FULSOME: Fuzzy Logic for Software Metric Practitioners and Researchers

Stephen G. MacDonell, Andrew R. Gray, and James M. Calvert
Department of Information Science
University of Otago, PO Box 56, Dunedin, New Zealand
+64 3 4798135 (phone) +64 3 4798311 (fax), stevemac@infoscience.otago.ac.nz

Abstract

There has been increasing interest in recent times for using fuzzy logic techniques to represent software metric models, especially those predicting development effort. The use of fuzzy logic for this application area offers several advantages when compared to other commonly used techniques. These include the use of a single model with different levels of precision for inputs and outputs used throughout the development life cycle, the possibility of model development with little or no data, and its effectiveness when used as a communication tool. The use of fuzzy logic in any applied field however requires that suitable tools are available for both practitioners and researchers – satisfying both interface and functionality related requirements. After outlining some of the specific needs of the software metrics community, including results from a survey of software developers on this topic, the paper describes the use of a set of tools called FULSOME (Fuzzy Logic for Software Metrics). The development of a simple fuzzy logic system by a software metrician and subsequent tuning are then discussed using a real-world set of software metric data. The automatically generated fuzzy model performs acceptably when compared to regression-based models.

1 Introduction

Fuzzy logic has recently been recognized as a useful addition to the software metrician's toolbox for developing models of the software development process [2]. In this way it has joined both traditional and robust statistical techniques, regression and classification trees, case-based reasoning, and neural networks as part of the model-based attempt to better manage software development [3].

This is of course only one aspect of improving project management – but it is an important area and one where fuzzy logic techniques have much to offer. It is not suggested here that other aspects of software development project management are not equally, or more, important in terms of the potential benefits from improvement. However model development is an area that can be improved without requiring drastic changes to development practices, and is not limited in terms of the types of system development projects to which it can be applied.

1.1 Survey

In a recent survey of software developers in New Zealand, it was found that a surprisingly high 31 out of the 44 information system managers who responded had heard of fuzzy logic. Of the 36 managers who were actively involved in managing development projects, 11 were interested in using fuzzy logic techniques, 23 stated that they would need to know more about the technique before making a decision, and only two did not think that fuzzy logic techniques would be useful to them. See Table 1 for more details.

Three advantages of fuzzy logic were proposed to readers of the survey and they were asked to indicate their interest or disinterest in each feature. These were being able to use expert knowledge for model development, using linguistic labels before numerical values are known, and having less precise estimates from the model. Interestingly, those expressing some interest in the use of fuzzy logic found each of the three advantages equally appealing (Table 1). The percentages do not sum to one hundred since most respondents selected more than one advantage.

No relationships were found between the organization's software development department size in terms of equivalent full-time personnel (six levels for full-time equivalent employees) and type (commercial, government, and software-house), and their knowledge or interest in fuzzy logic. These associations were all initially tested using χ^2 tests at the 0.05 level.

While the survey results reported above certainly reflect a self-selected sample since many surveys were not returned, they are encouraging in that they suggest that a significant number of project managers are prepared

Respondents	Number	Percentage
Respondents	44	—
Heard of fuzzy logic	31	71%
Actively managing projects	36	82%
Of those actively managing projects		
Interested in using fuzzy logic	11	31%
Would need to know more about fuzzy logic	23	64%
Not interested in fuzzy logic	2	6%
Advantages of fuzzy logic to those interested or potentially interested in using fuzzy logic		
Using expert knowledge for model development	19	56%
Linguistic inputs in place of numerical values	19	56%
Linguistic outputs in place of numerical values	21	62%

Table 1: Results of a survey on fuzzy logic for project management

to use such a technique. Some of these organizations are now being approached to evaluate the FULSOME system, as described below, in a more practical setting. Such feedback will be essential to the development of a truly usable system for practitioners. They will also be involved in developing a set of standard practices for the use of fuzzy logic for software metric model development.

