Cutter
IT Journal

The Journal of
Information Technology Management

Vol. 25, No. 11
November 2012

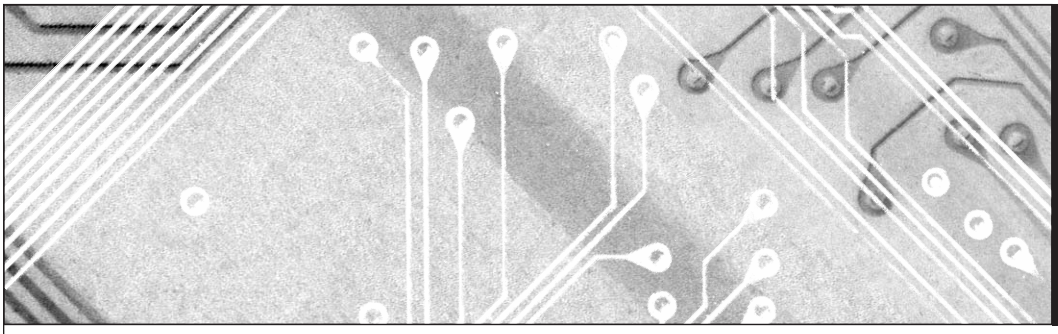
"If you still believe that agile and CMMI don't work well together, then what you'll learn in this issue is that, to put it plainly, you're wrong."

— Hillel Glazer,
Guest Editor

Agile CMMI:

Why Isn't This Conversation Dead Yet?

CUTTER CONSORTIUM



Cutter IT Journal

About Cutter IT Journal

Part of Cutter Consortium's mission is to foster debate and dialogue on the business technology issues challenging enterprises today, helping organizations leverage IT for competitive advantage and business success. Cutter's philosophy is that most of the issues that managers face are complex enough to merit examination that goes beyond simple pronouncements. Founded in 1987 as *American Programmer* by Ed Yourdon, *Cutter IT Journal* is one of Cutter's key venues for debate.

The monthly *Cutter IT Journal* and its companion *Cutter IT Advisor* offer a variety of perspectives on the issues you're dealing with today. Armed with opinion, data, and advice, you'll be able to make the best decisions, employ the best practices, and choose the right strategies for your organization.

Unlike academic journals, *Cutter IT Journal* doesn't water down or delay its coverage of timely issues with lengthy peer reviews. Each month, our expert Guest Editor delivers articles by internationally known IT practitioners that include case studies, research findings, and experience-based opinion on the IT topics enterprises face today — not issues you were dealing with six months ago, or those that are so esoteric you might not ever need to learn from others' experiences. No other journal brings together so many cutting-edge thinkers or lets them speak so bluntly.

Cutter IT Journal subscribers consider the *Journal* a "consultancy in print" and liken each month's issue to the impassioned debates they participate in at the end of a day at a conference.

Every facet of IT — application integration, security, portfolio management, and testing, to name a few — plays a role in the success or failure of your organization's IT efforts. Only *Cutter IT Journal* and *Cutter IT Advisor* deliver a comprehensive treatment of these critical issues and help you make informed decisions about the strategies that can improve IT's performance.

Cutter IT Journal is unique in that it is written by IT professionals — people like you who face the same challenges and are under the same pressures to get the job done. *Cutter IT Journal* brings you frank, honest accounts of what works, what doesn't, and why.

Put your IT concerns in a business context. Discover the best ways to pitch new ideas to executive management. Ensure the success of your IT organization in an economy that encourages outsourcing and intense international competition. Avoid the common pitfalls and work smarter while under tighter constraints. You'll learn how to do all this and more when you subscribe to *Cutter IT Journal*.

Cutter IT Journal®

Cutter Business Technology Council: Rob Austin, Ron Blitstein, Tom DeMarco, Lynne Ellyn, Israel Gat, Vince Kellen, Tim Lister, Lou Mazzucchelli, Ken Orr, and Robert D. Scott

Editor Emeritus: Ed Yourdon
Publisher: Karen Fine Coburn
Group Publisher: Chris Generali
Managing Editor: Karen Pasley
Production Editor: Linda M. Dias
Client Services: service@cutter.com

Cutter IT Journal® is published 12 times a year by Cutter Information LLC, 37 Broadway, Suite 1, Arlington, MA 02474-5552, USA (Tel: +1 781 648 8700; Fax: +1 781 648 8707; Email: citjeditorial@cutter.com; Website: www.cutter.com; Twitter: @cuttertweets; Facebook: Cutter Consortium). Print ISSN: 1522-7383; online/electronic ISSN: 1554-5946.

Subscription rates are US $485 a year in North America, US $585 elsewhere, payable to Cutter Information LLC. Reprints, bulk purchases, past issues, and multiple subscription and site license rates are available on request.

☐ Start my print subscription to *Cutter IT Journal* ($485/year; US $585 outside North America)

Name _____ Title _____

Company _____ Address _____

City _____ State/Province _____ ZIP/Postal Code _____

Email (Be sure to include for weekly *Cutter IT Advisor*) _____

Fax 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.

SUBSCRIBE TODAY

Request Online License Subscription Rates

For subscription rates for online licenses, contact us at **sales@cutter.com** or **+1 781 648 8700.**

Opening Statement

by Hillel Glazer, Guest Editor

Agile or CMMI. Agile and CMMI. AGILE AND CMMI! AGILE AND CMMI?!?!!#%&***@^@!!

Which is it? Where are you in it?

Is there an answer? The question is still out there.

There are many answers. There is no shortage of theories and angles from which to view both the questions and the answers. This likely explains why the agile CMMI conversation isn't dead yet.

Maybe you're an agile shop looking to adopt CMMI for the marketing benefits or (shh!) for its introduction of some structure, scalability, and discipline, but you're concerned about the expected documentation or the appraisal necessary to "prove" your accomplishments, and you're worried about losing your collaborative, trusting culture. Or perhaps you're a more traditional organization dipping your toe into the agile hot tub and are turned on by the rush of results and happy staff, but you're worried that you'll devolve from your predictable routines and paper trail.

The news in this issue of *Cutter IT Journal* is that it's not agile or CMMI that fosters your culture, and it's not agile or CMMI that shuns or requires documentation. In fact, as we'll see, it's really not about agile *or* CMMI. And perhaps "To agile or not to agile?" is the wrong question to ask and the wrong perspective on matters.

If culture and trust are not exclusive to agile methods, and documentation and artifacts are not required from CMMI, where's the problem? What if focusing on "agile" or "CMMI" is a misplaced effort? This issue looks into that. What if learning practices and rote copying by the "cargo cult" set is part of the problem that prevents learning by both sets of practitioners? What if CMMI actually contains valuable ideas that agile teams can use? What if agile teams already do many aspects of CMMI without knowing it? These are the questions our authors address in this issue.

Want case studies? Want a framework in which to make it all work? This edition of *Cutter IT Journal* has your back. We get a glimpse into several companies that are using CMMI and agile together and successfully earning "levels," and we also see how to put together modular processes that are entirely compatible with both CMMI and agile.

Got really big projects? Got unwieldy engineering? Need to scale agile to work such challenges? Need CMMI to not get in the way of progress? Stop saying "It can't be done here" and start reading how it's being done!

If culture and trust are not exclusive to agile methods, and documentation and artifacts are not required from CMMI, where's the problem?

I'm pleased to report that the question is no longer one of whether or not agile and CMMI can coexist. Instead, the question has, if you'll pardon the term, *matured* to the point where we're asking what's going wrong when they don't work well together and what's going right when they do.

If you still believe that agile and CMMI don't work well together, then what you'll learn in this issue is that, to put it plainly, you're wrong. In fact, not only are you wrong, but by screwing your eyes shut, sticking your fingers in your ears, and yelling "la-la-la-la" at the top of your lungs, you're missing out. Is someone eating your lunch? Oh, what a shame! They're probably successfully blending agile and CMMI in ways you haven't realized were possible. While you were too busy investing in the belief that agile and CMMI are "oil and water," your competition was making and selling salad dressing.

What is likely true of many believers of the oil-and-water myth is that *their* experiences with agile and CMMI did, in fact, result in very unpleasant outcomes. But why? In all likelihood — and this contention is supported by this month's contributors — these horror stories have several characteristics in common. Common causes of horror include: a mandate to "get agile" or "get a CMMI rating"; a culture that punishes failure and

experimentation and devalues trust; learning without understanding; CMMI appraisers or agile coaches who operate on practices without appreciating the values and principles of which the practices are merely derivatives; the plethora of appraisers/coaches who don't understand the other domain; mismatches in the work being done and the means of accomplishing the work — and let's not forget the persistent lack of basic discipline in which neither agile nor CMMI can survive.

It seems a fairly universal conclusion that it's not agile or CMMI that are incompatible but the way they're applied in a given situation that makes them incompatible.

The good news is that in this issue we have lined up six articles that address everything from the right questions to ask (in Bill Fox's lead-off piece) to how to deal in an agile manner with very large systems (in a discourse from a number of folks at the venerable Software Engineering Institute). It seems a fairly universal conclusion that it's not agile or CMMI that are incompatible but the way they're applied in a given situation that makes them incompatible.

What's important to note, however, is that situations aren't what cause the incompatibilities, it's *how* agile and/or CMMI are being applied that make them so. Very often, these incompatibilities are self-inflicted, not imposed by anything in agile or CMMI. To wrap our heads around the topic, the articles are organized to move us through the following stages: thinking, learning, applying, and broadening.

UPCOMING TOPICS IN CUTTER IT JOURNAL

DECEMBER

Roger Evernden

Enterprise Patterns

JANUARY

Vince Kellen

SMAC: Social, Mobile, Analytics, and Cloud

In our first article, Bill Fox looks at whether we're asking the right questions and reflecting on what we're trying to accomplish. By the time you read this, Fox will have interviewed three dozen experts in the field. Within two dozen, he came across a fascinating discovery that can change your own thinking on how to speed delivery and improve quality — and every other notion often attributed to agile, CMMI, or both. What if success with agile or CMMI really has nothing to do with agile or CMMI? Fox takes us down that thought path.

Next Brian Button and Nate McKie observe that many agile organizations' failures to effectively use CMMI likely result from copying what others are doing without understanding or learning why they do it or why it works. Instead of learning, they argue, too many organizations are rote copying — and that leads to trouble. Rote copying isn't an agile value and doesn't work in CMMI either. In their article, we get to see, through their eyes, what it's like to apply the agile principle of learning to the adoption of CMMI. The authors relate how using CMMI allowed them to expose and address some of their growing pains, which other companies committed to agile may also experience. From changing what they initially believed about CMMI, to learning from their doing (and certain wrong turns made along the way), Button and McKie discuss what they discovered when they stopped to understand what CMMI was trying to show their agile operation.

Transitioning our attention from learning to application, Paul McMahon exposes "the real underlying obstacles" to agile CMMI and also shows us what can be done about them. He further describes innovative ways to interpret and apply CMMI so that its practices actually make sense in agile settings. McMahon's real-world experience is recast to protect the guilty in a series of accessible cases of actually making it work.

Lest we ignore the demand for specific guidance, Jeff Dalton walks us through a fairly thorough application of CMMI in Scrum settings. He further demonstrates an approach to CMMI that is not only compatible with Scrum, but also uses Scrum and agile thinking to facilitate CMMI! It's not merely a matter of such-and-so Scrum practices demonstrating this-or-that CMMI practice — that would be both easy and disingenuous. Dalton practices what he preaches and would never lead a company down a path that only solves their performance needs once, leaving them with nothing with which to fend for themselves when circumstances change. Instead, he offers us a delightfully simple and robust architecture that we can use to build processes incrementally and iteratively. How agile!

We round out the issue by getting into more thought-provoking ideas with work from Cutter Senior Consultant Scott Ambler, a well-known columnist, Internet agile personality, and author. Ambler discusses an architecture for disciplined agile delivery and what happens when it … meets CMMI. It's not exactly a complete smackdown, but it definitely makes us rethink what we know about agile and CMMI! Ambler pulls data from several broad, well-thought-out industry surveys he's conducted and scrutinizes the results. He addresses the "agile vs. CMMI" rhetoric head on and describes a framework he's been working on based on the findings. The result, Disciplined Agile Delivery (DAD), is a decision framework for determining the right processes to use. In the article, he points out how DAD supports both agile and CMMI and provides the means of letting you figure out what will work best for your organization. He doesn't let CMMI practitioners off the hook for their predisposition to focus on processes rather than results, but he also holds the *agilistas*' feet to the fire to evolve and mature as well.

Finally, if ever there were a way to confront resistance from either agile or CMMI, it would be embodied in the SEI's broad perspective on "agility-at-scale." You think you've got agile/CMMI compatibility issues? They're child's play compared to what you'll see the SEI is up to. Unless you plan to work out of your minivan for the rest of your career, you don't want to miss the mind-expanding problems that Doug Schmidt and his coauthors are dealing with! Schmidt et al. observe that everyone wants to "be agile," but sticking to agile practices and lore when they've run out of utility is like sticking with a brand of running shoes when they've stopped providing you support. Looking at the matter from the perspective of questions about risk, measures, technical debt, and strategy sounds uncharacteristically like business value, not process compliance. Well, it may come as a surprise to some, but at the SEI, these topics have always been at the center of CMMI. Which returns us to our starting point: it's not what's in CMMI or agile that causes this conversation to remain in play, it's what you do with them.

I hope you enjoy this month's *Cutter IT Journal* and welcome your thoughts!

*Hillel Glazer, a Senior Consultant with Cutter's A*gile Product & Project Management *practice, is recognized as the world's leading authority on introducing Lean and Agile concepts into the compliance-driven world. Mr. Glazer has helped companies of all sizes and industries around the world successfully streamline their operations, increase value, and expose and eliminate practices that prevent them from achieving their performance goals. And he does so while simultaneously accounting for all the external compliance pressures on their operations.*

Mr. Glazer has been successfully pioneering the introduction of Lean philosophies, methods, and techniques into businesses and industries otherwise believed to be either too chaotic or too constrained by their compliance and regulation requirements to adopt high-performance approaches. His leadership, originality, excellence, and direct contribution to the community in this field have been recognized by the Lean Systems Society, which honored him as a Fellow of the Lean Systems Society in its inaugural induction of fellows.

As an in-demand speaker, presenter, and facilitator, Mr. Glazer is widely read, broadly published, and appears worldwide on the topics pertaining to operational excellence in compliance-driven industries. His work appears in many publications, including periodicals and the following SEI publications: CMMI for Services, CMMI for Development, *and* Integrating CMMI and Agile Development. *He is the lead author of the SEI's 2008 seminal Agile/CMMI work, "CMMI or Agile: Why Not Embrace Both!" and author of the well-received book* High Performance Operations.

Currently, Mr. Glazer is focusing attention on Agile in systems engineering and Lean at the enterprise. He lives in the Baltimore suburbs with his fabulous wife and four amazing children, and when not traveling around the planet making Lean and Agile things happen, he's an avid pilot, musician, and an active volunteer with several nonprofits and charities. He can be reached at hglazer@cutter.com.

The Agile CMMI Conversation Is a Dead End

by Bill Fox

HOW ARE AGILE, CMMI, LEAN, ETC. REALLY WORKING OUT FOR MOST ORGANIZATIONS?

Let's own up to it. For most organizations, focusing on agile, CMMI, both, or any other methodology is still not producing the desired outcomes. While the benefits and advantages of combining agile and CMMI have been documented and proven, continuing to discuss how these methodologies intersect only improves things at the margins. I'm not saying that agile and CMMI aren't valuable contributions to our efforts to improve our organizations, but is this where we should be concentrating our efforts? I believe we need to focus on a bigger conversation if we want to create empowered organizations capable of producing needle-changing results.

Over the past 30 years, I've worked to help organizations introduce change and transformation. In that time, one thing has become painfully clear: most organizations fail at transformation — miserably and repeatedly. A few organizations reach some level of success and make meaningful progress, but for how long? Any progress usually doesn't outlive the length of service of the senior executive sponsoring or leading the change. Or if the entire organization doesn't embrace the change, it gets thrown out because it doesn't fit.

The reason that the agile CMMI conversation won't go away is because we are all still fundamentally focused on the wrong things.

The premise of this article is not to suggest that CMMI, agile, or other methodologies are wrong. It's that the problems most organizations have with their processes and their process improvement initiatives are more fundamental and pervasive than a software development methodology can address. The reason that the agile CMMI conversation won't go away is because we are all still fundamentally focused on the wrong things. What follows are some of the key areas where I believe we are going astray, the results of my research and conversations with top experts, and my recommendations on where I believe we need to refocus our efforts.

