A METHODOLOGY FOR MEASURING THE RISK ASSOCIATED WITH A SOFTWARE REQUIREMENTS SPECIFICATION

Trevor T. Moores and Ralph E. M. Champion
Department of Information Systems
City University of Hong Kong
Kowloon Tong, HONG KONG.

ABSTRACT

This paper presents a six-step metrics-based methodology for assessing the risks associated with - and hence the resources required to implement - the requirements contained within a software requirements specification (SRS). The method seeks to eliminate the use of subjective probability assessments in models of risk exposure (RE) and risk reduction leverage (RRL). Measurements are taken of the number of requirements and the class of risk, the number of change requests and their date of issue, and the cost of each requirement change. The class of requirements risk is tailored to a given organisation using the Delphi method. The information collected is stored as an historical database for use in the analysis of subsequent SRSs.

INTRODUCTION

Identifying and controlling the risk associated with the development of an information system is seen as critical for the management of a software development project (Alter, 1979; McFarlan & McKenney, 1983; Charette, 1989; Chittister & Haimes, 1993). "Risk" is typically defined as 'the possibility of incurring misfortune or loss'. In software development terms, this usually implies the potential for some disruptive event to impinge upon the project. The key problem with risk is that it cannot be avoided, it can only be anticipated and its effects reduced. For a programmer this may result in the need to redo work. For a project manager this may result in a failure to deliver on-time and on-budget. For an organisation this may result in a failure to carry out its business (Tate, 1988; Haimes, 1991).

While a number of studies have been carried out to identify the range of factors which represent risk to a software development project (e.g., Anderson & Narasimhan, 1979; Zmud, 1980; Beath, 1983; Charette, 1990), the traditional use of probability or uncertainty as a measure of risk has meant that a quantitative assessment of risk exposure (RE) and risk reduction leverage (RRL) relies on a subjective estimate of the associated probability based on experience. It is well known, however, that humans subjects are notoriously bad at reasoning using probabilities (Kahneman & Tversky, 1982).

This paper will develop a six-step metrics-based methodology which allows the subjective probability ratings used in risk assessment models to be replaced by objective (empirical) measurements. The focus here will be on the verification of IS development (assessing the evolution of the system) rather than on validation (assessing the IS solution). Since the cost of reworking software requirements is more than 100 times cheaper than the cost of making changes at the operations stage (Boehm, 1981), the most cost-effective point at which to apply the risk management methodology is at the requirements stage. Hence, the methodology will focus on the applicability of risk analysis at this stage. The next section will discuss the RE and RRL models used to assess the potential impact of risk on a software development project. The applicability of software metrics to a software requirements document will then be discussed before providing a detailed description of the techniques required to implement such a methodology within an organisation. Areas of further research will then be identified, in particular, the possibility of developing a knowledge-based front-end to the methodology.

ASSESSING AND MANAGING RISK

Given the definition of risk as the *potential* for an event to occur, the control of risk is a two-step process involving: the assessment of which risk factors are likely to be important; and, the management of risk by measuring and prioritising the probability of the event and it's impact on the project. According to Kaplan & Garrick (1981), risk assessment asks three questions for each phase of the development process:

1. What can go wrong?
2. What is the likelihood that it will go wrong?
3. What would the consequences be?

These studies have typically been carried out by asking project managers or users to rate (an existing list of factors) or specify (where no prior model is in place) the importance of a list of potential risks. Factor analysis allows the subsequent list to be condensed and grouped under general headings. For instance, Boehm (1989, 1991) identifies four major headings of systems failure which set out what can go wrong, namely: hardware problems, software problems, organisational problems, and human problems.

