Extracted from:

Create Your Successful Agile Project

Collaborate, Measure, Estimate, Deliver

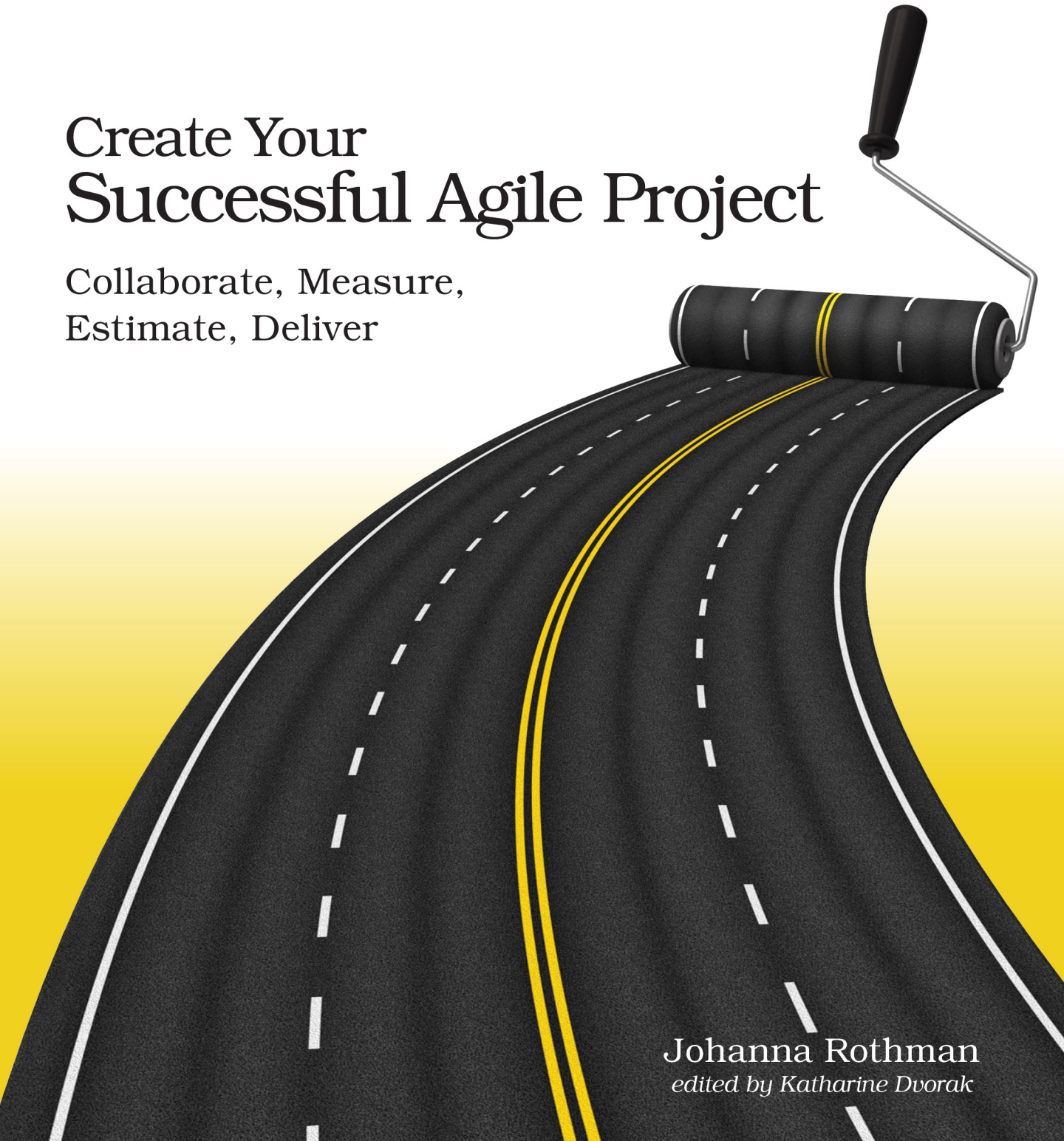
The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Create Your Successful Agile Project

Collaborate, Measure, Estimate, Deliver



Johanna Rothman
edited by Katharine Dvorak

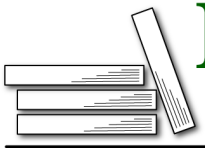
Create Your Successful Agile Project

Collaborate, Measure, Estimate, Deliver

Johanna Rothman

The Pragmatic Bookshelf

Raleigh, North Carolina



Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at *https://pragprog.com*.

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Development Editor: Katharine Dvorak
Indexing: Potomac Indexing, LLC
Copy Editor: Candace Cunningham
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

Printed in the United States of America.
ISBN-13: 978-1-68050-260-2
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—October 2017

To Mark, as always.

Learn to Estimate with Relative Sizing

If you have a new team or a team new to agile approaches, no one has any data about what they can do as a team. You may have been in this position before as a leader in your organization. Your managers may have asked you to estimate on behalf of the team.

Don't estimate for the team or even think about committing it to any deliverables in an agile environment. The team is in charge of its own work. The team manages its work. The team manages its estimation.

However, there is something you can do. You can help the team estimate its work with relative sizing, even if the team members have never worked together and have no historical data.

Relative sizing uses two ideas to improve estimates: the team compares story sizes (points) against what it's accomplished in the past (the relative part), and it estimates using the wisdom of the team as a form of Wideband Delphi

estimation. As everyone comments on the story and as people suggest a relative size, the team gathers better data and can therefore develop a better estimate.

