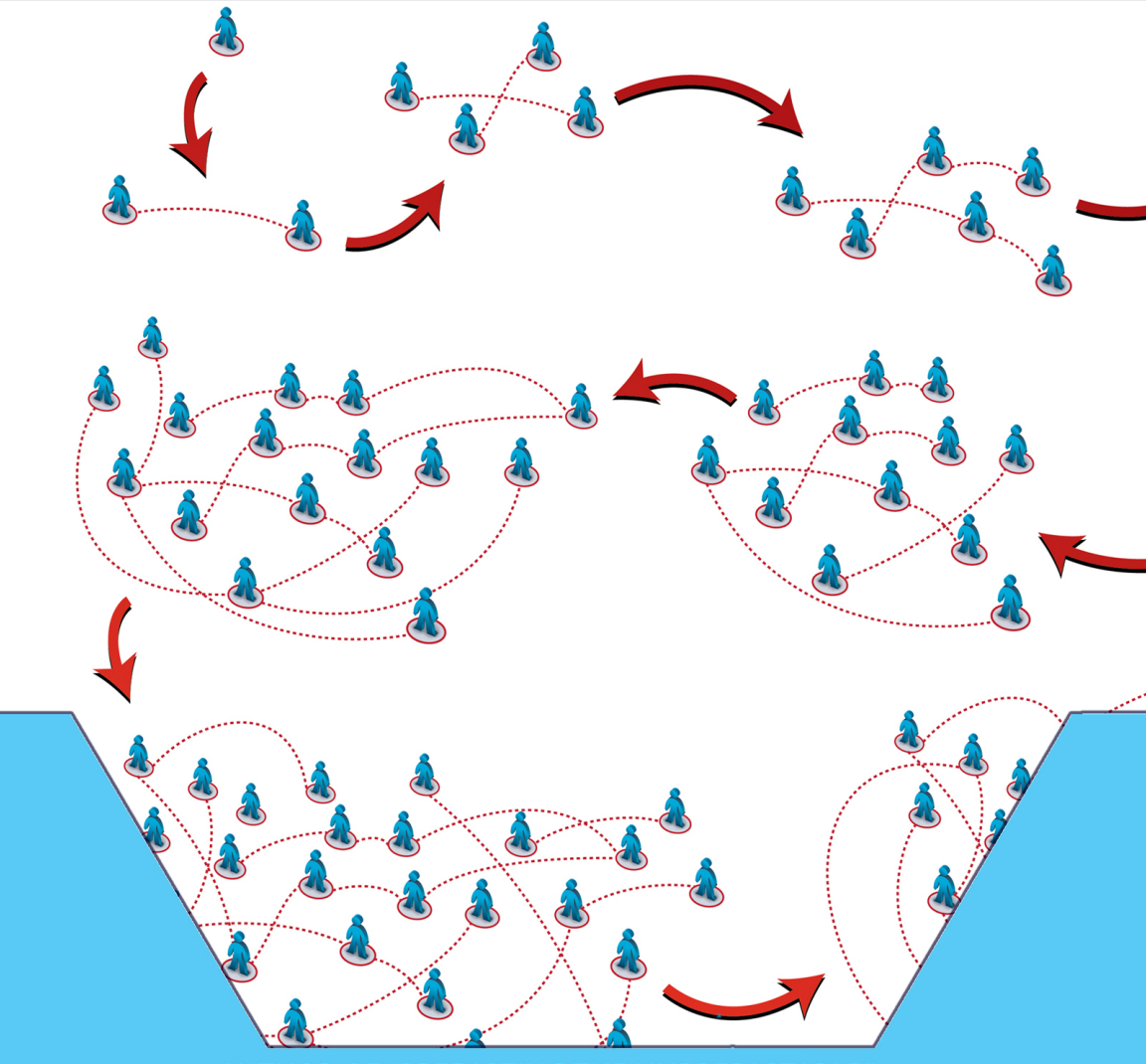
AGILE AND LEAN Program Management

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

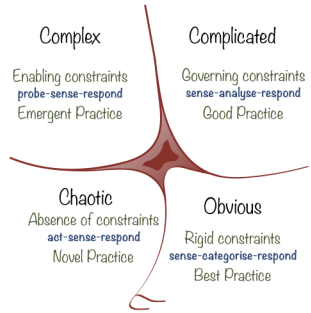
capital letters or in all capitals.

