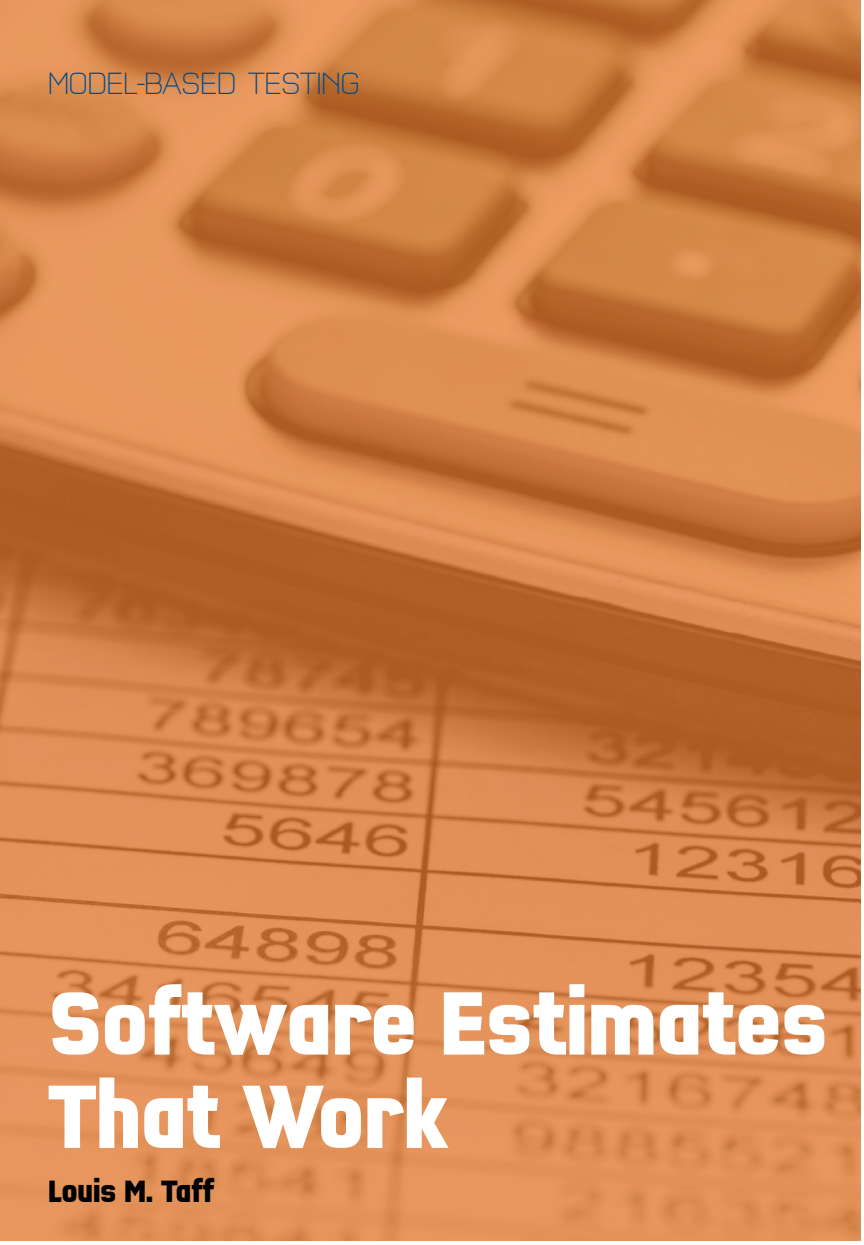

Software Estimates That Work

Louis M. Taff

Summary. We lay out a path for initial estimating of large new projects for systems and software development organizations that have experienced resource problems. The paper is targeted mainly at organizations not yet underway with efforts to improve estimating (following, e.g., the Software Engineering Institute's Capability Maturity Model); however, those already working on process improvements may find suggestions as well.

