Full citation: MacDonell, S.G., & Gray, A.R. (2005) The viability of fuzzy logic modeling in software development effort estimation: opinions and expectations of project managers, International Journal of Software Engineering and Knowledge Engineering 15(5), pp.893-918. http://dx.doi.org/10.1142/S0218194005002555

The viability of fuzzy logic modeling in software development effort estimation: opinions and expectations of project managers[1]

Stephen G. MacDonell
SERL, School of Computing and Mathematical Sciences, Auckland University of Technology,
Private Bag 92006, Auckland 1142, New Zealand
stephen.macdonell@aut.ac.nz (corresponding author)

Andrew R. Gray
Department of Preventive and Social Medicine
University of Otago, New Zealand

Abstract

There is a growing body of evidence to suggest that significant benefits may be gained from augmenting current approaches to software development effort estimation, and indeed other project management activities, with models developed using fuzzy logic and other soft computing methods. The tasks undertaken by project managers early in a development process would appear to be particularly amenable to such a strategy, particularly if fuzzy logic models are used in a complementary manner with other algorithmic approaches, thus providing a range of predictions as opposed to a single point value. As well as providing a more intuitively acceptable set of estimates, this would help to reduce or remove the unwarranted level of certainty associated with a point estimate. Furthermore, such an approach would enable organizations to 'store' their project management knowledge, making them less susceptible to employee resignations and the like. If fuzzy logic modeling is to be implemented in industry, however, managers must first believe it to be a realistic and workable option. This issue is addressed here by considering two related questions: one, what expectations do project managers have in relation to effort estimation? and two, what is their opinion of the methods that might be useful in this regard? This is followed by a discussion of the results of two surveys of project managers aimed at deriving membership functions using polling methods, the first using an interval declaration approach and the second using votes on fixed points. It is concluded that there is indeed support in the software engineering practitioner community for the use of methods based on the principles of fuzzy logic modeling.

1. INTRODUCTION

The vexed issue of accurate prediction of software development effort continues to challenge software engineering researchers. Several factors make this problem a very difficult one to solve – for instance, accurate and consistent data collection is problematic, meaning that records are often noisy or incomplete; a sound underlying cause and effect model of software development – one that takes into account all the variables and their impact – does not exist, but there is evidence to suggest that both quantitative and qualitative factors are influential in this regard; there are also conflicting demands in terms of what is needed from a predictive model – while accuracy is naturally a crucial requirement, model interpretability and the facility to incorporate existing knowledge are also desirable, as well as general applicability across a range of predictions. In considering the factors just described, it is no surprise that modeling methods falling into the *soft computing* domain have attracted increasing attention. After all, these methods are promoted as being especially appropriate for noisy or incomplete data, for relationships where there is significant uncertainty, or where the incorporation of knowledge as well as data is seen as

[1] An earlier version of part of this paper appeared in the *Proceedings of the ICONIP/ANZIIS/ANNES'99 International Workshop on Future Directions for Intelligent Systems and Information Sciences.*

important. On the surface, then, the match between need and solution would seem to be a good one.

What is less clear, however, is the degree to which these avenues are thought to be worth pursuing from the perspective of industry based project managers [41]. That is to say, most of the work on these modeling methods, including studies undertaken by the authors [13,14,33,34], has been carried out by researchers rather than practitioners, much of it without any definitive sense of the likely acceptability of such methods in an industrial software development setting [11]. For instance, while the alternative of using qualitative labeling for variables as utilised in fuzzy logic methods has been suggested in research (e.g. [18,53]), the stability and consistency of such labels in practice has not been established, and optimal elicitation techniques for deriving appropriate membership functions have not been determined or tested. When it is also considered that informal estimation methods based on experience, expert opinion or personally moderated analogies remain commonly used [6], it would seem sensible to consider the issue of industry acceptability before advancing too much further. (Note: this is not to say that *all* research should be determined purely on the basis of existing practitioner needs, but rather that those involved in practice-oriented research should be sufficiently well informed to have a sense of the likely relevance and applicability of their work in industry.) This paper therefore addresses three related issues – what expectations do project managers have in terms of effort estimation methods; what attitudes do project managers have in regard to existing and proposed methods; and how effectively and consistently can project managers specify ranges of values for a selection of relevant variables, or consistent labels for a selection of values, such that might be used in a fuzzy prediction system.

1.1 Project management challenges

As in other disciplines the management of software development projects makes significant use of measures to characterize the processes employed, the resources utilised and the artifacts consumed in or produced by those processes. Models are built and calibrated using these measures to assist the project manager to classify, predict, monitor, control or evaluate aspects of their development projects. One of the most common uses of such models in software engineering has been in the prediction of software development effort. For the developer, manager and user of any software product, project effort estimation is a very important activity. The estimate is frequently an input to contract negotiations, resource and personnel costing and charging for the project (although there is some very sound rationale for treating cost and price separately, particularly at the bidding stage of a competitive contract [23]). It provides a basis for the project manager to plan and control subsequent development activities. It may have run-on effects for the user, in that their operations may be planned around the delivery of a software product. Clearly then an accurate and robust estimation model is desirable from all perspectives. The inputs to such models of development effort commonly include some measure of system size, system complexity, the development tools employed and developer experience (among many other factors). The output can be any measure of effort, such as person-days, and may be specified at the system, sub-system or module level.

A common aim of effort prediction research is to build and validate models that deliver estimates within 25% of the actual effort at least 75% of the time [8]. Unfortunately, even this modest goal is seldom achieved in a holdout sample (that is, on data that was not used in model parameter calibration). Obtaining such a model is difficult for a number of reasons. If there existed a complete underlying theoretical foundation for software development then incorporating all the influential factors and their impact in a model to predict development effort would be a challenging but ultimately achievable goal. This is, however, clearly not the case – there is much about the software development process that remains 'uncontrollable', from the highly variable influence of project personnel to the unknown impact of new technologies. Under these circumstances obtaining a comprehensive, widely applicable and highly accurate predictive model of software development effort would appear to be highly unlikely. In such circumstances one might expect that locally calibrated useful, if unspectacular, models would be developed, accepted and used by all involved with an awareness of their inherent limitations. However, in many respects the contrary occurred – the desire for high precision in what are clearly acknowledged as estimates created an almost fanatical search for the ultimate estimation method. In the last five years, however, it does appear that there has been a change in attitude, and a greater acceptance of the notion that estimates are just that – values that have a degree of uncertainty associated with them. Indeed, given the lack of a formal model of the software process, the drive for a single one-fits-all model is fundamentally flawed and should be tempered by more pragmatic aims.

Underlying this challenge is an inherent trade-off in the development and use of effort estimation models – is it better to be approximately correct most of the time or precisely inaccurate all of the time? It is an acceptance of this trade-off that provides some of the rationale for introducing fuzzy sets and fuzzy logic into the estimation process. In extreme cases, managers are expected to predict to within a person-day the effort requirements for a project about which they know relatively little, perhaps at the preliminary specification stage. Their estimate is

likely to be based on best-guess figures for, among other things, system size and developer productivity, both of which are subject to significant variation. Moreover, if the project is open to competitive tender, significant pressure may fall on the manager to provide an even earlier estimate, so that a bid may be submitted in order to win the project contract. Unfortunately, these estimates tend to be set in stone and they take on significant meaning in the context of the client's expenditure and their subsequent operational planning. It is little wonder, then, that the software industry is frequently criticized by its clients for recurring cost and schedule over-runs – the fact is that in at least some cases the estimates provided were simply unrealistic. Use of estimation methods that take uncertainty into account and produce similarly qualified estimates has the potential to partially alleviate some of these difficulties.

