Evaluating Software Project Similarity by using Linguistic Quantifiers Guided Aggregations

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

Abstract
Software projects are often described by linguistic variables such as the experience of programmers and the complexity of modules. Because the existing software projects similarity measures take into account only numerical data, we have proposed a set of measures based on fuzzy logic to evaluate the similarity between two software projects when they are described by linguistic values. In this work, we improve the proposed measures by using linguistic quantifiers such as 'most', 'many' and 'few' in the computing process for the various measures.

1. Introduction

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. 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. Among the inadequacies identified (Shepperd et al.) in most of the proposed measures for the software project similarity, the most critical is that they cannot be used when the software projects are described by linguistic values such as 'low' and 'high'. To overcome this limitation, we have proposed a set of new measures based on fuzzy logic for software project similarity [4]. 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 to the various linguistic variables (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 [5]:

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

where V_j's are the linguistic variables describing the projects P_1 and P_2, A_k^j are the fuzzy sets associated to V_j, and $m_{A_k^j}$ are the membership functions representing the 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 by using three fuzzy sets operators (min, max and the i-or):

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

where

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

In this work, we prove that the use of these three operators cannot be always a good choice in many organizations. Consequently, we suggest other alternatives based on linguistic quantifiers for combining the individual distances $d_{v_j}(P_1, P_2)$.

This paper is organized as follows: In the first section, we briefly outline the principles of linguistic quantifiers. In the second section, we discuss why linguistic quantifiers guided aggregations can be used in the evaluation of software project similarity. In the

third section, we present the formal procedure that will be used in the evaluation of the similarity. In the fourth section, we illustrate, by an example, the computing process of the similarity using linguistic quantifiers. This illustration is based on COCOMO'81 software projects database. A conclusion and an overview of future work conclude this paper.

2. Linguistic Quantifiers

Human discourse uses a large number of linguistic quantifiers. In [11], Zadeh distinguish between two classes of linguistic quantifiers: absolute and proportional . Absolute quantifiers such as 'about 10' and 'about 20' can be represented as a fuzzy set Q of the non-negative reals. In this work, our concern is with proportional quantifiers.

A proportional linguistic quantifier indicates a proportional quantity such as 'most', 'many' and 'few'. Zadeh suggested that proportional quantifier can be represented as fuzzy set Q of the unit interval I. In this representation for any $r \in I$, Q(r) is the degree to which the proportion r satisfies the concept represented by the term Q. Furthermore, Yager distinguished three categories of proportional quantifiers [8]:

(a) A Regular Increasing Monotone (RIM) quantifiers such as 'many', 'most' and 'at least α' are represented as fuzzy subset Q satisfying the followings conditions:
   1- Q(0)=0,
   2- Q(1)=1, and
   3- $Q(x) \geq Q(y)$ if x>y

(b) A Regular Decreasing Monotone (RDM) Quantifiers such as 'few' and 'at most α' are represented as fuzzy subset Q satisfying the followings conditions:
   1. Q(0)=1,
   2. Q(1)=0, and
   3. $Q(x) \leq Q(y)$ if x>y

(c) A Regular UniModal quantifiers such as 'about α' are represented as fuzzy subset Q satisfying the followings conditions:.
   1- Q(0)=0,
   2- Q(1)=1, and
   3- There exists two value a and $b \in I$, where a<b, such that:
      i. For y <a, $Q(x) \leq Q(y)$ if x<y
      ii. For $y \in [a,b]$, Q(y)=1
      iii. For y>b, $Q(x) \geq Q(y)$ if x<y

Two interesting relationships exist between these three categories of proportional quantifier:

- If Q is a RIM quantifier then its antonym is a RDM quantifier and a vice versa: Examples of these antonym pairs are 'few' and 'many', and 'at least α' and 'at most α'.
- Any RUM quantifier can be expressed as the intersection of a RIM and RDM quantifier.

