Disciplined Agile Delivery in the Enterprise

Opening Statement
by Scott W. Ambler ................................................................. 3

Beyond Scrum + XP: Agile Architecture Practice
by Ipek Ozkaya, Robert L. Nord, Stephany Bellomo, and Heidi Brayer .................... 6

Beyond Software: Composite Agile Method and Strategy (CAMS) and Agile Business
by Bhuvan Unhelkar ................................................................. 10

Integrating Lean Thinking to Achieve Multi-Product and Multi-Team Agility
by Al Shalloway ................................................................. 18

A Disciplined Approach to Agile Project Deployments
by Matt Ganis and Sanjeev Kumar V Marimekala ........................................... 24

The Central Role of Personality in Agile Success
by Themist Themistocleous ....................................................... 30

“The Agile and Lean movements clearly have a lot of life left in them. My expectation is that we’ll spend the next decade or more adopting disciplined Agile strategies to enable us to solve more complex problems and make possible a truly Agile enterprise.”

— Scott W. Ambler, Guest Editor
Get global perspectives and solutions to some of the most critical business technology issues facing organizations today!

Your Front-Row Seat to IT Management Debate at the Highest Level!

Every day, your organization is confronted with the stark reality of having to achieve more aggressive goals with a shrinking budget, ever-changing requirements, and impossible deadlines.

Few of you have the time to develop well-supported arguments on how to get your organization to improve its IT operations. It’s a tough trap: you know solutions are out there, but you’re too busy to identify them and convince your organization to implement them.

Advice, Solutions, and Experience That You Can Rely On

A Cutter IT Journal subscription helps you break out of the trap. Every month, Cutter IT Journal features a select Guest Editor who articulates the controversial issues, offers his or her opinion on them, invites others to introduce opposing viewpoints, and sparks a lively debate.

Cutter IT Journal provides you the opportunity to experience a variety of perspectives: viewpoints that will be instrumental in advancing the cause of better software development. No matter where you stand on these issues, the thoughtful discourse delivered in Cutter IT Journal will certainly help you clarify your position.

In addition to your monthly journal issue, you will also receive its companion Cutter IT Advisor. Each Advisor brings you practical advice and thoughtful analysis from well-known and respected experts in the IT field. Learn from the experiences of others, including what you should avoid and what you should consider implementing.

As a subscriber to Cutter IT Journal, you’ll stay up to date on important IT issues such as project management, security, risk management, business intelligence, sourcing, enterprise architecture, requirements, trends in technology, and more. Whatever the topic, you can be sure you’ll receive frank, unbiased opinions, in the no-holds-barred manner Cutter IT Journal is known for.

Don’t miss upcoming issues on:

- Is IT Still Relevant?
- Security and the Internet of Things
- Profiting in the API Economy
- Social Media Data Analysis
- Educating Next Generation IT

Special offer for new subscribers!

Begin your subscription to Cutter IT Journal today and save $100 off the regular subscription rate! Plus receive all 2012 Cutter IT Journal issues on CD-ROM.

To subscribe for just $385 ($485 outside N. America) and receive your Free CD-ROM, go to bookstore.cutter.com and enter Coupon Code CITJSAVECD at checkout. Or complete and return the form below by fax +1 781 648 8707, phone +1 781 648 8700, or email service@cutter.com. For more on Cutter IT Journal, please visit www.cutter.com/itjournal.html.

Visit bookstore.cutter.com to order individual Cutter IT Journal issues.

Special Offer: Save $100 on a New Subscription!

☐ YES! Please start my new, one-year subscription to Cutter IT Journal for just $385 (US $485 outside North America) — I save $100 off the regular price of $485/US $585! Plus send all 2012 issues on CD-ROM!

Name

Title

Company

Dept.

Address/PO. Box

Mailstop/Suite

City

State/Province

ZIP/Postal Code

Country

Phone

Fax

Email

Order online at bookstore.cutter.com (enter Priority Code CITJSAVECD to save $100). Or fax form to +1 781 648 8707, call +1 781 648 8700, or send email to service@cutter.com. Mail to Cutter Consortium, 37 Broadway, Suite 1, Arlington, MA 02474-5552, USA.
I’m honored to be the Guest Editor of this month’s Cutter IT Journal. This issue describes the experiences and findings of senior consultants and researchers who have been actively involved with helping organizations around the world apply Agile strategies in enterprise environments. The term “Agile software development” was first coined in 2001, and since then Agile techniques and ideas have been applied in all industries, all geographies, and, most importantly, in a wide range of situations. But many of these successes are small-scale within organizations — a few projects here and there have benefited, but rarely do you hear about large-scale enterprises successfully making a complete transition to Agile. The implication is that we still have a long way to go.

There are several important themes running through these articles. First, to succeed at Agile, you need to take a hybrid approach composed of strategies from Scrum, XP, Agile Modeling, Kanban, Unified Process, and, in some cases, even traditional methodologies. Second, when organizations apply Agile techniques at scale, they need to adopt more robust — or dare I say “mature”? — strategies than the Scrum community typically promotes. Third, Agile teams in today’s enterprises need to go beyond their team-focused strategies and become enterprise-aware. Fourth, to transition to true enterprise agility, you need to adopt greater discipline at the practitioner, team, and even organization level that goes beyond the mainstream Agile philosophies that emerged in the 1990s.

In our first article, Ipek Ozkaya, Robert L. Nord, Stephany Bellomo, and Heidi Brayer of the Software Engineering Institute set the pace by summarizing results of research into the actual practices of Agile teams. This study explored how organizations are adopting hybrid approaches that build on the strengths of Scrum and XP with Agile architecture practices to achieve the benefits of Agile across the lifecycle. They share evidence of this hybrid strategy gathered via interviews with representatives of five public- and private-sector organizations that operate in highly regulated markets. They found that when experienced practitioners encountered scaling problems, most did not apply Agile practices in isolation: “Rather, they used their expertise to creatively combine them with other practices, especially architecture, to respond effectively to stability issues while rapidly fielding projects.” The architecture strategies these practitioners used include initial architectural modeling/envisioning, monitoring technical debt, and addressing nonfunctional requirements. The authors conclude, “The experiences of these organizations support the stance that a more expansive application of hybrid practices is not only necessary but essential in balancing the opposing objectives of speed and stability.”

Cutter Senior Consultant Bhuvan Unhelkar, author of The Art of Agile Practice, provides a complementary view in his article on the Composite Agile Method and Strategy (CAMS) and Agile business. Unhelkar highlights the “need to consider the value Agile offers to business, far beyond that originally anticipated by the proponents of Agile in the software domain.” His CAMS framework shows how “embedding agility within formal planning results in an overall balance that not only benefits close-knit software development teams, but also the entire business.” He argues that CAMS promotes a balanced strategy that enables the benefits of Agile at an organizational level. I believe you will find that CAMS presents a well-considered, practical adoption and deployment strategy for Agile.

The CAMS framework encompasses three levels: the development level (typical of Agile methods such as Scrum and XP), the project level (characterized by a composite/hybrid Agile approach), and an organizational level. Unkelkar argues that a primary Agile challenge is applying Agile across an organization. He notes, “What seems to work for a small, close-knit team suddenly seems ‘all over the place’ when applied across the organization.” It is here that the concepts of
the Disciplined Agile Delivery (DAD) process framework can offer assistance. As Unhelkar observes, “The DAD framework provides a carefully construed mechanism that not only streamlines IT work, but more importantly, enables scaling.” The article further highlights the need to avoid what the author refers to as “method friction.” When several different methods — each introduced by a different role — are in use within the organization, it can cause friction, and in turn reduce quality and flexibility. Such DAD concepts as a focus on solutions (not just software) and on scalability can enable the composite Agile approach CAMS represents.

Agile is not about team-iterative development, but rather about the incremental delivery of business value.

In our third article, Al Shalloway argues that Agile success has eluded many larger IT organizations and software development organizations. “Rather than productivity and profit,” Shalloway says, “they experience mediocre results or worse.” He believes that the Agile movement’s failure to adopt a solid systems perspective at the enterprise level is a significant factor. He writes, “It is not enough to try to build up team by team, with management trying to remove impediments along the way; instead, it requires an integration of business stakeholders, middle management, and effective teams. It requires both systems thinking and the discipline that systems thinking demands.”

Shalloway suggests that bringing discipline and coordination to and across Agile teams requires going beyond classic Agile with its team-centric point of view. In particular, “it requires extending Agile with Lean principles and practices,” a mindset he calls “Lean-Agile.” Chief among these is systems thinking, which encourages organizations to examine the development workflow as a whole, seeing how each part affects the others. He notes that “local optimizations without this view may have little — or even a harmful — impact on the whole.” For example, the “self-organizing team” is a very good Agile practice. It works at the team level because teams can use their own local knowledge to adjust to their particular situation. At scale, however, a broader perspective is required. “Management provides this perspective,” Shalloway writes, “by creating the context within which teams work together. When several teams are involved, a lack of management oversight inevitably leads to a lack of team coordination, which leads to wasted effort.”

This Lean-Agile mindset suggests three new disciplines for Agile organizations. The first is that stakeholders cannot begin more projects than the development organization has capacity for. In other words, manage your work in progress across your overall portfolio. Second, project teams must work in unison. If each team chooses stories from its backlog at its own discretion, the completed features will likely fail to produce end-to-end functionality. Third, Agile teams must let someone who sees the bigger picture decide what they should be working on. This implies the need both for a Lean-Agile enterprise architecture effort as well as a governance effort. Shalloway ends with the observation that “Agile methods informed by Lean Thinking lead to the disciplined type of Agile required for the Agile enterprise. We must remember that Agile is not about team-iterative development, but rather about the incremental delivery of business value.”

Next, IBM’s Matt Ganis and Sanjeev Kumar V Marimekala describe several key practices for becoming more disciplined, and thereby more effective, in your approach to Agile solution delivery. They describe how they used the DAD framework to create a process within their organization that supports a great number of both Agile and traditional project deployments. The authors write:

Early on, we realized that many of our enterprise processes wouldn’t be eliminated with the advent of an Agile approach. Many (if not all) of our projects, while claiming to be Agile, needed to collaborate with other teams/resources such as IT deployment (data center teams), architecture teams, database administrators, and others. While our Agile teams were creating usable software at a steady drumbeat, we struggled with the inability (or lack) of IT staff to deploy applications to the network where they could begin realizing value.

UPCOMING TOPICS IN CUTTER IT JOURNAL

JULY
Rebecca Herold
Big Data Privacy

AUGUST
Simon Woodworth
Mobile Security Challenges
Ganis and Marimekala recommend the adoption of six practices (or “disciplines,” using Shalloway’s vernacular) to facilitate application deployment. First, promote open communication channels across your IT organization to enable people on disparate teams to work together more effectively. Second, focus on reducing waste (or better yet, eliminate it completely) in order to streamline your overall strategy. Third, strive for the best utilization of resources from a business perspective and thereby maximize the return on your IT investment. Fourth, apply resources where needed so as to effectively utilize team skills. Fifth, maintain a proper lead time for each project. In other words, invest in project inception/initialization activities such as initial requirements envisioning, architecture modeling, planning, and team building. Sixth, keep it simple — an echo of the ninth principle behind the Agile Manifesto.

We round out this issue of Cutter IT Journal with an article by South African software industry entrepreneur Themi Themistocleous, who explores the softer aspects of Agile software delivery. As Themistocleous observes:

With Agile development, it becomes even more important to have the right people in the right roles, as all team members are required to be self-organizing and cross-functional and to take responsibility for the success of the project. Personality types are starting to be recognized as an important factor in the success of Agile projects, and the first studies and books on the topic are making their way into the Agile world.

Themistocleous describes three important factors that contribute to software development success. First is the culture of the organization. In particular, he describes how the best organization he ever worked for strived to provide an egalitarian, nurturing environment that enabled people to work well together. The second factor is the correct use of development processes. For example, he describes how the DAD framework provides ways to provide effective governance without needless bureaucracy by focusing on process goals instead of on paperwork. The third factor is enabling people to take on the roles best suited for them.

