Iterative development offers several opportunities to apply a simple use-case-based measure of project performance.

Ray Ashman

Project Estimation: A Simple Use-Case-Based Model

Software development estimates are, and continue to be, inaccurate. Overly optimistic estimates are major contributors to project failure, despite the fact that every completed project is a rich source of information about performance and estimation.

Modern development processes, such as the IBM Rational Unified Process, promote risk management, the realization of architecture first, the decomposition of the project into iterations, and the assignment of requirements to these iterations. When a project adopts these forms of best practice, it achieves a high degree of technical control, which in turn makes for easier management. One difficult project management task that remains, however, is to accurately determine the effort required to complete the project.

Here, I discuss a use-case-based estimation model for determining project effort. This technique calls for looking at the relationship between estimated and actual data, then using this information to improve future estimates. Using a simple set of metrics, it is possible to generate a credible model for project estimation. The model described here works best in an iterative development process, allowing comparisons between successive iterations. It thereby presents early feedback about each iteration's performance.

HOW EFFECTIVE IS YOUR ESTIMATE?

Cost and time overruns are common within software development. When project managers understand little about the requirements or the challenges ahead, they often present estimates with a stated accuracy on the order of 20 percent. These excessively optimistic estimates inflate the global failure rates for software projects. Actual overruns, identified in the *Chaos Report* (The Standish Group, 1994), make sober reading.

For example, fewer than 28 percent of projects fall into the type 1 category, the best of three project resolution types used in this report:

- *Type 1, successful.* These projects are on time, on budget, have all the original features, and function as initially specified.
- *Type 2, challenged.* Although completed and operational, these projects are over budget, over time, and have fewer features and functions than originally specified.
- *Type 3, impaired.* The project is cancelled at some point during the development cycle.

Tables 1 and 2 come from this report and support the argument for more realistic estimates, or at least that a wider margin of error should accompany an estimate. In accounting for challenged and impaired projects, the average cost overrun was 178 percent for large companies, 182 percent for medium companies, and 214 percent for small companies. Challenged and impaired projects had average time overruns of 230 percent for large companies, 202 percent for medium companies, and 239 percent for small companies. Recent reports indicate no significant change in the percentage of overruns.

The ability to accurately estimate a project in terms of cost and time is a skill that evidently eludes many project managers. Yet every completed project is a potential source of information: data on the project's behavior, the problems encountered along the way, and the overall cost. Unfortunately, this information often goes unnoticed, and its value, ignored. Using it could make possible much more accurate estimates.

WHAT TO MEASURE

For this model, I categorize measurement into three types. The first two types are associated with the product and the last one with time. Although I've separated the product measurement types, they are inextricably linked because each has an effect on the other.

- *Scheduled measurements.* These encompass estimates of cost and resource. These measures are generally understood and are normally the focus of the project planning efforts.
- *Unscheduled measurements.* Unscheduled measurements cover areas such as rework, requirements change, and other alterations to the project. They are difficult to predict and therefore rely on risk estimation to provide a degree of management.
- *Time.* This is a reference for trend and difference measurements. Unless a metric is measuring quantity only, it will require time as a common reference for comparative measurement. An important factor to consider in using time is its granularity, because the granularity selected dramatically affects the volume of data. Analyzing data requires comparing like to like, which includes having to standardize time measurements if the comparison relies on a time coordinate. (That is, all data must use the same sample rate, such as seconds, days, or weeks.)

With the three types of measurement identified, I can construct a first-cut model for project estimation. Prior to any modeling, you should state the aims of the exercise. These aims will serve as a guide to recording the degree of fit between the model results and reality, over successive projects. My model's aims are to be able to capture the estimator's experience and to increase the accuracy of successive estimates based on repeatable measurements.

ESTIMATION MODEL

Estimations based on project requirements provide the simplest form of modeling, because project requirements are readily available, and the derived estimates are easily assessable at the end of the project. Moreover, interproject estimation refinement is possible in processes like the iterative development life cycle, where the project realizes requirements over several iterations. Using interproject estimates in this way offers information for tracking current project performance. This ability to compare estimates with the actual outcome is crucial if estimation techniques are to improve.