3. Why linguistic quantifier guided aggregation can be used in the evaluation of software project similarity?

In evaluating the overall distance $d(P_1, P_2)$, we have used three fuzzy set operators to combine the individual distances, $d_{v_j}(P_1, P_2)$. The use of the *min* (or *max*) operator reflects a combining referred to as a universal 'all' linguistic quantifier by Zadeh [11] (or existential, 'there exists'). As noted by Yager when studying multi-criteria decision problems, these types of aggregation cannot be always the appropriate relationship among the criteria. So, Yager suggested a softer combining by the use of what he called quantifier guided aggregation [7, 8]. These kinds of aggregation can be implemented by the Ordered Weight Averaging (OWA) operators [9]. Recently, Yager has applied this to implementing a soft aggregation for fuzzy constraint satisfaction in the E-commerce domain [10]. For our case, we look at the aggregation of the individual similarities, $d_{v_j}(P_1, P_2)$, by using linguistic quantifiers such as 'most', 'many' and 'few'. The reasons for this are as follows:

- Each organization can choose its appropriate linguistic quantifier to guide the aggregation of the individual similarities $d_{v_j}(P_1, P_2)$;

- 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 P1 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* (or *max)* operator reflects a combining referred to as a universal 'all' (or existential, 'there exists') linguistic quantifier that is only a special case;

- Linguistic quantifiers 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 *i-or* operator has been used because it is a hybrid between a T-norm and an S-norm [1]. This is also the case for linguistic quantifiers implemented by OWA operators. 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 an OWA operator. Indeed, for two values a and b in the unit interval, the following system does not have a solution:

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

Furthermore in our case, the formula given in [1] can generate undefined values when some individual distances are equal to 1 and others are equal to 0. So, we have adopted a modification in order to avoid this; however, the formula (2.3) remains inappropriate in many case. Let us suppose that we have for one Vj_0 (the least significant of all factors), the corresponding individual distance is equal to 1 and for all others V_j, the corresponding individual distances are in the vicinity of 0. When applying the formula (2.3), the overall distance $d(P_1,P_2)$ is equal to 1, while a suitable combination seems to give other than 1. 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 $d(P_1, P_2)$ from the various individual distances.

4. How linguistic quantifiers guided aggregations can be used in the evaluation of software project similarity?

In the process of linguistic quantifiers guided aggregations, we must provide the appropriate linguistic quantifier Q to be used in a given environment. Q indicates the proportion of individual distances that we feel is necessary for a good evaluation of the overall distance. The following formal procedure is used to evaluate the overall distance $d(P_1, P_2)$ in a given environment. First, the linguistic quantifier, Q, 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. Second, we calculated the overall distance $d(P_1, P_2)$ by:

$$d(P_1,P_2) = \sum_{j=1}^{M} w_j d_{v_j}(P_1,P_2) \qquad (3)$$

where $d_{v_j}(P_1,P_2)$ is the j[th] largest individual distance.

The procedure used for generating the weights from the linguistic quantifier Q depends upon the type of Q. In our case, Q must be a RIM (Regular Increasing Monotone) quantifier. The use of a RIM quantifier to guide the evaluation of the overall distance essentially implies that more individual distances are satisfied the higher the similarity of two software projects it is. In the case of a RIM quantifier Q, the weights are generated as follows [8]:

$$w_j(P_1,P_2) = Q\left(\frac{\sum_{k=1}^{j} u_k}{T}\right) - Q\left(\frac{\sum_{k=1}^{j-1} u_k}{T}\right)$$

where u_k is the importance weight associated to the k[th] variable describing software project, and T is the total sum of all importance weights u_k

We note that the weights w_j used in the formula (3) will generally be different for each (P_1, P_2). This is due to the fact that the ordering of the individual distances $d_{v_j}(P_1,P_2)$ will be different and consequently lead to different u_k

