

# What Is an “Estimate”?

*It is very difficult to make a vigorous, plausible, and job-risking defense of an estimate that is derived by no quantitative method, supported by little data, and certified chiefly by the hunches of the managers.*

—Fred Brooks

You might think you already know what an estimate is. My goal by the end of this chapter is to convince you that an estimate is different from what most people think. A *good* estimate is even more different.

Here is a dictionary definition of *estimate*: 1. A tentative evaluation or rough calculation. 2. A preliminary calculation of the cost of a project. 3. A judgment based upon one’s impressions; opinion. (Source: *The American Heritage Dictionary*, Second College Edition, 1985.)

Does this sound like what you are asked for when you’re asked for an estimate? Are you asked for a *tentative* or *preliminary* calculation—that is, do you expect that you can change your mind later?

Probably not. When executives ask for an “estimate,” they’re often asking for a commitment or for a plan to meet a target. The distinctions between estimates, targets, and commitments are critical to understanding what an estimate is, what an estimate is not, and how to make your estimates better.

## 1.1 Estimates, Targets, and Commitments

Strictly speaking, the dictionary definition of *estimate* is correct: an estimate is a prediction of how long a project will take or how much it will cost. But estimation on software projects interplays with business targets, commitments, and control.

A *target* is a statement of a desirable business objective. Examples include the following:

- “We need to have Version 2.1 ready to demonstrate at a trade show in May.”
- “We need to have this release stabilized in time for the holiday sales cycle.”
- “These functions need to be completed by July 1 so that we’ll be in compliance with government regulations.”
- “We must limit the cost of the next release to \$2 million, because that’s the maximum budget we have for that release.”

Businesses have important reasons to establish targets independent of software estimates. But the fact that a target is desirable or even mandatory does not necessarily mean that it is achievable.

While a target is a description of a desirable business objective, a *commitment* is a promise to deliver defined functionality at a specific level of quality by a certain date. A commitment can be the same as the estimate, or it can be more aggressive or more conservative than the estimate. In other words, do not assume that the commitment has to be the same as the estimate; it doesn't.

**Tip #1**

Distinguish between estimates, targets, and commitments.

## 1.2 Relationship Between Estimates and Plans

Estimation and planning are related topics, but estimation is not planning, and planning is not estimation. Estimation should be treated as an unbiased, analytical process; planning should be treated as a biased, goal-seeking process. With estimation, it's hazardous to want the estimate to come out to any particular answer. The goal is accuracy; the goal is not to seek a particular result. But the goal of planning is to seek a particular result. We deliberately (and appropriately) bias our plans to achieve specific outcomes. We plan specific means to reach a specific end.

Estimates form the foundation for the plans, but the plans don't have to be the same as the estimates. If the estimates are dramatically different from the targets, the project plans will need to recognize that gap and account for a high level of risk. If the estimates are close to the targets, then the plans can assume less risk.

Both estimation and planning are important, but the fundamental differences between the two activities mean that combining the two tends to lead to poor estimates *and* poor plans. The presence of a strong planning target can lead to substitution of the target for an analytically derived estimate; project members might even refer to the target as an "estimate," giving it a halo of objectivity that it doesn't deserve.

Here are examples of planning considerations that depend in part on accurate estimates:

- Creating a detailed schedule
- Identifying a project's critical path
- Creating a complete work breakdown structure
- Prioritizing functionality for delivery
- Breaking a project into iterations

Accurate estimates support better work in each of these areas (and Chapter 21, "Estimating Planning Parameters," goes into more detail on these topics).

## 1.3 Communicating about Estimates, Targets, and Commitments

One implication of the close and sometimes confusing relationship between estimation and planning is that project stakeholders sometimes miscommunicate about these activities. Here’s an example of a typical miscommunication:

*EXECUTIVE: How long do you think this project will take? We need to have this software ready in 3 months for a trade show. I can’t give you any more team members, so you’ll have to do the work with your current staff. Here’s a list of the features we’ll need.*

*PROJECT LEAD: OK, let me crunch some numbers, and get back to you.*

Later...

*PROJECT LEAD: We’ve estimated the project will take 5 months.*

*EXECUTIVE: Five months!? Didn’t you hear me? I said we needed to have this software ready in 3 months for a trade show!*

In this interaction, the project lead will probably walk away thinking that the executive is irrational, because he is asking for the team to deliver 5 months’ worth of functionality in 3 months. The executive will walk away thinking that the project lead doesn’t “get” the business reality, because he doesn’t understand how important it is to be ready for the trade show in 3 months.

