Software project economics: a roadmap

Martin Shepperd
School of Information Systems, Computing & Maths
Brunel University
Uxbridge, West London, UB8 3PH
martin.shepperd@brunel.ac.uk

Abstract

The objective of this paper is to consider research progress in the field of software project economics with a view to identifying important challenges and promising research directions. I argue that this is an important sub-discipline since this will underpin any cost-benefit analysis used to justify the resourcing, or otherwise, of a software project. To accomplish this I conducted a bibliometric analysis of peer reviewed research articles to identify major areas of activity. My results indicate that the primary goal of more accurate cost prediction systems remains largely unachieved. However, there are a number of new and promising avenues of research including: how we can combine results from primary studies, integration of multiple predictions and applying greater emphasis upon the human aspects of prediction tasks. I conclude that the field is likely to remain very challenging due to the people-centric nature of software engineering, since it is in essence a design task. Nevertheless the need for good economic models will grow rather than diminish as software becomes increasingly ubiquitous.

Keywords: *cost models, effort prediction, empirical software engineering, software project management.*

1 Introduction

The economics of software projects has been seen as an important sub-discipline of software engineering for some time. This is because projects are generally embedded in a wider business environment such that the issues of cost, schedule and productivity are of considerable significance. One of the earliest papers to consider cost and productivity aspects of a software project was written by Bennington [8] more than half a century ago. It is striking that many issues remain unchanged, not least that developing large and complex software systems remains a costly and a somewhat unpredictable business; that despite the significant amount of research effort and the many advances project prediction remains a largely unsolved challenge.

Nonetheless, I believe this is still an important topic since software systems are procured, developed, deployed and maintained in order to solve business (business being defined in the broadest sense of the word) problems. Consequently, cost[1]prediction is an essential input into any cost-benefit analysis. Clearly it is also extremely useful when assessing bids in a sub-contracting situation. A closely related, and equally valuable, topic is schedule prediction particularly for business sectors, for example e-commerce, where time to market is the primary consideration.

The scope of this chapter is restricted to quantitative methods and the main unit of analysis is at the micro level i.e. the project rather than business, sector or even nation. Good examples of empirical comparison at the macro-level are by Cusumano and Kemerer [12] and more recently Cusumano *et al.* [13]. This article will not directly address the matter of software development productivity except inasmuch as this forms part of many cost models. For a thorough example of an empirical analysis of productivity see [46].

The remainder of the paper is organised as follows. The next section describes the results of a bibliometric analysis of published project prediction research. This is followed by an assessment of research progress and a discussion of the outstanding challenges. The paper concludes by proposing a research agenda for the community and suggesting some possibly fruitful avenues of enquiry.

2 Review of Past Research

This section aims to give an overall picture of research activity to date in the area of software project cost prediction. A more detailed survey based upon a systematic review[2] of

[1]Note that cost and effort are used interchangeably in the literature since labour is usually the dominant and least predictable component of overall project costs.

[2]A systematic review is, as the name implies, a systematic and repeatable search for *all* relevant literature that satisfies the inclusion criteria de-

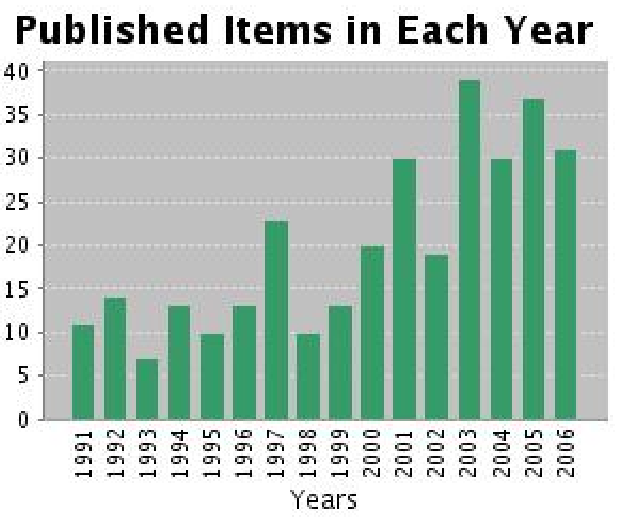

Figure 1. Journal Publications by Year

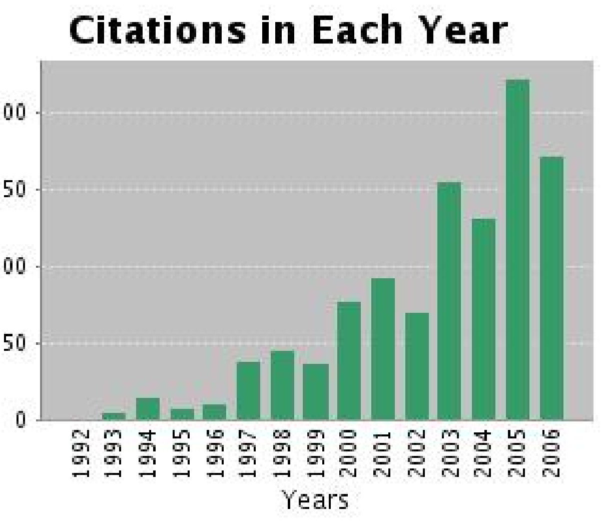

Figure 2. Journal Article Citations by Year

over 300 journal papers may be found in [25].

