

# Chapter 14. Seeing Lean

---

*What's in a name? That which we call a rose by any other name would smell as sweet. —William Shakespeare*

## In This Chapter

This chapter summarizes the ideas of this book—where we have been and where Lean and Agile fit in the context of software development. Lean may have its origins in manufacturing—Toyota is the greatest example—but it applies much more widely. We approach Lean by viewing it as a combination of three bodies of knowledge:

- Lean “science” (principles of pull, theory of constraints, flow)
- Lean management (management and teams can work together)
- Lean knowledge stewardship (how we can learn, coach, and keep alive our knowledge)

It is this approach that lets us apply Lean so powerfully to software product development. We include several descriptions of cases from our consulting experience that illustrate this.

This chapter shows how Lean’s mantra of fast, flexible flow helps to create a better development organization; then it concludes with suggestions for the next steps you can take to apply it to your organization.

## Takeaways

Key insights to take away from this chapter include

- Delivering smaller, well-defined features faster results in greater efficiency for teams.
- Fast delivery forces product managers to cooperate with each other.
- Lean is based on a century of practical application in a variety of contexts and it continues to evolve.
- Lean combines many levels of thinking.
- Factors that improve flow include
  - Working on smaller, well-defined, high-value features
  - Limiting the amount of work-in-process to capacity of the teams
  - Removing delays wherever possible
  - Looking to the system for causes of waste and continuously improving the process.

## **Toyota: The First Great Example of Lean**

In the 1950s, several forces combined that helped Toyota create Lean. Toyota faced a difficult situation. They didn't have a lot of experience in making cars. They were competing against the United States—the powerhouse of mass production at the time. Rather than competing directly, Toyota was forced to look closer to home at niche markets, which meant they could not build many units at any one time. This required making small runs of several models instead of huge runs of a few models. Impediments to this were enormous. For example, the turnover time for die-cast equipment took weeks. To compete, these times had to be reduced.

They did not know exactly what to do.

Ironically, this lack of knowledge was an advantage. They knew they had to rethink everything; and because they didn't have a large investment in what they knew, they could invent knowledge from the ground up.

One more force was at work: The U.S. Army brought Edwards Deming—one of the great pioneers of quality improvement—to Japan to help with reindustrialization. The Toyota Production System (TPS) was built on his ideas.

Taiichi Ohno, the implementer of Lean at Toyota, says<sup>1</sup>

*The basis of the Toyota production system (TPS) is the absolute elimination of waste. The two pillars needed to support the system are:*

- *Just-in-Time*
- *Autonomation, or automation with a human touch*

Just-in-Time means that, in a flow process, the right parts needed for assembly reach the assembly line at exactly the time they are needed and only in the required amounts.

Autonomation requires processes to run smoothly. When they don't, we don't just fix the symptom, but fix the root cause of the problem. Whenever there is an impediment, remove it. This is why management must be involved in autonomation: Almost always, root causes touch multiple systems or cross organizational barriers. This does not mean that managers fix the problems themselves or tell their staffs what to do but, rather, they lead and coach their teams in implementing change.

As it evolved, the Toyota Production System, founded on Deming, was composed of three parts: Lean practices, quality management, and continuous learning and improvement. This is shown in Figure 14.1.

---

<sup>1</sup> (Ohno 1988, page 4)

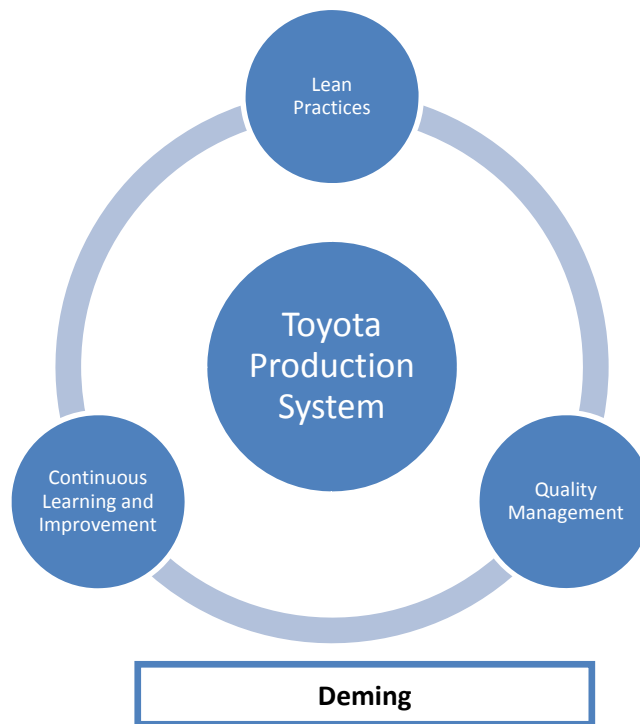


Figure 14-1. The Toyota Production System, built on the Deming foundation

Lean is focused on customer value; however, what that means depends on the context. In manufacturing, you more or less know what product the customer wants; the focus is on delivering that value (the product) to the customer as the customer expects it. On the other hand, in product development, you do not necessarily know what the customer wants; your focus is on learning, on improving your understanding of what the customer wants and needs. One of the brilliant insights at Toyota was that Lean principles are implemented differently in manufacturing than they are in product development.