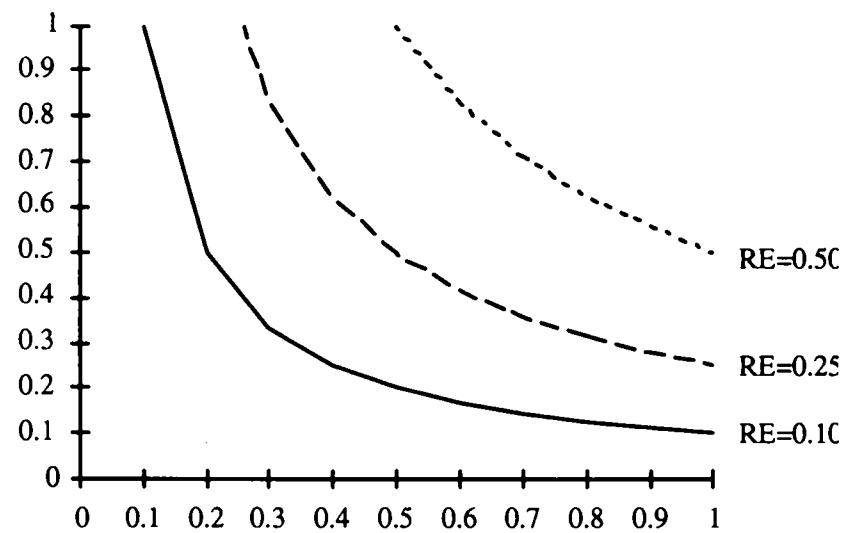
Boehm suggests that the full list of problems (see Table 1) can be used as a checklist by a project manager to identify the most likely risks. Boehm defines risk exposure (RE) and risk reduction leverage (RRL) as the key measures, where:

RE = Prob(UO) * Loss(UO)
RRL = $(RE_{before}-RE_{after})$/Risk Reduction Cost

Table 1. The 10 most important risk factors by category (Boehm, 1989, 1991)

Category		Risk factor
People	1.	Personnel shortfalls
Resources	2.	Unrealistic schedule and budgets
Requirements	3.	Developing the wrong software functions
	4.	Developing the wrong user interface
	5.	Gold-plating
	6.	Continuing stream of requirements changes
External	7.	Shortfalls in externally developed software
	8.	Shortfalls in external personnel
Technology	9.	Shortfalls in real-time performance
	10.	Straining at the bounds

Figure 1. Risk exposure (RE) contours



Risk exposure is the product of the probability of some unsatisfactory outcome, Prob(UO), and the loss due to this outcome incurred by the project, Loss(UO). The project manager would need to calculate RE for each identified risk and decide a level of acceptable risk exposure. For factors which exceed the acceptable level of RE, the project manager would then have to either reduce the probability of the event occurring, or reduce the loss associated with it (see Figure 1).

For instance (from Boehm, 1989, p8), if the loss associated with having a particular type of interface error is estimated at US$1million and from experience we estimate that the probability of such a fault - on a scale of 0 to 1.00 - is 0.30, then the risk exposure is given as:

RE = (0.30)*US$1000K
= US$300K

An approach to dealing with this problem could entail buying a requirements and design interface checker for US$20K which would reduce the Prob(UO) to 0.1, or, carrying out a detailed interface test for an additional US$150K of manpower which would reduce the Prob(UO) to 0.05. The RRL of these two approaches (denoted as RRL_1 and RRL_2, respectively) can be calculated as follows:

RRL_1 = ([0.30]*US$1000K-[0.10]*US$1000K)/US$20K
= 10
RRL_2 = ([0.30]*US$1000K-[0.05]*US$1000K)/US$150K
= 1.67

Therefore, the interface checker approach can be seen to deliver the highest risk reduction. Although Boehm's approach may seem reasonable, care must be taken in interpreting these figures. Firstly, although the models of RE and RRL appear to be objective they are not, since although the financial impact of the risk reduction may be accurately calculated, the crucial assessment of the probability of risk is subjectively derived. In other words, the inputs to these models are best guesses by the project manager and so, the risk associated with the guess may in itself account for the difference in result between RRL_1 and RRL_2. Furthermore, cheap alternatives to the above two approaches which do little to reduce the risk can appear to provide better RRL scores. For instance, employing a US$500-a-day consultant for one day may not reduce the risk a great deal, say the Prob(UO) becomes 0.29, but the RRL score, RRL_3, calculates as follows:

RRL_3 = ([0.30]*US$1000K-[0.29]*US$1000K)/US$0.50K
= 20

Neither can it be assumed that the consultant can reduce the risk exposure by 0.01 *every* day, since most errors are typically found early in the testing phase. Given that RRL is simply a ratio, it is unlikely that the project manager will be happy with option 3, since the risk to the project remains at 0.29 and it is only the cheapness of the risk reduction cost that suggests option 3 is a better solution than options 1 or 2. The problem with RE and RRL is the use of "probability" in the model. Although risk is sometimes defined in terms of "uncertainty" rather than probabilities (e.g., Barki *et al*, 1993), the fact remains that subjective ratings are inadequate measures of risk.

Principally, the calculation of a probability is based on an assessment of the number of choices available within an event. The difference between probabilistic and empirical ratings can be illustrated with the following example: Suppose a coin is flipped ten times and it came up heads nine times and tails once. According to probability, the chance of the next flip producing a head, $P(h)_{prob}$, remains 0.5 since it must be one of two equally likely outcomes; however, the empirical result suggests that $P(h)_{emp}=0.9$ since a head has been observed on nine of ten trials. Over a long period of time, it would be expected that $P(h)_{prob} \cong P(h)_{emp}$, but this could only be derived from empirical observation. In other words, $P(h)_{emp}$ is a surrogate for $P(h)_{prob}$ when the theoretical basis for calculating $P(h)_{prob}$ is not available.

Software development is sufficiently complex for it to be assumed that $P(h)_{prob}$ is not available and any models making use of probabilities - such as RE and RRL - must be revised. Rather than the use of probabilities, therefore, there is a need to apply objective measurements of the risks involved.

METRICS-BASED RISK ASSESSMENT

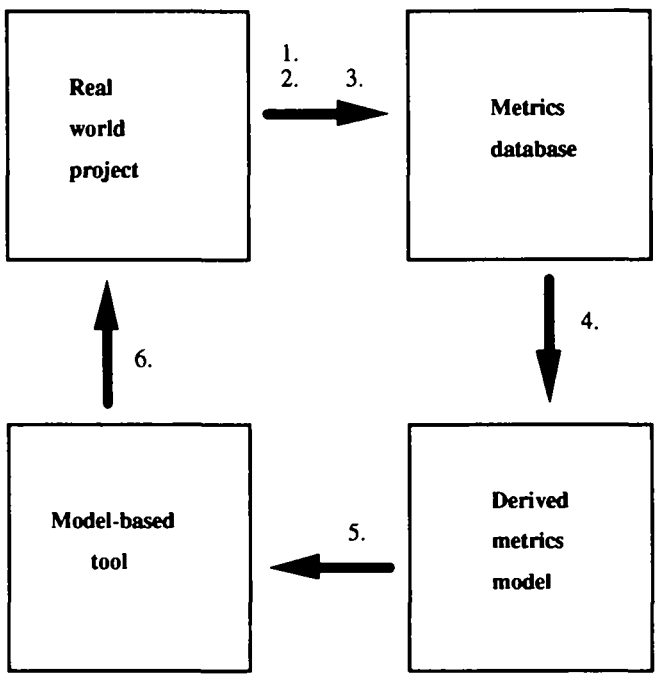
A software metric is a measurement of some attribute of the software process or products that relates directly or indirectly to some property of the system that needs to be assessed or predicted. For instance, metrics have been developed to measure the attribute "size" of a software system in order to *assess* properties such as the effort required to implement the system (e.g., Halstead, 1977; Albrecht & Gaffney, 1983), and *predict* properties such as the cost of the project as a whole (e.g., Putnam, 1978; Freiman & Park, 1979; Boehm, 1981). The general methodology for collecting metrics in order to quantify an attribute of the software development process is a six-step process, as follows:

1. Identify which attribute needs to be assessed (or predicted).
2. Identify those properties which are to be used as a measure of the attribute identified in '1'.
3. Collect the measures defined in '2' until a database of sufficient size has been created.
4. Quantify the relationship between the properties '1' and '2'.
5. Apply the model on the metrics database to test its accuracy.