WHERE WE ARE GOING ASTRAY

We Blindly Accept Industry "Best Practices" Without Regard to Context

We think that because someone declares something a "best practice," that means it works everywhere, regardless of context. In a *Forbes* blog post on best practices, Mike Myatt makes a bold statement in saying, "There is no such thing as best practices."[1] He goes on to note:

> The reality is best practices are nothing more than disparate groups of methodologies, processes, rules, concepts, and theories that attained a level of success in certain areas, and because of those successes, have been deemed as universal truths able to be applied anywhere and everywhere. Just because someone says something doesn't mean it's true. Moreover, just because "Company A" had success with a certain initiative doesn't mean that "Company B" can plug-and-play the same process and expect the same outcome. There is always room for new thinking and innovation, or at least there should be.[2]

Like Myatt, I also find myself "cringing anytime I hear someone use the phrase in an authoritarian manner as a justification for the position they happen to be evangelizing."[3]

We Believe a Methodology Provides a Complete Solution and Fail to Understand the Real Problem

This month's Guest Editor and Entinex CEO Hillel Glazer has stated that, in his experience, many of the fundamental challenges that organizations experience can be dealt with quicker and with greater success by introducing appropriate techniques *before* implementing agile, CMMI, or other such methodologies. For example, many organizations could improve performance simply by limiting the amount of work in progress and/or by minimizing changes in priorities.

Yet in my experience, the vast majority of organizations are quick to adopt a packaged methodology thinking it provides the complete solution, while not fully

appreciating its limitations. It's like going to the edge of a 50-foot cliff and jumping when all you've witnessed are others ahead of you jumping away. As a result, you don't see what they did before they jumped, what the landing area looks like, or what happened when they landed!

Unfortunately, many agile, CMMI, and other such experts are focused on their toolset, and they don't see outside the box. It's rare to find people who have the breadth and scope to see the whole landscape. Furthermore, organizations don't know they need to seek out these people with a broader perspective.

We Don't Recognize How the Limitations of Our Flawed Thinking Result in Poor Decisions

Daniel Kahneman, recipient of the Nobel Prize in Economics, has done seminal work in psychology that challenges the rational model of judgment and decision making. In his landmark book *Thinking Fast and Slow*,[4] Kahneman discusses two different ways of thinking — "System 1" and "System 2" — and warns us that the former type can derail our decision-making process:

> System 1 thinking corresponds to fast, intuitive, emotional, and almost automatic decisions, though it sometimes leaves us at the mercy of our human biases. System 2 thinking is more slow-going and requires more intellectual effort. To nobody's surprise, we humans are more likely to rely on System 1 thinking, because it saves us effort, even if it can lead to flawed thinking.[5]

From my perspective of observing organizations go through the decision-making process and from conversations with other consultants on this topic, it appears that decision making in many organizations is based on System 1 thinking. I first observed it with RUP, then CMMI, and now agile. Organizations make little or no effort to identify the fundamental problem to be solved or the ideal end state, nor do they examine the various options that could provide an appropriate solution.

We Don't Appreciate the Power of a Group of Empowered Individuals to Outperform Any Group of Experts and Best Practices

Anyone who has ever been associated with a winning sports team has probably witnessed the almost miraculous performance and elation of the team members. But how often do you see this type of performance and elation in the workplace? And why isn't that the case? Isn't this the type of fully engaged performance and atmosphere that will deliver the results we are seeking? In David Marquet's book *Turn the Ship Around!*[6] he tells the unlikely story of how he transformed the crew of the nuclear submarine USS *Santa Fe* from last to first in the fleet in performance. I believe Marquet's approach of creating leadership at every level is what organizations need to do today. In an Amazon review of his book, Marquet says:

> I wrote this book to help pioneer a better way of treating each other, a way of treating each other that will allow more of us to freely give of our passion, intellect, and creativity. In short, that way means treating others as equals, as co-leaders, and not as followers. It means giving control not taking control.[7]

I believe Marquet's results speak for themselves. Word of the stunning turnaround and his success traveled rapidly throughout the US Navy and caught the attention of the acclaimed author and leadership expert Stephen Covey. Covey was motivated to reach out to Marquet and witness the results firsthand. He told Marquet, "It was the most empowered organization he'd seen anywhere, not just in the military."[8]

FINDINGS THAT POINT US IN A NEW DIRECTION

For the past two years, I have interviewed top thinkers and experts to discover their best ideas and strategies for helping organizations improve. These interviews, which form the basis of my series *5 Minutes to Process Improvement Success*,[9] all begin with the same question: "What is your best performance or process improvement strategy that has worked well for you and your clients?" After more than 30 interviews, it's fascinating to find that none of the 30 strategies are the same. It's also interesting that none of the strategies include the words agile, CMMI, Kanban, lean, TQM, or the like.

I don't believe the above findings diminish any of these methodologies or suggest that they aren't great or a part of the solution, because they clearly are. But what's interesting is that no one I've interviewed has stated that we should *focus on* a methodology or model. Instead, the best strategy for improvement success is to ask great questions that help organizations get in touch with the ideal end state they'd like to achieve and why.

What are those questions? Below are a few examples of questions the experts recommend:

- **Alan Shalloway** (founder and CEO, Net Objectives): "I'd say the key thing is to really understand the client's problem. What we see a lot of in the industry now is taking some great methods such as Scrum, which is a phenomenal team process, and applying it without necessarily looking at the organization's true challenge.… And that's the first order of business, to understand what your challenge is and where you want to go."[10]

- **Scott Ambler** (Cutter Senior Consultant and former Chief Methodologist for Agile and Lean at IBM Rational): "I'll usually let them start off with conversations around, 'What are you doing now? What do you want to achieve? Why do you want to do process improvement at all?' Then that will sort of lead them to identify what they believe their pain points are. Particularly the goals will reflect what their current pain is, and what they are trying to achieve, and then I'll start going in on the, 'What are you currently doing?' and helping them discern what is not working well."[11]
- **Karl Wiegers** (Principal Consultant, Process Impact): "So the first thing I want to know is why people are doing this, and that leads to a very natural second question: How will you know if you've achieved the desired results? In other words, what can you measure or observe that would tell you if whatever changes you're making are getting you where you want to be?"[12]
- **David Marquet** (Former Commander, USS *Santa Fe*): "We had a basic rubric, which was to ask the following question when something went wrong, 'What was the process, and who was the owner?' So you end up in a two-by-two matrix. No owner, no process is chaos, but some of the things that you would think you had processes for, we didn't."[13]

I witnessed perhaps one of the most powerful demonstrations of the power of questions when I spent some time at a weeklong conference with IT consultant Dale Emery. At the conference, I saw Emery help others solve many difficult challenges simply by asking good questions. In a subsequent interview, Emery stated that he uses questions to:

> … help people use skills and knowledge they already have, but for some reason are not applying to their current goals or activities. People always have knowledge and skills that, for whatever reason, are just not occurring to them in the moment. So I try to help people tap into their existing abilities.[14]

Still another example comes from Kanban pioneer David Anderson. In his interview with me, Anderson talked about the Kanban method for catalyzing incremental improvements, which basically is this: find the number one problem, fix it, and then the number two problem gets promoted and you fix *it*. While Anderson isn't verbally asking a question, in effect he is using a system to continually ask, "What's the number one problem?" After you fix it, you ask the question again to discover the next most important problem to solve.[15]

What is it about questions that make them so powerful? I recently interviewed Steve Gladis, the author of *The Coach-Approach Leader*.[16] Gladis is an expert in asking questions, which is a primary focus of his leadership approach. His thoughts and experiences with asking questions reflect my own. We both believe in asking questions because they help us to:

- Slow down and be more mindful and deliberative
- Discover the real challenge to be solved
- Understand the impact of an issue or challenge
- Envision the ideal end state we are trying to reach
- Establish an intention, moving us toward meaningful and deliberate action

Given our limited progress on process improvement to date and the findings from top experts, I suggest it's time we refocus our efforts.

WHERE WE NEED TO SHIFT OUR FOCUS TO BRING ABOUT THE OUTCOMES WE ARE ALL SEEKING

What's the number one reason cited for lack of success with improvement projects? That's right, you've heard it many times — "lack of leadership support."

I've been hearing that lack of leadership support is the main cause of improvement project failure ever since I first got involved with such projects almost 30 years ago. Brad Power echoes this conclusion in a September 2012 post on the *Harvard Business Review* blog.[17] In that post, Powers reminds us of John Kotter's classic *Harvard Business Review* article "Leading Change: Why Transformation Efforts Fail,"[18] which shows that "few attempts at fundamental change are very successful, a few are utter failures, and most fall somewhere in between, with a distinct tilt to failure."[19]

Despite the almost universal agreement that lack of leadership support is the number one problem and has been for more than 30 years, are we seeing any improvement? From my perspective, the answer is a resounding "NO."

So where does that leave us? If lack of leadership support is the key contributing factor to failure, what are we really doing about it? Is it fair or even possible to expect leaders in today's complex and fast-changing world to know it all and be involved in all the details of everything they are responsible for?

I believe we need to refocus our attention on three key areas, because what we are doing now isn't very effective for most organizations. They are:

1. Creating leadership at every level
2. Understanding your strategy and purpose and their impact on value creation

3. Fostering evolutionary change through self-leadership at any level

Creating Leadership at Every Level

If lack of leadership support is a primary cause of the failure of change efforts, then why not share the burden and encourage *everyone* to be a leader?

Surely there can't be a more dramatic example of creating leadership at every level than Marquet's *Turn the Ship Around!* In my interview with Marquet,[20] he reveals how he used questions and changes in language to drive changes in behavior that unleashed excellence and the passion, intellect, and creativity of every member of the crew.

As noted above, Marquet's key strategy was to ask a two-pronged question anytime something went awry. That question was, "What was the process, and who was the owner?" It's a simple enough question, but it's powerful when used with sufficient motivation to address problems when they occur. The question quickly determines if there is a lack of a process, or lack of an owner, or perhaps both. If either or both are lacking, a responsible party is identified and/or a process is developed for preventing a similar problem in the future.

Another interesting aspect to Marquet's leadership approach was a careful selection of the words the crew would use. For example, Marquet insisted that the crew say "I intend to" rather than requesting his permission to carry out a task or an event. He discovered that this slight change in the use of language has a powerful impact on making subordinates think at a higher level.

These are just two facets of Marquet's approach to creating leadership at every level. I highly recommend his book[21] and our interview[22] for further study.

Understanding Your Strategy and Purpose and Their Impact on Value Creation

My purpose in creating the *5 Minutes to Process Improvement Success* interview series was to uncover the best strategies of top experts, so I could share those strategies with the community at large and we could all learn from them.

Recently, I published a review,[23] along with my colleague Samira Askarova, on Cynthia Montgomery's book *The Strategist: Be the Leader Your Business Needs*.[24] Montgomery is a professor of business administration at Harvard Business School, where she has worked with top executives from around the world. While reviewing the book, I discovered this fascinating insight:

> If leaders lack a clear idea about what they want their businesses to be, they cannot build coherent systems of value creation because they don't know what they should be designed to do exactly or how their success should be measured. That leaves them to fiddle at the margins of success with generically good practices such as "state of the art" sales management approaches or Total Quality Management. These may be helpful, but they're not what will help you find an edge and live on it.[25]

Have we been fiddling at the margins of success for too long?

It's becoming clear to me that asking the right questions helps organizations get in closer touch with what leaders want their businesses to be. That's why the experts I've interviewed have been more successful at helping organizations improve.

Fostering Evolutionary Change Through Self-Leadership at Any Level

Marie Maher, a director in education who has worked at well-known US nonprofit organizations for the past 19 years, recently contacted me after seeing my work at *5 Minutes to Process Improvement Success*. Marie's story is typical of those who reach out to me and, frankly, typical of mine before I created the interview series. Marie feels that her talents are being vastly underutilized in her current position and would like to make a bigger contribution:

> I definitely have the desire and motivation to attain that dream position. I feel that there is so much I can achieve but I just have to find the right opportunity. It just feels like I am in a very static place right now, but I have the impetus to change that. Static and unchallenging is not a good fit for me. While my company is a nice place to work, I have been in this business for 19 years and I know it very well. Regardless of role, I am no longer challenged. Any work I am tasked with has become rote. And between my work experience and MBA, I want to be in a more progressive, exciting environment.

Here's what Marie's comments and those of so many others mean to me. Most of what organizations are seeking is lying dormant in the knowledge and experience of their employees. Unfortunately, organizations are either blind to it or don't know how to tap this immense wealth of passion, creativity, and intellect.

It is my conviction that people who find themselves in such a position can begin to impact their situation by holding a conversation on improvement right where they are right now. Do you want to see change? Be the change. Start a conversation on improvement; ask your coworkers what frustrates them and what opportunities they see for change. Collect these ideas, communicate them, and promote them.

Be a leader wherever you are. Leaders decide what they want to see and be in the world — and then do whatever they can to make that a reality rather than simply reacting to circumstances. True leadership begins with you and your inner decision to lead. It doesn't start with power or require a following. It means taking the time to get in touch with what you truly want and working to make it real.

BEYOND AGILE CMMI

It's time to focus our efforts on what really matters and will make a difference. While the agile CMMI conversation has been useful, let's move beyond it to discussions that will add greater value and have a bigger impact.

ENDNOTES

[1]Myatt, Mike. "Best Practices — Aren't." *Forbes* (blog), August 2012 (www.forbes.com/sites/mikemyatt/2012/08/15/best-practices-arent).

[2]Myatt. See 1.

[3]Myatt. See 1.

[4]Kahneman, Daniel. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011.

[5]Kahneman. See 4.

[6]Marquet, David. *Turn the Ship Around! How to Create Leadership at Every Level*. Greenleaf Book Group Press, 2012.

[7]Fox, Bill. "What Can a US Navy Fast Attack Submarine Captain Teach You About Leadership?" Lead Change Group, 14 August 2012 (http://leadchangegroup.com/what-can-a-us-navy-fast-attack-submarine-captain-teach-you-about-leadership).

[8]Marquet. See 6.

[9]*5 Minutes to Process Improvement Success* (http://5minutespisuccess.com).

[10]Shalloway, Alan. "Understand What Your Challenge Is and Where You Want to Go." Interview by Bill Fox. *5 Minutes to Process Improvement Success*, 28 September 2011 (http://5minutespisuccess.com/?p=118).

[11]Ambler, Scott. "A Perspective from the Chief Methodologist for Agile/Lean at IBM Rational." Interview by Bill Fox. *5 Minutes to Process Improvement Success*, 24 January 2011 (http://5minutespisuccess.com/?p=63).

[12]Wiegers, Karl. "Three Key Questions to Ask at the Start of Any Process Improvement Initiative." Interview by Bill Fox. *5 Minutes to Process Improvement Success*, 24 May 2011 (http://5minutespisuccess.com/?p=103).

[13]Marquet, David. "What Was the Process, and Who Was the Owner?" Interview by Bill Fox. *5 Minutes to Process Improvement Success*, 8 August 2012 (http://wp.me/p2gSAP-c6).

[14]Emery, Dale. "The Amazing Power of Asking Questions and Following the Energy." Interview by Bill Fox. *5 Minutes to Process Improvement Success*, 16 January 2012 (http://5minutespisuccess.com/?p=152).

[15]Anderson, David. "Kanban Systems: An Evolutionary Incremental Approach to Improvements." Interview by Bill Fox. *5 Minutes to Process Improvement Success*, 26 January 2012 (http://5minutespisuccess.com/kanban-systems-an-evolutionary-incremental-approach-to-improvements-with-david-anderson).

[16]Gladis, Steve. *The Coach-Approach Leader*. HRD Press, 2012.

[17]Power, Brad. "Understanding Fear of Process Improvement." *Harvard Business Review* (blog), 27 September 2012 (http://blogs.hbr.org/cs/2012/09/understanding_fear_of_process_improvement.html).

[18]Kotter, John P. "Leading Change: Why Transformation Efforts Fail." *Harvard Business Review*, January 2007.

[19]Power. See 17.

[20]Marquet. See 13.

[21]Marquet. See 6.

[22]Marquet. See 13.

[23]Fox, Bill, and Samira Askarova. "Absorbed by Day-to-Day Challenges, Leaders Are Failing to Go to the Balcony, Robbing Themselves, Their Organizations, and the World." Lead Change Group, 20 September 2012 (http://leadchangegroup.com/absorbed-by-day-to-day-challenges-leaders-are-failing-to-go-to-the-balcony-robbing-themselves-their-organizations-and-the-world).

[24]Montgomery, Cynthia. *The Strategist: Be the Leader Your Business Needs*. HarperBusiness, 2012.

[25]Montgomery. See 24.