All the searches this paper were undertaken using the ISI Scientific Citation Index Web of Knowledge (WoK) and were conducted on November 30, 2006. The WoK indexes more than 22,000 journals and 12,000 conferences and has the advantage of allowing a single search query to access items from more than one publisher rather than being restricted to, say Elsevier when using ScienceDirect. The journals are cataloged back to 1970 and the conference proceedings from 1990. WofK supports searching of titles, keywords and abstracts but not full content searching. Thus there is a likelihood that it rather underestimates the body of literature.

In order to consider the research activity in the field of effort or cost prediction the following query was used:

```
(((cost model*) OR (effort
prediction) OR (effort
estimation)) AND (software
project))
```

Note that where several terms are placed in sequence then all terms are required but do not necessarily need to be adjacent.

This search retrieved 320 journal and 333 conference articles. Given that the conference search started from 1990, rather than 1970, this suggests that more material is presented in conferences than in journals and again underestimates the totality of the work since pre-1990 conference articles are not included.

fined in a protocol with the goal of reducing subjectivity and selectivity. These reviews are gaining popularity in software engineering, see [37].

Fig. 1 shows the distribution of journal papers by year of publication commencing from 1991. There is a clear upward trend suggesting increasing levels of activity. Likewise Fig. 2 shows inter-journal citations for the same period with an even more marked increase, although it must be borne in mind the cumulative effect derived from the fact that articles can be cited only after they are published.

Thus it is self evident that the topic is of some significance and has been attracting the attention of researchers over a number of years. Next we examine in more detail the state and breadth of this research activity.

3 Progress to Date

The next step is to consider the types of research that have been conducted under the general banner of cost prediction. Using the classification and analysis given in [25], Fig. 3 shows the breakdown of papers under the various headings together with a count of publications for each class. The dominant category (61%[3]) is research that either introduces or evaluates an estimation technique (Est meth). Another significant area is covered by papers that consider how to measure the size of a software system (Size) at 20%. Clearly this is an important topic since the majority of algorithmic cost prediction systems are based upon some function of size. Unfortunately how to measure software size is a non-trivial problem. Research papers that address organisational issues (Org) of effective project prediction make up 16% of output in the area. Measuring or

[3]Note that the percentages do not sum to 100 since some papers are classified into more than one category.

Title	Authors	Source	Year	Total Citations	Average per Year
Resource-constrained project scheduling: ...	Brucker et al.	EJOR	1999	120	13.33
Estimating software project effort using analogies	Shepperd and Schofield	TSE	1997	54	6
A review of studies on expert estimation ...	Jørgensen	JSS	2004	15	5
A cross-cultural study on escalation of commitment ...	Keil et al.	MISQ	2000	28	4
On the problem of the software cost function	Dolado	IST	2001	19	3.17
Reliability of function point measurement ...	Kemerer	CACM	1993	43	3.07
A controlled experiment ...	Myrtveit and Stensrud	TSE	1999	24	3
Impact of effort estimates on software project work	Jørgensen and Sjøberg	IST	2001	15	3
Modeling development effort in object-oriented systems ...	Briand and Wüst	TSE	2001	15	2.5
An empirical study of maintenance and development ...	Kitchenham et al.	JSS	2002	10	2.5

Table 1. Most cited journal papers

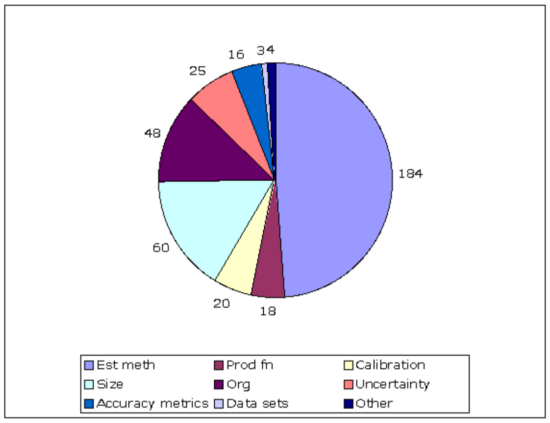

Figure 3. Proportions of Journal Publications by Classification

coping with the uncertainty inherent in a prediction is the next largest topic (8%). Interestingly this growing research topic was not much represented before about 1995. Other topics are model calibration (Calibration), the study of production functions, a concept from the field of economics (Prod fn), how to assess the accuracy of prediction systems (Accuracy metrics), characteristics of statistical analyses of various data sets that are typically used to evaluate prediction systems (Data sets) and finally other topics that do not fall into any of the above categories.

We now consider each of these topics in more detail. Given the substantial number of research papers — the searches in the previous section retrieved in excess of 600 articles — the following analysis is highly selective. Papers are selected either due to their perceived significance in that they have a citation count (normalised by years since publication) or in order to provide wider coverage of the topic.

3.1 Estimation techniques

The dominant area of activity is proposing new estimation techniques or evaluating extant techniques sometimes in order to provide a benchmark comparison with a new technique.

The work essentially falls into one of three techniques:

- algorithmic or parametric models
- induced prediction systems (via some kind of machine learning method)
- human centric techniques (usually referred to as expert judgement)

3.1.1 Algorithmic or parametric models

Some of the earliest models could be described as algorithmic and generally take the form of a function that relates size of the task to the degree of effort (cost) to perform it. Thus we have:

$$E = a\,(S)^b \qquad (1)$$

where E is project effort, S size, a is a coefficient to represent productivity (typically determined by combining a series of drivers that are intended to represent the development environment) and b is a returns to scale coefficient.