I want to leave you with several important observations. First, organizations are finding that it’s a lot of hard work figuring out how to make all this Agile stuff actually work in practice. Although the Scrum message of “start with a small kernel and add what you need” is attractive, in practice it’s proving difficult, time-consuming, and expensive. As you will see in these articles, it is likely more effective to start with a framework such as Disciplined Agile Delivery, which describes a hybrid approach that supports a full delivery lifecycle. Furthermore, DAD promotes a solution focus, not just a software focus, and a strategy that is enterprise aware and not just team aware. More importantly, it answers the question “How do all the Agile practices and strategies fit together?” with its goal-based approach. Instead of starting with something small, why not start with something sufficient?

Second, the Agile and Lean movements clearly have a lot of life left in them. My expectation is that we’ll spend the next decade or more adopting disciplined Agile strategies to enable us to solve more complex problems and make possible a truly Agile enterprise. Stay tuned for future writings on these subjects!

Scott W. Ambler is a Senior Consultant with Cutter Consortium’s Business & Enterprise Architecture and Agile Product & Project Management practices. He is the thought leader behind the Disciplined Agile Delivery (DAD) process decision framework, Agile Model Driven Development (AMDD), the Agile Data (AD) method, and the Enterprise Unified Process (EUP), working with clients around the world to improve the way they develop software.

Mr. Ambler is coauthor of several software development books, including Disciplined Agile Delivery, Agile Modeling, The Elements of UML 2.0 Style, Agile Database Techniques, and The Enterprise Unified Process. He is also a Senior Contributing Editor with Dr. Dobb’s. Mr. Ambler has spoken at a wide variety of international conferences, including Agile 20XX, Software Development, IBM Innovate, Java Expo, and Application Development. He can be reached at sambler@cutter.com.
The belief that Agile requires small colocated teams, downplays architectures, and delivers no documentation still prevails among many software practitioners. The reality is that organizations, in their quest to rapidly field projects, are building on the strengths of Scrum and XP. They are doing so to creatively combine Agile architecture practices to achieve the benefits of Agile across the lifecycle.

We found further evidence of this more expansive approach when we interviewed representatives of five government and commercial organizations that operate in highly regulated settings. Our goal was to gain a better understanding of success and failure factors in rapidly fielded projects using Agile software development practices. The reality of a highly regulated environment is that practitioners must often balance the demand to quickly deliver functionality with a desire for a stable, reliable system, especially when faced with the challenge of sustaining such systems for several decades. In addition, in highly regulated environments such as avionics, financial services, and healthcare, software development teams need to interface with system engineering, deployment, and quality assurance teams that may be operating under different development and delivery tempos. These competing pressures often result in projects marked by high initial velocity followed by a slowdown that results from stability issues (see Figure 1).

The interviews showed that most experienced practitioners, when faced with challenges, did not apply Agile practices in a silo. Rather, they used their expertise to creatively combine them with other practices, especially architecture, to respond effectively to stability issues while rapidly fielding projects. Doing so helped them avoid significant disruptions in velocity. The experiences of these organizations support the stance that a more expansive application of hybrid practices is not only necessary but essential in balancing the opposing objectives of speed and stability.

In this article, we highlight several approaches applied by these organizations and provide a more in-depth look at two of the practices: release planning with architecture considerations and roadmap planning with external dependency management.

**SPEED VS. STABILITY**

The essence of balancing speed and stability involves achieving and preserving a software development state that enables teams to deliver releases that stakeholders value at a tempo that makes sense for their business.

The desired software development state is different for each organization and needs to be understood clearly. This is a state in which architecture (often in the form of platforms and application frameworks), supporting tool environments, practices, processes, and team structures exist to support efficient and sustainable development of features. The entire organization, including development teams, management, and stakeholders, must have visibility into the desired state, so that they neither overoptimize the supporting development infrastructure nor quit working on it.

We asked senior software developers and managers to describe factors that either enabled or inhibited the speed of delivery and the degree of stability in the software product. The factors fall into one of three common situations:

1. When the project was going well, teams applied foundational Agile practices commonly touted as “enablers of success,” such as daily Scrum meetings,
a Scrum collaborative management style, continuous integration, test-driven development, and so on. Small dedicated teams are able to stay within bounds due to the well-known nature of the infrastructure and the limited scope of the project. Teams facing issues of scale and complexity are able to stay within bounds because of awareness across the organization of the need to maintain the infrastructure.

2. When teams encountered a problem that was taking them away from their desired state, they would often combine Agile practices with architecture and other disciplines, such as management and engineering, to make incremental adjustments to ensure they had sufficient technological infrastructure to support development.

3. When teams encountered problems that were not visible to management and stakeholders, the adjustments were disruptive. Solutions were delayed until the chronic problem became visible. In certain cases, it was not possible to adjust course because the problem became visible too late; the project was not able to deliver, and the team failed.

HYBRID PRACTICES THAT ENABLE FAST, STABLE DEVELOPMENT

Our interviews revealed the following examples of Agile architecture practices that enable speed and stability:

- Release planning with architecture considerations
- Prototyping with a quality attribute focus
- Roadmap planning with external dependency analysis
- Merging of test-driven practices (e.g., automated test-driven development and continuous integration) with a focus on runtime qualities (e.g., performance, scalability, and security)
- Technical debt monitoring with a quality attribute focus

These Agile architecture practices allow more experienced practitioners to avoid project slowdowns related to stability issues with minimal disruption to capability delivery. While these practices have been advocated for a while, using them within the confines of a well-defined process, such as Scrum, becomes challenging.

We will now describe in more depth two of these practices that all five organizations used: release planning with architecture considerations and roadmap planning with external dependency analysis.

Release Planning with Architecture Considerations

Development teams often incorporated architecture considerations into release planning in response to problems associated with prioritizing features visible to the stakeholder. One organization adopted the Scrum release planning management process without much visibility into the infrastructure needs. The increasing focus on rapid delivery inevitably made the organization realize that its teams needed to work in parallel to meet schedule demands. So the business moved from a centralized development model to a geographically distributed work model. Delivery slowed, however, because there was not enough architectural definition in the feature documentation to allow the teams to work independently. This triggered a closer look into the infrastructure and a more stable architecture. The outcome was to incorporate more explicit infrastructure and architecture planning into release planning and not simply focus on the high-priority features.

The important and unexpected observation revealed by our interviews is that all the organizations recognized that without incorporating architecture into release planning, it is not possible to achieve the expected delivery tempo after significant and unexpected change. In all cases, the projects initially appeared within bounds of their desired state, and issues were visible. It was only after the disruption that the teams sought deeper visibility into the project.

Roadmap Planning with External Dependency Analysis

One organization incorporated external dependency analysis into its roadmap planning process. This approach reduced the risk of being blindsided by unanticipated conditions due to dependencies on expertise external to the team, infrastructure components governed by other parties, or difficult-to-reach users. For instance, during the development of an operational sprint, several firewall ports governed by an external party were closed without notice, causing sporadic stability issues that were difficult to troubleshoot. Until this event, the dependency on security decisions governed by another team had not been realized as a critical dependency that could impact the development effort.

Since this problem was holding up development, the team took immediate action to analyze and reassess external dependency risks that could affect their design decisions and development. Team members then devised a mitigation strategy for each risk. Some strategies required modifications to the change management notification process, and others required a deeper understanding of dependencies on components being developed by...
other teams. The roadmap document, which contained a description of development by phases, was used to capture external dependency risks and mitigation strategies at the portfolio level. The team incorporated the change into the ongoing practices with limited disruption to workflow; therefore, it followed the incremental response cycle. The team also adopted the practice of continuing to conduct external dependency analysis regularly to identify external dependency risks at the roadmap level.

The criticality and impact of technical dependencies intensify at scale. Such dependencies are also easy to overlook because they may not be exercised daily or even at each sprint. Therefore, the roadmapping level is the right place to surface and track these dependencies. It allows the architectural decisions to propagate correctly among software elements as a system is developed in breadth and depth and among the multiple development teams across an organization.

**INHIBITING FACTORS**

The factors that prevented development teams from rapidly delivering the software product, or mired them in a state outside the bounds of acceptable software development, included often-observed inhibitors such as slow business decision-making processes, limitations in measuring architectural technical debt, overdependency on the architect for architecture knowledge, and stability-related efforts not entirely visible to the business.

The inability to deal with scale and complexity also emerged as a factor. Development teams from four of the five organizations described situations in which they were not able to complete test cases within the targeted iteration due to increasing software complexity and limitations in expertise and/or tools. They also reported that excessive focus on speed, and difficulty in making architectural problems visible to the business side of an operation, often led to major redesigns or bug-fixing sprints.

Many of these negative outcomes are the result of inconsistent and incorrect applications of Agile and/or architecture practices. Several factors can be traced back to their enabling counterparts. For example, a desire to quickly deliver features caused stakeholders to overlook the importance of stability and limited requirements analysis and stability-related work. Demonstrating the criticality of stability required improved measurement of technical debt. Two of the organizations took actions to improve their visibility into technical debt by tracking it explicitly in their backlogs.

Organizations acknowledged the tradeoffs they experienced when taking shortcuts in software development to accelerate delivery. They are now more aware of how degraded quality leads to technical debt and are taking steps to address it. In response to business pressure, organizations sometimes embedded architectural changes within unrelated features during development. This lack of transparency can result in incorrect productivity measures as well as unanticipated schedule impacts. They expressed the belief that if they were able to make technical debt more visible to stakeholders, they could avoid potentially costly and disruptive changes in favor of more incremental changes supporting a sustainable delivery tempo. We call this “technical debt monitoring with a quality attribute focus.”

**TRANSITIONING ENABLERS OF CHANGE**

In a recent blog post, Ken Schwaber said he would like to change the mindset of “Scrum But” to “Scrum And.” He explained that the use of “Scrum And” characterizes an organization that is on a path of continuous improvement in software development beyond the basic use of Scrum. He gave this example to illustrate the concept of extending Scrum: “We use Scrum, and we are continuously building, testing, and deploying our increments every Sprint.” The experiences described by organizations we interviewed support the stance that practice extensions are needed and anticipated in iterative and incremental development. Evidence from Guest Editor and Cutter Senior Consultant Scott Ambler’s Agile project initiation survey has also shown consistent results with the experience of the teams we interviewed.

In this new “Agile And” world, some organizations are moving toward Disciplined Agile Delivery (DAD), a hybrid, people-focused IT approach developed by Ambler. In line with our observations from practice, such new-generation software development practices place the emphasis on a full delivery lifecycle that is process goal–driven and incorporates architecture and enterprise awareness. Others are embracing the Scaled Agile Framework (SAFe) developed by Dean Leffingwell and colleagues, not only incorporating architecture explicitly into the development lifecycle, but also Lean methods and portfolio management.

At the SEI, we work with organizations that must operate in a climate of ever-shrinking budgets coupled with near-constant demands for new capabilities. We advise organizations looking to adopt or enhance Agile practices in their pursuit of rapid delivery to begin by conducting a review of architecture-centric risk factors for
adoption of large-scale Agile software development. While Agile architecture practices can help these organizations ensure the stability of the systems they are fielding, it is important to understand the root causes of the inability to deliver at the expected pace and how the tension between speed and stability is managed. Organizations must also make the problems more visible to developers, management, and stakeholders.  

When considering how to combine Agile and architecture practices, organizations must first ask the following questions:

- Are we delivering software to our customer at an expected pace?
- Are we aware of problems that are cropping up as a result of losing focus on architecting when Agile adoption activities become the primary focus?
- Does our technical roadmap address short-term and long-term issues?
- Does the team of software developers have skills that would enable them to successfully implement Agile and architecture?
- Do we have visibility into not only the project management of the system, but also the quality expected from the system?

