Can Fuzzy Mathematics enrich the Assessment of Software Maintainability?

Gerardo Canfora, Luigi Cerulo, and Luigi Troiano

RCOST - Research Centre on Software Technology
Department of Engineering - University of Sannio
Palazzo ex-Poste, Viale Traiano 82100 Benevento, Italy

Abstract. Software maintainability depends both on qualitative and quantitative data. Existing maintainability models aggregate data into hierarchies of characteristics with given dependencies. However, data used to score the characteristics can be uncertain or even completely unknown. Therefore, it would be meaningful to evaluate sensitivity of the aggregated result, i.e. the maintainability, with respect to the uncertainty and incompleteness of data. In addition, real cases require an aggregation model able to evaluate the impact of changing the dependencies among the characteristics in the hierarchy on the maintainability. In this paper we argue that fuzzy mathematics can help to solve these problems. In particular, we show how a fuzzy aggregation model can be adopted to evaluate maintainability according to a hierarchical model.

1 Introduction

Most maintainability assessment models describes maintainability as a function of directly or indirectly measurable attributes $A_1, \dots, A_n$, that is:

$$M = f(A_1, A_2, \dots, A_n) \tag{1}$$

Software metrics are used to quantitatively characterize the attributes. Metrics are then aggregated into a single index expressing the level of maintainability. This approach is fast, simple, and consistent, basing its robustness on the widely studied fields of software metrics and empirical software engineering.

Often, the models that link metrics to maintainability are hierarchical, i.e. measurable attributes are aggregated into intermediate characteristics, and these are further aggregated into the final maintainability level. For instance, the standard ISO/IEC 9126 [1] derives maintainability from four characteristics: *analyzability*, *changeability*, *stability*, and *testability*. However maintainability characteristics as prescribed by ISO/IEC 9126 are still too abstract to develop a metric directly. More detailed models are proposed by Berns [2], Sneed and Merey [16], and Oman et al. [14]. Each of these models derives the maintainability index by aggregating lower level attributes according to a hierarchical view of maintainability. In particular, Oman et al. [14] develop a tree structured model containing 92 known maintainability attributes starting from 35 published works on software maintainability. In this model Software Maintainability depends on *management*, *operational environment*, and *target software system*, as depicted in Fig.1.

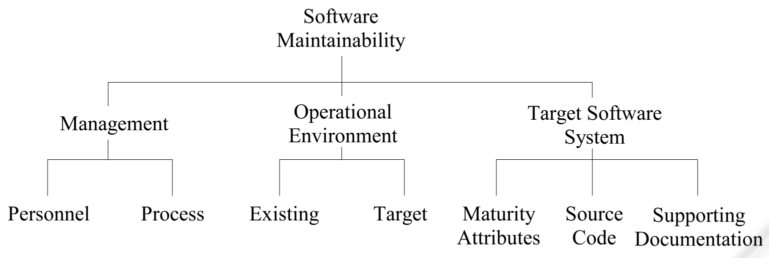

Fig. 1. Maintainability characteristics [14]

For the sake of simplicity, in our reasoning we focus on the target software system sub-tree as depicted in Fig.2.

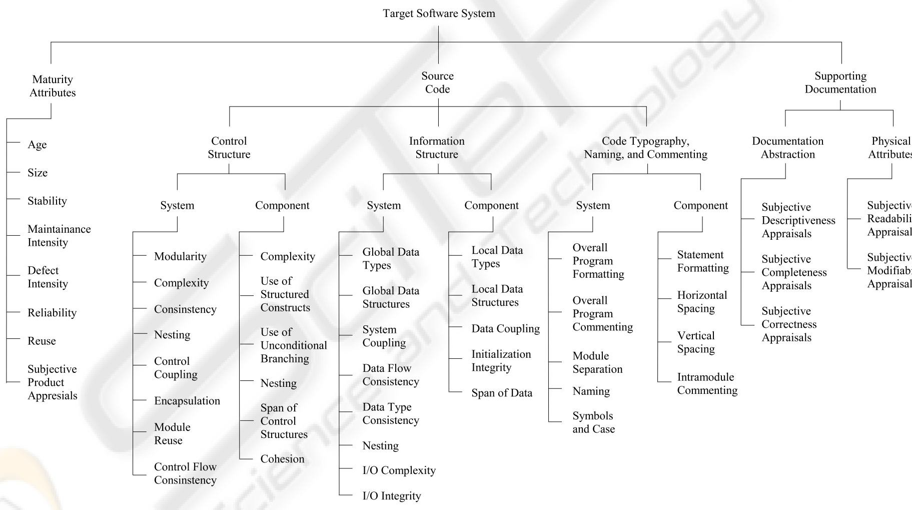

Fig. 2. Maintainability attributes [14]