Note in this example that the executive was not really asking for an estimate; he was asking the project lead to come up with a *plan* to hit a *target*. Most executives don’t have the technical background that would allow them to make fine distinctions between estimates, targets, commitments, and plans. So it becomes the technical leader’s responsibility to translate the executive’s request into more specific technical terms.

Here’s a more productive way that the interaction could go:

*EXECUTIVE: How long do you think this project will take? We need to have this software ready in 3 months for a trade show. I can’t give you any more team members, so you’ll have to do the work with your current staff. Here’s a list of the features we’ll need.*

*PROJECT LEAD: Let me make sure I understand what you’re asking for. Is it more important for us to deliver 100% of these features, or is it more important to have something ready for the trade show?*

*EXECUTIVE: We have to have something ready for the trade show. We'd like to have 100% of those features if possible.*

*PROJECT LEAD: I want to be sure I follow through on your priorities as best I can. If it turns out that we can't deliver 100% of the features by the trade show, should we be ready to ship what we've got at trade show time, or should we plan to slip the ship date beyond the trade show?*

*EXECUTIVE: We have to have something for the trade show, so if push comes to shove, we have to ship something, even if it isn't 100% of what we want.*

*PROJECT LEAD: OK, I'll come up with a plan for delivering as many features as we can in the next 3 months.*

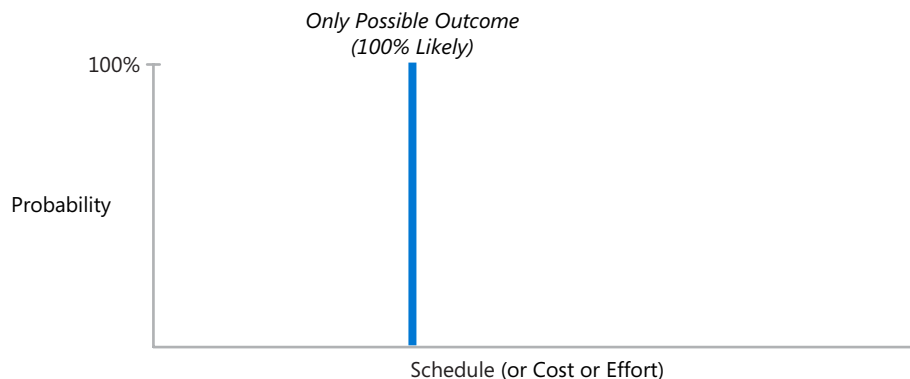
**Tip #2**

When you're asked to provide an estimate, determine whether you're supposed to be estimating or figuring out how to hit a target.

## 1.4 Estimates as Probability Statements

If three-quarters of software projects overrun their estimates, the odds of any given software project completing on time and within budget are not 100%. Once we recognize that the odds of on-time completion are not 100%, an obvious question arises: "If the odds aren't 100%, what are they?" This is one of the central questions of software estimation.

Software estimates are routinely presented as single-point numbers, such as "This project will take 14 weeks." Such simplistic single-point estimates are meaningless because they don't include any indication of the probability associated with the single point. They imply a probability as shown in Figure 1-1—the only possible outcome is the single point given.



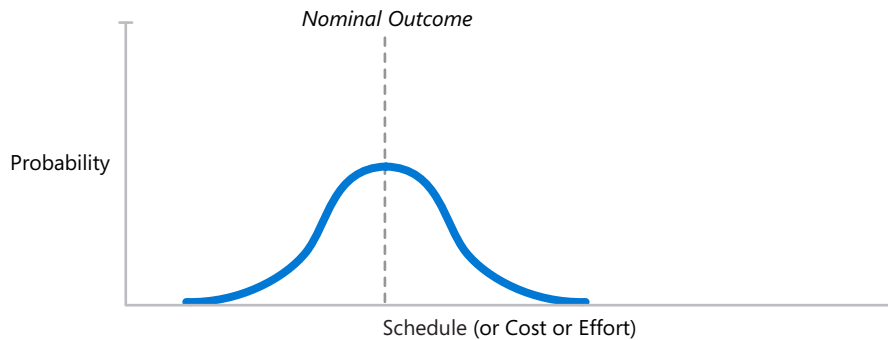
**Figure 1-1** Single-point estimates assume 100% probability of the actual outcome equaling the planned outcome. This isn't realistic.

A single-point estimate is usually a target masquerading as an estimate. Occasionally, it is the sign of a more sophisticated estimate that has been stripped of meaningful probability information somewhere along the way.

**Tip #3**

When you see a single-point “estimate,” ask whether the number is an estimate or whether it’s really a target.

