PERGAMON

International Journal of Project Management 18 (2000) 13–21

INTERNATIONAL JOURNAL OF PROJECT MANAGEMENT

www.elsevier.com/locate/ijproman

Experts' estimates of task durations in software development projects

J. Hill [a], L.C. Thomas [a,*], D.E. Allen [b]

[a] *Edinburgh University Management School, University of Edinburgh, Edinburgh, U.K.*
[b] *The School of Finance and Business Economics, Edith Cowan University, Perth, Australia*

Abstract

This paper reports a case study of how accurate were experts' subjective estimates of the durations of tasks in a software project. The data available included the estimated task durations given by experts and the subsequent actual duration times. By looking at the results of the case study, the paper shows that although the majority of tasks are overestimated, the mean error is an underestimate of about 1%. The experts however could do even better by taking more cognisance of the number of subtasks that make up a task and hence use the WBS at a lower level when they are estimating durations.

1. Introduction

The effectiveness of project management techniques depend heavily on the accuracy of the task duration estimates. Most of the project management tools—identifying critical activities, the baseline schedules, milestone determination, resource schedules, cost/time charts—depend on accurate duration estimates. This can be particularly difficult to make accurately in research and development situations such as software development. Although the Program Evaluation and Review Technique (PERT) methodology allows the inclusion of errors in the duration estimation it is also heavily affected by what are the most likely duration times of the tasks.

Although project management has been in use for more than 30 years there are few published case studies reporting on the over and under estimates that occurred in task duration estimates in actual projects. Even in software development, although there is a large literature on methods of estimating the total cost or effort of the whole development, (Kemerer [1], Jeffrey and Low [2], Finnie et al. [3]), there is little work reported on the estimates of the individual tasks that make up the project. It is these estimates that are required if project management techniques are to be used to plan and control the project as opposed to an initial pricing or scoping estimate. This paper reports the results of the task duration estimation procedure used by the software development experts of an international organisation on all its software development projects over a three-year period. The experts estimated the duration of the various tasks making up each project and these are compared with the actual time those tasks took. The results suggest that there may be biases that appear in such estimates and that one might be able to develop an automatic correction, which allows for such biases. This could improve project management of software development considerably. In a survey of information systems managers and software developers in over 100 organisations, Lederer and Prasad [4] found that only 25% of such projects were completed within estimated time and budgets. The most frequent cause of overrun was users changing the scope of the project. However the issues that were most highly correlated with overruns were individuals' performance reviews not considering if their

* Corresponding author. Tel.: +44-131-650-3798; fax: +44-131-668-3053.

0263-7863/99/$20.00
PII: S0263-7863(98)00062-3

estimates were met, lack of ways of setting standard durations for use in estimating common tasks and lack of project reviews to compare estimates with actuals. It is hoped that the results presented hereafter will alert other companies to the need to look at the relationship between their estimates and the actual duration times.

The next section outlines some of the ways in which task, total project duration and effort are estimated in software development. Section 3 describes the case study and how the data on estimates of task duration and the subsequent actual duration times were obtained. Section 4 analyses the errors in estimation found in the data while the final section draws some conclusions from this case study.

2. Methods of task estimation

Developing accurate estimates of the duration and effort of the project overall, and its separate tasks, is critical to the usefulness of project management ideas—both in the planning and monitoring of projects. Thus project managers have used a number of different ideas to aid their estimation of the duration and effort of the project and its tasks, particularly in the case of software engineering and software development projects. This is partly due to the development of this industry at the same time as project management techniques were coming to the fore, and partly that the skills needed by engineers in this industry mean they find it easy to use project management ideas. It is also the nature of software development, which is essentially a research and development activity with all the uncertainty involved in that, but where many of the tasks are not dissimilar to tasks that have been previously performed.

The techniques that have been used to estimate project effort and task duration include expert judgement, analogy, parametric models (the most widely used of which were COCOMO and Bang), Function Point analysis and recently neural nets and case based reasoning.