1.2 Adoption of fuzzy logic

Despite the high level of interest that was shown in fuzzy logic, only one organization responding to the survey was currently using fuzzy logic models as part of its software development management practices and another had done so in the past. One possible reason for this imbalance between interest and practice could be the lack of guidelines for metrics practitioners using fuzzy logic and accessible software packages.

One strength of Function Point Analysis (FPA) [1], the most commonly used software metric technique for effort estimation, is that it is well documented with quality control achieved through accreditation exams. Supporting software is also widely available and this software is often integrated into the development process. These two features – software and guidelines – are seen as essential for the adoption of fuzzy logic techniques in this field (as with any other technique).

Of course the software would need to satisfy many non-technical criteria as well as supporting the inference process itself. A user-friendly interface with built-in support for a generic development process model would be necessary for widespread usage. Ideally it would also be capable of communicating with standard analysis and design, source code editing and generation, and project management software. In this way many of the currently used fuzzy inference systems are unsuitable for non-specialists and would also fail to support features of modern large scale software development.

2 Project management

Traditionally, software metric models for predicting development effort have involved using some size measure(s) for the system which are then used in a simple linear regression model for estimating development effort. In some cases the size measure is transformed beforehand, as in COCOMO, but often the models ignore any economy (or diseconomy) of scale effects. In other cases, some assessment of complexity is used to produce a reweighted size, as in FPA [1].

These models have all failed to achieve even the seemingly unambitious goal of predicting effort using system specification based measures (available before coding has begun proper) to within $\pm 25\%$ for at least 75% of the systems (this is often referred to as $pred(25) \geq 0.75$). In fact, most published results fail to achieve this level even on *fitted samples*.

This inadequacy of model performance in terms of predictive accuracy is, perhaps unexpectedly, seen as encouraging towards the use of fuzzy logic for this application area. For what is essentially a predictive task, exact results are not realistically expected, nor are they necessarily required, let alone feasible in any case. This allows for the use of intuitive systems that provide "rough" estimates without having to resort to the all-too-common excessive tuning of fuzzy logic systems that trade off interpretability in favor of numerical accuracy.

Below some of the main advantages of fuzzy logic in software metric model building are discussed in turn. These are the use of a single model, the ability to cope with small or nonexistent data sets, robustness to data quality, and the use of fuzzy logic as a means of communicating project management issues.

2.1 Using a single model

One of the most appealing benefits of fuzzy logic for software metric models is the opportunity to use a single model (consisting of membership functions and rules) throughout the entire software development process. Table 2 from [4] shows one particular system for selecting the levels of precision across the development life-cycle. This is of course only one particular possibility, and others may be necessary depending on organization characteristics and practices.

Initially, at the analysis stage, very little information is available beyond using such terms as "large" and "high" for the system data model size and functional complexity, say. Later, at the design stage, more precise estimates can be made, for example "about 300" entities. Finally, during coding, fairly exact estimates can be made, such as 285 entities.

In order to cope with this gradual buildup of information most software metric models have required managers to make "guestimates" of the numerical independent variables both early in the life-cycle and subsequently. This discourages the use of such models since managers know that their earlier estimates are unlikely to be close to correct, and causes wild fluctuations in predictions as more information becomes available. There is a predictable reluctance by managers to provide values with an accuracy beyond their capabilities.

FPA, the most commonly used software metric for effort estimation, has been adapted for earlier use by lowering the information requirements. However this limited approach leads to multiple models that may or may not be entirely consistent. The fluctuation problem can again emerge, and managers are required to provide more information (with precise definitions) through the process.

In the same way, the effort predictions from a fuzzy logic model can be made with different levels of precision at different stages of the development life-cycle. For example, early estimates may be strictly linguistic ("high" effort, "medium" risk); later estimates may use fuzzy numbers ("about 1500" person-hours); and finally standard numerical estimates may be used (1525 person-hours).

