Transactions on Information and Communications Technologies vol 11, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

Improving the software cost estimation process

R. Jack, M. Mannion
Department of Mechanical, Manufacturing and Software Engineering, Napier University, 10 Colinton Road, Edinburgh, EH10 5DT, UK

Abstract

Effort estimation continues to be a weak link in software project management. As part of a research project into improving the cost estimation process, a review of current cost estimation techniques is described. A number of problems are identified with these techniques, in relation to the applicability of the techniques in different environments and in the omission of key project attributes. Improvement of the cost estimation process using locally-calibrated models with explicit product, process and resource issues is recommended.

Introduction

The problems of software development have been widely reported [1,2] and it is commonly accepted that the difficulties lie as much in project management as in technical development. Project management is a term embracing a number of management techniques that are necessary to develop a product which meets its technical requirements (both explicit and implicit) on time and within an agreed budget.

A key aspect of project management is the ability to estimate the required effort and schedule for the project. Estimation involves consideration, and quantification, of the factors involved in the development. These include the processes by which the software is developed, the personnel and support resources involved and the nature of the product itself. The ability to correctly estimate effort is a key skill for a project manager, which is critical to developing a product on time and to an agreed budget. Accurate estimation provides both the developer and the customer with clear targets for developing

the product and, as the project progresses, forms a baseline against which the actual values can be compared.

The success rate of many software development projects, however, suggests that predicting the likely cost and duration of a project remains a major problem. Surveys of organisations [3,4] indicate that :

- only 25% of projects come within their originally predicted cost and schedule
- 66% of companies significantly underestimate the time and cost
- costs for similar projects can vary by up to 200%

Given the importance of estimation to successful project development, these failures indicate that current estimation techniques do not appear to be appropriate for the task. A number of estimating techniques exist but each of them has limitations. This paper reviews some of these problems and suggests ways in which the estimation process can be improved.

Definition of an Estimate

The generally agreed definition of an estimate [5,6] is a prediction (of effort, schedule, staffing levels, etc.) that is equally likely to be above or below the actual result. The estimate is a prediction based not on one number but rather on three - the most likely value and the upper and lower bounds. The quality of an estimate is a function of how quickly it converges to the actual.

There are a number of problems in making estimates [5,6,7] which include :

- lack of estimating experience
- problems of interference from superiors
- bias inherent in the estimation process

The lack of experience in estimation, due to the size and duration of software projects, leads to non-estimation [5]. At the most extreme end of non-estimation is the situation where estimation simply means giving one's superiors the values that they want to hear.

This aspect of interference can be extended to the situation of using estimates for motivation, unrealistic estimates being used to attempt to make staff work harder. This inevitably leads to a reduction in the quality of the product and to demotivated staff [8].

Studies have clearly shown a tendency to over-estimate one's own abilities [5,6], leading to the situation where the effort involved in a task can be

underestimated. This situation is avoided when estimating someone else's capabilities. A key factor in successful estimation appears to be a detachment form the actual task itself, though this has practical difficulties.

A solution to these problems is to provide estimation techniques which use quantifiable project data at different points in the project lifecycle, making it possible to validate more objectively the prediction process. This will provide project managers with a framework for estimation, utilising the knowledge from previous projects. It should remove some of the bias in project estimates and, as the model will provide results based on specific scenarios and constraints, the effect of unrealistic objectives will be clearly shown.

Cost Estimation Techniques

Although known as cost estimation, these techniques generally predict effort and schedule rather than cost. A number of different estimation techniques [3,6,7,9] are in use to predict the cost of software projects, including :

- Estimation by Analogy. Comparing one or more completed projects to a similar new project to predict cost, schedule or effort
- Algorithmic Models. These provide one or more algorithms which predict effort and schedule as a function of product variables.
- Expert judgement. Predictions based on the skills and experience of one or more experts.

These techniques can be used in either a top-down or bottom-up fashion. Top-down estimation generates an overall cost from the global properties of the system, which is then split among the various components. A bottom-up estimate is generated by estimating each component separately and summing the results to give a global cost.

The strengths and weaknesses of all of these techniques are well documented [9]. It can be said that none of the methods are better than the others in all cases and, as such, more accurate estimates are likely to be obtained by using a combination of methods, rather than relying on a single technique.