1.2 Fuzzy logic for software development project management

In response to the problems and challenges just described, fuzzy logic has been proposed as a potential solution. This is founded on the fact that fuzzy logic allows for qualitative specification of input values and can generate qualitative outputs for the effort estimate using a stored fuzzy rule base. Furthermore, linguistic outputs can later be replaced by ranges, fuzzy numbers or crisp numeric values as more accurate input information becomes available. In other words, the precision of the input data can suitably reflect the amount of knowledge that is available at the time the estimate is made and the precision of the output(s) can usefully reflect and reinforce the appropriate degree of certainty in planning. For instance, initially the size of a system may be said to be "above average", later it becomes "about 100" components, and later still it is exactly 105 components. In terms of the model outputs, initially the project may be expected to require a "very high" amount of effort, later "about 300 developer months", and finally 360 developer months. All of these levels of detail can be managed in a single fuzzy system of membership functions and rules, making the process consistent and efficient over the life of a project. The use of such an approach is particularly suited to software measurement models early in the software process before even approximate counts of functions and their complexities are available.

In recent years, then, research investigating the application of fuzzy logic methods in software project management has begun to gain some momentum [16]. Kumar *et al*. [25] provide an excellent introduction to the potential of fuzzy logic and neural network modeling in the software engineering domain. A rule-based expert system to assist personnel in software project risk management is described by Madachy [35]. The system in question used a taxonomy of crisp set risk factors derived from the well-known COCOMO research projects [5,56]. Relationships between these factors were formed into rule sets based on knowledge and heuristics elicited from domain experts (emphasizing the need for methods to capture and codify personal expertise and local organizational knowledge, as well as more generically understood relationships). Similar use of the COCOMO factors in effort estimation, but in this case using fuzzy sets and rules, has been extensively reported [18,19,36,17,46,55]. Similar fuzzy extensions to function point analysis (FPA) [2] have also been proposed (and in some cases evaluated using artificial or industry data) in order to enable managers to avoid the crisp evaluations of function scale and complexity [29,44,46]. Idri and Abran further suggest that fuzzy logic could also be employed to describe the degree of similarity between projects or project attributes, a characteristic that would be especially useful in the growing field of research and practice in analogy-based prediction [18,20].

Rather than adapting and augmenting existing project management methods such as COCOMO and FPA, other research has resulted in the development of independent fuzzy systems, complete with fuzzy sets, rule bases and inference engines, in order to provide robust predictions of project size and effort [1,32,59]. The challenge of effective project scheduling has also been addressed by research employing fuzzy sets to represent uncertainty in start and finish times, activity durations, resource availability and resource capabilities [28,16,39,40,48]. This has been extended to then enable more robust risk identification i.e. noting activities that are subject to high uncertainty [15,31].

Other methods that fall within the wider domain of soft computing have also received attention in the last ten years. In particular, the evaluation of machine learning techniques in software engineering applications has been extensive. Multi-layer perceptron (MLP) neural networks trained using back propagation have been among the methods most commonly assessed (e.g. see [12,27]) although other structures, including radial basis functions [52] and Albus perceptrons [47], have also been considered, as have fuzzy neural networks [45]. In some cases these methods have indeed enabled the construction of accurate models (e.g. [22,27,58]). These methods are also relatively 'data-hungry', however, a requirement that cannot always be easily satisfied in software engineering practice. Furthermore, such methods do not lend themselves to being readily understood by project managers. While methods do exist to enable meaning or 'knowledge' to be extracted from neural networks, they are still generally perceived to be black box in nature. This has a detrimental effect on their acceptability in industry [21]. A sense of belief and trust in any model is a necessary precursor to its effective use

– a lack of transparency in neural network models does little to encourage such trust.

In many respects this need for trust and model credibility has led directly to increased activity concerning estimation on the basis of analogous observations (e.g. see [3,51,54,57]). Such methods reinforce the perception that local project histories are indeed valuable in this regard, particularly where their use is augmented by expert knowledge [54]. Both Myrtveit and Stensrud [37] and Walkerden and Jeffery [57] found that a combination of expert judgment and estimation tools (in their cases regression analysis and analogy-based prediction) proved to be the most effective approach, although the project managers' *perceptions* of tool performance tended to undervalue the contribution of the tools. Sauter [49, page 109] similarly supports the notion of decision support systems that "…blend analytical tools with intuitive heuristics to improve managers' insights about factors too complex to build into models", particularly under circumstances in which the data required for fully informed analysis is simply not available. While Lederer and Prasad [26] found evidence to suggest that estimation accuracy was more likely to be improved if guessing, intuition and personal memory were *not* used, they also found that the use of algorithmic modeling methods did not facilitate more accurate estimates. Again a combination of methods may instead prove to be the optimal approach [42]. Moreover, accuracy is just one (albeit important) characteristic of a useful estimation method – interpretability of models, for instance, is also a beneficial and therefore desirable trait.

A significant number of related papers have also described the various difficulties encountered by researchers and practitioners in software project management modeling. For instance, Ohlsson *et al.* [38] report the results of an experiment involving the iterative development of an experience base to enhance effort estimation over time, reinforcing the need for ongoing model refinement. Shepperd and Cartwright [50] investigate the use of analytic hierarchy processing (AHP) as a means of prediction given only a minimal amount of data, a situation that is particularly prevalent in smaller organizations [10]. Both discussions emphasize the value and importance of expert judgment in the estimation process. The need for managers and clients to accept and deal with the inherent uncertainty in software development projects is emphasized by, among others, Fenton and Neil [11], Kitchenham [23], Lin and Chen [30] and Putnam and Myers [43]. The latter authors suggest that such an approach would naturally lead to the provision of a range estimate (a strategy also recommended by Peeters and Dewey [42]), rather than a single point value, although they acknowledge that this may not be acceptable to higher-level management. In this case they recommend the use of probabilities, or the acceptance of a project portfolio approach (also supported by Kitchenham [23] and Bilalis *et al.* 2002 [4]), whereby a loss on one project is accepted on the understanding that there is likely to be a corresponding gain on another. In any case there is a need for all involved to moderate their expectations regarding the accuracy of any models developed.

These related discussions provide further rationale for the use of methods such as fuzzy logic in effort estimation and similar activities, although such methods are not mentioned specifically. They also reflect more generally the continued desire of researchers to enhance the performance of effort estimation activities. The *potential* of soft computing seems evident, but it remains to be seen whether such approaches are applicable or acceptable in industry settings [7], in part motivating the work presented here.

2. ATTITUDES TO EFFORT ESTIMATION

The improvement of software development effort estimation has thus been a long-time goal of software engineering researchers, driven by the apparent needs of industry project managers. Published evidence in support of these needs is not extensive, however. In fact, some authors have found that in spite of significant research efforts in the last twenty years project managers continue to both prefer and use *ad hoc* estimation methods based on personal experience, expert opinion and local analogies [6,42,57]. It is possible to speculate on why this might be – perhaps it is a matter of managers preferring to use the 'tried and true' over the innovative; there may be an assumption that in fact few advances have been made; or there may be a belief that such methods require excessive amounts of data to be effective. In order to directly address this scarcity of explicit information on project managers' needs and expectations, this work utilized a range of data collection exercises incorporating interviews, document analyses and the application of structured questionnaires, as follows.

2.1 Interviews and document analyses

The first phase of this research was a series of semi-structured interviews with project managers from five groups responsible for software development, to gain high-level insights into their requirements in terms of support for project effort estimation. The excerpts presented in this section are taken directly from those interviews and from the examination of the same organizations' internal documents. Two of the organizations are based in New Zealand, three in the United Kingdom. Three are software development organizations, one is a banking and insurance institution and the other is a national utility provider. The organizations were chosen to cover a range of sectors,

organization types and sizes. They had previously indicated an interest in effort estimation issues in response to a general mail-out request for participation. In this respect they self-selected with a declared awareness of the subject. Given the exploratory nature of this phase of the study, however, their responses can and should be treated as illustrative rather than definitive. (Names of both the individuals and their organizations are not given here for the obvious reason that many are quite critical of their own practices. An indication of the source of each comment/quote [Organization 1-5] is provided. The candor shown by the project personnel was appreciated.)

