Full citation: Gray, A.R., & MacDonell, S.G. (1999) Membership function extraction from software development project managers, in Proceedings of the ICONIP'99/ANZIIS'99/ ANNES'99/ACNN'99 International Workshop on Future Directions for Intelligent Systems and Information Sciences. Dunedin, New Zealand, University of Otago, pp.235-240.

Membership Function Extraction from Software Development Project Managers

Andrew R. Gray and Stephen G. MacDonell
Department of Information Science
University of Otago, PO Box 56, Dunedin, New Zealand
+64 3 4795282 (ph.) +64 3 4798311 (fax), agray@infoscience.otago.ac.nz

Abstract

Software metrics are measurements of development processes, products, and resources. Once these measurements have been specified and collected they can be used as variables in empirically calibrated models for a wide range of project management purposes; including the task of predicting development effort based on some combination of size, complexity, and developer experience metrics. One difficulty encountered when using traditional algorithmic approaches to estimation has been the collection of the appropriate metrics required to use the model in its predictive capacity. Project managers are generally unable to make precise quantitative estimates for the independent variables, especially early in system development when these models are at their most valuable. The alternative of using qualitative values for the inputs, as in fuzzy logic, has been suggested but the stability and consistency of such labels has yet to be established, as well as considering the elicitation techniques available for deriving the membership functions. In this paper we examine the perceptions of data model size, functionality size, developer experience, and project effort in terms of three fuzzy membership functions from two separate surveys of project managers, each with a very different approach. The consistency of results across the two surveys is examined, and some discussion about the strengths and weaknesses of the two approaches is provided.

1. INTRODUCTION

Software metrics are measurements that can be made of the software development process (how the development is performed), the resultant products (including code, executables, and documentation), and resources used in the development process [3]. In effect the process runs along the project time-line, taking inputs (resources) such as developers and transforming them into products (such as code).

The most common use for software metrics is in the creation and calibration of models for managing software development, which has become a crucial task in many organizations as software becomes larger and more complex. In general terms any measurable aspect can be regarded as a software metric, although some are considerably more useful than others. Such models can have the goals of prediction (how long will it take to finish the module, how many errors will there be?), monitoring (is this project progressing at an acceptable rate?), controlling (how can we best reduce defects?), or assessing (did the new testing methodology improve the quality of the software?) development projects.

One of the most popular uses of such models has been development effort prediction. The inputs into models of development effort tend to include some aspects of system size, system complexity, and developer experience. There are many other influential features of projects, but there are limitations brought about by cost and practical considerations. The output may be any measure of effort, such as person-days, and may be made at the system or sub-system level (perhaps based on individual modules) and could cover the entire development process or some stage (such as coding or testing).

Another use for software metrics is in classifying systems, tools, and developers. For example, developer remuneration may be based on some scale

of experience. Or different tools may be better suited for certain types (in terms of size and/or complexity) of projects. Perceptions of these characteristics may differ amongst project managers in some systematic manner with regard to organizational characteristics, making the creation of standards difficult.

Software metric models have not achieved the expected levels of performance when predicting development effort in real-world projects. That is not to say that software metrics have failed, but rather that their adoption has been more with larger organizations with well-defined development methodologies. Despite the problems of inadequate data collection, estimating input variables, and data contamination there is still a need for better project management techniques. The implications of even slightly better managed projects can be considerable, both in direct financial terms and in long-term organizational strategy.

The particular problem focused on in this paper concerns measuring system and process characteristics along with effort using fuzzy membership functions. Managers and developers can find it difficult, and also risky from both personal and organizational viewpoints, to assign what are in effect subjective numerical values for these measurements.

Attempts have been made to overcome the difficulties of quantitative metrics by using systems such as Function Point Analysis where a series of categories are used with levels of complexity and adjustment factors for the project [1]. Even here the number of functions needs to be precisely identified early in the management process, and they need to be individually assessed.