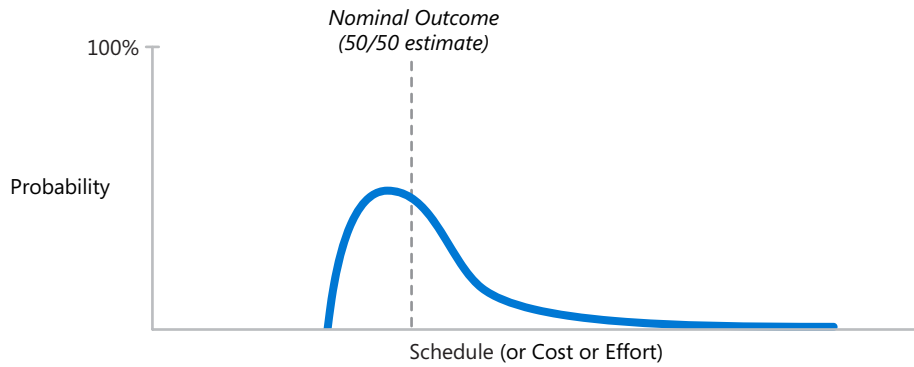
Accurate software estimates acknowledge that software projects are assailed by uncertainty from all quarters. Collectively, these various sources of uncertainty mean that project outcomes follow a probability distribution—some outcomes are more likely, some outcomes are less likely, and a cluster of outcomes in the middle of the distribution are most likely. You might expect that the distribution of project outcomes would look like a common bell curve, as shown in Figure 1-2.



**Figure 1-2** A common assumption is that software project outcomes follow a bell curve. This assumption is incorrect because there are limits to how efficiently a project team can complete any given amount of work.

Each point on the curve represents the chance of the project finishing exactly on that date (or costing exactly that much). The area under the curve adds up to 100%. This sort of probability distribution acknowledges the possibility of a broad range of outcomes. But the assumption that the outcomes are symmetrically distributed about the mean (average) is not valid. There is a limit to how well a project can be conducted, which means that the tail on the left side of the distribution is truncated rather than extending as far to the left as it does in the bell curve. And while there is a limit to how well a project can go, there is no limit to how poorly a project can go, and so the probability distribution does have a very long tail on the right.

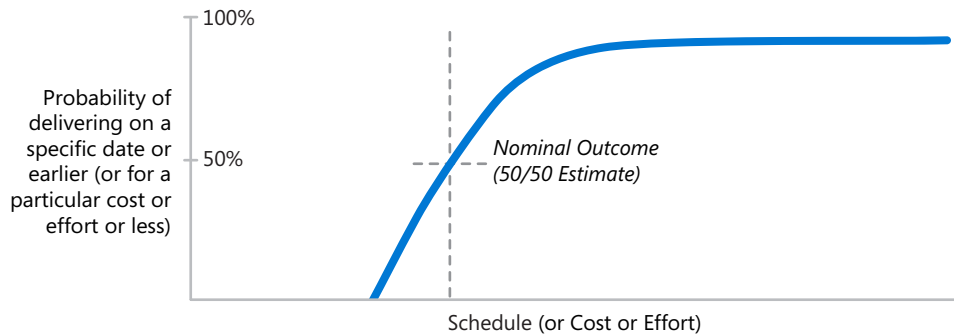
Figure 1-3 provides an accurate representation of the probability distribution of a software project’s outcomes.



**Figure 1-3** An accurate depiction of possible software project outcomes. There is a limit to how well a project can go but no limit to how many problems can occur.

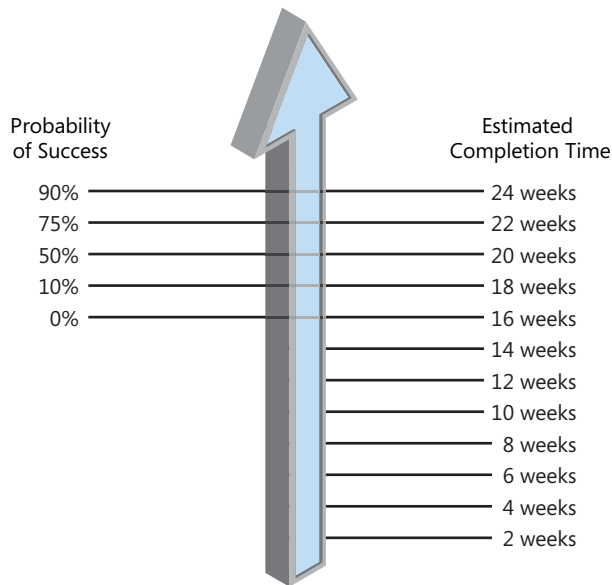
The vertical dashed line shows the “nominal” outcome, which is also the “50/50” outcome—there’s a 50% chance that the project will finish better and a 50% chance that it will finish worse. Statistically, this is known as the “median” outcome.