We propose a three-pronged approach. The first is to implement a standard estimation *process*, and the second is to deeply involve development staff — those who do the work — and other stakeholders in this process. The process itself will generally be unique to each organization and its particular culture, projects, circumstances, people, etc. A reasonably formalized process offers benefits such as better estimates, more consistent estimates, accountability and manageability of the estimate process, and a baseline for continuous incremental improvement of estimates.

Depending on the process, additional benefits may be derived, such as better technical designs and better relationships between staff. Better estimates in turn will lead to project-staffing profiles that avoid "back-end loading" projects and will yield higher software quality. Involvement of the staff should build "buy-in" to the estimates and improve the probability of meeting them. We have validated this approach on a large project (hundreds of staff-years) in a traditional waterfall development environment. It may not be directly applicable to more recently evolved Agile processes because their requirements may not be known sufficiently as they begin. Finally, the third prong is to put in place a method to measure the effort expended on projects in sufficient detail to capture historical data for future estimates.

Background

The software community has been notorious for its poor record in estimating the costs of system development. The results of poor estimates are well known — cost and schedule overruns, canceled projects, poor morale and poor software quality because of the rush to finish on time — and ultimately due to insufficient early staffing because of an estimate that was too low.

We attempt here to make the case that estimating system-development costs can be improved by formalizing estimation into a *process* — a planned sequence of inputs, work activity and outputs. It is difficult to know how many organizations have a defined process for estimating. A member of an organization with which we consulted stated before it defined a process: "Today there is no single method employed for estimating projects' resources, and the methods currently employed are somewhat informal and vary from estimator to estimator and project to project. These estimates are not based on a well-defined estimation process and a quantification of software size such as lines of code. It is common knowledge that we do not know where these estimates come from and how they are derived. As a result, the process of estimation is highly subjective and difficult to measure and improve. More critically, we do not deploy any feedback mechanism to check actual project spending against our estimates. Therefore, we do not learn from the past."

We feel that this unnamed organization is not unique. Vigder and Kark [2], in a Canadian study with a military focus, found "definite patterns in which projects are procured, costed, and executed. These patterns show that

- Software cost estimation is done too early in the procurement process and is based on – usually – wrong specifications.
- Once established, the estimates are very difficult to change.
- Parametric models for software cost estimation are rarely used within either the military or the industry developing systems for the military.
- Business-oriented software producers are more likely to use some parametric models.
- There are no historical data on which to base the estimates for the new software projects.
- Very often the price for the software development ... does not reflect the actual estimates for the software as a function of the requirements, but is a function of many other non-technical factors."

In addition to the ill-defined methods that produce such estimates, there are further drawbacks to not having a process. The estimates are not well documented, and it's not clear what is included in them and what, if any, assumptions were made. The time and effort used to make the estimates was probably "borrowed" from other work, which may be delayed because of it, and the time might not be accounted for anywhere. If the estimates are needed by some deadline (i.e., if there is a schedule), it may not be easily known whether the deadline will be met; the progress in coming up with the estimates is hard to track.

Why Make Estimates?

The Business Case

It's useful to consider why we make estimates in the first

place. For any project, software or otherwise, the criterion for deciding to go forward is some kind of business case — it is perceived that the value of the project's results exceeds its costs. This is true whether we are estimating one of many features for a new release, an entire software project, a bid on a job, or simply adding a new room to our house.

The costs in a business case may include many items in addition to software development. Depending on whether the developed system is to be sold or licensed or used only internally, there may be costs for requirements development, third-party software acquisition, marketing, sales, deployment, customer service, operation, data entry, maintenance, etc. And there will be trade-offs — extra attention and costs in human-interface development may reduce larger expenses later in operation and data entry. Thoughtfully designed screens, higher performance or more functions may increase development costs but yield more sales. This last notion illustrates that inherent in this business case is an estimate of the benefits as well as the costs. Notwithstanding, for large software projects, development is usually the dominant cost.

To estimate all costs, we need to know what the product will do after it's built — we need to know its requirements in sufficient detail to estimate the project's costs and its benefits as well. Some level of requirements is one of the major inputs for estimation. As an aside, in our experience we have not seen as much attention given to estimation and its failures on the benefits side of the business case as on the costs side.