A Short Explanation of Wideband Delphi Estimation

Wideband Delphi is a team-based estimation approach. Before agile approaches, some teams used Wideband Delphi to create an entire project estimate. The person(s) who wrote the requirements explained all the requirements to the team members. The team disbanded and the members created *their* estimates for their part of the work.

The team members then met to discuss their estimates and where their understanding of either the requirement or the complexity occurred. Teams might have met up to four times to review and re-estimate. (I provided a more detailed explanation and how to use Wideband Delphi in *Manage It! [Rot07]*.)

To effectively use relative sizing, first ask the product owner to create stories that are as small as the product owner can create. Make sure these stories are real user stories so the team can understand who will see the value in each story.

Ask the team to group the stories by size, from the smallest to the largest. Keep similar-size stories together. The entire team decides how large the stories are.

Assess the story sizes. Using the Fibonacci sequence (1, 2, 3, 5, 8, 13, and so on), assign all the smallest stories to the size 1 bucket. The next-sized stories are a 2 and the next are a 3. Continue until the team agrees on the relative sizes of the stories. (See *Agile Estimating and Planning [Coh05]*.)

Once the team agrees on the relative size, take the stories estimated as 2. Do all the 2 stories look like they're about the same size? If so, now estimate the duration for the 2 stories.

If the team thinks all the 2 stories will take about 10 person-hours, you now know how long the 1 stories will take. Divide the duration for the 2 stories by 2 to derive the duration for the 1 stories. In this example, our 1 stories would take five hours. Ask yourself whether that makes sense. If so, you now have the factor to use to multiply against all the other relative sizings.

If you see you have stories larger than an 8, size up to 13 and then use 20, or 40 for very large efforts. (The reality is that no one understands the size of anything past 13, but we can use these numbers in a different way later.)

If you have stories larger than say, 8, the team has plenty of unknowns, or thinks those stories are highly complex. Consider a spike first to break the task into smaller pieces. (See *Spikes Can Help Everyone Understand the Value*, on page ?, for more information.)

Here are some guidelines that have helped teams in the past:

- If a "1" is larger than a team-day, see if the team either has too-large stories or the team isn't a feature team.
- If the team regularly sizes stories as larger than 5, there could be several problems: the stories are too large; the team doesn't understand the stories; there is no defined MVP; or the code is in terrible shape. Ask the team to discuss and address the root cause of larger stories.
- Ask the entire team to workshop the stories with the product owner and to see what it takes to get to a size of 1.

When teams create stories of size 1 (where 1 is a team-day or less time), the team knows several things:

- The team can count the stories for the next estimation period and have confidence about what it can deliver.
- The team can deliver at least one story every day.
- The team has more confidence in its estimate, which reduces overall project risk.

Your Small Stories Might Be Larger than Mine

As you've seen, I like very small stories. However, you might find that "small" for you is a little larger than for me.

One product owner I know says, "As long as the team can collaborate on the story and finish it inside of two days, that's small enough for me." Me too. One agile coach I know says, "We don't want stories smaller than a 3 because it's not worth breaking them down and then people don't work together on a story."

I'm still going to talk about one-day stories—for a team, not a person. As long as your *team* releases a story every day, or every other day, that might be small enough. You might find it worthwhile to ask the team what it thinks is small enough. I'm sticking with my one-day stories.

The larger the number for the story, the greater the uncertainty the team has for the estimate. See *Predicting the Unpredictable [Rot15]* for more details about estimation.

Use Relative Estimation for Iteration-Based Estimates

If you use an iteration-based agile approach, your team will want to estimate what it can deliver for the next iteration. It can commit to what it can fit into an iteration.

To use relative estimation for iteration-based estimates, the product owner first creates the ranked backlog as described in *Plan the Backlog*, on page ?. The team then estimates each story as a team. Some teams use planning poker cards. Those cards have the Fibonacci series of 1, 2, 3, 5, 8, 13, and whatever larger numbers the team needs to estimate its work.

Joe asks:

How Do I Use Planning Poker?

When teams create or use the Fibonacci sequence (or any other relative sizing technique), they can use planning poker.

Every person has a deck of cards with the sizes on the cards. If you use Fibonacci, every person would have eight cards, one each with 1, 2, 3, 5, 8, 13, 20, and 40. When the team estimates, someone, often the product owner, holds up a story and asks, "What's your estimate for this story?" Each person takes his or her card showing the relative estimate of each story.

Planning poker is a Wideband Delphi estimation technique. It surfaces concerns and issues about a story so the team can resolve those issues or concerns, or know that the story might be troublesome. Planning poker is especially useful when team members disagree on the story's relative size. Teams decide what to do: go with a larger estimate or a smaller one—or break the story down into smaller chunks of value.

If you have stories of size 1, planning poker is easy. You ask, "Does anyone think this is larger than a 1?" If so, the team has other choices: spike the story to timebox the work to one day to understand what else to do, or break up this story, which is really a feature set, into other stories.

Here's the problem with relative estimation and deciding what a team can pull into an iteration: the larger the stories are (larger than a 1), the more uncertainty the team has about the work it can commit to for an iteration. Instead of more discussion around estimation, consider a workshop to create smaller stories or an additional backlog refinement meeting. See *Create or Refine the Stories as Preparation for Future Work*, on page ?.

Large relative story sizes provide qualitative data: it's possible the story is complex; the code might be cruft or this story might be an entire feature set. The team might need to explore if it can create a more accurate estimate. See what you can do to help the product owner create stories of size 1 before the team has to estimate the story.