Bill Fox is a Senior Consultant and a catalyst for improvement with AEGIS.net. He believes many organizations are too quick to buy into the latest "silver bullet" without fully understanding the problem that needs to be solved and the challenges they will face in the solution selection. At the same time, many other organizations are stuck on the sidelines, afraid to even try to change, or to try again. With over 30 years' experience in project management and leading successful process improvement projects, Mr. Fox has demonstrated a unique perspective on how to approach and focus improvement efforts on the right priorities to achieve results with certainty. Even so, he has witnessed a repeated pattern of failure among many organizations. Determined to have an impact on this repeating pattern, Mr. Fox created the 5 Minutes to Process Improvement Success *interview series, which is designed to uncover and leverage the best performance improvement strategies and tactics of top consultants from around the world. This work has provided him with a unique perspective, and he is now writing about what he has learned and working with clients so they can benefit from a more nuanced, informed, and disciplined approach to improvement. Mr. Fox blogs about his findings at http://5minutespisuccess.com. He can be reached at bill.fox@AEGIS.net or follow him on Twitter at @bi11fox.*

Blending Agile and CMMI

by Brian Button and Nate McKie

At first glance, agile and CMMI seem like oil and water. Agile, used for developing small projects with small teams with the least amount of process practically possible, would appear to be in direct opposition to CMMI. Everyone knows that CMMI is a heavyweight process specification that defines hundreds of different deliverables that every team must create just to have evidence for that one time their SCAMPI[1] appraiser strolls into their offices. Indeed, there seems to be no hope of ever merging the two approaches into anything useful.

If our story stopped there, this article would be very short and would offer nothing new on this subject. Fortunately, the above paragraph is patently untrue. There are documented cases of agile and CMMI coexisting and even benefiting each other. In this article, we describe how our organization used these two concepts together in a way that played off the strengths and weaknesses of each to build something better than either approach on its own. We will also talk about a common adoption antipattern, "cargo cult" adoption of agile or CMMI, and how to avoid the mistakes it engenders.

TWO CONCEPTS, ONE CONTEXT

Before we dive into the details of how our process improvements happened, it is important that we establish a base for understanding our terms and our context. Below, we describe the essence of what agile and CMMI are, particularly focusing on those aspects of each that are important for this article. We also describe our starting point, which had a big influence on how we adopted and adapted both sets of practices throughout our journey.

Overview of Agile

Agile methods have been around for a while. Scrum was first practiced in the early 1990s, while Extreme Programming, Feature-Driven Development, and others have evolved since then. By this time we know what these methods are, we know what they do, and we know their strengths and weaknesses.

The important parts of agile for us to focus on in this article are the underlying principles and values that guide agile teams throughout their activities. The individual practices change between methodologies and over time, but the values and principles — as defined by the Agile Manifesto[2] — remain constant. These values and principles, combined with any one of the individual agile methods, provide the framework used to accomplish the work. The values, principles, and practices of agile describe to people *how* they should do the work, *how* they should examine and improve the process, and *how* they should succeed together.

Agile teams tend to tailor their own processes to be sufficient for their own needs and no more. We've been on those teams, we've experienced this, and it works really well. What this self-organization doesn't provide, however, are values, principles, and practices that help teams deal with the unexpected challenges and issues that come up from time to time. Most of these issues are things many teams don't need to think about (or simply *don't* think about) most of the time — such as maintaining a baseline configuration for workstations and software or having a conscious approach to risk management. As organizations grow larger and projects become more complex, these additional concepts become more important.

Overview of CMMI

Similar to agile methods, CMMI has been around for a while. Originally developed for the US Department of Defense (DoD) in the mid-1980s, it has passed through several iterations to reach its present state. In its current form, it defines a framework for improving your existing processes so that they contain the elements necessary, according to the Software Engineering Institute (SEI), for a well-defined, well-run project.

CMMI communicates this through a series of goals, practices, and examples that define a process improvement model. The end state of that model becomes more mature as your processes improve through different levels, with different goals defined for each level.

CMMI, however, is only a process improvement framework. As such, it is possible to use it to improve a lightweight process, such as an agile process, or to improve

a very heavy process, such as those found in many larger, more traditional software shops. The hard part is to decide which pieces of CMMI apply to your context and which pieces you can safely ignore, at least for the time being.

We were looking for a process improvement framework that would allow us to define a common set of expectations for teams but didn't mandate specific practices or activities for the teams themselves.

Initial Context

Our organization, Asynchrony Solutions, has been working in an agile fashion since about 2003. The management of the company supports the values and principles of agile in very concrete ways, as well as the practices defining XP. As an organization, we are further evolving into a lean/Kanban-driven set of processes. Most of our work is entire projects, outsourced to us by organizations that rely on us to apply our project management and development skills to build capabilities for them. Projects tend to last for three to six months in most cases, with several outliers measured in years. We do our best to involve our customers as members of our teams in any role they choose to fill.

Our company has doubled in size over the past two years, and this growth has exposed some consistency issues with regard to how individual teams operate and the levels of support management provides for each team. Given that the company is poised for more significant growth over the next year, we were looking for a process improvement framework that would allow us to define a common set of expectations for teams and their surrounding environment but didn't mandate specific practices or activities for the teams themselves. This allowed us to improve as a company while still staying true to our strong agile roots. Between this goal and the relatively short lifetimes of our projects, we could define the areas of the CMMI for Development (CMMI-DEV) that were most relevant to our context. Management was able to establish consistent expectations and levels of support around teams, while the project teams were able to focus on those goals and practices that gave them the most benefits during their short lives.

As our company grows, and as projects grow more complex, understanding the way we tailor our use of CMMI as a process improvement framework will change as well, leading us to add processes where a growth in size or complexity has added risk to the organization. But we'll leave that for another day, when we need it.

USING CMMI IN AN AGILE WAY

Our organization undertook two different CMMI appraisals. In the first, we worked toward an appraisal for Level 2 and then toward an appraisal for Level 3 roughly 18 months later. In both cases, we were able to leverage our existing agile values, principles, and practices to satisfy the majority of the CMMI-DEV process areas (PAs). Also in both cases, our internal team and the appraiser found areas of needed improvement, which we addressed in ways that maintained our agile roots.

Areas Covered by Existing Agile Practices

During our appraisals, our existing agile processes served to address most of the CMMI-DEV process areas, and almost every one of our existing practices showed up at least once. Many of the goals were completely covered, and in other cases most of the practices were already handled with only a few changes needed. As a few concrete examples, consider the following CMMI process areas:

- **Requirements Management.** Our user stories, internal project tracking tool, and visual management through Kanban boards served to strongly manage requirements.
- **Project Planning.** Various story tracking tools — such as JIRA, Bugzilla, our internal tool, and even cards on a wall — helped to make estimating easier and spread the results of the estimation process to all interested parties.
- **Measurement and Analysis.** Focusing on value and eliminating or avoiding waste through our lean/Kanban processes created an environment in which measurement is part of the way a team operates, giving management visibility into the key metrics for a team.
- **Process and Product Quality Assurance.** Collaboration, retrospectives, and facilitator involvement resulted in continuous process evaluations and improvements.

Improvement Opportunities Uncovered by the Appraisal Process

Despite the strength of our existing practices, the appraisal group — made up of internal staff and our

CMMI appraiser — found several opportunities for improvement during the preparation and execution of our appraisal. In many of the cases, these recommended improvements were in process areas that transcended individual teams and applied to the organization as a whole. These PAs had received less focus, mostly due to our structure of individual project teams and the agile principle of allowing a team to create and tailor team practices based on the needs and desires of that team.

Each of the weaknesses found represented areas where the CMMI-DEV process model uncovered missing or ineffective practices in our existing teams' processes. Our organizational task at that point was to understand what the weaknesses were, assess what risks they posed to teams and our business, and decide how to use our agile values and principles to address — or not address — each of them. Below we discuss a few areas of improvement that we found and how we handled them.

Risk Management

Internally, we identified Risk Management as a CMMI process area in which our organization could improve. Since agile is so pragmatic and so immediate, teams were only concerning themselves with risks that were currently affecting projects. They considered this acceptable because across our teams and as part of our culture, there is a confidence that we can identify and address risks as they come up. That makes worrying about risks that may never manifest themselves a wasteful exercise.

But clearly there are additional risks we must consider and mitigate as a way of ensuring future success. Some of these risks are above the level of the team, and some are strictly internal to the individual teams. Our challenge was determining how to identify both classes of risks and finding ways to encourage them to be considered. Once that happened, we needed to help teams mitigate the risks in a manner that didn't create extra work for everyone.

Organization-Level Risks

Above the level of the teams, the organizational leaders identified several consistent shortcomings that added risks to projects, including:

- Project startup was an ad hoc process that always seemed to let some key piece fall through the cracks. This made it more difficult for new teams to start up quickly and begin delivering value to their customer.
- A lack of consistent metrics across teams did not allow management transparency into the operation and progress of teams.

The first risk was addressed at the management level by creating a repeatable process used to initiate projects. This process happened before teams began working, so they did not take part in mitigating this risk. Management decided on how to involve the right stakeholders, created the infrastructure to support the teams, and so on. This was done in a way that teams can now start up and begin working quickly, since the base set of tools and processes they need to function are already in place.

The second risk, consistent cross-team metrics, is still being mitigated. The Measurement and Analysis PA contains advice about how to structure and support a metrics activity across an entire organization. Our teams are already collecting metrics at their level, but what is missing is a tie-in to organization-wide metrics that senior managers can use. Management will need to decide what is important to measure at their level to provide for transparency into and across teams. It will be up to the teams to align their internal metrics to the information that management deems important and to decide the best way to collect these metrics in a simple, noninvasive way. That is a typical adoption pattern in our company, as decisions about implementation details are delegated to teams, which are presumed to know their tools and processes best. Delegating responsibility to the team level is one of the tenets of agile methods and lean, both of which are accepted and encouraged by our management. The eventual goal is a monthly "ops review" meeting, where the teams present these metrics to management in the style adopted from lean manufacturing.

Our teams are already collecting metrics at their level, but what is missing is a tie-in to organization-wide metrics that senior managers can use.

Team-Level Risks

At the team level, the same group identified other shortcomings that added risks to project teams:

- There was a lack of traceability from requirement to released code, which made it difficult to know why a particular feature was present in the system and who had asked for it.
- There were missing decisions from teams about how they would handle releases, plan, find and fix defects, find and mitigate process-specific risk, and a dozen other activities that teams should consider as part of

their working agreements. This caused confusion as situations arose that teams had to deal with.

Since these risks are found at the team level, individual project teams are directly involved with the mitigations. This means that teams have to make a careful tradeoff between ensuring their productivity and mitigating risks. They make this tradeoff by considering the likelihood of a risk occurring and the damage that the risk could do to the project.

The defined chartering process didn't tell teams *how* they had to solve problems, it just asked teams to talk about them and record their thoughts.

The lack of traceability has already caused issues on several projects, so there is an ongoing appreciation of the value that requirements traceability (as defined in the Requirements Management PA) provides to teams. Our teams are adopting the goal of traceability on their own and are devising automated ways of tracking. This enables them to provide traceability from requirements to releases without the expense of managing that mapping manually. Again, the implementation of process details is left to teams to figure out for themselves, with the belief being that they will know best how to use their tools to reach the goal most easily and with the least amount of overhead.

The Ups and Downs of Team Chartering

Finally, management introduced a team chartering process to give teams an opportunity to make some of the internal decisions that had been missing. These included how often and through what process releases to the customer should happen, how configuration baselines should be kept and updated, what the team's policy should be on pair programming, and a multitude of other important things to consider. The defined chartering process didn't tell teams *how* they had to solve problems, it just asked teams to talk about them and record their thoughts.

This process was one of the least successful changes we tried as a company as part of our CMMI-based process improvements. Through consultations with leaders throughout the company, both at the management and team levels, a laundry list of topics to consider was created, and teams were expected to discuss all of them early on in their projects to decide how their project would be run.

Teams balked at this for two reasons. First, as far as the teams were concerned, there was little value to most of the questions. Second, the process of discussing these questions could take several days. Teams tried many different ways to make the process more palatable, but no one found a recipe for success early on.

Acting on this feedback, management tried various things to make the process easier, including providing stock answers to most of the questions. Nevertheless, the process still took a long time, and teams perceived it to be of little value on the whole.

The current solution to this problem is to decide which questions out of the several dozen actually address issues that create risk for a team and have the team concentrate on those. There may be several others that are required, due to the consequences of an unmitigated risk, but the process is primarily based on assessing the risk posed to a project and letting the team members decide how much risk they are willing to accept by initially bypassing some of the questions. During regularly scheduled retrospectives, these questions are then slowly introduced to teams as discussion points, which are added to their evolving charter. This allows for the questions to be asked in a way that engenders good conversation.

Other Improvement Opportunities

The appraisal raised other suggestions and weaknesses, some of which we acted upon and some that we chose to accept for now. As our organizational situation changes, we'll revisit these and perhaps decide to consider them more carefully.

As an example, one suggestion concerned the Verification PA. Currently we accomplish verification through a combination of the agile practices of acceptance test–driven design, pair programming, frequent demos, and continuous integration and testing. This will serve for the kinds of projects that we are doing currently, but there is concern that it will not scale should we begin to work on much larger and more complex systems.

As an organization, this is a risk that we are likely to accept. Based on what we are doing now, and based on the cost of increasing the burden of verification testing on our existing teams, it is not worth investing in this improvement as a speculative benefit. At the first sign of trouble, however, we'll invest the time and energy needed to improve that process area, using the guiding values and principles of agile to limit the implementation to what is needed.

Transformation

The ultimate goal of an organization should be to embrace agile and CMMI so well that they each become part of the fabric of the organization's culture. Decisions about how to implement processes should flow from the agile values and principles, while CMMI should be constantly consulted for improvement opportunities. In short, agile provides the *how*, while CMMI provides the *how to do it better*.

The agile values and principles guide a team or organization into building a minimal process that supports its needs. The principles help an organization focus on finding good people and putting those people together in a way that maximizes their chances for success. Following the principles and values, teams are given opportunities to collaborate with stakeholders and customers frequently, and they learn to ship software early and often. They plan frequently and with an eye toward shorter time horizons to support those frequent deliveries. And teams are encouraged to periodically take a step back from their work and consider *how* they work, using the CMMI process models to help uncover and drive appropriate changes.

CMMI process models help organizations identify potential shortcomings that may exist in their processes and give them a way to picture what improvements may be helpful. Teams can be constantly appraising their existing processes and suggesting changes based on the CMMI process areas. Then these improvements can be implemented using agile techniques to provide gradual, incremental improvements with measurable results.

Our company has successfully made agile part of its culture. It colors everything we do. We are not there yet with CMMI. It will take time, effort, and education to make it a primary part of our process improvement toolkit. The fastest road to assimilating CMMI into our culture lies in incorporating it into our retrospective practices. Currently, our teams use regular retrospectives as process improvement opportunities, but those improvement opportunities are ad hoc. By training our retrospective facilitators in CMMI, they will be able to use its process improvement framework as a way to inform teams about potential improvement possibilities. Then gradually others will come to suggest changes from it, until it becomes one of our regular sources of process improvement ideas.

CARGO CULT ADOPTIONS OF AGILE AND CMMI

Perhaps the most ineffective and costly implementations of agile and CMMI are so-called cargo cult adoptions.[3] In cargo cult adoptions of either approach, organizations and people follow the forms without understanding why they are doing so. This leads to lots of rituals and ceremonies around everyday activities but no value added.

It is easy to identify organizations that are stuck in this behavior. If you look at what they do, it is exactly by the book. They adopt the forms of their method without appreciating its finer points. For agile teams, user stories are written exactly following the prescribed format: "As a *user* I want to *perform an action* so that I can *get some value*," but the user is a programmer or a product owner, not a real user, or the stories are months long. Stand-up meetings happen every day but have turned into status meetings that last for an hour. CMMI adoptions have the same problems — organizations read the CMMI documentation and assume that it is *all* mandatory. They implement it exactly as written without regard to how it should apply to their own environment.

Recovering from a cargo cult adoption is difficult, as first you have to realize it is happening. And organizations that are stuck in this situation usually don't realize it.

From the organizational point of view, adoptions of this type almost always lead to failure. Management gets large numbers of people involved in creating a change, usually without providing them any training or involving any experts, and months or years are wasted as the organization struggles to see some improvement. The net effect of such failures is that an organization's employees tend to become jaded about all process change initiatives, making the road harder for other changes yet to come.

Recovering from a cargo cult adoption is difficult, as first you have to realize it is happening. And organizations that are stuck in this situation usually don't realize it. Appreciation of their problems typically involves either a flash of insight from someone internal or from an external voice. In either case, understanding that the organization is stuck in place is the first step in fixing

the problem. From there, a strong change process, including education, can help to start things moving along. Chances are, though, that the culture in that organization is going to be extra resistant to further change, causing more problems throughout the transition.

CONCLUSION

The intent of this article was to show the reader that it is possible and practical to combine agile and CMMI in a way that is beneficial to the organization at large. Agile methods can provide the process for a team, creating a fast-moving engine of value creation, while CMMI can be used to help that team improve itself. At the end of the day, once both agile and CMMI are thoroughly understood and adopted by an organization, they can work together to create a culture of continuous improvement, raising the achievement levels for any team.

ENDNOTES

[1]"SCAMPI" stands for "Standard CMMI Appraisal Method for Process Improvement."

[2]Agile Manifesto (http://www.agilemanifesto.org).

[3]"Cargo cult" (Wikipedia).