This gave rise to another great example of Lean: the Toyota Product Development System, which is a better example for us in software development.

Let's consider Toyota as one of the classic examples of Lean thinking. What is important is not exactly *what they did* but *how they applied Lean principles to their context*. Looking at what they did in the Toyota Product Development System will give better insight to our sort of work.

For example, one of the practices frequently mentioned in the Toyota Product Development System is "Set-Based Concurrent Engineering" (SBCE), which is a very good risk-mitigation practice. In building physical products, establishing the right approach is often critical and difficult. What about in software development? Don't ask, "How can we apply SBCE to software?" Instead, ask, "What is the best *risk management strategy* we can apply to building software products?" (Risk mitigation is the objective behind the SBCE practice.) This may lead you to a different approach; you may decide that SBCE still makes sense, but you may choose to rely on proper encapsulation of unknown issues or other means.

## Three Bodies of Lean

Over the last 60 years, Lean has been applied in many organizations and contexts including, lately, software<sup>2</sup> development. We have learned to view Lean not as a collection of specific practices, but as a combination of three important bodies of knowledge (see Figure 14.2):

- Lean science (principles of pull, theory of constraints, flow)
- Lean management (management and teams can work together)
- Lean knowledge stewardship (how we can learn, coach, and keep alive our knowledge)

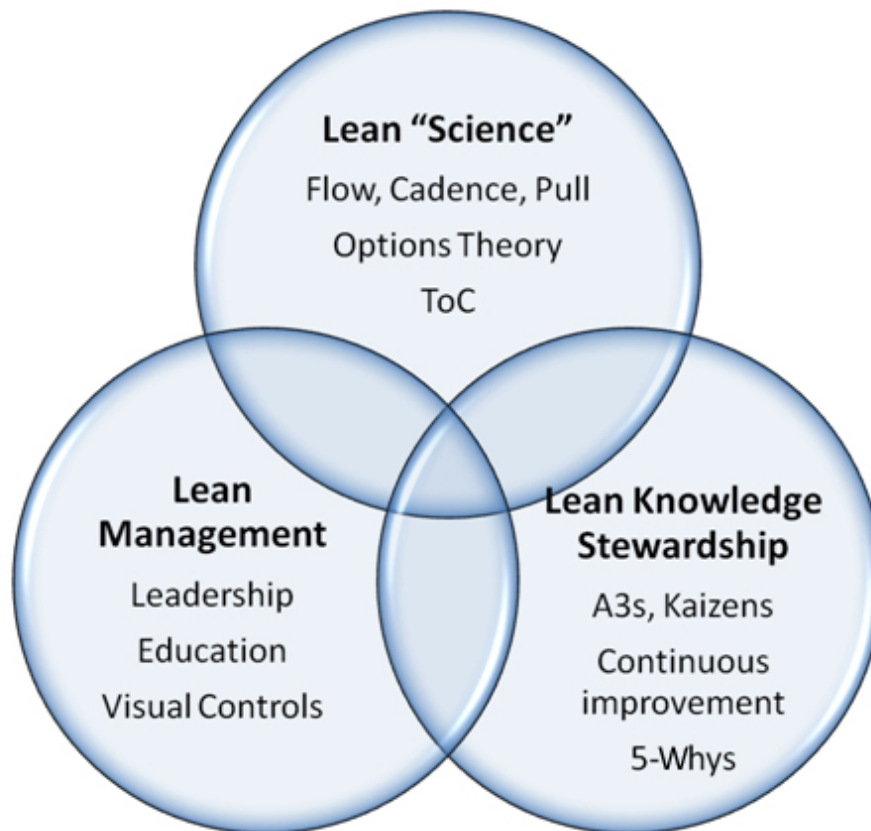


Figure 14-2. Lean thinking is based on science, management and knowledge

### Lean "Science"

Lean Science lays out the rules that Lean tells us to follow in product development, including:

- Just-in-Time
- Utilization theory
  - Small queues and batch sizes

---

<sup>2</sup> Little's law establishes the relationship between cycle time, WIP, and throughput: Cycle time = WIP/Throughput, and the key insight is that to reduce cycle time, we should start by reducing WIP.

- Limit WIP
- Little's law
- Causes of thrashing
- Pull management
- Real options

We call it Lean science to underscore that there are rules (like gravity), and we violate them at our own peril. Learning these rules increases our ability to accomplish what we want. And, like the scientific method, we can make hypotheses about how our work relates to these rules and then test those hypotheses by seeing the results of our work.