We hope that by codifying and sharing the practices described above, other organizations can learn to apply these approaches to contend with the demands of rapidly delivering software that is reliable, stable, and flexible in a fast-changing environment.

**ACKNOWLEDGMENT**

This material is based upon work funded and supported by the US Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

**ENDNOTES**


2Schwaber, Ken. “Scrum But Replaced by Scrum And.” Telling It Like It Is, 5 April 2012 (http://kenschwaber.wordpress.com/2012/04/05/scrum-but-replaced-by-scrum-and).


Ipek Ozkaya is a senior member of the technical staff at the Carnegie Mellon Software Engineering Institute (SEI). With her team at the SEI, she works to help organizations improve their software development efficiency and system evolution. Dr. Ozkaya’s work focuses on software architecture practices, software economics, and requirements management. She serves as chair of the advisory board of IEEE Software magazine and as an adjunct faculty member for the Master of Software Engineering Program at Carnegie Mellon University (CMU). Dr. Ozkaya holds a PhD in computational design from CMU. She can be reached at ozkaya@sei.cmu.edu.

Robert L. Nord is a senior member of the technical staff at the SEI, where he focuses on Agile and architecting at scale and works to develop and communicate effective methods and practices for software architecture. He is coauthor of the practitioner-oriented books Applied Software Architecture and Documenting Software Architectures: Views and Beyond, and he lectures on architecture-centric approaches. Dr. Nord is a member of the steering committee of the WICSA Conference Series, in addition to organizing events at software engineering, Agile, and architecture venues. He earned a PhD from CMU and is a distinguished member of the ACM. He can be reached at rr@sei.cmu.edu.

Stephany Bellomo is a senior member of the technical staff at the SEI, serving in the Architecture Practices group and the Value-Driven Incremental Development research team. Ms. Bellomo is currently focused on work in the area of Agile and architecture practices. Prior to that, she focused on other areas such as service-oriented systems and Agile and information assurance. In support of this work, she has authored papers/reports, delivered presentations, served on panels, and provided onsite technical support. Ms. Bellomo was Tutorial Chair for the 2013 SEI Architecture Technology User Network (SATURN) Conference and a technical reviewer for submissions to the SEI Software Product Lines conference. She teaches the SEI courses “Service-Oriented Architecture Migration of Legacy Components” and “Software Architecture Principles and Practice.” Ms. Bellomo received a master’s degree in software engineering from George Mason University. She can be reached at sbellomo@sei.cmu.edu.

Heidi Brayer is a writer and editor at the SEI. Her work focuses on writing and editing a weekly blog to highlight SEI research initiatives. She also produces the SEI Podcast Series, which is available on the SEI website and on CMU’s iTunes U channel. She earned a master’s degree from CMU’s Master of Professional Writing program. She can be reached at hap@sei.cmu.edu.
Beyond Software: Composite Agile Method and Strategy (CAMS) and Agile Business
by Bhuvan Unhelkar

The popularity of Agile methods, such as Scrum and XP, stems primarily from a rapid, high-quality, value-based, and collaborative approach to software development. The iterative and incremental character of agility together with its highly collaborative nature meets the changing needs of its stakeholders (as aptly discussed by this issue’s Guest Editor and fellow Cutter Senior Consultant Scott Ambler as early as 2006). Enhanced user satisfaction and project visibility have started endearing Agile to many different areas of a business organization much beyond software. To be of increasing value to business, Agile needs to be in balance, comprising a judicious combination of practices together with the well-known and often mandatory requirements of a formal, planned method. In order to realize Agile’s true benefits, senior decision makers need to let this balanced agility permeate their entire organization as a strategy rather than just a software development method.

The value of Agile when it becomes a balanced organizational strategy is encapsulated in what I call the Composite Agile Method and Strategy (CAMS). In my recently published book The Art of Agile Practice, I argue that CAMS presents a practical adoption and deployment strategy for Agile that is based on careful consideration of multiple organizational factors that require a disciplined, collaborative approach. Primary amongst these factors is a highly cohesive, close-knit, and disciplined — in fact, self-disciplined — team. Disciplined Agile Delivery (DAD) embodies this truth, as it calls itself a “people-first, learning-oriented hybrid Agile approach to IT solution delivery.” The DAD framework provides a carefully construed mechanism that not only streamlines IT work, but more importantly, enables scaling. This scaling of agility is vital in organizational adoption as it facilitates the use of Agile in the operational aspects of a business, not just in projects. The underlying concepts of DAD, combined with the way CAMS has been put together, pave the path for successful adoption of agility at an organizational level. This article presents CAMS as a highly practical way to use Agile and shows how DAD can help in the deployment of CAMS at an organizational level.

ORGANIZATIONAL AGILE STRATEGY

The opportunity to extend and apply the concepts of Agile across an entire organization promises to radicalize the organizational structure, transform the way in which business views IT, and further blur IT-business boundaries — for the better. What we have learned in the application of agility in software can hold true in any project and beyond, including the day-to-day operations of the organization. Practically, there is no reason why the nuggets of agility that help deliver software on time and within budget can’t help organizations become more responsive overall. For example, Agile can help a business put together new and varied products in a timely manner, create new services and content, and offer more value to its customers independent of time and location. Agile can make a business collaborative, highly interconnected, communicative, and in sync with the business ecosystem (composed of industry, government, and society).

Figure 1 summarizes the three evolving uses of Agile — firstly as commonly understood at the development level, then the composite Agile approach to projects, and finally applied across the entire organization. When thus viewed, Agile starts evolving into more than a method. It starts becoming an essential element of the business strategy of an organization. For example, an organization can internally decree a user story as a standard means of communication that will be displayed on a Big Visible Chart. This technique may appear across all tiers of an organization rather than being used only to specify the functional needs of a system (such as the HR department writing a user story/feature for recruitment purposes). Other Agile techniques start becoming visible too; for example:
The training department using Big Visible Charts to schedule a leadership seminar for senior executives

The daily production planning meeting following the precepts of a daily stand-up for its decision making

The accounts department conducting a retrospective of its last auditing exercise

Agility vitalizes the complete structure and dynamics of an organization by opening up communications across all tiers, creating an atmosphere of trust and honesty, and inculcating courage to change at the grass-roots level.

Not unexpectedly, though, organizational adoption and use of Agile are fraught with challenges. These challenges are different in nature and scale than those of training and up-skilling an individual team member in Agile techniques and practices. For example, at an organizational level, it is not enough to train individuals in how a daily stand-up meeting is conducted and how a story card is written. Instead, a strategic and holistic stance is required to deploy Agile across departments and divisions and amongst myriad roles, including developers, customers, architects, and business stakeholders. Agile needs to become “routine” in the organization in order to encompass its complex and often subjective rewards structure (a topic that could be a paper in itself). An organization trying to embrace Agile will have to come up with a different way to reward its employees, as Agile is invariably team-based.

While considering Agile as a strategy, an organization that wishes to adopt Agile must also be fully cognizant of the many methods already used in the organization and their impact on each other. For example, it must take into account the way in which formal project management (e.g., PMBOK) impacts Agile, the way IT governance limits agility, and the varying needs of formal quality assurance and testing programs together with business analysis. Overlooking such considerations can lead to “method friction,” which will be detrimental to organizational acceptance of Agile.8

**COMPOSITE AGILE METHOD AND STRATEGY (CAMS)**

In the *Art of Agile Practice,* I discuss why organizations should embed the values and practices of Agile within their existing planned and formal approaches and how they can go about doing so. Embedding agility within formal planning results in an overall balance that not only benefits close-knit software development teams, but also the entire business — including its non-IT processes. As noted above, CAMS represents such a balanced combination of agility together with formal planning, control, and documentation.

Waterfall, spiral, and iterative-incremental software development lifecycles (SDLCs) need not be the maligned words that they are sometimes made out to be in a pure Agile environment. These lifecycles exist because at one time (say, in the 1970s and early 1980s), the IT world was crying out for formality and planning. In fact, many a software disaster was averted even before the first line of code was written, as the plans and models “rang the bells” sufficiently in advance to indicate that the project would not succeed. In the solution space, experienced programmers created patterns, and designers used those patterns formally to create quality. In-depth requirements models helped users to understand proposed solutions as well as give input into what they expected. Despite these advantages of planned processes, however, the human tendency of swinging from one extreme to the other seems to have occurred in the methods space. Many projects got stuck in the quagmire of excessive planning, reams of documentation, endless meetings on process compliance, and no executable output the end user could see. The IT project world was ripe for Agile.

Those who put together the Agile Manifesto deserve both our thanks and credit for untangling the myriad knots resulting from excessive, formal application of methods in software development. And yet, if you observe closely, you may find concerns gradually creeping in. Scalability appears to be a major challenge in deploying Agile. My everyday work discussions with fellow consultants and with business stakeholders indicate that Agile is welcomed with a mixture of hope...
and uncertainty. Typically this happens in large-scale deployments, outsourced/offshored development, large ERP (package) configuration projects, and infrastructure developments.

The best way to use Agile, as I found in practice, is to use it in balance. Furthermore, it’s also helpful to view Agile beyond software and in the overall business solutions space. The following summarizes my view of an Agile Manifesto that strives for balance:

- Individuals and interactions together with processes and tools
- Working solutions together with comprehensive documentation
- Customer collaboration together with contract negotiation
- Responding to change together with following a plan

This balanced view of the manifesto clarified many things for me as I practiced Agile; it reminded me of the importance of the other side of each of the sentences. In turn, it led me to put together CAMS, a foundation for business technology synergy that embraces methods at various organizational levels depending on their relevance, principles, and practices. In bringing together planned and Agile behaviors, as well as business, technology, and operational views of an organization, composite Agile relies on the linchpin of balance. In fact, agility is all about balance — between technology and business, between formality and flexibility, between hierarchical structures and informal shared team structures, and so on. This underscoring of balance is an important factor in promoting Agile across the entire organization.

CAMS revolves around two key themes:

1. Agile provides maximum value to projects when used in a composite mode that gives due deference to the existing methods in the organization (e.g., Unified Process, or even PMBOK/Prince2)

2. Agile is more than a method. It’s a business strategy that needs to be applied in an holistic manner across the entire organization (effectively being a culture change in the use of existing frameworks such as Six Sigma, Lean, Kaizen, and Kanban, rather than a piecemeal application within the software development arena).

Figures 2 and 3 explain these themes. Figure 2 depicts multiple layers of methods used in an organization. These include, for example, Six Sigma, COBIT, Prince2, RUP, and eventually Scrum. Apart from indicating the iterations (three in this case), this figure stresses the fact that these methods need to work together cohesively. Figure 3 shows how this happens in practice. Various areas of work in an organization (e.g., user, business, project management, development, and quality) need
to work together, keeping all methods in mind — albeit using only the relevant slice from each of the methods. CAMS encompasses the entire gamut of methods in an organization, including business processes, governance standards, project management, quality management, and business analysis.

CAMS starts by setting the right mix of planning and agility in any organizational initiative. What is even more important in CAMS is maintaining balance as the initiative progresses, continuously changing the mix of planned and Agile elements. Agile experts Alistair Cockburn and Jim Highsmith\(^\text{10}\) stress amicability, talent, skill, and communication as critical factors in the success of Agile. CAMS builds on the overall understanding of these “soft” factors by including sociocultural and psychological factors and applying them at the organizational level. For example, training and coaching of the organization’s decision makers ensures strategy alignment, risk management, change management, and behavioral practices that can add significant agility to the overall enterprise (as against merely software development).

**FACILITATING CAMS ADOPTION WITH DAD**

Adoption of CAMS can present some interesting challenges, ranging from the technical and metrics-related through to the psychosociological. At the forefront of these challenges, though, is the issue of scaling Agile. What seems to work for a small, close-knit team suddenly seems “all over the place” when applied across the organization.