The interviews were all conducted in person by one of the authors with between one and three project managers from each organization. In each case, six topic areas were addressed: expectations regarding estimation, requirements for an effective estimation method, the role and impact of personal experience, the need for early estimation, availability of historical data, and the use of contingency in estimation. The interviews were semi-structured in the sense that the same topics were traversed – there were no pre-determined sets of questions for each, however. Rather, the interviews took the form of a free-ranging interactive discussion over the topics at hand. While this could be considered as compromising the interview process, on the positive side such an approach enabled a richer picture to be formed regarding attitudes and practices with respect to estimation.

What follows is a selection of indicative comments relating to each topic. As the interviews were discussion oriented, and formed only the initial part of this study, it is neither feasible nor appropriate to present the managers' views in a more structured manner. The comments shown are typical of those made. It should be acknowledged that the information is incomplete and anecdotal. In terms of the overall study, however, it served to highlight the issues that needed to be followed up in the subsequent stages of the work.

2.1.1 Expectations regarding estimates

"The risks inherent in the project, and the uncertainties in the estimate, should be clearly identified to provide assurance to the customer, especially about delivery dates and cost." [O2]

"Although software project estimation will never be an exact science, it can be transformed from a black art into a series of systematic steps that provide estimates with an acceptable degree of risk." [O3]

"An estimate is not an exact measure, but an approximation based on judgment. In the computing world this is usually bundled up in the term 'experience'. Is experience enough? We still get it wrong. We under estimate constantly. Maybe not so much on the small projects, but quite significantly on the larger ones." [O4]

These comments suggest that project managers are acutely aware of the uncertainty inherent in the estimation process. As a result it appears that at least some would prefer to see estimates provided with an associated confidence factor, along with an assessment of the source and level of risk associated with the likelihood of an estimate being incorrect.

2.1.2 Requirements for estimation methods

"[An effective estimation method needs to be] simple, thorough, realistic, inexpensive... practical, cost-effective, and versatile. It should not add to overheads." [O2]

"The current methodology... is far too complex and difficult to use. This is demonstrated by the fact that it is not used." [O4]

Although many estimation models exist, their complexity in use is inhibiting their adoption, at least as reflected in these quotes. An ideal modeling method should strike a balance between effectiveness, in terms of accuracy and consistency, and simplicity in calculation and application.

2.1.3 The role of personal experience in estimation

"Each person undertaking estimation tends to utilize their own heuristic approach." [O1]

"The Project Manager's own experience and other Project Managers' experience either within [the company] or in other organizations can provide a valuable range of reasonableness for a given development." [O3]

"The company's next project takes on board very few of the lessons that we should be learning from the past. The lessons are generally only transferred if the players are the same. That is, we are only learning as individuals not as a company." [O4]

"Current estimates are based on personal experience. Everyone's experience is different and thus different people produce different estimates for the same function. People are inclined to introduce their own expertise into the estimation process." [O4]

"A great deal of experience is required to produce accurate and reliable estimates… Software estimating tools cannot replace real experience in the real world of project work, but they can supplement such experience." [O5]

These remarks indicate the degree of influence that individual project managers have on the estimation process and its outcomes. While it could be said that individual influence should be removed entirely this is unlikely to be feasible in practice. Moreover, this may result in a significant loss of local knowledge that might be crucial to the production of useful estimates. A modeling and estimation method that sensibly *incorporates* the views of experienced personnel would therefore seem to be a potentially more useful approach.

2.1.4 Early estimation

"A 'simple' form of Function Point Analysis may be used to size development projects in their early stages… and may be used as a basis for producing project estimates." [O2]

"Quite often we do not know enough about the project to be confident in the estimates that we produce. This is consistent with the more you know and the better it is defined, the greater the accuracy of the estimation. However, we are usually asked to estimate design and development together, which is obviously a more risky situation." [O4]

"People tend to remember these earliest estimates of the cost or duration of a project, so it is important that these estimates should err on the side of caution, and be properly qualified." [O5]

Early estimates are demanded by clients, irrespective of the fact that a project manager may have very little information at that time on which to base their predictions. Modeling methods that therefore allow early 'ball park' estimates to be developed, but which deliver them to both manager and client with appropriate qualification, would appear to be an acceptable compromise.

2.1.5 Data availability

"Many firms do not have adequate records about the performance of the project team on similar projects." [O1]

"We operate solely on guess work and rules of thumb, not on any empirical data." [O4]

"We do not keep an adequate history of how well actuals meet estimates. This means that we lack indications of where our estimation is good; where it is bad; and what areas we did not estimate at all." [O5]

To some extent a lack of historical data could reflect a low level of maturity in an organization's software process. That said, it is imperative that even under such circumstances an organization is able to produce reasonably accurate estimates of effort for potential clients. Modeling methods that do not necessarily *require* large stores of data, but that can dynamically adapt as data becomes available, would be useful in this respect.

2.1.6 Estimation contingencies

"We use a standard formula to come up with our estimates and then we apply a 'fudge factor' as a contingency, based on our own experiences." [O1]

"Early in the project the effort contingency will be large due to the uncertainty of the detailed requirements, system design, likely problem areas etc. As the project progresses, the contingency should be reduced." [O3]

"Software estimates are frequently based on guesswork rather than fact." [O5]

The comments above would suggest that estimation methods that are not so 'locked down' as to prevent the incorporation of sensible contingencies based on organizational knowledge would be preferred. With this latter point in mind, some means of storing that knowledge needs to be implemented.

The excerpts above provide some valuable insights into the views and desires of project managers in relation to effort estimation. On the basis of their remarks it seems clear that managers would value modeling and estimation methods that addressed these desires. The characteristics of fuzzy logic modeling – being able to incorporate both data and expert knowledge, robustness, variable granularity in inputs and outputs, and transparency through the intuitive nature of fuzzy rules – make it a potentially effective approach in this domain.

2.2 Questionnaires

Having established some high-level anecdotal evidence supporting the need for alternative or complementary modeling methods, the study then considered the state of practice. This section reports the results of two postal surveys of project managers carried out in New Zealand software organizations. The focus of the surveys was on contemporary practices in software development and acquisition, addressing in particular any model-building methods used in effort estimation.

The first, one-page survey was distributed as part of a much larger data collection exercise investigating systems development and acquisition practices. The total sample size was just under 850 organizations – only 160 of these, however, were understood to conduct software *development* as opposed to *acquisition* through outsourcing and/or integration services. Of the 160, thirty-eight provided usable responses. Seventeen of these indicated the use of measurement/metrics in their project management activities, so the following information pertains to this set of respondents. These were all small organizations – most employed between 1 and 10 full-time equivalent (FTE) development personnel (see Table 1), an expected profile given the relative dominance of small organizations in New Zealand. The majority of these organizations (13 of 17, or 77%) used expert opinion to produce estimates of development effort, significantly more prevalent than any other method (reported in Table 2).

FTEs	Frequency	Percentage
1-10	10	59%
11-50	4	23%
51+	3	18%
Total	17	100%

Table 1. First survey: breakdown of organization size in terms of full-time equivalent development personnel

Method used	Frequency	Percentage
Expert opinion	13	77%
Function point analysis	5	29%
COCOMO	1	6%
Regression-based models	0	0%
Other techniques	3	18%

Table 2. First survey: breakdown of effort estimation methods used

A second more detailed survey was developed and sent out six months later to a smaller sample (330 organizations from a range of industry sectors, including 150 sites understood to undertake software development) to more fully address issues related to effort estimation. Forty-four usable responses were received. The distribution of organization size is shown in Table 3. While the organizations are again typically small, a wider range of sizes was represented in the respondents to this survey. Table 4 shows the nature of business of the forty-four organizations.