Probably the best known example of this form of model is COCOMO due to Boehm [10] which has a number of interesting features. It comprises a series of three models: basic, intermediate and detailed. The models assume two sets of relationships: between size (in terms of lines of code (LOC)) and effort similar to Eqn. 1 and also between effort and elapsed time T given by Eqn. 2. The latter implies that a certain size of project in effort E implies an optimal duration, T. In other words units of effort and time are not linearly interchangeable.

$$T = c\,(E)^d \tag{2}$$

The model coefficients are dependant upon the type of project and Boehm identifies three different classes of project (organic, semi-detached and embedded). The approach is widely cited and there are many implementations since the model and the data set for which it was originally developed are in the public domain [9]. Subsequently the model has evolved into COCOMO II [11] and a specialised version, COCOTS, has been produced for commercial off the shelf (COTS) component-based projects [3]. The underlying philosophy of these models is that software development environments vary significantly and that this can be, or even needs to be, accommodated by sophisticated models with a high degree of parameterisation. Of course the down side is that such models are also more difficult to calibrate.

A somewhat contrasting approach is for individual development environments to develop their own local models using simple statistical techniques such as regression analysis. Typically a small data set of local projects is analysed to find the best fitting model usually of the form:

$$E = \beta_0 + \beta_1 X_1 +, \cdots, +\beta_n X_n \tag{3}$$

where β_0 is the intercept and could be interpreted as fixed project costs, $\beta_{(1,\ldots,n)}$ are the model coefficients and $X_{(1,\ldots,n)}$ are the independent variables that are entered into the model often chosen by some stepwise procedure. The point is that the model derived will only have local significance and there is no expectation that the model can be (or should be) ported to new environments and achieve good prediction results.

More recently there has been some exploration of robust regression techniques the advantage being that this can reduce the problem of over fitting to a few influential data points, see for example [50, 52]. This can be useful where the data are skewed as frequently is the case for software project data sets.

Another application of regression modelling given its comparative simplicity, is as a benchmark when evaluating other more sophisticated techniques, see for example [16, 43]. The rationale is that if the more sophisticated technique cannot be shown to be significantly more effective than regression modelling then it is not be worth the additional analysis effort.

3.1.2 Induced prediction systems

In recent years there has been considerable interest in applying machine learning techniques to induce a prediction system from training data. The basic idea is that there is no requirement to specify the form or structure of the prediction system *a priori*, rather it is discovered or learnt from training cases or examples. The quality of the resultant prediction system is then assessed by testing it against unseen cases.

There are a variety of machine learning methods. These include: artificial neural networks (ANNs), rule induction algorithms, case based reasoning (CBR), hybrid approaches such as neuro-fuzzy methods and multiple learners.

The general approach has been to use a software project cost or effort data set and split it[4] into training and validation subsets. The accuracy of the prediction system is then usually assumed to be an average (or some other statistic) of the individual predictions from the validation subset.

The machine learning approach for cost prediction that has been most explored is that of CBR, in part because the idea of formalising the process of predicting by analogy is attractive. The idea is that history repeats itself but not exactly [54]. Hence we need a means of measuring similarity between two projects (cases) when it is possible that no single characteristic (feature) matches exactly. This is usually done by plotting each case in p-dimensional feature space where each of the p features is standardised (i.e. it has equal influence). Then similarity can be defined as a Euclidean distance[5].

A challenge for all the machine learning techniques — and indeed not restricted to software project prediction — is what is known as the feature subset selection problem. In practice most data sets contain features that are included for a range of reasons such as being easy to collect, relevant to a different problem and so forth. Thus some or even many of the features are either irrelevant or harmful to a particular prediction goal. In practice choosing the right subset of features to use for a prediction system turns out to be very influential upon the quality of the resulting prediction. Unfortunately it is also very difficult. Formally it is an example of an NP-hard search problem, however it is amenable to the use of heuristic search algorithms [4]. Kirsopp *et al.* [28, 29] have found that even simple search algorithms such as greedy search or hill climbers are able to yield substantially improved predictions when combined with a CBR system. An interesting result is that for each data set studied, there was a large reduction in the number of features, typically 60-80%.

Several variants of the main machine learning approaches have been studied using many data sets and differing validation techniques [44]. The upshot is that there does not appear to be a particularly consistent pattern although some studies have sought to systematically combine

[4]The process of splitting the data set into training and validation subsets turns out to have a significant impact upon the subsequent validation process particularly when the data is heterogeneous. Typically this either done randomly or by an n-fold method. Failure to use a sufficient number of samples can result in misleading results [28].

[5]The distance metric must be extended to handle non-continuous features such as weak orders and categorical types.

and compare results [45]. However I discuss this topic in more detail in Section 4.

3.1.3 Human centric techniques

Whilst surveys have been consistent in reporting expert judgement as the most common prediction technique in the literature this has been a somewhat neglected area in terms of research activity.

Hughes [19] conducted a study of estimation practice at Ericsson in the mid 1990s and found considerable variation. For example, practitioners reported a range of effort values to make the prediction from 5 minutes to 4 weeks. Thus it would seem that what is labelled 'expert judgement' is in reality a family of techniques united by the fact that no formal model is employed. Another finding was the lack of feedback provided to estimators thus significantly reducing the learning opportunities.