This is where the concepts of the Disciplined Agile Delivery process framework can offer the greatest support. The DAD framework provides a mechanism for streamlining IT work and, even more importantly, enabling its scaling. DAD is an enterprise-aware and goal-driven framework that brings together strategies from Scrum, XP, Agile Modeling, Agile Data, Kanban, and devops. As such, the key advantage of DAD is its configurability, which also enables its scalability. Consider the six DAD scaling factors as discussed by Ambler:\(^\text{11}\)

1. Team size
2. Geographic distribution
3. Organizational distribution
4. Compliance
5. Domain complexity
6. Technical complexity

Except for the last one, all others are nontechnical factors that can apply at the organizational level as much as at the technical, project level.

CAMS has incorporated these wide-ranging factors in its framework at a strategic level with particular attention to the four business-critical elements — people, process, and money as well as technology — both within and outside business. Business stakeholders are likely to find this aspect of CAMS deployment valuable, as their interest lies, rightly, in business agility rather than software (or project-level) agility. Following are some of the ways DAD facilitates the adoption of CAMS across a business organization:

- **Planned processes impose a rigorous discipline on individuals.** This discipline, further enforced by tools, enables individuals of even “average” capacity to perform well in a group. DAD emphasizes this discipline aspect in a project and lays the responsibility of that discipline on the individuals. This concept can be used, together with modeling and process tools, to enable individuals to understand the need to align themselves to the project/team goals. Discipline is utilized here not only in creation of new models and designs, but also in reusing the ones created in earlier iterations. The way DAD assists in self-alignment validates the view I expressed in my balanced view of the Agile Manifesto: “Individuals and interactions together with processes and tools.”

- **CAMS promotes the creation of software assets that are “beyond software.”** These are the architectures, designs, working models, documentation, and plans. These artifacts (elements) were quite prominent in the era of formal, planned (pre-Agile) methods. DAD
focuses on technical as well as domain complexity and encourages the use of models to handle these complexities. Furthermore, DAD recognizes and emphasizes regulatory compliance, which requires and benefits from the modeling effort. These modeling and documentation assets are also eagerly sought and valued during mergers and acquisitions of IT companies, as they facilitate business solutions in the context of the industry vertical rather than just software solutions.

- **Collaboration with the customer is indispensable.** Many other fields have learned the inevitability of customer collaboration — for example, the consulting doctor spends time listening to the patient, and the insurance provider discusses the needs of the client in detail before offering coverage. CAMS applies Agile by encouraging collaboration between users and solution providers. At the same time, it balances this with the need for contract negotiations, as is inevitably required in the case of software solutions that are developed and maintained via outsourcing/offshoring agreements. Service-level agreements (SLAs) influence the development and maintenance of external solutions and are an integral part of contract management. DAD encourages organizations to consider how geographical distribution and organizational culture affect their processes for customer collaboration.

- **The ability to respond to change is the crucial difference between success and failure** and is one of Agile’s core attractions for business. Change is best understood by the decision makers of an organization if sufficient planning has been undertaken in the first place. CAMS focuses on detailed, formal planning, as such plans provide the basis for sensible changes to the plans. Lack of formal planning can result in unbridled changes, which are almost impossible to scale. A detailed plan itself may not change — but the process of creating and maintaining that plan provides an excellent benchmark for change. DAD shifts the focus of the organization from a single process to a tailorable and Agile suite of processes. This notion of a tailorable process stream comes in very handy when dealing with regularly changing business processes that are benchmarked around a “standard” process.

Adoption of CAMS at an organizational level is effectively a business transformation exercise. Figure 4 shows the eight most commonly identifiable business focus areas that need to be considered in undertaking successful Agile business transformation. While these focus areas can change depending on the type, nature, goals, and complexity of the organization undergoing transformation, they still provide a good understanding of the business structures and the way they can be transformed to Agile.

![Figure 4 — DAD assisting in CAMS adoption.](image-url)
The concepts of Disciplined Agile Delivery potentially support each of these areas of work in a CAMS adoption scenario. When DAD teams make people, process, and tool decisions, they keep the following five selection factors in mind:

1. Team skills
2. Team culture
3. Organizational culture
4. Nature of the problem
5. Business constraints

The most relevant DAD scaling and selection factors are shown on the top in Figure 4. The organizational areas are shown in the bottom half of Figure 4, and the way in which DAD assists the organizational adoption is summarized in Table 1.

Figure 5 shows the six major elements in planning and executing CAMS organizational adoption and the way in which some of the concepts from DAD can help improve that adoption:

1. The CAMS business case is an independent pre-adoption activity that considers ROI based on business

<table>
<thead>
<tr>
<th>Business Areas in an Organization</th>
<th>CAMS Adoption</th>
<th>Importance and Relevance of DAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business model</td>
<td>CAMS elevates the impact of agility to the business-model level, requiring the organization to revisit the priorities and goals of the business.</td>
<td>DAD encourages the organization to consider the business constraints in revising the business model.</td>
</tr>
<tr>
<td>Product and service portfolio</td>
<td>Understanding the portfolio of products and services provides the basis for ascertaining the risks associated with the delivery of these products and services after CAMS adoption.</td>
<td>DAD promotes a focus on the “nature of the problem” in the use of a method, and that problem changes depending on the organization’s offerings.</td>
</tr>
<tr>
<td>Customers and partners</td>
<td>This business area describes the external parties that interact with the business. The manner in which these interactions will change needs to be modeled and studied for successful adoption of CAMS.</td>
<td>DAD encourages attention to both internal and external factors in projects; the same concept at the organizational level enables the study of external parties.</td>
</tr>
<tr>
<td>ICT systems (applications, and databases)</td>
<td>This area covers the software systems and technologies that support the business; all the systems and databases need to be revisited and upgraded to support a business based on composite Agile.</td>
<td>The DAD framework fosters an awareness of domain and technical complexities, which is necessary for understanding and integrating ICT systems.</td>
</tr>
<tr>
<td>Operational organization</td>
<td>The internal parties, such as employees, management, and their reporting hierarchies, all undergo change as the business bases itself on CAMS.</td>
<td>DAD’s attention to team and organizational culture eases the changes in the way people work in a project or an organization.</td>
</tr>
<tr>
<td>Business processes</td>
<td>Business processes undergo reengineering with changes to the process maps, output, and values as based on CAMS.</td>
<td>DAD’s geographic and organizational distribution factors provide a basis for reengineering globally distributed business processes.</td>
</tr>
<tr>
<td>Networks and infrastructure</td>
<td>This business area focuses on the changes to the underlying communications technologies to facilitate CAMS. Mobile networks can increase business agility through location independence, but they demand higher security.</td>
<td>The technical complexity and “nature of the problem” factors draw attention in the DAD framework, and both are invaluable when the technical infrastructure undergoes change.</td>
</tr>
<tr>
<td>Regulatory</td>
<td>This area deals with legal, accounting, and financial aspects of the business that may have special needs in terms of audits and traceability as the business goes Agile.</td>
<td>DAD’s consideration of regulatory compliance facilitates the acceptance of agility by the business because it provides for necessary governance.</td>
</tr>
</tbody>
</table>
constraints. This business case includes the costs, risks, and benefits of using CAMS across the organization.

2. Organizational culture and associated project culture — although discussed in context frameworks — is further underscored by DAD. This cultural context is immensely helpful in understanding potential areas of resistance to CAMS.

3. Configuration of tools that support CAMS will depend on team skills and corresponding team culture. Tools can further help in putting together metrics at both the project and organizational levels.

4. CAMS deployment and training will be based on the organization’s ability to accept and adopt Agile practices. Again, DAD’s emphasis on organizational and team culture enables practical, hands-on coaching of CAMS concepts, particularly the avoidance of method friction.

5. The project output and its value to the business are directly based on a disciplined approach to delivering that output or solution. Here DAD emphasizes the overall solution (more than just software), as well as its value to the business.

6. Organizational agility is considered an ongoing activity rather than a project-based activity that changes and then embeds itself in the organizational culture.

CONCLUSION

Organizations need to consider the value Agile offers to business, far beyond that originally anticipated by the proponents of Agile in the software domain. When adopting Agile at the organizational level, organizations must make sure that different methods in use within the enterprise — each spearheaded by a different role — don’t cause method friction, which can reduce quality and flexibility. Concepts from DAD, such as its focus on a solution (rather than software) and on scalability, provide an excellent basis for adopting a composite approach to Agile such as CAMS. Further experimentation (in a pilot project scenario) using DAD to adopt CAMS will provide valuable lessons that are organization-specific.

ENDNOTES


Disciplined Agile Delivery (www.disciplinedagiledelivery.com).


Unhelkar (see 2).

Unhelkar (see 2).

While the original Agile Manifesto uses the word “software,” the Disciplined Agile Manifesto talks about a solution (which may not be always a software one, especially if Agile is applied across the entire organization).


While the original Agile Manifesto uses the word “software,” the Disciplined Agile Manifesto talks about a solution (which may not be always a software one, especially if Agile is applied across the entire organization).


Such as the the Software Development Context Framework (SDCF) (http://disciplinedagiledelivery.wordpress.com/2013/03/15/sdcf).

Unhelkar (see 6).

Bhuvan Unhelkar is a Senior Consultant with Cutter Consortium’s Business Technology Strategies practice. He has over two decades of strategic and hands-on professional experience in the ICT industry. As founding consultant of MethodScience.com and Director of Competency Standards at the Australian Institute of Business Analysis (AIBA), Dr. Unhelkar exhibits notable practical consulting and training expertise in business analysis, software engineering (UML modeling, processes, and quality), enterprise architecture (SOA and Web services), project management, collaborative business, mobile business, and environmental/sustainability strategies for business. He has published 17 books, including his latest, entitled *The Art of Agile Practice: A Composite Approach for Projects and Organizations*. Dr. Unhelkar is a Fellow of the Australian Computer Society, life member of Computer Society of India, past President of his Rotary Club at St Ives in Sydney, and a previous TiE Mentor. He can be reached at bunhelkar@cutter.com.
Integrating Lean Thinking to Achieve Multi-Product and Multi-Team Agility
by Al Shalloway

Software development and IT shops around the world are embracing Agile methods. For teams and smaller organizations, the results have been impressive. And yet for larger IT organizations and software development organizations, success has been elusive. Rather than productivity and profit, they experience mediocre results or worse. Why is this?

It is a question that some within the Agile community are now starting to address. A big part of the issue is that, at the level of the enterprise, Agile has lacked a solid systems perspective. It is not enough to try to build up team by team, with management trying to remove impediments along the way; instead, it requires an integration of business stakeholders, middle management, and effective teams. It requires both systems thinking and the discipline that systems thinking demands.

This article describes three key principles of Lean Thinking for software development. It applies these to the value stream (the name Lean gives the workflow from “concept” to “consumption”). It also describes three disciplines Lean-Agile teams will need to follow to keep value flowing. Finally, it illustrates how Lean Thinking guides Agile enterprises in addressing challenges in their context. Lean-Agile lays out a different, more disciplined approach for scaling Agile.

THE DISCIPLINE OF LEAN-AGILE

“Disciplined Agile” may sound like an oxymoron and has certainly been controversial for some in the Agile community, but it is essential for sustained success. Discipline does not mean “heavy-handed” — we all know that too much management, overplanning, overdesign, and overly large projects are not effective. However, undisciplined teams that use Agile as a justification to avoid doing what is necessary are also not effective … and, by the way, are not Agile.

While discipline in Agile is required to improve performance at the team level, it is even more essential at the enterprise level. It provides a pathway for scale and sustainability.

Instilling discipline and coordination across Agile teams requires an extension to classic Agile and its team-centric point of view. It requires extending Agile with Lean principles and practices. My colleagues and I call this extended Agile mindset “Lean-Agile.”

Consider the classic approach to Agile across teams. You start Agile with one team, then add another and another and coordinate them with a “team of teams.” The problem is that coordinating teams with a team of teams approach tends to not work the more teams there are. Getting a few team members working together within one team is considerably different than getting a few teams working together. The reasons for this are intrinsic to the differences between intrateam dynamics and interteam dynamics. Cross-team dynamics are deceptively difficult.