Figure 1-4 shows another way of expressing this probability distribution. While Figure 1-3 showed the probabilities of delivering on specific dates, Figure 1-5 shows the probabilities of delivering on each specific date *or earlier*.



**Figure 1-4** The probability of a software project delivering on or before a particular date (or less than or equal to a specific cost or level of effort).

Figure 1-5 presents the idea of probabilistic project outcomes in another way. As you can see from the figure, a naked estimate like “18 weeks” leaves out the interesting information that 18 weeks is only 10% likely. An estimate like “18 to 24 weeks” is more informative and conveys useful information about the likely range of project outcomes.



**Figure 1-5** All single-point estimates are associated with a probability, explicitly or implicitly.

#### Tip #4

When you see a single-point estimate, that number's probability is not 100%. Ask what the probability of that number is.

You can express probabilities associated with estimates in numerous ways. You could use a “percent confident” attached to a single-point number: “We’re 90% confident in the 24-week schedule.” You could describe estimates as best case and worst case, which implies a probability: “We estimate a best case of 18 weeks and a worst case of 24 weeks.” Or you could simply state the estimated outcome as a range rather than a single-point number: “We’re estimating 18 to 24 weeks.” The key point is that all estimates include a probability, whether the probability is stated or implied. An explicitly stated probability is one sign of a good estimate.

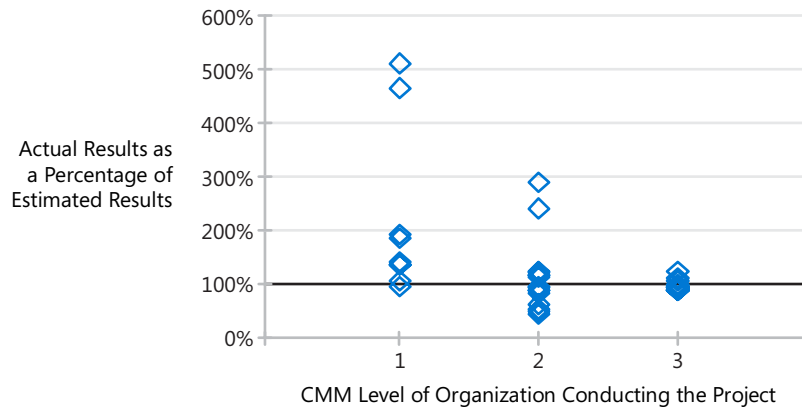
You can make a commitment to the optimistic end or the pessimistic end of an estimation range—or anywhere in the middle. The important thing is for you to know where in the range your commitment falls so that you can plan accordingly.

## 1.5 Common Definitions of a “Good” Estimate

The answer to the question of what an “estimate” is still leaves us with the question of what a *good* estimate is. Estimation experts have proposed various definitions of a good estimate. Capers Jones has stated that accuracy with  $\pm 10\%$  is possible, but only on well-controlled projects (Jones 1998). Chaotic projects have too much variability to achieve that level of accuracy.

In 1986, Professors S.D. Conte, H.E. Dunsmore, and V.Y. Shen proposed that a good estimation approach should provide estimates that are within 25% of the actual results 75% of the time (Conte, Dunsmore, and Shen 1986). This evaluation standard is the most common standard used to evaluate estimation accuracy (Stutzke 2005).

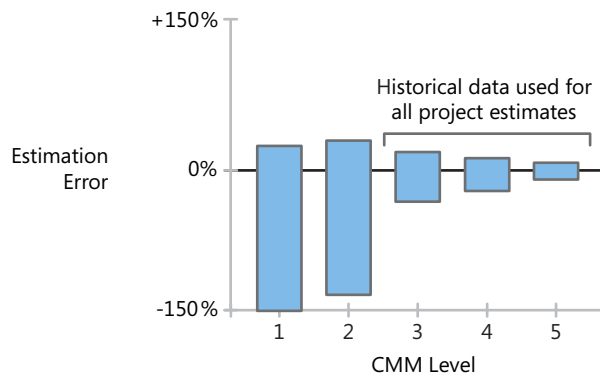
Numerous companies have reported estimation results that are close to the accuracy Conte, Dunsmore, and Shen and Jones have suggested. Figure 1-6 shows actual results compared to estimates from a set of U.S. Air Force projects.



Source: "A Correlational Study of the CMM and Software Development Performance" (Lawlis, Flowe, and Thordahl 1995).