Brian Button is the VP of Engineering and Director of Agile Methods at Asynchrony Solutions, Inc. Joining the company in 2006, Mr. Button instituted an Asynchrony Center of Excellence, through which he has been working to improve the effectiveness of internal teams by evangelizing Agile, Lean/Kanban, and technical excellence across the organization, as well as working to spread those same concepts to Asynchrony's clients. He has been in the software industry for 25 years and is a 12-year veteran in Agile practices. He can be reached at brian.button@asolutions.com.

Before starting Asynchrony in 1999, Nate McKie had 12 years' industry experience in corporate development and project management. In 2003, Mr. McKie became interested in Agile methods as a way of harnessing the amazing R&D work that Asynchrony was doing into a mature process that can create ready-to-ship applications for its customers. Since then, Asynchrony has become a leading practitioner of Agile techniques and ideas and has spread the disciplines of quality code and rapid implementation throughout its commercial and government customer base. Mr. McKie coaches teams in Agile techniques and has taught classes in Agile and test-driven development to various Asynchrony clients. He can be reached at nate.mckie@asolutions.com.

Agile CMMI: The Real Underlying Obstacles to Effective Integration and What You Can Do About Them

by Paul E. McMahon

Why is it, even after articles and books have been written and success stories shared explaining how CMMI and agile can work effectively together, that perceived conflicts continue to exist? Is it possible the conversation hasn't gotten to the real underlying obstacles to effective integration?

It isn't difficult to explain in theory how CMMI and agile can work effectively together, but long-held misperceptions related to what CMMI actually requires will trump theory every time. In this article I share two real case studies[1] that demonstrate why effective agile CMMI integration is turning out to be quite a bit harder than what the theory suggests.

Let's start with the theory, before we examine more closely the misperceptions and the underlying obstacles.

AGILE CMMI INTEGRATION THEORY

Contrary to popular opinion, the CMMI model is not a set of dictated practices, which is, unfortunately, how it is too often applied today. CMMI is an improvement reference model intended to help you ask the right questions, thus leading to the best decisions for your organization. Some examples of the questions CMMI prompts you to ask are:

- Do you have adequate resources on your project?
- Do your people have the right skills and competencies?
- Are you involving the right stakeholders at the right time?

These questions are methodology agnostic — they can help any organization using any methodology. But just using an agile method will not lead you to ask these questions. They are examples of how CMMI can help an agile approach. On the other hand, agile methods provide good *potential* "how to" alternatives to traditional practices. I want to stress the word "potential" because these practices will not work in all situations. For example, a daily stand-up meeting may not work well if part of your team is distributed across different time zones. Depending on the product, your customer may not be able to accept frequent product releases. Embracing change could be risky if you have fixed cost and schedule and a noncollaborative customer. What you should be observing is that CMMI can help you understand what you must do, while agile methods provide many good options for how to do it.

Now let's take a look at two case studies, which will help us understand why implementing the theory turns out to be more difficult than it might sound.

CASE STUDY 1

As hard as it may be for some to believe, this first case study provides an example of an organization that used CMMI to increase its agility. LACM is a successful high-tech organization that focuses on the US defense market. In 2007 the company experienced its greatest success. With over 50 active projects, only two were experiencing any difficulty at all. LACM achieved a CMM Maturity Level 3 many years ago and a CMMI Maturity Level 3 in 2008. The 2008 improvement effort was motivated by a VP who understood how he could use CMMI to address changing customer needs and improve his organization's agility at the same time. I was brought in as a consultant to help LACM with their 2008 CMMI improvement effort.

Where Do You Start to Improve Performance and Agility When Using CMMI?

There isn't a single required starting point when using CMMI, but the approach LACM took may be the best I have ever observed. I will explain it by sharing four techniques used at LACM along with common CMMI misperceptions that hinder many organizations' agile CMMI integrations.

The VP driving the CMMI initiative at LACM knew his organization needed to change, but he was cautious

because he didn't want to break what was working. So he began by asking his directors a few questions:

- "Why are our customers coming back to us now over the competition?"
- "What is the unique value our organization brings to our customers?"

These questions prompted a series of presentations and open forum discussions at lower levels of the organization, leading to Technique 1.

Technique 1: Start by Asking Key Questions to Focus on Specific Objectives

This isn't the way many organizations that use CMMI start out. Lots of first-time CMMI users begin with the common non-agile approach of just going out and implementing processes to meet all the CMMI expected practices. The reason this isn't recommended is that most organizations don't have unlimited process improvement budgets, and this approach often leads to the adoption of practices that aren't adding real value. Notice right here the conflict with agile.

A better way is to first establish specific objectives for your improvement effort. The CMMI model actually contains a specific practice to help you establish objectives, and through experience, my colleagues and I have learned that these objectives should be specific in order to help teams focus their improvement efforts where they need help the most. Let's look at a measurement example.

What Data Should You Collect?

Many CMMI-using organizations take the non-agile approach to data collection by first gathering data related to each process area in the model and then trying to figure out how they might use it.

In contrast, the agile approach is to begin by asking a couple of questions:

- Who will use the data if we collect it?
- How does the data we intend to collect relate to our specific objectives?

This method makes sense whether you are beginning a new CMMI effort or you are already using the CMMI model and just want to improve.

Why Involve Your Workers?

Why did that VP involve his lower-level workers in process improvement discussions? Because he recognized his workers knew best how the real processes worked.

The open forum discussions led to a few key realizations. First, the workers realized that the company gained great value from product reuse but that most of their processes had been written for new development. They also learned why there had been some recent employee turnover. In exit interviews, a number of employees said the reason they were leaving was that they had been hired to do a job but were given very little relevant training for that job. What is interesting about this is the fact that LACM had a very extensive training program, but the training being offered did not align with the real issues employees faced on the job. This leads to Technique 2.

Technique 2: Pruning the Processes

LACM built flow diagrams of what people really did to get their work done. These were not theoretical diagrams. They annotated these diagrams with assets from their process repository that the workers said they used to do their jobs. Any asset in the process repository that didn't end up on a flow diagram became a candidate for elimination. But first we asked a few more questions:

- If no one is using an asset in the process repository, why is it there?
- Are we wasting time training people in the use of certain assets that no one uses?
- Do we believe if people did use these assets, it would help them get their jobs done?

This effort led to a streamlining of the processes and supporting process assets in LACM's organizational repository. It also improved training by aligning it with how people really worked.

You may be getting concerned at this point that this technique could add risk to your CMMI maturity rating. This is due to a common CMMI misperception that I will explain shortly. But first, let's look at an example.

Pruning the Peer Review Process

LACM's peer review process demanded that a great deal of data be collected about each found defect. It also required periodic analysis of the collected data. When we built the flow diagrams, they showed that people were entering all this data because the peer review tool (which was mandated on all projects) required it, but no one was going back and analyzing the data. On further investigation, we discovered that the data collection and analysis activities had been added because someone wrongly thought CMMI required them. This onerous peer review process had actually discouraged people from entering real defects found. The pruning of the

peer review process not only led to improved performance and agility, it also improved CMMI implementation because people started using the peer review process more consistently.

Insight

Historically there has been a tendency to read things into the CMMI model that aren't really there. This has created unnecessary non-value-adding work. By using CMMI as intended — by which I mean using it to ask the right questions, which will lead to the best decisions for your organization — you can improve your CMMI implementation and increase your agility and performance at the same time.

Technique 3: Use CMMI Less Formally Before Using It Formally

LACM used the CMMI model first to discover where they needed changes to improve their performance. Then they prioritized the changes and focused on the most valuable improvements, considering cost and benefit. They effectively applied CMMI with an agile spirit.

Why don't more organizations do this? I have never heard anyone say pruning wasn't a great idea. Yet there are two reasons I have observed why more organizations don't prune their processes:

1. Too often, CMMI is only used when an organization is undergoing a formal appraisal. When process improvement teams face this pressure and aren't given adequate time to locate and work real issues, organizations rarely achieve real performance gains.
2. Effectively pruning your processes requires the participation of the people in your organization who really understand how the work is done. Often these people are the best performers and the busiest people in the company.

An Alternative Method of Process Pruning

I frequently hear people say that they could never get their management to approve a pruning effort because it would cost too much and they wouldn't be able to convince them of the payback. One alternative that I have seen work is for the process team members to build informal relationships with key technical experts on the projects. Through informal conversations with these experts, you can extract specific, small opportunities for pruning improvements that can more easily be cost-justified. Pruning your processes a little at a time is similar to refactoring your software a little at a time. After a while, you might find that you are gaining the same high payback.

What CMMI *Doesn't* Require of Your Process Repository

The techniques we have been discussing lead to a key question: with all this pruning going on, how do we ensure we aren't jeopardizing our CMMI rating by eliminating process assets that may provide key evidence during an appraisal? To answer this question, you need to understand what CMMI requires with respect to your process repository, and you need to understand another common CMMI misperception.

Some wrongly believe that CMMI requires a heavyweight process repository superstructure — it does not. Consider the following two examples.

LACM is a large product-centric organization. They mandate certain tools and standards because of the nature of the product they build, and they do have some detailed work instructions because these are needed to support tools LACM uses in the development of their products. BOND[2] is a small organization that focuses on providing software services. They mandate few tools, few standards, and have no detailed work instructions. This is necessary so that BOND can work in an agile fashion in many different environments. The process repository structures at LACM and BOND are very different, but both companies have achieved CMMI Level 3. *The process repository structure depends on the business need, not on specific requirements mandated by CMMI.*

That said, if you want to be agile and maintain your CMMI rating, I strongly encourage you to use the following technique.

Technique 4: Keep Your Process "Must Dos" Packaged Separately from Your Guidelines

Your process "must dos" are the bare minimums that everyone must do on every project. This should be a very lean set of requirements. Then you will use "tailoring up" to add whatever else you require based on your project needs. This approach is key to ensuring you have met all the expected practices in the CMMI model.

What is interesting is that the "must dos" are actually a much smaller set of requirements than many believe. The *CMMI for Development: Guidelines for Process Integration and Product Improvement* book[3] is over 600 pages long, but much of this material is explanatory text. When the BOND organization achieved their CMMI Level 3, for example, no process document was more than two pages. Furthermore, you do not need distinct processes related to each process area in the CMMI model, as some mistakenly believe. In the BOND case, the total number of pages of must-do processes

was less than 30, yet they still achieved a full CMMI Level 3 rating.

So why don't more organizations use the CMMI model this way? To help us understand, let's look at another case study.

CASE STUDY 2

RAVE is a large CMMI Level 5 organization also focused on the US defense market. In 2005 RAVE recognized that a stealth agile movement was taking place inside their organization. By "stealth agile," I mean that software teams were starting to use agile methods even though the company's formal CMMI Level 5 processes didn't recognize them.

RAVE knew they needed to do something, but they decided on a different approach to integrating agile methods into their existing processes than the one I just described. Rather than modifying their existing processes, they handled agile methods through their normal tailoring-down process supported by an agile developer's guide.

Technique 5: Consider a Developer's Guide to Aid Agility

Tailoring your processes is an expected practice within CMMI, but the CMMI model does not specify how this tailoring is done. You can tailor down by deleting standard practices that don't apply to your project situation, you can tailor up by adding practices, or you can use a combination of both. The agile developer's guide approach RAVE used allowed them to add agile practices during this tailoring process while they were also deleting other practices that did not apply.

There are advantages and disadvantages to the developer's guide approach.

Advantages

One advantage is it adds no risk to proven CMMI Level 5 processes, and it doesn't require key personnel to help with pruning. A developer's guide is also less costly and requires less retraining due to formal process changes.

Disadvantages

On the other hand, if you are hearing that your processes don't help your people with their real jobs, or that the current process requires extra work without providing value, then this approach isn't going to help. It could also result in an overall increase in work, including product reviews and progress reporting. With this approach, you may run the risk of tailoring out practices that might be critical to your CMMI rating.

CMMI MISPERCEPTIONS TO SEEK OUT AND COMBAT

1. CMMI is a set of dictated practices.
2. You must collect data related to every CMMI process area.
3. The CMMI model should only be used for formal appraisals.
4. CMMI requires a heavyweight process repository superstructure.
5. You need distinct processes for each process area in the CMMI model.
6. CMMI Level 5 means you are perfect and you don't need to keep improving.

CASE STUDIES 1 AND 2: THE REST OF THE STORY

Now let's look again at both of the case studies to learn how each turned out.

Living Agile on Level 3

LACM was a CMMI Level 3 organization that used selective CMMI Level 4 and 5 practices (quantitative project management and causal analysis and resolution) informally to achieve some very specific performance goals. This organization had a particular problem related to getting hardware ordered and installed on time. To address this problem, one project team derived specific measures, isolated the root cause, and implemented a corrective action. They effectively applied Level 4 and 5 practices with an agile spirit. This effort succeeded, and by "succeeded," I mean management saw a measurable improvement in project cost and schedules due directly to the corrective action the team implemented.

"Stuck" at Level 5

In contrast, RAVE, the CMMI Level 5 organization that used the agile developer's guide and tailoring-down approach, fell short of their goals.

Like LACM, RAVE wanted to improve, and to that end they funded a lean Six Sigma team to create the agile developer's guide. But the reason they took this approach was that they did not want to change their CMMI Level 5 processes out of fear that doing so

might risk their Level 5 rating. Rather than improving performance, RAVE's adoption of the agile developer's guide actually resulted in an increase in cost and reduced performance. This was because the new practices in the developer's guide were added on top of the mandated CMMI Level 5 processes rather than used to streamline the organization's processes.

ATTAINING LEVEL 5 DOESN'T MEAN WE DON'T NEED TO KEEP IMPROVING

The misperception that CMMI Level 5 means that you are perfect and cannot change is exactly the opposite of the level's intent and one of the primary reasons we continue to see conflict between CMMI and agile. The following words are taken directly from *CMMI for Development: Guidelines for Process Integration and Product Improvement*:

> At this point, of the many organizations that have been evaluated at CMMI Level 5, too many have essentially stopped working on improvement. Their objective was to get to Level 5 and they are there, so why should they keep improving? This is both an unfortunate and an unacceptable attitude ... it is unacceptable because the essence of Level 5 is continuous improvement.[4]

Agile practices can not only coexist with CMMI Level 5 practices, they can aid the intent of CMMI Level 5, which — as noted — is continuous improvement. As an example, Scrum's sprint retrospective is a great way to identify and implement actionable improvements similar to what the team at LACM did to solve their hardware order/installation problem.

I must point out that agile methods will not provide everything you need at CMMI Level 4 or 5, where the focus becomes quantitative management and continuous improvement. For example, they will not help you with your process performance baselines and models. They can, however, help you achieve your true intent — real performance improvement.

MOVING YOUR OWN AGILE CMMI INTEGRATION FORWARD

When you are trying to decide which agile CMMI integration method is best for your organization, consider that the right answer could be a combination of both the LACM and RAVE approaches. For example, you could create an agile developer's guide, do some isolated pruning, and start to modify your tailoring process toward a less risky tailoring-up approach.

If you are an agile expert, learn about CMMI and talk to the CMMI people in your organization so you can jointly discover how these initiatives can work together. If you are a CMMI expert, learn about the latest changes in CMMI v1.3 and explore the agile options that can help your organization. If you are a process specialist, even if you can't fully employ the techniques LACM used, consider building relationships with technical experts in your organization and looking for measurable improvement opportunities. If you are a manager, seek to understand the forces that are holding back your agile CMMI integration and search out and combat misperceptions.

Changing to a more agile approach is difficult in most organizations, especially those with long histories of entrenched traditional ways of working. Yet real improvements can be made as a series of small changes. CMMI and agile proponents should recognize that both groups are striving for common goals and should help each other. Continual open discussion and a willingness to keep learning are key to finding the right level of structure and agility for your organization.

ENDNOTES

[1]All case studies referenced in this article are discussed in greater detail in: McMahon, Paul. *Integrating CMMI and Agile Development: Case Studies and Proven Techniques for Faster Performance Improvement*. Pearson Education, 2011.

[2]McMahon. See 1.

[3]Chrissis, Mary Beth, Mike Konrad, and Sandy Shrum. *CMMI for Development: Guidelines for Process Integration and Product Improvement*. 3rd edition. Pearson Education, 2011.

[4]Chrissis et al. See 3.

Paul E. McMahon, Principal, PEM Systems, has 23 years' industry experience as a software developer, team leader, and manager. For the past 15 years, he has been an independent consultant focusing on coaching project managers, team leaders, and software professionals in the practical use of Lean and Agile techniques in constrained environments (e.g., physically distributed teams, CMMI compliance requirements, corporate governance requirements). Mr. McMahon takes a hands-on approach with his clients, providing tailored workshops and individual coaching as the situation warrants. He is a Certified ScrumMaster and a Certified Lean Six Sigma Black Belt. Mr. McMahon has published over 40 articles on software development; has written two books, Integrating CMMI and Agile Development: Case Studies and Proven Techniques for Faster Performance Improvement *and* Virtual Project Management: Software Solutions for Today and the Future*; and cowrote the forthcoming* The Essence of Software Engineering: Applying the SEMAT Kernel. *He has been a member of the* CrossTalk *editorial review board since 2004 and a leader in the SEMAT (www.semat.org) initiative since its inception. He can be reached at pemcmahon@acm.org; website: www.pemsystems.com.*

CMMI vs. Scrum? No — CMMI + Scrum!

by Jeff Dalton

When searching for inspiration, I turn to Charlie Parker.

