A Fuzzy Logic Based Set of Measures for Software Project Similarity: Validation and Possible Improvements

Ali Idri and Alain Abran
Software Engineering Management Research Laboratory
Department of Computer Science
UQAM, P.O Box. 8888,. Centre-Ville Postal Station
Montréal, Québec, Canada, H3C 3P8
idri@ensias.um5souissi.ac.ma
alain.abran@uqam.ca

Abstract

The software project similarity attribute has not yet been the subject of in-depth study, even though it is often used when estimating software development effort by analogy. Among the inadequacies identified (Shepperd et al.) in most of the proposed measures for the software project similarity attribute, the most critical is that they are used only when the software projects 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 in terms of an ordinal (or nominal) scale composed of qualifications such as 'very low', 'low' and 'high'. To overcome this limitation, we propose a set of new measures for similarity when the software projects are described by categorical data. These measures are based on fuzzy logic: the categorical data are represented by fuzzy sets and the process of computing the various measures uses fuzzy reasoning. In this work, the proposed measures are validated by means of an axiomatic validation approach, using a set of axioms representing our intuition about the similarity attribute and verifying whether or not each measure contradicts any of the axioms. We also present in this paper the results of an empirical validation of our similarity measures, based on the COCOMO'81 database.

1. Introduction

The similarity of two software projects, which are described and characterized by a set of attributes, is often evaluated by measuring the distance between these two projects through their sets of attributes. Thus, two projects are not considered similar if the differences between their sets of attributes are obvious. It is important to note that the similarity of two software projects also depends 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, one which can only be measured indirectly.

The way in which the similarity of software projects is gauged is fundamental to the estimation of software development effort by analogy, and a variety of approaches have been proposed in the literature [12]. Shepperd et al. [13,14] found three major inadequacies while investigating similarity measures. The first of these is that they are computationally intensive, and, consequently, many Case-Based Reasoning systems have been developed, such as ESTOR[15] and ANGEL[13]. The second is that the algorithms are intolerant of noise and of irrelevant features. The third and most critical is that they cannot handle categorical data other 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, 15 attributes out of 17 (22 out of 24 in the COCOMO II model) are measured with six linguistic values: 'very low', 'low', 'nominal', 'high', 'very high' and 'extra-high' [2,3,4]. Another example is the Function Points measurement method, in which the level of complexity for each item (input, output, inquiry, logical file or interface) is assigned using three qualifications ('low', 'average' and 'high'). Then there are the General System Characteristics, the calculation of which is based on 14 attributes measured on an ordinal scale of six linguistic values (from 'irrelevant' to 'essential') [8]. To overcome this problem, we are proposing a set of similarity measures which are based on fuzzy logic and which can therefore be used when software projects are described by categorical variables [7]. Such proposed measures must, of course, be validated. Consequently, in this work we present the results of an axiomatic validation approach for our measures, as well as the results of an initial empirical validation. This empirical validation is based on the COCOMO'81 database.

This paper is organized as follows: In the first section, we briefly outline the principles of fuzzy logic and we stress what we called the *normal condition*. In the second section, we present a set of candidate measures for similarity when software projects are described by categorical variables. In the third section, we validate the measures by using an axiomatic validation approach. The axiomatic validation of measures is the process whereby we ensure that the measures do not contradict any intuitive notions about the similarity of two software projects. Our intuition will be codified by a set of axioms. In the fourth section, the similarity measures, which were declared valid in the previous section, will be empirically validated; many remarks and observations will be made on the basis of this empirical validation. Consequently, in the fifth section, we present suggestions for improving our similarity measures. A conclusion and an overview of future work conclude this paper.

2. Fuzzy logic

Since its foundation by Zadeh in 1965 [20], Fuzzy Logic (FL) has been the subject of important investigations. At the beginning of the nineties, fuzzy logic was firmly grounded in terms of its theoretical foundations and its application in the various fields in which it was being used (robotics, medicine, image processing, etc.). The aim in this section is not to discuss fuzzy logic in depth, but rather to present those parts of the subject that are necessary for an understanding of this paper.

According to Zadeh [22], the term "fuzzy logic" is currently used in two different senses. In a narrow sense, FL is a logical system aimed at a formalization 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 classical set theory. The membership function $\mu_A(x)$ of x in a classical set A, as a 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 a student's budget, then it is obvious that this set has no clear boundary. Of course, it could be said that a PC priced at \$2500 is too expensive, but what about a PC priced at \$2495 or \$2502? Are these PCs too expensive? Clearly, a boundary could be determined above which a PC is too expensive for the average student, say \$2500, and a boundary below 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 assigned 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.

The 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 known as the membership function and X is known as the universe of discourse. 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 major advantage of the fuzzy set representation is that it is a gradual function rather than an abrupt-step function between the two boundaries of \$1000 and \$2500.

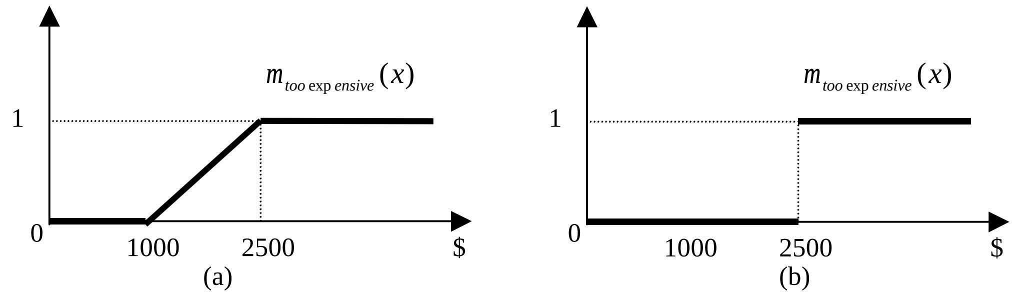

Figure 1. Fuzzy set (a) and Classical set (b) for the linguistic value 'too expensive

2.1 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 $T(a, b) \le T(c, d)$, whenever $a \le c$, $b \le d$
T-3 $T(a, b) = T(b, a)$
T-4 $T(T(a, b),c) = T(a, T(b, c))$
The condition defining a T-conorm (S-norm), besides T-2, T3 and T-4, is:
S-1 $S(a, 0) = a$
Table 1 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	$\mu_{A\cup B} = \max(\mu_A(x),\mu_B(x))$ $\mu_{A\cup B} = \min(\mu_A(x)+\mu_B(x)-1, 0)$

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

2.2 Properties of fuzzy sets

In the following, we present three commonly used fuzzy set properties [10]:
Property 1: Fuzzy set A is called *normal* if the height of A, hgt(A), is equal to 1.

