Introduction to Disciplined Agile Delivery

Second Edition

A Small Agile Team's Journey From Scrum to Disciplined DevOps

Mark Lines and Scott W. Ambler

v1.1

Contents

Acknowledgments

We'd like to thank Beverley Ambler, Rod Bray, David Dame, Matt Everard, Louise Lines, Glen Little, Valentin-Tudor Mocanu, Kristen Morton, David Shapiro, Paul Sims, and Michael Vizdos for their feedback and input into the development of this book. We couldn't have done it without you.

Chapter 1

Introduction

Many organizations are struggling to be successful with mainstream agile methods such as Scrum. Sometimes, the impulse is to give up and try the *next great thing* such as Lean or Scaled Agile Framework (SAFe®). The reality is that the source of failure of existing agile adoptions can often be traced to either the misapplication of core agile principles or a naïve approach to scaling agile and the need to address enterprise concerns.

Adding to the confusion is the constant bickering in the agile community about which method is best between Scrum, Extreme Programming (XP), Kanban, SAFe, and others. The reality is that most organizations can benefit from a number of these strategies, albeit with some consistency that a common tool kit can provide. This is where the Disciplined Agile™ (DA) tool kit comes in.

DA is a hybrid of existing methods that provides the flexibility to use various approaches, as well as plugging some gaps not addressed by mainstream agile methods. In a nutshell, DA, and more importantly, the Disciplined Agile Delivery (DAD) portion of it that is the focus of this book, is "pragmatic agile." It describes

proven strategies to adapt and scale agile initiatives that suit the unique realities of your enterprise, without having to figure it all out by yourself.

The book *Choose Your WoW! A Disciplined Agile Delivery Handbook for Optimizing Your Way of Working* is the definitive guide for DAD. We have key information about DA, in general, at PMI.org/disciplined-agile.

While *Choose Your Wow* is a great overview and reference guide, it is over 400 pages. We felt that it would be useful to summarize DAD into a quick read to show how it can be applied in a typical situation. That thinking led to the first edition of *Introduction to Disciplined Agile Delivery*, published in the third quarter of 2015. We have found that organizations that simply take a bit of time to understand what DAD is, as well as what it is not, see very quickly the obvious benefits of DAD.

We wrote this short book in the hope that after having invested just an hour or two to learn about what DAD actually is, you are convinced that it is worthwhile to further investigate the possible fit for DAD within your organization.

Some quick facts about DAD:

- DAD is the delivery portion of the Disciplined Agile (DA) tool kit. DAD is a key component of Disciplined DevOps, which in turn is part of DA's value stream layer, which is part of the Disciplined Agile Enterprise (DAE). The book *An Executive's Guide*

to the Disciplined Agile Tool Kit provides an overview of DA, as does the appendix at the back of the book.

- DA is resonating within organizations around the world. Organizations that have adopted, or are in the process of adopting, DA include large financial institutions, software companies, e-commerce companies, restaurant chains, government agencies, and many others. By adoption, we mean either planning or actively implementing it across their entire organization, not just with one or two teams.
- Although DAD was originally developed at IBM, it is now the intellectual property of Project Management Institute (PMI), and is available at no cost to PMI members.
- The certification program is described at PMI.org.
- DA is not a replacement for existing agile and lean methods. It complements them and, in many cases, extends them to be enterprise ready.

Chapter 2

Reality Over Rhetoric

One of the reasons why DAD is quickly growing in popularity is that it acknowledges how organizations that are effectively scaling agile are *really* doing things. We think that it is important to clear up some of the misconceptions regarding the hype of agile purism versus what we have found to be really happening based on both our hands-on experience with many clients around the world and our comprehensive industry research.[1]

Myth	Reality
Agile teams don't do requirements or planning.	The average agile team spends about one month doing some up-front planning and requirements modeling. While DAD seeks to minimize this work, we acknowledge this reality and suggest that teams new to agile spend a few weeks in an Inception phase to complete the work in a minimal, yet sufficient, fashion.
DAD is a form of "WaterScrumFall," with lots of up-front planning and requirements (Water), and testing and deployment at the end (Fall), with	While DAD acknowledges that some prework is required to secure funding, among other precoding activities, DAD suggests you minimize the work in Inception. Similarly, while DAD recognizes that the common pattern in enterprises is to deploy solutions in a structured fashion, which we describe as the Transition phase, the disciplined testing practices in the Construction phase, such as continuous integration, should minimize the transition effort and end-of-life-cycle testing. For more advanced applications of DAD, the teams may not even require an explicit Inception or Transition