**Figure 1-6** Improvement in estimation of a set of U.S. Air Force projects. The predictability of the projects improved dramatically as the organizations moved toward higher CMM levels.<sup>1</sup>

Figure 1-7 shows results of a similar improvement program at the Boeing Company.

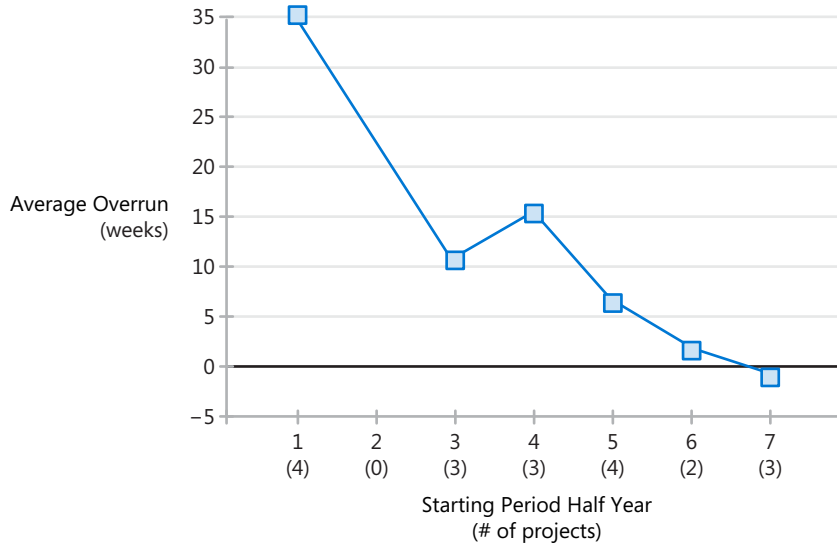


**Figure 1-7** Improvement in estimation at the Boeing Company. As with the U.S. Air Force projects, the predictability of the projects improved dramatically at higher CMM levels.

<sup>1</sup> The CMM (Capability Maturity Model) is a system defined by the Software Engineering Institute to assess the effectiveness of software organizations.



A final, similar example, shown in Figure 1-8, comes from improved estimation results at Schlumberger.



**Figure 1-8** Schlumberger improved its estimation accuracy from an average overrun of 35 weeks to an average underrun of 1 week.

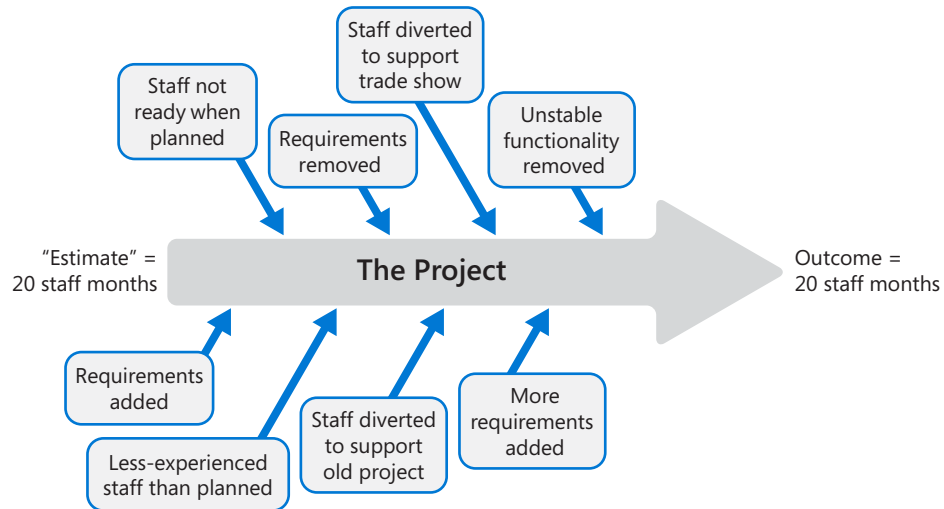
One of my client companies delivers 97% of its projects on time and within budget. Telcordia reported that it delivers 98% of its projects on time and within budget (Pitterman 2000). Numerous other companies have published similar results (Putnam and Myers 2003). Organizations are creating good estimates by both Jones’s definition and Conte, Dunsmore, and Shen’s definition. However, an important concept is missing from both of these definitions—namely, that accurate estimation results cannot be accomplished through estimation practices alone. They must also be supported by effective project control.

## 1.6 Estimates and Project Control

Sometimes when people discuss software estimation they treat estimation as a purely predictive activity. They act as though the estimate is made by an impartial estimator, sitting somewhere in outer space, disconnected from project planning and prioritization activities.