Another topic that has generated some interest although less actual empirical research is the use of groups of experts to generate software project predictions. Boehm [9] proposed the customisation of a general purpose group decision making technique for software prediction and named the approach wideband Delphi. Passing and Shepperd [51] studied groups of 4 postgraduate students given prediction tasks. The overall accuracy of group predictions was found to be significantly better than those of individuals. However, one group was dominated by an individual, who was able to influence the remainder of the group such that they finished up with a poorer prediction than as individuals.

3.2 Software size measurement

From the foregoing discussion of the algorithmic models (see for example COCOMO - Eqn. 1) it can be seen that the size of software is a major input. The challenge is how might software size best be measured? Traditionally size was measured as LOC however there are a number of drawbacks. First, LOC is not available until after coding is complete, i.e. late on in the development process. Second, it is difficult to make comparisons between different programming languages that may take varying numbers of statements to perform a given function.

Consequently an alternative approach to software size was developed known as function points (FPs). The pioneering work was done by Albrecht [5] who proposed that it would be more useful to measure the amount of functionality delivered by a particular software system. Effectively function points are the weighted sum of counts of different function types such as processing a system input, query, etc. Some variants then propose an adjustment to the raw FP count by taking into account various non-functional requirements such as whether there are multiple sites or that a high transaction rate is required. From this basic idea further variants have been proposed including Mk II function points which is more closely based on a entity-relationship models [57] and object points [58]. FPs cannot be used to directly predict effort, however, they are often used as a dependent variable when constructing a regression model as per Eqn. 3.

The other major application of FPs is for benchmarking particularly software productivity where there is a need to obtain some measure of output. There are several well known benchmarking data sets based upon the use of FPs including that of the ISBSG [40] and the so-called Finnish data set [46].

Despite their relatively widespread adoption by the software industry, FPs have also been subject to some criticism. Probably the biggest area of concern is the weighted combination of counts of individual function types since it has been claimed that this is equivalent to 'adding together apples and pears'. A study by Kitchenham and Kansala [33] found significant correlations between the individual counts indicating that the counts are not orthogonal. By using multiple regression they were able to obtain new weights and identify redundant aspects of the FP formulation that led to improved predictions. Another cause of some concern has been the subjectivity inherent in counting function points typically from requirements or design documents. An investigation by Low and Jeffery [41] found substantial variation between the FP analysts on occasions in excess of 30%, although a separate study by Kemerer [27] found rather smaller differences however these were between pairs of FP analysts. Lastly, adjustment factors have not generally been found to be useful, see for example [21].

3.3 Organisational issues

Some researchers have adopted a broader perspective than the individual project and considered the relationship (in both directions) between estimation at the project level and the organisation. Abdel-Hamid [1] has pioneered the use of simulation to explore the dynamics of software project behaviour. Although such models and simulations are interesting vehicles to explore the complex interactions and feedback behaviour of software development organisations a problem is such models are extremely difficult to validate.

Other research such as that by Lederer and Prasad [39] has endeavoured to find relationships between organisational factors such as the degree of project manager control and the accuracy of software project estimates. Their results were quite surprising. No relationship could be found between different types of technique and estimation accuracy (e.g. use of formal / algorithmic techniques had no noticeable impact upon reducing estimation errors) other than informal techniques were associated with larger errors. The only management or conextural factor that improved matters was making estimators responsible and providing feedback. (It is possible also that if staff are being assessed in terms of their predictions these may become self fulfilling.)

3.4 Uncertainty

Rather surprisingly, given that an estimate is by definition a probabilistic statement, uncertainty aspects of prediction have not formed a major area of research, or at least not until relatively recently. Indeed most work assessing and comparing different prediction systems has treated a prediction as being a single point value.

Some interesting work that has adopted an organisational perspective to deal with the uncertainty inherent in prediction has explored the idea of many software projects being viewed as a portfolio, see Kitchenham and Linkman [34]. They argue that there are four sources of prediction error: (i) model error (ii) measurement error (iii) assumption error and (iv) scope error. It is important to avoid double counting e.g. if the error is included in the model it shouldn't be included in the risk analysis. They also argue that model error (i.e. the prediction is a probabilistic output will be assymetric since effort cannot be negative) should be dealt with by a distribution such as the gamma distribution. As a consequence estimators should use the mean and not the median or mode, otherwise the overestimates will be less than the underestimates. Finally they argue the only way to manage risk is across a portfolio of software projects.

In a survey of expert estimation of software development effort research Jørgensen [23] identifies the need to assess the uncertainty of the estimate. Elsewhere [22] he suggests and evaluates a simple approach to quantifying the level of uncertainty by providing $< c, lower, upper >$ where c is the level of confidence expressed as a percentage that the true value lies within the range specified by $lower, upper$. Studies, for instance [24], have consistently indicated that estimators are over confident regarding their prediction accuracy. Clearly this is an important but somewhat under-investigated topic.

3.5 Calibration

Some research has addressed the question of whether the performance of general purpose algorithmic models such as COCOMO might be improved by calibrating the coefficients to the local environment, see for example Gulezian [17]. Jeffery and Low [20] also studied various algorithmic models including COCOMO (again), SLIM, ESTIMACS and Checkpoint. They concluded that calibration was 'essential' if such models were to be used effectively outside of the environments in which they were developed. This is an interesting conclusion since such models were intended to be general purpose through their many drivers and parameters. Note also that the more complex the prediction system i.e. the more parameters to be estimated the more data required. This can be problematic since there can often be a shortage of relevant, local data.