Lean-Agile takes a different approach. Lean-Agile uses systems thinking. Its focus is on the incremental delivery of business value by attending to the entire value stream as one system. It says that management needs to provide the big picture while teams implement within that context. Lean-Agile will suggest changes to the workflow to remove delays in receiving feedback, detecting errors, using information, and ultimately, delivering value.

Let’s look at these key principles to see how they effectively extend Agile to work at scale.

Systems Thinking

Systems thinking is the process of understanding how parts of a system influence one another within a whole. In software development, these parts are product selection, product prioritization, requirements, architecture, design, code, test, quality assurance, delivery, integration, management, HR, and more. Systems thinking suggests we look at the development workflow in its entirety, seeing how one part affects the others. Local
optimizations without this view may have little — or even a harmful — impact on the whole.

Lean offers a particular way to look at the system: the value stream. The value stream is the flow of work from when an idea is first conceived through implementation, deployment, and eventual use. The time from start to finish is called “cycle time.” Lean Thinking says that actions that shorten cycle time are usually good and those that lengthen cycle time are probably not. Lean Thinking looks for ways to remove delays, which also results in eliminating unnecessary work. This leads to improved quality and lowers costs.

When multiple teams and products are involved, this holistic approach means looking at the entire book of work. For example, it may be that delaying one project is worthwhile if another project can deliver value greater than the cost of delaying the first project. This gives us helpful questions for deciding whether to start a new project: “Will this new project add to the value being delivered?” and “Will adding this project slow down or negatively impact the ability of existing projects to deliver value?”

Management Oversight (Creating the Context for the Teams’ Work)

Having “self-organizing teams” is a very good Agile practice. It works at the team level because teams are able to apply their own local knowledge to adapt to their conditions. Lean says that, at scale, this is not sufficient and that a broader perspective is also required. Management provides this perspective by creating the context within which teams work together. When several teams are involved, a lack of management oversight inevitably leads to a lack of team coordination, which leads to wasted effort.

This enterprise awareness helps at the project level, too. Sometimes teams are able to continue providing value on their project, but other projects provide opportunity for greater value. The decision to switch to the other project must come from someone with eyes on the whole portfolio. Management provides this oversight to help a team stop its current project, at an appropriate place, in favor of a more valuable one.

Another area where management oversight helps is in continuous improvement. Lean Thinking suggests that management plays a crucial role in improvement: this includes both improving the environment in which teams work and acting as a coach, often by questioning the teams about their process and helping them think about ways to improve. This is management via leadership and coaching.

Removing Delays

Removing delays is a central tenet of Lean. In manufacturing, they call this “just-in-time” — making things just before they are needed. In software development, it means not doing work before it should be done. For example, don’t create requirements until you are ready to work on them, nor write code until you are ready to test it.

All Agile methods work well in reducing delays in the workflow by advocating small batch sizes, shortening feedback loops, and avoiding work before it is needed. This removes delays, which is essential because of the extra work these delays create. For example, consider how delays between coding and testing affect developers. It takes relatively little time for developers to fix bugs they are immediately told about compared to bugs they discover weeks later. Testing up front is a Lean-Agile way to remove delays.

Putting Them Together

Lean-Agile puts these three principles together — systems thinking, management creating the context for the teams, and removing delays — to provide a solid and proven approach to enterprise agility. This requires honoring:

- Your understanding of the best way to work on things when working alone
- The agreements between different roles as to how they are to work together
- The decisions of those who are in a better position to make them

LEAN-AGILE AND THE VALUE STREAM

Multi-product, multi-team agility can be achieved by attending to flow and ensuring everyone agrees on who makes certain decisions at certain places in the value stream. Before going into how this works, let’s first look at the reason for focusing on the value stream.

Most companies are organized hierarchically even though work flows laterally across the organization (see Figure 1). People are managed vertically; value flows horizontally.1
Utilization vs. Flow of Value

The problem is that managers will manage what they can see. In this case, they are going to manage employee workload, productivity, and quality of work. While this sounds reasonable, it is more important to manage the time to market of what is being built, the effects of upstream groups on their staff, and the effects their staff have on downstream groups. Focusing on people rather than on the workflow results in a lack of cooperation.

Consider the workflow for a project illustrated in Figure 2. Someone starts on the project only to hand it off to someone or to wait for someone for information. Although people are always busy, work starts and stops. There is activity on a work item, and there is waiting until someone can work on it (see Figure 3).

Time to Market

The important question to ask is, “What percentage of time is spent working on the item and what percentage is spent waiting?” The numbers may surprise you. In traditional organizations, people may work on four to six projects at a time. This suggests that, on average, they cannot devote more than 20% of their time to any one project. This in turn means that, on average, 80% of the time spent on any particular work item is devoted to waiting for someone to pick it up. And that is delay! Even though people are fully busy, value is not flowing smoothly and quickly through the value stream.

Does this number seem too large? If you are not tracking this, how would you know? In too many organizations, no one is managing this.

Lean Thinking tells us we are creating a problem due to an improper focus on employee utilization rather than on the flow of value. Yes, it is a good goal to make sure people are working. But it is counterproductive when doing so adds additional delays in the workflow. That is just what happens when you have too many projects going at once, even if people are fully busy.

We start with good intentions, allocating people to the most important projects. At some point, the ideal teams are set up. Then a new project comes along. If someone seems to have some “spare capacity,” they get assigned to the project even if this will require them to ask for help from key people who are already totally booked up. Key people are the ones who have essential subject matter expertise, certain skill sets (such as architecture), or happened to be around years ago when your now-legacy system was developed. Too often, these individuals are already overloaded. Then the new project starts, and they are asked for more help. Now they are seriously overutilized, and the delays in workflow start. And so it goes in an endless, downward cycle. The net result is that your best people are made to work in the most ineffective manner. This leads to …

Discipline #1: Stakeholders Cannot Start More Projects than the Development Organization Has the Capacity For

Stakeholders are required to identify and prioritize those projects that are most important, that provide the greatest value. Teams pull from this prioritized list when they are ready for work.
However, this is only one part of the big picture. The teams have to pull the work they are going to do in unison so that one team doesn’t finish its work, have to wait on another team to finish, and then thrash as the teams try to integrate their respective pieces. This leads to …

Discipline #2: Teams Capable of Delivering the Selected Value Must Work in Unison

There must be enough capacity to build the entire increment of work before pulling from the product queue.

Building and Integrating Across the Feature Level

Multiple teams working at the same time is good, but it is insufficient. This is often the case when companies are large and have component teams. Consider the situation shown in Figure 4. There are a hundred or so developers organized around two or more product lines. Call them Product Line A and Product Line B. Each of these applications has its own component team that develops shared functionality. There is also a component team that works across applications.

One solution would be to create cross-functional teams with people from each application, from the component team(s) for the application, and from the component team(s) that run across applications. This is a great approach if you can do it, but too often, it is just not possible.

Suppose Agile teams must work together (as shown in Figure 5) and over several sprints. This was the situation I saw at one client. These were highly functioning Scrum teams that were delivering their work well in each sprint and yet were not able to deliver the overall feature in an efficient manner. While this was confusing at first, a little value stream mapping helped to reveal what was happening.

Figure 6 shows how the work of a feature was spread out across these teams. Each team got a backlog consisting of their part of the feature. The teams would then take these backlogs and work at their own discretion. Figure 7 shows the first sprint. Note that the different shades on the backlog represent pieces dependent upon each other. While each team told the other teams what they were doing, little coordination actually took place.

What was happening was that each team chose stories from their backlog based on what would make them most efficient. After selecting what to work on, they coordinated with the others. But the damage had already been done. They were unable to do any significant integration after the first sprint because the completed pieces did not represent end-to-end functionality. This is shown in Figure 8.

And so it continued until the teams completed all the work for the feature (Figure 9). Now, they were finally able to integrate their work. The challenge was that there had been several sprints’ worth of work on these different parts of the system without any meaningful integration. It was almost a certainty that more work would have to be done. Thrashing ensued. Integration took longer than if they had been doing it on a continual basis. That is delay and waste.
They incurred another type of delay as well. Since they had not done integration, they were not able to show the functionality to the customer to get feedback. The feedback loop was much longer than the length of the sprint. And that is delay and waste.

To solve this problem, Lean Thinking says we want both short cycle time (the time from concept to consumption) and to have as few delays as possible along the way. This means we want quick feedback loops along the entire value stream.

Of course, achieving quick feedback is an Agile discipline. We strive to deliver working software at the end of every sprint. In the example above, while working software was delivered at the team (Agile) level, it was not being delivered at the enterprise (Lean) level. It is the team of teams that should be delivering working software every sprint. Therefore, instead of giving the teams their backlogs independently, the work should be divided such that the work done each sprint will deliver a piece of functionality that can be demonstrated to the customer. This is not an individual team decision, and it leads to …

Discipline #3: Teams Must Let Someone Who Sees the Bigger Picture Decide What They Should Be Working On

Taking this approach leads to a different value stream. Compare the improved value stream in Figures 10-12 with the original value stream in Figures 7-9. The new value stream enables demonstration of the software at a much quicker pace, resulting in shorter feedback cycles and less integration work. It also allows all the teams to have greater vision of what they are building.

Optimize the Whole (Macro-) Team

Does this mean we are just creating a bigger team? Perhaps. Instead of three teams that have to work together, there is now one larger team working together. The important thing to note is that it was Lean Thinking that guided us here to address a problem in this context. It was clear that building software that couldn’t be demonstrated was the problem.

Interestingly enough, the optimal solution would be to create three cross-functional teams from the teams being discussed. Each would then build a piece of functionality, but a single backlog would still be used to ensure that the teams were working together in the best way.

In making decisions on how to develop software, Lean’s mantra of “optimizing the whole while attending to flow” provides essential insights. In this case, instead of focusing on how each team works individually, we must consider the macro-team, the larger team actually building the software. How will it best develop software with the shortest feedback loops possible? It may seem surprising that competent teams couldn’t see this. We have found, however, that the focus on Scrum teams consistently obscures the bigger picture that Lean’s systems thinking naturally brings to the fore.
SUMMARY: THE PATTERN OF SUCCESS

Software development should be about delivering value quickly. It requires a relentless focus on removing delays, whether they come from too much work-in-progress, waiting, testing, or poor workflow. The larger the organization, the more teams that are involved, the more difficult this challenge becomes. Trying to scale up from individual teams to the enterprise rarely works. Lean Thinking is naturally focused on enterprise-level issues. Agile methods informed by Lean Thinking lead to the disciplined type of Agile required for the Agile enterprise. We must remember that Agile is not about team-iterative development, but rather about the incremental delivery of business value.

ENDNOTE

1Thanks to Dan North for this observation.

Al Shalloway is the founder and CEO of Net Objectives. With over 40 years' experience, Mr. Shalloway is an industry thought leader in Lean, Kanban, Scrum, and Agile design. He helps companies transition to Lean and Agile methods enterprise-wide and teaches courses in these areas. He has developed training and coaching methods for Lean-Agile that have helped his clients achieve long-term, sustainable productivity gains. Mr. Shalloway is the primary author of Design Patterns Explained: A New Perspective on Object-Oriented Design, Lean-Agile Pocket Guide for Scrum Teams, Lean-Agile Software Development: Achieving Enterprise Agility, and Essential Skills for the Agile Developer. He is a popular speaker at prestigious conferences worldwide, a cofounder and board member of the Lean Systems Society, and a certified SAFe Program Consultant. Mr. Shalloway has a master's degree in computer science from MIT and a master's degree in mathematics from Emory University. He can be reached at alshall@netobjectives.com.

Figure 10 — Work selected for first sprint to enable feedback on one slice of functionality.

Figure 11 — Work selected for second sprint to enable feedback of another slice of functionality.