Looking at the model, we notice that quantitative metrics come along with qualitative attributes. For instance, evaluating *software maturity*, we find quantitative measurements such as *age*, *size*, *defects intensity*, together with qualitative assessments such as *installation complexity appraisals*, *programming language complexity appraisals*, *development effort appraisals*. This information is usually collected by means of questionnaires [15], to which interviewed people answer by selecting a verbal judgment close to their own opinion. This is in accordance with the fact that software maintainability needs to take into account human factors and not just internal and structural factors.

For each attribute a software metric is defined and the following aggregation model is presented:

$$M = \prod_{i=1}^{m} W_{C_i} \left(\frac{\sum_{j=1}^{n} W_{A_j} M_{A_j}}{n} \right)_i \tag{2}$$

where:

- W_{C_i} is the weight influence of maintainability characteristic C_i
- W_{A_j} is the weight influence of maintainability attribute A_j
- M_{A_j} is the metric measure of maintainability attribute A_j

Usually quantitative and qualitative data aimed to assess software maintainability are simply aggregated together by means of numeric operators, sometimes forgetting scale assumptions and manipulation constraints.

There is a predicative model underlying Eq.(2), since product can be assumed as an *and-like* operation, whilst average as a *compensation* operation, as discussed later. Another relevant aspect to take into account regards uncertainty of information deriving from vagueness and incompleteness of data. Vagueness is due to impreciseness and ambiguity of human judgments. Moreover, data could be not fully available. Both vagueness and incompleteness induce a higher uncertainty and unpredictability of maintainability index.

In the attempt of building a model that simulates the way humans make evaluations we should pay attention to the way quantitative data are transformed into qualitative assessments. Predicative models require to express predicates on quantitative data on which evaluations of software products are based. Such predicates are expressions of judgments regarding quantitative data. An example is the well known rule of keeping the McCabe cyclomatic complexity [10] lower than or equal to 7 in order to keep complexity manageable. Predicates map quantitative data onto qualitative judgments. This mapping has been often assumed implicitly; even the simple normalization of data can be seen as qualitative assessment of quantitative data.

In order to consider issues discussed above, key points of our research are

- a continuum of aggregation models should be explored to have a broader view of software maintainability;
- qualitative data are relevant as well as quantitative data in software maintainability assessment;
- data could not be fully available;
- attributes and characteristics can have a different weight into determining the overall maintainability.

Our hypothesis is that fuzzy mathematics, and in particular fuzzy aggregation models can adequately address all these issues. In particular we claim that a predicative model based on fuzzy rules represents an insightful way to address the problem of assessing software maintainability. Such a model is able to highlight logical dependencies among characteristics and attributes that affect the overall maintainability and its uncertainty. The whole problem can be viewed as a Multi-Criteria Decision Making (MCDM) problem [9], thus suggesting the use of an aggregation model. The reminder of this work is organized as follows: in Section 2 we present a fuzzy aggregation model namely OFNWA; Section 3 contains an example of application; Section 4 closes the paper with some concluding remarks and future works;

2 Fuzzy aggregation and hierarchical maintainability models

2.1 A continuum of aggregation models

We use Ordered Fuzzy Number Weighted Averaging (OFNWA) [5] to address the need for a continuum of aggregation models. OFNWA derives by the concept of Ordered Weighted Averaging (OWA) introduced by Yager [17]. It is defined as

$$F^{(n)}(A_1, ..., A_n) = \sum_{i=1}^{n} w_i B_i \tag{3}$$

where $B_1, \ldots, B_n$ is a decreasingly ordered disposition of fuzzy numbers $A_1, \ldots, A_n$ so that B_i is the i-th largest element. Weights are such that

$$\sum_{i=1}^{n} w_i = 1 \tag{4}$$

Conversely from usual weighted average, in OWA/OFNWA aggregation model weight w_i is not associated with the i-th element but with i-th largest one. There are three special cases of OWA operators [17].