$$hgt(A) = \sup_{x \in X} m_A(x)$$

Property 2: Fuzzy set A is called *convex* if it is characterized by:

$$\forall\, x_1, x_2, x_3 \in X \quad x_1 \le x_2 \le x_3$$
$$m_A(x_2) \ge \min(m_A(x_1), m_A(x_3))$$

Property 3: A tuple of fuzzy sets (A_1, A_2, .., A_M) is called a *fuzzy partition* of universe X if:

$$\forall x \in X, \quad \sum_{j=1}^{M} m_{A_j}(x) = 1 \qquad A_j \ne \Phi \quad and \quad A_j \ne X$$

An important condition, which it is often satisfied in practice, arises if a fuzzy partition (A_1, A_2, .., A_M) is formed by *normal* and *convex* fuzzy sets. In the rest of this paper, such a condition will be called a *normal condition* (NC):

A tuple of fuzzy sets (A_1, A_2, .., A_M) satisfy the normal condition if (A_1, A_2, .., A_M) is a fuzzy partition and each A_i is normal and convex

Among the other branches of fuzzy set theory are fuzzy arithmetic, fuzzy graph theory and fuzzy data analysis.

3. Similarity measures based on fuzzy logic

The goal is to measure the similarity of two software projects P_1 and P_2 when their attributes are described by categorical values (linguistic values in fuzzy logic). In the software measurement literature, these categorical data are represented by classical intervals (or step functions). So, no project can occupy more than one interval. This is a serious problem in that it can lead to a great difference in effort estimations in the case of similar projects with a small incremental size difference, since each would be placed in a different interval of a step function [6]. Consequently, we have used fuzzy sets with a membership function rather than classical intervals to represent the categorical data. The advantages of this representation are as follows:

- 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 a representation, new similarity measures have been proposed [7]. Let projects P_1 and P_2 be described by M linguistic variables (V_j), and, 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 ($m_{A_k^j}$). Our measures of the similarity of P_1 and P_2 operate on two levels:

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

Figure 2 summarizes the process for computing the various measures.

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

The first step consists in calculating the similarity of 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 expresses the fuzzy equality according to Vj of P_1 and P_2.

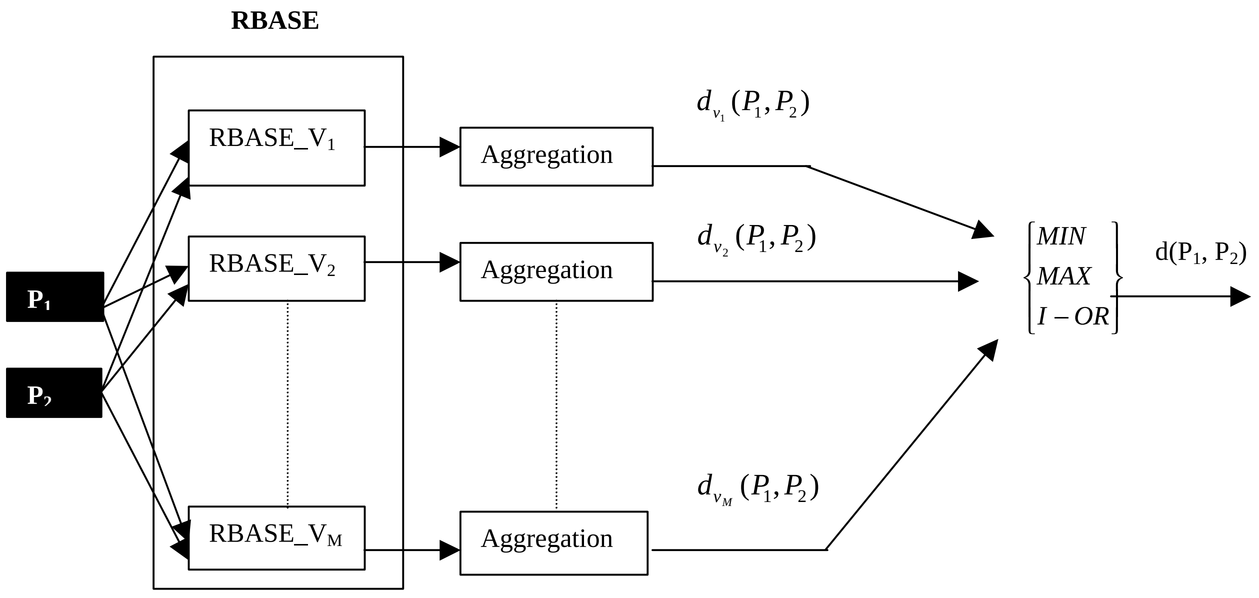

Figure 2. The computing process for the various measures, $d_{v_j}(P_1,P_2)$ and d(P$_1$, P$_2$)

This fuzzy set must have a membership function with two variables ($V_j(P_1)$ and $V_j(P_2)$). This type of fuzzy set is referred to in fuzzy set theory as a *fuzzy relation*. Such a fuzzy relation can represent an association or a 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 denote this fuzzy relation by $R_{\approx}^{v_j}$. $R_{\approx}^{v_j}$ is a combination of a set of fuzzy relations $R_{\approx,k}^{v_j}$. Each $R_{\approx,k}^{v_j}$ represents the equality of V$_j$ according to one of its linguistic values 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:

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

Hence, for each variable V$_j$, we have a rule base (RBASE_V$_j$) which contains the same number of fuzzy *if-then* rules as the number of fuzzy sets defined for V$_j$. Each RBASE_Vj expresses the fuzzy equality of two software projects according to V$_j$, $d_{v_j}(P_1,P_2)$. When we consider all variables V$_j$, we obtain a rule base (RBASE) which contains all rules associated with all variables. RBASE expresses the fuzzy equality of two software projects according to all variables V$_j$, d(P$_1$, P$_2$). $d_{v_j}(P_1,P_2)$ is defined by combining all fuzzy rules in DBASE_V$_j$ to obtain one fuzzy relation ($R_{\approx}^{v_j}$) which represents DBASE_V$_j$. 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 the various types of fuzzy implication functions adopted for the fuzzy rules. These fuzzy implication functions are based on distinguishing between two basic types of implication: the fuzzy implication which complies with the classical conjunction and the fuzzy implication which complies with the classical implication [10]. Using this basic distinction of two types of fuzzy implication, we have obtained three formulas for $d_{v_j}(P_1,P_2)$:

$$d_{v_j}(P_1,P_2) = \begin{cases} \max\limits_k \min(m_{A_k^j}(P_1), m_{A_k^j}(P_2)) & \\ \max-\min\ aggregation & (1.1) \\ \sum\limits_k m_{A_k^j}(P_1), m_{A_k^j}(P_2) & \\ sum-product\ aggregation & (1.2) \\ \min\limits_k \max(1-m_{A_k^j}(P_1), m_{A_k^j}(P_2)) & \\ \min-Kleene-Dienes\ aggregation & (1.3) \end{cases}$$

$d_{v_j}(P_1,P_2)$ equal to 1 implies a perfect similarity between P$_1$ and P$_2$ according to V$_j$; equal to 0, a total absence of similarity; between 0 and 1, a partial similarity.

