Full citation: MacDonell, S.G., & Shepperd, M. (2010) Data accumulation and software effort prediction, in Proceedings of the 4th International Symposium on Empirical Software Engineering and Measurement. Bolzano-Bozen, Italy, ACM Press.
doi: 10.1145/1852786.1852828

Data Accumulation and Software Effort Prediction

Stephen G. MacDonell

SERL, Auckland University of Technology
Private Bag 92006
Auckland 1142, New Zealand
stephen.macdonell@aut.ac.nz

Martin J. Shepperd

School of Comp., Info Systems and Mathematics
Brunel University
Uxbridge, Middlesex UB8 3PH, UK
martin.shepperd@brunel.ac.uk

Abstract

BACKGROUND: In reality project managers are constrained by the incremental nature of data collection. Specifically, project observations are accumulated one project at a time. Likewise within-project data are accumulated one stage or phase at a time. However, empirical researchers have given limited attention to this perspective. PROBLEM: Consequently, our analyses may be biased. On the one hand, our predictions may be optimistic due to the availability of the entire data set, but on the other hand pessimistic due to the failure to capitalize upon the temporal nature of the data. Our goals are (i) to explore the impact of ignoring time when building cost prediction models and (ii) to show the benefits of re-estimating using completed phase data during a project. METHOD: Using a small industrial data set of sixteen software projects from a single organization we compare predictive models developed using a time-aware approach with a more traditional leave-one-out analysis. We then investigate the impact of using requirements, design and implementation phase data on estimating subsequent phase effort. RESULTS: First, we find that failure to take the temporal nature of data into account leads to unreliable estimates of their predictive efficacy. Second, for this organization, prior-phase effort data could be used to improve the management of subsequent process tasks. CONCLUSION: We should collect time-related data and use it in our analyses. Failure to do so may lead to incorrect conclusions being drawn, and may also inhibit industrial take up of our research work.

Keywords: Software project management, empirical analysis, time series, phase data, effort prediction.

1. INTRODUCTION

Software engineering researchers and practitioners continue to be concerned with the need to obtain early, accurate measures and estimates of project size, effort, duration and cost. These are useful in many aspects of software project management — they inform tenders for contract development, they underpin the planning of development, deployment and integration activities, and they form benchmarks for the ongoing control of these activities as they occur. As these activities all have an impact on a project's budget in terms of optimum use of resources and likely return, accuracy in measurement and forecasting is highly sought after.

Most prior work considering effort and schedule prediction has focused on the development and evaluation of models to be used for entire projects and which are derived from information available at the beginning of the project to be estimated. However, in this paper we empirically investigate the utility of prediction (or re-estimation) *during* projects to enable managers to more effectively forecast the effort required for remaining project phases. In addition, we take into account the time sequence in which data becomes available from completed projects to reflect how data is accumulated in practice.

The remainder of this paper is structured as follows. In the next section we review past research concerning the impact of data sequencing and the notion of re-estimation and re-planning on empirical modeling of software systems development effort. We then describe a set of empirical analyses undertaken using data collected in an industrial setting over a period of eighteen months. The results of these analyses are then discussed in relation to project management practice, followed by the conclusions of our study and recommendations for future work.

2. RELATED WORK

While the passing of time is clearly a characteristic of all software projects it does not feature prominently in prior research work on software effort prediction. This is in stark contrast to research in, for instance, software reliability modeling, in which time is a crucial consideration.

2.1. The impact of sequential data collection

The development and evaluation of effort forecasts in

empirical software engineering research follows a reasonably consistent set of steps: data from completed projects are either collated or acquired, representing the work of one or more groups or organizations; if the data set is relatively small, a leave-one-out approach will be used to build and validate a prediction model; or if the data set is sufficiently large, a *k*-fold hold-out approach might be employed with records randomly allocated to building and validation subsets. In some instances this might be repeated over several runs, to reduce the impact of particular records being assigned to building or validation sets.

As commonly performed, such strategies ignore the fact that the data in question have a temporal dimension, in that the projects that they represent were started, undertaken and completed at various points in time. This aspect may be ignored because the data sets do not include a (reliable) time stamp; time may simply not be thought to be an important consideration; or a data set may be so heterogeneous that the influence of time is overwhelmed by the impact of other factors (e.g. industry sector, project type, technologies employed). The latter may be particularly true in respect to very large repositories of data collated using records from many organizations.