Table 1. Reported cost overrun.

Cost overruns (percentage)	Respondents reporting this level of overrun (percentage)
Under 20	15.5
20 to 50	31.5
51 to 100	29.6
101 to 200	10.2
201 to 400	8.8
Over 400	4.4

Table 2. Reported time overrun.

Time overruns (percentage)	Respondents reporting this level of overrun (percentage)
Under 20	13.9
20 to 50	18.3
51 to 100	20.0
101 to 200	35.5
201 to 400	11.2
Over 400	1.1

Within the iterative process, the project realizes its requirements over several iterations. Each iteration's structure follows the familiar waterfall disciplines of requirements, analysis, design, code, and test. With the project decomposed into iterations, it is much easier to determine progress and tolerate change. For an introduction to the iterative development process see Philippe Kruchten's book on the Rational Unified Process (*The Rational Unified Process: An Introduction*, Addison-Wesley, 2000).

When using the iterative process, it becomes possible to refine some estimates during the project. This *intraproject refinement* provides a fine-grained method of estimating, based on actual experience within the project. This in turn leads to increased awareness of the project's state throughout its life cycle.

One measure of an estimation model's success is how well successive estimates approximate actual project performance. Improving estimates indicate that the model is converging and has a degree of stability in that successive estimates modify the estimation process and provide new estimates of increased accuracy. Diverging successive estimates would characterize a poor model. If this were the case, you would have to redesign the model.

Figure 1 shows the model development life cycle used to refine the estimation technique. Using this form of iterative refinement, you compare successive models with reality, modifying the model to provide improved estimates.

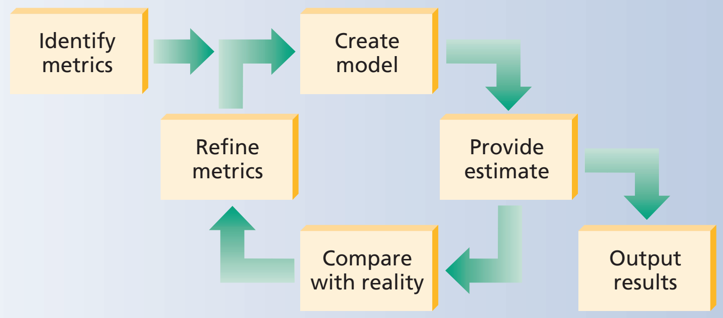
CANDIDATE FOR MEASUREMENT

As stated earlier, estimations based on project requirements are an attractive candidate for modeling because these requirements are readily available and easily assessable at the project's end. However, because project requirements take many forms, it is important to select a requirement format that will allow individuals to create requirements of consistent quality, across all projects. This maintenance of quality is difficult to achieve using traditional methods, because the requirement format differs among individuals and businesses. Fortunately, one method that reduces variability in quality and increases format consistency is the use case, which Alistair Cockburn effectively describes in *Writing Effective Use Cases* (Addison-Wesley, 2000).

A *use case* records a functional requirement in the form of a dialogue between the user and the application. Each use case focuses on the business, covers functional requirements, is easily read by the business user (because it is written in business terms), and has a generally accepted format.

The use-case-based estimation model aims to

Figure 1. Model development life cycle.



When creating a model for estimation, do not try and create a complex and detailed initial version. Rather, take a coarse-grained model and refine it over several iterations. In this way, the model will evolve to reflect the specific needs of your business and not become a compromised, generic model.

Figure 2. Use-case-based project estimation.