### **Lean Management**

Lean Management emphasizes the responsibility managers have for their team's performance. Far from being a micro-manager, the manager teaches the team to implement a new process, which includes helping the team discover the specific work-flow they need to follow. Managers become leaders, coaches, and trainers. They are not "servant leaders," in that they have to take a proactive approach to helping their teams. Perhaps it is best to say that they are fellow workers who are responsible to lead.

Because Lean has a science we can build on, it provides an opportunity for management to help their staff both learn this science and apply it.

### **Lean Knowledge Stewardship**

Lean's focus on continuous process improvement concerns many parts, including understanding how product/software development works; understanding our own problem domain, our own challenges, and how they relate to this; and understanding our customers' (either internal or external) needs.

There is clearly a lot of learning taking place here—a lot of information that must be transferred. How we manage this knowledge has a great impact on our effectiveness. Lean knowledge-stewardship includes the proper use of:

- A3s
- Kaizens
- After Action Reviews (AARs) and retrospections
- The Five Whys, getting to root cause practice
- Value-stream mapping

We won't go into any more detail, as there are a great many resources already available. However, it should be clear that obtaining, retaining, and moving knowledge around is critical. Teaching people how to learn is just as critical.

## Insights from Lean-Agile Coaches

It is perhaps helpful to have this higher-level perspective. Let's turn to real life. During our courses and coaching engagements, we have had some "Aha!" moments that have given us significantly deeper insights into Lean-Agile thinking. We hope the illustrations in this section will help you, too.

The stories are real but, of course, names have been left out to protect confidentiality.

### Focusing on One Project at a Time

The context and insight include the following:

**Company profile:** Develops software used by insurance companies

**Challenges:** Lots of projects, sometimes little clarity on requirements, looming deadlines, overworked development team

**Insight:** One software development director said, "I have two projects. One is for a key client who knows what they want and the enhancements we are working on will make a big difference to them. The other client is not as big, but more importantly, aren't as clear about what they want. I see that instead of having my team working on both at the same time, I should have them work on the project for the client who has clarity, but keep the work focused on the Minimal Marketable Features so I will get it done quickly. In the meantime, our product managers can be talking to the other client to get better clarity. Then we can focus on them. At the least we'll get value to our more important client quicker while being more efficient."

### Initiating Fewer Projects Instead of Imploring Teams to Work Better

The context and insight include the following:

**Company profile:** Health insurance company

**Challenges:** Lots of projects, non-co-located, non-matrixed teams

**Insight:** After a presentation on Lean's view of software development as fast, flexible flow, and underscoring how much waste results from pushing too many products through the development pipeline, a vice president of the company declared, "Oh, I see, I should be giving my development teams fewer things to work on instead of imploring them to work harder."

### Shortening Batch Times

The context and insight include the following:

**Company profile:** Large software-development company

**Challenges:** Lots of bugs with long cycle times to fix them.

**Insight:** The truly Lean approach is to get to the root cause of the bugs. Two directors were trying to determine how to address and prioritize the great number of bugs facing them. Finding

a root cause on all of them felt overwhelming. They realized that if all they did was work in three-month batches instead of six-month batches, they would get many more high-priority bugs fixed and released sooner. Although it didn't get to root cause, improvement was available by simply changing batch size. Sometimes it's worth getting what you can when that is virtually free.

## Getting to the Root Cause

The context and insight include the following:

**Group profile:** Member of a support group for a large IT organization

**Challenges:** Lots of internal-customer calls

**Insight:** Once he understood that more problems are of a systemic nature than because of the person using the system, one member of the team reconsidered how they handled support. Instead of just answering questions, he started asking, "What is the root cause of this question?" Since he was not able to change the computer system that the users were having troubles with, he answered that question in terms of the support system that was available to the users who had questions. He realized that he could improve the support system by getting to the root cause—why was manual intervention required? After one year of doing this, and improving the automated support, he found it took half the time to handle twice the number of users—a fourfold increase in value added by his group.

## Knowing Where You Are: Minimum Releasable Features

The context and insight include the following:

**Group profile:** Makes administrative software used by large healthcare organizations

**Challenges:** After committing to deliver a large program (two-plus years, several million dollars), the organization was not confident about their progress after five months

**Insight:** The organization was focused only on the end date, which was two years away. There was no sense of priority for any of the work that was underway, as it "all had to be done." Without this priority (sequence of business features) set, the teams had no control over how much work was started, and no visible work was completed. Once the organization came together and visibly sequenced features, the teams began to utilize iterations to deliver the features one by one. The key insight gained by the organization was that by working this way, they did *not* have to wait until the far-off date to release. Instead, they prioritized and sequenced the features so that smaller releases could begin, which enabled them to keep key customers happy without having to wait two years to deliver the program.