Scrum in the middle.	phase, as described in DAD's Continuous Delivery: Agile and Continuous Delivery: Lean life cycles.
"Governance" is an agile dirty word. The agile concept of self-organization means that enterprises should not interfere with how agile teams deliver their software.	Governance is actually a good thing when it's done in an agile/lean manner. Sponsors, and the enterprise as a whole, deserve to know that their investments are being properly spent and that the risk of delivery is monitored and controlled, albeit in a lightweight, agile fashion. DAD provides specific guidance to fulfill this responsibility in a relatively painless fashion.
DAD is complicated and would be disruptive to adopt.	DAD is quite simple to adopt, especially if your organization is familiar with common agile practices. The good news is that DAD provides guidance that addresses why some existing agile implementations are struggling, and DAD can help to bring these implementations back on track.
SAFe is *the* solution to scaling agile.	For large organizations that have delivery teams greater than 100 people, SAFe may indeed be a good fit for *some* teams. However, most organizations have a mix of small- to mediumsized teams delivering solutions for multiple lines of business. In these situations, SAFe may not be suitable. While SAFe is suitable in a specific context, DAD is much more flexible and adaptable to a broader range of situations in the enterprise.

[1] See Ambysoft.com/surveys/.

Chapter 3

Disciplined Agile Delivery in a Nutshell

Key Points

- DAD is the delivery portion of the Disciplined Agile (DA) tool kit, not just another methodology.
- If you are using Scrum, XP, or Kanban, you are already using variations of a subset of DAD.
- DAD provides six life cycles to choose from; it doesn't prescribe a single way of working. Choice is good.
- DAD focuses on achieving common goals in an agile manner, not the production of specific work products or following a prescriptive agile strategy.
- DAD addresses key enterprise concerns not described by mainstream methods such as Scrum.
- DAD is complementary to SAFe, yet far less prescriptive and more practical for most enterprises.
- DAD shows how agile works from beginning to end.
- DAD provides a flexible foundation from which to scale mainstream methods.

- While DAD's philosophy is consistent with that of the *Manifesto for Agile Software Development* (Agile Manifesto), it includes additional guidance to be effective in more complex enterprise situations.
- It is *not* difficult to get started with DAD.

Many organizations start their agile journey by adopting Scrum because it describes a good strategy for leading agile software teams. However, Scrum is only a small part of what is required to deliver sophisticated solutions to your stakeholders. Invariably, teams need to look to other methods to fill in the process gaps. When looking at other methods, there is considerable overlap and conflicting terminology that can be confusing to practitioners, as well as outside stakeholders. Worse yet, people don't always know where to look for advice, or even know what issues they need to consider.

To address these challenges, Disciplined Agile Delivery (DAD) is a people-first, learning-oriented, hybrid agile approach to IT solution delivery. Delivery (DAD) provides a more cohesive approach to agile solution delivery. It supports several risk value delivery life cycles, is goal driven, enterprise aware, scalable, and reflects the realities of enterprise solution delivery. DAD can—and should—be used to extend Scrum to address the critical aspects of software development that Scrum purposefully leaves up to you.

There are clearly some interesting aspects to DAD:

1. **Hybrid**. DAD is an agnostic, hybrid approach that puts proven strategies from Scrum, Agile Modeling (AM), Extreme

Programming (XP), Unified Process (UP), Kanban, Lean Software Development, SAFe, and several other methods into context.

2. **Full-delivery life cycle**. DAD extends the construction-focused life cycle of Scrum to address the full beginning-to-end delivery life cycle, from team initiation all the way to delivering the solution to its end users.
3. **Support for multiple life cycles**. DAD also supports lean, continuous delivery, and exploratory versions of the life cycle. Unlike other agile methods, DAD doesn't prescribe a single life cycle because it recognizes that one process size does not fit all.
4. **Complete**. DAD includes advice about how development, modeling, documentation, and governance strategies fit together in a streamlined whole. We like to say that DAD does the "process heavy lifting" that other methods leave up to you.
5. **Context sensitive**. Instead of the prescriptive approach seen in other agile methods, including Scrum, DAD promotes what we call a goal-driven or objective-driven approach. In doing so, DAD provides contextual advice regarding viable alternatives and their trade-offs, enabling you to tailor DAD to effectively address the situation in which you find yourself. By describing what works, what doesn't work, and more importantly, why, DAD helps you to increase your chance of adopting strategies that will work for you.