Perhaps the most common approach to estimating effort is to consider the opinions of experts. This does not require the existence of historic data and is particularly useful at the start of system development when requirements are vague and changing, and it is ballpark figures that are required. Experiments suggest expert judgement can be very accurate but it fails to provide an objective and quantitative analysis of what are the factors that affect effort and duration, and it is hard to separate real experience from the expert's subjective view [5]. The reliability of the estimate depends how closely the project correlates with past experience and the ability of the expert to recall all the facets of historic projects. Boehm [6] suggests it is easier to remember the functions developed in a project than the interfaces involved in the development, while it is true that reasons for large overestimates of tasks tend to be recalled more easily than reasons why tasks were completed more quickly than expected. To overcome these failings of the expert some authors [7, 8] suggested using the Delphi technique that allows a group of experts to arrive at a consensus. This approach could also be used to set the parameters for a PERT analysis, where for each task one wants estimates of the fastest possible duration; the slowest possible duration; and the most likely duration of a task [9].

Estimating by analogy is a powerful technique if there is a stable technological environment with some degree of historical data available. One identifies which was the most similar previous project or task and takes its actual time as the estimate of the new project or task. The advantages of analogy include low cost, relative simplicity and reasonable accuracy. The problem is to find suitable analogies [7] and as Yeates [10] emphasises it is unlikely that a previous case will exactly match the new project and hence expert judgement will be needed to adjust for these differences.

A great deal of work and not a little hype has gone into developing algorithmic or parametric estimation methods. These all identify what are the effort or cost drivers in a project and then seek to use data on past projects to fit the parameters of a model based on these cost drivers. The cost drivers tend to be measures of system size and complexity plus perhaps personnel capabilities and experience, hardware constraints and the availability of software development tools. The measures of size varies from likely number of lines of executable code, to number of functions, modules or program features required. The models themselves either use arithmetic formulae or regression approaches, and even the latter tend to split into those where the dependent variable is effort (additive ones), or the log of effort or time (multiplicative ones), or some other function of effort (analytic ones). Boehm [6] adds to this set, in his classification of such models, composite models which are combinations of the previous models. Jorgensen [5] gives a review and a critique of these approaches.

Some of these parametric approaches have become established methodologies in their own right. These include COCOMO, and SLIM. The COCOMO model, proposed by Boehm, [6] can be considered a composite model since it provides a combination of functional forms made accessible to the user in a structured manner. Its purpose is to predict the effort and duration of the total project, but not an estimate of size since the major factors in the model are the estimated number of delivered source instructions and the environment. The latter recognises three types of development mode,

each with its own equation, where the mode depends on the experience of the team and the innovative nature of the project. SLIM is another composite model outlined by Putnam, [11] and also based on lines of code, but using Rayleigh curves to modify the estimates.

Function Point Analysis (FPA) was developed by Albrecht at IBM [12, 13] for quantifying the size of a software system particularly in business applications. Function points are an alternative to source lines of code in measuring the size of a system, by capturing things like the number of input transaction types and the number of different types of reports to be available. Thus one first counts the number of user functions—external input, external output, internal files, external interface files, external inquiries, and then adjusts to allow for processing complexity. The original Albrecht approach [12] was a two-step approach where function points were used to estimate lines of code which were then used to estimate effort needed. Kemerer [1] tries to estimate project duration directly from the function point count. Neither approach was particularly accurate, and so Symons [13] revised the approach by making the method compatible with structured systems analysis techniques, which then made it easier to count logical transactions rather than user functions, and then by recognising that the model must be calibrated at the level of the organisation which is building the system, and not a general industry level. This made it closer to the approach adopted by DeMarco [14] in his BANG system. He had also recognised that at the design stage of a software system, lines of code is not an appropriate metric as one only has information on the business requirement. He therefore sought to develop metrics of the level of complexity of the business requirement by considering the latter as a network of functional primitives—such as calculating the interest on a loan. The review by MacDonell [15] examines how difficult it is to find such suitable metrics and comments on why reports of successful implementation of this method are so few. He felt the main reason was because it was hard to calibrate weightings at an organisation level.

The idea of using Artificial Intelligence techniques to aid estimation has only been tried since 1990. There have been two approaches. Case-based reasoning follows the analogy approach by trying to compare the proposed project with similar ones that have been completed and then identify what are the differences and what implications these have for the effort levels, see Mukhopadhyay et al. [16]. The second approach is to use neural networks where the inputs are the measures of size and complexity together with descriptions of the programming environment which are used in the algorithmic approaches [3].