## Priorities and Work-in-Process

The context and insight include the following:

**Group profile:** Creates reporting software used by large healthcare organizations

**Challenges:** Integrating across functional areas and technical debt

**Insight:** After struggling with integrating different functional areas, the organization was seeing a lot of delay-related waste. Toward the end of each development life-cycle, they usually discovered a large number of defects (bugs, or what we call “technical debt”). In part, this was because they did not integrate and test until it was too late to do anything except manage the bugs that were found. Their goal was to get the bugs to some sort of acceptable level.

The group’s next release was organized into a Lean Portfolio of prioritized features, which they scheduled using Agile release planning. The organization got huge insight when they discovered that the teams had only focused on the lowest priority features. The functional silos were reorganized into cross-functional teams, and much shorter cycles of integration were achieved. For the first time, the organization was able to do a full regression test and complete the User Acceptance Test cycle prior to releasing the software to their clients.

## Productivity and Quality

The context and insight include the following:

**Group profile:** Administrative Software for large enterprise storage array networks

**Challenges:** Seeking productivity and quality gains in releases that must deliver features into a complex matrix of different operating systems, databases, and hardware environments

**Insight:** After reorganizing into cross-functional teams that pulled work from a well-managed product portfolio, the teams began to focus on incrementally completing features in the order defined by the user’s view of the system (administrator). The product team could easily validate specific flows for their clients and give the team quick feedback. With ten-day iterations, the team completed its release commitments three months ahead of the regular time allotment for releases of this size, and with a third of the defects.

## Cross-functional Teams

The context and insight include the following:

**Group profile:** Software for healthcare data and intelligence

**Challenges:** Growing business made it difficult to scale

**Insight:** Increasingly larger releases were creating a disproportionate number of defects. The development teams had grown significantly beyond a core group of developers and yet deep knowledge of their code base had not kept pace. The individual developers were very comfortable in their silos, but the increasingly complex system was becoming unwieldy. Big-batch thinking seemed to be as basic as breathing air.

We selected a Minimum Releasable Feature and then defined the stories and tasks required to accomplish that one feature. The team rallied around this concept. They aggressively distributed

their knowledge and discovered blind spots among the silos that were contributing to the higher defect count.

## **The Mantra of Lean: Fast-Flexible-Flow**

Before concluding with next steps, we want to offer one more perspective on Lean thinking, using Womack and Jones's characterization of Lean as "Fast, Flexible, Flow" (Womack and Jones 1996).

Think of the software development value stream as a pipeline. At the beginning of the pipeline, there is a variety of items to work on. We have to select the right items to work on in order to provide us the greatest value. In order to have good flow through the pipeline, we need them to be the right size and the right number. Our goal is to reach maximum flow by maximizing our efficiency in the two main tasks: discovering how to build work items and then building them.

To accomplish both of these activities we need to optimize the entire value stream. The focus is not on productivity at each step, but rather the time for ideas to go from beginning to end, from initial concept to consumption. This includes

- **Short queues** Imagine a bank where, from the outside, you see lots of people going in and out each hour. This looks good, but if you see huge lines inside, the cycle time (waiting time) is long, even though the throughput (people exiting over time) is high. Items should have cycle times that are as short as possible.
- **Eliminating waste** This is accomplished by building things in the right sequence and with a just-in-time approach. When things go wrong, we look at our systems and get to the root cause of the errors.

While this description is simple, in real life it is usually hard. Often, it requires structural change with both top-down guidance (leadership) and bottom-up implementation. The focus is on continuously refining the system while working on delivering value faster, better, with less waste . . . and at lower cost.

To do this, *we must rely on utilization theory* to guide our decisions aimed at improving the process.

## **An Example of Fast-Flexible-Flow**

Here is an example of how proper planning and a focus on fast, flexible flow helped a small development organization become effective again. As you read through this story, you can probably remember a time when you went through this yourself. Or maybe you are still there!

### **Let's try specialization**

This organization starts out with one stakeholder and one team of 12 people. The stakeholder asks the team to do a month's work (240 person-days), with a schedule something like the one shown in Figure 14.3.

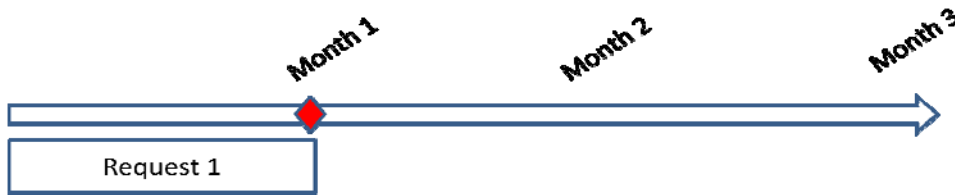


Figure 14-3. One stakeholder, one request, one month

The organization grows to include three stakeholders but is still supported by the same 12-person team. Each stakeholder asks for one month (again, 240 person-days) of work. The team works on these simultaneously. At best, they might get by with a three month schedule like the one in Figure 14.4.

