Disciplined Agile Delivery

The Foundation for Scaling Agile

Scott W. Ambler, Scott Ambler + Associates
Mark Lines, Scott Ambler + Associates

Abstract. Many organizations have adopted agile strategies to some extent, often applying simple methods such as Scrum on a few software development projects. Although they have succeeded on a handful of projects—clearly a good start—they now want to scale agile to address enterprise-class solutions. As the old saying goes about success, "What got you here is not going to get you to the next level."

This article explores several important issues when it comes to applying agile strategies at scale. First, it explores what it means to scale agile. Our experience is that without this understanding it is incredibly difficult to come to an agreement as to how to tailor agile strategies to meet your needs at scale. Second, we overview the Disciplined Agile Delivery (DAD) process decision framework and show how it provides a solid foundation from which to scale agile. Third, we explore how DAD's (process) goal-driven approach is the key for scaling agile solution delivery. Fourth, we warn you of the dangers of assuming that you just need to adopt a few new practices. Fifth, we review some survey-based evidence that summarizes the experiences of the industry when it comes to applying agile at scale.

Context Counts

There is more to scaling than the size of the team. Figure 1 depicts what we have found to be the six key factors when scaling agile—team size, geographic distribution, organizational distribution, regulatory compliance, domain complexity, and technical complexity—all of which are ranges. These six factors, a simplification of the Agile Scaling Model (ASM) [1] that Scott W. Ambler led the development of within IBM, form the scaling portion of what we call the Software Development Context Framework (SDCF). The process selection portion of the SDCF is comprised of four factors (team skills, team culture, organizational culture, and problem type) that are also ranges. The process selection factors are similar to the environment risk factors first described by Boehm and Turner [2] and the combination of the process selection and scaling factors are similar to Kruchten's situational agility octopus [3]. The fundamental point is that context counts – one strategy does not fit all.

Taken together, the process selection factors and the scaling factors drive the way that a team will tailor its organization structure, its process, and its work environment. Each team will find itself in a unique situation and will need to tailor their strategy accordingly, a potential challenge if your organization still clings to a "repeatable process" philosophy. For example a team of seven co-located people in a regulatory environment will work differently than a team of 40 people spread out across several locations in a non-regulatory environment.

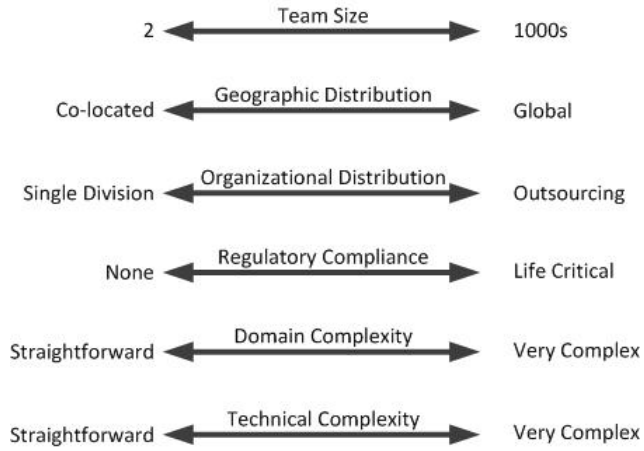

Figure 1. An overview of process scaling factors.

There are three challenges with a context-driven approach. First, few organizations want their solution delivery teams creating their own processes on the fly as this is very inefficient both at the team level and the organization level. Clearly there is need for a common starting point. Second, contrary to what a team may believe, they are unlikely to have the agile expertise required to tailor a strategy that works well for them. In this case there is need for some context-driven guidance. Third, many organizations, particularly in the public and IT service provider sectors, must work in a manner that is CMMI® compliant. When the team and organizational cultures are sufficiently flexible this is definitely possible although it does require some out-of-the-box thinking for everyone involved [4]. To address these challenges, and more, you need to adopt a disciplined approach to agile solution delivery.