There have been some comparisons of these methods of estimating total project effort even if there has been little reported work on task duration estimates. Kemerer [1] performed an empirical validation of four algorithmic models (SLIM, COCOMO, FPA and ESTIMACS, which is a proprietary system with similar features to FPA) using data on completed projects to construct estimates of the completion times and then comparing these with the actual times. Jeffrey and Low [2] conducted a similar investigation but allowed for the models to be calibrated at both the industry and the organisation level. Mukhopadhyay et al. [16] compared a case-based reasoning approach with COCOMO, FPA and expert judgement, and found that the other methods did not outperform expert judgement. Finnie et al. [3] compared FPA with case-based reasoning and neural network methods on 299 projects and found the AI models superior. His main point however was that good practical estimation demands good record keeping of estimates and actual outcomes on previous projects.

Heemstra [17] surveyed 364 organisations and found less than 155 used models to estimate software development effort and also that model-users made no better estimates than the non-model users. He also found that firms did not recalibrate their models in the light of their results.

It should be noted though that all these comparisons are of the total project effort or cost or duration needed for initially scoping and pricing the project. None of them consider the task estimates needed to make the project management approach successful.

3. Data on task durations and their estimates

Data on task durations was obtained from the information systems development department of a major international financial organisation. The organisation uses no formal estimation models but uses the expert opinion of project supervisors and managers to estimate the duration of the tasks that make up each project. They in turn probably use some form of analogy, and though estimates may be discussed informally between project team members there is no formal use of Delphi techniques.

Minor projects are planned at the supervisory level. In this planning exercise, the tasks and the subtasks involved are identified and effort needed for each task is estimated, and the tasks allocated to individuals. For more major projects senior management set guidelines regarding project estimates following agreement with the user steering group. The effort required for each task making up the project is then estimated by project managers as before, and the total effort of all tasks must be within the management guidelines. If it

is not, the project is referred back to the user steering group either to scale down the scope of the project or to increase these guidelines.

Once a project is agreed, the data on each task is loaded on the computer. This data includes the initial task effort estimate (recorded in workdays with a minimum input of 0.1 day), the project manager who made the estimate, the type of task, the number of subtasks that make up the task, the number of staff allocated for the task in total and whether the task is being tracked or not (tracking means the project's progress is being controlled by a project management package). During development the actual effort needed to complete the task is also recorded.

The database on tasks has been in existence since 1989 and includes details of over 16 600 tasks. A subset of these was chosen using the following filters. Only tasks which were part of enhancement or development projects were included, which removed projects coded as fixes and training, while tasks which were part of the administration of the project—meetings, walkthroughs, etc.—were also excluded. Only tasks undertaken in the three-year period 1993–1995 and estimated by one of the six project managers employed throughout that period were included. Lastly only tasks with at least 0.5 days of estimated effort were included. A sample of over 500 tasks meeting these criteria was then analysed. There were some extreme values. For example a task that was estimated to take 126 days only took 1.6 days, while another was estimated to take 1 day and took 150.9 days. These and other outliers were examined by the senior managers to see if they were true records of the estimates and actual times. Twenty tasks attracted comment, but only the two alluded to were felt to be spurious by the senior managers, and these were excluded from the sample leaving 506 tasks. The comments were themselves interesting, in that in some cases it was felt that very low duration estimates had been initially allocated for strategic reasons.

Lastly the subtasks making up each task were examined to determine the type of the task. These types were classified as:

T1: setting objectives and requirements;
T2: external design;
T3: internal design;
T4: construction—programme development and implementation;
T5: analysis;
T6: combinations of analysis and construction.

The year the project was planned and undertaken was also recorded in case there was a learning effect by the estimators.

4. Analysis of data

The descriptive statistics for all the tasks outlined in Tables 1 and 2 and show that although 60% of the tasks were overestimated, on average the estimated durations were slightly less than the actual durations, i.e., the error is less than 1%. This implies that the errors when tasks take longer than estimated are significantly higher than if they take less time than estimated. This is reasonable since there is a limit on how much quicker than estimated time the task can be completed, but no limit on how much longer.