Charlie "Bird" Parker was a brilliant, though seriously flawed, pioneer of the American jazz scene. While his personal challenges are well documented, Parker is known as the undisputed master of the idiom, who understood that being great was not simply about being talented. His legacy, regarded by music historians as one of the most powerful in the history of American music, was not an accident. It was arrived at through years of disciplined study, practice, and experimentation, which resulted in the very definition of the art form. But it was not simply about study, nor was it only about a disciplined adherence to structure. Parker knew what so many in software engineering still struggle to understand today: that great accomplishments are achieved through mastering and synthesizing all three elements — talent, learning, and discipline.

What really set Bird apart was his ability to master these concepts with extraordinary agility. When he performed, he heard something very different from the music of his predecessors. While standing firmly on the shoulders of the giants who came before him, he created a new, agile style that catapulted jazz into an entirely new dimension. With its short bursts of creativity, rapid real-time adaptations, and incremental, iterative improvisational character, this style — which came to be known as "be-bop" — would be better described as real-time composition.

Unlike the music that preceded Parker's 1939 debut, his was incremental and iterative in three dimensions. The first was internal to the skills of any accomplished musician, who learns to hear the sound in the split-second it takes for it to escape his or her instrument, and then incrementally *inspects and adapts* the tone, inflection, and pitch — sometimes before the sound wave has even reached the audience. The second dimension was the *real-time collaboration* among and between the members of his group: between saxophone and piano, between drums and bass, between piano and guitar, and a *continuous build* of those collaborations across the ensemble. The magic of be-bop is in the real-time composition created when a group of accomplished players collaborate as a team, *fail fast*, and deliver the *minimum viable product* throughout the course of the composition. Finally, Bird would *collaborate* with his audience — reading their reaction, inspecting and adapting, and recalibrating his compositions to meet the desires of his fans.

If it sounds as though I'm saying that agile methods have been around a lot longer than Scrum, XP, and the spiral model, I am. While I have immense respect for the authors of the Agile Manifesto,[1] they were 60 years behind Bird.

The lessons from Charlie Parker are as relevant to agile teams today as they were to musicians in the 1940s — most software organizations still struggle with the synthesis of talent, learning, and discipline. It's not for lack of trying. The landscape is littered with models, techniques, and tools in search of software's perfect chord, yet we continue to struggle with the processes required

Figure 1 — The first agile teams.

to improve productivity and increase the predictability and stability of software projects.

The theme of this issue is "Agile CMMI: Why Isn't This Conversation Dead Yet?" It's a great question, and one that I ask myself frequently when, for the 20th time each week, someone asks on my blog,[2] "Are CMMI and agile compatible?" For five years, I've dutifully answered the question with a resounding "Yes!" and a full explanation of how it might work. However, instead of exploring that question yet again, I suggest that we are focusing on the wrong question altogether.

I would posit that the right question is "CMMI and agile together: why aren't we having *this* conversation?" In a community whose most vexing problem is how to make agile work in a business that is decidedly *not* agile, it would seem that characteristics inherent in the architecture of CMMI — robustness, scalability, predictability, and structure — would be useful in scaling agile to the enterprise.

It's not that CMMI and agile don't work well together; of course they do. Given the fact the agile exists to improve the products we build, and CMMI exists to improve how that work is accomplished, why are we *not* using them together? Why not embrace CMMI to make agile better?

WHY NOT EMBRACE BOTH! A HISTORY

Agile *puristas* scoff at the idea. CMMI process teams shake their heads, struggling to understand how they can simultaneously "get a level" and be agile. In our Software Engineering Institute (SEI) Technical Note "CMMI or Agile: Why Not Embrace Both!"[3] my coauthors and I argue that there is no real conflict between these communities — only a complex misunderstanding based on an unfortunate confluence of events. Indeed, there are not two communities; there is just one, whose only desire is to bring excellence to software engineering. Think about it. Both CMMI and agile methods exist for the same reason: to help us build better products.

In the 1980s, before agile was "agile," and almost 15 years before CMMI reached maturity, the SEI sprang forth as a federally funded R&D center, forging a partnership between government and academia whose focus was on the research and development of new ways to improve the state of software engineering. While the SEI has done a remarkable job expanding its scope over the years to include other models and technologies — most notably CMMI for Services (CMMI-SVC), People-CMM, and CMMI for Acquisition (CMMI-ACQ) — it is the Capability Maturity Model Integration for Development (CMMI-DEV) that has been its crown jewel and most successful product for many years.

When considering the perspective of the research scientists at the SEI during that period and the methods that were dominating the software industry at the time, one might forgive them for not promoting agile methods, such as Scrum, which had not yet been established. That the CMMI hasn't done so since is not the result of an oversight or lack of understanding, as its critics often suggest. Rather, CMMI is a method-agnostic model that is more akin to a *behavioral improvement model* than an *engineering process improvement model*. It was never intended as a process to be followed, but as framework to improve the methods we *are* following. And to those who continue to believe that CMMI is about "filling out forms" and "getting a level," I would ask, how do those things improve the performance of the methods software teams are using?

CMMI was never intended as a process to be followed, but as framework to improve the methods we are following.

The language embraced within the CMM for Software (SW-CMM), and later CMMI, was based on the prevailing language of the day, such as "Work Breakdown Structure," "Configuration Status Accounting," "Project Audits," "Technical Data Package," and others that are still present in the latest version of CMMI-DEV (v1.3). For agile evangelists, this language appears to clash with agile values and has led to the persistent questions about CMMI's compatibility with agile. These questions about compatibility are rooted in the perception, voiced by many in our industry, that CMMI was intended for use in traditional waterfall-style project environments.

Members of the agile community routinely use the phrases "top-down," "command and control," and "low-trust environment" to describe both CMMI and waterfall-style methods, but associations between those phrases and CMMI are incorrectly applied. In order to understand how software professionals came to see CMMI as analogous to waterfall, and seemingly at odds with the values of the Agile Manifesto, we must consider the time frame and the predominant methodologies that existed when SW-CMM, and subsequently CMMI, were being developed. While researching and compiling their work, the staff at the SEI — who, like agile teams, are pretty good at collaboration themselves — enlisted the expertise

and input of a broad cross-section of organizations that were demonstrating success in software engineering. Many of these organizations were running successful, large-scale programs using what we now call "traditional" or "waterfall" lifecycle models, and many of these organizations became early adopters.

As CMMI became more popular, so did the case studies of the early adopters, thus setting a standard for CMMI implementation that mirrored their successes. The US federal government's mandate (now expired) that its software suppliers achieve a level of CMMI drove later adopters to emulate their predecessors, and thus a pattern was established for everyone to follow.

Today, a growing number of agile software teams are challenging the conventional wisdom that continuous process improvement requires substantial investments in overhead and documentation. Instead, they favor real-time collaboration, information radiators, colocated team rooms, and increased transparency within the team. These agile values are not in conflict with the intent of CMMI. They may be in conflict with the values of the early adopters, but their use of the model was only a single instantiation from an unlimited set of possibilities. Those early users were simply applying CMMI in the way it was intended — to make what they were already doing better!

Agile methods, like more traditional waterfall methods, are a collection of values and techniques that are applied in order to accomplish a task. Instantiations of agile such as Scrum encapsulate those values and techniques into a system that delivers value to customers in an iterative, collaborative, and transparent way. CMMI is neither a set of values, nor a set of techniques. It's a set of guidelines intended to make any software development approach — whether it be agile or traditional — more efficient, effective, and predictable.

CMMI CAN IMPROVE AGILE

As I often say to my students, CMMI is many things to many people.

Some see it as a way to measure organizational performance. Others see a requirement for bidding on federal contracts that only adds overhead. Some others see it as a giant checklist of "must have" process steps, while still others see a set of guidelines for continuous improvement.

CMMI-DEV is organized into a hierarchy of 22 process areas (PAs), each with multiple goals, practices, and subpractices. While these process areas contain clusters of related practices, they are not processes, and it is necessary to trace a path between multiple PAs and practices in order to "follow a thread" as one establishes a localized model that can be applied to an organization's instantiation of agile methods.

Traditionally, and all too often, companies take a "CMMI-centric" approach to process development that is model-focused rather than value-focused. This results in a process with excessive "process debt" that does not reflect the values of the organization.

Consider the process flow in Figure 2, which takes a model-centric approach to defining part of a planning process. We often see something like this when a process is designed to "comply" with CMMI.

I prefer to interpret CMMI as a set of questions about the environment in which we choose to work. The answers to those questions will determine the experience that

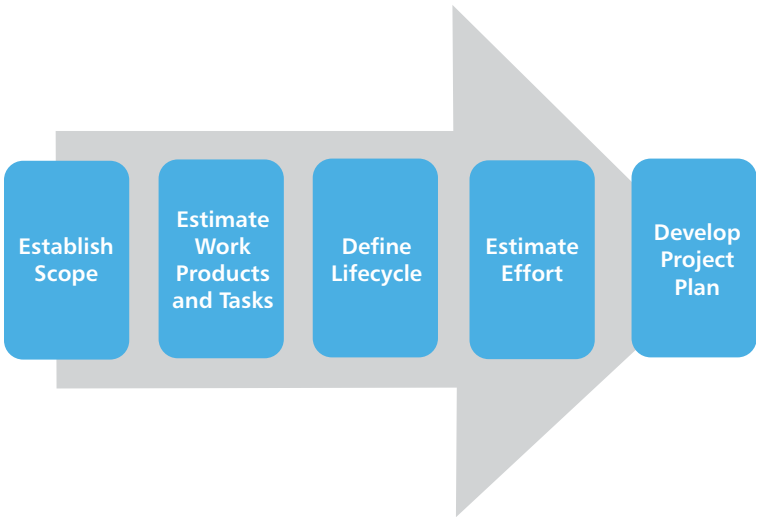


Figure 2 — Linear, model-centric processes don't reflect agile values.

practitioners will have, and they will also affect the quality of a software team's products and the predictability of project delivery.

The questions must be asked within the context of team norms and the methods and techniques that the team has agreed to adopt. Interestingly, this is precisely how we should conduct CMMI-based SCAMPI Appraisals for *any* type of software development organization.

A value-focused approach for an agile project might seek to answer the following questions, derived from CMMI, about planning for releases and sprints:

- How are we projecting our ability to deliver value?
- How long will our sprints be? How many sprints are in a release?
- Is our team clear on how we'll interact with project stakeholders?
- Do we know what we need to produce during the sprint?
- Have we broken down our stories into tasks?
- How will we know how our sprint is progressing?
- What keeps us up at night? Where do these risks originate?

Figure 3 shows the process flow from Figure 2 reimagined as a value-focused process using questions derived from CMMI but translated to the local team's language.

A value-focused approach is more concerned with the natural flow of the work that is to be accomplished and less concerned with adherence to a linear, step-by-step process, letting a task management framework such as Scrum drive the sequence. In this way, the "process assets" evolve into a set of *process objects* that have encapsulated within them a robust set of characteristics that enable them to be adopted by the enterprise and improved over time. These process objects are instantiated as needed, based on the goals and objectives of the organization and the desire to improve on the highest-priority aspects of agile performance.

How can we strengthen Scrum so that it can scale? The solution lies not in applying CMMI directly to Scrum, but in asking the "CMMI questions" of an organization's Scrum ceremonies.

The practices that CMMI makes available to us include those that help bring greater clarity and strength to the Scrum ceremonies themselves (the "specific practices," or SPs), and those that help strengthen the understanding, adoption, and continuous improvement of the agile values and behaviors (the "generic practices," or GPs). Scrum ceremonies are loosely defined, with implementation left to self-organizing teams. While teams that are performing Scrum with integrity would be loath to accept rigid process oversight, the fact remains that successful implementation across the enterprise has proven elusive to many companies, especially where organizations have attempted to scale agile by applying a "Scrum of Scrums" framework. How can we strengthen Scrum so that it can scale?

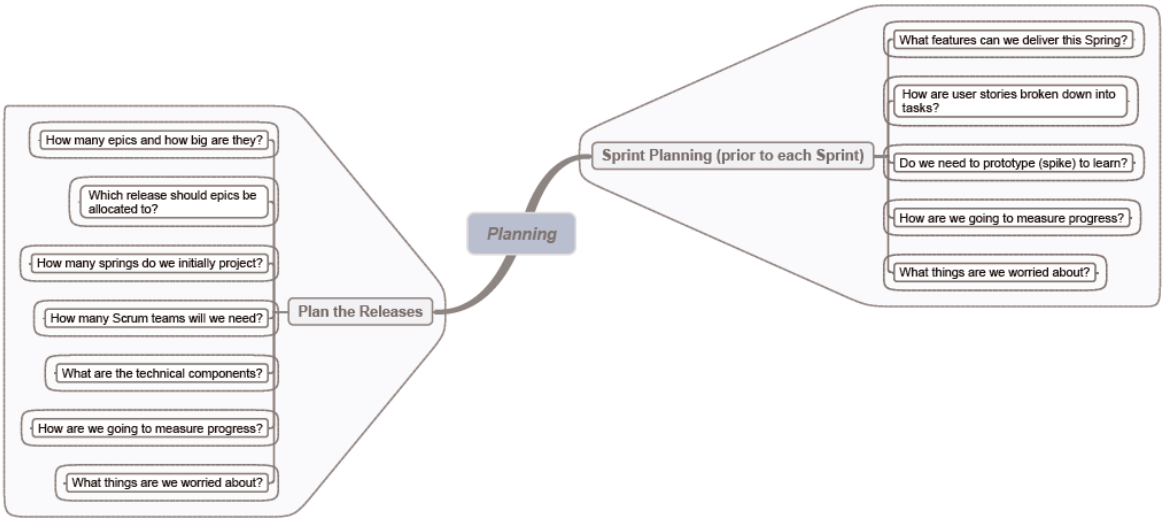


Figure 3 — Agile planning is iterative and incremental.

The solution lies not in applying CMMI directly to Scrum, but in asking the "CMMI questions" of an organization's Scrum ceremonies. For example, the following questions are derived from CMMI:

Release Planning

- How many epics should be in a release?
- How are epics allocated to each release?
- How often are releases scheduled?
- How are releases organized into a definable lifecycle that we can communicate to our business customer?
- How many sprints are contained within a release?
- How many story points are projected to be in each release?

Sprint Planning

- How long are our sprints?
- What criteria determine the length of sprints?
- How are stories allocated to each sprint?
- What criteria are used to determine allocation?

Sprints

- What design artifacts are useful to our teams?
- How are effective code reviews conducted?
- What criteria are included in our definition of "done"?

Retrospectives

- Which stakeholders should participate?
- How does the information gathered at retrospectives help the rest of the organization?
- What categories do we use to brainstorm improvements?

The mind map in Figure 4 ties these ceremonies, and others, to CMMI process areas, and the questions are derived from CMMI's SPs.

For a more detailed example, let's examine the retrospective. In Scrum, the retrospective is intended as a method for the team to learn lessons, in near real time, from the most recently executed sprint. Scrum itself gives little guidance, but many teams discuss the success or failure of the sprint, how closely they met their projected velocity, how well they collaborated together,

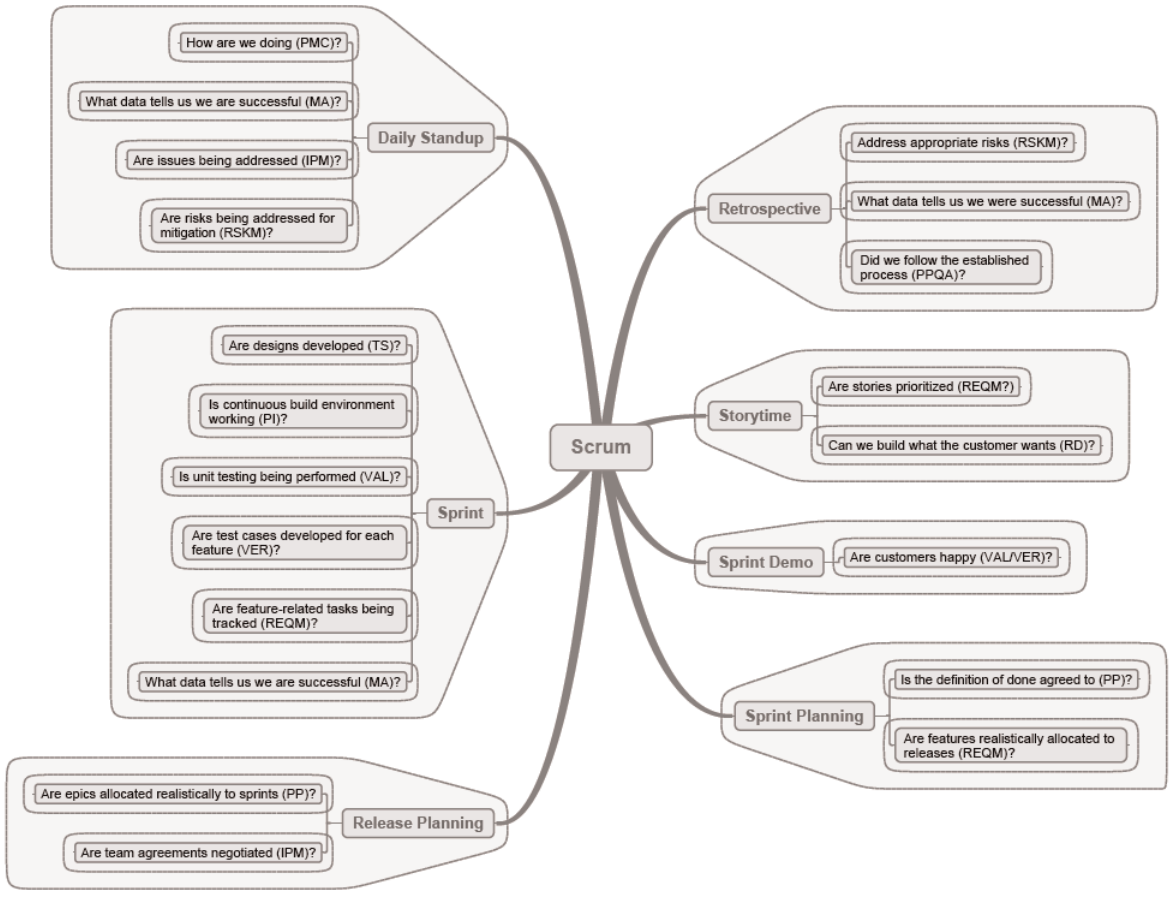


Figure 4 — Scrum ceremonies and events can be improved through questions derived from CMMI.

feedback from the sprint demo, and other sprint-related experiences. With a self-organized team, the retrospective can be very effective in improving performance. But the rest of the organization learns little from this exercise, and changes based on the experience of the Scrum team often do not translate into benefits for other teams.