Staffing

A second major purpose of estimates is to set the levels of staffing and other resources for the project once the decision is made to go forward. We have pointed out [1] that this initial staffing may have a strong influence on the quality of the product. For if the resources are underestimated, the early staffing profile will be too low — and the architectural and design phases will not receive the attention appropriate for the project's true size. As a result, the project will be "back-end loaded" to meet its schedule, and quality will suffer in a rush to finish on time.

Estimates and Buy-In

Thus, initial estimates are used to set the budgets and resources of managers and staff. If the estimates were made with methods that appear to these people to be arbitrary, can they be expected to "buy in" to them and strive to meet them? We feel that they cannot. In our experience, to gain buy-in they or someone they trust must participate in the estimate process. Some staff may have experienced an underestimated project with an end game of rushing, panic, and low-quality results. These people may mistrust any estimates and staffing profiles, overestimating if asked and generally poisoning the project atmosphere. Therefore, *a primary goal of the estimate process needs to be to gain the buy-in — agreement, acceptance, ownership and commitment — of the managers and staff who will commit to the content and schedule and do the work.*

Although not realistic in most projects, in an ideal situation the same individual makes and commits to an estimate, then personally executes the project and benefits (or suffers) from meeting the commitment (or not). This is a competitive bidding model. The same individual feels pressured to bid both low (to get the work) and high (to profit).

Problem Statement

In summary, most organizations need a method to generate software estimates that:

- Has buy-in from all stakeholders.
- Is consistent from project to project.
- Is based on the same documented requirements that are used for other purposes in the business case.
- Produces well-documented estimates.
- Includes all assumptions.
- Can be tracked and managed.
- Improves with experience.

What's Needed Is a Process

What Is a Process?

We feel that a big step toward meeting this need is establishing a formal process for generating estimates. We can define a process as a specified sequence of work activities and/or processes with inputs and outputs. This recursive definition allows processes to incorporate subprocesses. The availability of inputs and the production of (presumably objective) outputs gives us milestones with which to measure progress and manage — one of the benefits of a defined process. A formal process gives visibility to the effort needed for estimation — it can be budgeted, scheduled and tracked by project managers. If the process is written down and followed it should be repeatable over time. If, in addition, a method is in place for measuring the costs of the development process, the groundwork is laid for incremental improvement of the accuracy of the estimates. Finally, once a process is established, it can be augmented to produce additional outputs. For example, many feel that some measure of risk is useful to management.

Project Size

The principles we lay out scale up to large efforts (2,000–4,000 staff-years), as evidenced by our "Estimeeting" process [1].

Expectations of Cost and Accuracy

We wrote above that staff need time to do estimation. We have noted [1] that estimates with an accuracy of ±20 percent may cost up to 10 percent of the total effort of the entire project. This is a significant cost, and it should prove to be a useful barrier in filtering out requests for estimates that are not serious. In our Estimeeting process [1], this cost buys much more than just an estimate — it gives a high-level design, a breakdown of the cost into component parts, exposure of the prospective project across the entire development organization, and more. Similar benefits will apply to any process designed to give this level of accuracy.

Alternatively, lower accuracy is presumably achievable at lower cost. The appropriate trade-off between estimate cost and accuracy generally depends on the circumstances of each project and organization, the project size, the consequences (i.e., the risk of under- or overestimation), etc. If multiple projects are underway, underestimation might be alleviated by

borrowing resources from other projects. Clearly, however, if the project will consume all the resources of the organization, underestimation is extremely serious.

How We Would Go About It

Build Consensus

If we were to create a process for estimating, we might first find everyone who wants the estimates. We'd need to discuss with each of these people what the estimates are to be used for, what their inputs are (what the estimates are to be based on), what their expectations are for accuracy, other outputs desired from the estimating process, straw timelines for both the estimates and the project itself, and how the cost of making the estimates is to be funded. We'd then need to talk through a proposal as to how the process would work in our particular environment, with our unique culture and business and development processes. This would include finding someone to *own* the process — that is, someone to make the estimates happen. It also includes identifying all stakeholders, especially those who are going to make the estimates and those who will be expected to commit to them. At some point the support of upper management needs to be secured. Depending on the environment, this can be earlier (to help get everyone's buy-in) or later (once a realistic straw plan is formulated).