Data set	Count	Mean KSLOC	b
Yourdon	17	34	0.72
Kemerer	17	220	0.79
Walston	60	20	0.91
Behrens	22	n.a.	0.94
Bailey	19	29	0.95
Belady	33	92	1.06
Wingfield	15	180	1.06
COCOMO	63	67	1.11
Albrecht	24	66	1.49

Table 2. Economies / diseconomies of scale by empirical study (Adapted from [6])

3.6 Production functions

A topic that has generated considerable discussion is whether software development exhibits economies, diseconomies, both or fixed returns to scale [7, 36]. A presumption made by many researchers and embodied in many models such as COCOMO [10] is that of diseconomies, in other words, if we assume some production function of the form $E = a\ (S)^b$ where E is project effort, S size, then b is greater than one (see Eqn.1). Note that the values a and b can be derived empirically for this production function by building a linear regression model using the natural log transformation of the data (i.e. $ln(E) = b(ln(S))$) and then re-transforming the data back to its original scale.

Table 2 shows in order of increasing value, the economy of scale coefficient b where $b < 1$ implies economies of scale, $b = 1$ linear or fixed returns and $b > 1$ diseconomies of scale as proposed by models such as COCOMO [9]. It can be seen that there exists considerable diversity which cannot easily be explained in terms of project size. Moreover as Kitchenham [36] has argued, it is necessary to show that the production function is significantly non-linear, i.e. that the 95% confidence limits for b do not contain unity. For this reason she has argued that for most practical purposes one can assume constant returns to scale. This is despite the commonly held view that larger software projects are less productive than smaller projects.

The lack of evidence for more complex production functions is also supported by Dolado [15] who used a genetic programming algorithm to search for functions using a very rich set of operators not limited to exponentiation. He concluded that there were not significant deviations from a linear model.

3.7 Data sets and other topics

Until recently the topic of what data are used to evaluate various competing prediction systems has scarcely been considered. An important initiative by Menzies et al. [53] has been to promote a systematic collection of properly documented and catalogued data. This has been motivated by various concerns not least the difficulty of replicating other researchers' work when the data are not made publicly available.

Mair et al. [44] conducted an analysis of 50 journal papers that used data sets to evaluate cost prediction systems. They found substantial diversity between the data sets employed (in terms of number of projects and/or features, age, homogeneity and so on). Moreover the choice of data set employed seemed opportunistic yet highly influential upon the results. The systematic review of more than 300 journal papers by Jørgensen and Shepperd [25] reached a similar conclusion.

4 Challenges for the Future

From the foregoing discussion I will try to identify a research agenda for the future and then speculate about those avenues that I consider are likely to be the most worthwhile or promising.

Even from this short review it is clear that there are a multiplicity of approaches to building prediction systems for software project costs. This is perhaps unsurprising given that no one approach seems to dominate, that is no one approach is consistently more accurate.

These different prediction approaches have given rise to an increasing number of empirical studies commencing from pioneering work such as [31, 26] where the goal has been to validate and compare approaches using different project data sets. Many of these studies have been independent of the prediction system proposers which has clear benefits in terms of reduction of (unintended) researcher bias. In addition, an advantage of using different data sets is that we hope to better sample the rather ill-defined population of software projects and thereby gain a clearer idea as to how the prediction approaches might generalise. Thus our goal is to answer the question: which prediction system should a practitioner use and in what circumstances?

As stated there has been a substantial increase in the number of empirical studies of comparing various competing project cost prediction systems (see Fig.3). Whilst this offers a useful opportunity to build a picture there are, unfortunately, a number of difficulties. Consider the following example.

Group	Count
Support for Regression	7
Inconclusive	4
Support for CBR	9

Table 3. Summary of Study Support for Regression-Based and CBR Prediction

4.1 An example analysis of multiple empirical studies

As an example of the inconsistent nature of empirical results relating to the evaluation of prediction systems we consider regression models and CBR approaches. A total of 20 studies have been identified through a systematic review [45] of the journal and conference literature. No judgement is made concerning the quality of studies other than that all 20 studies have undergone peer review. Table 3 shows an almost even split between studies based upon reported accuracy levels with 35% favouring regression models, 45% favouring CBR approaches and the remaining 20% undecided. This is problematic since it is unclear what the overall body of evidence is indicating, and what advice should be given to practitioners.

The problem we need to address is: why are the results inconsistent? One might expect differing results when models are generated from different data sets, however, in several cases results were inconsistent even when utilising the same data set and the same prediction techniques. This is not necessarily due to carelessness but may be an artefact of the stochastic nature of some validation techniques and the fine tuning of prediction approaches. To try to quantify the heterogeneity between primary studies various statistics have been proposed including I^2 which is an adaption of Cochran's Q to accommodate meta-analyses based upon differing numbers of primary studies [18]. Unfortunately this is not an option when, as is generally the case for cost model validation studies, different response variables are used.

