)}{\prod_{j=1}^{M}(1 - d_{v_j}(P_1, P_2)) + \prod_{j=1}^{M} d_{v_j}(P_1, P_2)} & otherwise \end{cases} \end{cases} \quad (5)$$

The Figure summarizes the computing process of the various distances $d_{v_j}(P_1, P_2)$ and the distance d(P_1, P_2).

3. ILLUSTRATION AND DISCUSSION

In this section, we illustrate, by an example, the computing process of the various distances defined in the previous section. The intermediate version of the COCOMO'81 database was chosen as the basic for this example [2].

The original intermediate COCOMO'81 database contains 63 projects. Each project is described by 17 attributes: the software size is measured in KDSI[1], the project mode is defined as either organic, semi-detached or embedded and 15 other cost drivers which are generally related to the software environment (Appendix 1, Table 1). Each cost driver is measured using a rating scale of six linguistic values: 'very low', 'low', 'nominal', 'high', 'very high' and 'extra-high' (Appendix 1, Table 2). The assignment of linguistic values to the cost drivers (or project attributes) uses conventional quantization where the values are intervals (see [3], pp. 119

[1] Kilo Delivered Source Instructions

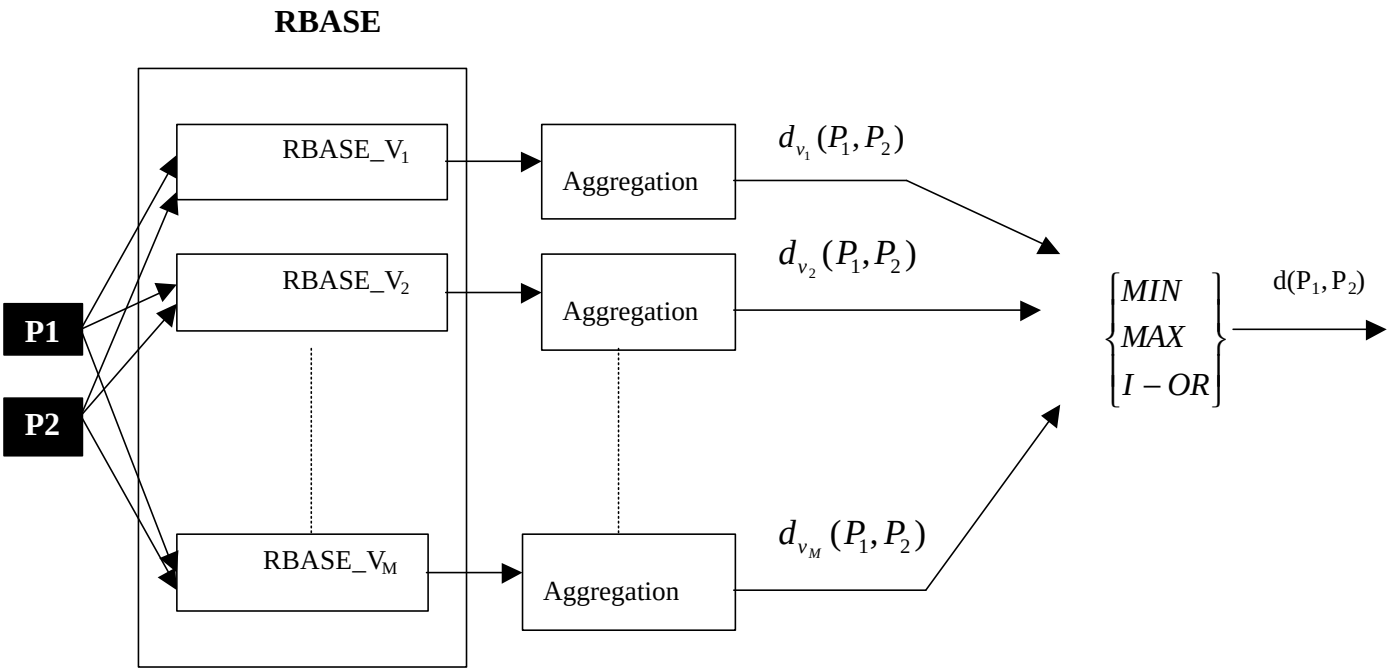

Figure 3. The computing process of the distances $d_{v_j}(P_1,P_2)$ and d(P_1, P_2)

For example, the DATA cost driver is measured by the following ratio:

$$\frac{D}{P} = \frac{Database\ size\ in\ bytes\ or\ characters}{Program\ size\ in\ DSI}$$

Then, a linguistic value is assigned to the DATA according to the following table:

Low	Nominal	High	Very High
D/P<10	10≤D/P<100	100≤D/P<1000	D/P≥1000

Table 2. DATA cost driver ratings.

