

CHAPTER 3

The Big Picture

“It is only possible to make a place which is alive by a process in which each part is modified by its position in the whole.” —Christopher Alexander

IN THIS CHAPTER

This chapter discusses the larger goal of going for Agility at the enterprise level. What is involved? What is required to create real value for the organization? Everything must drive toward delivering value for the overall organization.

Takeaways

Enterprise Agility requires looking at an organization’s entire value stream—from idea to implementation, from concept to consumption. In order to achieve Agility, several enterprise areas must be addressed:

- Identify the products whose creation or enhancement will make the greatest impact on the company’s bottom line.
- Match these product enhancements (projects) to the organization’s resources.
- Manage these projects so the product enhancements are achieved with the greatest quality and speed possible.
- Organize the software-development teams so they can work with each other in the most effective manner.
- Use proper software-engineering methods both to support the project management and to ensure long term viability and low-cost sustainability of the products created.
- Create a learning environment so that the process is continuously improved.

Aiming for Enterprise Agility

Many people have focused on introducing Agility at the team level. This can be the easiest place to start but often it is not the principal challenge that a development organization faces. And it definitely isn't where to end.

The real goal should be to create *enterprise* Agility. That is, the enterprise needs to be Agile. It needs to be able to respond to external competitive forces, better understandings of the marketplace, its own errors, changing technologies, and anything else that can have an adverse or positive influence on the enterprise. Team Agility is necessary but not sufficient. Enterprise Agility requires team Agility; but team Agility is only a means to an end: Enterprise Agility. Enterprise Agility enables a company to deliver higher quality products and services to their customers at a faster rate than their competition. This is a strong competitive advantage in any industry.

Getting to Enterprise Agility

Enterprise Agility requires looking at the entire value stream of an organization. By “value stream,” we mean the stream of delivered software solutions that flows from the delivery organization to the customer or consumer of those solutions, driven by business need. By focusing on the value stream, enterprise Agility clearly shows the time required for ideas (triggered by market opportunities, competitive threats, or business need) to flow through the entire life cycle through to deployment and use. As much as possible, we seek to minimize this cycle time and to remove waste and delays in this flow.

Figure 3.1 shows a rough timeline of a project. At some point, the project is envisioned. Management determines what value the project



Figure 3.1 A rough timeline for software development from initial concept to consumption

might have, how much it will cost, and whether it should be initiated. Once a project is approved, it gets staffed; shortly after that, software development begins and then it is deployed and supported. Feedback should occur throughout the development process, but once deployed, a new level of feedback is also available—how customers are using the software and what real value it has contributed to the business.

Unfortunately, many companies attend to only a part of the entire timeline. For example, one client hired us to help make their teams Agile. They said their projects were typically six months long. When we casually asked how long it took to go from idea generation to project initiation, they said, “Two years.” That means that, start to finish, a project runs for two and a half years and development is only 20 percent of that time!

This was surprising. Surely, they did not hire us to help them with only 20 percent of their problem. Yet, they asked us to focus on development because that is where they were spending most of their software-development dollars. True, they spent 80 percent of their budget during these six months. But Lean tells us we need to focus on the time spent, not the money. Focusing on removing delays by improving quality results in quicker time to market and lower costs. Focusing on lowering costs typically degrades quality and takes longer. Time and again, focusing on lowering costs instead of removing delays and improving quality slows development and lowers quality. Even focusing on the development time may not improve the value stream since it represents only one part of it. It is crucial to look at all of the work done from envisioning the product until it is actually deployed. Had we helped them with only their project management, we could have helped them with only that small 20 percent included in the Agile/Scrum ellipse (shown in Figure 3.2). We would have missed the greater opportunities for improving the steps prior to project initiation.