ClearTime UseCase Estimation

Date 22/05/2002

		How long	How long	How			Developer	Customer		
		Developer	Tester	Difficult	Complexity	Split	Priority	Priority	Order	Iteration
	edit timesheet	4	2	5	y	n	4	6	4	
	stop activity	2	1	3	n	n	2	2	2	1
	start activity	3	2	4	n	n	1	1	1	1
X	maintain favorites	4	2	6	n	n	5	3	5	
	maintain options	1	1.5	2	n	n	8	7	7	
	maintain activity sources	2	2	2	n	n	7	9	8	
	refresh activities	5		8	y	n	3	4	3	2
X	automatic time vault syn	4	2	8	y	n	6	5	6	
	show reminders	2	1	3	n	n	9	8	9	
	Total	19	9.5							

X = not in this build

Breaking the project down into functional areas makes estimating the time to realize the whole project easier. Here, the functional areas take the form of use cases, which are the elements for estimation. (The use cases are the rows in the model.) The columns represent the various activities and risks associated with each use case, and the numbers in the first two columns are the estimated days each use case will require for full development and test. The totals in these two columns are the estimates for development and test. Two use cases were omitted from the planned build so are not included in the estimation (they are shown in gray).

- capture the experience of the specialists doing the estimation,
- include disparate stakeholders in the decision making, and
- enable a consensus on the agreed timescale.

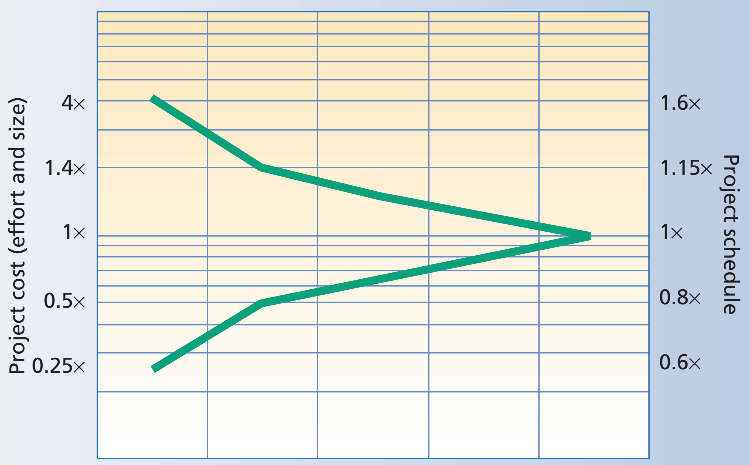
This method not only increases the accuracy of measurement but also produces a sense of shared ownership of the model results. Figure 2 shows the format for a current use-case-based estimation model.

Leveraging staff expertise

The model in Figure 2 serves two main purposes. First, it uses the expertise of the development and test staff to estimate their effort across the various functional requirements. A use case encompasses a discrete and significant proportion of the application's functionality, therefore it is easier to estimate effort using these large functional chunks. This is because the uncertainty associated with each use case is well within the overall error expected for this stage of the project. Overall errors of 50 to 75 percent are common for early project estimates and, as the *Chaos Report* indicated earlier, estimation error can be as much as 200 percent. The important point is that without measurement, the errors in the estimates are unknown and can be significantly larger than anyone might expect. Without measurement, there is no history, so it becomes impossible to gauge estimate accuracy for future projects.

Figure 3 shows the established error curve associated with project estimates; the error is a function of project unknowns. It is natural to expect that estimates will improve as the project matures, because developers know more about the product requirements, and project staff members are aware of the issues as well as their abilities. The generic project estimation convergence curve in Figure 3 shows that increasing accuracy comes through increased knowledge.

Figure 3. Estimate convergence graph.



The graph, from Steve McConnell's *Rapid Development* (Microsoft Press, 1996), shows that estimates improve with knowledge. Here, it is evident that as the project progresses along its life cycle, the estimates improve in a nonlinear fashion. This graph is based on the waterfall process but can still be valid for iterative development. Reprinted with permission; all rights reserved. © Steve McConnell, 1996.

Assessing risk and facilitating agreement

Secondly, the model is helpful in assessing the risk associated with each use case; it facilitates agreement among stakeholders on the content of each iteration. When assigning use cases to iterations, project managers should take several considerations into account: the degree of risk associated with a use case, its importance to the project, and the resources required to complete its implementation. It is preferable to assign difficult use cases to earlier iterations because the highest risks will then reside in the first iterations, and success in these iterations will significantly reduce the project's overall risk. This phenomenon is a product of iterative software development; use-case-based estimation aids the identification and assignment of use cases to iterations and therefore promotes early risk mitigation.

