Using Function Points to Find Cost Analogies

Karen Atkinson and Martin Shepperd

School of Computing and Cognition
Bournemouth University
Talbot Campus
Poole, BH12 5BB, England

Abstract

Finding effective techniques for the early estimation of project effort remains an important — and frustratingly elusive — research objective for the software development community. We have conducted an empirical study of 21 real time projects for a major software developer. The study collected a range of counts and measures derived from specification documents, including a derivative of Function Points intended for highly constrained systems. Notwithstanding the fact that the projects were drawn from a comparatively stable environment, traditional approaches for building prediction systems, (for example, regression analysis) failed to yield a useful predictive model. By contrast, estimation based upon the automated search for analogous projects produced more accurate estimates. How much this is a characteristic of this particular dataset and how much these findings might be more generally replicated is uncertain. Nevertheless, these results should act as encouragement for follow up research on a much under utilised estimation technique.

Keywords: Effort prediction, analogy, function points, real time.

1. Background

The concern of this research is to find an effective means to aid software project mangers predict development effort at an early stage in a project. We have conducted an empirical study of 21 real time projects for a major software developer. The study collected a range of counts and measures derived from specification documents, including an adaptation of Function Points [1, 5] by the European Function Point Users' Group (EFPUG)[1] intended for highly constrained systems [3]. Details are given in Table 1.

Abbreviation	Variable	Explanation
proj	project number	project number

[1]The European Function Point Users' Group has recently been renamed the UK Function Point Users' Group (UFPUG) and a distinct EFPUG umbrella group formed.

est1	estimate 1	first project effort estimate (not always available)
est2	estimate 2	second project effort estimate (at detailed specification stage)
act	actual effort	actual effort in person hours
est_dur	estimated duration	estimated duration in weeks
act_dur	actual duration	actual duration in weeks
extra	extra effort	effort for unplanned activities (person hours)
UA_RTFP	unadjusted FPs	unadjusted real time Function Points given as: $IMT + \sum_{i=1}^{i=IMT} IAT_i + OMT + \sum_{i=1}^{i=OMT} OAT_i + ER$
IMT	#input message types	#input message types
IAT	#input attributes	the sum of attributes for each all input messages
IT	IMT+IAT	
OMT	#output message types	output message types
OAT	#output attributes	the sum of attributes for each all output messages
OT	OMT+OAT	
ER	#entities referenced	the number of unique entities referenced by a system
EA	#entity attributes	the sum of all entity attribute counts
ERA	ER + EA	
EA_RTFP	entity attribute realtime FPs	UA-RTFP + EA

Table 1: Project Data Collected

The initial results from our empirical study were somewhat disappointing as we were unable to find an effective method for predicting effort despite the fact that the projects were drawn from a comparatively stable environment. traditional model building approaches, such as regression analysis, failed to yield a useful predictive model. We were unable to find a model that was significantly better than the expert judgement of the project managers involved. A detailed account is given by [4].

Given the lack of success with the traditional methods we have gone on to explore the use of analogy as a means of making a cost estimate. The next step is to find a method to automate the search for analogies. First, we characterise each project in terms of the various specification measures available, for example the number of input and output messages, message and entity size. Next, we use proximity analysis based upon a standardised measure of Euclidean distance in n-dimensional space to find the most similar project or analogy. Finally, having found the analogy, we use the known effort of the analogy as the basis for the estimate for the new project. This approach can be made more robust by using the weighted mean of more than one analogy.

2. Finding Analogies

As the name suggests, estimating by analogy, involves searching for one or more completed projects in similar domains and then using the known amount of effort — modified as appropriate — to form the new estimate. Analogies may be sought at either the total project or the sub-system level. Cowderoy and Jenkins [2] suggest the following steps:

1. Select analogies and rank in order of applicability.
2. Assess similarities and differences.
3. Assess quality of analogy itself e.g. how reliable was record keeping?
4. Consider known special cases e.g. ignore team X as they don't use the SSADM development method
5. Modify the analogy to reflect the current situation

OLD PROJECT (Building Society System)

CMPLX = nominal
DATA = nominal
RELY = high
LANGUAGE = 50% COBOL, 50% C

LENGTH = 4 KDSI

NEW PROJECT (Banking System)

CMPLX = nominal
DATA = nominal
RELY = high
LANGUAGE = 50% 4GL, 50% C
size = 20-40% larger

LOW LENGTH = 4 * 1.2 * (0.38* 100/50)
HIGH LENGTH = 4 * 1.4 * (0.38* 100/50)

Figure 1 Estimation by Analogy

By way of example, Figure 1 shows how size data from an analogous system can be used to generate estimates for the new Banking System. Note that in this case the three COCOMO product cost drivers, plus programming language information, are used to help assess the validity of the analogy. In this case the analogy is not exact, our estimator believes that the new system will be 20-40% larger and 50% will be developed using a 4GL rather than COBOL. Assuming that a conversion factor is known between the two programming languages then we are in a position to produce upper and lower bounds for our estimate of 4.256 and 3.648 KDSI.

Despite the simplicity of the concept, there are significant problems including how to find an analogy in the first place, especially within a large organisation, and how to gauge the representiveness of the analogy once found?

For these reasons we decided to explore mechanisms to automate the search for analogies and to use the dataset as a means of validating the success or otherwise of our approach. The next section describes the details of the different automated search mechanisms and their relative performances.

3. Results

Our approach to automating

To assess how each prediction method performed we jack knifed the dataset by successively witholding one project and using the remaining 20 as a source of analogy. This means that the effort for each project is estimated once using the other 20 observations as input to the prediction model.

The effectiveness of each prediction model is shown by means of the two indicators, mean magnitude of relative error (MMRE) which indicates the average percentage error, irrespective of the direction, and PRED(25) which gives the percentage of projects where the predicted value lies within 25% of actual value.

Prediction Method	**Mean Magnitude of Relative Error (MMRE)**	**Pred (.25)**
Expert judgement	39%	40%
Realtime FPs	99%	10%
Best case analogy	8%	95%
Analogy (2 cases unweighted) IT, OT, ER, EST2	50%	43%
Analogy (best case unweighted) (all variables)	68%	29%

Table 2: Comparative Performance of the Prediction Methods

Table 2 shows how estimating by analogy performs, compared with expert judgement and an algorithmic model using real time FPs and derived by linear regression.

4. Conclusions

Our study has shown, at least for this dataset, that searching for analogies is at least as effective method for software project effort estimation than either expert judgment or traditional algorithmic models derived by regression analysis.

Acknowledgements

The authors wish to thank the organisation that supplied the data which has made this study possible. Regretfully the persons concerned must remain anonymous. We also wish to thank Schlumberger Technologies and Butterworth-Heinemann for their generous contributions towards the project travel expenses.

References

[1] Albrecht, A.J. and J.R. Gaffney, 'Software function, source lines of code, and development effort prediction: a software science validation', *IEEE Trans. on Softw. Eng.,* 9(6), pp639-648, 1983.

[2] Cowderoy, A.J.C. and J.O. Jenkins. 'Cost estimation by analogy as a good management practice', in *Proc. Software Engineering 88.* Liverpool: IEE/BCS, 1988.

[3] EFPUG, Counting Practices for Highly Constrained Systems. Draft Report No. , European Function Point User Group, 1992.

[4] Shepperd, M.J. and R. Turner. 'Real-time Function Points: an industrial validation', in *Proc. European Software Cost Modelling Meeting.* Bristol: ESCOM, 1993.

[5] Symons, C.R., 'Function point analysis: difficulties and improvements', *IEEE Trans. on Softw. Eng.,* 14(1), pp2-11, 1988.