Disciplined Agile Delivery (DAD)

From the summer of 2006 to the summer of 2012 Scott W. Ambler worked in the role of Chief Methodologist for IT at IBM Rational, working with organizations around the world to apply agile and lean techniques at scale. The DAD process decision framework was developed as the result of the observation that there were common patterns for applying agility at scale successfully. The DAD framework reflects the experiences of IBMers working in the field with customer organizations, IBMers applying agile at scale internally, and business partners such as Mark Lines who were also working with customer organizations.

DAD is a second-generation framework that strives to provide a coherent, end-to-end strategy for how agile solution delivery works in practice. DAD is a people-first, learning-oriented hybrid agile approach to IT solution delivery. It has a risk-value lifecycle, is goal-driven, is scalable, and is enterprise aware. Although all of these characteristics are important, several of them are critical for scaling agile:

1. Hybrid. DAD is a hybrid in that it adopts and tailors proven strategies from methods such as Scrum, Extreme Programming (XP), Agile Modeling (AM), Unified Process (UP), Kanban, Outside In Development (OID), and Agile Data (AD) to name a few. Instead of starting with a process kernel such as

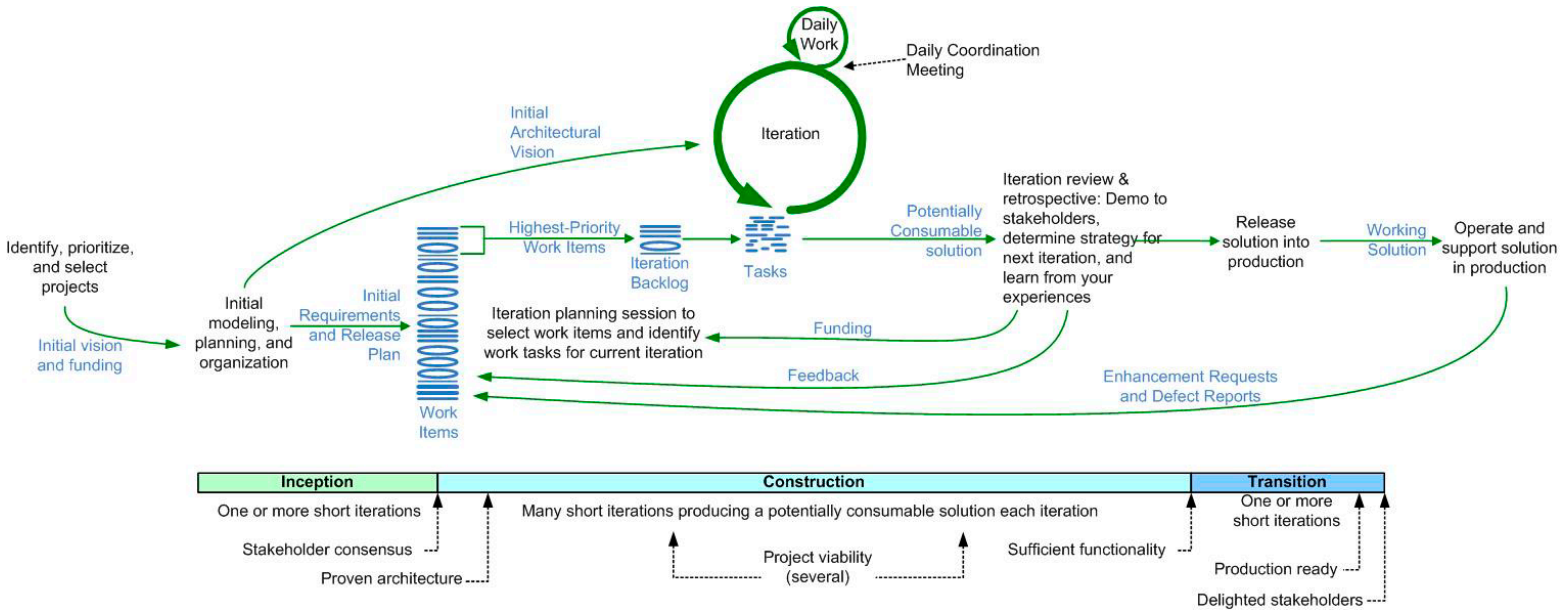

Figure 2: The basic lifecycle for DAD.

