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

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.

4.1 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 project to the parking lot, Section 8.1, *Keep a Parking Lot of Projects*, on page 123. 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.

4.2 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.

Schedule games occur often in serial life cycles. 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% Done schedule game near the end. 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?" [Rot08b] 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.

4.3 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.

If you have agile teams that have been fully staffed up until now, let the team tell you what they need. Sometimes, the team needs to have a conversation with the product owner about organizing 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 reqs 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.

Not Filling a Req Costs You Real Money

You have twenty projects you want to staff. You have people to staff ten of them. You also have enough open reqs to staff another five. And, you know that to interview will take time away from the people on the projects. What's a leader to do?

First, try to avoid this problem of having to hire many people. Hiring costs you time and money and slows down people on the projects. The project delays are not just from interviewing; they also occur when you bring someone on and have to train that person (see *Hiring the Best Knowledge Workers, Techies & Nerds* [Rot04a]).

If you are faced with the problem of open reqs and too many projects and a decision to hire, try this approach. First, look at your top-ranked project. Can you fully staff all of those projects? If you can, do. When you stop being able to fully staff and not disturb already-jelled teams, start hiring for the higher-ranked projects.

As you hire for those highest-ranked projects, keep the other projects in mind. If you find someone who will fit in another project, great. But

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

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. 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.

watch out for hiring just for the lower-ranked projects. You might be hiring people who don't have the skills to finish more valuable projects.

Remember, the projects you've ranked higher return more value to the organization. If you can't fully commit to those projects, the organization loses that value. For many of you, that's real money.

4.4 Kill a Project

Your second possible decision is the decision to kill a project. The key to killing a project is to make sure all activity associated with the project stops. Sometimes, that's harder than it should be. (See Section 4.5, *How to Kill a Project and Keep It Dead*, on page 60 for more information on good ways to kill a project and keep it dead.)

If the need for this project has changed, it's time to kill the project—and just the project. Move the people to another project. You may find that if a project is too ambitious, you'll have to kill it. Or, it may be that the market has vanished or your organization's strategy has changed.

Don't Recommit Because of Sunk Cost

Sunk cost is not a good reason to commit to a project again. In fact, you might need to kill the project and not throw money away on it anymore. When you hear "But we've already spent so much on this project," that's a cue to reconsider any more commitment.

If sunk cost keeps projects alive, reconsider your ranking mechanism (see Chapter 5, *Rank the Portfolio*, on page 68). You may need a different ranking approach.

And, consider closing the project quickly if you think you must finish something to take advantage of the value created so far. An iterative-incremental life cycle for the project makes this possible.

The most serious form of killing a project is to stop all work on the project and throw away all of the intellectual property associated with that project—not the people, just the code or tests or drawings or whatever you have as intellectual property. Don't throw away the people. Assign them to other projects.

Postponing a project is another form of killing the project. If all the team was supposed to do was learn about an architecture or proof of concept or your customer doesn't want that project now and won't fund it, you can postpone the rest of the project, realizing there will be a startup cost later if you choose to restart it. You can choose to put this project on the parking lot (Section 8.1, *Keep a Parking Lot of Projects*, on page 123) if you don't want to lose track of it.

The kill decision is difficult to make when you're using a nonagile life cycle. (Any decision is difficult with a serial life cycle. You have more data with an iterative or incremental life cycle. You have the most data with an agile life cycle.) If the project teams don't have to get to releasable product at the end of every iteration, the project team will have some putting-away work before it's safe to kill the project. This is where many project teams and managers get confused. How much time do they need to clean up? No matter how much time you give them, it's not going to be enough, which is why I recommend you give the project team no more than two working days. At the end of two working days, ask the team to conduct a retrospective, and assign the project team to another project.

My Project Was Canceled, Parked, and Restarted
by Vu, Project Manager

We started a new project, trying for a particular kind of communication product. We had to achieve a certain level of performance and reliability. We tried a number of ideas for several timeboxes, but we encountered the "laws of physics" and just couldn't do what we needed to without different hardware. So, our project was canceled.

In another company, that might have been the end of the story. But our managers knew we had to keep this project in mind, so it went onto the "parked projects" list. I'd thought that's where projects went to die, but every quarter our managers assessed this list. One day, my boss came to me and asked about a new chip he'd heard about. Would it work on our project to give us the speed we needed? I had no idea.

At the next portfolio evaluation review meeting, our project was reinstated. We had a couple of different people—it had been two years since we'd tried to do this project. But that was OK. It was cool to see that we could kill a project for excellent reasons and then reinstate it for excellent reasons.

4.5 How to Kill a Project and Keep It Dead

Think you've killed a project? Maybe you've never worked with Marty. Marty is a well-meaning manager who didn't want to kill a particular product or its associated projects because of his strong customer relationships.

I Kept Several Dead Projects Alive Until I Realized the Cost
by Marty, Group Manager

I was responsible for several products for three years. After the third year, my management decided to phase out Product2 in favor of Product4. So, I was responsible for Product1, Product2 phase out, Product3, and Product4. I was supposed to finish the Product2 phase out in two months.

Well, we did. Except, not all of our customers wanted to move to Product4. I've known these customers for years and had personal relationships with them. I wanted to be responsive, so I kept an active branch open on Product2 and provided updates to those customers for about a year. And then, my manager, Shelley, learned about project portfolio management.

Shelley realized I'd been staffing the Product2 phase out for more than a year, not the three months she'd expected. Luckily, Shelley was nice about it and didn't fire me. I'd explained that the customers had paid for