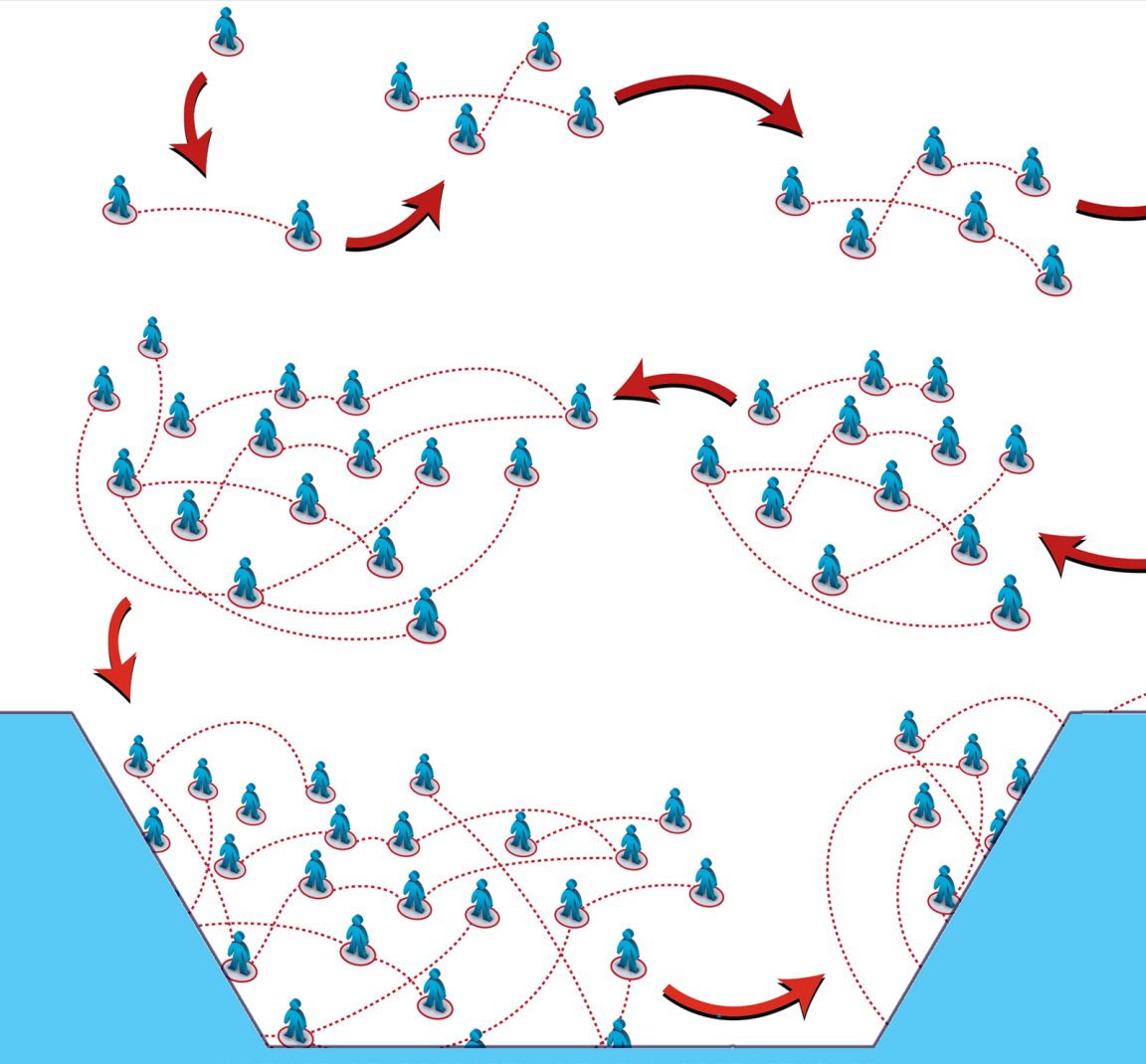
AGILE AND LEAN Program Management

Scaling Collaboration Across the Organization



AUTHOR OF "PREDICTING THE UNPREDICTABLE"

JOHANNA ROTHMAN

Agile and Lean Program Management

Scaling Collaboration Across the Organization

Johanna Rothman

ISBN 978-1-943487-04-2

Practical ink

capital letters or in all capitals.

For my family. Thank you for your support.

Contents

5. Use Continuous Planning

You've seen how to create your first roadmap and maybe even the first couple of product backlogs for any given team. Now, consider how you will update the roadmap and backlogs.

As you replan, consider how small you can make the features and minimum viable products. Your program will increase its throughput as the batch size remains small.

5.1 Differentiate Between Internal and External Releases

If you have continuous delivery, you can deliver something internally, to your organization, every day or multiple times a day. If you don't have continuous delivery, you might not be able to release every day.

Release something internally to your organization at least once a month. Releasing that often provides the entire program with feedback. It also provides a cadence that others will find dependable. When you release internally, you build trust across the organization. It makes sense to release as often as possible.