Two other methods, which are in fact examples of non-estimation [5], are used on projects [6,9] :

- Parkinson. A Parkinsonian principle ("work expands to fill the available volume") is used to equate resources with cost.
- Price-to-win. The cost or schedule estimates is equated with that believed necessary to win the contract.

While all estimation techniques have proponents and have had some degree of success [43,44], the most widely reported technique is the algorithmic model. This paper will focus on the use of algorithmic methods for estimation.

Algorithmic models

An algorithmic model is a formula, or set of formulas, used to predict the costs likely to be incurred in a project. The variables involved in a project fall into three categories - product, processes and resources. The number of variables are considerable and it would be very difficult to model and assess all of them. The approach taken in algorithmic models is to generate single factor models, which are a simplification of all the factors involved.

The models generally provide direct estimates of effort or duration. Most are empirical models (generated by applying statistical techniques to data from previous projects) with one main input, which is usually a prediction of software size. The models take the general form :

$$\text{effort} = a + b * (\text{size})^{c}$$

where a, b and c are constants derived by the statistical analysis of the data.

A selection of the more-widely known models is described by Boehm [9], COCOMO being the most widely used. All of these use Lines of Code as the main input, though the definition varies between models.

The format of each of the models is generally similar, the exception being the exponent term (c in the equation above). While most studies suggest that $c > 1$, exhibiting diseconomy of scale [9], others suggest the relationship should be linear [6,41] i.e. $c = 1$. There is even the suggestion of economy of scale, as described by Walston-Felix study, but this relationship is not generally accepted. The techniques usually also support a method for predicting the nominal duration of the project, from which most researchers agree that the relationship has the form that nominal duration is proportional to effort to the power of one third [6,9,10,11].

The models have a number of adjustment factors, known as cost drivers [9], which may effect productivity and are used to adjust the basic estimates. These drivers are assumed to be independent of each other.

A variation of these models are constraint (or time-sensitive) models, which describe a relationship over time between two or more cost parameters. These models are used principally to quantify the effects of varying schedule against

Transactions on Information and Communications Technologies vol 11, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

effort and staffing levels. The most widely known technique is the SLIM estimation tool [12], a proprietary product utilising the Rayleigh curve representation of staffing levels [13,14]. COCOMO [9] has a similar mechanism, though it also places a limit on schedule compression beyond which a project is not feasible.

Some Problems with Algorithmic Methods

Algorithmic models are derived from existing data and are tested against that data. Another important test, however, is the validation of the model [9, 21] by evaluating it against systems other than those involved in determining the model. The models have been used and analysed in a number of ways [7,10,15-20,22-28], which have led to a number of consistent points emerging :

- the methods rely on an estimate of size, a product measure usually expressed in lines of code (LOC). This has two main drawbacks - the inability to estimate LOC accurately at the start of the project and the late availability of the actual LOC. Better sizing techniques, using actual data available early in the project, would improve this.

- lack of evidence for the successful use of cost drivers. The cost drivers allow the effort and schedule predictions to be tailored to specific project circumstances. Most studies suggest that :
  - application of cost drivers does not improve accuracy of estimates
  - the subjective nature of the drivers causes difficulties in their use
  - the drivers are not always appropriate
  - the drivers are not independent of each other
  - only a limited set of product, process and resource issues appear in the cost drivers. This is especially true in the case of personnel issues and is a function of what was considered important when the model was generated.

  It appears [10,18] that locally-derived models, as they more closely reflect a specific environment, generate more accurate results than general-purpose models. For specific projects, however, an assessment of particular characteristics must still influence the predictions and this needs to be incorporated.

- the time constraint models are suspect, in particular :
  - the zero staffing assumption at start of design, from the Rayleigh curve, is not valid in practice
  - different models predict different behaviour . SLIM [12] predicts that constraining the schedule (i.e. the nominal duration) increases effort, while relaxing schedule reduces the effort. COCOMO [9] predicts an increase in effort if the nominal duration is increased or decreased.

An investigation of the relationship of effort and schedule to staffing levels, and the effect on project personnel of these effort and schedule constraints, needs to be further investigated.

The way ahead in cost estimation research is to develop cost estimation models using site-specific information but, for different projects within that site, to allow specific variation due to differences in processes, the actual product and the available resources (especially personnel). Some of the directions suggested above are discussed further in the following sections.

Product Issues

Research into the area of product issues has focused on specification of product characteristics and, especially for effort estimation, functional sizing techniques.