6. If the model works sufficiently well, use the model to estimate '1' on a live project; otherwise, start again at step '2'.

This sequence of steps sets up a project→metrics→model→tool cycle (see Figure 2), where the tool is evaluated against real world projects, and the data collected (now including the estimate) is used to update and refine the models.

Figure 2. The project→metrics→model→tool cycle



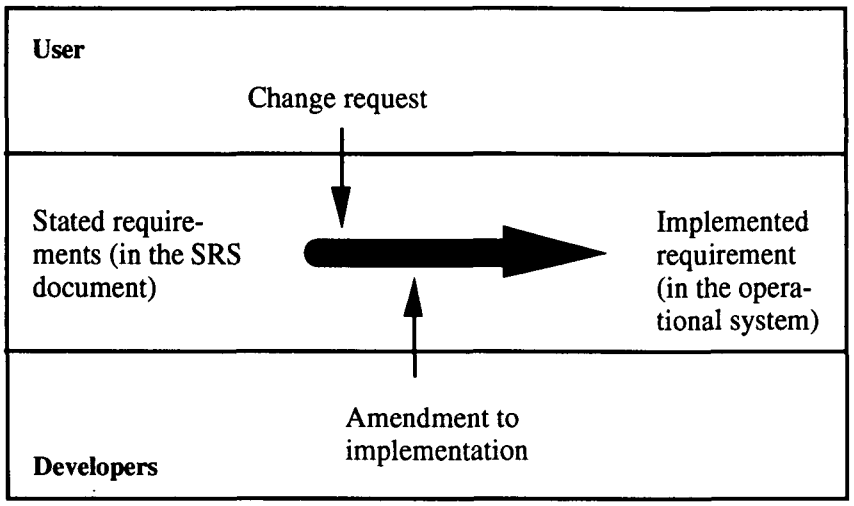
Defining a set of measurements

Steps '1' and '2' are the key, since only with a sensible definition of the attribute to be assessed and the measurements required can useful metrics models be derived. Given that software requirements specification (SRSs) are more structured than the initial user-defined requirements document, the methodology will be applied to the SRS rather than the user document. The application of software metrics requires both a purpose and an intuitive link between the attribute to be measured and the metric used to measure the effect of the attribute (Basili & Rombach, 1988; Fenton, 1994). In this case, the attribute to be measured is risk; the purpose is to quantify models of RE and RRL; while, the property which is meant to have an intuitive link to risk is the changeability of the requirements and the cost of change. While the model of RRL remains the same, the model of RE becomes:

$$RE = Risk_{change} * Cost_{change}$$

In other words, the probabilistic Prob(UO) factor is replaced by a quantifiable measure of the risk associated with a change to a requirement, $Risk_{change}$, and the cost needed to accommodate this change, $Cost_{change}$. Both $Risk_{change}$ and $Cost_{change}$ can be quantified if the changeability of any given requirement and the cost of the change can be measured.

Figure 3. A representation of the requirements change process



The measurements defined here are based on the model given in Figure 3, which puts a boundary around the process under study. As can be seen, the implementation of a requirement is represented as a transformation from its written statement in the software requirements specification (SRS), to its implementation in the operational system. Two parties are involved in this process: the users who decides on the requirements; and, the developers who implement each of the required functions. A change to a requirement is initiated by a change request from the user and in response, the developers make some change to its implementation. It should be noted that the change may include cancellation, and the cost of the change in this case would be the effort already expended. In order to quantify this process, the measurements required are as follows:

- the number of requirements (actual number and category type);
- the number of change requests (actual number with date of issue);
- the cost of each change request (in the relevant currency, e.g., US$).

Counting the actual number of requirements depends principally on the way in which the requirements document is written. On the assumption that "simplest is best", the suggestion here is to take high-level counts of the number of requirements, such as the number of functions defined within a form-based approach (Heninger, 1980).