2. Consider Your Program Context

You and all the members of your program will make multiple decisions on a daily basis. The Cynefin Framework is a way of thinking about your context with the intent of guiding your actions. I use Cynefin to think about how I solve problems: Can we use good practices that everyone else uses? Do we need to experiment to know how to proceed? Do we have so many unknowns that we don't know where to start?

2.1 Cynefin Helps with Decisions

The Cynefin Framework (SNB07) is a sense-making framework you can use to solve problems. Use it to guide your approach to your program.



Cynefin Framework

Based on the fact you are working in a program, you are not in the Obvious context. A program, by its very nature, is at least in the Complicated context, because of the number of communication paths.

If everyone is in a single physical location, you may be in the Complicated context. In the Complicated context, you can see straight cause-and-effect relationships among the different stresses in your program. If all your teams are experienced agile or lean teams, who know how to deliver small stories each day or so, you might be in the Complicated context. You understand what your unknowns are. You can use known and reasonable practices for organizing and working on your agile program.

As soon as you and the people in your program are not in the same location, you are no longer in the Complicated context. You have moved into either the Complex or Chaotic context. That's because your communication will have delivery or communication lags and other interferences. Problem causes or effects may be unclear and even unknown, if only due to communication lags.

If people on your program are multitasking, or if you have people or teams who can't commit to the program, or if many of your feature teams are new to agile, you are at least in the Complex context. You may be in the Chaotic context. In either of these contexts, the unknowns create many risks and potential problems.

In my experience, if you can say, "We have done work like this, but never at this complexity or with this many teams, or never as distributed as we are now," you are in the Complex context. You have many unknown unknowns. You will have to manage the risk of those unknowns.

As you look at the Cynefin Framework, ask yourself: what context reflects your reality? How will that context help you decide whether you should sense, probe, or act as an experiment first?

If you are in the Complicated part of the framework, you need experts to solve the problems in your program. I'm not talking about experts that create bottlenecks by working alone. Instead, develop a community of experts—maybe most of the people on your program, working in their Communities of Practice—to help solve the problems.

If you are in the Complex part of the framework, consider these actions: What experiments will you use to probe, to discover your unknowns? And, what problems can you solve to move the program back to the Complicated part of the framework, where you can know your challenges?

Cynefin is not a two-by-two matrix where you locate your program, use that to make decisions, and never return to the framework. Instead, especially with programs of nine teams or more, different parts of the program will have different challenges. The more unknowable the challenges, the more that part of the program is in the Complex part of the framework. As the teams deliver features, they learn more. That part of the program moves to the Complicated part of the framework.

Sometimes, teams in the Complicated part of the framework finish features. As they learn, they uncover a huge "gotcha." That might cause them to be in the Complex part of the framework until they run some experiments to see what they can do.

As a program manager, how can you identify issues early when you encounter Complex again? How can you help the program move from Complex to Complicated?

There are no easy answers. There is no recipe. This is work. It's the reason why we need program management, to recognize and solve problems across the organization.

The Cynefin Framework reveals why agile program management can be difficult. As teams complete their features, the product owners need to update the roadmap and the backlogs. It's possible the program will finish before expected. Completing—or not—other projects or programs may affect the organization's project portfolio. Certainly, one team's feature completion might affect the ability of other teams to deliver.

Regardless of your context, a program is emergent. With emergent projects, you can't plan everything at the beginning. You can see a roadmap, plan a little, and continue learning and adapting as you proceed. You might want to keep the same vision of the product, but teams (with their product owners) might select different work. Or, as your customers/product owners see the product, they might want to change the product direction.

If the teams don't complete features on a short, regular basis, no one can understand what the program status is. If the core team doesn't solve problems that allow the program to create a *product*, you have plenty of risks, many of them unknown.

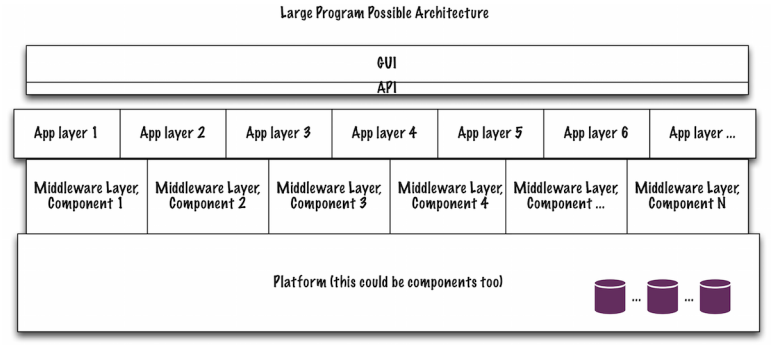
Manage by principles, not practices.

