Extracted from:

Manage Your Project Portfolio, Second Edition

Increase Your Capacity and Finish More Projects

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Manage Your Project Portfolio

Second Edition

Increase Your Capacity
Finish More Projects

Johanna Rothman

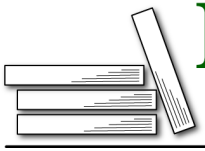
Edited by Rebecca Gulick

Manage Your Project Portfolio, Second Edition

Increase Your Capacity and Finish More Projects

Johanna Rothman

The Pragmatic Bookshelf
Raleigh, North Carolina



Pragmatic Bookshelf

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:

Rebecca Gulick (editor)
Potomac Indexing, LLC (index)
Liz Welch (copyedit)
Gilson Graphics (layout)
Janet Furlow (producer)

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: 978-1-68050-175-9
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—July 2016

CHAPTER 4

Evaluate Your Projects

Once you have collected all your work into a draft portfolio, you'll evaluate each project or program. For the purposes of managing the project portfolio, treat your programs and projects in the same way. You don't have to do anything special to evaluate programs.

It's difficult to decide which work is most valuable and which work should be done first, so separate those decisions. The first time you organize your work into a portfolio, you don't have to make the ranking decision. Your very first decision is about whether you *want* to commit to this project, kill the project, or transform the project in some way before continuing.

Resist the temptation to say "I want to do this project first" or third or seventh as you proceed with the initial evaluation. When you separate the evaluation from the ranking, it's easier to make all the decisions. You'll have an opportunity to rank after you evaluate each project or program.

Should We Do This Project at All?

Before you try to decide where each project fits in the portfolio, ask, "Should we do this project at all?" If the answer is no, take the project off the list. If the answer is yes, select a way or ways to rank-order the projects in the portfolio. Take the time to ask this simple question of each and every project in your portfolio.

You may not feel as if you have the right to ask this question. You do.

If you're a first-level manager in terms of influence, you have intimate knowledge of the project and the product it will create or extend. You know about the strategic importance of this project with respect to the product. If you're a middle manager, you can see all the initiatives and can consider the evaluation of this project with respect to the others. If you're a senior manager,

you can see the entire organization's strategic direction and see whether this project should be done at all with respect to all the initiatives across the organization.

Any time you have a chance to eliminate a project from consideration, do so. As you review the portfolio over time, note which projects you don't ever give many points. Can you take those projects off the list altogether? If not, can you create a small project or a short iteration to provide you with some information about whether this project is worth the aggravation of considering?

Sometimes, you'll put projects into the portfolio and not get to them for a while. Sometimes a very long while. If that's the case with some of your projects, check to see whether you still need to consider these projects. It could be that the answer is no. If you're not sure, move the projects to the parking lot, as described in *Keep a Parking Lot of Projects*, on page ?. Whatever you do, remove projects that you don't need to consider now. At some point, you can address the projects in the parking lot. But keep projects you don't have to consider out of your immediate decision making. Don't waste your energy on decisions you don't have to make right now.

Decide to Commit, Kill, or Transform the Project

Once you've decided you should do this project, you have a limited number of decisions to make. You can commit to a project, kill a project, or transform a project to increase its chances of success.

Making a commit/kill/transform decision requires data about project progress, project value, and obstacles. If your projects are all using a serial life cycle, the data doesn't exist. In a serial life cycle, you have no data about whether this project is valuable until very near the end of the project—after you've spent virtually all the money and assigned people to this project, excluding other potential projects.

The problem with serial life cycles is that they do not adapt to change. Phase-gate and waterfall approaches are examples of serial life cycles. In a serial life cycle, you are supposed to plan the entire project, and then finish first the requirements, then the architecture, then the design, then the code, and finally the testing. Serial life cycles do not expect change. Serial approaches do not use deliverable-based planning.

Schedule games can occur in other life cycles, but serial life cycles hide the games longer. For example, if your project teams have been implementing through the architecture instead of by feature, they run a high chance of encountering the 90 Percent Done schedule game near the end. That's where

you finish 90 percent of the project and realize you have 90 percent remaining. The team discovers they have all that work to do at the "end" of the project because they haven't integrated features as they proceeded.

You can use a serial approach to your projects and succeed. The shorter the project, the more useful a serial approach is. And, for the purposes of managing the project portfolio, where you want flexibility on which teams to assign where, decide how long your project should be.

The shorter your serial project, the more often it delivers value. You might have releases: release 1, release 1.1, release 1.2, and so on as one way to shorten your serial approach. Then you can assess the value of the remaining projects for that product against all the other projects in your backlog.

If you're a manager responsible for a number of serial life-cycle projects, you may have succumbed to the Split Focus or Pants on Fire schedule games as a way to manage the risk of any one project being a true failure. If you feel you have no other choices in life cycles, please read *What Lifecycle? [Rot08]* or the appendixes in *Manage It! [Rot07]* and reconsider. And read *Return on Software: Maximizing the Return on Your Software Investment [Toc05]* to try to do the project evaluation math that a serial life cycle requires.