3.2 Second step: Project 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$_2$, and F can be any function with M variables from $[0,1]^M$ to IR^+. Because the various

$d_{v_j}(P_1,P_2)$ are membership functions associated with fuzzy relations $\underset{\approx}{R}^{v_j}$, we have defined F as one of the three operators: *min* as a T-norm, m*ax* as an S-norm and the *i-or* operator as a hybrid between a T-norm and an S-norm. Since in our case, the formula given in [1] can generate undefined values, we have adopted a modification in order to avoid this. Also, the modification will allow the *i-or* operator to work as a very conservative T-norm. The three formulas obtained are:

$$d(P_1,P_2)=\begin{cases}\min\limits_j(d_{v_j}(P_1,P_2)) & 2.1\\ \max\limits_j(d_{v_j}(P_1,P_2)) & 2.2\\ \underset{j}{i-or}(d_{v_j}(P_1,P_2)) & 2.3\end{cases}$$

where

$$\underset{j}{i-or}(d_{v_j}(P_1,P_2))=\begin{cases}0 \quad \exists k,h/\, d_{v_k}(P_1,P_2)=0 \; and \; d_{v_h}(P_1,P_2)=1\\ \dfrac{\prod\limits_{j=1}^{M} d_{v_j}(P_1,P_2)}{\prod\limits_{j=1}^{M}(1-d_{v_j}(P_1,P_2))+\prod\limits_{j=1}^{M} d_{v_j}(P_1,P_2)} \quad otherwise\end{cases}$$

We note that d(P$_1$, P$_2$) using these three formulas is always in the unit interval. The natural interpretation of these three formulas will be discussed in section 5.

4. Axiomatic validation of the similarity measures

The software engineering community has always been aware of the need for validation. As new measures are proposed, it is appropriate to ask whether or not they capture the attribute they claim to describe. This allows us to choose the best measures from a very large number of software measures for a given attribute. However, validation of software measures is one of the most misunderstood procedures in the software measurement area. The first question is: What is a valid measure? A number of authors in software metrics have attempted to answer this question [5, 9, 11, 23, 24]. However, the validation problem has up to now been tackled from different points of view (mathematical, empirical, etc.) and by interpreting the expression "metrics validation" differently; as suggested by Kitchenham et al: '*What has been missing so far is a proper discussion of relationships among the different approaches*' [11]. Beyond this interesting issue, we use Fenton's definitions to validate the two measures, $d_{v_j}(P_1,P_2)$ and d(P$_1$, P$_2$) [8]:

Validating a software measure is the process of ensuring that the measure is a proper numerical characterization of the claimed attribute by showing that the representation condition is satisfied.

This is validation in the *narrow sense, meaning it is internally valid*; if the measure is a component of a valid prediction system, the measure is valid in the *wide sense*. In this section, we deal with the validation of $d_{v_j}(P_1,P_2)$ and d(P$_1$, P$_2$) in the narrow sense.

$d_{v_j}(P_1,P_2)$ and d(P$_1$, P$_2$) satisfy the representation condition if they do not contradict any intuitive notions about the similarity of P$_1$ and P$_2$. Our initial understanding of the similarity of 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 governs the behavior of objects. Below, we present a set of axioms that represents our intuition about the similarity attribute between software projects and we check whether or not the two measures, $d_{v_j}(P_1,P_2)$ and d(P$_1$, P$_2$), satisfy these axioms.

4.1 Axiom 0 (specific to $d_{v_j}(P,P_i)$)

The similarity of two projects, according to a variable V$_j$, is not null if these two projects have a degree of membership different from 0 to at least one same fuzzy set of V$_j$

$$d_{v_j}(P,P_i)\neq 0 \;\; iff \;\; \exists\, A_k \;/\; m_{A_k^j}(P)\neq 0 \; and \; m_{A_k^j}(P_i)\neq 0$$

To illustrate this axiom, we use the following figure:

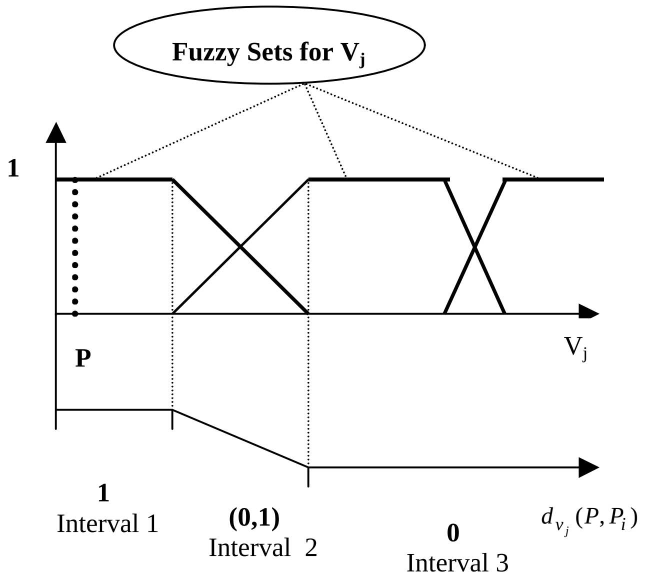

Figure 3. An explanatory example of Axiom 0

- $d_{v_j}(P,P_i)$ must be equal to 1 if V$_j$(P$_i$) is in Interval 1;
- $d_{v_j}(P,P_i)$ must decrease 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.

It is easy to show that similarity according to V$_j$ measured by *max-min* or *sum-product* aggregations (formulas (1.1) and (1.2)) respects Axiom 0. This is not the case when it is measured by *min-Kleene-Dienes* aggregation (formula (1.3)). Intuitively, P and P$_4$ are not similar according to V$_j$ (Figure 4). So, $d_{v_j}(P,P_4)$ must be equal to 0; but by applying the formula (1.3), $d_{v_j}(P,P_4)$ is equal to 0.5!