2.2 Data availability and quality

For a variety of reasons, software metrics data is difficult to collect and quickly become out-of-date as development technologies and methodologies change. Further, organizations are seldom willing to share their data with other organizations (and when they do it is without much of the necessary detail regarding the development process required to calibrate the results to a different environment). These features of software metric data leave the software metrician with in general only small quantities of data with which to develop a model, if any at all.

Fuzzy logic models can be easily constructed without any data whatsoever, or with a small sample used to validate the model. This is another striking advantage when compared to data-driven model building techniques such as neural networks, regression, and case-based reasoning. As such fuzzy logic provides a useful argument to the often encountered criticism of software metric models that data collection is too difficult and expensive.

Similarly, by reducing the reliance on data to derive the model some of the problems with data quality can be overcome. This leads to the issue of model robustness as discussed next.

2.3 Model robustness

Software metric data is often contaminated by unusual systems that are impossible to identify based on the measured variables, leading to strong robustness requirements for any automated model building technique. In order to reduce the risk of influence from these observations, robust statistical methods can be used and the final model can be examined for plausibility. This is one reason why statistical models have remained simple in their structure and a significant disadvantage of neural network and other "black box" type techniques.

Since fuzzy logic models are easily interpreted they can be, comparatively speaking, easily checked for reasonableness with experts. This provides some protection from influential points adversely affecting the empirical model building process.

2.4 Communication

Another useful feature of fuzzy logic models is as a communication tool for management. Equations derived from regression models are not always easy to explain if interactions and transformations are present. Neural network models are even worse in this respect. With a rule-based system this interpretability can be maintained with a relatively powerful mapping mechanism. The only other technique used here that provides such transparency in reasoning is case-based reasoning, which suffers greatly from high data requirements.

Phase	Inputs	Outputs
Analysis	fuzzy label	fuzzy label
Design	fuzzy number	fuzzy number
Coding	crisp value or fuzzy number	fuzzy number
Testing	crisp value	fuzzy number
Maintenance (small project)	crisp value	fuzzy number

Table 2: Suggested levels of precision across the life-cycle when estimating development effort (from [4])

3 FULSOME

FULSOME provides the software metric model developer with a series of tools that include data entry and importing facilities, membership function construction, rule creation, inference (including tracing and visualisation), and online explanations of fuzzy logic. Additional tools include a basic initial system generator that uses fuzzy c-means clustering to derive initial membership functions and rules (with the automatic membership functions or ones that have been already created), and a model performance evaluator.

Figure 1 shows the base components of the system which would generally be used from top to bottom, left to right. The information in each of these modules is stored in separate files, which can be collected together within a single "system file". Standard options such as a variety of membership functions (triangular, trapezoidal, Gaussian, bell, and sigmoidal), t-norm and t-conorm options, and defuzzification strategies are all provided.

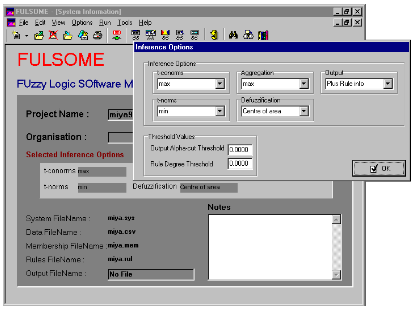
3.1 Generating systems

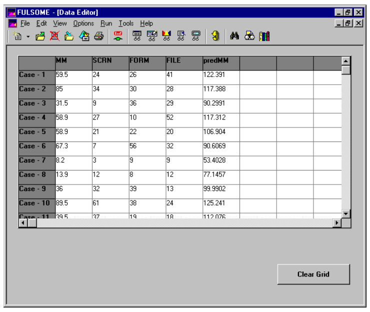
One algorithm for automatically generating membership functions is simply to select the desired number of functions and make the centers of each function the center of a data cluster. This simple approach is implemented in FULSOME by using fuzzy c-means clustering (with standard adjustable parameters). The simplicity of this approach is also consistent with the goal of making the model development process both transparent and convincing for managers.