In reality, there is little that is pure about software estimation. If you ever wanted an example of Heisenberg’s Uncertainty Principle applied to software, estimation would be it. (Heisenberg’s Uncertainty Principle is the idea that the mere act of observing a thing changes it, so you can never be sure how that thing would behave if you weren’t observing it.) Once we make an estimate and, on the basis of that estimate, make a

commitment to deliver functionality and quality by a particular date, then we *control* the project to meet the target. Typical project control activities include removing non-critical requirements, redefining requirements, replacing less-experienced staff with more-experienced staff, and so on. Figure 1-9 illustrates these dynamics.



**Figure 1-9** Projects change significantly from inception to delivery. Changes are usually significant enough that the project delivered is not the same as the project that was estimated. Nonetheless, if the outcome is similar to the estimate, we say the project met its estimate.

In addition to project control activities, projects are often affected by unforeseen external events. The project team might need to create an interim release to support a key customer. Staff might be diverted to support an old project, and so on.

Events that happen during the project nearly always invalidate the assumptions that were used to estimate the project in the first place. Functionality assumptions change, staffing assumptions change, and priorities change. It becomes impossible to make a clean analytical assessment of whether the project was estimated accurately—because the software project that was ultimately delivered is not the project that was originally estimated.

In practice, if we deliver a project with about the level of functionality intended, using about the level of resources planned, in about the time frame targeted, then we typically say that the project “met its estimates,” despite all the analytical impurities implicit in that statement.

Thus, the criteria for a “good” estimate cannot be based on its predictive capability, which is impossible to assess, but on the estimate’s ability to support project success, which brings us to the next topic: the Proper Role of Estimation.

## 1.7 Estimation’s Real Purpose

Suppose you’re preparing for a trip and deciding which suitcase to take. You have a small suitcase that you like because it’s easy to carry and will fit into an airplane’s overhead storage bin. You also have a large suitcase, which you don’t like because you’ll have to check it in and then wait for it at baggage claim, lengthening your trip. You lay your clothes beside the small suitcase, and it appears that they will almost fit. What do you do? You might try packing them very carefully, not wasting any space, and hoping they all fit. If that approach doesn’t work, you might try stuffing them into the suitcase with brute force, sitting on the top and trying to squeeze the latches closed. If that still doesn’t work, you’re faced with a choice: leave a few clothes at home or take the larger suitcase.

Software projects face a similar dilemma. Project planners often find a gap between a project’s business targets and its estimated schedule and cost. If the gap is small, the planner might be able to control the project to a successful conclusion by preparing extra carefully or by squeezing the project’s schedule, budget, or feature set. If the gap is large, the project’s targets must be reconsidered.

The primary purpose of software estimation is not to predict a project’s outcome; it is to determine whether a project’s targets are realistic enough to allow the project to be controlled to meet them. Will the clothes you want to take on your trip fit into the small suitcase or will you be forced to take the large suitcase? Can you take the small suitcase if you make minor adjustments? Executives want the same kinds of answers. They often don’t want an accurate estimate that tells them that the desired clothes won’t fit into the suitcase; they want a plan for making as many of the clothes fit as possible.

Problems arise when the gap between the business targets and the schedule and effort needed to achieve those targets becomes too large. I have found that if the initial target and initial estimate are within about 20% of each other, the project manager will have enough maneuvering room to control the feature set, schedule, team size, and other parameters to meet the project’s business goals; other experts concur (Boehm 1981, Stutzke 2005). If the gap between the target and what is actually needed is too large, the manager will not be able to control the project to a successful conclusion by making minor adjustments to project parameters. No amount of careful packing or sitting on the suitcase will squeeze all your clothes into the smaller suitcase, and you’ll have to take the larger one, even if it isn’t your first choice, or you’ll have to leave some clothes behind. The project targets will need to be brought into better alignment with reality before the manager can control the project to meet its targets.

Estimates don’t need to be perfectly accurate as much as they need to be *useful*. When we have the combination of accurate estimates, good target setting, and good planning and control, we can end up with project results that are close to the

“estimates.” (As you’ve guessed, the word “estimate” is in quotation marks because the project that was estimated is not the same project that was ultimately delivered.)

These dynamics of changing project assumptions are a major reason that this book focuses more on the art of estimation than on the science. Accuracy of  $\pm 5\%$  won’t do you much good if the project’s underlying assumptions change by 100%.

## 1.8 A Working Definition of a “Good Estimate”

With the background provided in the past few sections, we’re now ready to answer the question of what qualifies as a good estimate.

*A good estimate is an estimate that provides a clear enough view of the project reality to allow the project leadership to make good decisions about how to control the project to hit its targets..*

This definition is the foundation of the estimation discussion throughout the rest of this book.