Internal releases help the feature teams to obtain feedback about the product. The internal releases will also show your management and sponsors the value of your work. Internal releases show people inside the organization what you have completed.

External releases show your customers what you have done. External releases are a business decision. Maybe your customer can take the updated product now, maybe not. However, the teams still need feedback on their work more often than once a quarter or whenever

your customer can take a release. This is why you need internal releases at least as often as once a month.

You can release internally more often than once a month. Make the once-a-month the *minimum* time between internal releases.

5.2 What Do You Want to Release This Month?

Teams need small features so they can integrate and release often. Even though you want to release something every month, it will be small. What do you want to release this month?

Let's assume you have two-week iterations. Two two-week iterations fit into one month. If you work in three-week iterations, you could release at the end of each iteration. If you work in flow, maybe you want to release every time you complete a feature, instead of two weeks. Maybe you want to release when you have a minimum viable product (MVP).

I assume you release internally *at least* as often as once a month. More often is great. The more often you release internally, the more everyone—the program participants, your sponsors, anyone interested in your program—can see your progress. Everyone sees feedback.

The less often you release, the more the feature teams have to estimate. With more internal releases, the product owners can change the backlogs. It's a win-win.

Create internal releases so everyone can see program progress. The larger the program, the more you need frequent internal releases.

If you use continuous delivery, you might not need the one-quarter agile roadmap as on Example of an Agile Roadmap for One Quarter.

Your program would release features faster than a product owner could maintain the roadmap.

Consider the lack of frequent-enough delivery an impediment. See if the feature teams can solve this problem, or if it is a program issue.

5.3 Create Minimum Releasables

From the big roadmap, you can generate something that allows you to see what your minimum viable products, your MVPs, are for each internal release.

Maybe the product owners for a given feature set say something like this, "We don't have something minimum unless at least 80% of the features exist." They are correct when they consider an *external* release. However, your program needs minimum *internal* releases.

Maybe instead of a minimum viable product, the product owners can consider a *Minimal Indispensable Feature Set*, MIFS (BRO14).

MVPs or MIFS will vary in size. Each feature set might need something different for an MVP.

"Our Product Grew Differently Over Time"

We had an email system as part of our product. We had an MVP of basic get-and-send emails in our first MVP. But, we didn't do forwarding or attachments until our second internal release. We didn't do group emails until our third internal release. We took other features from other feature sets, even though we were the "email" team.

I was surprised that the team didn't have such a difficult time with that. I had a harder time because I was the product owner.

> I wanted to finish the email system, already! But, the team saw where the product roadmap was going, and it made sense to them. They were okay with doing different features, and they had fun with it.
>
> They called themselves the "Email and… Team," because they did email and lots of other features. They said that knowing their MVPs made a difference for them.
>
> *—A feature team product owner*

Do not try to plan specifics of the feature sets/themes for more than one quarter at a time. Even one-quarter is a ton of planning. Note that you need to consider your MVPs for release.

If you restrict your planning to the MVPs for the internal releases for a quarter: what has to be in your MVPs for each internal release each quarter and then work towards that, you will do enough planning for most projects.

If you release something every month, you never have to do big release planning. If you update the agile roadmap every iteration, or after every few features when teams work in flow, you can direct the product development without big release planning. It's all about MVPs, minimum viable product. As long as you select your MVP for the feature set, or for the entire product, and create small stories, the teams will work towards that.

Continuous Delivery and Quarterly Planning

If you use continuous delivery, do you still need quarterly planning? You might.

If you need to commit across the organization or to customers,

> use a roadmap. The roadmap will show people the small items for product direction now, and the larger items later. Everyone can see the product direction.
>
> The fact that you do continuous delivery makes it much easier to deliver as needed and to commit to those predicted deliverables.
>
> The roadmap is a wish-list. The deliverables are the reality.

5.4 Plan for External Releases

If the product owners always define MVPs, and the teams always deliver MVPs, and the MVPs move the product towards the release criteria, no one has to worry about what goes into external releases.

If you have continuous delivery, you don't have to worry about external releases. You release all the time.

You have to worry about external releases when:

- The program doesn't release all the time.
- The feature teams don't do continuous integration and release what they have into the mainline.
- Teams work on architecture as opposed to features (when the feature teams don't create features).

If you get caught in these traps, the program has problems. Either the teams have problems at the team level, or the entire program has problems. The product owners can start addressing these problems by creating MVPs and making sure the teams deliver value, not architectural stories.

5.5 Deliverable and Rolling Wave Planning Helps

Internal releases are deliverable-based planning. The product owners specify the deliverable chunks they want to see. As the teams finish the chunks, they can take more.

Rolling wave scheduling is this:

- Schedule your next deliverable. Make sure that deliverable is no longer than two to four weeks away.
- At the end of your first week, schedule the next deliverable.
- Repeat, after each week.

Now you always have a two-to-four week schedule with deliverables.

The teams can use iterations or flow. It doesn't matter. Each team has this responsibility: provide a constant flow of value without incurring technical debt. See Continuous Integration and Testing Supports Collaboration for more information about ways to remove technical debt.

