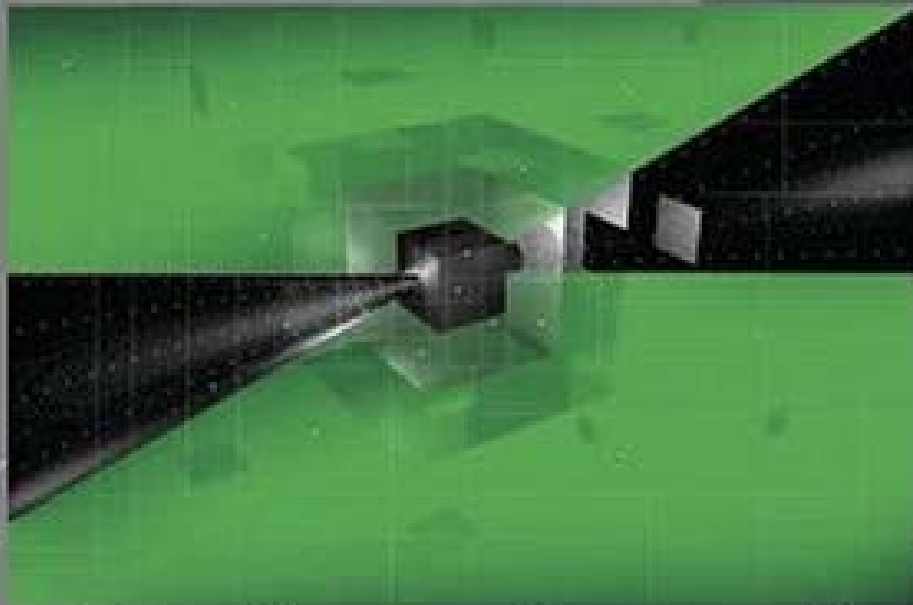


Microsoft

BEST PRACTICES

SOFTWARE ESTIMATION



Demystifying the Black Art

Steve McConnell

Two-time winner of Software Development magazine's Jolt Award

This material is excerpted from *Software Estimation: Demystifying the Black Art* by Steve McConnell (Redmond, Wa.: Microsoft Press).

© 2006 All Rights Reserved.

Welcome

The most unsuccessful three years in the education of cost estimators appears to be fifth-grade arithmetic.

—Norman R. Augustine

Software estimation is not hard. Experts have been researching and writing about software estimation for four decades, and they have developed numerous techniques that support accurate software estimates. Creating accurate estimates is straightforward—once you understand how to create them. But not all estimation practices are intuitively obvious, and even smart people won't discover all the good practices on their own. The fact that someone is an expert developer doesn't make that person an expert estimator.

Numerous aspects of estimation are not what they seem. Many so-called estimation problems arise from misunderstanding what an “estimate” is or blurring other similar-but-not-identical concepts with estimation. Some estimation practices that seem intuitively useful don't produce accurate results. Complex formulas sometimes do more harm than good, and some deceptively simple practices produce surprisingly accurate results.

This book distills four decades of research and even more decades of hands-on experience to help developers, leads, testers, and managers become effective estimators. Learning about software estimation turns out to be generally useful because the influences that affect software estimates are the influences that affect software development itself.

Art vs. Science of Software Estimation

Software estimation research is currently focused on improving estimation techniques so that sophisticated organizations can achieve project results within $\pm 5\%$ of estimated results instead of within $\pm 10\%$. These techniques are mathematically intensive. Understanding them requires a strong math background and concentrated study. Using them requires number crunching well beyond what you can do on your hand calculator. These techniques work best when embodied in commercial software estimation tools. I refer to this set of practices as the *science of estimation*.

Meanwhile, the typical software organization is not struggling to improve its estimates from $\pm 10\%$ to $\pm 5\%$ accuracy. The typical software organization is struggling to avoid estimates that are incorrect by 100% or more. (The reasons for this are manifold and will be discussed in detail in Chapters 3 and 4.)

xvi **Welcome**

Our natural tendency is to believe that complex formulas like this:

$$\text{Effort} = 2.94 * (\text{KSLOC})^{[0.91 + 0.01 * \sum_{j=1}^5 SF_j]} * \prod_{i=1}^{17} EM_i$$

will always produce more accurate results than simple formulas like this:

$$\text{Effort} = \text{NumberOfRequirements} * \text{AverageEffortPerRequirement}$$

But complex formulas aren't necessarily better. Software projects are influenced by numerous factors that undermine many of the assumptions contained in the complex formulas of the science of estimation. Those dynamics will be explained later in this book. Moreover, most software practitioners have neither the time nor the inclination to learn the intensive math required to understand the science of estimation.

Consequently, this book emphasizes rules of thumb, procedures, and simple formulas that are highly effective and understandable to practicing software professionals. These techniques will not produce estimates that are accurate to within $\pm 5\%$, but they will reduce estimation error to about 25% or less, which turns out to be about as useful as most projects need, anyway. I call this set of techniques the *art of estimation*.

