

PREDICTING THE UNPREDICTABLE

PRAGMATIC APPROACHES TO ESTIMATING PROJECT SCHEDULE OR COST



AUTHOR OF "MANAGE IT!"
YOUR GUIDE TO MODERN, PRAGMATIC PROJECT MANAGEMENT"

JOHANNA ROTHMAN

Predicting the Unpredictable

Pragmatic Approaches to
Estimating Project Schedule or Cost

Johanna Rothman

This version was published on 2015-05-26

ISBN 978-1-943487-03-5



Practical ink

No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without written permission from the author.

Every precaution was taken in the preparation of this book. However, the author and publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information contained in this book.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Practical Ink was aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals.

©2015 Johanna Rothman

*To everyone who was ever asked,
“How much will this project cost?” or
“How long will this project take?”*

Contents

Acknowledgements	i
1. Introduction	1
1.1 Estimates Are Guesses or Predictions	1
1.2 Estimates Change	2
1.3 Estimates Expire	3
2. What Estimates Are	5
2.1 Provide an Accurate but Not Precise Estimate .	6
3. Why Do We Estimate Anyway?	8
3.1 Why Do You Estimate?	9
3.2 Ask This Question Before You Estimate	11
4. Software is Learning, Not Construction	12
4.1 Inch-Pebbles or Small Stories Show Progress .	14
4.2 Learn With Spikes	14
5. Think About Estimation	15
5.1 Estimating the Unknown: Dates or Budgets . .	16
5.2 Determine Your Degrees of Freedom	17
5.3 Insist on a Ranked Backlog	19
5.4 The Team Doing the Work Provides the Estimate	20
6. How to Estimate	22

CONTENTS

6.1	Your First Best Bet: Make Your Stories and Chunks Small	22
6.2	Your Second Best Bet: SWAG and Refine . . .	23
6.3	Collect Data	24
6.4	When You Have a Decreed Date	26
6.5	Wrap Up	29
6.6	Estimating a Program	32
6.7	Beware of These Program Estimation Traps . .	35
7.	Rolling Wave Planning	38
7.1	Learn as the Project Proceeds	38
8.	There Is No Correct Estimation Model	40
8.1	We Invent; We Don't Repeat	42
9.	But I Need to Know When the Project Will Be Done	44
9.1	What You Can Say to Managers Who Think They Need to Know	45
10.	NoEstimate or Do Your Estimates Provide Value?	47
11.	Use All Four Parts of Project Estimation	49
11.1	Part 1: Create an Initial Estimate	49
11.2	Part 2: Track Estimation Quality Factor to Understand the Project Estimate	51
11.3	Part 3: Use EQF to Manage Project Concerns .	52
11.4	Part 4: Update Your Estimate as You Know More	52
12.	Show Your Status and Update Your Estimate . . .	54
12.1	Probabilistic Scheduling	54
13.	Refocusing: 90% Done Is Not Almost Done	57
14.	Future Fixes	60

CONTENTS

14.1 Plan Ahead	60
14.2 Hindsight is 20-20	61
14.3 Track Your Estimates	62
14.4 Measure Bad Fixes	63
14.5 Manage Your Buffers	65
14.6 Incorporate Agility	66
 Troubleshooting Your Estimation Problems	68
15.Avoid Multitasking	69
16.Avoid Student Syndrome	71
17.Estimation Units Predict Schedule Slippage	73
18.Edit Those Epics	75
19.What You Can Do For Estimation	79
20.Estimation Depends On...	82
21.Estimating Testing Time	85
22.Need to Learn More About the Work You're Doing? Spike It!	90
22.1 How Does a Spike Work?	90
22.2 Spikes Are About Learning	92
22.3 What Happens to the Code at the End of the Spike?	92
22.4 "Use the Code As Is..."	93
22.5 How Many People Were Involved in the Learn- ing?	94
23.Use Targets as Estimates	95
23.1 How to Use Targets	95