FTEs	Frequency	Percentage
0-4	10	23%
5-9	4	9%
10-14	5	11%
15-24	5	11%
25-49	7	16%
50+	5	11%
Not given	8	18%
Total	44	~100%

Table 3. Second survey: breakdown of organization size in terms of full-time equivalent development personnel

Organization type	Frequency	Percentage
Commercial organization	18	41%
Government	4	9%
Commercial software house	14	32%
Not given	8	18%
Total	44	100%

Table 4. Second survey: breakdown of respondents by type of organization

Thirty-six of the forty-four organizations replied to the questions concerning development effort prediction, with thirty-two of these stating that they employed one or more methods to estimate effort. Of the four who said that they did not predict effort, one commented that they "…did not know how to go about making such predictions", two said that there was no need for such predictions, and one remarked that their development environment was too unstable to allow such management. In terms of the modeling methods actually employed by those respondents who performed estimation, expert opinion again totally dominated those others considered (as shown in Table 5). Given that this sample included a significant proportion of software houses the relative lack of infiltration of the function point analysis (FPA) [2] and COCOMO [5] techniques was unexpected.

Technique	Always	Mostly	Occasional	Never	Not heard of	Missing
Expert opinion	22	9	2	0	1	10
Function point analysis	1	2	14	10	4	13
COCOMO	0	0	3	3	18	20
Regression-based models	2	5	12	5	5	15
Other techniques (SLIM)	0	0	1	0	0	-

Table 5. Second survey: breakdown of effort estimation methods used

The next question was concerned with the stage at which estimates were actually produced – as expected the need for predictive capability at the analysis phase is most evident (see Table 6). Estimation at the development task level for individual components was also undertaken by most of the respondents (see Table 7). The importance of estimation during the early stages of the process further supports the use of qualitative estimation techniques, since it is often difficult to accurately estimate some of the factors used in effort estimation models at these stages.

Stage of process	Frequency	Percentage
Analysis	30	94%
Design	19	59%
Prototyping	11	34%
Programming	11	34%
Testing	4	13%
Maintenance	6	19%

Table 6. Second survey: stages of process in which effort estimates are made

Level at which predictions are made	Frequency	Percentage
System	8	25%
Component	10	31%
Task	14	44%
Both task and component combined	10	31%

Table 7. Second survey: levels of prediction for organizations performing effort estimation

The next section of the survey investigated problems or difficulties encountered by the organizations in using the

specified estimation methods. (Note that only expert opinion, function point analysis and regression-based methods were considered as they each had a reasonable number of users able to comment on their effectiveness.) Problems cited in relation to expert opinion are shown in Table 8. It is interesting to note that, even though it is by far the most widely used method (Table 5), nineteen of the thirty-two respondents acknowledged that the technique was not sufficiently accurate. While this is clearly the view held by the respondents, it could reflect the assertion that expectations of accuracy are indeed very high. Other issues revealed in response to this question included a lack of appropriate experts in one organization, leading to an assertion that the technique should not be used as the sole or final approach. Another respondent noted the problems caused by inconsistencies in the opinions given by various staff. One respondent suggested that this method was reasonably accurate for small to medium sized projects, which for them was up to three months duration, but was less effective for larger projects.

The degree to which organisations had experienced various difficulties in using function point analysis is reported in Table 9. Relatively speaking, accuracy seemed to be less of a problem when compared to other difficulties. In particular, it appears that there is too much certainty required in the input information, and that this information is not available at the stage when the predictions need to be provided. An additional problem cited in regard to FPA was that productivity rates fluctuated too much from project to project to enable consistent predictions to be produced. Another respondent mentioned that there was resistance to FPA from personnel given its "seemingly weird basis". One of the respondents who checked the box to indicate that FPA was too complex felt sufficiently strongly about this to use five ticks! This may be interpreted as supporting the need for simple-to-understand modeling techniques in this domain.

Problem	Frequency
Technique is not accurate enough	19
Do not collect necessary information accurately enough	12
Need predictions too early in development	13
Information is too costly or difficult to collect	1
Data requirements for calibration cannot be met	4
Technique is too complex	0

Table 8. Second survey: problems with expert opinion as an estimation method

Most prominent among the difficulties cited in regard to regression-based modeling was the need for precision in the information collected, and to a lesser extent the costs incurred or obstacles encountered in the collection process. It may be that this method is used most often to provide an alternative estimate to compare against that obtained through expert opinion or function point analysis (given that it tends to be used only 'occasionally' (Table 5)).

Problem	Frequency
Technique is not accurate enough	3
Do not collect necessary information accurately enough	7
Need predictions too early in development	9
Information is too costly or difficult to collect	4
Data requirements for calibration cannot be met	2
Technique is too complex	4

Table 9. Second survey: problems with function point analysis as an estimation method

Problem	Frequency
Technique is not accurate enough	1
Do not collect necessary information accurately enough	6
Need predictions too early in development	1
Information is too costly or difficult to collect	3
Data requirements for calibration cannot be met	0
Technique is too complex	2

Table 10. Second survey: problems with regression-based models as an estimation method

Project managers were then asked to indicate the degree to which they believed that certain factors ought to be included in a predictive model for development effort, reported in Table 11. (Note that these factors were specified in the survey – others suggested by the managers themselves are mentioned below.)

Measurement	Not important	Could be included	Should be included	Must be included
System size	0	4	6	25
System complexity	0	2	7	26
Developer experience	2	3	18	12
Developer tools	1	6	11	17
Developers' methodology	1	7	16	11
Team size	1	6	15	11

Table 11. Second survey: variables to be considered in a predictive model of effort

Several other factors were suggested by the project managers as being important in such models, ranging from the number of tables in the system and data conversion requirements through to the degree of interaction between developers and users and specific system performance needs. The most frequently mentioned factor, however, was the existence of a user-imposed timeline – this was suggested as being an essential component in a predictive model by six different project managers.

3. ATTITUDES TO FUZZY LOGIC MODELING

The information presented in the previous sections provides indirect support for the view that fuzzy logic modeling would have appeal to at least some project managers. The final component of the second survey addressed this issue directly, in order to assess the possible acceptability of fuzzy logic modeling in effort estimation (see also [34]). Thirty-one of the thirty-six respondents actively involved in managing development projects had heard of fuzzy logic (see Table 12). Eleven stated that they were interested in using fuzzy logic techniques, twenty-three stated that they would need to know more about the technique before making a decision, and two considered that fuzzy logic techniques would not be useful to them (see Table 13). Whilst the high awareness of fuzzy logic was not entirely unexpected, given that it has received a reasonable amount of coverage in the popular press in recent years, finding eleven organizations overtly interested in using fuzzy logic modeling was not anticipated, and is an encouraging result in terms of the likely acceptability of such an approach in industry.

Response	Number	Percentage
Heard of fuzzy logic	31	86%
Have used fuzzy logic	2	6%
Using fuzzy logic	1	3%

Table 12. Second survey: awareness of fuzzy logic

Response	Number	Percentage
Interested in using fuzzy logic	11	31%
Interested, but need to know more before using fuzzy logic	23	64%
Not interested in using fuzzy logic	2	5%

Table 13. Second survey: interest in using fuzzy logic in effort estimation

Some of the advantages of fuzzy logic in regard to development effort estimation have already been described. Three of these advantages – the opportunity to use expert knowledge in model development, the possibility of using linguistic (non-numeric) inputs, and the ability to generate linguistic or range outputs – were put to survey respondents, with a request that they indicate their degree of interest in each (reported in Table 14). Those respondents expressing some interest in using fuzzy logic (in answer to the previous question) found each of the three advantages equally appealing. Of the three, that receiving strongest support was the opportunity to produce linguistic or range outputs rather than a crisp point value. These results provide direct support for the comments reported above suggesting that managers are indeed aware that they often cannot (or at least should not) realistically provide exact numbers as inputs to models, and do not necessarily want to get exact numbers back out either, given the uncertainty inherent in their measurement and estimation processes.