CMMI provides guidelines for the conduct of retrospectives in GP 3.2: "Collect Process Related Experiences." The guidelines exist in every process area and can be applied to all Scrum ceremonies and activities being performed by Scrum teams.

By using CMMI to drive structured brainstorming, retrospectives become more powerful and provide increased value to the rest of the teams in the organization. Using the mind map in Figure 4, or better yet, a Scrum team's own interpretation, a team can categorize retrospectives into logical buckets and use questions from the applicable PA to enrich their learning. For example:

- **Technical lessons** — practices from Technical Solutions and Product Integration to drive discussions
- **Sprint lessons** — practices from Project Planning, Project Monitoring and Controlling, and Requirements Management
- **Testing lessons** — practices from Validation and Verification
- **Scrum performance lessons** — CMMI's generic practices

Teams may want to take care not to overload their retrospectives. They might consider rotating topics from one sprint to the next, or from process area to process area, and then share the information outside of the team so that others can learn and benefit from the experience. Collaboration exists beyond the Scrum team, and it's the key to enterprise-wide performance improvement.

MASTERY AND SYNTHESIS OF TALENT, LEARNING, AND DISCIPLINE

For more than a quarter of a century, software organizations have launched one attempt after another to establish a useful framework for effective continuous improvement. Both CMMI and agile methods have had a positive effect on the software industry, but neither approach has yet to succeed in driving the industry-wide levels of adoption we need in order to claim victory. Why not combine the best of each?

The entire enterprise can leverage CMMI to make agile methods like Scrum better and more powerful. Instead of trying to achieve "a level of CMMI," embrace CMMI to improve what is already being done within the Scrum team, to scale Scrum to the enterprise, and to expand the scope and influence of agile methods from the team room to the boardroom.

Seventy years ago, Bird proved that the integration of agility with talent, learning, and discipline can break down the barriers to greatness, and in doing so he created a revolution in music that lives on to this day. We can do the same with agile.

ENDNOTES

[1]Agile Manifesto (http://www.agilemanifesto.org).

[2]*Ask the CMMI Appraiser* (http://asktheCMMIAppraiser.com)

[3]Glazer, Hillel, Jeff Dalton, David Anderson, Michael Konrad, and Sandra Shrum. "CMMI or Agile: Why Not Embrace Both!" SEI/Carnegie Mellon University, November 2008.

ADDITIONAL READING

Sims, Chris, and Hillary Louise Johnson. *The Elements of Scrum*. Dymaxicon, 2011.

Chrissis, Mary Beth, Mike Konrad, and Sandy Shrum. *CMMI for Development: Guidelines for Process Integration and Product Improvement*. 3rd edition. Addison-Wesley Professional, 2011.

Agile CMMI (www.agileCMMI.com).

Jeff Dalton is President of Broadsword Solutions Corporation, a process innovation firm based in southeastern Michigan. Mr. Dalton is an SEI Certified SCAMPI Lead Appraiser, CMMI Instructor, ScrumMaster, and author of agileCMMI, Broadsword's methodology for incremental and iterative process improvement designed to eliminate the process debt that demoralizes engineers and destroys any potential productivity gains. He is Chairman of the SEI's Partner Advisory Board and President of the Great Lakes Software Process Improvement Network. Mr. Dalton is author of the popular blog Ask the CMMI Appraiser *and builds experimental aircraft in his spare time. He can be reached at appraiser@broadswordsolutions.com.*

Disciplined Agile Delivery Meets CMMI

by Scott W. Ambler

Contrary to what you may have heard, organizations around the world are applying agile and CMMI together, and some are doing it effectively. Sadly, some are not as successful as they would like to be. In this article, I start by sharing some important results from several surveys that I've run over the years that explore how people are applying agile and CMMI in practice. Second, I address some of the debilitating rhetoric still swirling around this topic. I then describe some prerequisites for getting agile and CMMI to work together and end by making the case for a disciplined agile CMMI strategy.

SOME INDUSTRY STATS

Let's begin with a quick discussion of what's actually happening out there. The *Dr. Dobb's Journal* (*DDJ*) January 2008 Process Framework Survey[1] explored whether people were combining agile and CMMI together. Of the respondents who said their organizations are doing agile:

- 10% indicated that they were exclusively doing CMMI-compliant agile projects.
- 45% said they were exclusively doing non-CMMI agile projects.
- 45% said that their organization had both CMMI-compliant and non-CMMI agile projects.

The same survey found that agile teams in CMMI-compliant environments statistically had the same success rates as those in non-CMMI environments. It also found that when given the choice, most people prefer repeatable results over repeatable processes, contrary to the mantra of some CMMI proponents. To summarize, of the survey respondents whose organizations were doing agile, 55% had one or more teams doing so within the scope of CMMI compliance, CMMI was neither helping nor hindering agile project teams on average, and repeatable processes aren't as attractive as some would lead us to believe.

I have to say, I was uncomfortable with the 55% figure. Although the 2008 survey revealed that many organizations were applying agile and CMMI together, it didn't determine what percentage of teams were combining them. For example, think of a company that has 20 agile project teams and only one of them is doing CMMI too. Yes, the company is doing agile and CMMI together, but only with 5% of its agile teams.

About a year and a half later, I explored the agile and CMMI issue again in a new survey that I targeted to the agile community itself. The Ambysoft July 2009 Agile Practices survey[2] asked several questions, one of which was whether the person's existing agile team was currently working in a CMMI-compliant manner. Of the respondents who indicated they were currently on an agile team, 9% said that they were working on projects that were CMMI-compliant. This result — that roughly 1 in 10 agile teams need to be CMMI-compliant — seems a lot more realistic to me.

This prompts the question of how easy it is to combine agile and CMMI. In the *DDJ* Summer 2012 State of the IT Union Survey,[3] I touched on this issue. The goal of the survey was to explore whether organizations were successful or unsuccessful at various levels of the scaling factors called out in the Agile Scaling Model (ASM).[4] One of the ASM scaling factors is regulatory compliance, which includes both legal compliance, such as adhering to US Food and Drug Administration (FDA) regulations, and self-imposed compliance, such as conforming to CMMI or ISO 900X. The survey found that of the respondents whose organizations had achieved success applying agile techniques in practice, 44% said that one or more of their project teams had done so under self-imposed compliance requirements. Of the respondents whose organizations had experienced one or more failed agile projects, 30% indicated that one or more of their projects had self-imposed compliance requirements. Unfortunately, the survey did not ask about CMMI compliance directly, although CMMI was used as one example of self-imposed compliance.

The survey results lead me to three important observations. First and foremost, people are in fact successfully applying agile and CMMI together. Second, it can be a rocky road when doing so, and some organizations are running into problems. Three, there isn't any blatantly

obvious evidence for or against applying the two together. Granted, this third observation is based on averages — your organization may have very good reasons to apply the two together. More on this shortly.

ADDRESSING THE RHETORIC

You don't have to spend much time online to discover that when it comes to applying agile and CMMI together, there is some questionable rhetoric being bandied about. I feel it's important to surface this rhetoric and describe the reality of the situation. Common misperceptions about agile CMMI include:

- **Agile and CMMI are incompatible.** This is clearly not the case, as we learn from the aforementioned surveys as well as from several publications about the topic.[5, 6] From what I've seen, most of this problem stems from agile protagonists not understanding CMMI and CMMI protagonists not understanding agile.
- **Scrum is CMMI Level 5.** Scrum is a very, very small part of what you need to do to succeed at agile. Scrum's focus is on some aspects of project leadership and requirements management, and it relies on other methodologies such as Extreme Programming (XP), Agile Modeling (AM), the Unified Process (UP), and many others to fill in the blanks. Yes, Scrum can be used in CMMI environments, but Scrum on its own clearly doesn't even address all CMMI Level 2 issues let alone those at the higher levels. By the same token, agile methods such as XP and AM can just as easily be applied in CMMI environments to address portions of one or more process areas (PAs) in an agile manner.
- **CMMI doesn't add value.** You can demonstrate empirically that this is not the case by simply hopping on a flight to Bangalore to see how the Indian IT service providers have leveraged CMMI into a multibillion-dollar industry. Furthermore, there are numerous studies showing that as organizations move up CMMI levels, their productivity improves.
- **CMMI equals needless bureaucracy.** The way organizations address CMMI compliance is up to them. They can choose to adopt a documentation-heavy strategy (which many unfortunately do), or they can choose to adopt a more streamlined agile approach. Many agilists have had very bad experiences in heavy CMMI shops, and in many cases that is their only CMMI experience — hence the bitterness toward CMMI.

MAKING AGILE AND CMMI WORK TOGETHER

For agile and CMMI to coexist together effectively, in my experience four things need to occur. Organizations must:

1. Take an enterprise view of IT
2. Focus on quantifiable business value
3. Adopt a full delivery lifecycle
4. Take a hybrid approach

Let's examine each of these success factors in greater detail.

Take an Enterprise View of IT

Adopting an enterprise view of IT can require an open mind of both CMMI and agile practitioners. An important implication of enterprise awareness is that different teams are in different situations; therefore, they need to tailor their strategies accordingly. Enterprise professionals who deal with many project teams need to be capable of working with and governing traditional teams in a traditional manner and agile teams in an agile manner. For agilists, being enterprise-aware can be very difficult at first due to the prevailing project mindset within that community. Agilists need to become disciplined enough to leverage and enhance the existing software and data infrastructure, to follow common guidelines, to work with enterprise professionals such as enterprise architects and operations professionals, and to be governed effectively. The need for enterprise awareness is a key philosophy of Disciplined Agile Delivery (DAD), which I describe below.[7]

Focus on Quantifiable Business Value

The second success factor is for your organization to focus on delivering quantifiable business value to your stakeholders in all activities that you perform. In other words, focus on real process and organization improvement and squeeze out the needless bureaucracy that is all too prevalent in CMMI environments. I have yet to evaluate a CMMI organization, including those rated at Level 4 and Level 5, that didn't have very large opportunities for improving their productivity by adopting more agile ways of working. For example, I often run into existing development processes in which the project team creates a requirements specification, writes test plans and test cases so that the solution may be validated, and then maintains traceability between these artifacts (and more) for good measure. Yes, business value is being delivered via this process, but we can achieve the same goals while working more effectively. For example, by adopting an acceptance test–driven

approach, the acceptance tests become both the detailed tests and the detailed requirements specification, with full requirements-to-test traceability between them with no extra work.[8] By working smarter, not harder, you not only reduce the work required to provide the same business value as before, but you do so with a shorter feedback cycle between requirements elicitation and implementation, thereby reducing project risk.

Adopt a Full Delivery Lifecycle

Next, you need to adopt a full delivery lifecycle. Minimally this includes project initiation, construction, and deployment activities, although you should also consider enterprise activities such as portfolio management, enterprise architecture, asset management, operations, support, and others. Figure 1 depicts the "agile" version of the DAD lifecycle (there are also lean and continuous delivery lifecycles to choose from). Notice how the lifecycle starts with project identification and selection, a portfolio management activity. It also includes an explicit Inception phase, sometimes erroneously referred to as Sprint 0 or Iteration 0, where fundamental project initiation activities occur. Even agile project teams need to get started somehow. There is, of course, an explicit Construction phase as well as a Transition phase[9] focused on deploying your solution. After deployment into production, the work continues with operations and support, and hopefully the team is allowed to begin work on the next release.

Take a Hybrid Approach

The final success factor is to take a hybrid approach to development. Many agile methodologies — including Scrum, XP, AM, Agile Data, Kanban, and more — focus on a subset of the activities required to deliver a solution from project initiation to delivery. Before DAD was developed, you needed to cobble together your own agile methodology to get the job done. As you can see in Figure 1, DAD clearly adopts strategies from Scrum, AM, and the UP, and you'll need to take my word for it that it also adopts ideas from other sources such as XP, Kanban, lean software development, and many more. The bottom line is that if you intend to address all the CMMI process areas, you will need to adopt a hybrid approach such as DAD or do the work to invent your own.

DISCIPLINED AGILE DELIVERY IN DETAIL

I have already covered a few aspects of the DAD process decision framework — it is a hybrid agile framework that supports a full delivery lifecycle and enterprise awareness — but there's more to it than that. For CMMI practitioners, there are several interesting aspects about DAD that I haven't yet described. First, it is very flexible. Although Figure 1 depicts a basic agile version of the lifecycle, that is only one of many options available to you. For example, you may choose to follow a leaner lifecycle in which Construction and Transition activities occur when appropriate, not within the confines of iterations and phases. Or you may want to follow a continuous delivery version of the lifecycle in which you release into production on a very regular basis, perhaps even several times a day. Because different teams will find themselves in different situations, an effective — or dare I say mature? — process framework needs to support several types of lifecycles.

Second, when it comes to process tailoring, DAD starts in the middle, not at the extremes, as do other methods

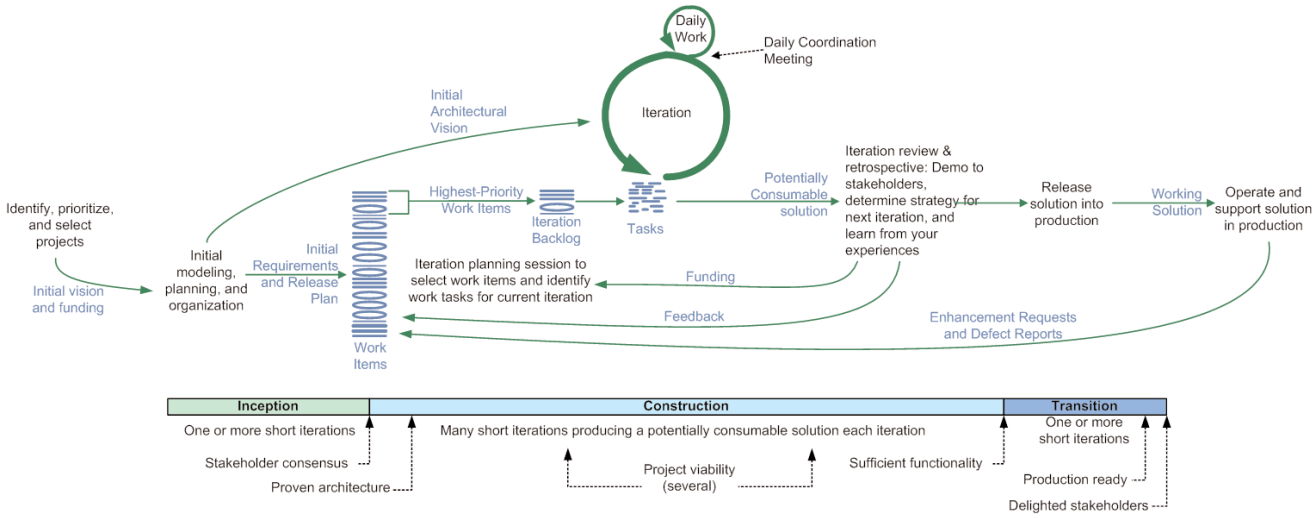


Figure 1 — Agile DAD lifecycle.

such as the Rational Unified Process (RUP) and Scrum. Many organizations that adopted the RUP ran into trouble because its tailoring strategy assumes that individual teams will have the process expertise to tailor the very large RUP library down into something manageable. At the other end of the spectrum, many organizations that have adopted Scrum have also run into problems because Scrum starts with something very small and assumes that your teams have the process expertise to tailor it up to what you need. These strategies are doubly flawed — teams typically *don't* have the required process expertise and both methods start at the extremes and insist that you tailor them into the middle ground. DAD starts in the middle ground.