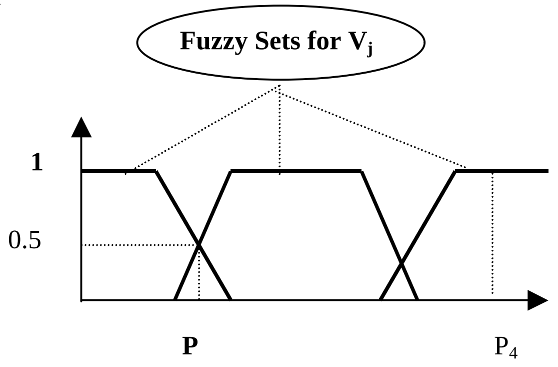

Figure 4. A counter-example showing that min-Kleene-Dienes aggregation does not respect Axiom 0

4.2 Axiom 1

We expect any measure m of the similarity of two projects to be positive:

$$m(P_1, P_2)³0;\ m(P, P)>0$$

$d_{v_j}(P_1,P_2)$, in all cases formulas (1.1), (1.2) and (1.3), is always higher than or equal to 0. So, it is also the case that d(P$_1$, P$_2$). $d_{v_j}(P,P)$, when using *min-Kleene-Dienes* aggregation, is higher than 0. But when it uses *max-min* or *sum–product* aggregations, it can be equal to 0. This is the case when $m_{A_k^j}(P)$ is equal to 0 for all A_k^j. This implies that project P does not have any qualification for the variable V$_j$. This case can be avoided if the fuzzy sets (A_1^j , A_2^j …, A_k^j , …, $A_{N_j}^j$) form a fuzzy partition for V$_j$. This is always the case in practice. Consequently $d_{v_j}(P,P)$ will be considered higher than 0 for all types of aggregation.

4.3 Axiom 2

The degree of similarity of any project to P must be lower than the degree of similarity of P to itself:

$$m(P,\ P_i)£\ m(P,\ P)$$

4.3.1 $\mathbf{d_{v_j}(P,P_i)}$ using max-min aggregation. We show that, for any project P$_i$, $d_{v_j}(P,P_i) \le d_{v_j}(P,P)$ (Appendix 1, Proof 1).

4.3.2 $\mathbf{d_{v_j}(P,P_i)}$ using sum-product aggregation. We show by a counter-example that $d_{v_j}(P,P_i)$ does not respect Axiom 2:

- For P$_i$, $m_{A_{k_0}^j}(P_i)$ is equal to 1 and for all other A_k^j (k≠k$_0$), $m_{A_k^j}(P_i)$ is null.
- For P, $m_{A_{k_0}^j}(P)$ is equal to 0,7, $m_{A_{k_1}^j}(P)$ is equal to 0.3, and, for all other A_k^j (k≠k$_0$ and k≠k$_1$), $m_{A_k^j}(P)$ is null.

In this case, $d_{v_j}(P,P_i)$ is equal to 0.7, while $d_{v_j}(P,P)$ is equal to 0.58 (0,58=0.7^2 + 0,3^2).

4.3.3 $\mathbf{d_{v_j}(P,P_i)}$ using min-Kleene-Dienes aggregation. In general, $d_{v_j}(P,P_i)$ does not respect Axiom 2 (Figure 5). But, if (A_1^j , A_2^j …, A_k^j , …, $A_{N_j}^j$) satisfy the *normal condition*, then *min-Kleene-Dienes* aggregation respects Axiom 2 (Appendix 1, Proof 2). The *normal condition* involves that (A_1^j , A_2^j …, A_k^j , …, $A_{N_j}^j$) does not contain more than two overlapping fuzzy sets.

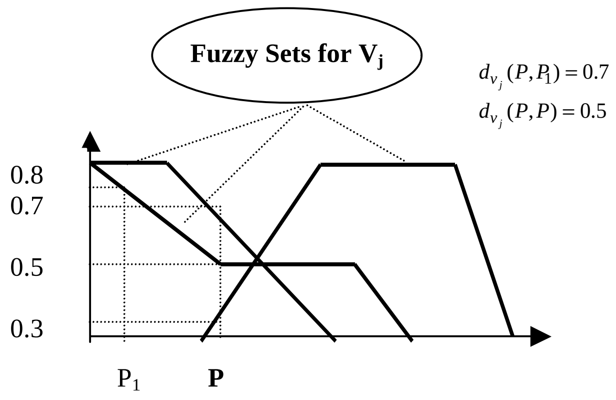

Figure 5: A counter-example showing that $\mathbf{d_{v_j}(P,P_i)}$ using min-Kleene-Dienes aggregation does not respect Axiom 2

4.3.4 Distance d(P, P$_i$). It is calculated from the distances $d_{v_j}(P,P_i)$ by using the *min*, the *max* or the *i-or* operators. Thus, to check whether or not d(P, P$_i$) respects Axiom 2, we will use the results of the validation of the distance $d_{v_j}(P,P_i)$. It is easy to show that d(P, P$_i$) respects Axiom 2, some is the operator used, when $d_{v_j}(P,P_i)$ uses *max-min* aggregation (Appendix 1, Proof 3). We proceed in the same way to validate the distance d(P, P$_i$) when $d_{v_j}(P,P_i)$ uses *sum-product* or *min- Kleene-dienes* aggregation. Table 2 shows the results obtained.

$d_{v_j}(P,P_i)$	d(P, P$_i$)		
	Min	**Max**	***i-or***
Max-min	Yes	Yes	Yes
Sum-product	No	No	No
Kleene-Dienes	Yes if NC	Yes if NC	Yes if NC

Table 2: Results of the validation of the distance d(P,P$_i$) for Axiom 2

4.4 Axiom 3

We expect any measure m of the similarity of two projects to be commutative:

$$m(P_1, P_2)= m(P_2, P_1)$$

$d_{v_j}(P,P_i)$ respects Axiom 3 when it uses *max-min* or *sum-product* aggregation. Consequently, d(P$_1$,P$_2$) is also some is the operator used. But, this is not the case when $d_{v_j}(P_1,P_2)$ uses *min-Kleene-Dienes* aggregation. We can check that $d_{v_j}(P,P_1)$ is equal to 0.7 and $d_{v_j}(P_1,P)$ is equal to 0.5 (Figure 5).

By looking at the results of this validation, which takes into account four axioms (Table 3), we can conclude that $d_{v_j}(P,P_i)$ using *max-min* aggregation respects all the axioms (as, consequently, does d(P,P$_i$)). So, according to Fenton [8], this is a valid similarity measure in the *narrow sense*. $d_{v_j}(P,P_i)$, using *sum-product* aggregation does not respect Axiom 2. Although Axiom 2 is interesting, we will retain *sum-product* aggregation in order to be validated in the wide sense. There are three reasons for this:

- The difference between $d_{v_j}(P,P_i)$ and $d_{v_j}(P,P)$ is not obvious if the fuzzy sets associated with V$_j$ satisfy the *normal condition*. We can show that this difference, in the case where $d_{v_j}(P,P_i)$ is higher than $d_{v_j}(P,P)$, is in the interval [-1/8, 0].
- *Sum-product* aggregation respects the other axioms, specifically Axiom 0.
- As was noted by Zuse [24], validation in the *narrow sense*, contrary to validation in the *wide sense*, is not yet widely accepted and mostly neglected in practice.