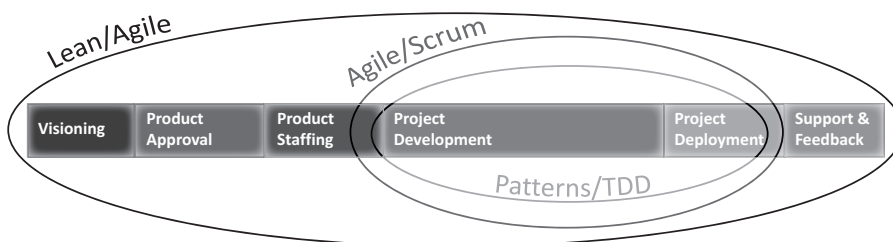


Figure 3.2 Where Lean, Agile/Scrum, and technical methods apply to the software-development timeline

Time after time, this is the case. In theory, Agile could apply broadly. In practice, most Agilists focus on the local team or teams and, as a result, concentrate mostly on the project level and the challenges of the team. Too often, they view issues of staffing, project and process, and impediments through the lens of their impact on the local team. They consider technical disciplines, such as design patterns and test-driven development (TDD), because these help teams get their work done. And make no mistake: Scrum and other Agile methods do make teams much more effective than they were before. But more guidance is needed to address the challenges of multiple teams working together across an entire organization. Lean thinking inherently offers the approach and guidance required to address the entire value stream.

How to Create Real Value for an Organization

To improve software development, several essential areas must be addressed, including:

- Identify the products whose creation or enhancement will make the greatest impact on the company's bottom line.
- Match these product enhancements (projects) to the organization's resources.
- Manage these projects so the product enhancements are achieved with the greatest quality and speed possible.
- Organize the software-development teams so they can work with each other in the most effective manner.
- Use proper software-engineering methods both to support the project management and to ensure long-term viability and low-cost sustainability of the products created.
- Create a learning environment so that the process is continuously improved.

Identify Value

In many organizations, the identification step is done by the software department itself (IT in IT organizations and product development in software product companies). This is the wrong driver—the tail is wagging the dog. Software creation or enhancement should be driven by business

needs—so it should be driven by the business-management end of the organization.

The question is not, “What can we technologists build” (a software-development perspective) but rather, “What products will return the greatest value to the business?” and “When can the business or customers start using them?” (business-driven perspectives).

That last question is important. Just because software has been released does not mean the business realizes value from it; there is often a lag between release and use. People must be trained, product must be shipped, support processes must be set up, and marketing and communication must be implemented. The value does not begin until all of that is in place. The relevant value stream must extend from when the idea is formulated until value is realized.

Manage the Organization’s Resources

The difficulty of assigning resources to projects varies from company to company. Mary and Tom Poppendieck (Poppendieck and Poppendieck 2003) say that we should adopt a *product* focus, rather than a project focus, in our development efforts (and therefore in our staffing practices). Project thinking assumes a well-defined beginning and end to the work. And it staffs for the project rather than for the life of the product: Resources are put together and disbanded. Ideally, the same people could work on a product throughout its lifetime. However, resources often become scarce and it becomes increasingly difficult to keep teams together in the most effective way.

When many projects are in process, the contention for resources becomes so great that projects get scheduled based on resource availability or political clout rather than what would contribute the most value to the business. We’ve seen extreme cases where it seems that teams don’t even exist; rather, there are several people working on a project together while they also work on other projects with other people. In organizations like this, creating a business focus is often the impetus for creating effective teams. Project thinking virtually guarantees inefficient use of personnel.

Suppose we have a company that is organized as shown in Figure 3.3. This is a typical organization. Resources are assigned to projects one at a time according to project need. For example, Figure 3.4 shows how staff will be assigned to Project 1: a business systems analyst, an architect, three UI people, and so on.