People First: Roles in Disciplined Agile Delivery

As shown in Figure 3.1, DAD suggests a robust set of roles for agile solution delivery. A common question that we get is what is the difference between primary and supporting roles? Primary roles exist in all DAD teams regardless of scale. Supporting roles, however, typically occur only at scale and sometimes only for a temporary period of time. Another common question that we get is: "Why are there so many roles?" Scrum has three roles—scrum master, product owner, and team member—so why does DAD need 10? The key issue is one of scope. Scrum mostly focuses on leadership and change management aspects during Construction, and therefore has roles that reflect this. DAD, on the other hand, explicitly focuses on the entire delivery life cycle and all aspects of solution delivery, including the technical aspects that Scrum leaves out. So, with a larger scope comes more roles. For example, DAD encompasses agile architecture issues so it includes an architecture owner role. Scrum doesn't address architecture, so it doesn't include this role.

An Agnostic, Hybrid Tool Kit

Disciplined Agile (DA) is an agnostic, hybrid tool kit that builds upon the solid foundation of other methods and software process frameworks. The DAD portion of DA adopts practices and strategies from existing sources and provides agnostic advice for when and how to apply them together. In one sense, methods such as Scrum, Extreme Programming (XP), Kanban, and Agile Modeling (AM) provide the process bricks, and DAD provides the mortar to fit the bricks together effectively, as depicted in Figure 3.2.

One of the great advantages of agile and lean software development is the wealth of practices, techniques, and strategies available to you. This is also one of its greatest challenges, because without something like the DAD, it's difficult to know which practices to choose and how to fit them together. Worse yet, many teams new to agile will treat a method like Scrum or SAFe as if it's a recipe, ignoring advice from other sources and thereby getting into trouble.

Choice Is Good: Full-Delivery Life Cycles

The focus of DAD is on delivery, although remember that other aspects of IT and your organization still exist, such as data management, enterprise architecture, operations, portfolio management, and more. A full-system/product life cycle goes from the initial concept for the product, through delivery, to operations and support, and often includes many iterations of the delivery life cycle. Figure 3.3 depicts a high-level view of the DAD life cycle. The inner three phases—Inception, Construction, and Transition—form the delivery portion of the life cycle. During delivery, you incrementally build an increasingly more consumable solution.

Obviously, there's more to DAD than what this high-level diagram shows. DAD, because it's not prescriptive and strives to reflect reality as best it can, supports several versions of a delivery life cycle. Six versions of the life cycle follow: the Agile life cycle that

extends the Scrum Construction life cycle with proven ideas from Unified Process to support early mitigation of risk and lightweight governance; the Lean life cycle based on Kanban; the Continuous Delivery: Agile life cycle; the Continuous Delivery: Lean life cycle; the Exploratory life cycle based upon a Lean Startup approach; and the Program life cycle for a team of teams. DAD teams will adopt the life cycle that is most appropriate for their situation and then tailor it appropriately. The visual depictions of the life cycles are shown first, then descriptions follow.

Explicit Phases Make Agile More Palatable to Management

As shown in Figure 3.4, DAD life cycles can have phases. Daniel Gagnon has been at the forefront of agile practice and delivery for almost a decade in two of Canada's largest financial institutions. He had this to say about using DA as an overarching framework: “At both large financials that I have worked in, I set out to demonstrate the pragmatic advantages of using DA as a ‘top-of-the-house’ approach. Process tailoring in large, complex organizations clearly reveals the need for a large number of context-specific implementations of the four (now six) life cycles, and DA allows for a spectrum of possibilities that no other framework accommodates. However, I call this ‘structured freedom,’ as all choices are still governed by DA's application of Inception, Construction, and Transition

with lightweight, risk-based milestones. These phases are familiar to project management offices (PMOs), which means that we aren't carrying out a frontal assault on their fortified position, but rather introducing governance change in a lean, iterative, and incremental fashion."

Consumable Solutions Over Working Software

The Agile Manifesto suggests that we measure progress based upon "working software." But what good is that when the customer can't—or worse—doesn't want to use it? What we deliver should be consumable, which is *functional + usable + desirable*. Additionally, what we produce usually is not just software. There may be business changes and other supporting deliverables, so we suggest striving to deliver "consumable solutions."