In addition, an unsolved problem is how to calibrate these models for different environments. Function Point Analysis requires extensive, and expensive, training and is notoriously subjective in any case. What the software engineering community really needs is a more accessible and realistic means of assessing systems and their development characteristics. Ideally such a system would allow for increasing the level of precision in inputs into the models, as more detailed information becomes available.

The outputs from such effort models developed using conventional methods are generally in precise numerical terms, and these can create an unhealthy and unrealistic adherence to estimates made early in the development life-cycle. Here a worthy goal would be to represent the expected effort in a sufficiently vague manner early in the development process to prevent these problems, but with the capability to refine the estimate progressively as development is completed and more accurate plans are required.

The idea behind using fuzzy variables in software metric models is that fuzzy logic allows for qualitative estimates of inputs and also qualitative outputs for the effort estimate. These can later be refined using fuzzy-numbers, and eventually become numerical values if this is desirable (and possible for that matter). All of these levels of detail can be obtained from a single system of membership functions and rules making the process more consistent and efficient over the projects' life cycles. There is no need to have separate models for each level of precision in inputs and outputs, which introduces problems in terms of maintaining the models and in ensuring that the models behave consistently.

Fuzzy logic however introduces a number of new problems for the software metrician, including the question of the stability of managers' perceptions across organization types and sizes and the manner of elicitation for the rules and membership functions. Here in this paper the focus is on membership functions, which must be overcome before attention can be paid to rule elicitation.

2. OBTAINING MEMBERSHIP FUNCTIONS

One method of eliciting membership functions is that of polling [4, p. 258]. This involves interviewing a number of experts and asking them to categorize values as belonging to two or more labels. The values of the membership functions at each point are determined as the proportions of experts who used those labels to describe the point. This provides a simple method for deriving membership functions without demanding high levels of understanding of fuzzy logic from the project managers in this case. It is also an effective technique when faced with a large number of experts where reaching consensus would prove impractical. This also allows for weighting expert opinion based on the degree of their expertise.

While fuzzy logic is often argued, sometimes with considerable fervor, as being distinct from probability, it can also be argued that proportions are related to membership degrees. For example, a system that a typical manager would regard as medium to a degree of 0.8 and large to a degree of 0.2 may also be regarded as medium by 80% of all managers and large by 20% of all managers (with a suitable defined population of course). Managers from a small company with a handful of developers could reasonably be expected to differ from those of a

manager in a large multi-national software development company, so a 'base population' is necessary for meaningful discussion of the membership functions.

Other alternatives that involve managers drawing such membership functions require these managers to have an understanding of fuzzy logic that defeats, in our opinion, some of its ease of use as a modeling technique. Other expert-based methods include exemplification where memberships are ascribed to values at a set number of levels of belief [4, p. 257] and directly as membership functions [2, pp. 282–286]. These will not be considered here. Many other methods are available in the literature for deriving the membership function from the numerical data using statistical and machine-learning techniques [2, pp. 290–300], but these, in our opinion, again defeat the purpose of fuzzy logic as an intuitive method.

3. SURVEYS

The survey results presented here were obtained from two mail-based surveys of developers in New Zealand. The initial mailing list, and its additions and subtractions for the second survey, was constructed from lists of major New Zealand software developers, previous surveys, and lists of major New Zealand companies who could be thought to be likely to carry out their own software development.

The number of respondents were 38 and 34, although not all answered all questions. These surveys are therefore not claimed to be representative of New Zealand developers, nor is the sample size sufficient for any concrete conclusions to be drawn for the population of developers that did respond (even if it could be defined). However, the following analyses do provide some suggestive ideas that could be used in subsequent surveys of a similar nature as well as illustrating the process of deriving and critiquing membership functions.

The first survey involved getting 38 project managers to indicate the range within which three labels, being small, medium, and large, were most appropriate for three variables; namely, data model size (as measured by the number of entities in the Entity Relationship Diagram), module size (as measured by the number of distinct modules, defined as screens, reports, and processing modules), and developer experience (measured in years).