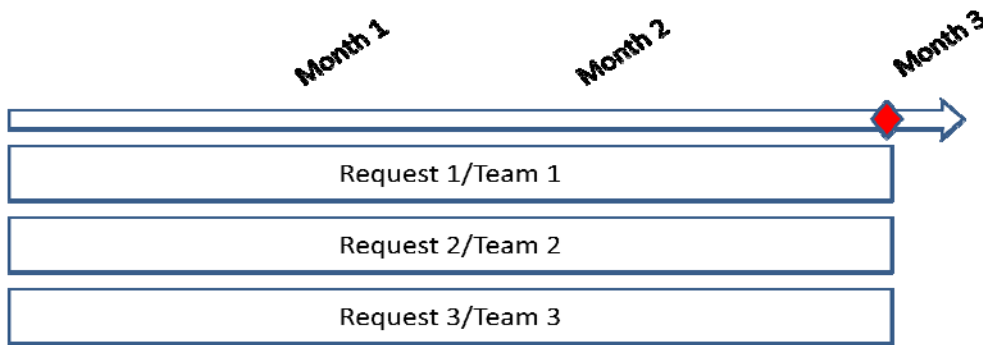


Figure 14-4. Three stakeholders three requests, three months

The original stakeholder wonders why something that used to take one month now takes three! This causes a lot of problems. He now has to forecast his needs three months in advance instead of only one.

The team tries to gain productivity by creating specialized subteams: UI, mid-tier, dataflow, and enterprise data. This helps a little: Figure 14.5 shows the work for each stakeholder by subteam and Figure 14.6 shows what happens when they schedule the work by subteam. They shorten the flow by a few weeks.

The following is an excerpt from Lean-Agile Software Development: Achieving Enterprise Agility by Shalloway, Beaver and Trott  
 No portions may be reproduced without the express permission of Net Objectives, Inc.

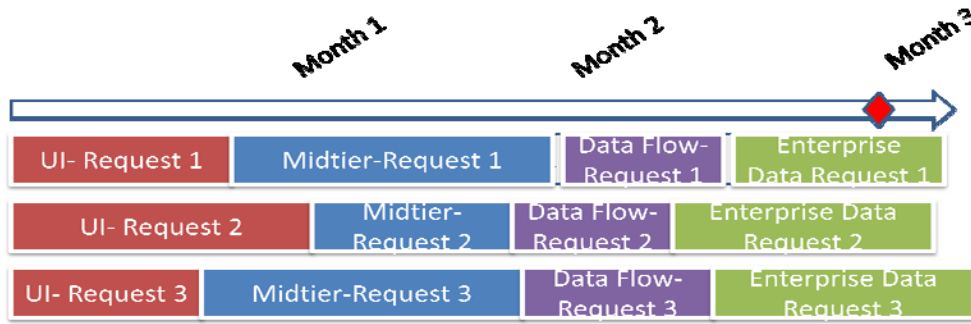


Figure 14-5. Specialized team working on three requirements simultaneously

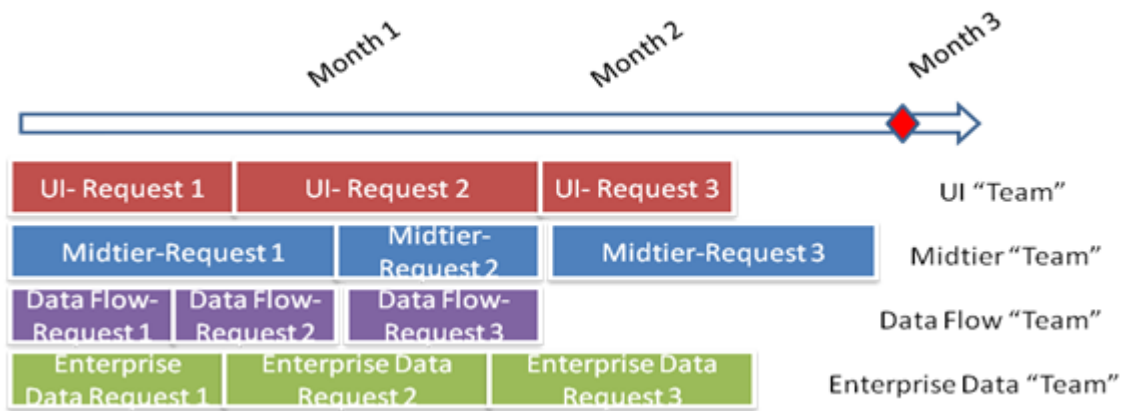


Figure 14-6. Specialized team working on three requirements in sequence

Unfortunately, it is not that easy. They find that subteams cannot simply do their work in sequence. There are interactions. Now, they have to plan ahead and figure out how to integrate the results. So the result looks like Figure 14.7. It takes even longer!