$d_{v_j}(P,P_i)$, using *min-Kleene-Dienes* aggregation does not respect Axiom 0 and Axiom 3. Although it respects Axiom 1 and Axiom 2, we rejected it because of Axiom 0. For us, Axiom 0 represents the definition of the similarity of two software projects according to a fuzzy variable. Consequently, any similarity measure must satisfy this axiom.

	$d_{v_j}(P,P_i)$ /d(P, P$_i$)		
	max-min	**sum-product**	**Kleene-Dienes**
Axion0	Yes/	Yes/	No/
Axiom1	Yes/Yes	Yes/Yes	Yes/Yes
Axiom2	Yes/Yes	No/No	Yes /Yes if NC
Axiom3	Yes/Yes	Yes/Yes	No/No

Table 3: Results of the validation of the distance $d_{v_j}(P,P_i)$ and d(P, P$_i$)

5. Towards an empirical validation of the proposed similarity measures

After validation in the narrow sense of the similarity measures (the measures are measuring what they claim to measure), we present, in this section, the first results of an incomplete empirical validation (validation in the wide sense) of our measures. According to Fenton [8], a measure is valid in the wide sense if it is both valid in the narrow sense and a component of a valid prediction system. The prediction system that we consider here is the estimation of software development effort by analogy. It is based on three steps. First, each project must be described by a set of linguistic variables which must be relevant, independent, operational and comprehensive. Second, we must determine the similarity between the candidate project and each project in the historical database by using the measures that are declared valid in the *narrow sense*. Third, we use the known effort values from the historical projects to derive an estimate for the new project. Below, we present only the results of the two first steps. The intermediate COCOMO'81 database was chosen as the basis for this empirical validation.

The original intermediate COCOMO'81 database contains 63 projects. Each project is described by 17 attributes: the software size is 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 using a rating scale of six linguistic values: 'very low', 'low', 'nominal', 'high', 'very high' and 'extra high'. The assignment of linguistic values to the cost drivers (or project attributes) uses conventional quantification where the values are intervals (see [3], pp. 119). For example, the DATA cost driver is measured by the following ratio:

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

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 4. DATA cost driver ratings.

To use the proposed similarity measures, and because of the advantages of representation by fuzzy sets rather than classical intervals (section 2), the 15 cost drivers must be fuzzified. 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 6). We note that the fuzzy sets associated with the DATA cost driver satisfy the *normal condition*. For the other cost drivers of the intermediate COCOMO'81 database, we proceed in the same way as for DATA. Of the 15 cost drivers, the four factors RELY, CPLX, MODP and TOOL are not studied because their relative descriptions are insufficient. So, we consider the 12 cost drivers that we have fuzzified [6].

Because the original COCOMO'81 database contains only the effort multipliers, our evaluation of the similarity will be made on an artificial dataset deduced from the original COCOMO'81 database. This artificial dataset contains 63 projects with the real values that are necessary to determine $m_{A_k^j}(P)$ of the formulas (1.1) and (1.2). For example, the DATA cost driver for the fifth project in the COCOMO'81 database is declared 'low'; thus, the randomly generated value for the fifth project in the artificial dataset is between 0 and 10. For simplification, we calculate the similarity of the first five projects (P_1, P_2, P_3, P_4, P_5) of the COCOMO'81 database. Because our measures are computationally intensive, we have developed a software prototype to automate the calculations. This prototype uses Microsoft Excel to store data and Microsoft Visual Basic to implement the various processing steps. The tables 5 and 6 shows the results obtained for the similarity measured by *max-min* and *sum-product* aggregation (formulas (1.1) and (1.2)).

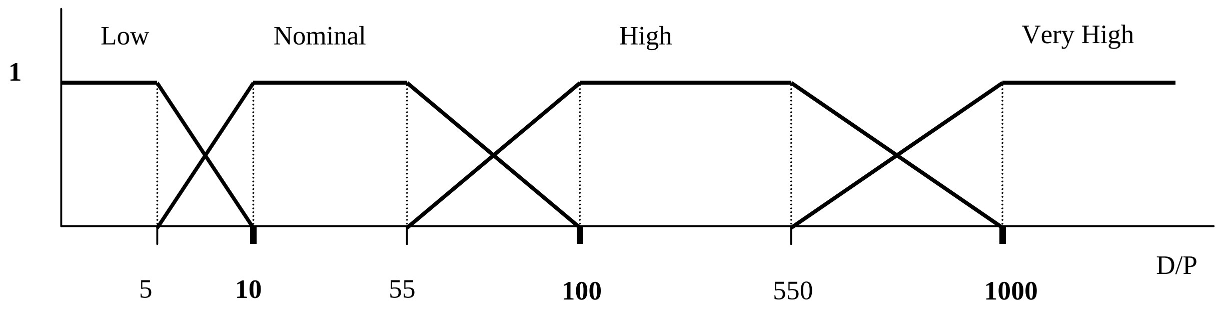

Figure 6. Membership functions of fuzzy sets defined for the DATA cost driver

	Max-min aggregation $d_{v_j}(P_m,P_n)$														
	$d(P_m, P_n)$														
	Min					**Max**					**i-or**				
	P_1	**P_2**	**P_3**	**P_4**	**P_5**	**P_1**	**P_2**	**P_3**	**P_4**	**P_5**	**P_1**	**P_2**	**P_3**	**P_4**	**P_5**
P_1	**.521**	0	0	0	0	**1**	1	1	1	.849	**1**	0	0	0	0
P_2	0	**.742**	0	0	0	1	**1**	1	1	1	0	**1**	0	0	0
P_3	0	0	**.659**	0	0	1	1	**1**	1	1	0	0	**1**	0	0
P_4	0	0	0	**.849**	0	1	1	1	**1**	1	0	0	0	**1**	0
P_5	0	0	0	0	**.897**	.849	1	1	1	**1**	0	0	0	0	**1**

Table 5. Results obtained for $d(P_m, P_n)$ when $d_{v_j}(P_m,P_n)$ uses max-min aggregation.

Sum-product aggregation $d_{v_j}(P_m,P_n)$

	d(P_m, P_n)														
	Min					**Max**					**i-or**				
	P_1	**P_2**	**P_3**	**P_4**	**P_5**	**P_1**	**P_2**	**P_3**	**P_4**	**P_5**	**P_1**	**P_2**	**P_3**	**P_4**	**P_5**
P_1	**.501**	0	0	0	0	**1**	1	1	1	.807	**1**	0	0	0	0
P_2	0	**.610**	0	0	0	1	**1**	1	1	1	0	**1**	0	0	0
P_3	0	0	**.550**	0	0	1	1	**1**	1	1	0	0	**1**	0	0
P_4	0	0	0	**.744**	0	1	1	1	**1**	1	0	0	0	**1**	0
P_5	0	0	0	0	**.815**	.807	1	1	1	**1**	0	0	0	0	**1**

Table 6. Results obtained for d(P_m, P_n) when $d_{v_j}(P_m, P_n)$ uses sum-product aggregation.