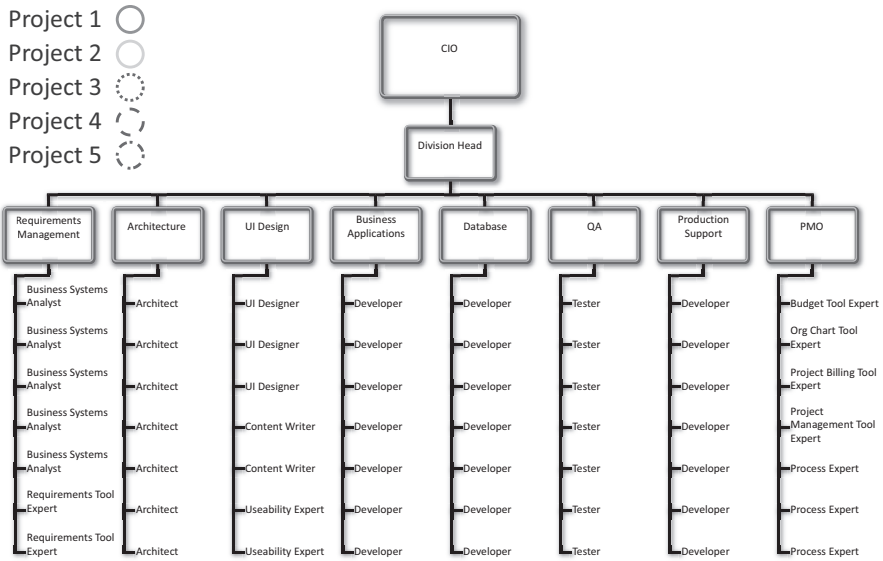


Figure 3.3 Matching project needs to available staff

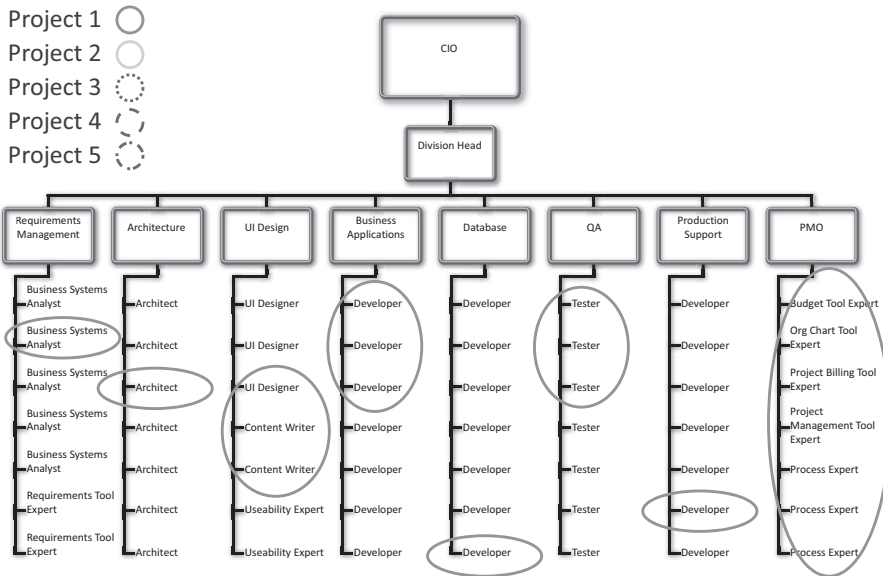


Figure 3.4 Matching resources for one project

The next project also needs resources: a requirements tool expert, an architect, some UI designers, and so on. Note the new circles in Figure 3.5.