CONTENTS

- 23.2 When the Target Is a Trap 96
- 24. How to Avoid Three Big Estimation Traps 97
- 25. Understanding Multitasking and the Cost of De-
lay on Estimation 100
- 26. What You Know About Estimation Now 106
 - 26.1 Transition to an Agile Approach or an Incre-
mental Approach for Your Projects 106
 - 26.2 Make Your Features Small 107
 - 26.3 Iterate on your Estimate 107
 - 26.4 Don't Multitask 108
 - 26.5 Don't Let Defects Dictate Your Estimate 108
 - 26.6 Final Thoughts 108
- Glossary 109
- References 112
- More from Johanna 113

Acknowledgements

I thank my [Managing Product Development](#) blog readers. You asked me enough questions that I had to write the answers.

I thank my editors, Rebecca Airmet and Nancy Groth. I thank Jean Jesensky for her indexing for the print book. I thank Karen Billipp for her layout for the print book.

Any errors are mine.

7. Rolling Wave Planning

Project teams can use rolling wave planning to deliver interim milestones and then replan the next chunk of the project.

Here's how rolling wave planning works:

Loop:

- Plan what you know for the next few weeks (I use a 3-4 week rolling wave). If you're managing a traditionally planned project, make this as detailed a Work Break-down Schedule (WBS) as you like. If you're managing an agile project, you may not have to do any more planning than what you already have done.
- As each week goes by, use the knowledge you've gained about the project to replan the already-planned weeks and plan the next week at the end of the current schedule.

Endloop

As the project proceeds, you'll replan frequently, but you won't replan a lot of the work.

7.1 Learn as the Project Proceeds

The idea behind rolling wave planning is that you can't know everything about the project in advance, so don't bother

trying to plan a lot in detail. Plan the next few weeks in detail, always staying about three to four weeks ahead of the project. In my experience, it's not worth trying to look more than four weeks ahead. Things will change too much.

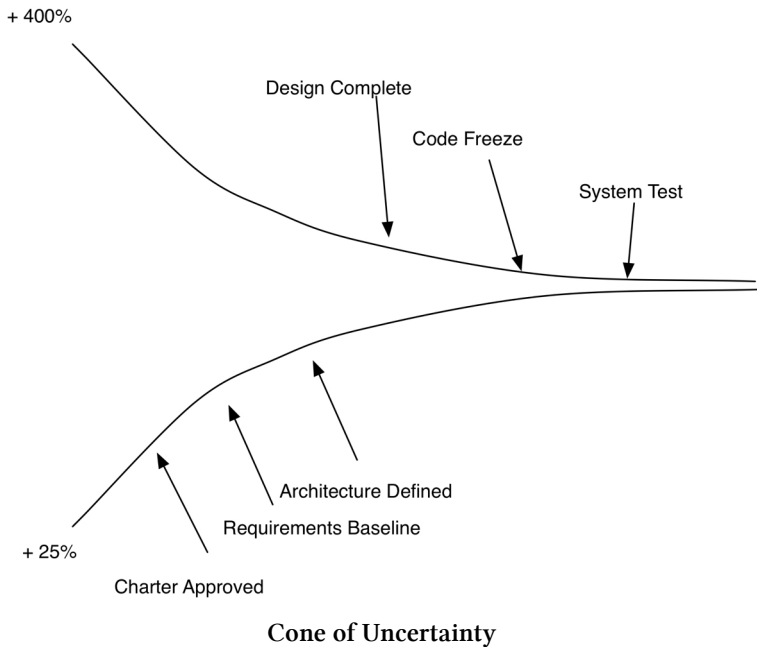
Of course, if you know you have hard dates like end-of-quarter or a trade show, put those events in the schedule. But rolling wave planning is much more likely to help you achieve those hard dates.

I incorporate adaptive planning into my rolling waves, by using the knowledge I've gained about the project to (re)organize the work as necessary.