Response	Number	Percentage
Using expert knowledge for model development	19	56%
Linguistic inputs in place of numeric values	19	56%
Linguistic outputs in place of numeric values	21	62%

Table 14. Second survey: interest in perceived advantages of fuzzy logic

Overall these results suggest that the challenges faced by project managers in using existing modeling techniques could be partially alleviated through the use of fuzzy logic modeling. Given the apparently high level of interest in the method, one might ask why there is such a low level of actual use. This disparity could be due to a low level of fuzzy logic software availability for non-specialists, a lack of accessible guidelines (function point analysis, by comparison, has an internationally developed education and certification framework), and an absence of published successful case studies (something of a 'catch-22' situation). Alternatively there may be uncertainty regarding more fundamental issues, such as the likely stability of managers' perceptions in relation to fuzzy membership or effective means of elicitation for rules and membership functions. This latter issue is now addressed, as the determination of appropriate fuzzy sets is clearly a necessary precursor to building useful fuzzy logic estimation models.

4. DERIVING MEMBERSHIP FUNCTIONS

The work described in this section is principally concerned with the task of determining appropriate membership sets for software development effort estimation. In order to further assess the feasibility of fuzzy logic modeling in this domain the two surveys discussed above were also used to examine project

managers' perceptions of three potentially relevant variables – data model size, functionality size and developer experience.

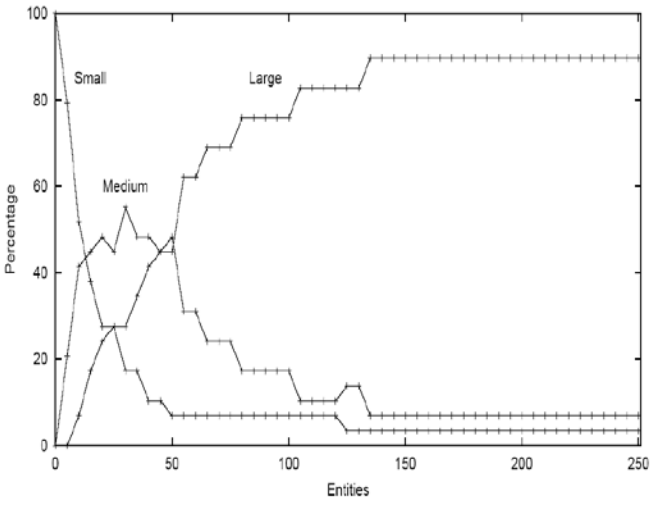
One method of eliciting membership functions is that of polling, as discussed by Yager and Filev [60]. This involves interviewing a number of experts and asking them to categorize values as belonging to two or more labels. The values of the membership functions at each point are determined from the proportion of experts who use 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 domain experts – software project managers in this case. It is also an effective technique when faced with a large number of experts and reaching consensus is unlikely or impractical. Furthermore, it enables opinions to be weighted based on the level of expertise of the manager providing the data. Other alternatives to membership function determination that involve managers drawing the functions require those managers to have an understanding of fuzzy logic that to some extent defeats its ease of use as a modeling technique. Further expert-based methods include exemplification, where memberships are ascribed to values at a set number of levels of belief [60] or directly as membership functions [9]. Many methods are available for deriving membership functions from numerical data using statistical or machine-learning techniques [9,24,60], but these obviously assume the existence of sufficiently large volumes of data, something that is not always the case in software engineering practice. In this study the use of such data-driven approaches was not feasible. The surveys therefore used two different polling methods, the first employing an interval declaration approach and the second using votes on fixed points.

The first survey elicited the views of 38 project managers, with each indicating the range within which three labels, small, medium, and large, were most appropriate for three variables, namely data model size (as measured by the number of entities in an entity relationship diagram), functionality size (as measured by the number of distinct system modules, defined as screens, reports, and batch processing modules), and developer experience (measured in years). The second survey involved 34 managers indicating the most appropriate label from a range of three (low, medium and high) for pre-specified values of the same three variables. The proportion of managers ascribing a label to a particular value is plotted as the membership function. In the graphs below (Figures 1 to 3) the set of sub-graphs for the first survey were determined using equally spaced intervals, while the second set of sub-graphs resulted from the use of predetermined points. Each figure includes graphs of the raw membership functions along with the same membership functions smoothed using Bezier curves to produce typically more continuous representations. It can be noted that some unusual values were returned in the first survey that, if correct, would reflect very unusual development practices or a lack of understanding as to what constituted an entity or a system module, even though these were explained in the survey. These observations have not been edited out since they presumably reflect at least some uncertainty in the membership functions. The results presented here are therefore from the entire data sets collected from both surveys.

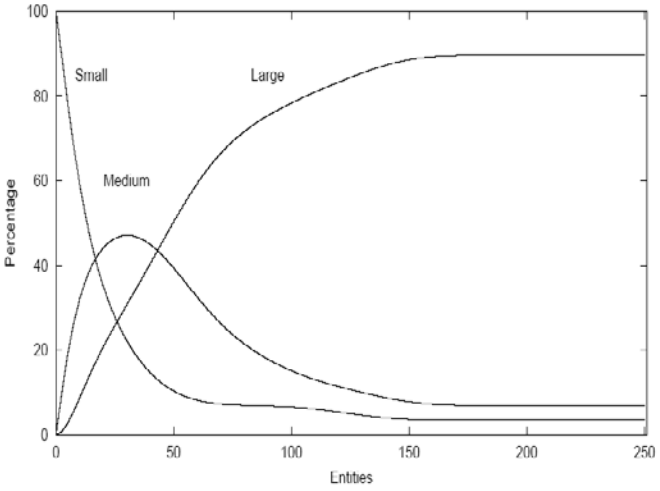
Figure 1 provides depictions of the membership functions for data model size for the labels small, medium and large. While some insights can be drawn from these graphs, the results are confounded by the inclusion of some surprisingly high values in the category of small data model size in the first survey and medium size in the second, resulting in long right-hand tails. (In fact, this is a common occurrence for all three factors in both surveys with the exception of developer experience in the first survey.) Generally, however, the relative shapes of the membership functions are as expected, with, for instance, low numbers of entities being labeled as small to a high degree.

Functionality size was measured in this study using a count of the number of screen and report modules in a system. In examining graphs (a) and (b) of Figure 2 it is clear that range-based classification of the number of modules between approximately 30 and 60 is problematic, with the labels of small, medium and large all being applicable to a degree of between 0.15 and 0.5. The membership functions obtained for the same factor in the second survey, which employed the strategy of point-labeling, are of the form more commonly expected.

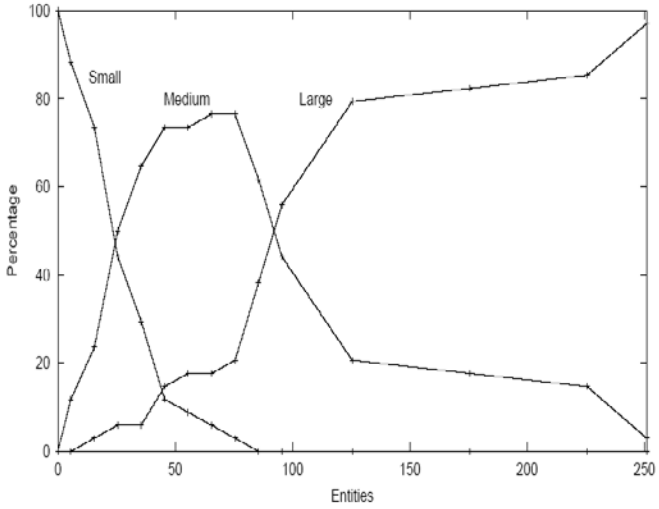
Fig. 1. Size of data model.