Figure 12 — Work selected for third sprint to enable feedback of last slice of functionality.
A Disciplined Approach to Agile Project Deployments

by Matt Ganis and Sanjeev Kumar V Marimekala

Conventional Agile methods emphasize a number of teeming practices such as pair programming, planning sessions, and colocation of the team. One of the often-overlooked practices, however, is that of the whole team. An Agile whole team is one in which the entire team works as a unit of generalizing specialists to share responsibility for the production of high-quality software. Agile projects tend not to think in terms of “departments” or “sister groups.” Instead, they consider only a single group that consists of all the skills, knowledge, and tools needed to deliver a software project — from concept and planning to development, qualifying, and rollout.

In large enterprises where development teams have the opportunity to work with specialized IT deployment teams that support the development environment, the whole team approach can be a challenge. The IT organization is likely working with a number of internal organizations that use a number of methods, including Agile, waterfall, or other hybrid methods. In our experience, we’ve found that this IT deployment function doesn’t lend itself as effectively to Agile teams when compared to typical development or architectural roles, yet its contribution is just as critical to the proper functioning of an Agile team and its delivery of incremental functionality. The issue isn’t that the deployment staff don’t provide value; the issue is more that the value they provide comes later in the project’s lifecycle, and thus they don’t consider being part of an Agile team a very effective use of their time.

In this article, we present an Agile delivery model that enables IT deployment staff to be as nimble and flexible as possible while allowing them to serve the needs of both traditional waterfall projects and Agile product teams. Specifically, our solution enables the following techniques, which could be adopted in a disciplined Agile project deployment model:

- **An Agile and responsive IT delivery model.**
- **A project prioritization mechanism** through which priorities are set by the business owner (in a Scrum of Scrums model). Based on the project prioritization matrix, a score is assigned to each project, and the list is sorted on a first in, first-out basis and according to predefined rules.
- **A disciplined approach to new project deployment,** so as not to overwhelm an already overworked/overbooked IT staff.
- **Effective utilization of resources based on the project/business needs,** which requires keeping the project stakeholders updated on changes in project priority resulting from lack of information/insufficient requirements or faulty code. The penalty for these deficiencies could be a shift in project priority such that control of the project rollout (deployment date) lies with the deployment team rather than the development team.
- **Deployment pattern analysis** of an Agile project team based on the trends in the deployment reports.

**THE PROBLEMS WE FACED**

As the adoption of Agile methods grew within our organization, we began to experience problems that our IT department had to deal with. First, as projects matured, they would go through their sprints and frequently deploy into production while they were incrementally adding improvements. The problem was that as the number of projects increased, applications that were deemed not valuable enough by a stakeholder were never decommissioned; the development team had moved on to other projects and never planned for a cleanup process.

In our framework, once a project is rolled out, each stakeholder is assigned units of capacity within the server environment. This unit of capacity is used for calculating the cost and resources required for the project’s maintenance and support. Once assigned, the unit value is then evaluated quarterly to determine how the project has matured over time. The projects are further classified as to business-critical value (BCV), which is measured on a scale of 1 to 5. A project with a BCV value of 1 might require a smaller number of middleware components,
Early on, we realized that many of our enterprise project deployments but also traditional deployments. This constant rollout, coupled with the need to maintain optimal operational support for existing deployed applications, could overload the IT staff.

The second problem was the need for specialized skills across a number of teams. While Agile teams like to believe that they can assign individuals to user stories related to something outside of their knowledge base, or that these generalists can learn by pairing with an expert, we have found this to be more problematic than true when it comes to highly specialized skills such as database administration or architecture. What we have found to be more productive is to have teams do their own database work and then have it reviewed by the lead DBA and possibly refactored based on his or her recommendations. Since virtually every project has a database requirement, this has the advantage of preventing the DBA from being a “blocker” in any given sprint, while still best servicing all the applications under construction. The issue then becomes which team gets the “time slice” for a specialized skill first. Project prioritization thus becomes a key step in the process, as it allows for sprints to be planned properly.

**OUR DISCIPLINED AGILE FRAMEWORK AND PRACTICES**

Using the Disciplined Agile Delivery (DAD) framework as a model, we’ve created a process within our organization that supports not only a large number of Agile project deployments but also traditional deployments. Early on, we realized that many of our enterprise processes wouldn’t be eliminated with the advent of an Agile approach. Many (if not all) of our projects, while claiming to be Agile, needed to collaborate with other teams/resources such as IT deployment (data center teams), architecture teams, database administrators, and others. While our Agile teams were creating usable software at a steady drumbeat, we struggled with the inability (or lack) of IT staff to deploy applications to the network where they could begin realizing value.

While our Agile teams were creating usable software at a steady drumbeat, we struggled with the inability (or lack) of IT staff to deploy applications to the network where they could begin realizing value.

The overall business sponsor is responsible for setting up the direction and priorities for the deployment team. This means assigning project priorities (high = most business critical, to be done above all other projects, and low = less business critical, higher-priority projects may override it). This is a key step in determining which of the projects get a slice of time from the deployment team, since projects can be half-deployed to the stage environment.

If the presented project has sufficiently high business priority and is deemed “complex” (W105-Complex Application Rollout Process), the members of the deployment team participate as regular members of the Agile team, taking on all aspects of deploying the project. Other projects that are simple fall into W104-Simple Application Rollout Process (i.e., code upgrades as opposed to new deployments). In such cases, the deployment team is allowed to determine its own priorities based on resource availability and inter-project dependencies.

In order to achieve this streamlined approach to the deployment of software applications, we have developed several practices (highlighted below) and a supporting process (see Figure 1).
**Practice 1: Establish Open Communication Channels**

As with any Agile project, frequent and detailed conversations are a key element for success. Since we are proceeding under the premise that the deployment team is a vital but separate component of the individual teams, it is imperative that we establish effective lines of communication. One good way to maintain open communication was to establish a weekly Scrum teleconference between the deployment team and the various Agile software development groups. (As none of our teams are collocated, the primary method of communication is through teleconference calls.) This allows any previously unknown blockers to be identified and discussed in an open environment.

Once an Agile project has been initiated, we expect that the individual project teams will hold their standard daily Scrum meetings. However, any deployment issues that arise (such as scheduling conflicts, escalations, and/or postponements) will be discussed in this “Scrum of Scrums” call with the deployment team. In this way, we track all our projects on regular intervals to see if there are any roadblocks or potential...
conflicts/issues. If there are, then these are addressed in order of priority and business value.

**Practice 2: Focus on Reducing Waste (Time/Effort)**

One way we reduce wasted time is to have members of the deployment team attend the development team’s daily Scrum meetings only when it is required. In fact, we look to engage the various scarce resources only when these resources are required to provide an update on the progress of a specific problem. While this may sound harsh to Agile true believers, the fact is that a database administrator simply does not need to be involved in a discussion that has to do with such unrelated subjects as final deployment dates or user experience issues. This is not to say that DBAs could not bring valuable insights into other projects, but when we looked at where their time is best spent, we found that it’s in working on database issues/concerns/installs rather than participating in other projects where they have little or no involvement. It’s a tradeoff we chose to make in order to fully utilize their scarce skills.

Another way to look at waste is to limit the number of deployed projects/applications. Each deployment “costs” the deployment team in maintenance of the servers, security audits, patching of operating systems, and so on. One way we limit the number of servers that must be maintained is to instantiate a “limiting number” of applications that can be deployed within the infrastructure (based on the sponsoring business owner). At the beginning of the year, the various business owners purchase (fund) the deployment team for a number of support “units.” Units are simply an abstraction of the amount of resources (machine and human) needed to support a given application. For example, a single Web server with a relatively simple database may cost 4 units for the year, while a complex environment of 9 or 10 servers with a complex code base and infrastructure requirements would cost more; say, 30-40 units.

Our objective is to prevent the bloat of too many applications while still allowing the various teams to be innovative. Think of a piece of graph paper. The business owners purchase a specific number of units (i.e., squares on the piece of graph paper). As they deploy applications into this environment, units that are “used up” are shown as colored blocks (see Figure 2). When all the blocks are full, no more applications can be deployed unless the owners choose to remove something — thereby freeing up some units — or move their applications to other hosting environments. Moving applications to a “standard” hosting environment is an attractive solution when the project has stopped iterating and no longer has frequent deployment requirements.

**Practice 3: Ensure Best Utilization of Resources from a Business Perspective**

Our model assumes that there is a scarce resource that is needed across a number of projects in our portfolio, namely the deployment team. At any point in time, we have a large number of projects looking to deploy new releases or make modifications in an effort to meet their stakeholders’ needs. With all those changes flowing through a single group, we are bound to encounter issues and slowdowns.

While many would argue that the answer is simply to staff each team with its own deployment component operating independently, we’ve found that this is not only a costly solution, but also risky in a shared infrastructure environment where increased team size could lead to communication breakdowns or confusion among projects. To best satisfy all these teams, we have an overall priority-setting mechanism in place. Think of this as a Scrum of Scrums meeting where the interlocking of teams and schedules takes place. Our business sponsor (the overall owner of all projects) sets priorities based on the business needs. The priority scheduling comes in the form of a list that ranks projects from the one that should get the top amount of resources down to the one that should get the least for a given time period. Once produced, this list of priorities is then carried out by members of the deployment team, but they are also given some flexibility to reprioritize within that list. That is to say, if they have a resource available to reconfigure a database, and no higher-priority project needs one, the team can service the lower-priority need without any impact to agreed-upon schedules (see **W103-Priority Influencers** in Figure 1).
Practice 4: Apply Resources Where Needed/Required (Effective Utilization of Team Skills)

As the number of project deployment requests increases, there is always a greater demand for resources with the specialized skills required. In order to meet the demand of the Agile teams that need these skills, the ScrumMaster of the deployment team plays the role of “maestro,” always guiding the deployment team members as to when to get involved in a project and what their role is.

If the development teams do not adhere to their lead times, then the commitments for other project rollout dates could be in jeopardy.

At times, the deployment ScrumMaster also takes ownership of the problems that arise and drives the effort to get them resolved with the help of resources with specialized skills. This also helps to reduce the deployment team's time and effort while avoiding distraction. What we have found is that when resources are working on several projects, their productivity increases, provided they can focus on their tasks (which may involve their participation in several Agile development Scrums). The ScrumMaster makes every effort to balance the needs of the project by taking constraints such as scope, time, and cost into consideration while reprioritizing the deployment team’s tasks if needed. This management of what we call “local priorities” (see W103.2 in Figure 1) is truly an example of trust between the deployment team and the Agile development teams. The development teams understand that when deployment is invoked within a Scrum at the proper time, they receive full commitment not only from the resource invoked, but the full deployment team.

Practice 5: Maintain Proper Lead Time for Each Project

The lead time is the specific amount of time needed before “execution” of a project and is something that the development teams must strictly adhere to. As shown in Figure 1, we derive this lead time as part of W102-Project Analysis and gain team agreement in W102.4.

The lead time helps the project team eliminate all the known unknowns that may arise during the project’s deployment while helping to keep the schedule intact. In a shared infrastructure environment, this is a crucial component if we are to keep a steady stream of application deployments flowing through the deployment team. For example, lead times could include such tasks as: Provide the database schema seven days prior to the expected deployment date. This enables the deployment team to properly plan its time and commitments and possibly adjust its priorities (W103.2), but it puts the onus on the Agile teams to deliver on time. If the development teams do not adhere to their lead times, then the commitments for other project rollout dates could be in jeopardy. Indeed, if the lead-time activity is not delivered within a reasonable period, the deployment slot may be forfeited for a given sprint so that other projects aren’t affected. Of course, escalations and overall business priorities can be adjusted (W103.1) depending on the criticality of the project.

Practice 6: Keep It Simple

If the project is complex in nature (i.e., if the number of components is large or the level of software is fairly complex), it can become very difficult to manage in terms of timelines for deployment, dependencies, and communication. In this case (W105-Complex Application Rollout Process), as in any good Agile project, we look to break down infrastructure requirements into subcomponents/projects where the project team members are grouped and assigned subcomponent responsibility based on the required skills. Here, our deployment team would act as full-time member of an Agile project team (a traditional approach). Notice that components are broken out, so we can bring the correct resource into the team, and that stories are created specifically for the necessary infrastructure components (W105.3).