4.2 Difficulties in evaluating prediction systems

One of the reasons for the equivocal results in the previous example is that there are variations in:

- the treatment of data, dealing with outliers and missing values.
- the choice of response variable(s) (accuracy indicators) when comparing prediction performance. The study by Kitchenham *et al.* [35] identified no less than 12

different accuracy indicators that have routinely been used. In addition to the problem noted above of being unable to compute a measure of inconsistency within the meta-analysis, these tend to capture different properties which can lead to rank reversal problems i.e. accuracy indicator A prefers PS1 to PS2, whilst indicator B prefers PS2 to PS1.

- validation strategy and the use of holdout data. These fall into four general classes, namely, model fitting, the jackknife, n-fold validation and cross validation. Making comparisons between strategies is not easy since some strategies are more conservative than others.

- the expertise of research teams to use sophisticated prediction techniques. This problem has also been noted by the machine learning community in an effort to explain their inconsistent results [49].

Further, information necessary for meaningful comparison may not be reported. Therefore, I believe it is a matter of some urgency that we as a research community define and agree reporting protocols and methods for comparison. Moreover this is a more important task than performing yet more empirical studies given our present difficulties in interpreting and combining the results from such work.

There are also issues relating to the use of data sets that are not in the public domain thus inhibiting re-analysis and replication. Finally the choice of data set(s) for the analysis is often *ad hoc* or opportunistic. Thus as [44, 25] note we have the curious situation of certain, very old data sets that are easily available being substantially over-represented in terms of this kind of analysis. That this matters is also beyond dispute, since simulation studies such as [55] have shown the significant relationship between data set characteristics (e.g. number of cases and features, feature scalar types, missingness patterns, distributions, noise, etc.) and the relative performance of different prediction systems. In other words prediction techniques are often highly sensitive to specific data set characteristics.

4.3 What data should we use?

Data quality is clearly an important topic since inaccurate data can mislead analysis and hence any conclusions derived from it. Surprisingly it has not been the topic of much research in the context of cost estimation models. The primary focus to date has been dealing with missing values and in particular using imputation techniques to replace the missing value with one that is artificially generated [56]. However, there are many other data quality problems including that of noisy observations. Unless these contain implausible values (for example a 12 year old being the parent of 10 children), it is not possible to identify noisy data items with certainty. This is akin to the testing problem for software, i.e. it is not possible to demonstrate the absence of defects. One possibility, building upon the testing metaphor, is to explore capture-recapture techniques to estimate the number of noisy data items. Simulation is another alternative. Here the true model can be known so that the level and type of noisy items can also be known.

Nevertheless data quality is an important topic [14]. Essentially we need to answer two questions. First can we objectively assess the quality of a data set? The potential consequence of asking this question is we may stop using some of the publicly available (and therefore widely used by researchers) data sets. The second question is can we identify problematic data items and carry out some editing procedure?

Another challenge, arising from the difficulties in collecting high quality data is that of amassing sufficient relevant data from which to build or train a prediction system. This is because data may rapidly become obsolete, for instance due to technology change, and furthermore project completion is a relatively infrequent event. A systematic review [38] has endeavoured to answer the question what evidence is there that cross-company estimation models are at least as good as within-company estimation models for predicting effort for software projects? Unfortunately, as per the previous systematic review, the answer is somewhat equivocal and there is a lack of a consistent result. Again such analysis is plagued by different research methods, different data sets and different prediction techniques. One observation is that this kind of analysis is making assumptions about heterogeneity within a data set and across data sets. It would be useful to make such notions more rigorous or even quantifiable.

4.4 Combining predictors

Another potentially fruitful avenue is that of using multiple predictors which of course begs the question how should they be combined? Whilst the idea of using multiple classifiers or prediction systems is not uncommon in machine learning this has not been widely deployed in cost prediction. A study by MacDonell and Shepperd [42] studied three different prediction techniques (regression, CBR and expert judgement). The results indicated that no single technique dominated, however it is was not generally clear *a priori* which technique should be selected for a given software project. Meta-level learning (using a rule induction algorithm to determine which predictor to use in which circumstance) was also unsuccessful. Nor did the various combination strategies (e.g. use the mean) improve matters. Thus we are left with the frustrating situation that we know that it is theoretically better to use multiple predictors but we don't practically know how to select them.

Menzies et al. [48] also consider the problem of determining which prediction system to use and note the problem of high levels of variance between prediction system comparisons, in other words the lack of reliability between and even within empirical studies. For this reason they propose a machine learning based environment termed COSEEKMO to automate the search for the best prediction system. The main limiting factor for this work is that it requires data to be collected to conform to the COCOMO data model. Neveertheless the idea of using machine learning (sometimes known as meta-level learning) to discover relationships between predictor and data set is attractive.

Another area in which more research is needed is the less formal combining of prediction systems with expert judgement. In the past the implication — if unvoiced — including from some of the author's work has been that formal prediction systems will one day replace experts. However this is extremely unlikely not least because software project cost predictions are infrequent but very high value decisions. Hence we need more work to consider how formal models might support and assist experts rather than replace them.

5 Summary

In this paper I have indicated that one of the major challenges for empirical software engineering researchers interested in project prediction is how to effectively combine results. Presently this is not feasible due to the diversity of evaluation methods employed. The problem is further exacerbated by the *ad hoc* manner in which data sets are selected and very weak notions we have of population and how it is being sampled.Moreover, I have shown for two prediction techniques that have undergone considerable investigation (regression and analogy) the results are almost evenly split.