## Additional Resources

Conte, S. D., H. E. Dunsmore, and V. Y. Shen. *Software Engineering Metrics and Models*. Menlo Park, CA: Benjamin/Cummings, 1986. Conte, Dunsmore, and Shen’s book contains the definitive discussion of evaluating estimation models. It discusses the “within 25% of actual 75% of the time” criteria, as well as many other evaluation criteria.

DeMarco, Tom. *Controlling Software Projects*. New York, NY: Yourdon Press, 1982. DeMarco discusses the probabilistic nature of software projects.

Stutzke, Richard D. *Estimating Software-Intensive Systems*. Upper Saddle River, NJ: Addison-Wesley, 2005. Appendix C of Stutzke’s book contains a summary of measures of estimation accuracy.

# How Good an Estimator Are You?

*The process is called estimation, not exactimation.*

*—Phillip Armour*

Now that you know what a good estimate is, how good an estimator are you? The following section will help you find out.

## 2.1 A Simple Estimation Quiz

Table 2-1, appearing on the following page, contains a quiz designed to test your estimation skills. Please read and observe the following directions carefully:

For each question, fill in the upper and lower bounds that, in your opinion, give you a 90% chance of including the correct value. Be careful not to make your ranges either too wide or too narrow. Make them wide enough so that, in your best judgment, the ranges give you a 90% chance of including the correct answer. Please do not research any of the answers—this quiz is intended to assess your estimation skills, not your research skills. You must fill in an answer for each item; an omitted item will be scored as an incorrect item. Please limit your time on this exercise to 10 minutes.

(Also, you might want to photocopy the quiz before taking it so that the next person who reads this book can take it, too.)

The correct answers to this exercise (the latitude of Shanghai, for example) are listed in Appendix B in the back of the book. Give yourself one point for each of your ranges that includes the related correct answer.

**Table 2-1** How Good an Estimator Are You?

[Low Estimate – High Estimate]	Description
[ _____ – _____ ]	Surface temperature of the Sun
[ _____ – _____ ]	Latitude of Shanghai
[ _____ – _____ ]	Area of the Asian continent
[ _____ – _____ ]	The year of Alexander the Great's birth
[ _____ – _____ ]	Total value of U.S. currency in circulation in 2004
[ _____ – _____ ]	Total volume of the Great Lakes
[ _____ – _____ ]	Worldwide box office receipts for the movie <i>Titanic</i>
[ _____ – _____ ]	Total length of the coastline of the Pacific Ocean
[ _____ – _____ ]	Number of book titles published in the U.S. since 1776
[ _____ – _____ ]	Heaviest blue whale ever recorded

Source: Inspired by a similar quiz in *Programming Pearls*, Second Edition (Bentley 2000).

This quiz is from *Software Estimation* by Steve McConnell (Microsoft Press, 2006) and is © 2006 Steve McConnell. All Rights Reserved. Permission to copy this quiz is granted provided that this copyright notice is included.

How did you do? (Don't feel bad. Most people do poorly on this quiz!) Please write your score here: \_\_\_\_\_

## 2.2 Discussion of Quiz Results

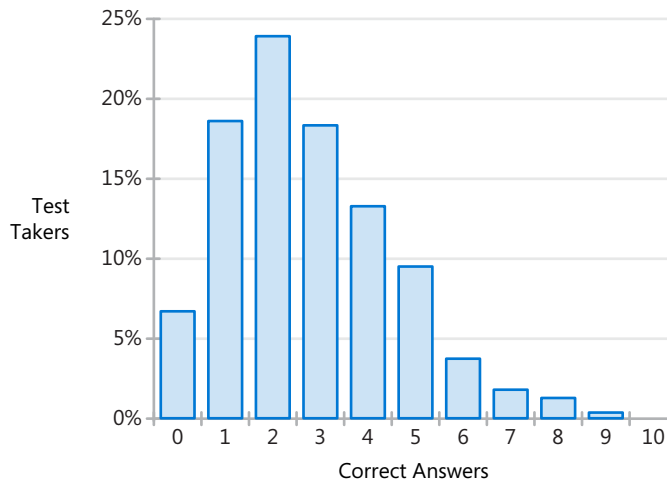
The purpose of this quiz is not to determine whether you know when Alexander the Great was born or the latitude of Shanghai. Its purpose is to determine how well you understand your own estimation capabilities.

### How Confident Is “90% Confident”?

The directions above are specific that the goal of the exercise is to estimate at the 90% confidence level. Because there are 10 questions in the quiz, if you were truly estimating at the 90% confidence level, you should have gotten about 9 answers correct.<sup>1</sup>

If you were cautious, you made your ranges conservatively wide, in which case you scored 10 out of 10 correctly. If you were just a little hasty, you made your ranges narrower than they needed to be, in which case you scored 7 or 8 out of 10 correctly. I've given this quiz to hundreds of estimators. Figure 2-1 shows the results from the most recent 600 people who have taken the quiz.