This book draws from both the art and science of software estimation, but its focus is on software estimation as an art.

Why This Book Was Written and Who It Is For

The literature on software estimation is widely scattered. Researchers have published hundreds of articles, and many of them are useful. But the typical practitioner doesn't have time to track down dozens of papers from obscure technical journals. A few previous books have described the science of estimation. Those books are 800–1000 pages long, require a good math background, and are targeted mainly at professional estimators—consultants or specialists who estimate large projects and do so frequently.

I wrote this book for developers, leads, testers, and managers who need to create estimates occasionally as one of their many job responsibilities. I believe that most practitioners want to improve the accuracy of their estimates but don't have the time to obtain a Ph.D. in software estimation. These practitioners struggle with practical issues like how to deal with the politics that surround the estimate, how to present an estimate so that it will actually be accepted, and how to avoid having someone change your estimate arbitrarily. If you are in this category, this book was written for you.

The techniques in this book apply to Internet and intranet development, embedded software, shrink-wrapped software, business systems, new development, legacy systems, large projects, small projects—essentially, to estimates for all kinds of software.

Key Benefits Of This Book

By focusing on the art of estimation, this book provides numerous important estimation insights:

- What an “estimate” is. (You might think you already know what an estimate is, but common usages of the term are inaccurate in ways that undermine effective estimation.)
- The specific factors that have made your past estimates less accurate than they could have been.
- Ways to distinguish a good estimate from a poor one.
- Numerous techniques that will allow *you personally* to create good estimates.
- Several techniques you can use to help *other people on your team* create good estimates.
- Ways that *your organization* can create good estimates. (There are important differences between personal techniques, group techniques, and organizational techniques.)
- Estimation approaches that work on agile projects, and approaches that work on traditional, sequential (plan-driven) projects.
- Estimation approaches that work on small projects and approaches that work on large projects.
- How to navigate the shark-infested political waters that often surround software estimation.

In addition to gaining a better understanding of estimation concepts, the practices in this book will help you estimate numerous specific attributes of software projects, including:

- New development work, including schedule, effort, and cost
- Schedule, effort, and cost of legacy systems work
- How many features you can deliver within a specific development iteration
- The amount of functionality you can deliver for a whole project when schedule and team size are fixed
- Proportions of different software development activities needed, including how much management work, requirements, construction, testing, and defect correction will be needed
- Planning parameters, such as tradeoffs between cost and schedule, best team size, amount of contingency buffer, ratio of developers to testers, and so on

xviii **Welcome**

- Quality parameters, including time needed for defect correction work, defects that will remain in your software at release time, and other factors
- Practically anything else you want to estimate

In many cases, you'll be able to put this book's practices to use right away.

Most practitioners will not need to go any further than the concepts described in this book. But understanding the concepts in this book will lay enough groundwork that you'll be able to graduate to more mathematically intensive approaches later on, if you want to.

What This Book Is Not About

This book is not about how to estimate the very largest projects—more than 1 million lines of code, or more than 100 staff years. Very large projects should be estimated by professional estimators who have read the dozens of obscure journal articles, who have studied the 800–1000-page books, who are familiar with commercial estimation software, and who are as skilled in both the art and science of estimation.

Where to Start

Where you start will depend on what you want to get out of the book.

If you bought this book because you need to create an estimate right now... Begin with Chapter 1 (“What Is an “Estimate”?”), and then move to Chapter 7 (“Count, Compute, Judge”) and Chapter 8 (“Calibration and Historical Data”). After that, skim the tips in Chapters 10–20 to find the techniques that will be the most immediately useful to you. By the way, this book's tips are highlighted in the text and numbered, and all of the tips—118 total—are also collected in Appendix C, “Software Estimation Tips.”

If you want to improve your personal estimation skills, if you want to improve your organization's estimation track record, or if you're looking for a better understanding of software estimation in general... You can read the whole book. If you like to understand general principles before you dive into the details, read the book in order. If you like to see the details first and then draw general conclusions from the details, you can start with Chapter 1, read Chapters 7 through 23, and then go back and read the earlier chapters that you skipped.

Bellevue, Washington
New Year's Day, 2006

Microsoft Press Support

Every effort has been made to ensure the accuracy of this book. Microsoft Press provides corrections for books through the World Wide Web at the following address:

<http://www.microsoft.com/learning/support/books/>

To connect directly to the Microsoft Press Knowledge Base and enter a query regarding a question or issue that you may have, go to:

<http://www.microsoft.com/mspress/support/search.asp>

If you have comments, questions, or ideas regarding this book, please send them to Microsoft Press using either of the following methods:

Postal Mail:

*Microsoft Press
Attn: Software Estimation Editor
One Microsoft Way
Redmond, WA 98052-6399*

E-Mail:

mspinput@microsoft.com