The ability to derive useful categories of requirements risk is clearly crucial. Boehm (1989) derived his five categories (see again Table 1) based simply on experience of developing large-scale software systems. The approach here, however, is to derive the categories more rigorously using the Delphi method (Delbecq *et al*, 1975). The Delphi method is often used to generate agreement on such things as MIS issues seen as being important for IS executives (e.g., see Niederman *et al*, 1991; Wang, 1994). The objective of the technique is to discuss drafts of a list of issues in order to surface new issues and move participants towards a consensus. Starting with a base-line questionnaire, participants are asked to rate the issues, typically on a scale of 1 to 10 with 1 being the lowest and 10 being the highest. Succeeding questionnaires summarise group responses and the participants are asked to re-evaluate their opinions based on the new evidence. A three-round Delphi method is common and stops after three rounds of questionnaire, although it is possible for the procedure to continue until a reasonable level of consensus is reached.

Table 2. Reasons cited for projects failing to meet user requirements

Reason	No. of responses
Poor/conflicting user participation	14
Lack of understanding	12
Too changeable	12
Too speculative	9
Insufficient time for analysis	7
Uncontrolled changes	6
No agreement made	6
Specification incomplete	6
Inadequate testing	2
Poor development standards	2
No formal review/acceptance	1
Lack of IT/business awareness	1
Inadequate performance	1

As an example, in a survey of IS managers in large UK corporations and computing companies (Moores, 1993), one of the questions asked was:"What are the major reasons for projects failing to meet user requirements?" A ranked list of responses is given in Table 2. Clearly, this particular list would need further analysis since it contains both symptoms and causes of requirements risk. The symptoms listed include a general lack of understanding, requirements being too changeable and too speculative, while the potential causes listed include poor/conflicting user participation, insufficient time for analysis, uncontrolled changes and inadequate testing. This list can be used as the baseline Delphi questionnaire, however. Subsequent questionnaires are then phrased so as to focus participants towards specific causes rather than symptoms of requirements risk.

The list can be tailored for a particular organisation by asking project managers to rate each of the reasons given in the baseline questionnaire or suggest their own additional reasons. Once consensus has been reached, factor analysis based on the rating scores can help to identify the underlying concepts being used. The final list is then the range of categories which are to be used to classify requirements risks. For instance, data would be collected on the number of requirements changes that were identified as being due to *Insufficient time for analysis*. The advantage of this approach over other studies is that it takes into account the idiosyncrasies of the particular organisation, a factor seen as important in developing some types of local metrics models (Kitchenham, 1992).

Counting the number of change requests and the cost of each change depends on their being an adequate change management process in place. The count of change requests may be taken to be the number of Change Request Forms (CRFs) which were actioned, that is, actions were initiated in response to the request. CRFs tend to be simple forms (e.g., see Sommerville, 1992, pp555-8) documenting problems within the system. By only counting those CRFs which were actioned, CRFs that report the same or trivial problems can be ignored.

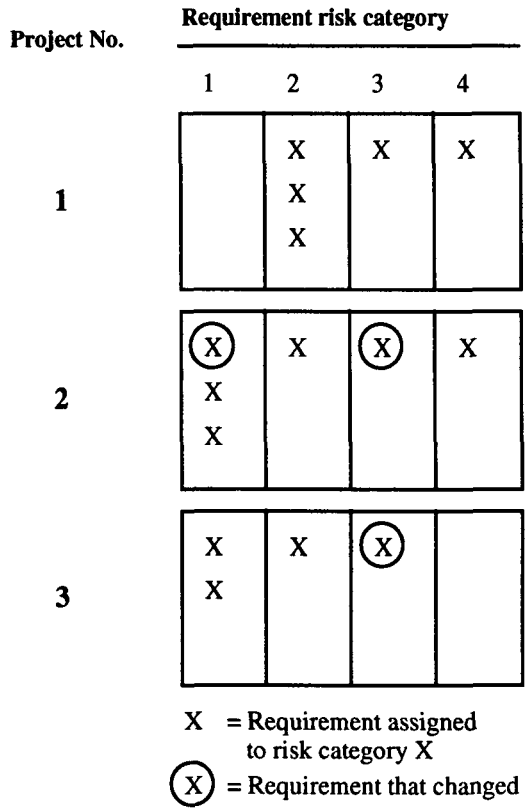
Before a CRF is actioned, the change manager needs to analyse the cost of the change and it's impact on the system (Pressman, 1992). Within this process the cost of the change is immediately available as the cost of the rework scheduled to deal with the change request. Since change requests are possible at any point during the development process, and as Boehm (1981) points out the cost of any change increases down the life-cycle, both $Risk_{change}$ and $Cost_{change}$ are likely to have different values depending on when the change occurs. By recording the date at which the change request is received, this issue can be taken into account.