This implies five issues for the research community. First, we need to consider more carefully what population of software projects is of interest and then whether the data sets we use represent an unbiased sampling of this population? Second, we need to have better evaluation and reporting protocols to expedite comparison and combination of results. Third, we should consider what data ought we use when conducting empirical validations. Fourth, as has been stated in the past [45], researchers should ask questions such as 'when might it be better to use technique A rather than B?', as opposed to 'is technique A better than B?'. Finally, we might consider how multiple prediction systems rather than one might be deployed and in particular whether we should modify the question into one of how formal techniques can best assist the human experts.

Acknowledgments

This work was funded by the UK Engineering and Physical Sciences Research Council under grant GR/S45119.

References

[1] Abdel-Hamid, T. and Madnick, S. "Impact of schedule estimation on software project behaviour", *IEEE Software*, 3(4), pp70-75, 1986.

[2] Abran, A. and Robillard, P. "Function points analysis: An empirical study of its measurement processes," *IEEE Trans. on Softw. Eng.*, vol. 22, pp. 895-910, 1996.

[3] Abts, C., Boehm, B. and Clark, B. "COCOTS: A COTS software integration lifecycle cost model - model overview and preliminary data collection findings". Technical Report No. 00 03 31, University of Southern California, 2000.

[4] Aha, D.W. and R.L. Bankert, "A comparative evaluation of sequential feature selection algorithms", in *Artificial Intelligence and Statistics V.*, Fisher, D. and Lenz, J.-H., Editors, Springer-Verlag: New York, 1996.

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

[6] Banker, R. and Kemerer, C. "Scale economies in new software development," IEEE Transactions on Software Engineering, 15(10), pp1199-1204, 1989.

[7] R. D. Banker, H. Chang, and C. F. Kemerer, "Evidence on economies of scale in software development," *Info. & Softw. Technol.*, 36, pp275-282, 1994.

[8] Benington, H.D. "Production of large computer programs", *Proc. Symp. on Advanced Computer Programs for Digital Computers*. Washington, D.C.: Office of Naval Research.

[9] Boehm, B. *Software engineering economics*, Prentice-Hall, Englewood Cliffs, 1981.

[10] Boehm, B. "Software engineering economics," *IEEE Trans. on Softw. Eng.*, vol. 10, pp. 4-21, 1984.

[11] Boehm, B. *Software cost estimation with Cocomo II*, Prentice Hall Upper Saddle River, NJ, 2000.

[12] Cusumano, M. and Kemerer, C. "A quantitative analysis of U.S. and Japanese and performance in software development.", *Management Science*, 6(11), pp1384-1406, 1990.

[13] Cusumano, M., et al., "Software development worldwide: The state of the practice", I*EEE Software*, 20(6) pp28-34, 2003.

[14] De Veaux, R. and Hand, D. "How to Lie with Bad Data," *Statist. Sci.*, vol. 20, pp. 231-238, 2005.

[15] Dolado, J. "On the problem of the software cost function". *Information & Software Technology*, 43(1) pp61-72, 2001.

[16] Finnie, G. Wittig, G. and Desharnais, J.-M. "A comparison of software effort estimation techniques using function points with neural networks, case based reasoning and regression models", *J. of Systems & Software*, 39: pp281-289, 1997.

[17] Gulezian, R., "Reformulating and calibrating COCOMO", *J. of Systems & Software*, 16: pp235-242, 1991.

[18] Higgins, J., et al., "Measuring inconsistency in meta-analysis", *British Medical Journal*, 327, pp557-560, 2003.

[19] Hughes, R. "Expert judgement as an estimating method" I*nformation & Software Technology*, 38(2) pp67-75, 1996.

[20] Jeffery, R. and Low, G. "Calibrating estimation tools for software development," *Software Engineering Journal*, 5(4), pp215-221, 1990.

[21] Jeffery, R. Low, G. and Barnes, M. "A comparison of function point counting techniques," *IEEE Trans. on Softw. Eng.*, 19(5), pp529-532, 1993.

[22] Jørgensen, M. and Sjøberg, D. "An effort prediction interval approach based on the empirical distribution of previous estimation accuracy". *Information & Software Technology*, 45(3) pp123-136, 2003.

[23] Jørgensen, M. "A review of studies on expert estimation of software development effort," *Journal of Systems & Software*, 70(1-2), pp37-60, 2004.

[24] Jørgensen, M., Teigen, K. and Moløkken, K."Better sure than safe? Overconfidence in judgment based software development effort prediction intervals," *J. of Systems & Software*, 70, pp79-93, 2004.

[25] Jørgensen, M. and Shepperd, M. "A Systematic Review of Software Development Cost Estimation Studies", *IEEE Transactions on Software Engineering*, 33(1): pp33-53, 2007.

[26] Kemerer, C. "An empirical validation of software cost estimation models," *Communications of the ACM*, 30, pp416-429, 1987.

[27] Kemerer, C. "Reliability of Function Point Measurements: A Field Experiment", Communications of the ACM, 36(2) pp??, 1993.

[28] Kirsopp, C., M. Shepperd and J. Hart, "Search heuristics, case-based reasoning and software project effort prediction", *Genetic and Evolutionary Computation Conference*, New York, USA, July 9-13, 2002.

[29] Kirsopp,C. and Shepperd, M. "Case and Feature Subset Selection in Case-Based Software Project Effort Prediction", *Research and Development in Intelligent Systems XIX*, Springer-Verlag, 2002.