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

- In the first, the 'and' logical operation is used. It is implemented by the *min* operator; so, the distance between two projects, d(P_m, P_n), is null if only one $d_{v_j}(P_m,P_n)$ is the same; it is equal to 1 if all $d_{v_j}(P_m,P_n)$ are the same. From the results obtained (*Min* columns in Tables 5 and 6), we note that d(P_m, P_n) is null for the two types of aggregation if P_m is other than P_n. This can be explained by the fact that often in the COCOMO'81 database two projects have at least one variable for which the associated linguistic values are different. The results obtained for d(P_m, P_m) are different for the two types of aggregation. We can show that the absolute value of this difference is lower than 1/8 if all the variables describing software projects satisfy the *normal condition*; this is our case with the COCOMO'81 database (Appendix 1, Proof 4). An interesting observation is that d(P_m, P_m) is not equal to 1; but, with the *normal condition*, it is always higher than 1/2.
- In the second, the 'or' logical operation is used. It is implemented by the *max* operator; so, d(P_m, P_n) is null if all $d_{v_j}(P_m,P_n)$ are the same; it is equal to 1 if only one $d_{v_j}(P_m,P_n)$ is the same. Contrary to the case of the *min* operator, d(P_m, P_n) is not null if P_m is other than P_n. This implies that often in the COCOMO'81 database two projects have at least one variable for which the associated linguistic values are the same. In general, d(P_m, P_n) for the two types of aggregation should be different (as in the case of d(P_1,P_5)), and, as in the case of the *min* operator, if the *normal condition* is satisfied, the absolute value of this difference is lower than 1/8. We note that d(P_m, P_m) is equal to 1.
- In the third, between the 'and' and the 'or' logical operations by using the *i-or* operator; d(P_m, P_n) is null if only one $d_{v_j}(P_m,P_n)$ is the same; it is equal to 1 if only one $d_{v_j}(P_m,P_n)$ is the same and all other $d_{v_j}(P_m,P_n)$ are different from 0. From the results obtained (*i-or* columns in Tables 5 and 6), we note that d(P_m, P_n) is null for the two types of aggregation if P_m is other than P_n. This is true for the same reason that it is true in the case of *min* operator. In general, d(P_m, P_n) for the two types of aggregation should be different if all $d_{v_j}(P_m,P_n)$ are other than 0 and 1. This case is not represented in the COCOMO'81 database. If the *normal condition* is verified, this difference is not obvious because that between $d_{v_j}(P_m,P_n)$ using *max-min* aggregation and $d_{v_j}(P_m,P_n)$ using *sum-product* aggregation is always lower than 1/8 and the *i-or* function is continuous. Contrary to the case of the *min* operator, d(P_m, P_m) is equal to 1 because for all the variables $d_{v_j}(P_m,P_m)$ are different from 0, and it is likely that according to at least one variable $d_{v_j}(P_m,P_m)$ is equal to 1.

From these results, we can conclude that there is no significant difference between *max-min* and *sum-product* aggregation if the *normal condition* is satisfied; while the differences are obvious when evaluating d(P_m, P_n) by using the *min*, *max* or *i-or* operators. In the following section, we discuss the meanings and the uses of these three operators.

6. Discussion

In evaluating the overall distance d(P, P_i), we have used three fuzzy set operators to combine the individual distances, $d_{v_j}(P,P_i)$. The use of the *min* (or *max*) operator reflects a combining referred to as a universal 'all' linguistic quantifier by Zadeh [21] (or existential, 'there exists'). As noted by Yager when studying multi-criteria decision problems, these two combinations may not always be the appropriate relationships among the criteria. For example, a decision-maker may be satisfied if 'most' of the criteria are satisfied; other linguistic quantifiers can be used, such as 'many', 'at least half', 'some' and 'few'. So, Yager suggested a softer combining by the use of what he called quantifier guided aggregations [16,18]. These kinds of aggregation are implemented by the Ordered Weight Averaging (OWA) operators [17]. Recently, Yager has applied this to implementing a soft aggregation for fuzzy constraint satisfaction in the E-commerce domain [19]. For our case, we are now looking to use OWA operators to calculate the overall distance d(P,P_i). The reasons for this are as follows:

- The *min* and *max* operators are not always a good combination of the individual distances. Let us suppose that we have two software projects P_1 and P_2 such that $d_{v_{j_0}}(P_1,P_2)$ =0, $d_{v_j}(P_1,P_2)$ =1 for $j \neq j_0$ (or $d_{v_{j_0}}(P_1,P_2)$ =1, $d_{v_j}(P_1,P_2)$ =0 for $j \neq j_0$) and V_{j0} is the least significant of all the factors describing projects P_1 and P_2. When we use a *min* (or *max*) operator, the overall distance d(P_1, P_2) is null (or equal to 1), while a suitable combination would seem to give a value in the vicinity of 1 (or of 0).
- The *min* and *max* operators are special cases of OWA operators.
- The OWA operators can be used in environments in which the individual distances to be aggregated have an importance associated with them. This is often the case for software projects where the importance of some factors is greater than that of others. Consequently, the contributions of the various individual distances into the calculation of the overall distance should not be equal.
- The OWA operators can implement other linguistic quantifiers used in practice, such as 'most', 'few' and 'many'.
- The *i-or* operator has been used because it is a hybrid between a T-norm and an S-norm. This is also the case for an OWA operator. Although it is claimed that the *i-or* has a natural interpretation and can be used in many situations (evaluation of scientific papers, quality of a game developed by two tennis players in a doubles tennis match, for example), the *i-or* operator cannot be represented by a linguistic quantifier; so, it is not an OWA operator. Indeed, for two values a and b in the unit interval, the following system does not have a solution:

$$\frac{ab}{(1-a)(1-b)+ab} = w_1 a + w_2 b$$
$$\begin{cases} w_1, w_2 \in [0,1] \\ w_1 + w_2 = 1 \end{cases}$$

Clearly, there is a need for further investigation into the acceptance or rejection of the use of the *i-or* operator in the evaluation of the overall distance d(P, P_i) from the various individual distances $d_{v_j}(P,P_i)$.

In a further empirical validation, we used the following formal procedure rather than the formulas (2) to evaluate the overall distance d(P, P_i) in a given environment. First, we had to determine the appropriate linguistic quantifier to be used in such an environment. Second, this linguistic quantifier is used to generate an OWA weighting vector W (w_1, w_2, .., w_M) of dimension M (M is the number of variables describing the software projects) such that the w_is are in the unit interval and the sum of w_is is equal to 1. Third, we calculated the overall distance d(P, P_i) by:

$$d(P,P_i) = \sum_{j=1}^{M} w_j d_{v_j}(P,P_i) \qquad (3)$$

where $d_{v_j}(P,P_i)$ is the J^{th} largest individual distance.