RESULTS

By applying the above practices, we have catered to the business needs of Agile software development projects with a limited and highly skilled resource. As our process matured through the adoption of these practices, the number of deployments our teams could perform increased over time with the same number of resources (see Figure 3).

We can also see in Figure 3 that the number of deployments in both our stage and production environments increased. The increased number of deployments in the stage environment is important, as it allows our Agile...
teams to take advantage of a preproduction environment to perform integration and validation that they could not have otherwise achieved due to environmental constraints or complex integration requirements. By leveraging the use of this stage environment before promoting the code to production, our Agile development teams are very confident in the final product/application deployment. Through the use of these practices, we have achieved a rate of successful Agile project deployments that was not previously possible. We believe these practices provide a structured approach that can be used by other organizations in which specific highly skilled individuals may not be readily available.

ENDNOTES


The Agile Manifesto states that the Agile movement values “individuals and interactions over processes and tools.” However, very little is discussed about the personality aspect of people and how important it is to have the right personality types in the right roles. Failure to do so may have a major impact on the success or failure of Agile projects.

Personality types play a major role in determining the career paths of individuals, and they are widely used at many universities and career guidance organizations internationally to guide students’ course and career choices. That is because people are most likely to excel at careers suited to their personality type. The role personality plays in the workplace has also been studied at length, and is equally critical when it comes to the types of personalities that make up a work group or team. Often a “dream team” is created and fails mainly due to the wrong mix of personalities, particularly when everyone is trying to lead the group.

With Agile development, it becomes even more important to have the right people in the right roles, as all team members are required to be self-organizing and cross-functional and to take responsibility for the success of the project. Personality types are starting to be recognized as an important factor in the success of Agile projects, and the first studies and books on the topic are making their way into the Agile world.1-3

This is not only about training and skilling people up to carry out their roles. Rather, some individuals are simply not suited to certain roles due to their personality profile. How often have companies promoted a brilliant technical person to a managerial role in order to move him or her onto a higher pay grade, for example, and subsequently ended up losing a great technical resource and gaining a useless manager?

That there are many different personality types has been long established, and many psychometric theories have been developed to measure psychological preferences in how people perceive the world and make decisions, such as the Myers-Briggs Type Indicator4 developed from Carl Jung’s personality types, John L. Holland’s RIASEC personality test,5 and the DISC personality test,6 to name just a few.

To illustrate how important the right personality types are, think of Jung’s basic conception of introverts and extroverts. According to Jung’s theories, “an introvert is a person whose interest is generally directed inward toward his own feelings and thoughts, in contrast to an extrovert, whose attention is directed toward other people and the outside world.”7 On this basis, we would hesitate to appoint an introverted individual as a salesperson, as he or she would hate that role. By choice, introverts get strength from within, like to be left alone to solve problems, and are reserved with strangers. For the same reason, we would hesitate to make an extrovert a programmer, as he or she would probably feel stifled in the role.

THE MORE THINGS CHANGE, THE MORE THEY STAY THE SAME

How does this insight apply in the Agile world? One of the main problems experienced today with software projects is an age-old one: poor requirements, which lead to project failure. Projects get cancelled, go over budget and schedule, and/or exhibit poor quality.

In the early days, computer scientists would obtain the requirements from the business and would deliver software that might have met the specifications gathered but did not actually meet the business’s requirements. IT professionals and businesspeople would often misunderstand each other, as they came from completely different perspectives and did not grasp each other’s frame of reference. These misunderstandings led to project churn and necessitated rework so that the correct (or previously missing) requirements could be implemented. In the process, the business’s cost and time requirements were grossly overrun. At the same time, budgets and timetables were squeezed by everyone in the chain until they were unrealistically low. In order to finish on time and on budget, requirements were then prioritized and items were dropped at the last moment to get the release out.
Several steps were taken along the way to try to address these problems:

- Business analysts were introduced to bridge the gap.
- Project management was introduced, as well as project management and software development processes and methodologies, to help ensure that projects come in on time and budget.
- The industry adopted an evolving set of development processes, from structured programming to waterfall, iterative development (e.g., Rational Unified Process [RUP]), Unified Process, OpenUp, Agile (e.g., Scrum), Lean, and others, on up to the Disciplined Agile Delivery (DAD) framework.

While these developments improved the overall success of software development projects, they have not gone far enough. According to a 2011 survey by Guest Editor and Cutter Senior Consultant Scott Ambler, we are not where we can be; only 69% of iterative projects, 67% of Agile projects, and 50% of traditional projects are successful.8 Clearly we haven’t fully solved our problem yet.

PEOPLE FACTORS IN SUCCESSFUL SOFTWARE DEVELOPMENT

What has also not changed is human nature and people’s personalities. Problems are mostly caused by people. Thus, if one can improve the people factor, it stands to reason that the success of projects will increase.

In my experience, the three important factors that contribute to making people more successful in software development are:

1. The culture of the organization
2. The correct use of development processes
3. Having the right people in the right roles

The Culture of the Organization

We all acknowledge that people are our most important asset. If that is so, surely we need to create the right culture and environment to support and nurture them. Such an environment stems from the leadership of the organization. Most highly productive teams have the kind of leaders who make it a point to do this.

Year in and year out, the companies that are voted as the best places to work or the most admired are the companies that understand the importance of an enabling culture and also happen to be very successful at creating one. Back in 1983, the first software company I joined as an executive, Systems Programming Limited (SPL), happened to be the largest software development company in South Africa. It had a few hundred developers and specialized in developing bespoke software for large organizations in South Africa. SPL was quite unique and had a culture that was revolutionary for its time:

- Status was not key. Everyone in the company — from the messengers to the CEO — had the same desk and chair.
- No memos were allowed. People had to communicate personally with each other.
- People could buy things they needed without a formal process as long as a purchase met one criterion: would they have bought it if it were their own company?
- Rules where kept to a minimum. Why introduce rules that affected everyone because one or two people did the wrong thing?
- Everyone had to be tested by an industrial psychologist to determine their personality type and fit to their role, as well as to the company culture.
- Even though there was a job description for each role (the box), the stated intention was to fit the box around the individual.
- Over 50% of the staff were female, and the company provided a fully staffed daycare center to accommodate their childcare needs.
- Over 30% of the executives were female.

It was an extremely successful company and a great place to work. It also had the lowest attrition rate in the industry. One of the main reasons for its success was that it placed the right people in the right roles. This enabled management to get the best out of people by knowing how each could be managed most effectively. I have always tried to emulate these principles in my own companies.

The Correct Use of Development Processes

As Ambler’s survey showed, iterative development (e.g., RUP) has improved the success of software development over the traditional waterfall process. But a common problem arose after the successful implementation of pilot projects using iterative processes. During the subsequent rollout of these processes, they were suboptimized and got a bad reputation through no fault of their own because:
People ended up filling in forms (artifacts) without understanding the goals behind the paperwork, thus defeating the purpose of the process. They went as far as looking at previous project documentation and simply altering the project details for new projects.

Organizations added additional process steps and documentation to the original lightweight iterative process in order to integrate it with existing non-Agile corporate project management strategies based on the likes of PMBOK or Prince. This added unnecessary bureaucracy to the process, resulting in projects taking much longer to complete.

It is far better to focus on project goals than follow a process for the sake of a process.

That’s where Disciplined Agile Delivery comes in. DAD offers ways to provide non-bureaucratic governance, such as built-in risk mitigation and proving the architecture in the early part of the project. DAD requires you to understand and address the goals of the process rather than focusing on paperwork. This is a great improvement in terms of being able to do the right things to have a successful project. It is far better to focus on project goals than follow a process for the sake of a process.

When helping customers to implement RUP and avoid turning a simple Agile process into a bureaucratic nightmare, I used to try to get them to understand what I call the philosophies behind the process, which DAD describes as the goals. Understanding the goals behind each step of the process framework establishes a consistent way of doing things in an organization and, done properly, identifies risks and problems early. It also allows people to change projects with ease and be highly productive in the shortest possible time. The goals further enable a project team to tailor the process so as to address only these goals and not waste time, money, and resources filling in unnecessary documentation and doing unproductive work. This goal-oriented approach to using RUP has enabled me to deliver large projects successfully and even come in earlier and at a lower cost than the estimated schedule and budget.

Having the Right People in the Right Roles

Having the right people in the right roles will improve the success rate of projects. I believe that the lack of understanding of the importance of doing this is one of the reasons that cause Agile teams to fail. When implementing Agile strategies, it is important to make sure that coaches, mentors, and team leads do not assume that all people will behave and act like themselves, thereby biasing the Agile implementation toward what works best for them. By understanding the different personality types, Agile leaders can help guide their teams a lot more successfully and ensure that the right resources are on each team.

In my more than 30 years’ experience, I have seen how important it is to put the right personality types into the right roles. When I joined SPL in the early 1980s, I went through a battery of personality tests, as did all new hires. Even though people were hired for a specific job, the company would rather resolve any mismatches by changing the job description to fit people’s strengths and thus utilize their skills more effectively and increase their job satisfaction. Personality profiles also enabled the effective management of individuals, as one knew what their strengths and weaknesses were.

I have subsequently used personality profiling for all hires in the companies I owned or worked for. This contributed greatly to my project successes. In the early days of object orientation, I had some of the best OO programmers working for my company, and they were continually being head-hunted at double their salary or more. It became a real business challenge to match these offers and retain these employees.

I developed an idea to skill up people as the auditing profession did. We used the money that we would have used to match salaries to fund the recruitment of other top IT graduates and to fast-track them into becoming productive ASAP. We put these new recruits through a six-week boot camp and established a three-year training and growth plan with certain milestones that needed to be achieved. Achieving these milestones, which included relevant industry certifications, resulted in increases in salary for each individual. Thus, our hires could earn salaries way above the market. The speed of achievement was up to them.

What made this program special was that we selected a broad range of personality types that could spread across all the roles within a team. In the boot camp, we split the group into teams and get each team to obtain the requirements from the same set of users or product owners to create a solution. The product owners were asked not to volunteer any information unless it was asked for. The teams would present their designs for the system to be built in the first few days of the boot camp. It was no surprise that each team’s presentation for the solution was vastly different from the others.
The teams would then be combined to build the system using RUP. They were left to self-organize and build the system. People gravitated to the roles that we had thought best suited them based on their personalities, and at the end of the project, we did a 360-degree review. Inevitably, every member agreed that they were most effective in the roles that they had chosen. These boot-camp graduates were then trained up according to their chosen specialty, which resulted in people who were excellent at their work. Furthermore, the retention rate was superb, as people not only got to do the things they loved, but also were encouraged and rewarded to learn and become experts in their fields. This approach also spurred the growth of the business, as the graduates moved into more senior roles earlier than they would otherwise have been able to do.

WHEN TEAMS CAN'T SELF-ORGANIZE

The “Principles Behind the Agile Manifesto” states that “the best architectures, requirements, and designs emerge from self-organizing teams.”

Sometimes teams will work well together and be able to self-organize and succeed, but sometimes they will fail. The group dynamics will fail. The self-organization will fail. This can apply even to a carefully selected dream team of experts. Assuming that a team’s knowledge about Agile is adequate, the cause of this failure could very well be poor group dynamics due to the wrong mix of personalities.

In his five tips for assembling successful DAD teams, Mark Lines states:

> Be wary of self-organizing teams, especially if your team lacks experience…. A lack of skills and experience can lead to chaos and an undisciplined team environment. There’s also a personality component. Some team members aren’t comfortable deciding what tasks they should be performing or in what order. Some might actually prefer to be directed in a more traditional manner. If you’re a team lead, you must be sensitive to these attitudes and habits and understand when somebody needs to be nudged or directed regarding their daily activities.⁹

In their book *Individuals and Interactions: An Agile Guide*,¹⁰ Ken Howard and Barry Rogers use the DISC personality profiling method shown in Table 1. They state that “the best teams comprise a blend of behavioral profiles. You want team members to consist of a combination of D’s, who will make quicker decisions and help drive the project; C’s, who will focus on the details as they analyze requirements, test, and write code; S’s, who will bring harmony to the team; and I’s, who will keep the communication going, optimism high, and energy flowing.”