<sup>1</sup> The mathematics behind “90% confident” are a little complicated. If you were really estimating with 90% confidence, you would have a 34.9% chance of getting 10 answers correct, 38.7% chance of getting 9 answers correct, and a 19.4% chance of getting 8 answers correct. In other words, you'd have a 93% chance of getting 8 or more answers correct.



**Figure 2-1** Results from administering the “How Good an Estimator Are You?” quiz. Most quiz-takers get 1–3 answers correct.

For the test takers whose results are shown in the figure, the average number of correct answers is 2.8. Only 2 percent of quiz takers score 8 or more answers correctly. No one has ever gotten 10 correct. I’ve concluded that most people’s intuitive sense of “90% confident” is really comparable to something closer to “30% confident.” Other studies have confirmed this basic finding (Zultner 1999, Jørgensen 2002).

Similarly, I’ve seen numerous project teams present “90% confident” schedules, and I’ve frequently seen those project teams overrun their “90% confident” schedules—more often than not. If those schedules represented a true 90% confidence, the project teams would overrun them only about 1 time out of 10.

I’ve concluded that specific percentages such as “90%” are not meaningful unless they are supported through some kind of statistical analysis. Otherwise those specific percentages are just wishful thinking. How to get to a *real* 90% confidence level will be discussed later in the book.

**Tip #5**

Don’t provide “percentage confident” estimates (especially “90% confident”) unless you have a quantitatively derived basis for doing so.

If you didn’t take the quiz earlier in this chapter, this would be a good time to go back and take it. I think you’ll be surprised at how few answers you get correct even after reading this explanation.

## How Wide Should You Make Your Ranges?

When I find the rare person who gets 7 or 8 answers correct, I ask “How did you get that many correct?” The typical response? “I made my ranges too wide.”

My response is, “No, you didn’t! You didn’t make your ranges wide enough!” If you got only 7 or 8 correct, your ranges were still too narrow to include the correct answer as often as you should have.

We are conditioned to believe that estimates expressed as narrow ranges are more accurate than estimates expressed as wider ranges. We believe that wide ranges make us appear ignorant or incompetent. The opposite is usually the case. (Of course, narrow ranges are desirable in the cases when the underlying data supports them.)

**Tip #6**

Avoid using artificially narrow ranges. Be sure the ranges you use in your estimates don’t misrepresent your confidence in your estimates.

## Where Does Pressure to Use Narrow Ranges Come From?

When you were taking the quiz, did you feel pressure to make your ranges wider? Or did you feel pressure to make your ranges narrower? Most people report that they feel pressure to make the ranges as narrow as possible. But if you go back and review the instructions, you’ll find that they do not encourage you to use narrow ranges. Indeed, I was careful to state that you should make your ranges neither too wide nor too narrow—just wide enough to give you a 90% confidence in including the correct answer.

After discussing this issue with hundreds of developers and managers, I’ve concluded that much of the pressure to provide narrow ranges is self-induced. Some of the pressure comes from people’s sense of professional pride. They believe that narrow ranges are a sign of a better estimate, even though that isn’t the case. And some of the pressure comes from experiences with bosses or customers who insisted on the use of overly narrow ranges.

This same self-induced pressure has been found in interactions between customers and estimators. Jørgensen and Sjøberg reported that information about customers’ expectations exerts strong influence on estimates and that estimators are typically not conscious of the degree to which their estimates are affected (Jørgensen and Sjøberg 2002).

**Tip #7**

If you are feeling pressure to make your ranges narrower, verify that the pressure actually is coming from an external source and not from yourself.



For those cases in which the pressure truly is coming from an external source, Chapter 22, “Estimate Presentation Styles,” and Chapter 23, “Politics, Negotiation, and Problem Solving,” discuss how to deal with that pressure.

## How Representative Is This Quiz of Real Software Estimates?

In software, you aren’t often asked to estimate the volume of the Great Lakes or the surface temperature of the Sun. Is it reasonable to expect you to be able to estimate the amount of U.S. currency in circulation or the number of books published in the U.S., especially if you’re not in the U.S.?

Software developers are often asked to estimate projects in unfamiliar business areas, projects that will be implemented in new technologies, the impacts of new programming tools on productivity, the productivity of unidentified personnel, and so on. Estimating in the face of uncertainty is business as usual for software estimators. The rest of this book explains how to succeed in such circumstances.