$$\begin{aligned}
&w^T = w^* = [1\ 0 \ldots 0] \\
&F^*(a_1, \ldots, a_n) = \max_{i=1,..,n}(a_i) \\
\\
&w^T = w_* = [0\ 0 \ldots 1] \\
&F_*(a_1, \ldots, a_n) = \min_{i=1,..,n}(a_i) \\
\\
&w^T = w_{ave} = [1/n\ 1/n \ldots 1/n] \\
&F_{ave}(a_1, \ldots, a_n) = \frac{1}{n}\sum_{i=1}^{n} a_i
\end{aligned} \tag{5}$$

More in general, families of OWA/OFNWA operators can be derived by particular weight distributions [11, 12, 19]. OWA/OFNWA operators offer an ideal bridge between and-like operators and or-like operators. An important measure associated to OWA/OFNWA operators is *orness* (aka *attitude of character*, σ) [17], defined as

$$\sigma = \frac{1}{n-1} \sum_{i=1}^{n-1} (n-i) w_i \tag{6}$$

It is easy to verify that

$$\begin{aligned} \sigma^* &= 1 \\ \sigma_{ave} &= 0.5 \\ \sigma_* &= 0 \end{aligned} \tag{7}$$

Orness can be interpreted as a measure of aggregation *severity*: the higher it is, the more attention to worse attributes is paid. It provides a measure of the distance between the set of and-like operators and the set or-like operators. When orness is 0, OWA aggregation corresponds to min operation, that is the biggest among the and-like operators. Conversely, when orness is 1, OWA aggregation corresponds to max operation, that is the smallest among the or-like operators. With orness equal to 0.5 we get the arithmetic average.

2.2 Incompleteness of data

Maintainability assessment often deals with incomplete and uncertain data. This is something we need to take into account when questionnaires are adopted to collect data [8], as incompleteness and uncertainty of data affect the information made available to software engineers for decision making. It has been argued [4] that although incompleteness and uncertainty contribute to make decision effects more unpredictable, it is useful to keep their effects separate. Information is uncertain when we are not sure about the "value" assumed. Information is incomplete when it is not completely available, so that we can have certain information but incomplete and vice-versa. For instance, let us consider all cases where it is not possible to provide an answer to a questionnaire, checking the "Don't Know" box (DK), or when a quantitative measure is not available. Traditional numeric approaches simply ignore any source of uncertainty. However, taking uncertainty into account is a key to manage risks and to improve the effectiveness of decision making. *Numbers with indeterminateness* [5] represent a way to deal with DK answers and more generally with incomplete data [4]. Numbers with indeterminateness are defined as

$$F = \xi \cdot \mathbf{I} + \zeta \cdot F_N \tag{8}$$

where F_N is the *numeric component*, $\mathbf{I}$ is the *indeterminate element*, the coefficient ξ is called *indeterminateness*, and $\zeta = 1-\xi$ is called *determinateness*. While F_N keeps all available qualitative and quantitative data, $\mathbf{I}$ represents "total ignorance": we assume not to be able to describe it by any membership function. Thus $\mathbf{I}$ does not belong to the realm of ordinary numbers. It is a primitive entity that can be just considered symbolically. We can use ξ as direct measure of information incompleteness. When $\xi = 0$ ($\zeta = 1$) information is complete; on the opposite side $\xi = 1$ ($\zeta = 0$) means information is not complete at all.