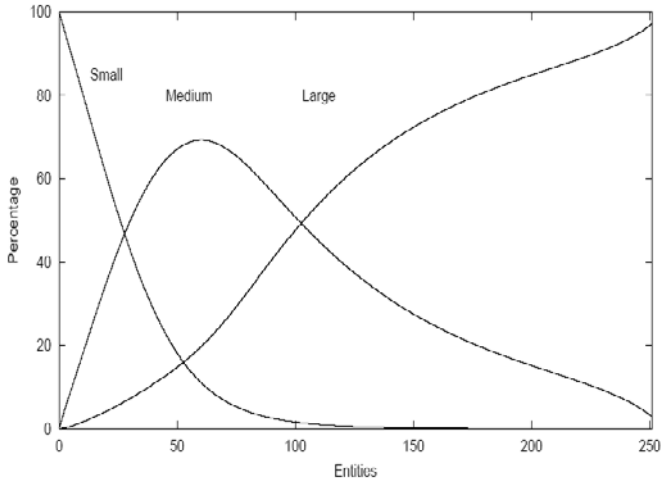
(a) First survey: Raw



(b) First survey: Smoothed

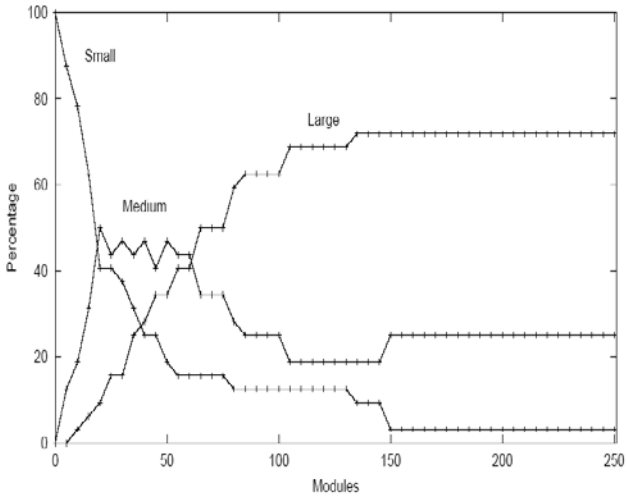


(c) Second survey: Raw

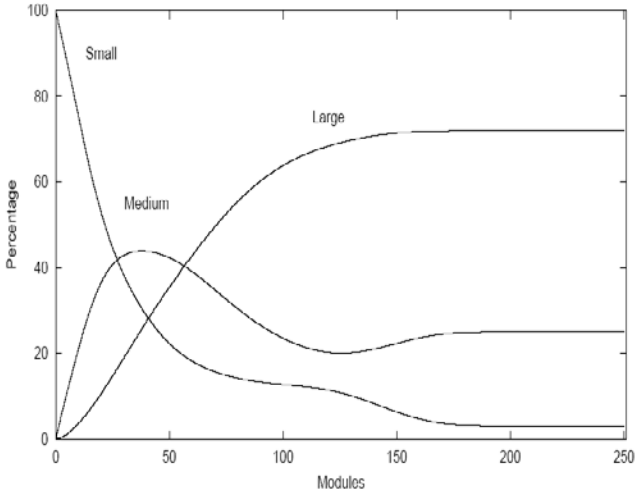


(d) Second survey: Smoothed

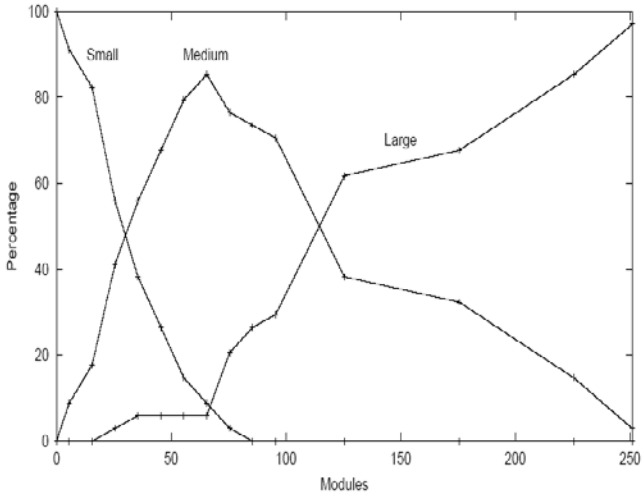
Fig. 2. Number of modules.