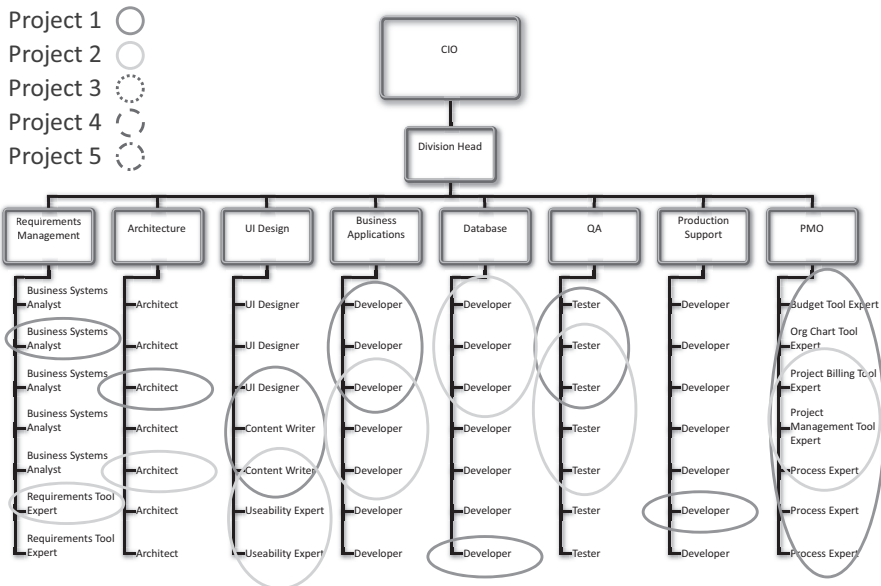


Figure 3.5 Matching resources for another project

Continue staffing the projects in this way and you get something like Figure 3.6.

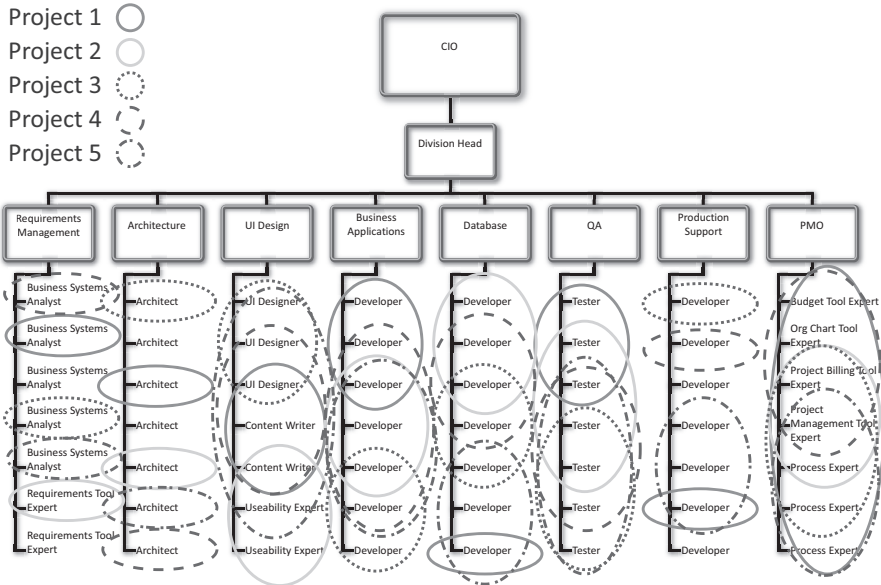


Figure 3.6 Resource map for all projects

Unfortunately, this causes many problems, including:

- People are assigned to multiple projects.
- We now require projects to start and stop on time so that resources are available as needed; that is, we are assuming a predictable future.
- Resource constraints make it impossible to accelerate the truly critical projects.

The first problem, having people work on multiple projects, is particularly bad. Studies¹ have shown that when people are working on even two projects at a time, their efficiency decreases by 20 percent. If they work on as many as three projects at a time—and this is not uncommon in IT organizations—their efficiency drops 40 percent! They are operating at only 20 percent of their capacity on each project. This has a staggering impact and it is often ignored. The continual stopping, restarting, and waiting for information causes a tremendous productivity loss.

It is even worse for key staff. They are certainly being pulled in many more than three directions at once. As keepers of some crucial information, as many as five or six projects are vying for their attention at any given time. This drives their productivity to very low levels—people who, given their essential knowledge, should be the most productive. Just as damaging, most of the others in the development organization waste time waiting for these people—further degrading the productivity of the entire organization.

The second and third problems mentioned above make this even more challenging. Lean thinking provides a better way.

Manage Projects

Once the product creation or enhancement has been committed to, and resources allocated for it, we must be able to efficiently and effectively manage the projects themselves. This is where Agile and Scrum come in. As we have already seen, these projects are not being done in isolation. If we attempt to manage the projects individually and merely coordinate them while doing so, we are likely to miss a variety of opportunities as well as cause great inefficiencies.