We can then check that the axiomatic validation of d(P, P_i) using a softer combining of $d_{v_j}(P,P_i)$ (formula (3)) gives the same results as those in Table 3.

7. Conclusion & future work

In this paper, we have validated a set of new similarity measures based on fuzzy logic. They can be used when the software project attributes are described with linguistic variables. Our measures are also applicable when the variables are numeric while relocating numeric values into a singleton fuzzy set (no uncertainty) or into a fuzzy number (uncertainty). Our measures operate on two levels. The similarity measures according to only one variable $d_{v_j}(P,P_i)$ and those according to all variables, d(P, P_i). We have adopted Fenton's definitions to validate these two measures. First, the measures are validated in the narrow sense by using four axioms. These axioms codify our initial understanding of the similarity of software projects and can also be used in the case of the similarity of entities other than software projects. From this axiomatic validation, we have retained the individual distance $d_{v_j}(P,P_i)$ using *max-min* or *sum-product* aggregation and rejected the using of *min-Kleene-Dienes* aggregation. The overall distance combines the individual distances by means of three operators. So, its axiomatic validation depends on the axiomatic validation of $d_{v_j}(P,P_i)$. Second, we have started the validation in the wide sense of the retained measures that are declared valid in the narrow sense. We have chosen estimation by analogy of software development effort and the COCOMO'81 database as the basis for this validation. The results obtained from the application of the first two steps of estimation effort by analogy have shown that there is no significant difference between *max-min* and *sum-product* aggregation; while the use of the three operators (*min*, *max* and *i-or*) in the evaluation of d(P, P_i) from the individual distances $d_{v_j}(P,P_i)$ gives significant differences and cannot be always a good choice. Consequently, we have proposed other alternatives for combining the individual distances. These alternatives, such as 'most', 'few' and 'many', are implemented by the OWA operators. The most significant advantage of this is that for each environment the appropriate linguistic quantifier (alternative) can be chosen for use in the evaluation of the overall distance. To complete the empirical validation of the retained measures, we must validate the estimation of effort by analogy approach. This validation will consist in comparing the accuracy of the estimated effort values with actual effort values.

8. References

[1] J.M. Benitez, J.L. Castro, and I. Requena, "Are Artificial Neural Networks Black Boxes?", *IEEE Transactions on Neural Networks*, Vol. 8, no. 5, September, 1997, pp. 1156-1164

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

[3] 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.

[4] 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.

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

[6] 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

[7] 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

[8] IFPUG, "Function Point Counting Practices Manual", Release 4.0, *International Function Point Users Group – IFPUG*, Westerville, Ohio, 1994.

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

[10] R. Jager, "Fuzzy Logic in Control", Ph.D. Thesis, Technic University Delft, Holland, 1995.

[11] B. Kitchenham, S.L. Pfleeger, and N. Fenton, "Towards a Framework for Software Measurement Validation", *IEEE Trans. on Software Engineering*, Vol. 21, December, 1995.

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

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

[14] 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

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

[16] R.R. Yager, "On ordred weighted averaging aggregation operators in multi-criteria decision making", *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 18, 1988, pp. 183-190

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

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

[19] R.R. Yager, "Fuzzy Constraint Satisfaction for E-commerce Agents", *7th International Conference on Fuzzy Theory & Technology*, Atlantic City, NJ, February, 2000. pp. 111-114

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

[21] L.A. Zadeh, "A computational approach to fuzzy quantifiers in natural languages", *Computing and Mathematics with Applications*, 9, 1983, pp. 149-184

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

[23] H. Zuse, "Foundations of Validation, Prediction and Software Measures", *Proceedings of the AOSW*, Portland, April, 1994

[24] H. Zuse, "Validation of Measures and Prediction Models", *9th International Workshop on Software Measurement*, Lac-Supérieur, Canada, September, 1999.

Appendix 1

Proof 1: $\mathbf{d_{v_j}(P,P_i)}$ using max-min aggregation

$$\forall k \quad \min(m_{A_k^j}(P), m_{A_k^j}(P_i)) \le m_{A_k^j}(P) = \min(m_{A_k^j}(P), m_{A_k^j}(P))$$
$$\max_k \min(m_{A_k^j}(P), m_{A_k^j}(P_i)) \le \max_k \min(m_{A_k^j}(P), m_{A_k^j}(P))$$
$$d_{v_j}(P,P_i) \le d_{v_j}(P,P)$$

Proof 2: $\mathbf{d_{v_j}(P,P_i)}$ using min-Kleene-Dienes aggregation

$$\exists! k \;/\; m_{A_k^j}(P) \ne 0, m_{A_{k+1}^j}(P) \ne 0, m_{A_k^j}(P) + m_{A_{k+1}^j}(P) = 1$$
$$\exists! h \;/\; m_{A_h^j}(P_i) \ne 0, m_{A_{h+1}^j}(P_i) \ne 0, m_{A_h^j}(P_i) + m_{A_{h+1}^j}(P_i) = 1$$
$$d_{v_j}(P,P) = \min(\max(1 - m_{A_k^j}(P), m_{A_k^j}(P)), \max(1 - m_{A_{k+1}^j}(P), m_{A_{k+1}^j}(P))$$
$$Or\ m_{A_k^j}(P) + m_{A_{k+1}^j}(P) = 1$$
$$\Rightarrow \max(1 - m_{A_k^j}(P), m_{A_k^j}(P)) = \max(1 - m_{A_{k+1}^j}(P), m_{A_{k+1}^j}(P))$$
$$\Rightarrow d_{v_j}(P,P) = \max(1 - m_{A_k^j}(P), m_{A_k^j}(P)) = \max(1 - m_{A_{k+1}^j}(P), m_{A_{k+1}^j}(P))$$
$$d_{v_j}(P,P_i) = \min(\max(1 - m_{A_k^j}(P), m_{A_k^j}(P_i)), \max(1 - m_{A_{k+1}^j}(P), m_{A_{k+1}^j}(P_i)),$$
$$if\ m_{A_k^j}(P_i) \le m_{A_k^j}(P)$$
$$\Rightarrow \max(1 - m_{A_k^j}(P), m_{A_k^j}(P_i)) \le \max(1 - m_{A_k^j}(P), m_{A_k^j}(P))$$
$$or\ d_{v_j}(P,P_i) \le \max(1 - m_{A_k^j}(P), m_{A_k^j}(P_i))$$
$$\Rightarrow d_{v_j}(P,P_i) \le \max(1 - m_{A_k^j}(P), m_{A_k^j}(P)) = d_{v_j}(P,P)$$
$$if\ m_{A_k^j}(P_i) > m_{A_k^j}(P) \Rightarrow m_{A_k^j}(P_i) \ne 0 \Rightarrow k = h\ ou\ k = h+1$$
$$if\ k = h \quad m_{A_k^j}(P_i) \ge m_{A_k^j}(P)$$
$$\Rightarrow 1 - m_{A_k^j}(P_i) \le 1 - m_{A_k^j}(P) \Rightarrow m_{A_{k+1}^j}(P_i) \le 1 - m_{A_k^j}(P)$$
$$\sin ce \quad 1 - m_{A_{k+1}^j}(P) = m_{A_k^j}(P)$$
$$\max(1 - m_{A_{k+1}^j}(P), m_{A_{k+1}^j}(P_i) \le \max(1 - m_{A_k^j}(P), m_{A_k^j}(P))$$
$$\Rightarrow d_{v_j}(P,P_i) \le d_{v_j}(P,P)$$