1. select an appropriate mathematically defined function for the membership functions of the variable of interest (i), say $f_i(x)$
2. select the number of membership functions that are desired for that particular variable, m_i functions for variable i
3. call each of the m_i functions $f_{ij}([x])$ where $j = 1 \ldots m_i$ and $[x]$ is an array of parameters defining that particular function (usually a center and width parameter are defined, either explicitly or implicitly)
4. using one-dimensional fuzzy c-means clustering on the data set find the m_i cluster centers, c_{ij} from the available data
5. sort the cluster centers c_{ij} into monotonic (generally ascending) order for the given i
6. set the membership function center for f_{ij}, generally represented as one of the parameters in the array $[x]$, to the cluster center c_{ij}
7. set the membership function widths for $f_i j$ in $[x]$ such that $\sum_{n=1}^{m_i} f_{in}([c_{in}, \ldots]) = 1$, or as close as possible for the chosen $f(x)$ where this can not be achieved exactly (for example for triangular membership functions each function can be defined using three points, a, b, and c where a is the center of the next smaller functions and c is the center of the next larger function)

Similarly, rules can be extracted from data by using the same process of clustering with multiple dimensions (the number of dimensions matching the number of antecedents).

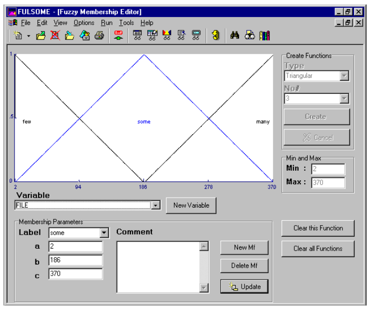
1. start with known membership functions $f_{ij}([x])$ for all variables, both input and output, where j represents the number of functions for variable i and $[x]$ is the set of parameters for the particular family of function curves
2. select the number of clusters k (which represents the number of rules involving the $k-1$ independent variables to estimate the single output variable)
3. perform fuzzy c-means clustering to find the centers (i dimensional) for each of the k clusters
4. for each cluster k with center c_k
   (a) determine the k^{th} rule to have the antecedents and consequent f_{ij} for each variable i where $f_{ij}(c_k)$ is maximized over all j.

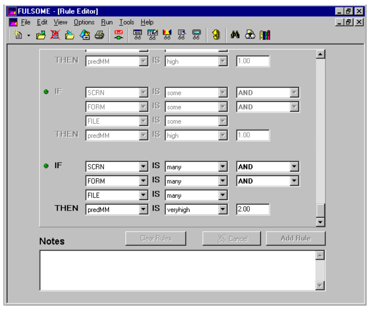


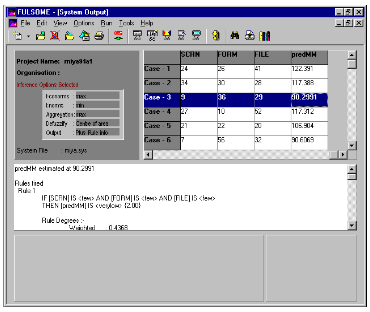


(a) System screen

(b) Data editor







(c) Membership function editor

(d) Rule editor

(e) Output screen

Figure 1: Structure of the FULSOME system

   (b) weight the rule, possibly as $\prod_{n=1}^{i} f_{ij}(c_k))$ or $\sum_{n=1}^{i} f_{ij}(c_k))$

5. combine rules with same antecedents and consequents, either summing, multiplying, or bounded summing rule weights together

6. (optionally) ratio scale all weights so that the mean weight is equal to 1.0 to aid interpretability

The automatic extraction of systems from data is seen here as an essential feature to encourage use of fuzzy logic. Organizations who are simply presented with a software package may not be, at least initially, comfortable creating a system from scratch and so this provides them with an opportunity for incremental learning.

4 Case study

For this section a set of size and effort data from a published study [5] was used to initialize a simple fuzzy logic system. The data consists of three size-related measures available from the system specifications and an effort measure for 48 systems. One of these systems was removed since it is by far the largest on all three measures and would not be used for model development in practice and nor would it have predictions made for it using a model developed with any of the other observations. This was presumably the motivation behind its inclusion in a paper testing the worthiness of robust regression techniques.