With your risks, consider principles for your program, not practices. I could try to create a recipe for you, but that won't work. Think and recognize your context.

2.2 Understand Your Product's Complexity

Your program is unique. Your program may have complexity in a variety of areas: architecture, pressure to release, where the people sit in relationship to each other, the languages everyone uses, and each team's agility.

In my experience, the overall architecture of your product can drive much of the complexity. The more complex the architecture and the larger your program is, the more complexity you will have to manage. Here are some program architectures I have seen.



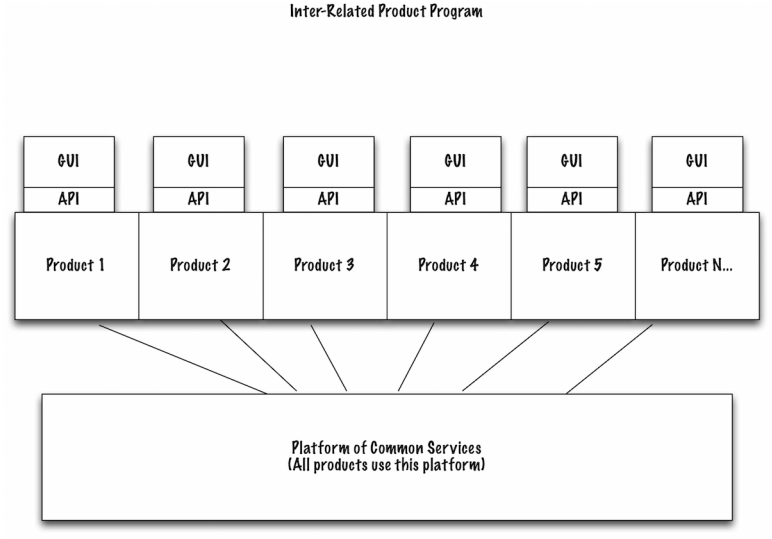
Large Program, One Coherent Product

In this case, you have one large product. It's not integrating other products or systems. Your program creates the entire product. It's big with multiple feature teams, which is why you have complexity in your program.

As an example, an operating system might look like one coherent

product. Maybe a large web-based store might look like one coherent product.

Inter-related products are different. If you ever say, "Platform and layered products," you have an example of an inter-related product.



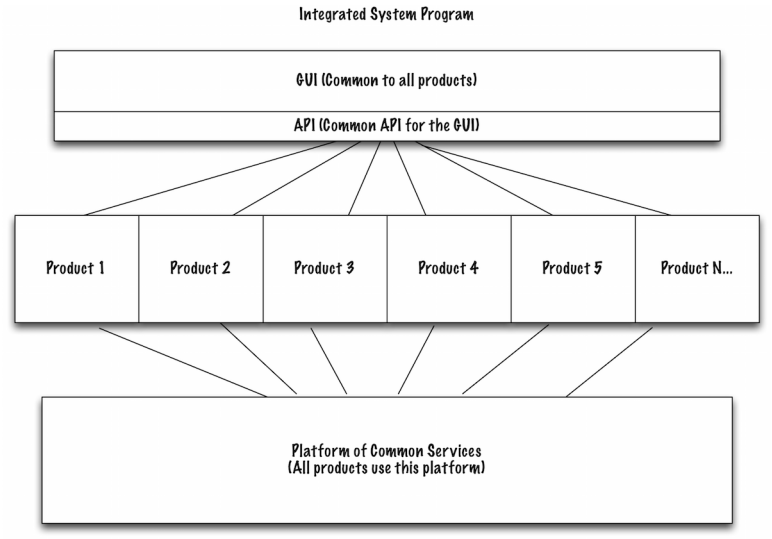
Inter-Related Product Program

In this case, you have a platform of common services with what feel to the customer as separate products. The GUIs may have their own look and feel, but the GUI is not common across your program's product.