DAD is based on the assumption that project teams are unlikely to have the process expertise to tailor their process effectively even though they will need to. This leads to the third aspect of DAD that CMMI practitioners will find intriguing — it is goal-driven, not prescriptive. Rather than prescribing certain practices or activities, the DAD process instead suggests that your team needs to address certain goals, that for each goal there are several issues to contemplate, that for each issue you have a number of techniques you can employ to address the issue, and that each technique has its strengths and weaknesses, so you will need to choose the one that's best suited for your situation. The goals of the DAD framework are depicted in Table 1. Notice how some are applicable for a given project phase and some are applicable throughout the entire project lifecycle.

What "Goal-Driven" Looks Like

To understand the difference between prescriptive and goal-driven approaches, let's compare similar aspects of Scrum and DAD. Scrum prescribes that you manage your work via a queue called a "product backlog" whose items are ordered by business value. DAD, on the other hand, suggests that you address changing stakeholder needs in some manner and that one of the associated issues is choosing a strategy to prioritize your work. It further indicates that strategies could be driven by several factors — business value, risk, due date, dependencies — and that strategies may in fact be combined. DAD goes further yet to describe the tradeoffs associated with each strategy and provides advice for when each strategy is viable. The implication is that, in some situations, Scrum's product backlog strategy is appropriate, but the RUP's risk-value approach might be better for some of your teams, or perhaps a combination of prioritization strategies is best. Rather than obsessing over repeatable processes, organizations that want to achieve repeatable *results* — such as producing quality solutions that meet the needs of their stakeholders and spending their IT budget wisely — will want delivery teams that manage changing stakeholder needs effectively. Because different teams are in different situations, each team may do this differently.

Let's dive a bit deeper into the workings of a goal-driven strategy. Figure 2 depicts the overview diagram for the Construction phase goal Move Closer to Deployable Release. To fulfill this goal, you need to address several issues, such as adopting a deployment strategy and a validation strategy. For each strategy, there are several techniques that you may choose to adopt. In some cases the techniques may be combined, and in some cases it wouldn't make sense to do so. For example, to validate your solution, you might apply both developer and acceptance test–driven development as well as continuous integration (CI)

Table 1 — The DAD Lifecycle Goals

Goals for Inception Phase	Goals for Construction Phase Iterations	Goals for Transition Phase
• Form Initial Team • Develop Common Project Vision • Align with Enterprise Direction • Explore Initial Scope • Identify Initial Technical Strategy • Develop Initial Release Plan • Form Work Environment • Secure Funding • Identify Risks	• Produce Potentially Consumable Solution • Address Changing Stakeholder Needs • Move Closer to Deployable Release • Improve Quality • Prove Architecture Early	• Ensure Solution Is Consumable • Deploy Solution
Ongoing Goals		
• Fulfill Project Mission • Grow Team Members • Address Risk	• Improve Team Process and Environment • Leverage/Enhance Existing Infrastructure	

and parallel independent testing. For verification, you might adopt non-solo techniques such as pair programming but not invest time in other types of reviews (either formal or informal), while still adopting both static and dynamic analysis techniques. The point is that an individual team will need to address this goal and its issues, but it may do so using a unique combination of techniques depending on its situation.

Reconciling DAD and CMMI

So ... how do DAD and CMMI fit together? Some CMMI process areas[10] are subsumed in a single DAD goal. For example, CMMI's Configuration Management PA maps to the Asset Management issue of the Move Closer to Deployable Release goal. The Requirements Development PA, on the other hand, is addressed by two DAD goals, in this case Explore Initial Scope and Address Changing Stakeholder Needs. And while the DAD goals don't directly address the Organizational Training PA, they do address it indirectly through DAD's People First aspect, which underlies the overall DAD framework.

One challenge with directly mapping CMMI to DAD is that the underlying assumptions between the two are not identical. For example, the CMMI's Requirements Management (REQM) PA states that an aspect of REQM is bidirectional traceability. DAD, on the other hand, adopts the AM and XP philosophy that a particular effort, such as investing in bidirectional traceability, should only be undertaken when it adds value for one or more stakeholders *and* the stakeholders are aware of its ongoing maintenance costs *and* they are willing to pay those costs. In other words, maintain traceability because it makes sense to do so, not just because a self-imposed regulation tells you to.

It is valid to ask whether DAD's goal-driven approach forms the basis of an agile maturity model. I would be reticent to go down this path. Consider the Move Closer to Deployable Release goal overview diagram of Figure 2 again. Some of the techniques associated with some issues are in fact ordered according to something resembling "maturity." For example, the Document Strategy issue has techniques such as Continuous Documentation (Same Iteration), Continuous Documentation (Following Iteration), Document Late,

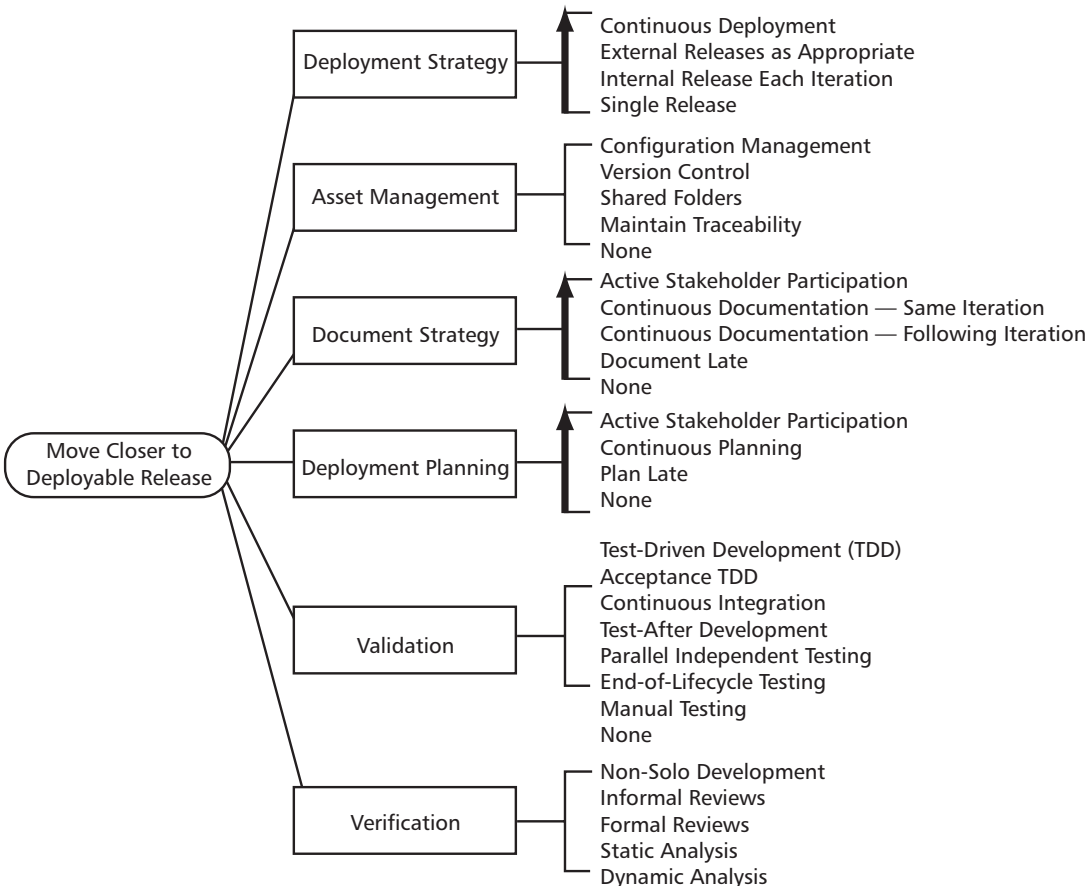


Figure 2 — Goal overview diagram for Move Closer to Deployable Release.

and None (at all). Ignoring the orthogonal technique of Active Stakeholder Participation, these techniques are presented in order of most mature to least mature (the arrow indicates that it's an ordered list and points upward toward greater maturity). In contrast, consider the Validation issue, which has a variety of testing strategies that could be used in combination and that require various levels of maturity. Perhaps it would be possible to refactor this issue into two or more smaller issues for which a maturity ordering makes sense, but there are other issues in other goals where such refactoring wouldn't be a viable option.

PARTING THOUGHTS

We can observe that organizations are succeeding at applying agile and CMMI in practice. We can also observe that there is significant motivation to do so for some organizations, regardless of the rhetoric of either the CMMI or agile camps. If we're going to help make the adoption of "agile CMMI" smoother, we will need to focus on several things. First, we must start with respectful and meaningful conversations, and that isn't always happening. Second, CMMI advocates need to open their minds to agile ways of working that still achieve the same goals, albeit in different, more effective ways. Third, agile advocates need to be open to the idea that CMMI can in fact add some real value in their environments. Fourth, we must evolve CMMI so that it truly focuses on specifying what a process should address without bringing process design issues into the conversation. Fifth, agile must evolve to reflect a more mature way of working, something that I feel has been achieved with Disciplined Agile Delivery.

I believe the evidence is very clear that agile approaches to working prove to be more effective in practice than traditional ones. I also believe that for some organizations there are compelling reasons to adopt CMMI. Having said that, let me leave you with this thought: If you're going to be CMMI-compliant, wouldn't you rather take an agile approach — or better yet, a *disciplined* agile approach — when doing so?

ENDNOTES

[1]Ambler, Scott W. "Dr. Dobb's Journal Process Framework Survey: January 2008" (www.ambysoft.com/surveys/processFramework2008.html).

[2]Ambler, Scott W. "Ambysoft Agile Practices Survey: July 2009" (www.ambysoft.com/surveys/practices2009.html).

[3]Ambler, Scott W. "Dr. Dobb's Journal Summer 2012 State of the IT Union Survey" (www.ambysoft.com/surveys).

[4]Ambler, Scott W. "The Agile Scaling Model: Adapting Agile Methods for Complex Environments." IBM/Rationale Software December 2009.

[5]McMahon, Paul E. *Integrating CMMI and Agile Development: Case Studies and Proven Techniques for Faster Performance Improvement*. Addison-Wesley Professional, 2010.

[6]*Agile CMMI* (www.agilecmmi.com).

[7]Ambler, Scott W., and Mark Lines. *Disciplined Agile Delivery: A Practitioner's Guide to Agile Solution Delivery in the Enterprise*. IBM Press, 2012.

[8]Gärtner, Markus. *ATDD by Example: A Practical Guide to Acceptance Test–Driven Development.* Addison-Wesley Professional, 2012.

[9]The Transition "phase" eventually evolves into an activity with the adoption of continuous delivery practices.

[10]CMMI Product Team. "CMMI for Development, Version 1.3." SEI/Carnegie Mellon University, November 2010.

Scott W. Ambler is a Senior Consultant with Cutter Consortium's Agile Product & Project Management *and* Business & Enterprise Architecture *practices. Mr. Ambler works with organizations around the world to help them improve their software processes. He provides training, coaching, and mentoring in Disciplined Agile and Lean strategies at both the project and organizational level. Mr. Ambler is the founder of the Agile Modeling (AM), Agile Data (AD), Disciplined Agile Delivery (DAD), and Enterprise Unified Process (EUP) methodologies. He is the author or coauthor of 20 books, including* Disciplined Agile Delivery, Refactoring Databases, Agile Modeling, Agile Database Techniques, The Object Primer 3rd Edition, *and* The Enterprise Unified Process. *Mr. Ambler is a Senior Contributing Editor with* Dr. Dobb's Journal. *He can be reached at sambler@cutter.com; website ScottAmbler.com.*

What Will It Take to Achieve Agility-at-Scale?

by Douglas Schmidt, Anita Carleton, Erin Harper, Mary Ann Lapham, Ipek Ozkaya, and Linda Parker Gates

The Software Engineering Institute (SEI) has a long history of process improvement research, and — like almost everyone who has proposed a model or method for software — a long history of fighting misperceptions. Chief among these is the idea that picking the "best" model or method and checking off a series of boxes will fix whatever ails you, indefinitely. Today, software development methods have their own T-shirts and bumper stickers, exacerbating the problem: some people pick a method the way they pick a brand of shoes and remain just as loyal.

CMMI is primarily about applying proven best practices and can be used with many other methods and models. Four years ago, the SEI published a Technical Note whose very title removes any ambiguity about its position: "CMMI or Agile: Why Not Embrace Both!"[1] To see real improvement, we need to recognize that it's not about how well we "do" agile or what level of CMMI we reach — it's about the software quality and development performance those methods enable.

But while enabling better performance today is important, it's not enough. We need research and measurable indicators to determine how to achieve better performance in the face of the significant and growing problems of tomorrow. The tangle of challenges resulting from complexity, exacting regulations, and schedule pressures is constantly expanding. We'll all be left behind if we continue the same old bickering.

The benefits of being agile are widely recognized, but the engineering disciplines needed to apply agility to the development of mission-critical software-reliant systems at scale are not as well defined or practiced. While agile and CMMI are often presented as competitors, they were both created to address many of the same concerns about software development. As we contemplate looming software challenges, it's important to again identify the most pressing concerns before arguing about the solutions. Understanding and explicating future challenges will help guide the evolution of existing methods and models, including CMMI and agile, and identify new areas of research that will enable the success of ever larger and more complicated systems.

Over the past several years, the SEI has embarked on a multidimensional approach to achieving *agility-at-scale*, which means improving the agility of software development for large-scale software-reliant systems, while providing visibility into release planning and system quality. The remainder of this article describes five related areas of concern that are prompting SEI research on agility-at-scale:

1. Reducing integration risk with architecture
2. Defining metrics to measure and model performance outcomes
3. Managing technical debt
4. Applying agile principles in the strategic planning processes
5. Defining practical guidance and piloting agile methods in the US Department of Defense (DoD)[2]

REDUCING INTEGRATION RISK WITH ARCHITECTURE

Software architecture is critical to successful development, serving as the blueprint for both the system and the project developing it. Architecture is especially important when developing large, complex, and distributed systems, but determining how much architecture is needed is not easy. Early production or overproduction of architecture can create delay, while too little architecture can result in integration defects that lead to costly rework. While agile techniques have been successful in improving efficiency when projects involve small teams, applying the same concepts to large-scale distributed systems — including the rest of an organization that has priorities larger than the development team — is critical for success at scale. One way the SEI is addressing the challenge of scaling is by investigating the amount of architecture needed, when it is needed, and the influence architecture exerts on managing teams.[3]

The SEI architecture work relies on the definition of quality attributes, such as those for performance, security, modifiability, reliability, and usability. Architects need to understand their designs in terms of these attributes; for example, they need to understand whether they will achieve deadlines in real-time systems, what kind of modifications are supported by their design, and how the system will respond in the event of a failure. The quality attributes important for an architecture are rigorously defined with the stakeholders and used to drive functional requirements. CMMI v1.3 is significantly more architecture-oriented than previous versions and maintains a quality attribute thread through most process areas. But we still need a way to determine how much architecture is enough to support integration without causing unnecessary delay.

Ongoing SEI research is focused on providing software and systems architects with a decision-making framework for reducing integration risk with agile methods, thereby reducing the time and resources needed for related work. A primary objective is to identify critical properties of agile software development that are influenced by architectural decisions that reduce lifecycle time. Identifying these properties is an important first step in giving stakeholders the tools they need to make informed decisions about which properties to control for better outcomes. After the properties are identified in this first phase of the work, SEI research collaborators and customers will review them. Based on such properties, software or systems architects can create an architectural decision model that allows them to analyze the ramifications of their decisions, especially those related to interfaces and how the parts of the system are connected. The model also helps determine a structure for a system that will achieve quality attribute requirements.

Modeling architectural decisions during early stages of development will reduce cycle time by not only enabling an earlier start of development, but also avoiding over- or under-architecting. Likewise, cycle time could be reduced downstream by decreasing rework costs attributed to architectural decisions that affect integration.

Another component of SEI's research in this area explores strategies for improving the relationship between architecture decisions and complex collaborative team interaction. An obstacle often arises when partitioning decisions made by architects are not aligned with the groups of teams needed for large systems. Another barrier is alignment of the teams beyond the development team, including system engineering, testing, validation and verification, and certification and accreditation teams. The ramifications of a misalignment frequently appear during the integration of components built by different teams, where incompatibilities can lead to significant rework.

To map, analyze, and support the architectural decisions of industry collaborators, the SEI is mapping its approach into a cyber-social conceptual model being developed by the University of Virginia as part of its research on social networks and the value of relationships between decision makers in a system.