If you haven't tried rolling wave planning, give it a shot. I find it especially helpful when I want to timebox to meet a specific date and I want an early warning if the date is impossible.

8. There Is No Correct Estimation Model

For years, we bought the cone of uncertainty for estimation—that is, our estimates were just as likely to be over as under.



Laurent Bossavit, in [The Leprechauns of Software Engineering \(BOS\)](#), shows us how that assumption is wrong. (It was an assumption that some people, including me, assumed was real.)

This is a Gaussian (normal) distribution. It's what we expect.

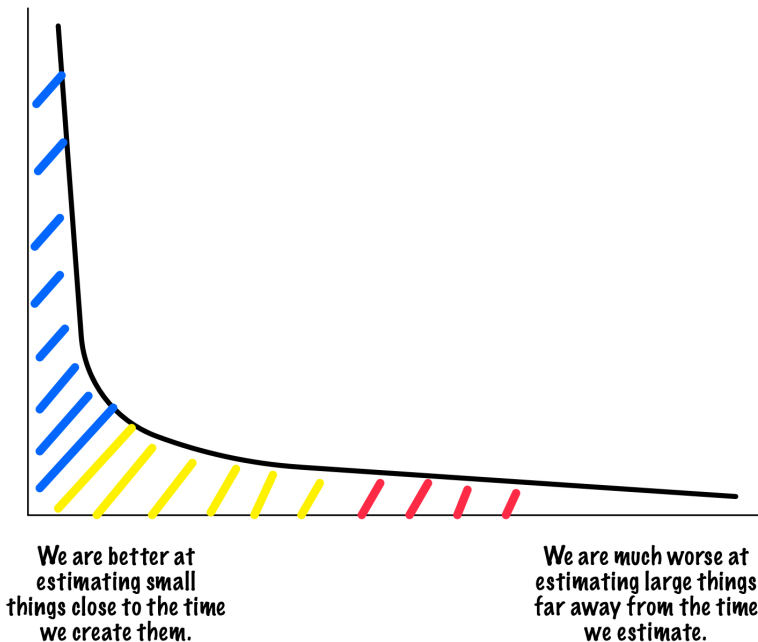
But, it's almost never right. As Laurent says,

“Many projects stay in 90% done for a long time.”

What curve do our estimates follow if they don't follow a Gaussian distribution?

Troy Magennis, in [The Economic Impact of Software Development Process Choice - Cycle Time Analysis and Monte Carlo Simulation Results \(MAG\)](#), suggests we should look at the Power-Law (Weibull) distribution.

Power Law Distribution: Example for Estimation



Power Law Distribution

What this distribution says with respect to estimation is this: We are good at estimating small things. We get much worse with our estimation quickly, and for the long tail (larger and larger chunks of work), we are quite bad.

Why? Because creating software is innovation. Building software is about learning. We better our learning as we proceed, assuming we finish features. Maybe take another look at [Software is Learning, Not Construction](#).

8.1 We Invent; We Don't Repeat

We rarely, if ever, do the same thing again. We can't apply precise estimation approaches to something we have never done before.

The question is this: What effect does understanding an estimation model have on our estimates?

If we know that the Gaussian (normal) distribution is wrong, then we won't apply it. Right, why would you do something you know to be wrong? You would not estimate large chunks and expect to have a +/- 10% estimate. It doesn't make sense to do that.

But what can we do? In Troy's paper, he says that if you have large, unique work items or you have large WIP, you will have poor predictability.

My suggestions for your estimation:

- Estimate small chunks of work that a team can complete in a day or so.
- Keep WIP low.

- Replan as you finish work.
- Watch your cycle time.
- No multitasking.

What should you do when people ask you for estimates? What kind of requirements do you have? If you have large requirements, follow my advice and use the percentage confidence, as in [Estimating a Program](#).

You can predict a little for estimates. You can refine your prediction. And, you may have to predict a large effort. In that case, it helps to know what distribution model might reflect your estimate.