$$\begin{aligned}
&if\ \ k = h+1 \Rightarrow m_{A^j_{k+1}}(P_i) = 0\\
&\max(1 - m_{A^j_{k+1}}(P), m_{A^j_{k+1}}(P_i)) = 1 - m_{A^j_{k+1}}(P) \le \max(1 - m_{A^j_{k+1}}(P), m_{A^j_{k+1}}(P))\\
&\Rightarrow d_{v_j}(P,P_i) \le d_{v_j}(P,P)
\end{aligned}$$

Proof 3: distance d(P, Pi) with $\mathbf{d_{v_j}(P,P_i)}$ using max-min aggregation

$$\begin{aligned}
&d_{v_j}(P,P_i) \le d_{v_j}(P,P)\\
&\Rightarrow \prod_{j=1}^{M} d_{v_j}(P,P_i) \le \prod_{j=1}^{M} d_{v_j}(P,P)\\
&\Rightarrow \prod_{j=1}^{M} (1 - d_{v_j}(P,P_i)) \ge \prod_{j=1}^{M} (1 - d_{v_j}(P,P))\\
&\Rightarrow \frac{\prod_{j=1}^{j=M}(1 - d_{v_j}(P,P_i))}{\prod_{j=1}^{j=M} d_{v_j}(P,P_i)} + 1 \ge \frac{\prod_{j=1}^{M}(1 - d_{v_j}(P,P))}{\prod_{j=1}^{M} d_{v_j}(P,P)} + 1 \Rightarrow d(P,P_i) \le d(P,P)\\
&d_{v_j}(P,P_i) \le d_{v_j}(P,P)\\
&\Rightarrow \min_j(d_{v_j}(P,P_i)) \le \min_j(d_{v_j}(P,P)) \Rightarrow d(P,P_i) \le d(P,P)\\
&d_{v_j}(P,P_i) \le d_{v_j}(P,P)\\
&\Rightarrow \max_j(d_{v_j}(P,P_i)) \le \max_j(d_{v_j}(P,P)) \Rightarrow d(P,P_i) \le d(P,P)
\end{aligned}$$

Proof 4: Difference between d($\mathbf{P_m}$, $\mathbf{P_n}$) using max-min aggregation and d(Pm, Pn) using sum-product aggregation

We want to prove that the absolute value of the difference between d(P_m, P_n) using formula (2.1) with $d_{v_j}(P_m,P_n)$ which uses *max-min* aggregation and d(P_m, P_n) using the formula (2.1) with $d_{v_j}(P_m,P_n)$ which uses *sum-product* aggregation is lower than 1/8. In the case of d(P_m, P_n) using formula (2.2), the proof is the same. We suppose that all variables satisfy the *normal condition.*

First, we prove that the absolute value of the difference between $d_{v_j}(P_m,P_n)$ using *max-min* aggregation and $d_{v_j}(P_m,P_n)$ using *sum-product* aggregation is lower than 1/8.

$$\exists k / \begin{cases} m_{A^j_k}(P_m) + m_{A^j_{k+1}}(P_m) = 1\\ m_{A^j_k}(P_n) + m_{A^j_{k+1}}(P_n) = 1 \end{cases}$$

$$\begin{aligned}
&\text{to simplify we uses:}\\
&x = m_{A^j_k}(P_m),\ y = m_{A^j_{k+1}}(P_m),\ z = m_{A^j_k}(P_n)\ and\ w = m_{A^j_{k+1}}(P_n)\\
&\max(\min(x,z),\min(y,w)) - xz - yw = \max(x,w) - xz - wy \quad if\ \ x < z\\
&= \begin{cases} x(2x-1) \text{ if } \mathrm{x = w}\\ \mathrm{w(2x\text{ - }1)} \text{ if } \mathrm{x > w}\\ \mathrm{x(2w\text{ - }1)} \text{ if } \mathrm{w > x} \end{cases}
\end{aligned}$$

By studying these three functions, we can note that each of them has a minimum equal to –1/8 or a maximum equal to 1/8.

Second, d(P_m, P_n) combines $d_{v_j}(P_m,P_n)$ by the *min* operator. d(P_m, P_n) with $d_{v_j}(P_m,P_n)$ using *max-min* aggregation is denoted by d(P_m, P_n)$_{\text{max-min}}$ and d(P_m, P_n) with $d_{v_j}(P_m,P_n)$ using *sum-product* aggregation is denoted by d(P_m, P_n)$_{\text{sum-product}}$:

$$\begin{aligned}
&\exists i_0 / \quad d(P_m,P_n)_{\max-\min} = d^{MM}_{v_{i_0}}(P_m,P_n) = \min_j(d^{MM}_{v_j}(P_m,P_n))\\
&\exists j_0 / \quad d(P_m,P_n)_{sum-product} = d^{SP}_{v_{j_0}}(P_m,P_n) =\\
&\qquad \min_j(d^{SP}_{v_j}(P_m,P_n)) = \min_j(d^{MM}_{v_j}(P_m,P_n) \pm e_j) \ \ |e_j| \le \frac{1}{8}\\
&if\ i_0 \ne j_0 \Rightarrow d^{SP}_{v_{j_0}}(P_m,P_n) \le d^{SP}_{v_{i_0}}(P_m,P_n)\\
&\Rightarrow d^{MM}_{v_{j_0}}(P_m,P_n) \pm e_{j_0} \le d^{MM}_{v_{i_0}}(P_m,P_n) \pm e_{i_0}\\
&while \quad d^{MM}_{v_{j_0}}(P_m,P_n) \ge d^{MM}_{v_{i_0}}(P_m,P_n)\\
&\Rightarrow \quad \left| d^{SP}_{v_{j_0}}(P_m,P_n) - d^{MM}_{v_{i_0}}(P_m,P_n) \right| \le \frac{1}{8}
\end{aligned}$$