For the measurement of similarity using project attributes described by categorical values, fuzzy sets must be defined for the 15 cost drivers. For example, in the case of the DATA cost driver, we have defined a fuzzy set for each linguistic value with a trapezoid-shaped membership function μ (Figure 4). For the other cost drivers of the intermediate COCOMO'81 database, we proceed in the same way as for DATA. Among its 15 cost drivers, the four factors RELY, CPLX, MODP and TOOL are not studied because these relative descriptions are insufficient. So, we takes 12 cost drivers that we have already fuzzified [7]

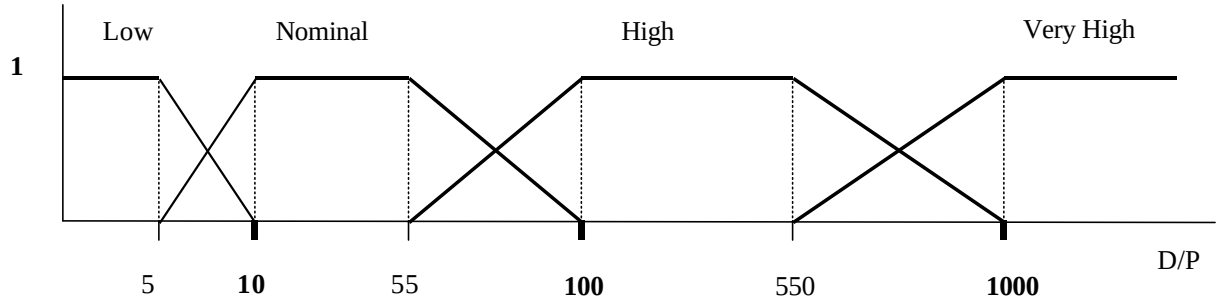

Figure 4. Membership functions of fuzzy sets defined for the DATA cost driver

To simplify the illustration, we calculate the similarity between the first project (P_1) and the first ten projects (P_1, P_2, …, P_{10}) of the COCOMO'81 database. Because our measures are computationally intensive, we have developed a specific prototype to automate the calculations. This prototype uses Excel to store data and Visual Basic to implement the various processing. The following table shows the results obtained for the similarity measured by *max-min*, *sum-product* and *Kleene-Dienes* aggregations (formulas 3, 4 and 5).

	P_1								
	Max-min aggregation $d_{v_j}(P_1, P_i)$			**Sum-product aggregation** $d_{v_j}(P_1, P_i)$			**Kleene-Dienes aggregation** $d_{v_j}(P_1, P_i)$		
	$d(P_1, P_i)$			**$d(P_1, P_i)$**			**$d(P_1, P_i)$**		
	Min	***Max***	***i-or***	***Min***	***Max***	***i-or***	***Min***	***Max***	***i-or***
P_1	0.5214	1	1	0,5009	1	1	0.5214	1	1
P_2	0	1	0	0	1	0	0	1	0
P_3	0	1	0	0	1	0	0	1	0
P_4	0	1	0	0	1	0	0	1	0
P_5	0	0.8409	0	0	0.8070	0	0	0.8409	0
P_6	0	0.9331	0	0	0.9331	0	0	0.9331	0
P_7	0	0.9331	0	0	0.9331	0	0	0.9331	0
P_8	0	0	0	0	0	0	0	0.4785	0
P_9	0	0.5679	0	0	0.4926	0	0	0.5679	0
P_{10}	0	1	0	0	1	0	0	1	0

Table 3. Results obtained for $d(P_1, P_i)$ when $d_{v_j}(P_1, P_i)$ uses the three types of aggregation.

$d(P_1, P_i)$ combines the various $d_{v_j}(P_1, P_i)$ in three ways:

- First, is the 'and' logical operation by using the *min* operator; so, the distance between two projects, $d(P_1, P_i)$, is null if only one $d_{v_j}(P_1, P_i)$ is the same one; it is equal to 1 if all $d_{v_j}(P_1, P_i)$ are the same ones. This imply that all variables Vj describing the projects must be significant and independent. In practice, the characterization of the projects uses often available features which can be irrelevant, dependent or insufficient. So, for correct use of the *min* operator, practitioners must analyze all variables Vi, by using for example the PCA[2] method, in order to satisfy the independence and the significance tests. From the results obtained (Min columns), we can notice that there is no significance difference between $d(P_1, P_i)$ for all types of aggregation. $d(P_1, P_i)$ is null for the three types of aggregation if P_i is other than P_1. This can be explained by the fact that often in the historical database the projects are independent. An interesting observation is that $d(P_1, P_1)$ is not equal to 1!
- Second, is the 'or' logical operation by using the *max* operator; so, $d(P, P_i)$, is null if all $d_{v_j}(P_1, P_i)$ are the same ones; it is equal to 1 if only one $d_{v_j}(P_1, P_i)$ is the same one. From the results obtained (Max columns), we can notice that for project P_8, the *Kleene-Dienes* aggregation gives significant different value (0.4785) than *max-min* and *sum-product* aggregations. So, this case must be studied separately in order to explain if the problem is in the project P_8 or in the measures. For the other projects, there in no significance difference between $d(P_1, P_i)$ for all types of aggregation. Contrary to the *min* operator, $d(P_1, P_i)$ is not null if P_i is other than P_1. This contradicts the fact that the projects in the historical database must be independent. We can notice that $d(P_1, P_1)$ is equal to 1.
- Third, between the 'and' and the 'or' logical operations by using the *i-or* operator; $d(P, P_i)$, is null if only one $d_{v_j}(P_1, P_i)$ is the same one; it is equal to 1 if only one $d_{v_j}(P_1, P_i)$ is the same and all other $d_{v_j}(P_1, P_i)$ are different from 0. From the results obtained (*i-or* columns), there is no significance

[2] Principal Components Analysis

difference between all types of aggregation. Contrary to the *min* operator, $d(P_1, P_1)$ is equal to 1. $d(P_1, P_i)$ is null if P_i is other than P_1.

From this illustration, many observations can be made:

- The results obtained when the combining of $d_{v_j}(P_1, P_i)$ uses the *max* operator does not gives a good indicator that this is a valid measure for similarity;
- One of the two types of aggregation *max-min* (*sum-product*) or *Kleenes-Dienes* aggregation is not valid for software projects similarity;
- The results obtained when the combining of $d_{v_j}(P_1, P_i)$ uses the *i-or* seems to define a valid measure for similarity.

To check the observations, our measures must be validated. The validation of measures is the process of ensuring that they not contradict any intuitive notions about the similarity between two projects. Our initial understanding of the similarity between projects will be codified by a set of axioms. This axiom-based approach is common in many sciences. For example, mathematicians learned about the world by defining axioms for a geometry. Then, by combining axioms and using their results to support or refute their observations, they expanded their understanding and the set of rules that govern the behaviour of objects. Therefore, our fuure work will be devoted to the development of a set of axioms that represent our intuition about the similarity and to the checking we if the two measures $d_{v_j}(P_1, P_i)$ and $d(P_1, P_i)$ satisfy these axioms.

CONCLUSION & FUTURE WORK

In this paper, we have proposed a new type of measures for software projects similarity. These measures are based on fuzzy logic. Consequently, they can be used when the software project attributes are described with linguistic variables. Each linguistic variable was measured by a set of linguistic values which were represented by fuzzy sets rather than classical sets. Our measures are also applicable when the variables are numeric while relocating numerical value into singleton fuzzy set (no uncertainty) or into fuzzy number (uncertainty). We have illustrated their use by using the intermediate COCOMO'81 database. From this illustration, many observations were made. Clearly not all of the proposed measures seem to be valid. So, each proposed measure must be validated according to the axiomatic validation approach [5, 9, 21, 22]. This included the development of a set of axioms for software projects similarity that any measure must satisfy.

BIBLIOGRAPHY

[1] Benitez J. M., Castro J. L., Requena I., 'Are Artficial Neural Networks Black Boxes', IEEE Transactions on neural networks, Vol. 8, NO. 5, September, 1997, pp. 1156-1164

[2] Boehm B., W, 'Software Engineering Economics', Prentice-Hall, 1981.

[3] Boehm B., W, et al., 'Cost Modems for Future Software Life Cycle Processes: COCOMO 2.0', Annals of Softwrae Engineering on Soft. Process and Product Measurement, Amsterdam, 1995.

[4] Chulani D., S., 'Incorporating Bayesian Analysis to Improve the Accuracy of COCOMO II and Its Quality Model Extension', PhD. Qualifying Examen Report, USC, February, 1998.

[5] Fenton N, Pfleeger S. L, ' Software metrics: Arigorous and Practical Approach', International Computer Thomson Press, 1997.

[6] Idri A, Griech B, El Iraki A, 'Towards an Adaptation of the COCOMO Cost Model to the Software Measurement Theory', In Proc. ESEC/FSE, Sep., Zurich, 1997.