Hmm. Maybe specialization isn't the right approach after all.

The following is an excerpt from Lean-Agile Software Development: Achieving Enterprise Agility by Shalloway, Beaver and Trott. No portions may be reproduced without the express permission of Net Objectives, Inc.

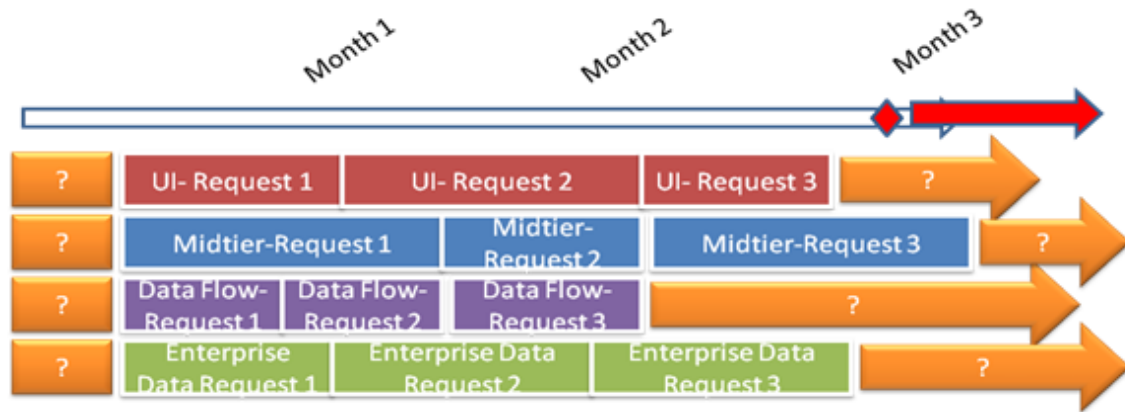


Figure 14-7. Now, extra planning and integration that is needed

### Let's try Lean thinking

Lean thinking says to do the following:

- Minimize cycle time.
- Do things Just-in-Time.
- Complete one task before going on to the next task.

This can best be accomplished by addressing one request at a time, with a team swarm, as we did in the beginning. This is illustrated in Figure 14.8.

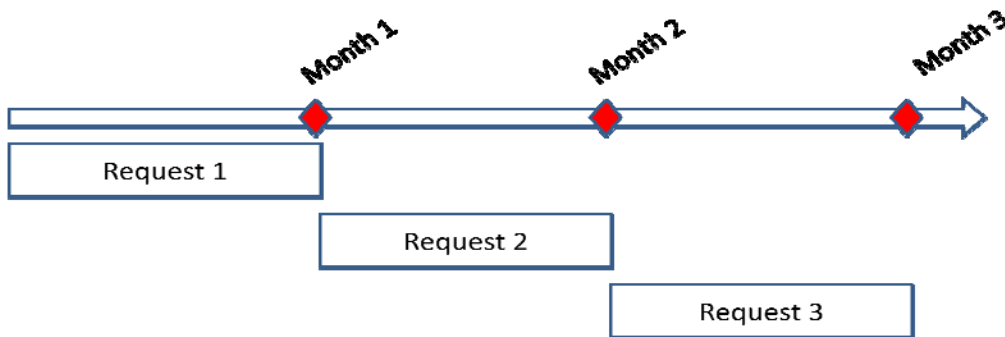


Figure 14-8. One team doing multiple jobs, each in sequence

This is clearly better than what we had before. In Figure 14.7, the average cycle time is three-plus months whereas in Figure 14.8, the average cycle time is only two months (that is, Request 1 took one month, Request 2 took two months, and Request 3 took three months). This shows good improvement.<sup>3</sup>

<sup>3</sup> Note that the benefit actually scales. For example, if we tried to work on ten concurrent requests, the average cycle time would increase to ten months, but if we worked them in sequence, the average cycle time would be only five months.

However, as you can easily imagine, the stakeholder of Request 3 is not very happy. What happens if things don't go so well and the first two requests take all of the time available? Her request will never happen!

Stakeholders are nothing if not creative at getting what they want. When the three stakeholders get together to talk about the plan, they decide to compromise. Each stakeholder agrees to build the smallest functionality required to satisfy immediate needs and then work on more. Of course, this is exactly what should be done: When Product Champions start talking to each other about the smallest functionality that is actually needed, it is more likely that the most valuable features for the enterprise will be built.

This seems like a simple example. We have seen the application of these principles work again and again to help organizations become highly effective.