Table 2 shows that for almost all the different ways of grouping the tasks the majority of tasks are overestimated. There are two groups in which the majority of tasks are underestimated. The first group is tasks involving analysis and construction and the second group is more complex tasks, including ones with more than three staff or with more than five subtasks. The underestimating of joint construction and analysis tasks suggest that perhaps the requirements were not fully understood before the work started and it might be better to split the tasks into separate analysis and construction tasks. The underestimating of tasks with large teams and large numbers of subtasks suggest a common cause. It may be that estimators tend not to understand the true complexity of larger systems development tasks.

The distribution of the error (estimated time−actual time), as displayed in Fig. 1, is unimodal. There are far too many errors (25% of the sample) in the region [−1.0, 1.0] for the distribution of errors to be normal (i.e., the tails are too thin) and 10% of the cases have no error at all. The standard tests for normality confirm this, and the extremely high kurtosis of this distribution.

Looking in more detail at the errors in the estimates for different types of tasks confirm the results of Table 2. Table 3 gives the mean, median and standard deviation for the errors (estimate−actual) and actual times for each of the task types. Again the major differences occur with tasks of large size—be it by the number of subtasks or staff allocated—and the life cycle option.

Table 1
Summary statistics of estimates and actual durations

All tasks	Estimated days (E)	Actual days (A)	Est−Act (E−A)
Mean	13.51	13.63	−0.12
Stand. Deviation	27.83	27.2	20.43
Minimum	0.5	0.1	−214.4
Maximum	200	255.4	140.1
Median	5.0	3.9	0.3

Table 2
Proportion of tasks which are over and under estimated

Description of group	Overestimate	On target	Underestimate
All tasks	59.5%	8.1%	32.4%
Project manager			
Manager 1	56.3%	16.7%	27.0%
Manager 2	61.3%	2.7%	36.0%
Manager 3	60.9%	4.7%	34.4%
Manager 4	54.3%	7.4%	38.3%
Manager 5	68.5%	5.6%	25.9%
Manager 6	60.4%	5.7%	33.9%
Life cycle option			
T1-objectives	75.0%	0%	25.0%
T2-external design	75.0%	0%	25.0%
T3-internal design	64.1%	20.8%	15.1%
T4-construction	65.6%	9.3%	25.1%
T5-analysis	46.2%	7.8%	46.2%
T6-analysis/construction	44.1%	1.5%	54.4%
Completion date			
1993	61.5%	7.4%	31.1%
1994	57.6%	8.7%	33.7%
1995	57.8%	8.9%	33.3%
Tracking			
Tracking	56.8%	8.2%	35.0%
No tracking	64.0%	7.9%	28.1%
Staff allocation			
3 or less	64.5%	10.2%	25.3%
More than 3	42.1%	0.9%	57.0%
No. of subtasks			
1–5	66.4%	11.4%	22.2%
6–19	42.9%	1.0%	56.1%
20+	45.6%	0%	54.4%
Effort estimation			
Less than 2 days	54.2%	15.3%	30.5%
2–5 days	59.0%	9.5%	31.5%
5–10 days	53.7%	3.0%	43.3%
10–25 days	67.9%	0%	32.1%
More than 25 days	72.1%	0%	27.9%

As is expected, the more the number of tasks the larger the estimated and actual durations, but what is significant is that for tasks with 5 or less subtasks the estimated durations were more than 40% on average greater than the actual durations. This compares with the tasks with more than 19 subtasks where the estimated average durations are 13% on average less than the average actual durations. There are considerable differences in tasks that employ 3 or fewer staff compared with those that need more staff than that. The former have an average actual duration of five and a half days while the latter have an average of over 41 days. Again the estimation procedure over-estimates the tasks with few staff and underestimates those with large numbers of staff. On the first category, the average overestimate is 20% and 65% of the tasks were overestimated. In the second group, the average duration underestimate is over 10% and 57% of the tasks were underestimated. There seems no doubt that experts tend to overestimate small tasks and underestimate complex tasks whether one defines complexity by number of staff used or by number of subtasks. One should not confuse complexity with long duration. When the tasks are split according to the estimated duration, although the majority of tasks of all estimated duration categories are overestimated, the average error is an underestimation of 1.78 days for tasks estimated 2 days or less, an underestimate of 1.46 days for those estimated between 2 and 5 days and an underestimate of 7.3 days for those between 5 and 10 days. On the other hand the average error on tasks estimated between 10 and 25 days is an overestimate of 2.7 days and for even large tasks, an average overestimate of nearly 13 days. Thus tasks with long estimated durations are over estimated and those with low estimated durations are underestimated. This makes sense, but is the opposite result to the one concerning complexity of the tasks.