DEFINING METRICS TO MEASURE AND MODEL PERFORMANCE OUTCOMES

Unproven assertions make up a lot of what people think they know about software development, an inevitable consequence of a dearth of quantifiable and validated information in the field. Yet data is the critical, irreplaceable component of every successful effort to model, improve, or measure performance. As such, the SEI maintains that using data-centric approaches is integral to advancing the discipline of software engineering. There is significant variation, however, in exactly what is collected and how it is reported. For example, the speed and effort related to completing a unit of work is used to track velocity, but it is not generally beneficial to compare velocity across agile projects.

To be useful for analysis in large-scale, mission-critical projects, data must be consistently defined, validated, and shared. That's why SEI researchers are focusing on the creation and implementation of measurement frameworks and on measuring actual outcomes. A good measurement framework guides the collection, analysis, and — what many forget is the ultimate goal — *use* of the data to take action. If used as intended, CMMI can help organizations build a successful process and measurement framework compatible with any development method. The measurement framework used in the SEI's Team Software Process (TSP) method — compatible with both CMMI and agile methods — has yielded some of the most high-quality, fine-grained, statistically validated data available. In fact, TSP's intrinsic measurement framework and its ability to generate this type of data for analysis have made it a preferred method for developing safety-critical software.

As agile use has grown, so has the need for related measurement information. Software developers want to know which practices pay off, which work best in organizations similar to theirs, and why some of their agile projects produce better results than others. Customers acquiring software want to know if a software

development organization is simply saying they are "doing agile" or if their performance outcomes actually show the benefits of agile methods. It is notable that the "Ten Year Agile Retrospective" cites data from several implementations of agile and CMMI used together.[4] As the agile community evaluates its past achievements and makes plans to build on its success, it too has to learn from implementations that included a measurement framework and produced accessible, analyzable data.

When evaluating the use of agile, the community should avoid measuring compliance instead of performance, a problem that has dogged some poor implementations of CMMI. Those who use CMMI primarily as a compliance model — checking off processes for an appraisal — do not necessarily see performance improvements. Similar problems emerge with agile. You can't determine if an organization is actually using agile and experiencing related performance enhancements by evaluating the organization against a checklist of agile practices. If being "agile" means responding rapidly and efficiently to change, with consistency, then outcome measures need to be in place for response time, efficiency, and consistency.

Current SEI research in measuring performance is being conducted with Tecnológico de Monterrey, using TSP data to validate past outcomes independently.

MANAGING TECHNICAL DEBT

Ward Cunningham, who coined the term "technical debt," stated that shipping code quickly is like going into debt. A little debt speeds up development and can be beneficial as long as the debt is paid back promptly with a rewrite that reduces complexity and streamlines future enhancements. A delicate balance is needed between the desire to release new software capabilities quickly to satisfy users and the desire to practice sound software engineering that reduces subsequent rework. Adopting agile practices at scale without considering their long-term implications can easily lead to technical debt.

There are many different measures of technical debt. Some agile projects define and measure technical debt as lines of code that are poorly written or poorly refactored, do not follow coding standards, or are not supported with sufficient unit tests; they may also measure the amount of code duplication. Much of the existing literature on technical debt focuses on code-level issues, such as reducing the time needed to modify software functions, add new features, or fix bugs.

SEI research reveals that it's also important for organizations to consider how to best describe technical debt from architecture- and system-level perspectives. Our research in this area focuses on managing debt as an agile software architecture strategy.[5] Specifically, SEI researchers are identifying the impacts on cost when architectural changes are needed. Often, when a particular symptom in a system is described as technical debt, it's not just the code that is bad; the system is also accumulating problems in terms of architectural changes that have occurred throughout its development.

The SEI's work on quantifying technical debt does include code, but it primarily focuses on quantifying debt early in the lifecyle, when code analysis alone may provide insufficient direction. Specifically, it focuses on understanding the right set of architectural models that can be used seamlessly within agile methods to help development teams understand the impact of rework. Modeling software development decisions is important in solving key issues observed in large-scale, long-term projects. Architectural decisions should be continuously analyzed and actively managed, since they incur cost, value, and debt.

While SEI research into effective payback strategies for technical debt is ongoing, there are some immediate actions you can take to manage technical debt, no matter what development method you are using:

- Make technical debt visible, even if it's just acknowledging that there is a problem.
- Differentiate strategic, structural technical debt from unintended debt incurred as a result of factors like low code quality, bad engineering, or practices that have not been followed.
- Bridge the gap between the business and technical sides.
- Associate technical debt with risk and track it explicitly.

APPLYING AGILE PRINCIPLES IN STRATEGIC PLANNING PROCESSES

As the appeal of agile development methods has grown in the software development community, so have concerns that organizational culture is impeding adoption and results. Aligning management and planning practice with agile development can help to create a cohesive organizational culture.[6] We are exploring the use of agile strategic planning to help organizations focus on adaptive, iterative delivery. Adopting agile processes

to move the organization toward its mission — agile strategic planning — allows agility to become a cohesive organizational property. Strategic planning is the process of defining an organization's plans for achieving its mission and is a critical foundation for executing work.[7] It sets the stage for enterprise-wide initiatives such as enterprise architecture, process improvement, risk management, and portfolio management, as well as division- and unit-level planning. Strategic planning typically follows this general sequence:

1. Scope the strategic planning effort.
2. Build a foundation of organizational information.
3. Define goals and objectives in terms of the organizational need and desired outcome.
4. Identify potential strategies for achieving the objectives.
5. Develop action plans.
6. Identify project measures.
7. Execute the work.
8. Track progress.

SEI research previously investigated the integration of the critical success factor (CSF) method and future scenario planning into the strategic planning process.[8] CSFs are factors that determine group or company success, including key jobs that must be done exceedingly well. Future scenarios are a tool for exploring multiple possible "futures" and developing strategies that will serve the uncertain future well. Together, these techniques foster strategic thinking, a strong complement to strategic planning. They also provide some leverage points for making strategic planning more nimble and can be used to help an organization pursue the four key principles of the Agile Manifesto:[9]

1. **Individuals and Interactions Over Processes and Tools.** Both the CSF method and scenario planning rely heavily on individuals and interactions. A strong process is important for effective strategic planning, but face-to-face conversations between leadership and staff enhance the value of the strategic planning process.
2. **Working Software Over Comprehensive Documentation.** A common criticism of strategic planning is that it overemphasizes deconstruction of the past and present while creating the illusion that we can anticipate the future. Replacing the second agile principle's "working software" with "tangible results" (to accommodate the strategic planning domain) represents a productive strategy for focusing efforts on shorter-term accomplishments over thoroughly documented commitments about the future.
3. **Customer Collaboration Over Contract Negotiation** is not quite as pertinent, as strategic planning does not involve contract negotiation. A strategic plan does, however, serve as an informal contract with the organization, and including a broad cross-section of staff is essential. In addition, involving customers in scenario planning brings an external perspective that can be critical to getting quality results.
4. **Responding to Change Over Following a Plan** offers perhaps the greatest potential for improving the way strategic planning is conducted and the results are implemented. A good strategic planning process has always done more than just produce a plan; it supports ongoing strategic thinking, discussion, and behavior. Strategic thinking focuses on finding and developing organizational opportunities and creating dialogue about the organization's direction — these are the foundation of an organization's readiness to respond to change. Strategic planning is enhanced by strategic thinking, which makes planning adaptive to its evolving environment.

To bring agility to the strategic planning process, adjustments are probably most effective when made to Steps 3 and 8 of the strategic planning process outlined above. Step 3 (define goals and objectives) benefits from a focus on the first value (individuals and interactions) and the third value (customer collaboration). Step 8 (track progress) is enhanced through a focus on the second value (working software recast as "tangible results") and the fourth value (responding to change).

The SEI is conducting additional applied research in other areas of management and acquisition that can be usefully aligned with agile development methods. An agile organization seeks to change what the team values, measures, and delivers. Embracing agile principles wholly, and understanding the challenges and disconnects organizations face when their organizational cultures and management systems do not fully embrace agile principles, can make the difference between success and failure.

DEFINING PRACTICAL GUIDANCE AND PILOTING AGILE IN THE DoD

Through applied research, the SEI is exploring the use of agile approaches at scale in software-intensive systems developed by the DoD.[10, 11] While the goal of this

work to date has been to provide prudent, pragmatic support for implementing agile methods in the DoD, the lessons learned also apply to those who want to incorporate these methods in other highly regulated environments. Policies, regulations, and other governing documents aside, the main difference between using agile and a more traditional method in these environments is the requirement for different management and technical approaches, as we discuss below. Through this work we have identified several issues and challenges when implementing agile in a DoD environment. A few of the more broadly applicable ones involve the mismatch between the documentation produced and required, adjusting teams to include new types of team members (such as end users), and understanding the different approaches agile and traditional methods use for integration and test.

One stumbling block for the adoption of agile in the DoD is the requirement for capstone technical review events, such as preliminary design review and critical design review. Agile methods typically do not produce the types of documentation expected at these milestones. Instead, they provide working prototypes and, in some cases, a subset of requirements implemented as usable software. Therefore, at the beginning of a contract, expectations and acceptance criteria must be established that meet both the contractual needs and allow for the use of agile methods. Since agile development produces the final product iteratively, the expectations and acceptance criteria must be compatible.

Another disparity involves end-user access and involvement, which in agile often means end users are integral members of the project team. This approach is not always practical with joint programs and the myriad of stakeholders served by large software-reliant systems. Some effects of this problem can be mitigated by agreeing on an end-user proxy for day-to-day interactions but inviting actual end users to all demos. Other changes in team composition are frequently needed as well. Two important positions new to those accustomed to more traditional teams are the agile advocate and the end-user representative. An agile advocate provides real-time responses to immediate agile issues, while the end-user representative not only represents the end users, but also has the authority (within delegated limits) to direct the contractor.

Integration and test is also different when using agile methods. Continuous integration and test of some form is done within agile teams, which is often contrary to the traditional approach, in which integration is done at the end of a release cycle. If the final integration and test is being used for system acceptance, then most likely an independent external team will conduct the work. If an agile team has been doing continuous integration and test during development, however, there should be fewer risks to overcome since more issues should have been found earlier.

CONCLUDING REMARKS

The mission of the SEI is to advance software engineering and related disciplines. While some practitioners may continue to debate *ad nauseam* whether CMMI and agile can coexist and which method is "right," our research remains focused on the big picture: what problems still exist or loom in the future that impede progress and introduce significant risk in our software-dependent society? As we set our research agenda to find solutions to these problems, we focus not on hype and self-reported outcomes, but on measured performance and practices that can be empirically validated.

We believe maintaining agility in the development of mission-critical software-reliant systems at scale is an issue that will significantly affect our future. Agility is not a capability that you achieve by accident; like agility in sports, it requires teamwork, strategy, training, planning, measurement, and discipline. The SEI's research efforts have concentrated in recent years on enhancing and evaluating lightweight approaches and processes to address what's needed and/or mandated by our sponsors, partners, customers, and other stakeholders. Meeting these needs involves scaling up agile methods to the mission-critical software-reliant systems common in the DoD, as well as other domains such as finance, telecommunications, energy, space exploration, and aviation.

In our experience, striking the right balance between agility and discipline is essential for organizations that seek to implement agile methods at scale.[12] In particular, while organizations work to improve agility, they must do so in a measured, disciplined way. Understanding how and when to implement agile methods in large, mission-critical software-reliant systems is possible only through research into and evaluation of the performance outcomes those methods provide, whether they are classified as agile or CMMI or something else. The areas of concern described in this article represent the underpinnings of the SEI's research on agility-at-scale.

ENDNOTES

[1]Glazer, Hillel, Jeff Dalton, David Anderson, Michael Konrad, and Sandra Shrum. "CMMI or Agile: Why Not Embrace Both!" SEI/Carnegie Mellon University (CMU), November 2008.

[2]*Agile Research Forum* (SEI Webinar Series). SEI/CMU, 2012 (www.sei.cmu.edu/go/agile-research-forum).

[3]Nord, Robert. "Rapid Lifecycle Development in an Agile Context." *SEI Blog*, 9 April 2012 (http://blog.sei.cmu.edu/post.cfm/rapid-lifecycle-development-in-an-agile-context).

[4]Sutherland, Jeff. "Ten Year Agile Retrospective: How We Can Improve in the Next Ten Years." MSDN Library, Visual Studio, 2010 (http://msdn.microsoft.com/en-us/library/hh350860%28v=vs.100%29.aspx#CommunityContent).

[5]Bachmann, Felix, Robert L. Nord, and Ipek Ozkaya. "Architectural Tactics to Support Rapid and Agile Stability." *CrossTalk*, Vol. 25, No. 3, May/June 2012.

[6]Lapham, Mary Ann, Suzanne Miller, Lorraine Adams, Nanette Brown, Bart Hackemack, Charles (Bud) Hammons, Linda Levine, and Alfred Schenker. "Agile Methods: Selected DoD Management and Acquisition Concerns." SEI/CMU, October 2011.

[7]Gates, Linda Parker. "Toward Agile Strategic Planning." *SEI Blog*, 5 March 2012 (http://blog.sei.cmu.edu/post.cfm/toward-agile-strategic-planning).

[8]Gates, Linda Parker. "Strategic Planning with Critical Success Factors and Future Scenarios: An Integrated Strategic Planning Framework." SEI/CMU, November 2010.

[9]Agile Manifesto (http://www.agilemanifesto.org).

[10]Lapham, Mary Ann, Ray Williams, Charles (Bud) Hammons, Daniel Burton, and Alfred Schenker. "Considerations for Using Agile in DoD Acquisition." SEI/CMU, April 2010.

[11]Lapham, Mary Ann. "DoD Agile Adoption: Necessary Considerations, Concerns, and Changes." *CrossTalk*, Vol. 25, No. 1, January/February 2012.

[12]Schmidt, Douglas C. "Balancing Agility and Discipline at Scale." *SEI Blog*, 23 July 2012 (http://blog.sei.cmu.edu/post.cfm/balancing-agility-and-discipline-at-scale).

Douglas C. Schmidt is a Professor of Computer Science at Vanderbilt University and a Visiting Scientist at the Software Engineering Institute (SEI). He is a member of the Air Force Scientific Advisory Board, was the CTO at the SEI, and was a Program Manager at DARPA. Dr. Schmidt has published 10 books and over 500 papers on software-related topics. He also led the development of the widely used ACE and TAO open source middleware. Dr. Schmidt received BS and MA degrees in sociology from William and Mary and an MS and PhD in computer science from the University of California, Irvine. He can be reached at dschmidt@sei.cmu.edu.

Anita Carleton is currently the Director of SEI's Software Engineering Process Management (SEPM) Program, where she provides leadership in researching, developing, and broadly transitioning technologies and methods of best engineering, management, and measurement practices for software-intensive systems. She has over 25 years' technical and management experience in the software engineering industry. She has coauthored Measuring the Software Process: Statistical Process Control for Software Process Improvement*, is a Senior Member of the IEEE Computer Society, and serves on the IEEE Software Advisory Board. Ms. Carleton can be reached at adc@sei.cmu.edu.*

Erin Harper is a member of SEI's Program Development and Transition group. She works with SEI technical staff members to develop effective communication and transition strategies for SEI methodologies, products, and services. Since 2005 she has worked closely with the Software Engineering Measurement and Analysis (SEMA) group, which provides organizations with qualitative and quantitative tools and methods to measure and analyze the results they are achieving at the project, process, program, and enterprise levels. Ms. Harper received an MAPW from Carnegie Mellon in 2001. She can be reached at eah@sei.cmu.edu.

Mary Ann Lapham is the technical lead of SEI's Acquisition Support Program, leading research on the use of Agile methods in US Department of Defense acquisition settings. In addition to leading the authoring of several SEI technical notes on this topic, she is also active in consulting with military service clients of the Acquisition Support Program on the use of Agile methods within their programs. Ms. Lapham has over 30 years' program management experience in commercial and government contexts. She can be reached at mlapham@sei.cmu.edu.

Linda Parker Gates is a senior member of the technical staff in SEI's Acquisition Support Program. She has over 20 years' experience in strategic planning, performance excellence, technology transition, and change management. Ms. Gates joined SEI in 1989. She has contributed to the development of major SEI technologies, including the Integrated Strategic Planning Framework, the Process Change Methodology, the Mastering Process Improvement training course, and the Defining Software Processes training course. Ms. Gates has served on the 2011 and 2012 Baldrige Board of Examiners, evaluating applications for the Malcolm Baldrige National Quality Award. She can be reached at lpg@sei.cmu.edu.

Ipek Ozkaya is a senior member of the technical staff in SEI's Research, Technology, and System Solutions Program. Dr. Ozkaya is currently engaged in activities focusing on large-scale Agile and architecture and works to develop, apply, and communicate effective methods to improve software development efficiency. She has coauthored multiple articles on these subjects, serves as Chair of the advisory board of IEEE Software*, and is also a member of the technical faculty for the Master of Software Engineering Program at Carnegie Mellon University. She can be reached at ozkaya@sei.cmu.edu.*

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