The second survey involved 34 managers indicating the most appropriate label from seven possible labels for pre-specified values of the four variables. With the greater number of labels, to produce results comparable to the first study, the labels were combined into three groups: being very low and low (combined to produce low), below average, average, and above average (combined to produce average), and high and very high (combined to produce high). The proportion of managers ascribing a label to a particular value is plotted as the membership function. This survey used the same three variables as the first with the addition of development effort (in person hours). There were 13 levels for expertise, 14 each for the two size measures, and 17 for the effort measure.

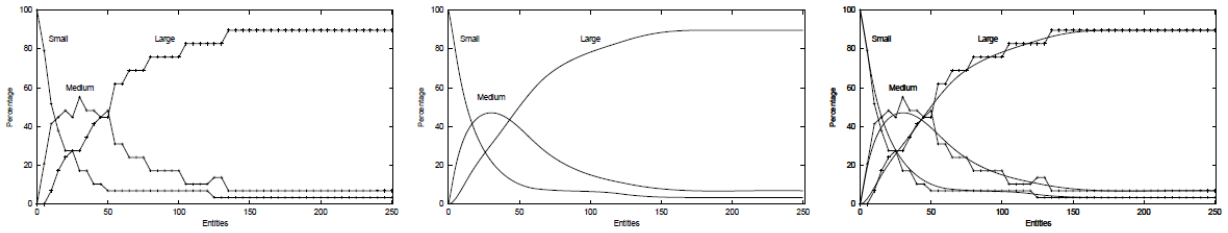
In the graphs below (Figures 1, 2, 3, and 4) the set of subgraphs for the first survey use equally spaced intervals, while the second set of subgraphs uses the predetermined points.

It should also be noted that some very unusual values were returned in the first survey that may reflect very unusual development practices or a lack of understanding as to what constituted an entity in an ERD or a program module, although these were explained in the survey. These have not been edited out since they presumably reflect at least some uncertainty in the membership functions. The results presented here are from the entire data set collected from the survey.

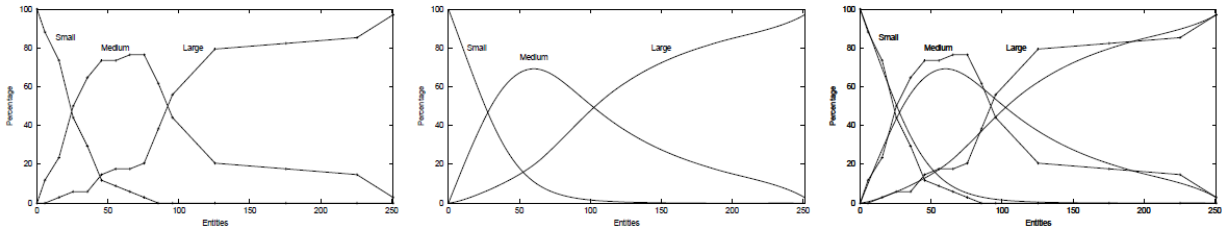
4. RESULTS AND DISCUSSION

The graphs of the raw membership functions, membership functions smoothed using Bezier curves, and combined membership functions (to show that the Bezier curves preserve the apparent meanings in the functions) are shown in Figures 1, 2, 3, and 4.

Despite the arbitrariness of the reduction in granularity from the second survey, there is remarkable good agreement with the experience functions. This is also the one that was best defined in the first survey, suggesting that the perception of developer experience is more consistent amongst developers. In the second survey the medium and large membership functions for the size variables has shifted to the right by about the same amount in each case, suggesting that the samples could perhaps differ in terms of organization size despite being from very similar mailing lists.