Looking at the task types, the combined tasks T6 are on average underestimated by about 15% while all the other types are overestimated by an average of 15%. This suggests the lack of clarity in type T6 tasks, which shows up in their mix of subtasks and means that their complexity is underestimated. The other surprising results are negative ones, in that project management tracking seems to make little difference to the errors in the estimates. Similarly when the projects are split into the projects undertaken and estimated in 1993, 1994 and 1995 to see if there is a learning effect, one finds there is a slight drop in number of tasks overestimated between 1993 and the next two years. Similarly the estimated error changed from an average of 1.69 (over) to −1.57 (under) and then to −2.3 (under). This suggests that the estimators are becoming meaner if not more accurate over time. However this trend may be related to the fact that the average task duration time was considerably greater in 1993 than in subsequent years and as was suggested above there is a tendency to overestimate

So far we have looked at the effect on discrepancies between the estimates and the real durations in terms of each facet of a task separately. In the rest of this section we will use regression to see what are the effects of all the tasks combined. This is done by applying regression analysis to the data, using White's correction to allow for an unknown form of hetereoskedasticity. One finds that regressing the actual time (A) on the estimated time (E) only leads to an equation

$$A = 0.708E + 4.07 \quad (1)$$

which has an R^2 of 0.53 suggesting that the estimate only explains about half the variation. Obviously one way of improving the experts' estimates are to including all the other factors that describe the tasks. Doing

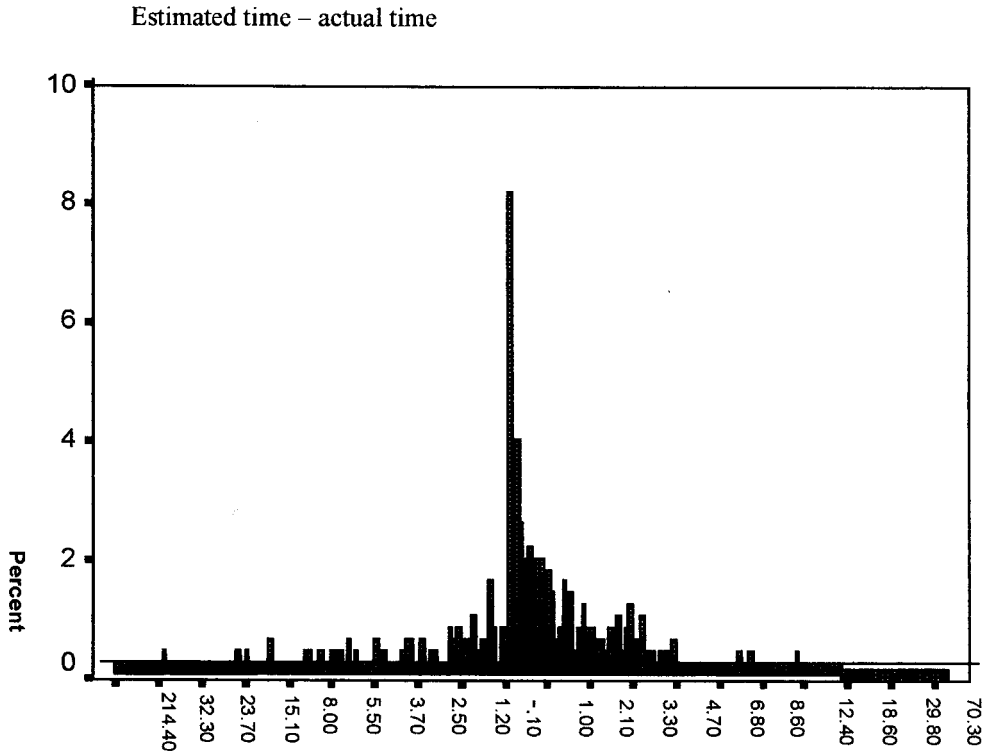

Fig. 1. Distribution of E-A.