Building the risk models

The final steps of the metrics-based methodology (steps '3' to '6') are carried out by using the measures defined above to quantify the following relationships:

ReqX = The total number of requirements assigned to risk category X
ChangeX = The total number of ReqX that required a change
CostX = The total cost of ChangeX
$Risk_{change}$ = ChangeX/ReqX
$Cost_{change}$ = CostX/ChangeX

Figure 4. Calculating $Risk_{change}$ for four categories of requirements risk



With measurements taken across a number of projects, the risk exposure can be calculated for each category of requirements risk. For example, consider that measurements are taken across three projects using four broad categories of requirements risk (see Figure 4). With five instances of requirements placed in category 1 and only one instance of a change, $Risk_{change}$ is calculated as 0.20. Similarly, $Risk_{change}$ for category 3 is calculated as 0.66, while categories 2 and 4 have no recorded changes and so $Risk_{change}=0$, $Cost_{change}=0$ and RE=0. In other words, the perceived risk has not been found to entail any actual cost. $Cost_{change}$ for categories 1 and 3 is the average value of ChangeXY and this completes the RE calculation. Each category of risk would be derived using the method described above.

The accuracy of a metrics model (step '5') is typically assessed as a function of both the general predictive ability of the model, PRED, and the size of error attributable to each estimate, MMRE (Conte *et al*, 1986). A metrics model is said to be accurate if the estimate is within 25% of the actual 75% of the time, given as PRED(0.25)≥0.75, *and* the mean magnitude of the relative error is less than 0.25, given as MMRE≤0.25, where,

$$MMRE = \frac{1}{n} \cdot \sum_{i=1}^{n} \left(\frac{|Actual_i - Estimate_i|}{Actual_i} \right)$$

and,

$Actual_i$ = actual value for the i^{th} data point

$Estimate_i$ = estimated value for the i^{th} data point

This definition has been taken on board and widely used although the value of ±25% is clearly an arbitrary threshold. For instance, without formally defining PRED as a relationship, Boehm (1981) took it that a model could be deemed to be accurate if it produced estimates within 20% of the actual 70% of the time, that is, PRED(0.20)≥0.70, while other studies have required a higher level of PRED at PRED(0.20)≥0.80 (Moores & Edwards, 1992). Given the accuracy of risk models have not been properly assessed to-date a weaker level of accuracy is set initially, using PRED(0.25) ≥0.70 and MMRE≤0.25. As more data becomes available and the model is fine-tuned, the accuracy of the models will increase over time.

Having completed the above steps, the risk models derived using this methodology can now be employed on live projects. The person responsible for reviewing the SRS first decides which requirements are subject to some level of risk. For those requirements where a risk category applies, the risk exposure is calculated. For instance, if a requirement dealing with checking valid data entry is known to have been specified without much discussion with the user, then the requirement might bring to the project a risk under the heading of *Poor/conflicting user participation*. If this category of risk is recorded as effecting changes in 1-in-4 projects with an average cost of US$10K, then the RE can be calculated as:

$$\begin{aligned} RE &= Risk_{change} * Cost_{change} \\ &= 0.25 * US\$10K \\ &= US\$2.5K \end{aligned}$$

An SRS with twenty similar requirements would result in RE=US$50K. It is at this point that either the requirements would have to be investigated further so as to eliminate the risk, or instigate strategies of risk management (using RRL) in order to reduce the level of RE to one acceptable to the project manager.