[30] Kirsopp,C. and Shepperd, M. "Making inferences with small numbers of training sets". *IEE Proceedings - Software*, 149(5), 2002.

[31] Kitchenham, B. and Taylor, N. "Software cost models", *ICL Technical Journal*, 4(3), pp73-102, 1984.

[32] Kitchenham, B. "Empirical studies of assumptions that underlie software cost estimation models," *Info. & Softw. Technol.*, vol. 34, pp. 211-218, 1992.

[33] Kitchenham, B. and Kansala, K. "Inter-item correlations among function points". *Proc. of 1st Intl. Symposium on Software Metrics*. Baltimore, MD: IEEE Computer Society Press, 1993.

[34] Kitchenham, B. and Linkman, S. "Estimates, uncertainty and risk", *IEEE Software*, 14(3) pp69-74, 1997.

[35] Kitchenham, B. MacDonell, S. Pickard, L. and Shepperd, M. "What accuracy statistics really measure," *IEE Proceedings - Software Engineering*, 48, pp81-85, 2001.

[36] Kitchenham, B. "The question of scale economies in software - why cannot researchers agree?" *Info. & Softw. Technol.*, 44, pp13-24, 2002.

[37] Kitchenham, B. Dybå, T. and Jørgensen, M. "Evidence-based Software Engineering", *Proc. of 27th IEEE Intl. Softw. Eng. Conf. (ICSE 2004)*, Edinburgh: IEEE Computer Society, 2004.

[38] Kitchenham, B., Mendes, E. and Travassos, G. “A systematic review of cross- vs. within-company cost estimation studies, *Proc. 10th Intl Conf Empirical Assessment in Softw. Eng.* Keele University, UK, 2006.

[39] Lederer, A. and Prasad, A. “Software management and cost estimating error”, *J. of Systems & Software*, 50(1) pp33-42, 2000.

[40] Lokan, C. Wright, T. Hill, P. and Stringer, M. “Organizational benchmarking using the ISBSG Data Repository,” *IEEE Software*, 18(5), pp26-32, 2001.

[41] Low, G. and Jeffery, R. “Function points in the estimation and evaluation of the software process,” *IEEE Transactions on Software Engineering*, 16(1), pp64-71, 1990.

[42] MacDonell, S. and Shepperd, M. “Combining Techniques to Optimize Effort Predictions in Software Project Management,” *J. of Systems & Software*, 66, pp91-98, 2003.

[43] Mair, C., et al., “An investigation of machine learning based prediction systems”, *J. of Systems & Software*, 53(1) pp23-29, 2000.

[44] Mair, C. Shepperd, M. and Jørgensen, M. “An Analysis of Data Sets Used to Train and Validate Cost Prediction Systems”, *Proc. PROMISE 2005*, ACM Computer Press, St Louis, MI, May 16, 2005.

[45] Mair, C. and Shepperd, M. “The Consistency of Empirical Comparisons of Regression and Analogy-based Software Project Cost Prediction”. *Proc. of 4th Intl. Symp. on Empirical Softw. Eng. (ISESE).* Noosa Heads, Australia: IEEE Computer Society, 2005.

[46] Maxwell, K. and Forselius, P. “Benchmarking software development productivity”, IEEE Software, 17(1) pp80-88, 2000.

[47] Maxwell, K. Van Wassenhove, L. and Dutta, S. “Performance evaluation of general and company specific models in software development effort estimation,” *Management Science*, vol. 45, pp. 787-803, 1999.

[48] Menzies, T. Chen, Z. Hihn, J. and Lum, K. “Selecting Best Practices for Effort Estimation,” *IEEE Transactions on Software Engineering*, 32(11), pp883-895, 2006.

[49] Michie, D. Spiegelhalter, D. and Taylor, C. *Machine learning, neural and statistical classification*, Ellis Horwood: Chichester, Sussex, UK, 1994.

[50] Miyazaki, Y., et al., “Robust Regression for Developing Software Estimation Models”, *J. of Systems & Software*, 27(1) pp3-16, 1994.

[51] Passing, U. and Shepperd, M. “An Experiment on Software Project Size and Effort Estimation” in *2nd Intl. Symp. on Emp. Softw. Eng.* Rome: IEEE Computer Society, 2003.

[52] Sentas, P., et al., “Software productivity and effort prediction with ordinal regression”, *Information & Software Technology*, 47(1) pp17-29, 2005.

[53] Sayyad Shirabad, J. and Menzies, T. “The PROMISE Repository of Software Engineering Databases”. School of Information Technology and Engineering, University of Ottawa, Canada. Available: http://promise.site.uottawa.ca/SERepository [Last accessed 21 February, 2005].

[54] Shepperd, M. and Schofield, C. “Estimating software project effort using analogies”, *IEEE Trans. on Softw. Eng.* 23(11) pp736-743, 1997.

[55] Shepperd, M. and Kadoda, G. “Comparing Software Prediction Techniques Using Simulation,” *IEEE Trans. on Softw. Eng.*, 27(11), pp987-998, 2001.

[56] Strike, K. El Emam, K. and Madhavji, N. “Software cost estimation with incomplete data”, I*EEE Transactions on Software Engineering*, 27(10) pp890-908, 2001.

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

[58] Teologlou, G. “Measuring object-oriented software with predictive object points”. *Proc. of 10th European Software Control & Metrics Conference*, Herstmonceux, England: Shaker Publishing, 1999.