Table 3
Means and medians of errors and actual times

Description of group	Est–Act mean	Est–Act st. dev.	Est–Act median	Actual mean	Actual st. dev.	Actual median
Life cycle option						
T1-objectives	6.62	8.96	3.9	1.12	0.89	1.15
T2-external design	2.16	7.41	1.2	11.46	15.7	4.15
T3-internal design	1.06	4.8	0.4	4.45	7.0	2.9
T4-construction	1.27	7.75	0.4	8.12	18.2	2.9
T5-analysis	4.04	23.7	0.2	30.81	48.3	11.05
T6-analysis/construction	−4.36	36.58	−0.5	27.74	38.25	12.8
No. of subtasks						
1–5	1.32	6.44	0.4	3.44	3.88	2.3
6–19	−0.31	19.05	−0.9	18.65	16.6	13.95
20+	−8.78	52.79	−3.9	67.84	49.2	74.5
Staff allocation						
3 or less	1.26	8,28	0.4	5.49	9.87	2.75
More than 3	−4.93	39.97	−2.4	41.66	44.14	28.65
Effort estimation						
Less than 2 days	−1.78	7.44	0.1	3.16	7.53	1.1
2–5 days	−1.46	8.65	0.55	5.53	8.84	3.3
5–10 days	−7.3	27.24	0.7	15.93	27.5	8.6
10–25 days	2.7	9.34	3.3	15.0	9.56	13.15
More than 25 days	12.97	44.87	12.2	58.27	49.6	41.9
Tracking						
Tracking	−0.02	23.56	0.2	17.3	31.87	5.0
No Tracking	−0.3	13.7	0.4	7.49	14.82	2.6
Completion date						
1993	1.69	22.38	0.5	15.62	30.33	3.95
1994	−1.57	13.92	0.25	12.01	20.27	4.1
1995	−2.3	24.7	0.2	11.36	29.6	3.15

this leads to the equation

$$\begin{aligned} A =& 0.177E + 0.07STAFF + 1.33SUBTASKS \\ & - 1.68D_{tracking} + 0.21D_{M1} - 2.13D_{M2} - 3.30D_{M3} \\ & + 0.23D_{M4} - 6.15D_{M5} - 0.70D_{T1} + 2.31D_{T2} \\ & - 0.55D_{T4} + 2.65D_{T5} + 2.04D_{T6} + 1.79 \end{aligned} \quad (2)$$

where STAFF is the number of staff used, SUBTASKS the number of subtasks, $D_{tracking} = 1$ if tracking was used, D_{Mi} is the effect of manager i (compared with manager 6) and D_{Tj} is the effect of task type Tj (compared with type T3). This equation has a R^2 value of 0.845 which means that the right hand side of Eq. (2) is quite a good estimate of the actual durations. Two of the most significant variables are SUBTASKS with a *t*-ratio of 6.21 and E with a *t*-ratio of 2.05. The coefficient of E is 0.177 which means that one takes less than 20% of the estimated time into the equation, whereas the number of subtasks is multiplied up by 1.33 when combining all the factors to get the best estimate of the duration. The details are found in Table 4 which shows that other significant factors are whether the estimates are made by managers 3 or 5. Since the subtasks turn out to be so important it is worth looking at what happens if one uses only those to estimate the actual durations.. The regression equation in this case is

$$A = 1.55SUBTASKS + 0.47 \quad (3)$$

and the $R^2 = 0.82$. This high R^2 for Eq. (3) means the actual times are almost as well explained by the number of subtasks alone as when all the variables are put together, and the fit is better than just using estimated time.

This suggests that if the experts counted subtasks and incorporated this extensively into their estimate they would get better results. So there is the question, do the experts consciously or subconsciously already do this. One way of checking is to do a regression of the estimated times against the task variables. If one does this one finds that the regression has an R^2 of 0.49 and the equation is

$$\begin{aligned} E =& 1.18SUBTASKS - 0.11STAFF + 2.92D_{tracking} \\ & \pm 2.93D_{M1} + 1.18D_{M2} + 0.94D_{M3} + 1.92D_{M4} \\ & - 3.375D_{M5} + 3.87D_{A} + 3.81D_{B} - 0.37D_{D} \\ & + 12.42D_{E} + 2.19 \end{aligned} \quad (4)$$