1. For one example, see (Aral, Brynjolfsson, and Van Alstyne 2008).

Agile project management done in isolation violates the Lean principle of optimizing the whole. Chapter 5, *Going beyond Scrum*, describes how Scrum can be readily extended to include Lean principles and be much more effective than Scrum by itself.

There are other alternatives to Scrum, including Crystal, feature-driven development, and Kanban software development. They are good and they address specific challenges for teams; however, they, like Scrum, do not entirely address the bigger picture.² By themselves, they do not address the entire value stream, which is what is needed.

Proper Software Engineering

Although software development itself may not be the limiting factor in organizations, ignoring the best practices of software development eventually results in a major impediment: poor code quality. Software must be well written and properly tested to be sustainable over the long haul. Our experience dictates that emphasis on quality design through the use of design patterns and up-front automated testing are essential protocols that need to be understood and applied by the development team.

Case Study: Financial Services

One of our consultants was working with the IT department of a large financial-services company. The team was working on a software product that supported the company's consumer financial-service offerings (for example, IRAs or money-market funds). The team was getting ready to roll out some product enhancements that would greatly simplify the experience for the consumer. In fact, they based their business case on the hope that these enhancements would reduce support-desk headcount.

One of the enhancements was “basic” functionality—the usual experience the consumer would have when things were working fine. Another enhancement supported consumers when things were not so simple. As you can imagine, the second enhancement was much more complex. They developed three estimates for completing the work:

- Time to do both enhancements: nine months
- Time to do the basic enhancement: six months
- Time to do the second enhancement (handling complex situations): four months

The extra month was required because they would need to re-factor the database schema and transition the data from the first release to the new one.

2. This is not necessarily true of Kanban, which can be used to include the beginning part of the value stream. We are referring here to when Kanban methods are used to manage only the development team.

What would you say? What would you expect the business to say?

The development team felt that splitting up the project was a bad idea because it would take them an extra month and would require extra work. Looking chiefly at costs to themselves and the project, this might seem reasonable. However, the business saw it differently. Releasing the first enhancement three months early would result in cost savings that greatly outweighed the extra month of development. In fact, they gained so much with that first release that they decided not even to do the second enhancement. They could realize more value by having the developers do something else and retaining the support-desk people to handle those complex situations.

The point is that simply focusing on speeding up development teams is insufficient. Instead, you have to see the product in the context of the bigger picture—the entire value stream. Doing so lets the enterprise see all the alternatives for achieving its objectives, realizing more value (and more cash), and satisfying its customers. Agile approaches help teams do well; adding Lean thinking and technical best practices extends the enterprise's capacity to optimize the entire value stream.

Summary

Transitioning to Agile software product development is not done in isolation. It operates in the larger context of integrating Agility at the enterprise level. Enterprise Agility entails looking at the entire value stream of an organization, from initial concept to when the customer can utilize the product.

Try This

These exercises are best done as a conversation with someone in your organization. After each exercise, ask each other if there are any actions either of you can take to improve your situation.

- How is your product-development staff organized? By product? By skill set? By process activity?
- Sample your organization for the average number of projects assigned to each person or group. Can you justify the productivity losses due to task switching?

- How can you reduce the amount of work-in-process with your current organizational structure?
- How can you restructure in order to bring control to the number of projects in flight?

Recommended Reading

The following works offer helpful insights into the topics of this chapter.

Aral, Brynjolfsson, and Van Alstyne. December 2008. *What Makes Information Workers Productive*. <http://sloanreview.mit.edu/smr/issue/2008/winter/12/> (accessed October 2008).

Bain. 2008. *Emergent Design: The Evolutionary Nature of Professional Software Development*. Boston: Addison-Wesley.

Collison and Parcell. 2004. *Learning to Fly: Practical Lessons from One of the World's Leading Knowledge Companies*. Milford, CT: Capstone.

Poppendieck and Poppendieck. 2003. *Lean Software Development: An Agile Toolkit*. Boston: Addison-Wesley.

Townsend and Gebhardt. 2007. *How Organizations Learn: Investigate, Identify, Institutionalize*. Milwaukee, WI: ASQ Quality Press.