[7] Idri A., Kjiri L., Abran A., ' COCOMO Cost Model Using Fuzzy Logic', 7th Intenational conference on Fuzzy theory & Technology, Atlantic city, NJ, February, 2000.

[8] IFPUG, 'Function Point Counting Practices Manuel', Release 4.0, International Function Points Users Group –IFPUG-, Westerville, Ohio, 1994.

[9] Jacquet J. P, Abran A. 'Metrics Validation Proposals: A Structured Analysis', 8th International Workshop on Software Measurement, Magdeburg, Germany, Sep. 1998.

[10] Jager R, ' Fuzzy Logic in Control', Ph.D. Thesis, Technic University Delft, ISBN 90-90008318-9, Dutsh, 1995.

[11] Kitchenham, B., Pfleeger S. L., Fenton N., 'Towards a Framework for software Measurement Validation', IEEE, Trans. On Soft. Eng., Vol. 21, Dec., 1995.
[12] Kolodner, J. L., 'Case-Based Reasoning', Morgan Kaufmann, 1993
[13] Schofield C., ' Non_algorithmic effort Estimation Techniques', TR98-01, March, 1998.
[14] Shepperd M., Schofield C., Kitchenham B., 'Effort Estimation using Analogy', Proceedings of ICSE-18, Berlin, 1996, pp. 170-178
[**15**] Shepperd M, Schofield C., 'Estimating Software Project Effort Using Analogies', IEEE Trans. On Soft. Eng., Vol. 23, NO 12, November. 1997, PP. 736-743
[16] Vicinanza S., Prietolla M. J., 'Case Based Reasoning in Software Effort Estimation' Proceedings 11th Int. Conf. On Information Systems, 1990
[17] WalstonC. E, Felix A. P, ' Amethod of Program*min*g Measurment and Estimation', IBM Systems Journal, Vol 16, NO. 1, 1977.
[18] Zadeh L. A, ' Fuzzy set', Information and Control, Vol. 8, 1965, pp. 338-353.
[19] Zadeh L. A, 'Fuzzy Logic, Neural Networks, and Soft Computing', Comm. ACM, Vol. 37, No. 3, March, 1994, pp.77-84
[20] Zimmerman H. J, 'Fuzzy Set Theory and it Applications', 2d ed, Kluwer-Nijhoff, 1990.
[21] Zuse H., 'Foundations of Validation, Prediction and Software Measures', Proceeding of the AOSW, Portland, April, 1994
[22] Zuse H., 'Validation of Measures and Prediction Models', 9th International Workshop on Software Measurement, Lac-Supérieur, Sep., Canada, 1999.

Appendix 1 : The 15 cost drivers and there 75 effort multipliers

Attributs Produit
RELY (Required Software Reliability)
DATA (Data Base Size)
CPLX (Product Complexity)

Attributs Matériel
TIME (Execution Time Contraint)
STOR (Main storage Contraint)
VIRT (Virtual Machine Volatility)
TURN (Computer Turnaround Time)
VEXP (Virtual Machine Expreriene)

Attributs Personnel
ACAP (Analyst Capability)
AEXP (Application Experience)
PCAP (Programmer Capability)
LEXP (Programming Language Experience)

Attributs Projet
MODP (Modern Programming Practices)
TOOL (Use of Software Tools)
SCED (Required Development Schedule)

Table 1. The 15 cost drivers of the intermediate COCOMO'81

	Linguistic values					
Attribute	Very Low	Low	Nominal	High	Very High	Extra High
RELY	0.75	0.88	1.00	1.15	1.40	
DATA		0.94	1.00	1.08	1.16	
CPLX	0.70	0.85	1.00	1.15	1.30	1.65
TIME			1.00	1.11	1.30	1/66
STOR			1.00	1.06	1.21	1.56
VIRT		0.87	1.00	1.15	1.30	
TURN		0.87	1.00	1.07	1.15	
ACAP	1.46	1.19	1.00	0.86	0.71	
AEXP	1.29	1.13	1.00	0.91	0.82	
PCAP	1.42	1.17	1.00	0.86	0.70	
VEXP	1.21	1.10	1.00	0.90		
LEXP	1.14	1.07	1.00	0.95		
MODP	1.24	1.10	1.00	0.91	0.82	
TOOL	1.24	1.10	1.00	0.91	0.83	
SCED	1.23	1.08	1.00	1.04	1.10	

Table 2 :The 75 effort multipliers used in intermediate COCOMO'81