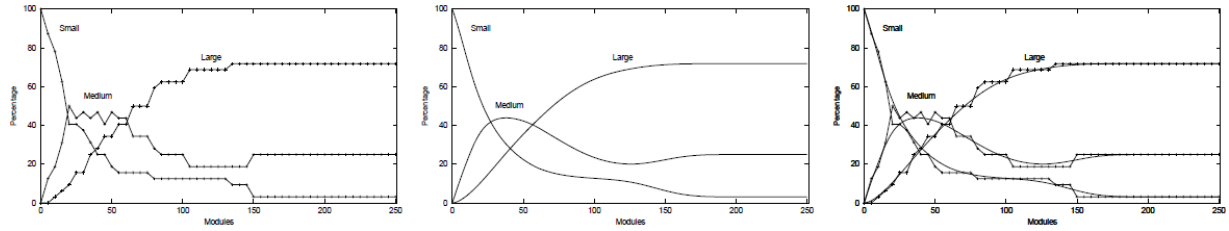


(a) First survey: Raw (b) First survey: Smoothed (c) First survey: Combined

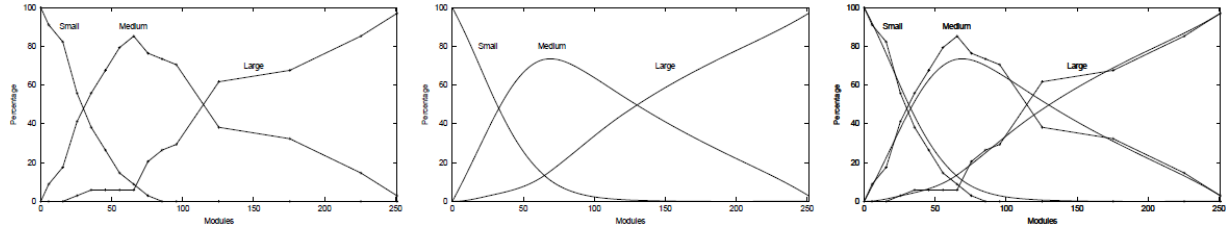


(d) Second Survey:Raw (e) Second Survey: Smoothed (f) Second Survey: Combined

Figure 1: Size of data model

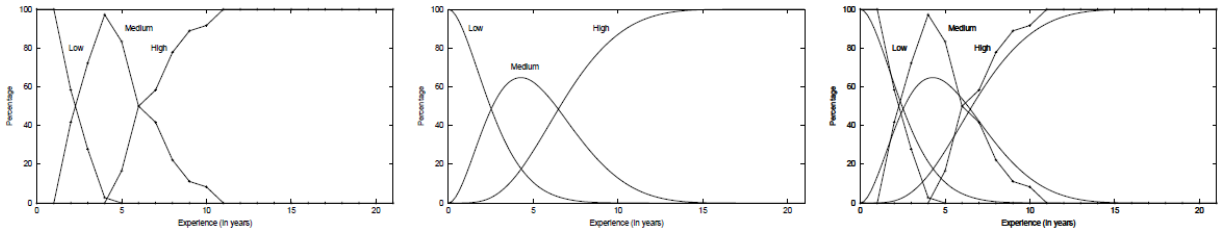


(a) First survey: Raw (b) First survey: Smoothed (c) First survey: Combined

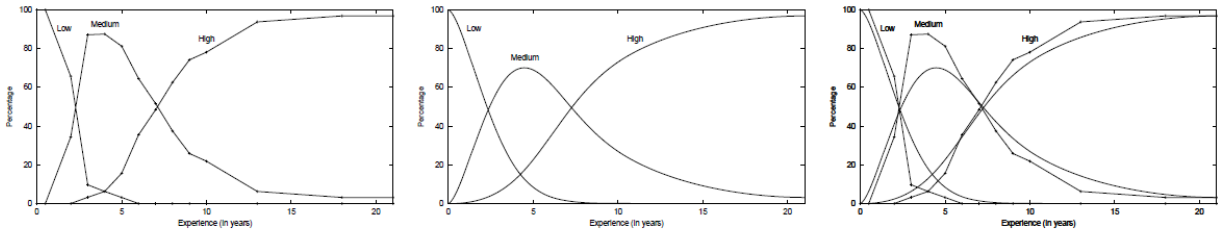


(d) Second Survey:Raw (e) Second Survey: Smoothed (f) Second Survey: Combined

Figure 2: Number of modules

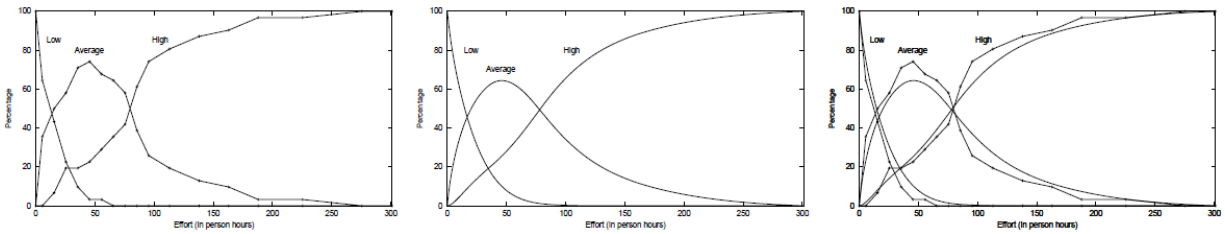


(a) First survey: Raw (b) First survey: Smoothed (c) First survey: Combined



(d) Second Survey:Raw (e) Second Survey: Smoothed (f) Second Survey: Combined

Figure 3: Developer experience



(a) Second Survey:Raw (b) Second Survey: Smoothed (c) Second Survey: Combined

Figure 4: Development effort

5. CONCLUSIONS FROM MEMBERSHIP FUNCTION SURVEYS

Using the 'classify pre-specified values' approach seems to work much better than 'give a range of values that fits the label'. Membership functions show more consistent definition under the first approach, especially for the middle functions. No difference were observed for developer experience though, which suggests that this is a much less variable concept for managers. The two surveys involved similar (in some cases the same) project managers so there is some evidence that the differences are due to the elicitation method rather than the vagaries of the samples.

Questions that need to be considered include the possibility that giving values leads to clearer classification, or it may simply limit choice. A new survey is being designed to test these two possibilities

It would also appear from the above analysis that the use of standardized fuzzy logic models for software metrics is likely to be hampered by the significant individual variation in perceptions of even three membership categories for size measures (although the three functions in the second survey are slightly artificial since they involved reducing the granularity of the actual results).