Specification of Product Characteristics

All software products are a combination of factors, such as functionality, performance and reliability [50,51,53]. The presence, or absence, of these functional and non-functional requirements will affect the prediction of effort and schedule on a project. Each software product will address these attributes in a different manner but, in all cases, the attributes need to be specified in terms that can be measured. Various methods for quantifying these factors have been proposed [51,54,55] along with various measurement techniques [6]. The COQUAMO approach [50,51] has adopted a similar approach to COCOMO [9] but based on assessing and measuring the requirements of a system. It remains to be shown, however, how this approach could be used in effort prediction.

Functional Sizing Techniques

The earlier a size measure is available the more useful it is as a predictor [5]. User requirements can be analysed to give some measure of size but the first meaningful measure available to a project occurs at the end of the requirements analysis phase, just prior to the commencement of the software design [4,5]. A number of specification metrics have been postulated, the most widely known and used being Albrecht's Function Points [29,30] and its variant known as Mark II [31], both of which are now in very common use [6,37]. DeMarco developed a technique ideally suited to the structured analysis methodology [5,40], while other measures have been suggested for use with CASE tools [35].

In essence, these techniques rely on counting the number of occurrences of different types of entities, classifying each with a weighting factor, and summing these weighted values to produce an unadjusted count, a measure of functional size. By considering a number of technical complexity factors, each with different weightings for different perceived complexity levels, an adjustment factor relating to the product environment is determined. This is subsequently

Transactions on Information and Communications Technologies vol 11, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

multiplied to the unadjusted count. This subsequent variation in size is intended to reflect the variation in difficulty between products with the same functionality but differing environments. The value can be related to productivity (from past projects) or, by use of Line of Code expansion figures, as an input to algorithmic models such as COCOMO [9].

Problems with Functional Sizing Techniques

A number of major problems, however, exist with FPA as a sizing or an effort estimation technique [7,10,18,19,32,34-40], which can be summarised by :

- variations in functional size due to counting differences. In particular, two areas cause concern :
  - the wide variation in counting practices between organisations and even between individuals within organisations
  - the variation in size from the requirements document against the size from the actual system, due in part to the effect of 'creeping elegance' but also due to the amount of information available

- the use of adjustment factors is subjective and appears to add no value to the accuracy of results. The factors are not independent and many are contentious. There is also the problem of similarities in adjustment factors with those in algorithmic models, leading to aspects being counted more than once

- the lack of accuracy as a predictor of effort. This is especially true outside of the areas for which they were originally intended but even in the data processing area the relationship of size to effort does not give consistently accurate results.

The use of a counting technique using information early in the lifecycle is undoubtedly the best approach but a consistent counting method must be developed, independent of adjustment factors, and the relationship of this count to effort predictions needs to be better understood. Work is currently underway to standardise the counting process [49].

Process Issues

Recent interest in software engineering has focused on the development process as a management problem. Standards [51,52,56] such as the TickIT scheme, ISO 9001 and the Capability Maturity Model have been developed to provide defined processes for all of an organisation's activities. These processes cover not only development but supporting issues such as configuration management and corrective action. The philosophy behind these documents is that good processes will lead to good products.

While these techniques seek to improve the quality of the product, there is as yet little evidence to equate different processes to the effort required on a project.

Personnel Issues

Any successful estimation technique needs to address resource issues such as available hardware, software and personnel. While different hardware and software will affect the effort and schedule on a project, the most difficult aspect to quantify is the personnel issue. A few studies have been carried out in this area [18,34,42,46,48] but a number of interesting points have been made, including :

- use of new support tools has an initial detrimental effect on productivity
- back-end support tools do not appear to have a significant effect on productivity
- it has been suggested that more experienced staff are more productive than inexperienced staff
- some studies suggest team structure and individual characteristics have an effect on the quality of the product, while others suggest that project management techniques cancel out these aspects
- it has been demonstrated that, regardless of feedback from a project, estimators will continue to be influenced by the initial estimate
- the effect on staff of constraining project duration is not well defined

It would appear that the effect of personnel on software development costs, both at the beginning of a project and as the project progresses, is still not very well understood.

