Extracted from:

Manage Your Project Portfolio

Increase Your Capacity and Finish More Projects

The Pragmatic Programmers

Manage Your Project Portfolio

Increase Your Capacity and Finish More Projects

Johanna Rothman

Edited by Daniel H Steinberg

ISBN-10: 1-934356-29-8

ISBN-13: 978-1-934356-29-6

Printed on acid-free paper.

B5.0 printing, June 1, 2009

Version: 2009-7-1

Preface

So many things can bring a project to its knees. Maybe there's too much multitasking... or so much technical debt that the project team can't make progress on the current release... or so many emergency projects that emergencies have become normal... or so many high-priority projects that no one knows which to work on first.

Sure, there are other things that could be a problem: technical staff estimations are way off, your major competitor just released a huge update and your project won't be ready for another six months, the technical challenges of testing (or writing or developing) are more than your testers (or writers or developers) can manage, the project needs more machines or memory or disk drives... and the list goes on. But if you dig down far enough, you will often find multitasking—and its associated issues of emergency projects, technical debt, or people spread across way too many projects—is what has led to most of your problems.

Multitasking occurs when managers don't make decisions about which projects to do first, second, third, last, and, even more important, never. Some managers don't realize it's their job to make those decisions—they think it's their job to try to staff every project. Some managers don't know how to make those decisions. Some management teams can't agree on the decisions. Whatever the reason, when managers don't decide in which order to execute projects and which projects to leave alone, the project teams suffer from multitasking.

Why am I so passionate that you should manage your project portfolio and you shouldn't multitask? I fell into managing the project portfolio when I was working at a company that made complex hardware/software systems. I had first been hired as the director of SQA and continuing engineering. Then they decided they needed a program manager to manage the largest program the engineering organization had attempted. I stopped being the director and ran that program. About four months before our planned release, we had a customer cancel a contract. Management decided to lay off about half the engineering staff. They assigned someone else to manage the program and asked me to be the director of software engineering.

We were down to about thirty to forty people in development. We had to finish the development work on the program and continue to respond to problems in the field. Each field problem was a crisis and required several weeks of work. So here I am a director, with an interim VP, people who'd been working crazy hours for months, and a release we had to get out. And huge problems we had to fix. We had to do it all. We had no choice.

I made a spreadsheet of what everyone was working on so I could understand where the time was going. I'd made spreadsheets like that before when I'd managed testers and developers who were matrixed into projects, but I had never had to staff quite so many simultaneous projects.

After two weeks of people working the way they'd been assigned, I realized no one was making progress on anything. I sat down with the managers who reported to me. We discussed what work we would staff and not staff. We assigned people to no more than two projects in any given week. We made sure people had team members they could work with to finish the work.

I took the heat from senior management—and there was plenty of it. "You have to do this project and that one and that other one and that other one over there. This week."

I said, "Sorry, we can't do that much in one week. You have to choose."

And of course they rebutted with, "No, you have to do it all."

I said, "Well, then I'll choose."

"You'd better be right."

After another two weeks, we started to make progress, but it still wasn't fast enough. I had a team meeting with my managers and asked, "What will it take to finish this project? Just this one here?" One of the managers said, "If Tom and Harry and Jane can concentrate on just that project for a week, we can finish it."

I said, "OK, have them do that. Now, what about that crisis over there?"

"Well, we need Harry...."

"Sorry, you can't have him. Who else can you use, and how long will it take?"

We had that conversation for all the outstanding work. Now we had small teams assigned to a bunch of the problems so we could fix them and get some breathing room. In about a week, we would be half-staffed on the program, and in about two weeks, we would be back to full staffing on the program. The developers were thrilled to finish something. The managers were happy about not having to move people around. I was happy that we finally got some things done. My senior managers were unhappy with my progress.

After two months of this, we finally had just new development to do on the program, because the continuing engineering department was able to keep up with the field problems. That's when we started to make huge progress on the release, because we were working by feature and assessing our progress biweekly.