In fact, such standards would appear to be doomed unless they were restricted to a single organization with a reasonably homogeneous development process and managers with comparable perceptions. In many cases from the first survey the membership functions for the

medium categories are not even strictly convex and there is considerable evidence of disagreement in terms of size perceptions when expressed as ranges of values.

The most agreed upon measure was obviously that of developer experience. The membership functions derived here have a nice textbook look about them and would be easy to implement in a fuzzy logic system and subsequently use for inference.

While it is disappointing that the size based measures were so inconsistently viewed by the managers, this does not invalidate any of the managers using such membership functions themselves. Merely, it would appear unwise for them to share such functions or use labels for communication without ensuring that they shared common perceptions. Presumably each manager has a well-defined set of membership functions that could be used if they were able to understand the fundamentals of fuzzy logic well enough to draw these.

The next stage of this research project will be to examine the stability and consistency of rule extraction methods from project managers, given their particular set of membership functions and given predetermined membership functions. In order to extract such rules a variety of elicitation techniques will be examined in terms of their efficacy.

REFERENCES

[1] A. J. Albrecht and J. Gaffney. Software function, source lines of code and development effort prediction. *IEEE Transactions on Software Engineering*, 9(6):639–648, 1983.

[2] D. Dubois and H. Prade. Fuzzy Sets and Systems: Theory and Applications. Academic Press, London, 1980.

[3] N. E. Fenton and S. L. Pfleeger. *Software Metrics*: A *Rigorous & Practical Approach*. PWS, Cambridge, 1996.

[4] R. R. Yager and D. P. Filev. *Essentials of Fuzzy Modeling and Control*. Wiley, New York, NY, 1994.