Local Calibration

Calibration is the process of tailoring a model to the specific environment in which it is being used. The original intention of the cost drivers (of whatever form) was to allow a single model to be applied on different sites and to different types of development. The problems with these adjustment factors have been highlighted already and this approach seems to have limited use for the majority of industry. An alternative technique is to develop a model unique to a particular environment, which should more accurately reflect that environment. A number of studies [16,17,19,22] suggest that local models are more accurate than all-purpose models. Generation of a local model, however, is not an easy task. It involves [47] the following activities :

- collection and maintenance of a database of past projects

Transactions on Information and Communications Technologies vol 11, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

- a statistical analysis of the data to derive an appropriate model relationship
- testing and validation of the model

The MERMAID [18,41,45] approach to effort estimation utilises this locally-defined modelling technique. It allows for a model to be directly calibrated using only local data and, for a specific project, makes use of feedback data as the project progresses. The take up of this technique, however, will require a clear demonstration of the accuracy of the approach and the provision of tools and techniques to facilitate collection and analysis of the data.

It will also be necessary to understand exactly how project managers use the feedback data and what effect this will have on controlling the project. While the technique supports some aspects of the process view, by developing different models for different lifecycles, the impact of different personnel or product attributes needs to be investigated.

Conclusions

Software effort estimation is of crucial importance in software development. Recent results indicate that, while current algorithmic methods are of some use, the process of effort estimation needs to be improved. The way forward in effort estimation would appear to be the generation of locally-defined models with the following characteristics :

- the primary input to be a size measure, which can be compared with other organisations
- the size measure needs to be adjusted to reflect product characteristics other than simply those concerning functionality
- the adjusted product size be tailored by local factors (such as the processes and resource issues) to derive an effort and schedule prediction

It is in the area of addressing process and resource issues that our current research is focused. In particular, the following aspects are being investigated :

- how the effects of personnel issues can be explicitly modelled in the estimation process, especially in relation to the modelling of staffing for different process and products
- how the feedback information available to a project is used and, in particular, how this affects the personnel involved

The intention of this work is to develop a framework in which these different attributes can be assessed and measured within a locally-defined estimation model.

REFERENCES

1. Wingrove, A (1986). " The problems of managing software projects" , *SEJ* 1(1), p3-6.
2. Rook, P (1986). 'Controlling software projects', *SEJ* 1(1), p7-16.
3. Charette, R (1993). 'Improving cost and schedule estimates', *Software Management* 36, p4-9
4. *Computer Weekly*, 19 May 1994.
5. DeMarco, T (1982). *Controlling Software Projects*, Prentice-Hall
6. Fenton, N (1992). *Software Metrics*, Prentice-Hall
7. Heemstra, F (1992). 'Software Cost Estimation', *IST* 34(10), p627-639.
8. DeMarco, T (1990). 'Software productivity : the hidden agenda', *IST* 32(3), p225-227.
9. Boehm, B (1981). *Software Engineering Economics*, Prentice-Hall
10. Kemerer, CF (1987). 'An empirical validation of software cost estimation models', *CACM* 30(5).
11. Kitchenham, BA, Taylor NR (1984). 'Software cost models', *ICL Tech* J 4(1), p73-102.
12. Putnam, LH (1978). 'A general empirical solution to the macro software sizing and estimating problem', *IEEE TSE* 4(4), p345-361.
13. Putnam, LH, Fitzsimmons, A (1979). 'Estimating software costs', *Datamation*, Sept p189-198.
14. Norden, P (1963). 'Useful tools for project management', *Operations research in Research and development*, John Wiley and Sons.
15. Boehm, BW (1984). 'Software engineering economics', *IEEE TSE* 10(1), p7-19.
16. Marwane, R, Mili, A (1991). 'Building tailor-made software cost model : TUCOMO', *IST* 33(3), p224-231.
17. Pfleeger, S (1991). 'Model of software effort and productivity', *IST* 33(3), p224-231
18. Kitchenham, B (1992). 'Empirical studies of assumptions that underlie software cost-estimation models', *IST* 34(4), p211-217
19. Mukhopadhyay, T , Kekre S (1992). 'Software Effort Models for Early Estimation of Process Control Applications', *IEEE TSE* 18(10), p915-923
20. Watson, K (1992). 'COCOMO as a schedule prognosis and validation tool : a case study', *SQJ* 1(4), p193-208
21. MacDonell, SG (1994). 'Comparative review of functional complexity assessment methods for effort estimation', **SEJ** 9(3), p107-116.
22. Stricker, C (1993). 'Evaluating effort prediction systems', *Proc 10th CSR Workshop*.