We used a combination of approaches: continuing engineering used a kanban approach because the problems were smaller than the features for a release. They could limit their work in progress and work on one problem at a time until it was fixed. Development (and the test group) used two-week timeboxes, working in features, so we could finish chunks of work.

By the end of the four months, we had a release, although we didn't have all the features our senior management wanted. We had the field problems under control. We hadn't added a ton of technical debt.

But the people who remained learned that they could work on one project at a time, one task at a time, until it was done. They could make more progress doing one thing at a time than splitting their time among several pieces of work, even if the work was related.

If I could manage the project portfolio with an organization reeling from a layoff, where we had an unstated strategic plan, where the senior managers had trouble deciding what to do on any given day, you can do this for your work. You may need different approaches for different groups. One group might need to limit the work in progress, especially if you're in a serial life cycle and people with different specialties cycle in and out of the project. One group might need to work in one-week or two-week timeboxes, while another might find three-week timeboxes easier to manage.

Here's the secret of project portfolio management: you can do it all. Just not all at the same time.

Whether you're a manager or not, you need to have a view of all the work underway and all the work you want to do. That's the only way you can make good decisions about which projects to do when and with which people. Even if those people are only you.

Effective managers and leaders create and use a pipeline view of all of the projects, those in progress and those desired. They can see all the potential work, as well as the people and other resources available, and then match the work to the company's (or group's) mission and strategic direction. This collection of projects is an essential tool called the *project portfolio*.

Managers who lead make the difficult portfolio decisions. They look at their company's mission, they look at their mission, and they decide. They decide on the mix of projects the technical staff can start at one time, how long they are willing to let those projects run, and when they need results. They decide on the strategically and tactically important work. Then they do it.

Managers who manage the project portfolio decide when they need to review project status. They have criteria by which to decide whether the projects should continue. And if a project is not providing value to the organization and should not continue, these managers kill those projects. These managers learn their technical staff's capacity so they can plan. And, they make all of these difficult decisions that will allow the organization to be successful.

Managers are leaders when they make the portfolio decisions. Managers are leaders when they guide a team to success. Managers are leaders when they request the team commit to finishing a doable amount of work in a reasonable amount of time.

You don't need to be a senior manager to manage the portfolio. Sure, it helps if the organization has a strategy that translates into a mission, which guides the top-level portfolio development down to the bottom-most level. But let's face it, most organizations don't have that.

You need to understand your mission, understand all the missions between you and the top of the organization, and know how to collaborate across the organization. Then you can develop and manage a project portfolio successfully. Once you've defined the project ranking, you communicate that project ranking to the entire organization, staff the highest-ranked projects, and let your staff know which projects they can ignore for now.

You may never have heard of project portfolio management. Or, maybe you've heard of a bunch of mathematical formulas that even if you can understand, you'll never get your peers or managers to understand and see. We'll use some measurements, but no math. It's easy to understand, and it will help you make decisions. But the hard part of portfolio management is not the math. You may find some of the decisions difficult, but the hard part is sticking with those decisions until it's time to reevaluate the portfolio.

Great managers build trusting relationships with their teams (*Behind Closed Doors* [RD05]). In addition, great managers lead their organizations by selecting the work to do and *not* to do, and therefore they deliver results to the organization and build capacity in their teams. This book is about that kind of leadership.

Before we get started, I thank all the people who took the time to review and help prepare this book. They are Clarke Ching, Linda Cook, Esther Derby, Mark Druy, Mike Dwyer, George Dinwiddie, Don Gray, Ron Jeffries, Andy Hunt, Hannu Kokko, Tim Lister, Robert McBride, Steve Peter, Dwayne Phillips, Dave W. Smith, Daniel Steinberg, Dave Thomas, Gerald M. Weinberg, and Kim Wimpsett.

Any remaining mistakes are mine.

If you're ready to lead your team, group, or organization, this book is for you. Let's start.

Johanna Rothman
July 2009