Again, if one only looks at the relationship between estimated times and number of subtasks one gets the regression equation

$$E = 1.22SUBTASKS + 3.18 \quad (5)$$

with a $R^2 = 0.48$. What these results seem to say is

Table 4
Results of regression in Eq. (2). Ordinary least square regression applying White's corrections for hetereoskedasticity. Actualtime = α + β1estimate + β2staff + β3subtasks + β4tracking + β5M1 + β6M2 + β7M3 + β8M4 + β9M5 + β10T1 + β11T2 + β12T4 + β13-T5 + β14T6 + ε

Variable name	Estimated coefficient	Standard error	T-ratio 491 DF	P-value	Partial corr.	Standardized coefficient	Elasticity at means
Estimate	0.17686	0.8605E–01	2.055	0.040	0.092	0.1810	0.1752
Staff	0.0717	0.4944	0.1452	0.885	0.007	0.0060	0.0134
Subtasks	1.3306	0.2142	6.211	0.000	0.270	0.7765	0.8276
Tracking	−1.6774	1.408	−1.192	0.234	−0.054	−0.0299	0.0770
M1	0.20860	1.081	0.1929	0.847	0.009	0.0033	0.0038
M2	−2.1308	2.045	−1.042	0.298	−0.047	−0.0279	−0.0231
M3	−3.3011	1.546	−2.135	0.033	−0.096	−0.0404	−0.0306
M4	0.22777	1.958	0.1163	0.907	0.005	0.0031	0.0027
M5	−6.1461	1.850	−3.322	0.001	−0.148	−0.0698	−0.0481
T1	−0.69838	1.839	−0.3798	0.704	−0.017	−0.0023	−0.0004
T2	2.3073	1.899	1.215	0.225	0.055	0.0106	0.0027
T4	−0.54811	0.801	−0.6842	0.494	−0.031	−0.0100	−0.0231
T5	2.6525	4.019	0.6600	0.510	0.030	0.0160	0.0054
T6	2.0388	1.369	1.489	0.137	0.067	0.0333	0.0402
Constant	1.7928	1.690	1.061	0.289	0.048	0.0000	0.1314

Durbin-Watson = 1.8597

Ordinary least square regression applying White's corrections for heteroscedacity giving Eq. (2) 0.506 observations.
Dependent variable = ACTUALT IME.
Using White's heteroskedasticity-consistent covariance matrix, R-square = 0.8453, R-square adjusted = 0.8408.
Sum of squared errors-SSE = 57850.
Mean of dependent variable = 13.643. 491 DF.
Log of the likelihood function = −1916.97.

that of all the factors described, it is the number of subtasks that has the biggest impact on the time estimated. However since R^2 is only 0.48 it means that there is a lot of variation in the estimated times that cannot be explained by the number of subtasks. It is not possible to say from the data what it is that affects the experts estimates. All that can be said after examining Eq. (4) and noting that it is no better an estimate than Eq. (5) is that it is not anything to do with the type of task or the manager who is doing the estimating.

Although almost all the explanation of the estimated times is in its relationship with subtasks, these estimates are still not putting enough emphasis on subtasks, since the relationship of subtasks with actual times is even stronger than that of subtasks with estimated times.

Since the real error is A–E, regressions that estimate real errors in terms of estimated times and the other factors lead to an equation which is just a rearrangement of Eq. (2), though of course the R^2 will be larger because the values of the dependent variable are so much smaller. Looking at the relative error so that the dependent variable is (E–A)/A or (E–A)/E does not give very good fits (R^2 of 0.13 and 0.26 respectively) and although the effect of subtasks remain significant in these regressions it is not so great as in the straight line error regressions. Other variables, like number of staff and type of task, which manager does the estimating, become significant at the 5% level in these regressions, as well as the estimated time being significant.

5. Conclusions

The results suggest that in this case study the expert project managers all tended to overestimate the majority of the tasks (around 60%), but if they underestimated, the errors tended to be greater so the mean error was an underestimate of about 1%. The distribution of errors was a mixture, with some tasks being estimated completely accurately, others taking one day less than estimated and the rest having almost a normal distribution. One should be cautious at giving explanations for this, but it could be that some tasks are routine and there is an understanding in the organisation of the length of time they are to take, which is known by project managers and analysts. Hence such tasks tend to come in on time or a little early. The others where their development is less clear are the ones that can be approximated by a normal distribution, with a slightly longer underestimating tail than an overestimating one.