Whatever the reason, the influence of the sequence in which data become available in practice seems not to have been considered in most prior work addressing development effort. Particularly relevant exceptions to this general situation are the studies reported by Lokan and Mendes (see, for instance, [6, 7]). In their work, project completion data was used to inform the building and validation of predictive models of development effort. In their investigations comparing cross-company and single-company predictive models [7], they found that the consideration of the timing of data availability was not useful. In a study utilizing data from a single organization [6], however, their application of a moving window of recent records did prove to be significant. The research reported here complements their work in utilizing an even more homogeneous data set, as well as considering within-project prediction.

2.2. Within-project effort prediction

While the idea of continuous planning and prediction of within-project attributes such as effort and schedule has been described favorably in the literature [2, 9], the degree to which it occurs in practice is difficult to estimate. This is in spite of the fact that systems development and implementation are inherently uncertain activities, which might suggest that project management practices should themselves be dynamic – managers should be permitted to adjust plans as information emerges [10]. Given that this information is increasingly concrete it could be expected that subsequent forecasts would become increasingly accurate, facilitating more efficient resource use.

Although support for ongoing adjustment of project plans and predictions is quite extensive there have been only a few empirical investigations of this issue. The earliest such study we have identified is that of Kulkarni et al. [5], which described the life-cycle phase-based prediction of size and effort for Ada systems. Their approach used object measures of the system (e.g. source lines of code, Ada packages, data flows) at the end of each phase as the input to predictions for the next phase. The strong impact of planning estimates on effort actually expended has been empirically investigated by Jørgensen and Sjøberg [4]. They found that estimates made very early in the software process can be afforded unwarranted significance, even if they are found to be wrong as the project progresses.

Ohlsson and Wohlin [11] employed artifact measures (e.g. number of requirements, flowcharts, input signals) as inputs to predictive models of effort for life-cycle phases. While they found that these measures did not correlate particularly strongly with effort, they concluded that the measures were useful in enabling managers to gain an evolving picture of a project's progress and highlighting the need to re-plan. Rainer and Shepperd [12] conducted a longitudinal case study of planning and effort expenditure at IBM. This enabled them to illustrate the need for the organization to continually re-plan to cope not so much with external events but with the fact that the initial schedule was so unrealistic. Abrahamsson et al. [1] report an empirical investigation of an agile development project in which effort prediction and management occurred with increments in delivery of the software system. They did so utilizing measures of OO design appropriate to the project at hand and coding effort per class. They concluded that the use of several incremental predictions related to iterative releases was more effective than a global, project-level prediction of effort. Finally, Wang et al. [13] promote the use of a machine learning method utilizing small-sample grey models to retrospectively predict software effort per month over a range of large-scale projects, concluding that such an approach could be useful for within-project adjustment of plans and resource management. While few in number, all of these studies point to the potential of within-project analysis that takes time into account.

3. EMPIRICAL ANALYSIS

We now consider the impact of time in two straightforward but illustrative experiments. In the first we look at whether the sequence of project completion affects the accuracy of predictive models built from a single-organization industrial data set related to sixteen projects[1] . In the second experiment we consider these same sixteen projects, but we use within-project data to assess whether useful predictions for development phase effort can be generated.

The data used in the analysis that follows were provided to us by a large multi-national manufacturing company that had a strong and enduring interest in software process improvement. Their data collection activities were quite extensive, at least in relation to development effort

[1] The data set will be made publicly available on the Promise archive [3] as MacDonell2001.

prediction, and they were able to provide us with quite detailed records relating to the prediction and expenditure of effort over multiple waterfall-like phases for several projects, all completed with an eighteen month period. In contrast, their early project sizing data was minimal — only one factor of relevance was routinely calculated or estimated, this being the total number of requirements. Size in comment lines of code was recorded, and is also shown in Table 1 to give an idea of the systems' scale, but of course as this measure is not known until the project is completed it has no direct value as a predictor variable.

	Total Effort	Requirements	CLOC
Mean	2843	41	12088
Median	2113	21	6391
Min	511	3	882
Max	7734	154	36701
Skew	1.00	1.54	1.09
Kurtosis	0.02	1.79	-0.28

Table 1: Descriptive statistics for the 16 projects

Table 1 reveals that within the data set there are a few systems with relatively high numbers of requirements (hence the higher skewness and kurtosis values), but this appears to have had only some degree of influence on the effort required in development. In addition, at sixteen data points this is clearly a small data set, and this may preclude the development of models with sufficiently low standard errors. However, our aim is primarily to illustrate the impact of data accumulation on empirical analyses more so than arriving at robust and sustainable predictive models. It is therefore not our intent to produce a model that has applicability beyond the specific context considered here — rather, we hope to encourage other such analyses in other contexts to also consider the possible impact of the timing of data accumulation on effort prediction.