2.3 Relevance of criteria

Aggregation as defined in Eq.(3) does not deal with the relevance of attributes and characteristics. When a characteristic/attribute is irrelevant, it should be not considered at all; otherwise it must be took into account. We can describe this reasoning using the following set of propositions

$$\begin{aligned} p_{i,1} : & \textit{ if } \operatorname{imp}(C_i) \text{ is high} \\ & \textit{ then } \text{aggregation should consider } C_i. \\ p_{i,2} : & \textit{ if } \operatorname{imp}(C_i) \text{ is low} \\ & \textit{ then } \text{aggregation can ignore } C_i. \end{aligned}$$

where $\operatorname{imp}(C_i)$ is the relevance associated to characteristic/attribute C_i. Ignoring a characteristic/attribute means that aggregation result is independent of it: the result does not change no matter which value it assumes. These rules are applied in sequence to each criterion. Thus, we will consider the aggregation applied to different subsets of criteria, including also the empty one. In that case we are not able to derive the result. This produces a number with indeterminateness, as described in [5], whose ξ depends on the relevance given to characteristic/attribute and the completeness of available data.

2.4 Quantitative and qualitative data

In [7] we have proposed a formal characterization of transformations on quantitative data into qualitative assessments by means of *judgment functions*, and we discussed how the usual mathematical properties (i.e. derivative and measure) of functions have a semantic interpretation in the light of qualitative assessment of data. Briefly, a judgment function is a map that transforms measures into judgments

$$\psi : \varXi \mapsto \varUpsilon \tag{9}$$

where $\varXi$ is the *quantitative measurement domain* and $\varUpsilon$ is called the *judgment domain.* To better qualify the concept of *judgment* we can refer to it as [7] *dimension of measure associated to the qualitative perception of objects.* If we assume $\varUpsilon$ to be a scale, we can look at function ψ as a derived measure. If we consider also that the aggregation itself is a map between judgments, the result can be viewed as a derived measure. A judgment function transforms quantities measuring an object attribute to a degree of criteria satisfaction. Thus judgment domains and functions can be characterized in terms of some semantic properties.

3 An example of application

As an example of application we used the OFNWA aggregation model to assess maintainability of a small open source software project (about 6 KNCSS[1])

[1] Thousand of Non Comment Source Statements

according to model depicted in Fig.1 and Fig.2. The project regards the development of a Linux device driver for the Emu10k chipset used in some PC Sound Cards (http://sourceforge.net/projects/emu10k1/). By inspecting the CVS tree at SourceForge, we got data about the project history since it started in 2001. We derived most of the software metrics by inspecting the driver source code trough the HP-MAS tool [13]. We investigated other sources of data such as bug reports, change logs, patches, and diffs, in order to derive other metrics such as age, size, stability, maintenance intensity, defect intensity, reliability, and reuse. Qualitative attributes, such as product and document appraisals, have been assessed by the use of questionnaires filled in by researchers in our group. All data we used in the evaluation process with a brief description are available on our web site (http://cise.rcost.unisannio.it). In Tab.1, we report the average percent fit at various category level using Eq.(2).

Table 1. Results by applying Eq.(2)

Category	Weighted formula
Maturity	46.03
Source Code	59.99
Documentation	40.62
Maintainability	11.21

Then, we applied the OFNWA aggregation to assess the system maintainability by using a tool we developed [3]. The results are shown in Fig.3, Fig.4 and Fig.5. Graphs show how maintainability and characteristics scores change depending on the different perspective given by different level of orness (Eq.(6)). In particular, Fig.3 shows maintainability score for low orness levels. The result is in accordance with Tab.1. It is made by a continuous sequence of fuzzy numbers whose uncertainty is bounded by dotted lines. We notice that maintainability grows with higher levels of orness: the higher is the level of orness, the more relevance is given to better characteristics, the better it is the aggregation result. Our analysis is restricted to low levels of orness ($\sigma \in [0, 0.5]$), due to the fact that maintainability is viewed as composed by complementary characteristics. Thus, an and-like aggregation is required, and this is consistent with low levels of orness. Dotted lines give an information regarding uncertainty of maintainability. This means the maintainability index, whose prototypic value is marked by the continuous line, varies within the band, with values shading toward the borders marked by dotted lines. Moreover Fig.3 makes a section view of maintainability index at some specific orness values, showing the resulting fuzzy numbers.