Scrum and then adding in a large number of other practices and techniques, or starting with a comprehensive process framework such as the UP and tailor it down to something usable, why not simply start in the middle which is where you actually need to get to anyway? The upshot is that with DAD much of the expensive and time-consuming work of combining agile techniques has already been done for you.

2. Enterprise aware. When you are working at scale you cannot afford to have a delivery team take a stovepipe mentality and work on their own, even when they are producing "potentially shippable software" on a regular basis. The reality is that agile delivery teams do not work in a vacuum. There are often existing systems currently in production, and minimally your solution should not impact them although hopefully your solution will leverage existing functionality and data available in production. There are often other teams working in parallel to your team, and you may wish to take advantage of a portion of what they are doing and vice versa. There may be a common vision that your organization is working towards, or that your program is working towards, a vision that your team should contribute to. There will be a governance strategy in place, hopefully an agile/lean one that enhances what your team is doing—in fact, when we developed DAD we invested significant effort describing how to effectively measure and govern agile teams [5]. Disciplined agile teams recognize that they are part of a larger, organizational ecosystem and act accordingly.

3. Solution focused. With DAD you mature your focus from just producing software to instead providing consumable solutions that provide real business value to your stakeholders within the appropriate economic, cultural, and technical constraints. Yes, software is clearly important, but in addressing the needs of our stakeholders we will often provide new or upgraded hardware, change the business/operational processes that stakeholders follow, and even help change the organizational structure in which our stakeholders work. When you are working at scale you cannot afford to fall into the "potentially shippable software" trap.

4. Delivery focused. The basic DAD lifecycle, depicted in Figure 2, addresses the project lifecycle from the point of initiating a project through construction to the point of releasing the solution into production (it also shows some pre-initiation portfolio management activities as well as post-delivery production activities). This differs from first generation agile methods that typically focus on the construction aspects of the lifecycle, leaving the details about how to perform the rest of it up to you. In fact, you can see how the construction portion of the basic DAD lifecycle reflects an improved version of the Scrum lifecycle. Lightweight milestone reviews, one aspect of DAD's support for agile governance, are depicted along the bottom of the lifecycle. More importantly, because DAD is not prescriptive, the lifecycle of Figure 2 is only one of several supported by DAD. There are also lean/advanced and continuous delivery versions of the lifecycle that abandon many of the constraints prescribed by Scrum.

5. Goal-driven. Every team finds itself in a unique situation, often facing the risks associated with one or more of the scaling factors discussed earlier. To address this challenge the DAD process framework takes a process goal-driven approach where a team tailors their strategy to reflect the context of the situation they find themselves in. Instead of saying that you should organize your work as a stack prioritized by business value (what Scrum prescribes) DAD instead says that you need to manage changing stakeholder needs in some way. DAD also describes how you have several ways of addressing this goal (the Scrum product backlog being one of them) and that there are advantages and disadvantages to each. A goals-driven approach provides disciplined agile teams with the guidance they require to tailor their approach to appropriately address the scaling factors that they face. More on this later.

Goal Driven

Very likely the most intriguing aspect of the DAD process decision framework is the fact that it is goal driven instead of prescriptive. Figure 3 summarizes the DAD process goals from a lifecycle point of view. These goals are applicable regardless

of the situation faced by a team, but the way that the goals are addressed may vary substantially.

Let us do a deep dive into a single goal. Figure 4 presents a goal diagram for identifying an initial technical strategy, something that you do fairly early in a project. The goal is depicted as a rounded rectangle, issues that you should consider as normal rectangles, and potential strategies to address an issue as a list. Some strategy lists are prioritized in order of agile preference, something that is indicated with an arrow to the left of the list (as you see in Figure 4 with Level of Detail strategies). The DAD framework also recommends starting points; these strategies are shown in bold and italics, so as to provide guidance to teams new to agile.