3.1. Predicting project effort in sequence

In this experiment we consider the impact of the accumulation of project observations over time. The sixteen projects were undertaken some in parallel and some in sequence and were completed over an eighteen month period, as indicated by the recorded project end date. Leveraging the end-dates enables us to simulate the in-practice situation the organization would have found itself in over that time, in that the data for those projects would have become available in sequence. We had two possible predictor variables available — the original effort estimate provided by the project manager and the number of system requirements — as well as the dependent variable — the total development effort recorded for the project in person-hours. Initial exploratory analysis using nonparametric correlation and X-Y plots indicated that the project manager estimate was a feasible predictor variable for actual project effort, so we chose to use this variable only in building our models. We took a very straightforward approach, as follows:

1 use least squares linear regression with a leave-one-out procedure to predict effort for each project; calculate the total error over projects six through sixteen (leave-one-out subset -LOO)

2 use least squares linear regression *with a time-aware approach* to predict effort, by using the first five projects to predict the sixth, the first six to predict the seventh, the first seven to predict the eighth, and so on, until projects six through sixteen have predictions; calculate the total error over projects six through sixteen (time-aware subset -TA)

3 use least squares linear regression *with a time-aware, moving window approach* to predict effort, by using the first five projects to predict the sixth, projects two through six to predict the seventh, projects three through seven to predict the eighth, and so on, until projects six through sixteen have predictions; calculate the total error over projects six through sixteen (time-aware moving window subset -MW)

	LOO	TA	MW
Total error (hrs)	6477	10814	-2602
Relative error	17%	28%	-7%

Table 2: Prediction errors -projects in sequence

Note that we chose five as the minimum number of observations on which to base a time-aware prediction. It may be that a higher or lower number would produce different results, but even if this were the case it would only add weight to the contention that the timing of project data availability warrants attention in our analyses. The summarized results of the above analysis are presented in Table 2.

The results in Table 2 show large differences in total error across the three approaches. The leave-one-out (LOO) approach, our most optimistic in that it utilizes data from all but the project being predicted, underestimates actual project effort by close to 6500 person-hours (equating to around 17% of the total hours expended). The time aware (TA) approach, in which we predict effort for the next project using data from those completed to date, is less accurate still, underestimating by more than 10000 hours. In contrast, prediction using a moving window (MW) of the five most recently completed projects overestimates effort expenditure by around 2600 person-hours, or around 7% of the total.

3.2 Within-project effort prediction

We now turn our attention to within-project prediction. In this experiment we have utilized the effort data collected by the organization regarding the various phases of development to build prediction models for subsequent phase effort. An initial analysis of some of this data was reported previously [8]. Here we extend this work by considering other variables as well as applying a moving window approach to the analysis. In predicting the development effort required in each major phase — Design, Implementation, and Testing — we had at our disposal the total number of requirements for the project (as above) as well as the actual recorded effort per phase. Therefore, in predicting Design effort, we could use the total number of requirements as well as the effort expended in requirements specification. In predicting Implementation effort, we had these same variables as well as actual Design effort; and in predicting Testing

effort we also had records of actual Implementation effort. Our analysis procedure (performed separately for each of the three phases) is as follows:

1 use least squares linear regression with a leave-one-out procedure to predict phase effort for each project; calculate the total error over projects six through sixteen (leave-one-out subset -LOO)

2 use least squares linear regression *with a time-aware approach* to predict phase effort, by using the first five projects to predict the sixth, the first six to predict the seventh, the first seven to predict the eighth, and so on, until projects six through sixteen have predictions; calculate the total error over projects six through sixteen (time-aware subset -TA)

3 use least squares linear regression *with a time-aware, moving window approach* to predict phase effort, by using the first five projects to predict the sixth, projects two through six to predict the seventh, projects three through seven to predict the eighth, and so on, until projects six through sixteen have predictions; calculate the total error over projects six through sixteen (time-aware moving window subset -MW)

		LOO	TA	MW
Total error (hrs)	-Design	4019	4630	1506
Relative error		57%	65%	21%
Total error (hrs)	-Implement	1319	542	-1200
Relative error		11%	5%	-10%
Total error (hrs)	-Testing	-205	602	31
Relative error		-2%	6%	0%

Table 3: Prediction errors -per project phase

We again applied the same admittedly arbitrary minimum of five projects to establish the time-aware and moving window predictions. The results of these analyses of Design, Implementation and Testing effort are shown in Table 3. We observe that the levels of predictive error vary extensively depending on whether or not the timeliness of data availability is considered. It also appears that the increasing certainty that accrues as projects proceed contributed to the generally decreasing error levels in each phase — our Design predictions were rather poor but those for Implementation and then Testing were far more accurate.