We can go further in our analysis wondering in which direction to invest in order to improve the software maintainability. Thus, we need to investigate target software characteristics. Fig.4 shows how the assessment of each characteristic varies within an interval of orness values. In particular the *source code* characteristic has been evaluated for low levels of orness, since it is considered composed by complementary sets of sub-characteristics. Therefore the aggregated value is computed by means of an and-like aggregation. The result is in accordance with Eq.2 (see Tab.1). The dotted circle in Fig.4 highlights the area of maximum similarity between OFNWA aggregation and the original Oman et al. model [14]. Indeed, the value in Tab.1 has been computed as weighted product, that is part of and-like aggregations, of source code sub-characteristics. The value improves for higher levels of orness. Differently, the score assigned to documentation and maturity is in accordance with Tab.1 when the orness is about $\sigma = 0.5$. This concords with the weighted averaging formula used for their evaluation.

However Fig.4 allows us to have a broader view of result than the simple application of Eq.(2), providing visibility of surrounding values provided by the continuum of models. In this way we are able to make a deeper analysis. The source code is good enough and no urgent operation is required on it to improve maintainability. Moreover, we notice that although documentation and maturity have pretty much the same score in Tab.1, their profiles shown in Fig.3 are different. Documentation evaluation exhibits a larger variation within the considered orness range. This means the most of documentation attributes have been judged as bad or good, with a few judgments in the average. Thus, a targeted improvement of some documentation aspects entails an improvement in software maintainability. Such considerations would not have been possible with traditional numerical techniques.

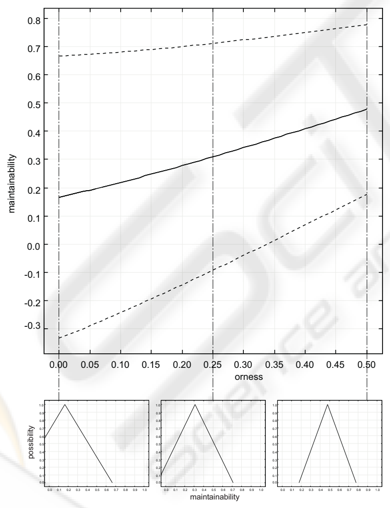

Fig. 3. Maintainability

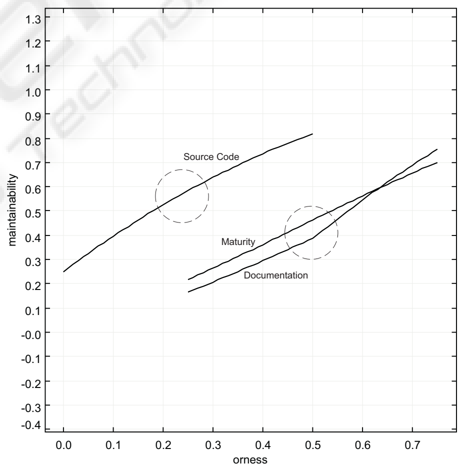

Fig. 4. Maintainability characteristics

Finally, Fig.5 shows how maintainability changes when some answers are not provided (DK). In particular we considered subjective appraisals about documentation and maturity not provided. We notice that although there is just a slight variation in prototypic values (continuous lines), there is a much larger variation in uncertainty. Therefore, the decision graph suggests that we cannot be confident in the result and we really need to get data not provided. Instead Fig.6 shows an example in which DK answers do not affect heavily the profile of maintainability.

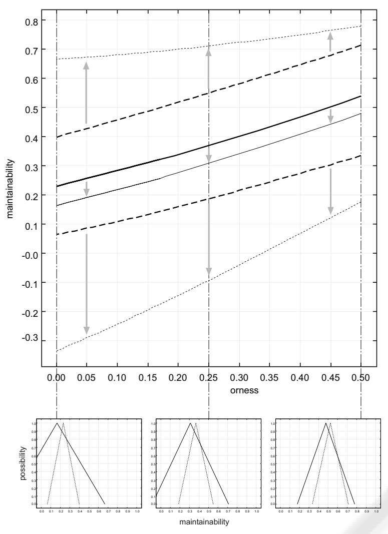

Fig. 5. Maintainability variation (1)