As an example, a smartphone is an integrated system product. Each app on the phone has its own GUI where you set the preferences and use the app. Each app uses services from the phone's operating system.

Sometimes, inter-related products integrate other products into the one product.

It's more likely if you integrate other vendors' products into your own, that you have an integrated system program.



Integrated System Product Program

In this case, customers buy your entire product. The product still has the platform of common services. However, you have one coherent GUI that the products have to integrate with. You might be integrating systems or hardware from vendors.

These programs tend to need programs of programs. Different products will run on their own schedule. Unless your vendors are also agile and lean, you may have to manage integration risks.

2.3 Know Which Program Teams You Need

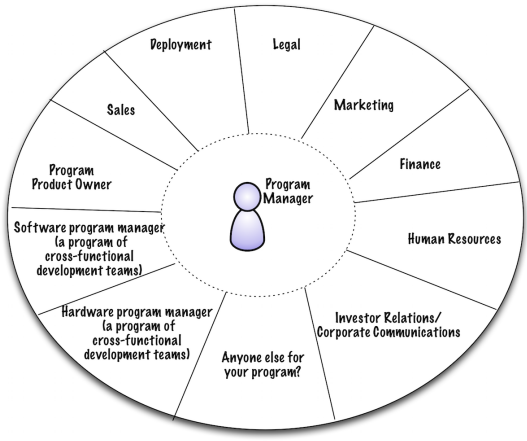
Every program needs the ability to work across the organization. You might need a *core team*, the cross-functional business team that has members from all around the organization. The core team helps coordinate the efforts that make the entire product a successful deliverable.

If you have more than two feature teams, you might also need a *software program team*. The software program team helps deliver the working software. The software program manager is a delegate to the core program team. That means that the software program manager must have a program team of his/her own. This is true for a large program.

You, as a program manager, need to understand which program teams you need. Does your program require both a core team and a software program team? Do you need a core team program manager and a software program team manager?

You can only manage one program team. One of the problems I see in too many agile programs is that they have neither a core team nor a software program team. They have many feature or component teams. They might have Scrum-of-Scrum meetings, but no real forum for solving the deep problems or managing the risks that can occur across the organization.

Each program team has a responsibility to solve problems that the teams it represents can't solve by themselves. The program team, whether it is a core team or a software program team, works across the organization, solving problems and removing obstacles for the program.



What Your Core Team Might Look Like

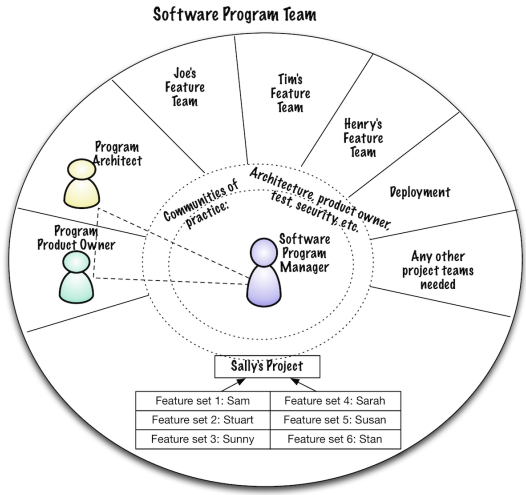
If you are coordinating and collaborating across the entire organization, you are managing or are a part of the *core* team. If you take a look at the What Your Core Team Might Look Like, you can see that there are plenty of potential participants on this program team.

Aside from the program manager, there is the software program manager, the potential hardware program manager, the program product owner, as well as the sales, deployment, legal, marketing, finance, human resources, and investor relations project managers. And those are only the people I could imagine. There might be other or different people in your organization.

Do You Have A Process Program?

Sometimes, organizations run process improvement projects, such as transitioning to agile, as if they are programs. That's fine. In this, your core team will look different. You might not have feature teams in your program.

> Know what kind of a program you have. Not all programs are the same. Use a core team that makes sense for your program.



What Your Software Program Team Might Look Like

Take a look at What Your Software Program Team Might Look Like to see a prototype composition.

Notice that the program product owner and the program architect might work as a triad with the software program manager to make risk decisions. Does this mean that the program product owner does not work with the core program manager?

It depends. It depends on who needs the program product owner. Maybe you need a product owner team, and the program product owner works with the core team and the technical product owner works with the software program owner. It depends on what your program needs.

Look at the program architect. Your feature teams need their