INTRAPROJECT REFINEMENT

In a waterfall process, comparisons typically take place at the end of the project. However, in an iterative process, project managers can make comparisons at the end of each iteration. The result is a measure of the initial estimates' accuracies. It is then possible to evaluate any risks associated with poor initial estimates and amend each iteration's content as the project progresses. This intraproject refinement reduces risk to the project because project managers can reschedule tasks based on quantifiable evidence as opposed to using just intuition.

Figure 4 shows how to compare the captured actual effort with the original estimates. For clarity, I call the unit of measure for recording estimates a *developer day*. I coined this term to distinguish estimated days of effort from actual days because, when estimating effort, developers do not instinctively account for external project considerations, such as milestone reviews and rework. By comparing successive iterations of developer days with actual days, I can derive a unit scaling factor (USF). This factor shows the error in the estimates over the project's life. Figure 4 shows an example of how to calculate USF.

Figure 4. Worksheet to calculate unit scaling factor.

ClearTime Actual Time Date 19/07/2002							
Iteration	Functionality	Iteration					
		1	2	3	4	5	6
3	edit timesheet			6.5	6.5	6.5	6.5
1	stop activity	5	5	5	5	5	5
1	start activity	7	7	7	7	7	7
X	maintain favorites	0	0	0	0	0	0
6	maintain options						2
5	maintain activity sources					25	25
2	refresh activities		13.5	13.5	13.5	13.5	13.5
X	automatic time vault syn	0	0	0	0	0	0
4	show reminders				3.5	3.5	3.5
4	Maintain Tasks				3.5	3.5	3.5
6	Time Analysis						
6	Install ClearTime						
	Cumulative Total Unit	12	25.5	32	39	64	66

Unit Scaling Factor	1.20	1.28	1.14	1.03	1.52	1.50
Projected Time for project in days	52.80	56.10	50.29	45.16	67.05	66.00

This worksheet makes successive comparisons between the estimated and actual times required to complete an iteration. It also shows the predicted project completion time based on each iteration's USF value (estimated effort divided by the actual effort). At the end of each iteration, I derive the projected time by multiplying the original estimate by the calculated USF. Looking across this row, note how the project slips by a week in a day! This chart could also potentially point to areas of architectural instability by detecting whether USF increases over time.

COMMENTS ON USAGE

This model was created to provide a quantitative estimation tool; it uses information readily available during the project's inception phase. Reestimating at the end of each iteration improves the accuracy of estimates. As the project progresses, it improves because developers gain a greater understanding of the requirements and also increase the accuracy of their estimates. The result is better feedback on estimation performance (at least once every iteration) and the ability to apply lessons learned to the next iteration. This form of reestimation is a feature of iterative development; this model simplifies such reestimations.

When producing the first-cut estimate, the entire development team became involved the process. We thought that this would create consensus and produce a process that tolerated the biases of individual team members.

We have also found that large USF values indicate potential problem areas. In those cases, the USF values also provide information relevant to the reordering of contents in each iteration.

We are now applying the techniques we've learned creating this model to produce estimates for contract bidding, an area notorious for a qualitative approach to estimating cost and effort.

Project estimation cannot be effective without measurement. The simple techniques described here allow you to document experience and use it to improve future measurement. The model provides information for project managers during inception and throughout the project life cycle.

It's important to resist seeing estimates as accurate point values. An estimate reflects a project manager's current understanding of the project and its perceived risks at that point in time. Often, project managers provide estimates as a range of values, using the upper and lower values to reflect their uncertainty. Unfortunately, the recipient of this estimation invariably selects a value that suits himself; he fails to grasp the meaning in the estimate range until it is too late. ■

***Ray Ashman** is a senior consultant at FMI Solutions. Contact him at ray.ashman@fmisolutions.com.*