## Next Steps

How do you learn more? There are many ways available. First, you have, we hope, learned a few principles here that you can see from your own experience are true. Start applying these when you have challenges. Understanding principles by way of your own experience can be powerful. They should help you when you need to solve a problem that you haven't solved before. If you are currently using an Agile process, but not using Lean, we suggest doing a value-stream map of your entire process—from customer request to deployment. Identify where your problems are. Don't assume that creating an Agile team makes you an Agile enterprise. You may need to start out with your product-management team. Notice where your problems are coming from—don't just jump in with the popular Agile methodology of the moment.

There are many ways to gain more knowledge as well—user groups, books, and web sites. We have put together a portal of Lean-Agile information at our own web site for the book:

[www.netobjectives.com/lasd](http://www.netobjectives.com/lasd)

We've included a few examples of those resources here, but please refer to the site, as things change quickly in this industry.

## User Groups of Interest

- **Lean-Agile User Group.** This list is moderated by Alan Shalloway and lurked on by many Lean dignitaries. Discussions regarding any aspects of Lean and/or Agile methods are welcome. <http://tech.groups.yahoo.com/group/leanagile>.
- **Lean Development.** This list focuses on Lean Software Development and is moderated by the Poppendiecks. This list is more focused on Lean Software Development, but it is not that dissimilar from the Lean-Agile list. <http://tech.groups.yahoo.com/group/leandevlopment>

- **Kanban Dev.** This list focuses on Kanban Software Development, moderated by David Anderson. Many Kanban practitioners lurk here as do many others just trying to understand Kanban. <http://finance.groups.yahoo.com/group/kanbandev>

## Books to Read

Table 14.1 lists some essential books you should read, based on your role in the organization. Also, check the resources section of the Net Objectives website (following, in Other Resources) for more up-to-date sources.

If you are...	These are essential books...
<b>Mid-level Manager or above</b>	<p><b>Lean Thinking: Banish Waste and Create Wealth in Your Corporation.</b> (Womack and Jones, 2003)</p> <p><b>Implementing Lean Software Development: From Concept to Cash.</b> (Poppendieck and Poppendieck, 2006)</p> <p><b>Product Development for the Lean Enterprise: Why Toyota's System Is Four Times More Productive and How You Can Implement It.</b> (Kennedy, 2003)</p> <p><b>Ready, Set, Dominate: Implement Toyota's Set-based Learning for Developing Products and Nobody Can Catch You.</b> (Kennedy, Harmon, and Minnock, 2008)</p>
<b>Team Director or Lead</b>	<p><b>Lean Thinking: Banish Waste and Create Wealth in Your Corporation.</b> (Womack and Jones, 2003)</p> <p><b>Managing the Design Factory.</b> (Reinertsen, 1997)</p> <p><b>Implementing Lean Software Development: From Concept to Cash</b> (Poppendieck and Poppendieck, 2006)</p>
<b>Product Manager</b>	<p><b>Software by Numbers: Low-Risk, High-Return Development,</b> (Denne and Cleland-Huang, 2003)</p> <p><b>Managing the Design Factory.</b> (Reinertsen, 1997)</p>
<b>Team Lead, or Interested in Kanban</b>	<p><b>Scrumban Essays on Kanban Systems for Lean Software Development.</b> (Ladas, 2009)</p> <p><b>Kanban - Successful Change Management for Technology Organizations.</b> (Anderson, Forthcoming 2010)</p>
<b>Interested in Lean Knowledge Stewardship</b>	<p><b>Learning to Fly: Practical knowledge management from some of the world's leading learning organizations.</b> (Collison and Parcell, 2004)</p> <p><b>Managing to Learn: Using the A3 Management Process to Solve Problems, Gain Agreement, Mentor, and Lead.</b> (Shook, 2008)</p>
<b>Interested in Lean Science</b>	<p><b>Managing the Design Factory.</b> (Reinertsen, 1997)</p> <p><b>Lean Thinking: Banish Waste and Create Wealth in Your Corporation.</b> (Womack and Jones, 2003)</p> <p><b>The Principles of Product Development Flow: Second</b></p>

	<b>Generation Lean Product Development.</b> (Reinertsen, 2009)
<b>Interested in Lean Management</b>	<b>Creating a Lean Culture: Tools to Sustain Lean Conversions.</b> (Mann, 2005) <b>The Leader's Handbook: Making Things Happen, Getting Things Done.</b> (Scholtes, 1997)
<b>Involved in Transitioning Teams</b>	<b>Managing Transitions: Making the Most of Change.</b> (Bridges, 2003)

## Other Resources

All three authors of this book work for Net Objectives. At Net Objectives, we provide services including assessments, consulting, and training. We have had many experiences with many different companies. We help businesses realize value faster from their software development investment. Our typical clients are trying to extend their agile endeavors from the team to include management concerns and business priorities. Our range includes Lean, Agile, Kanban, Scrum, Acceptance Test Driven Development, Test Driven Development, Design Patterns and more. All of our trainers and consultants are seasoned practitioners, authors and thought leaders.