CONCLUSION

This paper has sought to replace the probabilistic models of risk exposure and risk reduction leverage (RE and RRL, respectively) with a 6-step metrics-based methodology which quantifies the risk of change and cost of change to requirements ($Risk_{change}$ and $Cost_{change}$, respectively). The metrics required to quantify risk and the techniques required for their collection have been described in detail. The key to the methodology is that subjective ratings of the probability of a particular risk impacting on a project is replaced by empirically derived weights. These weights accurately reflect the number of times a class of risk has actually affected a project. The classes of requirements risk are derived for a given organisation using the Delphi method. A further advantage of the metrics database is that data would be provided on the most common risks. Such information would enable organisations to investigate reasons for their occurrence and remedies for the problem.

As with most research, a number of issues require further investigation. Firstly, the methodology described above suggests the RE for a project is the sum of the RE for each risk. Clearly, there will be a certain amount of interdependence between risk factors. For example, when considering general project risk, Boehm (1989) has noted that *Personnel shortfalls* and *Unrealistic schedule and budgets* are two of the most important risk factors (see again Table 1). However, any calculation of schedules and budgets would take into account the efficiency of the development team, and it is only by knowing that they are of below average quality that the schedules and budgets can be judged to be "unrealistic." Care must be exercised in using the Delphi method and factor analysis to derive a meaningful set of risk categories, therefore, since it is this list which drives the collection of data and allows the appropriate $Risk_{change}$ and $Cost_{change}$ to be assigned.

Finally, although the metrics-based methodology has removed one potential flaw in the use of risk models - the use of subjective risk probability assessments - there is still an element of skill involved in risk assessment and management. Primarily, it is still up to the project manager to recognise that a requirement contains a certain class of risk. Furthermore, even if the RE is established there will almost certainly be a major difference in how changes to requirements are dealt with. One project manager may postpone delivery of some functionality in order to meet deadlines, whilst another may negotiate extra funding and deliver the system the customer needs. Clearly, skill in terms of negotiation has an impact here.

The use of skill opens up the possibility of developing a knowledge-based front-end to the methodology described in this paper. Such a front-end would help to further remove subjectivity from the assessment or risk by making operational the heuristics in use. The metrics database would provide the information upon which to test the veracity of the heuristics, while the heuristics would help to clarify the data needed to assess and manage risk. The authors are currently attempting to implement such a research programme.

REFERENCES

Albrecht, A. J. and Gaffney Jr., J. E. (1983) **Software function, source lines of code, and development effort prediction: A software science validation**. IEEE Transactions on Software Engineering, Vol 9, No 6, pp 639-648.

Alter, S. (1979) **Implementation risk analysis**. TIMS Studies in Management Sciences, Vol 13, No 2, pp 103-119.

Anderson, J. and Narasimhan, R. (1979) **Assessing implementation risk: A methodological approach**. Management Science, Vol 25, No 6, pp 512-521.

Barki, H., Rivard, S. and Talbot, J. (1993) **Toward an assessment of software development risk.** Journal of Management Information Systems, Vol 10, No 2, pp 203-225.

Basili, V. R. and Rombach, H. D. (1988) **The TAME project: Towards improvement-oriented environments.** IEEE Transactions on Software Engineering, Vol 14, No 6, pp 758-773.

Beath, C. M. (1983) **Strategies for managing MIS: A transaction cost approach.** In Proceedings, 4th ACM International Conference on Information Systems, Houston, USA, December, 1983, pp 415-427.

Boehm, B. W. (1981) **Software Engineering Economics.** Prentice-Hall, Englewood Cliffs, NJ.

Boehm, B. W. (1989) **Tutorial: Software Risk Management.** IEEE Computer Society Press, Los Alamitos, CA.

Boehm, B. W. (1991) **Software risk management: Principles and practice.** IEEE Software, Vol 8, No 1, pp 32-41.