DAD's goal-driven approach underlies the idea that to be effective at applying agile a team must understand the context in which they are working. For example, a team that is co-located has the option of capturing their technical strategy using a few whiteboard sketches via informal modeling sessions. However, a team that is globally distributed would likely need to take a detailed interface approach, a strategy called API First in the Eclipse Way and Contract Modeling in Agile Modeling, to overcome the communication challenges surrounding geographic distribution. A team working in a rigid organizational culture or in a regulatory environment may decide on formal modeling sessions over informal ones. A team facing significant technical complexity may want to consider several candidate architectures to increase their chance of success.

The point is that different teams face different situations; therefore they will need to adopt their strategy to reflect the situation. Each team needs to identify an initial technical strategy, explore their initial scope, develop an initial plan, and fulfill many other goals but they will achieve these goals in different ways. The DAD process framework provides straightforward guidance to help you to make these tailoring decisions effectively. It does this by explicitly describing the process decision that you are making and then walks you through the process of making it. It does this in a two-fold manner, first by overviewing common options available to you in a visual manner via goal diagrams and second by describing the advantages and disadvantages of each option via a table-based approach.

More than Just a Few Extra Practices

Yes, you are going to add some practices over and above what a small, co-located agile team typically does. For example, although architecture is always important, it is critical at scale, thus DAD adopts architecture-oriented practices from several sources. From AM there are practices as initial architecture envisioning and just-in-time model storming throughout the project, from the UP there is proving the architecture early in the lifecycle with working code, and from XP architecture spikes, to name a few. These agile architecture practices not only help you to get started on the right foot they help you to avoid taking on unnecessary technical debt which otherwise would bog your team down.

You will also find that you need to adopt a more sophisticated approach to testing that goes beyond the whole-team strategy favored by agile-in-the-small teams. This includes parallel independent testing, reviews (both formal and informal), and even end-of-lifecycle testing in some cases. Similarly your requirements strategy will become more sophisticated. It should come as no surprise that user stories written on index cards do not scale well.

Goals for the Inception Phase	Goals for Construction Phase Iterations	Goals for the Transition Phase
- Form initial team - Develop common project vision - Align with enterprise direction - Explore initial scope - Identify initial technical strategy - Develop initial release plan - Form work environment - Secure funding - Identify risks	- Produce a potentially consumable solution - Address changing stakeholder needs - Move closer to deployable release - Improve quality - Prove architecture early	- Ensure the solution is consumable - Deploy the solution
Ongoing Goals		
- Fulfill the project mission - Grow team members - Address risk	- Improve team process and environment - Leverage and enhance existing infrastructure - Coordinate activities	

Figure 3. Goals addressed throughout a DAD project.