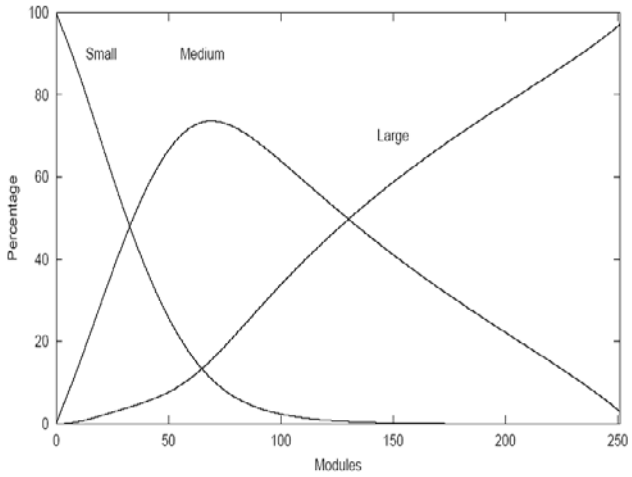
(a) First survey: Raw



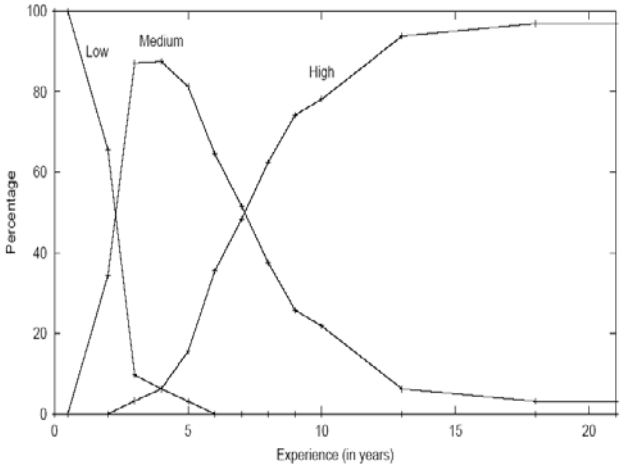
(b) First survey: Smoothed



(c) Second survey: Raw

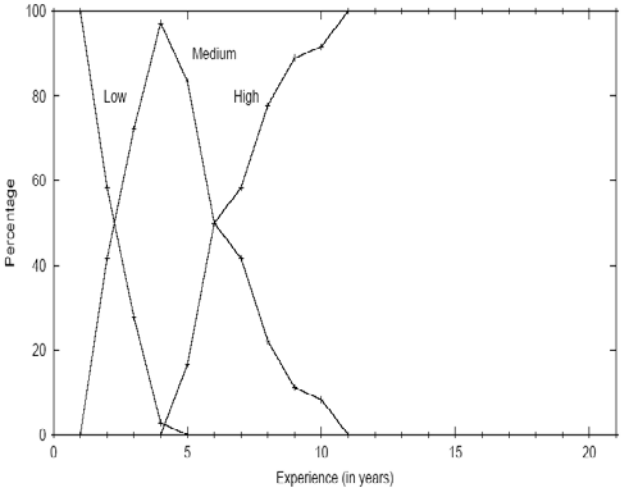


(d) Second survey: Smoothed

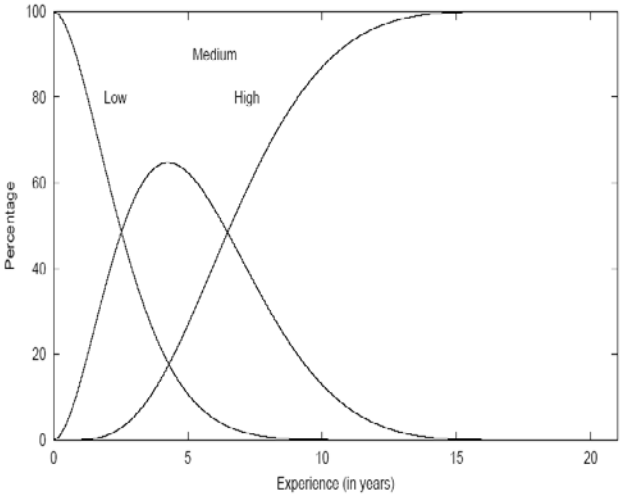


(c) Second survey: Raw

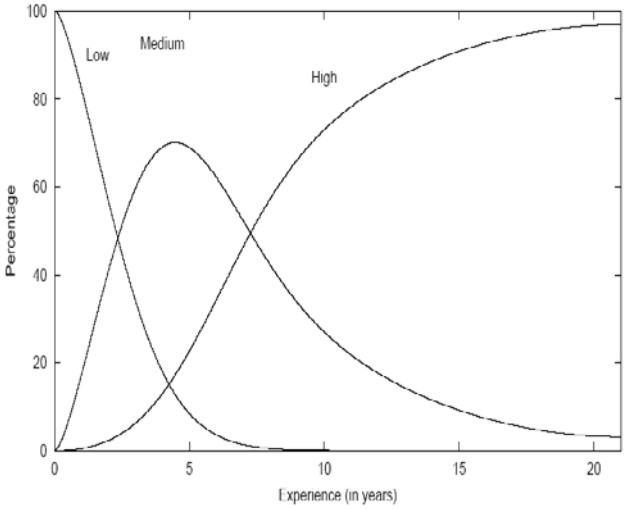
Fig. 3. Experience.



(a) First survey: Raw



(b) First survey: Smoothed



(d) Second survey: Smoothed

In terms of obtaining what might be considered to be 'typical' membership function profiles, those determined from polling for developer experience provide the closest approximation (as shown in Figure 3). There appears to be, for this variable at least, general agreement as to what constitutes a low, medium or high amount of developer experience, implying that the perception of developer experience is more consistent among project managers.

In comparing the two polling approaches it appears that using the 'classify pre-specified values' approach (as adopted in the second survey) resulted in more useful membership functions than that requiring managers to 'give a range of values that fits the label' (as used in survey one). The membership functions show more consistent definition under the second survey approach, especially for the medium size functions. That said, no difference was observed for the developer experience factor, which suggests that this is a much less variable concept for managers. The two surveys involved similar

(in some cases the same) project managers, reinforcing the view that the differences are due to the elicitation method rather than the vagaries of the samples.

It would appear from the above analysis that the use of *standardized* fuzzy logic models for software project management is unlikely, given the significant individual variation in perceptions of membership categories. Since in some cases the membership functions for the medium categories are not even strictly convex there is evidence of considerable disagreement in terms of perceptions. While it is disappointing that the size-based measures were assessed so inconsistently by the managers, this does not in itself invalidate any of the managers using such membership functions. Merely, it would appear unwise for them to share such functions or to use linguistic labels for inter-organization communication without first ensuring that they shared common perceptions. Organizations with managers who were not 'typical' in their perceptions would need to ascertain what they personally defined as suitable membership functions. Furthermore, given that the managers who took part in the survey *were* able to specify membership functions for developer experience with some consistency, it may be that the difficulties related more to the concepts of data model and functionality size rather than to the approach itself. As a result, it appears that fuzzy logic modeling using membership function derivation from knowledgeable personnel remains a potentially effective complementary approach for prediction in software project management.

It is acknowledged here that research into the underlying structure and computation of fuzzy logic modeling has advanced significantly even in the last five years, and that the methods and techniques discussed here are not anywhere near that level of sophistication. That said, this study was intended to be an assessment of the *viability* of fuzzy logic-type approaches in the software management domain, a domain that is relatively immature in its use of non-statistical analysis methods. If project managers could see no appeal in even simple models that incorporate uncertainty and vagueness, then significantly more complicated approaches in terms of their algorithmic form and content would almost certainly be rejected even more readily, reinforcing the need for balance in respect to sophistication and understandability of any method.

5. CONCLUSIONS

On the basis of the evidence presented in the previous sections it is clear that at least some project managers among the survey respondents:

1. would like to have the opportunity to use estimation methods that are able to incorporate both data and knowledge and that also take uncertainty into account
2. continue to employ expert opinion extensively in project management – in fact it may still be the dominant technique
3. are aware of the limitations of existing estimation methods
4. are aware of fuzzy logic modeling and of what it could offer in terms of software project management, with some of these managers interested in using such an approach
5. are able to specify membership functions via polling methods, but are able to do so more consistently using fixed point voting rather than interval declaration.

As the above analyses were based on the views of small samples of project managers it is not possible to say how generally the conclusions of this study might apply. Furthermore, some of the evidence, particularly the quotes derived from the interviews, is anecdotal, and in itself does not provide direct support for the use of fuzzy logic modeling in software project management. However, the overall outcomes are consistent with those of previous studies, and they do support the notion that such an approach could be readily accepted in an industrial software development setting, thus providing sufficient motivation for further research to be undertaken. Ongoing work to investigate the effectiveness of fuzzy logic modeling in comparison to other estimation methods is occurring with an industrial partner. An exploration of the stability and consistency of various rule extraction methods is also under way. While the analysis is not yet complete, results to date suggest that fuzzy logic models, when employed in an organizational learning framework, can perform at least as well as and often better than more commonly employed statistical models. It is also clear that decisions regarding sampling and rule set size can have a considerable impact on the resulting models and their effectiveness in effort estimation. When combined, these results indicate that fuzzy modeling approaches are certainly worth considering further in aspects of software project management, but that care must be taken to ensure that outcomes are not achieved simply as a result of a particular data sample or collection of rules.

6. REFERENCES

1. M.A. Ahmed, M. Omolade Saliu and J. AlGhamdi, Adaptive fuzzy logic-based framework for software development effort prediction, Information and Software Technology 47 (2005) 31-48.
2. A.J. Albrecht and J.E. Gaffney Jr., Software function, source lines of code, and development effort prediction: a software science validation, IEEE

Transactions on Software Engineering 9(6) (1983) 639-648.

3. L. Angelis and I. Stamelos, A simulation tool for efficient analogy based cost estimation, Empirical Software Engineering 5 (2000) 35-68.

4. N. Bilalis, D. Lolos, A. Antoniadis and D. Emiris, A fuzzy sets approach to new product portfolio management, in: Proceedings of the 2002 International Engineering Management Conference, IEEE, 2002, 485-490.

5. B.W. Boehm, Software Engineering Economics, Prentice Hall, Englewood Cliffs NJ, USA, 1981.

6. B.W. Boehm and K. Sullivan, Software economics: status and prospects, Information and Software Technology 41 (1999) 937-946.

7. L.C. Briand, On the many ways software engineering can benefit from knowledge engineering, in: Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, Ischia, Italy, ACM, 2002, 3-6.

8. S.D. Conte, H.E. Dunsmore and V.Y. Shen, Software Engineering Metrics and Models, Benjamin/Cummings, Menlo Park CA, USA, 1986.

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

10. M.E. Fayad, M. Laitinen and Robert P. Ward, Software engineering in the small, Communications of the ACM 43(3) (2000) 115-118.

11. N.E. Fenton and M. Neil, Software metrics: successes, failures and new directions, Journal of Systems and Software 47 (1999) 149-157.

12. G.R. Finnie, G. E. Wittig and J-M. Desharnais, A comparison of software effort estimation techniques: using function points with neural networks, case-based reasoning and regression models, Journal of Systems and Software 39 (1997) 281-289.

13. A.R. Gray and S.G. 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, IEEE Computer Society Press, 1997, 394-399.

14. A.R. Gray and S.G. 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. IEEE Computer Society Press, 1999, 258-262.

15. M. Hapke, A. Jaszkiewicz and R. Slowinski, Fuzzy project scheduling system for software development, Fuzzy Sets and Systems 67 (1994) 101-117.

16. W. Herroelen and R. Leus, Project scheduling under uncertainty: Survey and research potentials, European Journal of Operational Research (In Press).