5. Illustration

In this section, we illustrate, by an example, the computing process of the overall distance using different linguistic quantifiers and we compare the obtained results with those obtained in the case of '*all'* and '*there exists'* quantifiers. The original intermediate COCOMO'81 database was chosen as the basis for this example [2]. 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

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 uses conventional quantization where the values are classical intervals (see [2], pp. 119). To use the proposed measures and because the advantages of representation by fuzzy sets rather than classical intervals, the 15 cost drivers must be fuzzified. Among these, we have retained 12 attributes that we had already fuzzified (see the case of the DATA cost driver in the appendices); the other attributes are not studied because these relative descriptions are insufficient [3]. The goal of this illustration is to evaluate the similarity between the COCOMO'81 software projects assuming that are described by these 12 cost drivers.

For simplification, we discuss only the case of the similarity of the first project P_1 and the five first 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 Access to store data and Microsoft Visual Basic to implement the various processing steps. The prototype allows us to try various RIM linguistic quantifiers Q to the COCOMO'81 historical data; normally, each environment must define its appropriate quantifier by studying its features and its requirements. In this illustration, we use RIM quantifiers defined by :

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

To calculate the weights w_j's, we must determine the importance weights u_k's associated to the 12 variables describing COCOMO'81 software projects (Table 1). For this, we use 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 1).

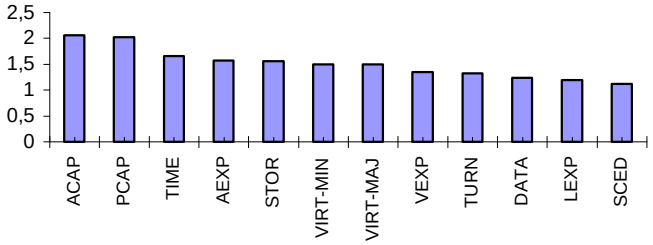

Figure 1. Comparison of the productivity ratios for the 12 variables describing COCOMO'81 software projects.

Cost driver (V_j)	Weight U_j	Cost driver (V_j)	Weight U_j
ACAP	2,05	VIRT-MIN	1,49
PCAP	2,30	VEXP	1,34
TIME	1,66	TURN	1,32
AEXP	1,57	DATA	1,23
STOR	1,56	LEXP	1,20
VIRT-MAJ	1,49	SCED	1,11

Table 1 : The weights u_j's associated to the 12 COCOMO'81 cost drivers

Max-min aggregation $\mathbf{d_{v_j}(P_1, P_n)}$
$d(P_1, P_n)$

			P_1	P_2	P_3	P_4	P_5
P_1	a	Max	**1**	**1**	**1**	**1**	**0,84096**
		1/100	0.99824	0,98529	0,97938	0,99095	0,82482
		1/30	0,99416	0,95189	0,93298	0,97018	0,78841
		1/20	0,99128	0,92879	0,90124	0,95564	0,76343
		1/15	0,98842	0,90629	0,87061	0,94134	0,73927
		1/10	0,98275	0,86304	0,81253	0,91344	0,69331
		1/7	0,97559	0,81069	0,74370	0,87890	0,63855
		1/5	0,96625	0,74617	0,66117	0,83505	0,57245
		1/3	0,94533	0,61618	0,50335	0,74178	0,44446
		1	0,85691	0,24612	0,132939	0,417857	0,13026
		3	0,69875	2,0948E-02	2,9783E-03	8,4882E-02	4,4606E-03
		5	0,62337	2,3918E-03	7,4220E-05	1,9233E-02	2,0768E-04
		7	0,58426	3,2462E-04	1,8898E-06	4,6167E-03	1,1676E-05
		10	0,55523	1,8895E-05	7,7274E-09	5,7808E-04	1,8904E-07
		15	0,53637	1,8861E-07	8,0929E-13	2,0205E-05	2,3544E-10
		20	0,52940	1,9602E-09	8,4763E-17	7,9559E-07	3,0742E-13
		30	0,52445	2,1614E-13	9,2985E-25	1,7490E-09	5,3075E-19
		100	0,52145	4,4160E-41	1,7777E-80	1,1339E-26	2,4419E-59
		Min	**0,52144**	**0**	**0**	**0**	**0**

Table 2. Results obtained for $d(P_1, P_i)$ when aggregation uses various linguistic quantifiers

The table 2 shows the results obtained using only the *max-min* aggregation to evaluate the individual distances (formulas (1.1)). We have not used *sum-product* aggregation (formula (1.2)) for two reasons [5]:

- 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.