4. DISCUSSION AND CONCLUSIONS

The differences in total error evident in the project-level predictions achieved via the leave-one-out and time-aware approaches indicate that the timing of data accumulation is indeed influential, and suggests that analyses that utilize 'complete' data sets may not provide reliable indications of the levels of accuracy that could actually be achievable in practice. For example, there is clear evidence that ignoring time leads to considerable over-optimism.

In our experiment, the organization in question would be best served to use a subset of recent projects for their new project predictions, even if more data were available. Our within-project prediction analysis also demonstrates that timing is influential in two respects: within-project data can be used to predict later-phase effort, and leveraging project timing is also beneficial in phase prediction (although predicting Design phase effort proved difficult with all three methods). We therefore encourage practitioners to ensure that time-related data is faithfully recorded as part of their regular data collection and reporting activities.

Of course sixteen projects is not many, and an organization might be reluctant to predict effort on the basis of just five observations. This is, however, the reality for an organization new to development. In addition, our analyses suggest that using a larger data set may in fact be detrimental to obtaining accurate predictions. In terms of ongoing research, we are testing out these ideas in regard to incremental data accumulation on other data sets to see if the same phenomena are evident, and whether increased heterogeneity means that the impact of project (or phase) timing is reduced. Also of interest would be research considering other forms of iteration — perhaps the approach could be applied to feature sets rather than phases.

ACKNOWLEDGMENTS

We are grateful to our industrial partner for their contributions to this research.

5. REFERENCES

[1] P. Abrahamsson, R. Moser, W. Pedrycz, A. Sillitti, and G. Succi. Effort prediction in iterative software development processes – incremental versus global prediction models. In *1st Intl Symp. Emp. Softw. Eng. & Meas.*, pages 344–353., Madrid, 2007. IEEE Computer Society.

[2] T. DeMarco. *Controllin*g *Softwar*e *Projects*. *Management*, *Measuremen*t *an*d *Estimation*. Yourdon Press, NY, 1982.

[3] The PROMISE Group. Promise data sets. Available: http://promisedata.org/repository/, Last accessed 10 June, 2010.

[4] M. Jørgensen and D. I. K. Sjøberg. Impact of effort estimates on software project work. *Informatio*n & *Softwar*e *Technology*, 43(15):939–948, 2001.

[5] A. Kulkarni, J.B. Greenspan, D.A. Kriegman, J.J. Logan, and T.D. Roth. A generic technique for developing a software sizing and effort estimation model. In *COMPSAC '88.*, pages 155–161, Chicago, 1988. ACM.

[6] C. Lokan and E. Mendes. Applying moving windows to software effort estimation. In *3rd Intl Symp. Emp. Softw. Eng. & Meas.*, pages 111–122, Lake Buena Vista FL, 2009. IEEE Computer Society.

[7] C. Lokan and E. Mendes. Investigating the use of chronological splitting to compare software cross-company and single-company effort predictions: a replicated study. In *EASE 2009*, Durham, 2009.

[8] S. MacDonell and M. Shepperd. Using prior-phase

effort records for re-estimation during software projects. In *9th IEEE Intl. Metrics Symp.*, pages 73–86, Sydney, Australia, 2003. IEEE Computer Society.

[9] R. Madachy. Distributed global development parametric cost modeling. In *ICSP 2007*, pages 159–168, Minneapolis, 2007. Springer.

[10] K. Moløkken-Østvold and M. Jørgensen. Group processes in software effort estimation. *Empirical Software Engineering*, 9(4):315–334, 2004.

[11] M.C. Ohlsson and C. Wohlin. An empirical study of effort estimation during project execution. In *6th Intl Softw. Metrics Symp.*, pages 91–98, Boca Raton FL, 1999. IEEE Computer Society.

[12] A. Rainer and M.J. Shepperd. Replanning for a successful software project. In L. Briand, editor, *IEEE 6th Intl. Metrics Symp.*, Boca Raton, Fl., 1999. IEEE Computer Society.

[13] Y. Wang, Q. Song, and J. Shen. Grey prediction based software stage-effort estimation. *Wuhan Uni. Jnl of Natural Sci.*, 12:167–171, 2007.