17. X. Huang, L.F. Capretz, J. Ren and D. Ho, A neuro-fuzzy model for software cost estimation, in: Proceedings of the Third International Conference on Quality Software, IEEE Computer Society Press, 2003, 126-133.

18. A. Idri and A. Abran, A fuzzy logic based set of measures for software project similarity: validation and possible improvements, in: Proceedings of the 7th International Symposium on Software Metrics, London, IEEE Computer Society Press, 2001, 85-96.

19. A. Idri, A. Abran and L. Kjiri, COCOMO cost model using fuzzy logic, in: Proceedings of the 7th International Conference on Fuzzy Theory and Technology, Atlantic City NJ, 2000, 1-4.

20. A. Idri, A. Abran and T.M. Khoshgoftaar, Estimating software project effort by analogy based on linguistic values, in: Proceedings of the 8th International Symposium on Software Metrics, Ottawa ON, Canada, IEEE Computer Society Press, 2002, 21-30.

21. A. Idri, T.M. Khoshgoftaar and A. Abran, Can neural networks be easily interpreted in software cost estimation? in: Proceedings of the 2002 World Congress on Computational Intelligence, Honolulu, Hawaii, 2002, 1162-1167.

22. T.M. Khoshgoftaar and E. B. Allen, Neural networks for software quality prediction, in: W. Pedrycz and J.F. Peters (eds), Computational Intelligence in Software Engineering, Singapore, World Scientific, 1998, 33-64.

23. B. Kitchenham, The certainty of uncertainty, in: Proceedings of the European Software Measurement Conference FESMA'98, Antwerp, Belgium, 1998, 17-25.

24. G.J. Klir and B. Yuan, Fuzzy Sets and Fuzzy Logic: Theory and Applications, Prentice Hall, Englewood Cliffs NJ, USA 1995.

25. S. Kumar, B.A. Krishna and P.S. Satsangi, Fuzzy systems and neural networks in software engineering project management, Journal of Applied Intelligence 4 (1994) 31-52.

26. A.L. Lederer and J. Prasad, Software management and cost estimating error, Journal of Systems and Software 50 (2000) 33-42.

27. A. Lee, C.H. Cheng and J. Balakrishnan, Software development cost estimation: integrating neural network with cluster analysis, Information & Management 34 (1998) 1-9.

28. J-Q. Li and Y-S. Fan, Coordination scheduling based on fuzzy concepts, in: Proceedings of the 1st International Conference on Machine Learning and Cybernetics, Beijing, China, IEEE, 2002, 1489-1492.

29. O. de S. Lima Jr, P.P.M. Farias and A.D. Belchior, Fuzzy modelling for function point analysis, Software Quality Journal 11 (2003) 149-166.

30. C-T. Lin and Y-T. Chen, Bid/no-bid decision-making – a fuzzy linguistic approach, International Journal of Project Management 22 (2004) 585-593.

31. X. Liu, G. Kane and M. Bambroo, An intelligent early warning system for software quality improvement and project management, in: Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence, IEEE Computer Society, 2003, 32-38.

32. S.G. MacDonell, Software source code sizing using fuzzy logic modeling, Information and Software Technology 45(7) (2004) 389-404.

33. S.G. MacDonell, A.R. Gray and J.M. Calvert, FULSOME: A fuzzy logic modeling tool for software metricians, in: Proceedings of the 1999 Annual Meeting of the North American Fuzzy Information Processing Society. IEEE Computer Society Press, 1999, 263-267.

34. S.G. MacDonell, A.R. Gray and J.M. Calvert, FULSOME: Fuzzy logic for software metric practitioners and researchers. In Proceedings of the 6th International Conference on Neural Information Processing ICONIP'99, ANZIIS'99, ANNES'99, and ACNN'99. Perth, Western Australia, IEEE Computer Society Press, 1999, 308-313.

35. R.J. Madachy, Heuristic risk assessment using cost factors, IEEE Software (May/June 1997) 51-59.

36. P. Musílek, W. Pedrycz, G. Succi and M. Reformat, Software cost estimation with fuzzy models, ACM SIGAPP Applied Computing Review 8(2) (2000) 24-29.

37. I. Myrtveit and E. Stensrud, A controlled experiment to assess the benefits of estimating with analogy and regression models, IEEE Transactions on Software Engineering 25(4) (1999) 510-525.

38. M.C. Ohlsson, C. Wohlin and B. Regnell, A project effort estimation study, Information and Software Technology 40 (1998) 831-839.

39. L. Özdamar and E. Alanya, Uncertainty modeling in software development projects (with case study), Annals of Operations Research 102 (2001), 157-178.

40. H. Pan, C-H. Yeh and R.J. Willis, Computer-aided system to solve uncertainty in project management, in: Proceedings of the IEEE International Fuzzy Systems Conference, IEEE Computer Society Press, 2001, 1376-1379.

41. M.J. Pazzani, Knowledge discovery from data? IEEE Intelligent Systems (March/April 2000) 10-13.

42. G. Peeters and G. Dewey, Reducing bias in software project estimates, CrossTalk – The Journal of Defense Software Engineering (April 2000) 20-24.

43. L.H. Putnam and W. Myers, How solved is the cost estimation problem? IEEE Software (Nov/Dec 1997) 105-107.

44. A. Raman and A. Noore, Software metrics for real-time systems using fuzzy sets, in: Proceedings of the 35th Southeastern Symposium on System Theory, IEEE, 2003, 74-78.

45. M. Reformat, W. Pedrycz and N. Pizzi, Building a software experience factory using granular-based models, Fuzzy Sets and Systems 145 (2004) 111-139.

46. J. Ryder, Fuzzy modeling of software effort prediction, in: Proceedings of the IEEE Information Technology Conference, IEEE, 1998, 53-56.

47. B. Samson, D. Ellison and P. Dugard, Software cost estimation using an Albus perceptron (CMAC), Information and Software Technology 39 (1997) 55-60.

48. U.Z. Sanal, A decision support system for fuzzy scheduling of software projects, in: Proceedings of IEEE AUTOTESTCON, IEEE, 2000, 263-272.

49. V.L. Sauter, Intuitive decision-making, Communications of the ACM 42(6) (1999) 109-115.

50. M. Shepperd and M. Cartwright, Predicting with sparse data, IEEE Transactions on Software Engineering 27(11) (2001) 987-998.

51. M. Shepperd and C. Schofield, Estimating software project effort using analogies, IEEE Transactions on Software Engineering 23(12) (1997) 736-743.

52. M. Shin and A.L. Goel, Empirical data modeling in software engineering using radial basis functions, IEEE Transactions on Software Engineering 26(6) (2000) 567-576.

53. M.F. Shipley, A. de Korvin and K. Omer, BIFPET methodology versus PERT in project management: fuzzy probability instead of the beta distribution, Journal of Engineering and Technology Management 14 (1997) 49-65.

54. E. Stensrud and I. Myrtveit, Human performance estimating with analogy and regression models: an empirical validation, in: Proceedings of the 5th International Symposium on Software Metrics, Bethesda ML, USA, IEEE Computer Society Press (1998) 205-213.

55. L. Tian and A. Noore, Multistage software estimation, in: Proceedings of the 35th Southeastern Symposium on System Theory, IEEE, 2003, 232-236.

56. USC COCOMO II Model Definition Manual, version 1.4, University of Southern California, 1997.

57. F. Walkerden and R. Jeffery, An empirical study of analogy-based software effort estimation, Empirical Software Engineering 4 (1999) 135-158.

58. G.E. Wittig and G.R. Finnie, Using artificial neural networks and function points to estimate 4GL software development effort, Australian Journal of Information Systems 1(2) (1994) 87-94.

59. Z. Xu and T.M. Khoshgoftaar, Identification of fuzzy models of software cost estimation, Fuzzy Sets and Systems 145 (2004) 141-163.

60. R.R. Yager and D.P. Filev, Essentials of Fuzzy Modeling and Control, Wiley, New York, 1994.