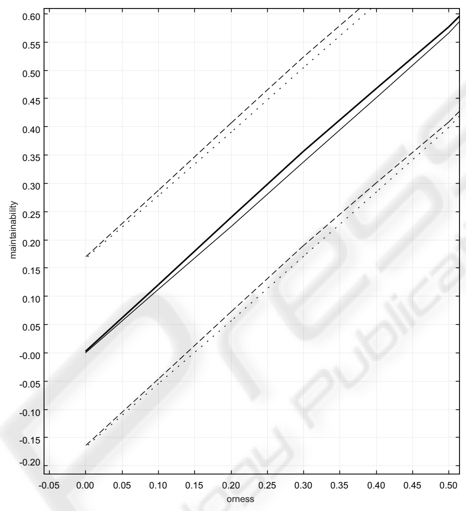

Fig. 6. Maintainability variation (2)

4 Conclusions and future works

This paper has shown how fuzzy mathematics can be used to assess the maintainability of a software system according to a hierarchical model, namely [14]. The advantage of using fuzzy mathematics is twofold:

- to consider the vagueness and incompleteness of available data, as well as mixing quantitative and qualitative data;
- to analyze how the maintainability change when considering different kinds of aggregation among characteristics and attributes.

Of course, more sophisticated models can be considered, for example by considering multi-expert evaluations of subjective attributes. In short, the aim of this paper is not to provide a definitive model, but to demonstrate how, using a fuzzy aggregation model, a rich set of analysis on software maintainability can be performed, thus empowering managers and decision makers.

References

1. (1999). *ISO/IEC 9126-1:2001. Part 1: Quality model.* ISO.
2. Berns, G. M. (1984). Assessing software maintainability. *Communications of the ACM*, 27(1):14–23.
3. Canfora, G., Cerulo, L., Preziosi, R., and Troiano, L. (2003). A tool for decision support implementing OFNWA approach: a case study. In *Proceedings of the International conferrence on Software Engineering and Knowledge Engineering*, pages 714–720.
4. Canfora, G. and Troiano, L. Dealing with the "don't know" answer in risk assessment. In *Proc. Int. Conf. on Enterprise Information Systems - ICEIS'03.*
5. Canfora, G. and Troiano, L. (2001). An extended model for ordered weighted averaging applied to decision making. Technical report, RCOST — University of Sannio.
6. Canfora, G. and Troiano, L. (2003a). *A rule-based model to aggregate criteria with different relevance*, volume LNAI 2715 of *Lecture Notes in Computer Science*, pages 311–318. Springer, Istanbul.
7. Canfora, G. and Troiano, L. (2003b). Transforming quantities into qualities in assessment of software systems. Dallas, USA. COMPSAC 2003.
8. Converse, P. (1964). *The Nature of Belief Systems in Mass Publics*, pages 206–261. Free Press, New York, d. e. apter edition.
9. Fodor, J. and Roubens, M. *Fuzzy Preference Modelling and Multi-Criteria Decision Support*, volume 14 of *D.* Kluwer, Dordrect.
10. McCabe, T. (1976). A software complexity measure. *IEEE Transactions on Software Enginnering*, SE-2(4):308–320.
11. O'Hagan, M. (1987). Fuzzy decision aids. In *21th Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 624–628. IEEE and Maple Press.
12. O'Hagan, M. (1988). Aggregating template rule antecedents in real-time expert systems with fuzzy set logic. In *22th Annual IEEE Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA.
13. Oman, P. (1992). HP-MAS : A tool for software maintainability assessment. SETL Report #92-06-TR, University of Idaho.
14. Oman, P. and Hagemeister, J. (1992). Metrics for assessing a software system's maintainability. In *Proceedings of the International Conference on Software Maintenance 1992*, pages 337–344. IEEE Computer Society Press.
15. Oman, P. and Hagemeister, J. (1994). Construction and testing of polynomials predicting software maintainability. *Systems Software*, 24:251–266.
16. Sneed, H. M. and Mérey, A. (1985). Automated software quality assurance. *IEEE Transactions on Software Engineering*, 11:909–916. Special Issue on COMPSAC 1982 and 1983.
17. Yager, R. (1988). On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Trans. on Systems, Man, and Cybernetics*, 18(1):183–190.
18. Yager, R. (1992). On the inclusion of importances in multi-criteria decision making in the fuzzy set framework. *Int. J. of Expert Systems: Research and Application*, (5):211–228.
19. Yager, R. (1993). Families of owa operators. *Fuzzy Sets and Systems*, (59):125–148.