Charette, R. N. (1989) **Software Engineering Risk Analysis and Management.** McGraw-Hill, New York, NY.

Charette, R. N. (1990) **Applications Strategies for Risk Analysis.** McGraw-Hill, New York, NY.

Chittister, C. and Haimes, Y. Y. (1993) **Risk associated with software development: A holistic framework for assessment and management.** IEEE Transactions on Systems, Man, and Cybernetics, Vol 23, No 3, pp 710-723.

Conte, S. D., Dunsmore, H. E. and Shen, V. Y. (1986) **Software Engineering Metrics and Models.** The Benjamin/Cummings Publishing Co., Inc., Menlo Park, CA.

Delbecq, A. L., Van de Ven, A. H. and Gustafson, D. H. (1975) **Group Techniques for Program Planning: A Guide to Nominal Group and Delphi Processes.** Scott-Foresman, Glenview, IL.

Eysenck, M. W. (1984) **A Handbook of Cognitive Psychology.** Lawrence Erlbaum Associates, Hillsdale, NJ.

Fenton, N. (1994) **Software measurement: A necessary scientific basis.** IEEE Transactions on Software Engineering, Vol 20, No 3, pp 199-206.

Freiman, F. R. and Park, R. E. (1979) **Price software model - version 3. An overview.** In Proceedings, IEEE-Pliny Workshop on Quantatative Software Models, pp 32-41.

Haimes, Y. Y. (1991) **Total risk management.** Risk Analysis, Vol 11, No 2, pp 169-171.

Halstead, M. H. (1977) **Elements in Software Science.** Elsevier North-Holland, New York, NY.

Heninger, K. L. (1980) **Specifying software requirements for complex systems: New techniques and their applications.** IEEE Transactions on Software Engineering, Vol 6, No 1, pp 2-13.

Humphreys, P., Svenson, O. and Vári, A. (1983) **Analysing and Aiding Decision Processes** (eds.). North-Holland Publishing Co., Amsterdam.

Kahneman, D. and Tversky, A. (1982) **Judgement under Uncertainty: Heuristics and Biases.** Cambridge University Press, Cambridge.

Kaplan, S. and Garrick, J. B. (1981) **On the quantitative definition of risk.** Risk Analysis, Vol 1, No 1, pp 11-27.

Kitchenham, B. A. (1992) **Empirical studies of assumptions that underlie software cost-estimation models.** Information and Software Technology, Vol 34, No 4, pp 211-218.

McFarlan, F. W. and McKenny, J. L. (1983) **Corporate Information Systems Management: The Issues Facing Senior Management.** Irwin, Homewood, IL.

Moores, T. T. and Edwards, J. S. (1992) **Could large UK corporations and computing companies use Software Cost Estimating tools? A survey.** The European Journal of Information Systems, Vol 1, No 5, pp 311-319.

Moores, T. T. (1993) **Metrics and models to support the development of hybrid information systems.** PhD Thesis, Department of Operations and Information Management, Aston Business School, University of Aston in Birmingham, UK.

Niederman, F., Brancheau, J. C. and Wetherbe, J. C. (1991) **Information systems management issues for the 1990s.** MIS Quarterly, Vol 15, No 4, pp 475-496.

Pressman, R. S. (1992) **Software Engineering: A Practitioners Approach (3rd ed.).** McGraw-Hill, New York, NY.

Putnam, L. H. (1978) **A general empirical solution to the macro software and estimating problem.** IEEE Transactions on Software Engineering, Vol 4, No 4, pp 345-361.

Sommerville, I. (1992) **Software Engineering (4th ed.).** Addison-Wesley, Wokingham.

Tate, P. (1988) **Risk! The third factor.** Datamation, 15 April, pp 58-64.

Wang, P. (1994) **Information systems management issues in the Republic of China for the 1990s.** Information & Management, Vol 26, pp 341-352.

Zmud, R. W. (1980) **Management of large scale software development efforts.** MIS Quarterly, Vol 4, No 2, pp 45-55.