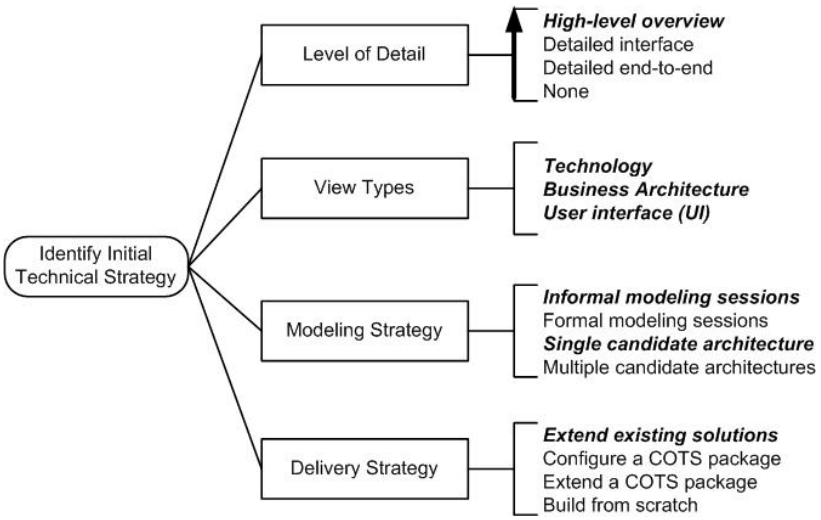

Figure 4. Goal diagram: Identify Initial Technical Strategy

And you will need to go beyond some of the simplistic team organization advice. Scrum's three roles work well in some situations but you will find DAD's approach of five primary roles supported by another five secondary roles will support scaling more effectively. DAD introduces primary roles of Architecture Owner (an agile solution architect) and Stakeholder to supplement the Scrum-like roles of Team Lead (ScrumMaster), Product Owner, and Team Member. The secondary roles of Domain Expert, Independent Tester, Integrator, Technical Expert, and Specialist address the complex environment faced by teams working at scale. Furthermore, this more robust set of roles makes it easier to transition to agile because it is clearer how existing experienced staff can still add value on their team. Coordination on large teams requires more than a 15-minute "scrum of scrums." Instead, you will find that your team leads will coordinate project management issues, your product owners requirements issues, and your architecture owners technical issues. This leadership team will typically be headed up by a program management specialist [5].

The scaling factors we described earlier should make it clear that there is more to scaling agile than adopting a few additional practices. You saw in the discussion around how to tailor your approach to identifying an initial technical strategy that various scaling factors will change how you approach certain issues. The level of detail in your technical strategy will vary, the way in which you approach modeling will vary, the views you capture may vary (each of the architecture view types listed in Figure 4 break down into multiple views), and even how you choose to deliver the solution may vary. And this is one of the simpler goals to understand, which is why we used it as an example. Construction goals of Produce a Potentially Consumable Solution and Move Closer to Deployable Release are much more complex and thus trickier to tailor.

Are Organizations Succeeding With Agility at Scale?

Heck yes! In the summer of 2012, Scott W. Ambler ran a survey for Dr. Dobb's Journal—a follow up to a similar 2009 survey, which explored whether organizations were successful at applying agility at scale [6]. The survey was designed to determine whether organizations were attempting to apply agile techniques at scale, whether anyone was succeeding at doing so, and whether anyone was struggling to do so. The answers to those three questions, respectively, are yes, yes, and yes. Although most agile teams are 20 people or less, some organizations are applying agile techniques on team sizes of several hundred. Organizations are successfully applying agile techniques at all levels of geographic distribution, at all levels of organizational distribution, and at all levels of domain and technical complexity. They are even succeeding at applying agile in regulatory environments, including life critical ones.

Where some organizations are succeeding with agility at scale, the survey also found that some organizations are unfortunately struggling to do so, the implication being that it is not a slam dunk after all. Our experience has been that the teams that get into the most trouble when applying agility at scale are the ones that are still trying to apply strategies geared for small, co-located teams in relatively straightforward situations. Another common failure pattern seems to be overly focused on construction-phase issues while underestimating the challenges associated with initiating an agile project successfully. A third common failure pattern is not involving at least a few people on the team with experience applying agile successfully in similar situations.

Woodward et. al. [7] describes the experiences of several IBM teams at applying agile at scale, particularly in geographically distributed situations as well as on large teams often in complex scenarios. Several teams were globally distributed, several teams numbered in the hundreds of developers, and several teams had both of these attributes. In some cases the teams were working on existing products with millions of lines of legacy code that had been written years, and sometimes decades earlier, using non-agile techniques. In all cases the IBM teams found they needed to tailor a hybrid, disciplined agile approach that reflected the situation they found themselves in. Alan Brown [8] describes his experiences helping organizations to scale agile, including a European financial institution applying early versions of the DAD framework. This organization successfully adopted and tailored agile delivery strategies at scale with project teams that were "in flight" as they could not simply shut down IT for several months while they retooled their staff to become agile.

Other surveys have explored how scaling factors affect project success rates [9]. For example as team size gets larger, the success rate goes down—something which is true for all development paradigms. Additionally, as a team becomes more geographically distributed, the success rate corresponding drops—something that is also true for all development paradigms. Perhaps more importantly the surveys have found that regardless of size or geographic distribution, the success rate of agile teams is statistically as great or greater than that of traditional teams. If your organization is still struggling with the decision as to whether they should apply agile at scale they may take comfort in this observation.

An interesting aspect of these surveys is that they did not force a single definition of success, such as "on time and on budget", on the respondents. Instead they asked respondents to answer the questions from the point of view of the actual success criteria for the projects. The surveys were designed this way to reflect the fact that there is no one single definition of project success but instead it varies between teams. For all of these surveys the questions as they were originally asked and the source data from the survey are available free of charge. If you do not trust our analysis of the data you are welcome to do your own.