In this application, we have used different values for α (1/100, 1/30,…,1,…,30,100) to evaluate the overall distance $d(P_1,P_i)$. From the obtained results, we conclude that $d(P_1,P_i)$ using α-Rim quantifiers is always between the *min* and the *max* of $d_{v_j}(P_1, P_i)$. Also, $d(P_1,P_i)$ tends towards the *max* of $d_{v_j}(P_1, P_i)$ when α tends towards zero (see the case of α equal to 1/100); $d(P_1,P_i)$ tends towards the *min* of $d_{v_j}(P_1, P_i)$ when α tends towards infinity (see the case of α equal to 100). Indeed, the *min* (*max*) aggregation is the less (more) softer combining for RIM linguistic quantifiers; so, $d(P_1,P_i)$ is null (equal to 1) when using *min* (*max*) operator if only one $d_{v_j}(P_1, P_i)$ is the same.

By contrast, the other α-RIM quantifiers can tolerate some restrictions associated with the needs of a given environment. First, we can choose the appropriate weights associated to each linguistic variable describing 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's. These weights represent the importance associated to 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 where α is equal to 1. Consequently, formula (3) gives the ordinary weighted average.

6. Conclusion

In this paper, we have proposed to use linguistic quantifier guided aggregations to evaluate the similarity between two software projects; this improve our measures for software project similarity proposed in [5]. The most significant advantage of this is that for each organization, the appropriate quantifier can be chosen and used in the evaluation of the similarity. Further research work has been initiated to look at the use these measures to complete the empirical

validation of the estimation effort by analogy approach started in [5].

7. 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] 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

[4] 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

[5] A. Idri, and A. Abran, "A Fuzzy Logic Based Set of Measures For Software Project Similarity: Validation and Possible Improvements", *accepted at IEEE Metrics 2001, London, 2-6 Avril*, 2001.

[6] 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

[7] 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

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

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

[10] 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

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

Appendices : Fuzzification of the DATA cost driver

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

Table 3. DATA cost driver ratings.

where:

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

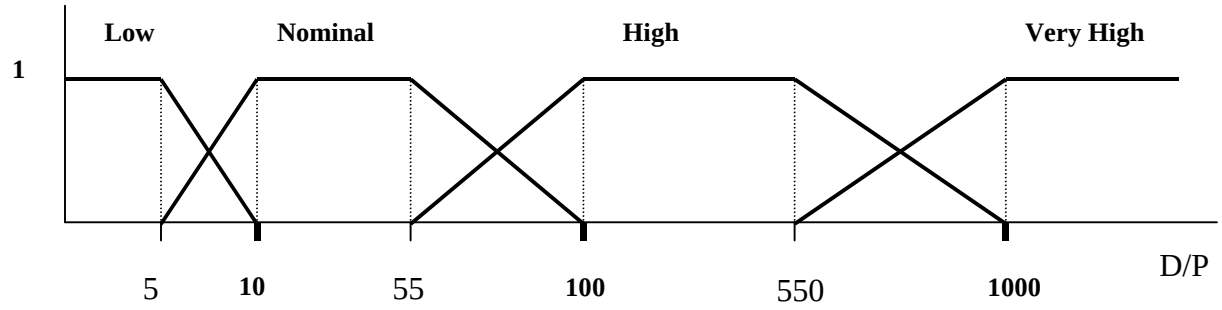

Figure 2. Membership functions of fuzzy sets defined for the DATA cost driver