A project team that chooses any life cycle other than a serial life cycle can provide you with data about the project much earlier than a serial life cycle. And, because they can provide data, they also receive feedback about the work and risks in the project and can manage those risks by reorganizing, replanning, or even redoing the work. You can manage the project portfolio with a life cycle other than serial.

Let's examine each of the decisions: commit to, kill, or transform.

Commit to a Project

When you commit to a project, it's a real commitment, not a partial commitment. Here's what a real commitment means: that until you make another conscious portfolio decision, you commit to funding this project. You commit to assigning the necessary people to the project—and only to this project. If the project needs something else (space, capital equipment, desks, whatever), you commit to delivering that to the project.

When managers don't fully commit, they revisit their projects again and again. This creates both management debt and project technical debt. They also create capacity debt because people (managers and technical) can't improve their capabilities when they're overburdened with too much work.

Recommitment Is Easy Now

by: Sam, Scrum Master

We just finished our management review with senior management. Now that we're using agile, we take only about five to ten minutes to explain our status in the meeting. We just show them our velocity and a demo. I let them know about project-level obstacles. Management calls them *risks*—which is fine with me.

Our management meets with us only quarterly, because our market isn't changing that fast, so quarterly is fast enough. But it takes me only about ten minutes to prepare. The hardest part is noting what has changed in the product since the last time we demo'd. Once we demo and show our velocity data, management recommits to this project, assuming there is more valuable work on our product backlog.

Before we moved to an agile life cycle, it took us a full week to prepare for the meeting, and then our management took hours and sometimes days to decide whether we should finish a project.

Understand the Requirements of Commitment

A commitment to an ongoing project is not a blind commitment. If the project requires two DBAs and you have only one available because the other is on another project, think about your options. You might

- Start the project and realize the team will be slower than you wish. That's because you committed, but the commitment is not complete. The team is missing necessary people.
- Wait to start the project until you can create a fully staffed team.

If you have agile teams that have been fully staffed up until now, let the team tell you what they need. The team might work with the product owner to discuss the need for this much database work into this iteration's backlog. Sometimes the team will estimate more time for the features so that other people can learn more about what the DBA does. With those discussions, the product owner might make other decisions.

What I most often hear is not a discussion about which features to commit to when but a blind request or demand from people who say, "We need these features now." If you are not working in an agile way now, it's easy to encounter those demands. If you or your team receives a demand for this feature now and you don't have another DBA, you have to decide which project is more important than the other. If you can't fully commit to a specific project, don't start it in this time period. Wait until the next time you evaluate the portfolio.

You might tell the project team you are happy with their progress and the potential value on the product backlog—you recommit, and they continue. If

you're ready to recommit to an ongoing project, do so. If you're ready to commit to a new project, do that. If you're not ready to commit, you might need to kill or transform the project.

Commit Fully

Commitment is not a "We'll give you part of what you need, but…." It's a full commitment.

This is a hard line. Here's the problem. If you can't fully commit the necessary people and money to a project, you are guaranteeing the project will not provide the value you want it to provide. Why would you waste your people, time, and money on a project you can't fully commit to? If you have open requisitions and you haven't filled them yet, know that part of the value the project team will provide to the organization is the interviewing and hiring work. You won't get the benefit of those people on the project, but the project will be providing value to the organization.

But if you don't have open reqs and if you know you need more people, you are fooling yourself if you think you can somehow commit to a partially funded or staffed project. You would be better off, from a throughput perspective, either having generalists or having at least a pair of specialists so you can fully staff a project when you need to do so.

If you are trying to staff a project with people who are working part-time on your project and part-time on other projects, you have an uncommitted project. That's because the cost of context switching will erase any potential ability to focus on this project. Don't partially commit to a project; that's a lack of commitment. Be honest. Take that project off the committed list. You may have to move the project to the parking lot. You might have to transform it. But never make a partial commitment.

Two Part-Time People Do Not Make One Full-Time Equivalent

If you're trying to staff a project with people who are working part-time on your project and part-time on other projects, you have an uncommitted project. Don't do that.

You may have heard of the term *full-time equivalent* (FTE). Originally it was used by accounting departments to explain that several part-time staff added up to one full-time person. Gradually, it moved from part-time staff to multitasked staff.

The problem is that if you have two people, each half on your project and half on another, you don't have a half-time person at all. You might have someone who's closer to 40 percent. If you're unlucky and each of those people are context switching like crazy, you might have the equivalent of 10 percent of a person. But you definitely do not have one FTE.

Some number of multitasked people are not an FTE. Some number of part-time people who are assigned to just one project could be close to some number of FTEs, because they don't have the cost of context switching. But counting multitasked people is wrong arithmetic.