Only a subset (31 observations) of the data was used for the "training" procedure. The remainder (16 observations) was withheld for validating the developed models. The data is from a series of systems developed by a single organization in a relatively homogeneous environment, so this should provide a reasonably realistic test of the model building techniques.

Table 3 shows the results of a least squares regression model and the weighted fuzzy system on both the training and validation data. The error measures used here are the mean magnitude of relative error (MMRE) and the *pred()* measure which is the proportion of absolute relative errors which are less than the threshold (25%, 50%, and 100%).

Three membership functions were first extracted using one-dimensional clustering for each of the four variables. These membership functions were used to extract 11 unique rules from 30 cluster centers. The fuzzy system was unable to make predictions for all observa-

Technique	Data	MMRE	Pred(25)	Pred(50)	Pred(100)
Regression models					
Least squares regression	Training	0.96	0.32	0.52	0.74
	Testing	1.00	0.13	0.25	0.50
Fuzzy clustering with 30 clusters (11 unique rules)					
Replacing missing values with training set mean	Training	1.52	0.26	0.35	0.68
	Testing	2.27	0.31	0.38	0.50
Omitting missing values	Training	1.58	0.28	0.38	0.66
	Testing	1.87	0.31	0.38	0.54

Table 3: Results for the automatically generated fuzzy systems

tions due to insufficient rule coverage – leading to either using the mean effort from the training data as a "best estimate" or omitting the five missing observations (two from the training set and three from the testing set) when evaluating the system.

The results from the automatically generated fuzzy system are rather worse than the regression results according to the MMRE measure, but compare well using the *pred()* measures, outperforming the regression model at the lower levels on the validation set when using mean values for the missing predictions. It is therefore difficult to rank one model as better than another. The regression model is more consistent overall, but the fuzzy model is better on some systems and much worse on others. As such this automatically generated model would be a useful first draft for an expert to extend.

It should also be noted here that the fuzzy logic model was not augmented with any expert knowledge. We are not familiar with the projects in question beyond the published descriptions, and such knowledge could have been used to tune the rules beyond the automated extraction stage if available. Presumably this would have improved performance, or at least not have led to any deterioration. Rules for the uncovered regions in the input space would have been useful in this respect.

5 Conclusions

In this paper we have briefly discussed the benefits of using fuzzy logic modeling techniques for software metric models, specifically those for predicting development effort. These benefits, coupled with the results from the survey, suggest that there is considerable worth in the techniques discussed here being applied to this field. In addition, we have described a supporting suite of applications for the development of such fuzzy logic models and demonstrated a simple model development process using a real-world data set.

We are currently beginning a program of collaboration with three New Zealand organizations to investigate their use of fuzzy logic on *live* projects. The feedback from this will be used to develop the next generation of FULSOME along with more concrete guidelines for using fuzzy logic for software metric models.

References

[1] N. E. Fenton and S. L. Pfleeger. *Software Metrics: A Rigorous & Practical Approach*. PWS, 1997.

[2] A. Gray and S. MacDonell. Applications of fuzzy logic to software metric models for development effort estimation. In *Proceedings of the 1997 Annual meeting of the North American Fuzzy Information Processing Society - NAFIPS'97*, pages 394–399. IEEE, 1997.

[3] A. Gray and S. MacDonell. A comparison of model building techniques to develop predictive equations for software metrics. *Information and Software Technology*, 39:425–437, 1997.

[4] A. Gray and S. MacDonell. Fuzzy logic for software metric models throughout the development life-cycle. In *Proceedings of the 1999 Annual meeting of the North American Fuzzy Information Processing Society - NAFIPS'99*. IEEE, 1999.

[5] Y. Miyazaki, M. Terakado, K. Ozaki, and N. Nozaki. Robust regresison for developing software estimation models. *Journal of System and Software*, 27:35–16, 1994.