Looking at the estimates in more detail it seemed that estimators made the largest underestimating average errors on tasks that were not clearly defined (category T6) or were complex, involving a number of staff and a large number of subtasks. The major over estimates came in tasks involving setting requirements (T1) or analysis (T5). Tracking of the tasks seemed to have little effect, and the learning effect over time was to tighten up the estimates so that the average estimated task times instead of being 1.7 days more than actual became 2.3 days less than the actual times. This effect might have as much to do with pressure within the organisation for projects to be done as efficiently as possible as any learning effect on what really happens in projects. One way this pressure exerted itself was in a drive to squeeze 'more value for money' as a way of improving the returns to the clients.

The main conclusion though is the strong relation between task time and the number of subtasks involved in the task. This was by far and away the best estimate of the likely time of the task—better than the estimated time even. What this suggests is that a careful use of work breakdown structure approaches to identify the packets of work at the lowest level is not just useful for smooth management of the project, but is also one of the most useful thing experts can do to estimate the times of the tasks better. It is clear that the managers in this case study understated the full effect that the number of subtasks has on the time of a task. This may be because such tasks were poorly thought through and the dependencies between the subtasks not really understood; or it could be that there is poor project management at the subtask level.

Acknowledgements

The paper was written while one of us (LT) was a visiting professor at Edith Cowan University. We wish to acknowledge the University's support in funding this visit.

References

[1] Kemerer CF. An empirical validation of software cost estimation models. Commun. ACM 1987;30:416–29.
[2] Jeffery DR, Low GC. Calibrating estimation tools for software development. Software Engineering Journal 1990;5:215–21.
[3] Finnie GR, Wittig GE, Desharnais JM. A comparison of software effort estimation techniques: using function points with neural networks, case-based reasoning and regression models. J. Systems and Software 1997;39:281–9.
[4] Lederer AL, Prasad J. Causes of inaccurate software-development cost estimates. J. of Systems and Software 1995;31:125–34.
[5] Jorgansen M. Experience with the accuracy of software maintenance task effort prediction models. IEEE Transactions on software engineering 1995;21:674–81.

[6] Boehm BW. Software engineering economics. Englewood Cliffs: Prentice Hall, 1981.
[7] Goodman PA. Application of cost estimation techniques: industrial perspective. Information and Software Technology 1999;34:379–82.
[8] Shepperd M. Foundations of software measurement. Englewood Cliffs, New Jersey: Prentice-Hall, 1995.
[9] Kerzner H. Project Management: a systems approach to planning scheduling and control. New York: van Nostrand Reinhold, 1995.
[10] Yeates D. Systems project management. London: Pitman, 1986.
[11] Putnam LH. A general empirical solution to the macro-software sizing and estimation problem. IEEE Trans. on Software Engineering, 1978;4.
[12] Albrecht AJ, Gaffney JE. Software function, source lines of code and development effort prediction: a software science validation. IEEE Trans. on Software Engineering 1983;9:639–48.
[13] Symons CR. Software sizing and estimating MkII FPA. Chichester: Wiley, 1991.
[14] DeMarco T. Controlling software projects: management, measurement and estimation. New York: Yourdon, 1982.
[15] MacDonell SG. Comparative review of functional complexity assessment methods for effort estimation. Software Engineering Journal 1994;9:107–16.
[16] Mukhopadhay T, Vicinanza SS, Prietula MJ. Examining the feasibility of a case-based reasoning model for software effort estimation. MIS Quarterly 1992;16(2):155–71.
[17] Heemstra FJ. Software cost estimation. Information and Software Technology 1992;34:627–39.

J. Hill obtained a Masters degree in Business Administration from the University of Edinburgh. He is a senior manager in the research and development department of an international financial organisation, and has been overseeing software development projects for a number of years.

L. C. Thomas is Professor of Management Science at the University of Edinburgh. His undergraduate degree and his D.Phil were in Mathemtics from the University of Oxford. He has authored and edited seven books and over 100 papers in the Management Science area.

D. E. Allen is the foundation Professor of Finance at Edith Cowan University, having previously been at Curtin University, the University of Western Australia and the University of Edinburgh. He received a degree in economics from the University of St.Andrews, a M.Phil from Leicester University and his Ph.D. in finance from the University of Western Australia. His research interests include a number of areas of business economics and finance, portfolio analysis, estimation of risk and the statistical estimation of financial and other data.