You or the program product owner might decide that the program can take some technical debt to meet a specific deliverable. (I don't recommend this.) As part of your deliverable-based planning, add the resolution of that debt to the product roadmap or a future backlog.

Using rolling wave budgeting and incremental budgeting is especially helpful if you have people who want to know how much the project will cost. You can update the spend and plan numbers with every release.

5.6 Small is Beautiful for Programs

Some people think as you create an agile and lean program, it's difficult to have short iterations. They tell me that because more people and teams are on the program, you need to make the iterations longer.

The problem is this: the more you want the benefits of agile or lean, the more you need feedback. The larger the program, the more frequently you need feedback. Why? You do not want to drive the company under while it is waiting for you to complete the program. The longer it takes to get feedback on any feature or set of features, the more difficult it is for the company to know whether the program is succeeding.

The larger the program, the more the organization spends on your work. You need to deliver—at every level—often. The value of making progress every day is that everyone gets feedback. People learn early if anyone is going down the wrong path. You don't have the opportunity to bankrupt your organization because you are not delivering.

If you Review the Twelve Principles of Agile Software Development, and Review the Seven Lean Principles, you can see that the principles are about delivering working software, as fast as possible. Shorter iterations allow you to do that.

What if the people on your teams think that short iterations encompass overhead for planning and estimation and, even retrospectives? There are several reasons for that.

- When you hear the word "overhead," you are hearing someone who has not yet fully transitioned to agile. Overhead is code for "we have impediments, and we don't yet realize what they are, so we call them overhead." These impediments might be large stories, and the lack of understanding that they can spike a large story to break it into smaller chunks; or

it could be a misunderstanding of what a minimum viable product could be.
- Those folks might not realize how little planning they need to do, to complete small deliverables and achieve an internal release each month.
- If your organization has not yet started to manage the project portfolio, people are multitasking among several projects or features. Under those conditions, you will have trouble building and maintaining a program of small features.
- You have a complex product, so the teams extend their iterations to more than two weeks to achieve some form of an MVP. I'll talk more about this in Shepherd the Architecture.

What if *you* think the iterations need to be longer? If you think planning is overhead, I bet you don't have small stories, or that you are trying to use estimation to manage the product roadmap or the project portfolio.

Start thinking about value. Start thinking about the smallest feature that will show everyone the progress of a feature or feature set.

5.7 How Often Can You Replan?

Continuous planning works in much the same way as continuous integration. When the feature teams integrate all the time, code integration is easier. When you replan all the time, the planning takes less time and is easier.

When you use continuous planning, you don't have to have big plans. You can plan for the next iteration (or two). You can plan for the next deliverable (or two). You never have to have everyone in the same room for release planning.

As the product owners see and accept the features that the teams complete in their backlogs, they can update the roadmap as a

product value team. Continuous planning avoids the need for a large "let's get everyone in the same room" to plan a quarter's worth of work.

Very few teams can plan for a quarter at a time and meet that plan. Your program might have interruptions from operations/support, the rank of some features might change, and teams encounter problems every day. If you plan for a quarter, you are not likely to accomplish everything you plan.

With continuous planning, you update the backlogs just in time and keep your program open to change. The smaller your planning, the more likely the teams are to be able to achieve the vision and release criteria.

Keep planning small. With small stories, small planning, and small teams, your program is more likely to have faster throughput and faster feedback. Small and frequent planning helps your program be more resilient.

The more you can move to continuous planning, the more agile and lean your program will be. The point of the roadmaps is to show the team the big picture of the product, and how that vision changes over time. The backlogs are the specifics for each team.

The more risk you have in your program, the more feedback you need. The more you want to keep the sponsors engaged, the more often you might have to change the roadmap—and by extension—the backlogs.

If you want more feedback, release more often. Can you release every day? If not every day, what impediments do you have for creating an internal release at least once a month? As a program manager, remove those impediments. Then you can ask the program product owner to update the roadmap at least as often as once a month.

"It's not the Plan; It's About Planning"

I used to use a more waterfall approach to my programs. I tried to plan once and have it be "the plan of record" for the entire program.

It didn't work so well. I was always replanning. Then I discovered rolling wave planning, and I learned about the value of *planning*, where we discussed what we could do when, and where the risks were, versus the actual plan, which was always out of date the next day.

Now, I use our planning as a way to understand problems and risks. I use the planning to help make decisions over the next few weeks. I never expect the plan to last past a couple of weeks. But I'm in better shape because of the risk discussions we had.

—A senior program manager

Good planning, in the sense of providing a roadmap for the teams and reflecting the current reality depends on more feedback, not less. When you plan less often, you don't see your current reality. The plans become targets, instead of plans the teams can use to guide their work.

5.8 Separate the Product Roadmap from the Project Portfolio

The larger your program, the more you might have projects in the form of feature sets to sequence. I like to think of this ranking as a form of feature portfolio management. The sequencing occurs when you say something like this, "We need to work on enough