You Need Greater Discipline

In this article we argued that there are many factors, not just team size, to consider when scaling agile. We also argued that you need an adequate foundation from which to scale agile, and that foundation we believe is DAD. The DAD process decision framework defines a full delivery lifecycle, showing how to deliver an agile project from start to finish, which is a hybrid of existing agile methodologies and techniques. DAD promotes an enterprise-aware, governed, goal-driven approach to agile solution delivery. DAD's goal-driven nature is the key to tailoring it to support agility at scale. Although many organizations are successfully applying agile strategies at scale, at the same time many organizations are getting into trouble doing so. The teams that run aground at scale invariably do so because they apply the simplistic approaches espoused by other agile methods in situations where they have little hope of success. It is definitely possible, and desirable, to apply agile techniques at scale—you just need to take a disciplined approach to doing so.

Acknowledgments:

Material for this article was summarized from the book Disciplined Agile Delivery: A Practitioners Guide to Agile Software Delivery in the Enterprise (IBM Press, June 2012) by Scott W. Ambler and Mark Lines. For more about the DAD framework, visit <http://www.DisciplinedAgileDelivery.com>.

Disclaimer:

CMMI® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

ABOUT THE AUTHORS

Scott W. Ambler is a Senior Consulting Partner at Scott Ambler + Associates and he helps organizations around the world to help them to improve their software processes. He is the founder of the Agile Modeling (AM), Agile Data (AD), Disciplined Agile Delivery (DAD), and Enterprise Unified Process (EUP) methodologies and creator of the Agile Scaling Model (ASM). Scott is the (co-)author of 21 books, including Disciplined Agile Delivery, Refactoring Databases, Agile Modeling, Agile Database Techniques, The Object Primer 3rd Edition, and The Enterprise Unified Process.

E-mail: Scott@scottambler.com.

Mark Lines is the Senior Managing Partner at Scott Ambler + Associates. He is a disciplined agile black belt and mentors organizations on all aspects of software development. He is passionate about reducing the huge waste in most IT organizations, and demonstrates hands-on approaches to speeding execution and improving quality with agile and lean techniques. Mark provides IT assessments, and executes course corrections to turn around troubled projects. He writes for many publications and is a frequent speaker at industry conferences.

E-mail: Mark@scottambler.com.

REFERENCES

1. Ambler, S.W. (2009). The Agile Scaling Model: Adapting Agile Methods for Complex Environments. IBM Whitepaper. <ftp://ftp.software.ibm.com/common/ssi/sa/wh/n/raw14204usen/RAW14204USEN.PDF Retrieved Jan 2, 2013>.
2. Boehm, B.W. and Turner, R. (2003). Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley/Pearson Education.
3. Kruchten, P (2012). The frog and the octopus: a conceptual model of software development. <http://arxiv.org/abs/1209.1327> Retrieved Jan 2, 2013.
4. McMahon, Paul E. (2010). Integrating CMMI and Agile Development: Case Studies and Proven Techniques for Faster Performance Improvement. Addison-Wesley Professional.
5. Ambler, S.W. and Lines, M.J. (2012). Disciplined Agile Delivery: A Practitioner's Guide to Agile Solution Delivery in the Enterprise. IBM Press.
6. Ambler, S.W. (2012). Agility at Scale Survey: Results from the Summer 2012 Dr. Dobb's Journal State of the IT Union Survey. <http://www.ambysoft.com/surveys/stateOfITUnion201209.html> Retrieved Jan 2, 2013.
7. Woodward, E., Surdek, S., and Ganis, M. (2010). A Practical Guide to Distributed Scrum. IBM Press.
8. Brown, A.W. (2012). Enterprise Software Agility: Bringing Agility and Efficiency to the Global Software Supply Chain. Pearson Education.
9. Ambler, S.W. (2013). Surveys Exploring the Current State of Information Technology Practices. <http://www.ambysoft.com/surveys/> Retrieved Jan 4, 2013.