The estimating process we describe in reference [1] included people from marketing (customer needs), product management (business case), systems engineering (requirements), architecture and design/development — all the stakeholders of the business. We advise involving people from areas not always included in the early stages, such as documentation, project management, product support/maintenance and (if not included in design/development) system test/verification. Their roles are to ensure a common understanding of requirements and to estimate if necessary. If the product is to be deployed within the enterprise, prospective users (i.e., customers) should be included as well to verify that the requirements meet their needs. Various stakeholder roles within the estimation process are different from each other, however. For example, the business side is not involved to "negotiate" an estimate but to ensure that what is estimated fulfills business and marketing needs. If during the process it becomes clear that the cost will be too high, these people must have the authority to cut back the project scope to get an affordable estimate while ensuring that the business case remains positive.

Possible Milestone	Possible Owner
Estimate process owner identified	Upper management
Process funding established	Management
Market requirements	Marketing and/or customer(s)
Technical requirements proposal	Systems Engineering
Allowable interval (schedule) of project	Marketing, product and project management
Identify available staff	Development management
Inventory of staff experience level	Development management
Identify estimators trusted by staff	Development management
Compile estimators' needs.	Estimate process owner
Survey of size/complexity of existing software base, if any	Development management
List of special hardware available (e.g., labs, communication facilities, etc.)	Development management
High-level architecture/design proposal	Architects/designers
Work items/work-breakdown structure enumerated.	Designers and/or Estimators
Assumptions listed.	Estimators
Open issues to be resolved.	Estimators
Technical staff weeks, e.g., by specialty (coders, testers, documenters, etc.) with a measure of risk – e.g., min/max/most probable – for each estimate.	Estimators
Equipment needed (testing labs, instruments, etc.)	Estimators
Achievable end date with reasonable built-in contingency and assumptions about staffing.	Estimators or managers
Commitment and sign-off to the (possibly adjusted) estimate by responsible manager(s)	Managers

Table 1: Possible Milestones for Estimating Process

List the Deliverables and Their Owners

Some possible milestones and deliverables associated with an estimating process are compiled in Table 1. First, we need to identify estimators trusted by the development staff and ask them what inputs they need. If these are not all available, then as part of the process, staff and budget need to be found to produce them from what is already available. This may include considerable work, such as producing a more detailed specification proposal and/or a design proposal. We term these documents "proposals" and suggest that they shouldn't be binding on those who later implement the project, particularly if they are not the people who produced them. The design proposal can function as an existence proof, bounding the cost.

We suggest that the process output, the estimate, by itself, *not* be considered a commitment. Commitments are made by those with the authority to decide how to use resources and usually with influence on the reward system. Therefore, we have included a separate commitment milestone with management owners in Table 1.

How Are Estimates Made?

Vigder and Kark [2] note that "the bulk of the current literature and research on cost estimation is devoted to formal models, particularly as relates to new system development." However, they "found that formal models are *not* in general used by estimators as a primary tool for cost estimating. By an overwhelming majority, informal analogy was the most commonly used estimating method for all types of software and for all organizations. Estimators used their past projects as a basis for estimating the cost of future projects." A similar conclusion was reached by Hihn and Habib-Agahi [4]. Indeed, this method is the basis of our Estimeeting process[1].

Analytical methods such as function points, noncommentary source lines (NCSL), and other objective measures require training in their use and may require more detailed requirements than are available when the estimates are needed. These methods also need to gain the confidence of the managers and workers expected to commit to them. If the conditions of training, detailed requirements, and confidence exist, analytical methods may be an important ingredient in the accuracy of and buy-in to a formal process.