 Auckland University of Technology researchers Sherlock Licorish, Anne Philpott, and Stephen G. MacDonell echo this conclusion in their study of Agile team composition, which probed the influence of personality types on team performance.¹¹ The authors found that the higher the team heterogeneity, the higher its performance. Moreover, teams with balanced personality types performed even better than heterogeneous teams.

<table>
<thead>
<tr>
<th>D</th>
<th>Dominance</th>
<th>Emphasis on shaping the environment by overcoming opposition to accomplish results</th>
<th>Priorities</th>
<th>Getting immediate results</th>
<th>Taking action</th>
<th>Challenging self and others</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Influence</td>
<td>Emphasis on shaping the environment by influencing or persuading others</td>
<td><strong>Motivated by</strong></td>
<td>Social recognition</td>
<td>Disapproval</td>
<td>Loss of influence</td>
</tr>
<tr>
<td></td>
<td>Steadiness</td>
<td>Emphasis on cooperating with others within existing circumstances to carry out the task</td>
<td><strong>You will notice</strong></td>
<td>Patience</td>
<td>Team person</td>
<td>Calm approach</td>
</tr>
<tr>
<td></td>
<td>Conscientiousness</td>
<td>Emphasis on working conscientiously within existing circumstances to ensure quality and accuracy</td>
<td><strong>Fears</strong></td>
<td>Social criticism</td>
<td>Slipshod methods</td>
<td>Being wrong</td>
</tr>
</tbody>
</table>
Agile teams where collaboration is necessary required this balanced mix even more.

WANTED: MULTIFACETED TEAMS

Agilists agree that Agile team members need to be trained or up-skilled to be multifunctional individuals. While the thinking behind this is sound, in reality the problem — especially with more advanced skills — is that even if people have had all the training and up-skilling required for a specific role, they might never excel at their job, or may even fail, if their personality is not suitable for that role.

We know, for example, that some people make good requirements gatherers and some don’t. Sending an introvert to gather requirements will probably result in getting the bare minimum. Introverts typically can’t wait to finish a meeting and will fill in any missing information not by asking the user, but by solving the problem themselves.

In contrast, providing the right mix of skills according to the right mix of personalities will allow team members to excel and will result in a greater number of successful projects. Again, it’s all about human nature. We know that software development is a team sport, and much like football, baseball, soccer, and so on, most roles are specialized and few are generalist. That is why DAD calls for team members that are “T skilled,” having deep knowledge and expertise in their specialization along with a general knowledge of — and ability to collaborate with — the other roles in the team.

The Agile Manifesto puts “people and interactions” first, since the root of successful software development is people. But the success of software projects is not only about skill — it’s about getting the people factors right.

ENDNOTES

10Howard and Rogers (see 2).
11Licroish et al (see 3).

An entrepreneur with more than 30 years’ experience in the software industry, Themistocleous has worked for and/or owned companies with up to 500 software developers providing bespoke software and services for large companies in South Africa. This experience has given him a deep understanding of software development and the software development lifecycle. Mr. Themistocleous popularized many technologies in South Africa. In 1993, he began promoting the use of object orientation in commercial systems. He also introduced the Java development language to South Africa in January 1996 and operated one of the first authorized Sun Java centers in the world.

Since 1993, Mr. Themistocleous has introduced more than 200 organizations to the iterative approach to software development. He is a firm believer that Agile — done properly — is the way to go but that Agile processes such as Scrum are missing elements that enhance the success of customer projects. As a result, he is a vocal promoter of Disciplined Agile Delivery (DAD).

Mr. Themistocleous listed the first IT company on the Johannesburg Stock Exchange in 1995. He has served on the founding committee for the Johannesburg Centre for Software Engineering, the advisory board for information systems at Wits University (Johannesburg, South Africa), and as an advisor to the South African government. Mr. Themistocleous has a B.Com. degree, is a Chartered Accountant, and holds various technical certifications and a diploma in marketing. He can be reached at themi@mweb.co.za.
Cutter Events

Agile Masterclass
with Israel Gat, Fellow and Director, Agile Product & Project Management Practice, Cutter Consortium; and Hubert Smits, Senior Consultant, Cutter Consortium

In this half-day Masterclass, Israel Gat and Hubert Smits will teach you — in a hands-on, practical way — how you can use Agile and Lean concepts, artifacts and practices in a broader and deeper context to make your organization more innovative and competitive. They will divulge the tools you need to move well beyond the team-level basics. You will:

- Discover how to scale frameworks like Scrum for larger projects.
- Learn how to weave together the outcomes that result when multiple frameworks (agile, lean or traditional) are used by different teams, projects, or programs.
- Discover how to get advanced software engineering disciplines like technical debt assessment/remediation and devops' infrastructure as code to take root in your organization; and,
- Find out, via a case study, how Agile is being extended outside of software into other departments, such as Marketing, and how success in IT/product development can be leveraged to drive change both up- and down-stream.

When you combine the experience-based strategies Israel and Hubert share with the techniques and skills they impart in this half-day Masterclass, you'll be another step closer to creating the strong foundation that continuous value delivery requires.

Agile and Outsourcing: A Disciplined Approach
Recorded Webinar with Scott Ambler, Senior Consultant, Cutter Consortium

In this webinar, Senior Consultant Scott Ambler explores strategies for effectively initiating and governing an outsourced IT delivery project in an agile manner.

Outsourcing introduces a collection of risks that can be uniquely addressed with a disciplined agile strategy. Luckily, the Disciplined Agile Deliver y (DAD) framework provides a foundation from which you can tailor a viable strategy for disciplined agile outsourcing. DAD is a goal-driven, hybrid agile, full delivery methodology that is enterprise aware and scalable.

This webinar addresses:

- Why agile and outsourcing?
- The risks associated with outsourcing.
- What the Disciplined Agile Delivery (DAD) framework is.
- Disciplined agile outsourcing from the point of views of the customer and the provider.
- What you need to do to succeed at disciplined agile outsourcing.
- Industry statistics regarding agile outsourcing in practice.
- Criteria to determine if you’re ready for outsourcing IT delivery projects.

Spend an hour learning about DAD from its creator, Scott Ambler, and determine if outsourcing IT delivery projects is a smart move for your organization.
Cutter Events

Cutter Consortium’s webinars, Q&As, and virtual Peer-to-Peer sessions are offered for no fee to Cutter members as part of their Membership. Online sessions are delivered via Adobe Connect. Webinar recordings are available in the Cutter Consortium research archives at www.cutter.com/events.html.

Peopleware

Member Exclusive Peer-to-Peer session with Tim Lister, Cutter Fellow

In this exclusive Peer to Peer for Cutter Members, Cutter Fellow Tim Lister will lead a discussion on how major issues of software development are human, not technical. Lister, co-author with Tom DeMarco of the groundbreaking book, *Peopleware: Productive Projects and Teams*, will answer your questions on why some projects succeed and others fail. Both success and failure have a dramatic financial impact on the organization — yet we know far too little about the underlying reasons for project effectiveness.

Lister will also address how throwing technology and money at projects is wasted if the project team does not understand how they should respond to unforeseen circumstances, or if the team has some pathological behavior that prevents any worthwhile accomplishment, or lacks the appropriate skills and strategies. Other possible discussion topics include:

- Development of productive persons
- Organizational culture
- Organizational learning
- Development of productive teams
- Modeling of human competencies

*Peopleware*, with its third edition just published, continues to inspire intense devotion from software project managers around the globe. Please join Tim for what promises to be a lively discussion about *Peopleware*!

A Practical & Tested Approach to Measuring IT Cost and Value

Webinar with Bob Benson, Fellow, Cutter Consortium

Business and government executives often struggle to measure and understand the value of their IT investments, especially the ongoing costs of infrastructure and existing applications. In this webinar, Cutter Fellow Bob Benson describes a methodology that gives a clear perspective on:

- IT costs (such as the cost of IT for each business unit)
- IT values (for example, the performance of the IT investment for each of the organization’s strategic objectives)

The methodology’s results, which have been applied in more than 100 organizations, provide cost and value metrics that can be compared to other organizations and with other business units inside the enterprise. Key elements include a powerful cost model and an effective methodology for connecting IT costs to IT’s performance.

Please join us for this informative discussion on how you can apply this methodology in your own organization.

Answers to Your Questions

The last 15 minutes of the webinar are dedicated to questions from the audience. You can ask Bob Benson your questions about measuring IT cost and value.

Date: Tuesday, September 10, 2013
Time: 12:00 pm EDT
Duration: 60 minutes
Location: Wherever you are! Just dial in and/or log in.
Fee: Complimentary
Executive education on leadership, teaming, and cutting-edge IT.

Summit 2013
4–6 November 2013 | Cambridge, MA, USA

Summit 2013: Executive Education+ is a world-class Exec Ed program for C-level business technologists.

Cutter Fellows who are members of the Harvard Business School faculty will both guide the discussion of HBS cases/exercises you’ll work through as a group and keynote on topics that enhance your case learnings.

In addition, you’ll benefit from hands-on seminars and breakfast and lunch sessions led by Cutter’s Practice Directors and Senior Consultants on topics such as Agile Management, Business and Enterprise Architecture, CIO/CTO issues, Agile Analytics, and Social BI, to name a few.

You’ll enjoy (and join in on!) raucous panel debates; networking at lunches, breaks, and entertaining evening events; and get one-on-one guidance and input from expert presenters and participants.

Learn more at www.cutter.com/summit.html
About Cutter Consortium

Cutter Consortium is a truly unique IT advisory firm, comprising a group of more than 100 internationally recognized experts who have come together to offer content, consulting, and training to our clients. These experts are committed to delivering top-level, critical, and objective advice. They have done, and are doing, groundbreaking work in organizations worldwide, helping companies deal with issues in the core areas of software development and Agile project management, enterprise architecture, business technology trends and strategies, enterprise risk management, metrics, and sourcing.

Cutter offers a different value proposition than other IT research firms: We give you Access to the Experts. You get practitioners’ points of view, derived from hands-on experience with the same critical issues you are facing, not the perspective of a desk-bound analyst who can only make predictions and observations on what’s happening in the marketplace. With Cutter Consortium, you get the best practices and lessons learned from the world’s leading experts, experts who are implementing these techniques at companies like yours right now.

Cutter’s clients are able to tap into its expertise in a variety of formats, including content via online advisory services and journals, mentoring, workshops, training, and consulting. And by customizing our information products and training/consulting services, you get the solutions you need, while staying within your budget.

Cutter Consortium’s philosophy is that there is no single right solution for all enterprises, or all departments within one enterprise, or even all projects within a department. Cutter believes that the complexity of the business technology issues confronting corporations today demands multiple detailed perspectives from which a company can view its opportunities and risks in order to make the right strategic and tactical decisions. The simplistic pronouncements other analyst firms make do not take into account the unique situation of each organization. This is another reason to present the several sides to each issue: to enable clients to determine the course of action that best fits their unique situation.

For more information, contact Cutter Consortium at +1 781 648 8700 or sales@cutter.com.

The Cutter Business Technology Council

The Cutter Business Technology Council was established by Cutter Consortium to help spot emerging trends in IT, digital technology, and the marketplace. Its members are IT specialists whose ideas have become important building blocks of today’s wide-band, digitally connected, global economy. This brain trust includes:

- Rob Austin
- Ron Blitstein
- Tom DeMarco
- Lynne Ellyn
- Israel Gat
- Vince Kellen
- Tim Lister
- Lou Mazzucchelli
- Ken Orr
- Robert D. Scott