In general, we suggest the following guidance regarding which life cycles fit in certain circumstances:

Agile life cycle. This life cycle, shown in Figure 3.4, is based largely upon Scrum and XP with a set of timeboxed iterations (sprints) being the core of the Construction phase. It is the most commonly used life cycle that is suitable in these types of

situations:

- The work is primarily enhancements or new features.
- The work can be identified, prioritized, and estimated early in the process.
- The team is new to agile.
- The team is familiar with Scrum and XP.
- The team is typically working on a project.

Note that while the diagram shows that you should have increments of a consumable solution at the end of each iteration, for a new product or solution you may not have something truly consumable until after having completed several iterations.

Continuous Delivery: Agile life cycle. This life cycle, shown in Figure 3.5, is a natural progression from the Agile life cycle. Teams typically evolve to this life cycle from the Agile life cycle, often adopting iteration lengths of one week or less. The key difference between this and the Agile life cycle is that the continuous delivery life cycle results in a release of new functionality at the end of each iteration, rather than after a set of iterations. Teams require a mature set of practices around continuous integration and continuous deployment and other DevOps strategies. This life cycle is suitable when:

- Solutions can't be delivered to stakeholders on a frequent and incremental basis.
- Work remains relatively stable within an iteration.
- Organizations have streamlined deployment practices and

procedures.
- There is a critical need to get value into the hands of stakeholders rapidly, before the entire solution is complete.
- Teams have mature DevOps practices in place, including continuous integration, continuous deployment, and automated regression testing.
- The team is long-lived (stable), and is working on a series of releases over time.

Lean life cycle. This life cycle, shown in Figure 3.6, promotes lean principles such as minimizing work in process, maximizing flow, having a continuous stream of work (instead of fixed iterations), and reducing bottlenecks. New work is pulled from the work item pool as the team has capacity.

Note that Scrum prescribes the use of a set of "ceremonies," such as the daily coordination meeting (Scrum), iteration (sprint) planning sessions, and retrospectives to be done on certain cadences within the iterations (sprints). Lean does not prescribe this overhead. In fact, it considers these practices to be a source of waste and instead suggests that they be done only when necessary. This requires a degree of discipline and self-awareness not usually found on teams new to agile. While the concepts of lean and the Kanban system are very easy to learn, it can be difficult to master the principles of lean flow and maximizing system throughput. It is suitable in these situations:

- Work can be broken down into very small work items of roughly the same size.
- Work is difficult to predict in advance. For example, teams that are focused on fixing defects or handling support issues are good candidates for this life cycle.
- The team favors the lean approach of minimizing batch size (which helps to reduce work in process) and reducing or eliminating any planning in advance of doing the work.
- The team is typically working on a series of small changes and then releasing them periodically in a manner such as a monthly release cycle.

Continuous Delivery: Lean life cycle. This life cycle, shown in Figure 3.7, is a natural progression from the Lean life cycle. It supports the goal of delivering increments of the solution in a more frequent manner than the other life cycles. Teams typically evolve into this life cycle from either the Lean life cycle or the Continuous Delivery: Agile life cycle. It requires a mature set of practices around continuous integration and deployment in order to be practical. It also requires the technical infrastructure and advanced DevOps practices that support this approach. It is best suited in these types of situations:

- Solutions can be delivered to stakeholders frequently and incrementally.
- New work, including both new requirements and defect reports, arrives often.
- Organizations have streamlined deployment practices and

procedures.

- It is critical that solutions get value into the hands of stakeholders rapidly, even before the entire solution is complete.
- Teams have mature DevOps practices in place, including continuous integration, continuous deployment, and automated regression testing.
- The team is long-lived (stable), and is working on a series of releases over time.

Exploratory life cycle. This life cycle, shown in Figure 3.8, is based on the Lean Startup principles advocated by Eric Ries, and is extended with ideas from complexity theory to increase its effectiveness. The philosophy is to minimize up-front investments in solutions in favor of small experiments (called minimal viable products, or MVPs) that are market tested and measured early and often. As the solution is being developed, the delivery team has the opportunity to deliver what is truly required based on feedback from actual usage. It is useful in these types of situations:

- The solution addresses high-incertitude cases such as a new, unexplored market or a new product.
- The stakeholders and delivery team are very flexible in adapting the solution as it is being developed.
- You have a valid hypothesis/strategy to test with clear go/no-go criteria for when the test is over.
- You are willing to experiment and evolve your idea based on your learnings.

Q

R

S

T