Forms and Checklists

We strongly recommend what we feel is an important additional task, namely the production of appropriate forms and checklists to record and guide the process. This captures the knowledge of the enterprise, incorporating the issues and tasks that have proven important in the past and helping ensure that nothing is forgotten. We like to quote Donaldson [3]: "...provided you are reasonably systematic it is not difficult to estimate the duration of an activity, the problem is to identify the right activities."

Form	Filled Out By	Given To	Reason
Estimate request	Product manager, marketing	Estimate coordinator	Asks for estimate, authorizes funding for it
Function or subsystem estimate	Estimator	Estimate coordinator	Lists work items, needs, assumptions, issues and estimate(s) for function/subsystem.
Estimate summary	Estimate coordinator	Development manager	Adds function/subsystem estimates and gives best technical estimate
Committed estimate	Development manager	Product manager, marketing	Gives committed estimate.

Table 2: Possible Forms in an Estimating Process

Forms and checklists serve additional purposes as well in that they *document* the estimates without anyone needing to write a report. If the appropriate reference source information is on the forms, the estimates become traceable as well. Forms and checklists lend themselves to process improvement also, because new knowledge can be added to the forms easily for future estimation. The particular content of the forms and checklists depends on the estimation process.

What's in the Estimates?

In addition to focused development, staff members typically engage in myriad activities not directly related to producing software. They may variously participate in training, traveling, interviewing job candidates, consulting and reviewing documents on other projects, general meetings, team building, making phone calls, answering email, fighting project and nonproject fires, etc. These activities often interrupt project time and incur get-back-on-track costs. The question is how time spent on them is accounted for. Ideally, most such activities would be listed individually in timekeeping records, but this is unrealistic because the durations spent on them happen in small pieces, because no one can track at that granularity level, and because it's unlikely that anyone really cares. Management must tell project staff how to treat these intervals. Staff might be asked to estimate the total time spent this way each time card interval and include it in a catch-all category. If, alternatively, they are told to basically ignore it, estimators need to adjust their estimates for it based on their experience and on measurements of previous projects.

Example: The Estimeeting Process

We have previously described a process in the telecom field [1] built around estimating meetings we call "Estimeetings." This process is based on breaking down the work architecturally into impacts on a number of existing subsystems. Estimators representing the subsystems estimate work in their own areas of expertise. This process is particularly suited to multiple releases of a product with continuity of knowledgeable people.

Notes on Measurements

If our process is written down, it will be repeatable and will lend itself to incremental improvement — doing better next time than last time. Improvement implies knowing how well we did — it implies *measurement*. We can thus distinguish between the benefits of simply having an estimation process — and these benefits are real — and having a measurement method. That is, estimation and measurement are separate and have distinct benefits. A measurement method may be even more worthwhile than an estimation process. If accurate, the benefits of measurement include:

- The determination of the real total costs of the project. Knowing how accurate the originate estimates turned out is a key input in estimating future projects.
- The determination of the costs of each subprocess or function of the development process, ideally by stage (design, unit testing, integration, etc.).
- Applied to the estimation process, costs of the estimates themselves can be measured.

A method for measurement of costs is not within the scope of this paper, but we do have a few remarks on the subject. The most straightforward way to record cost data is to use the work time reporting mechanism (time cards). Time reporting categories and subcategories would reflect the activities and tasks estimated.

However, there may be issues with the use of time cards for measurements. First, the organization's budgeting or funding mechanism may not encourage straightforward time reporting. This can happen if time-reporting categories cannot easily record the plethora of different activities estimated separately. Further, if a project runs over budget, staff may be asked to charge their time to a different project. Depending on the environment, as a financial practice this can range from "doesn't matter" to being illegal. But for measurement purposes, it is unacceptable. Second, there may be issues with reporting overtime. Some staff may not be authorized for overtime but work the hours anyway, not reporting them on their time cards (see below). This practice will lead to continued underestimation.