In our attempts to help our clients, we have collected and maintained a set of resources that span virtually all Lean-Agile disciplines and provide information and value for all roles that are involved. You can get access to this information by going to [www.netobjectives.com/resources](http://www.netobjectives.com/resources). This site offers days of free online training in the form of recorded webinar sessions, as well as dozens of articles to read.

## Summary

Lean-Agile software development is about building software faster, better, and with less waste and cost than ever before. It requires a different mindset than many of us have had. Lean principles are based on a combination of science, management and, knowledge stewardship. Our journey doesn't end with Lean. It is an ongoing process. Our goal of continuous process improvement means always being in transition to better methods.

Lean-Agile thinking is grounded in the day-to-day realities of creating software. It is a way of thinking more than a fixed prescription to follow. Using its principles, you can apply Lean-Agile to address your own local conditions. Over time, you will realize the benefits— building the most important items as quickly as you can with high quality, getting products to the market quicker, adding value for the customer, and lowering costs.

## 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.

*The following is an excerpt from Lean-Agile Software Development: Achieving Enterprise Agility by Shalloway, Beaver and Trott  
No portions may be reproduced without the express permission of Net Objectives, Inc.*

- As a manager:
  - Identify some delays that affect your work. Why are they there? What purpose do they serve? Would eliminating them add any value?
- As a developer:
  - Identify some delays in your development process that you can eliminate without asking management's permission. Would doing so help your process?
  - What principles of Lean thinking can you apply throughout your organization?
  - Do you see any counter-productive delays that you could eliminate or that someone else could eliminate by changing something?
  - What are some examples of waste in your organization that you could decrease?

## NET OBJECTIVES LEAN-AGILE APPROACH

### INTEGRATED AND COHESIVE

All of our trainers, consultants, and coaches follow a consistent Lean-Agile approach to sustainable product development. By providing services at all of these levels, we provide you teams and management with a consistent message.

### PROCESS EXECUTION

Net Objectives helps you initiate Agile adoption across teams and management with process training and follow-on coaching to accelerate and ease the transition to Lean-Agile practices.

### SKILLS & COMPETENCIES

Both technical and process skills and competencies are essential for effective Agile software development. Net Objectives provides your teams with the knowledge and understanding required to build the right functionality in the right way to provide the greatest value and build a sustainable development environment.

### ENTERPRISE STRATEGIES

Enterprise Agility requires a perspective of software development that embraces Lean principles as well as Agile methodologies. Our experienced consultants can help you develop a realistic strategy to leverage the benefits of Agile development within your organization.



Contact Us:  
sales@netobjectives.com  
1-888-LEAN-244  
(1-888-532-6244)

*We deliver unique solutions  
that lead to tangible improvements in software development  
for your business, organization and teams.*

## SERVICES OVERVIEW

### TRAINING FOR AGILE DEVELOPERS AND MANAGERS

Net Objectives provides essential Lean-Agile technical and process training to organizations, teams and individuals through in-house course delivery worldwide and public course offerings across the US.

### CURRICULA — CUSTOM COURSES AND PROGRAMS

Our Lean-Agile Core Curriculum provides the foundation for Agile Teams to succeed.

Lean Software Development

- Implementing Scrum for Your Team
- Agile Enterprise Release Planning
- Sustainable Test-Driven Development
- Agile Estimation with User Stories
- Design Patterns

In addition, we offer the most comprehensive technical and process training for Agile professionals in the industry as well as our own Certifications for Scrum Master and Product Champion.

### PROCESS AND TECHNICAL TEAM COACHING

Our coaches facilitate your teams with their experience and wisdom by providing guidance, direction and motivation to quickly put their newly acquired competencies to work. Coaching ensures immediate knowledge transfer while working on your problem domain.

### LEAN-AGILE ASSESSMENTS

Understand what Agility means to your organization and how best to implement your initiative by utilizing our Assessment services that include value mapping, strategic planning and execution. Our consultants will define an actionable plan that best fits your needs.

### LEAN-AGILE CONSULTING

Seasoned Lean-Agile consultants provide you with an outside view to see what structural and cultural changes need to be made in order to create an organization that fosters effective Agile development that best serves your business and deliver value to your customers.

### FREE INFORMATION

#### CONTACT US FOR A FREE CONSULTATION

Receive a free no-obligation consultation to discuss your needs, requirements and objectives. Learn about our courses, curricula, coaching and consulting services. We will arrange a free consultation with instructors or consultants most qualified to answer all your questions.

Call toll free at 1-888-LEAN-244 (1-888-532-6244) or email [sales@netobjectives.com](mailto:sales@netobjectives.com)

### REGISTER PROFESSIONAL LEAN-AGILE RESOURCES

Visit our website and register for access to professional Lean-Agile resources for management and developers. Enjoy access to webinars, podcasts, blogs, whitepapers, articles and more to help you become more Agile. Register at <http://www.netobjectives.com/user/register>.