23. Nevalainen, R (1993). 'An empirical software cost estimating tool : LATURI', *Proc ESCOMM.*
24. Jeffrey, DR (1987). 'Time sensitive cost models in commercial MIS environments', *IEEE TSE* 13(7), p710-730
25. Jeffrey, DR (1987). 'The relationship between team size, experience and attitude and software development productivity', *Proc COMPSAC*, IEEE Comp Soc Press
26. Jeffrey, DR, Low G (1990). 'Generic estimation tools in management of software development', *SEJ* 5(4), p215-221
27. Cuelenaere AME, van Genuchten MJI, Heemstra HJ (1987). 'Calibrating a software estimation model : why and how' *IST* 29(10), p558-567
28. Parr, FN (1980). 'An alternative to the Rayleigh curve model for software development effort' *IEEE TSE* 6(3), p291-296
29. Albrecht, AJ (1979). 'Measuring application development productivity', *Proc IBM App Dev Joint SHARE/GUIDE*, p83-92
30. Albrecht AJ, Gaffney JE (1983). 'Software function, source lines of code and development effort prediction : a software science validation', *IEEE TSE* 9(6), p639-648
31. Symons, CR (1988). 'Function point analysis : difficulties and improvements' *IEEE TSE* 14(1), p2-11
32. Low, GC, Jeffrey DR (1990). 'Function points in the estimation and evaluation of the software process' *IEEE TSE* 16(1), p64-71
33. Kusters RJ, van Genuchten MJIM, Heemstra FJ (1990). 'Are software cost models accurate ?', *IST* 32(3), p187-190
34. Low, GC, Jeffrey, DR (1991). 'Software development productivity and back-end case tools', *IST* 33(9), p616-621
35. Verner J, Tate G (1992). 'A software size model', *IEEE TSE* 18(4), p265-278
36. Betteridge, R (1992). 'Successful experience of using function points to estimate projects costs early in the lifecycle', *IST* 34(10), p655-658
37. Kemerer,C, Porter BS (1992). 'Improving the reliability of function point measurement : an empirical study', *IEEE TSE* 18(11), p1011-1024.
38. O'Brien SJ, Jones DA (1993). 'Function points in SSADM', *SQJ* 2(1), p1-11
39. Jeffrey, DR, Low GC & Barnes M (1993). 'A comparison of function point counting techniques', *IEEE TSE* 19(5), p529-532
40. Rask, R , Laamanen, P & Lyytinen K (1993). 'Simulation and comparison of Albrecht's Function Point and DeMarco's Function Bang metrics in a CASE environment', *IEEE TSE* 19(7), p661-671
41. Kitchenham, B (1993). 'Using function points for software cost estimation : some empirical results', *Proc of 10th CSR Conf*

42. Abdel-Hamid TK, Sengupta K, Ronan D (1993). 'Software project control : an experimental investigation of judgement with fallible information', *IEEE TSE* 19(6), p603-612.
43. Goodman, P (1992). 'Application of cost estimation techniques : industrial perspective', *IST* 34(6), p379-382
44. Fairclough J (1993). 'Graduating from a defined software development process to a managed process - a case history', *Proc SESS93*, IEEE Comp Press
45. Kitchenham, B (1990). *Esprit 90 Conference Proceedings*, Summary of Mermaid Project, p296-314
46. Tamai, T (1992). 'Experiment on coordination with software development teams', *IST* 34, 437-442
47. Bailey JW, Basili VR (1981). 'A meta model for software development resource expenditure', *Proc 5th Int Conf SE*, IEEE Comp Press
48. Tsui, F (1992). 'A software development post mortem summary', *SEJ* 7(4), p277-284
49. Mark II Function Point Analysis Counting Practices Manual Version 1.0, March 1994
50. Kitchenham, B (1987). 'Toward a constructive quality model', *SEJ* 2(4), p105-126
51. Gillies, A (1992). *Software Quality*, Prentice-Hall
52. TickIT. A Guide to Software QMS Construction using EN 29001. Version 2. 1992
53. ISO 9126. Information technology - software evaluation - quality characteristics and guidelines for their use. ISO 1991
54. Kitchenham, B (1989). 'A quantitative approach to monitoring software development', *SEJ* 4(1), p2-13
55. Linkman, S (1990). 'Quantitative monitoring of software development by time-based and intercheckpoint monitoring', *SEJ* 5(1), p43-49