More subtle personnel issues may play a role as well. For example, a staff member who, based on his or her estimate, has made a commitment to "finish the module by month's end" may be working evenings and weekends to get the job done, and may not report the unpaid overtime because of pride or embarrassment at having underestimated the job. Some staff may fear being penalized for spending too little or too much time on some special function. Or there may be a macho culture of status for those who consistently work late. These issues illustrate the difficulty of accurate measurement, important though it is. They also illustrate an ethical bind on management — the conflicting desires 1) to meet or exceed budget

goals (and increase competitiveness in winning future projects) and 2) to fairly compensate staff who might not be reporting all their time (and prevent burnout and departures).

Perhaps a way to avoid some of these concerns is to use an anonymous system separate from time cards. The system needs to track whether individuals have made their weekly entries but would not associate their inputs with them. The measuring method should measure every activity that was estimated. If significant activities took place that were not estimated, these need to be measured and added to the estimation process — for example, to the forms and checklists. Some mechanism needs to be included to make this easy. Many commercial products exist for time reporting, comparison with estimates, integration with project management software, etc.

Function/Subsystem Estimate Form					
Product					
Feature estimated					
Function/Subsystem					
Target Release					
Estimate Date					
Estimate Based On					
Estimator					
#	**Work Item**	**Min**	**Avg**	**Max**	**Note**
1					
2					
…					
#	**Notes / Issues / Assumptions / Special Equipment or Labs, Etc.**				

Table 3: Simple Function/Subsystem Estimation Form. Possible mandatory/optional breakdown is not included. The minimum/average/maximum ranges are intended to provide a measure of risk. Example functions or subsystems might be screens/human interface, database, communications, math library, etc. A large project will require many of these forms.

Estimate Summary Form			
Product			
Feature estimated			
Target Release			
Estimate Date			
Estimate Coordinator			
Function / Subsystem	**Min**	**Avg**	**Max**
Total estimate			
Notes			

Table 4: Possible Summary Form for Assembling an Estimate. For changing an existing product, the list of functional areas or subsystems can be hard-coded into the form to ensure none is forgotten.

Conclusion

We have built a case that making estimates for significant software developments is best accomplished through a formalized process of estimating that involves all stakeholders. We cannot give a one-size-fits-all approach, but we feel that a repeatable process with input from all the stakeholders and documented outputs will go far in relieving confusion and misunderstandings, will focus attention on the importance of estimation and will achieve support from the people who must execute the plan. We further advise that a method be established for measuring development costs, though we find it difficult to give detailed guidance on how to achieve this.

Acknowledgement

It is a pleasure to acknowledge the contribution of the ideas, insights and perspectives of my Estimeetings co-author Jim Borchering. It will be clear to readers of reference [1] that I have borrowed heavily from it in constructing this paper.

ABOUT THE AUTHOR

Lou Taff created the first structured process for estimating software development for a large switching project at AT&T Bell Laboratories, where he was Distinguished Member of Technical Staff. At AT&T he planned and engineered new services for the AT&T network and held the supervisory rank of technology consultant. Subsequently with Ericsson, he worked on wireless systems and platforms. Taff holds an S.B. in physics from MIT and a Ph.D. in nuclear physics from Iowa State University; he initially worked in physics at the University of Groningen in the Netherlands and in the computing department at Fermilab. He is a co-inventor of four patents and has authored over 20 published articles and many proprietary reports. He now maintains a consulting practice in which he assists clients in implementing software estimating processes.

taff@softwareestimatingprocesses.com

REFERENCES

1. Taff, L. M.; Borchering, J. W. & Hudgins, W. R. Jr. (August 1991.) "Estimeetings: Development Estimates and a Front-End Process for a Large Project." IEEE Transactions on Software Engineering 17, No. 8. 839-849.
2. Vigder, M. R. & Kark, A. W. (February 1994.) "Software Cost Estimation and Control." National Research Council Canada. Institute for Information Technology, Rpt. No. 37116. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.9301.
3. Donaldson, H. (1978.) "A Guide to the Successful Management of Computer Projects." London: Assoc. Business Press.
4. Hihn, J. & Habib-Agahi, H. (1991.) "Cost estimation of software intensive projects: a survey of current practices." 13th International Conference on Software Engineering. IEEE Comput. Soc. Press. Los Alamitos, California. 276-87.