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

Dr. Douglas Schmidt, SEI Visiting Scientist
Anita Carleton, SEI Director Software Engineering Process Management Program
Linda Parker Gates, Senior Member of the Technical Staff
Erin Harper, SEI Technical Communications Senior Writer
Mary Ann Lapham, SEI Principal Engineer, Acquisition Support Program
Ipek Ozkaya, SEI Senior Researcher, Research, Technology, and System Solutions Program

Abstract

The Software Engineering Institute (SEI) has long maintained that CMMI is primarily about applying proven best practices and can be used with many other methods and models. 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. While Agile and CMMI are sometimes presented as competitors, it's important to remember that they were both created to address many of the same concerns about software development.

The benefits of Agile are widely discussed, but the engineering disciplines needed to apply agility to mission-critical software-reliant systems at scale are not as well defined or practiced. This article describes research being conducted at the SEI to understand and explicate solutions to challenges that will help guide the evolution of existing methods and models, including CMMI and Agile, and enable the success of development efforts building ever larger and more complex software-reliant systems.

1. Introduction

The SEI has a long history in process improvement research, and—like almost everyone who has proposed a model or method for software—a long history in fighting misperceptions. Chief among these misperceptions 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 like they pick a brand of shoes and remain just as loyal.

Using 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!* 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.

For the SEI and others involved in software process research, enabling better performance today is important, but it's not enough. We need research and measureable 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 over "lightweight vs. heavyweight," "bureaucracy vs. chaos," or "CMMI vs. Agile.

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 sometimes presented as competitors, it's important to remember that 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 multi-dimensional approach to achieving *agility-at-scale*, which means improving the agility of software development for large-scale software-reliant systems (such as mission-critical acquisition programs developed by the Department of Defense (DoD) and other organizations), 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:

- reducing integration risk with architecture
- defining metrics to measure and model performance outcomes
- managing technical debt
- applying Agile principles in the strategic planning processes
- defining practical guidance and piloting Agile methods in the DoD [1]

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 or overproduction of architecture can create delay, while too little architecture production can result in integration defects that lead to significant and 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 [2].

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 quality 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 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 identified in this first phase of the work are reviewed with collaborators, an architectural decision model can be made that allows software or systems architects 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.

This architecture-centric approach can be used to test the hypothesis that modeling architectural decisions during early stages of development will reduce cycle time. Cycle time could be reduced upstream by enabling an earlier start of development. Likewise, it could be reduced downstream by decreasing rework costs attributed to architectural decisions that affect integration.

Another component of the SEI's research in this area explores strategies for improving the relationship between architecture decisions and complex collaborative team interaction. A barrier 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. The ramifications of a misalignment often 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.

3. 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 supports the principle 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—usually a story point—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, which can be a challenge no matter the development method used. 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, the CMMI can be a source of guidelines for building 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 has seen widespread use, the need for related measurement information has also grown. Software development organizations 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 developer 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 [3]. As the Agile community evaluates its past achievements and makes plans to build on its success, it too has to rely on implementations that included a measurement framework and produced accessible, analyzable data.

When evaluating the use of Agile, however, the community should avoid the pitfalls of measuring compliance instead of performance, a problem that has dogged some poor implementations of CMMI. Those who use CMMI primarily as 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 them against a checklist of Agile practices. For example, 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 Tecnologico de Monterrey, using TSP data to validate past performance outcomes independently.

4. Managing Technical Debt

Technical debt refers to the degraded quality resulting from the rapid delivery of software capabilities to users. A delicate balance is needed between the desire to release new software capabilities rapidly 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, as well as measuring the amount of code duplication. 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. In fact, 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 [4]. Specifically, SEI researchers are identifying implications to 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; it's also accumulating problems in terms of architectural changes that have occurred throughout the system's development.

The SEI's work on quantifying technical debt does include code, but primarily focuses on quantifying debt early in the lifecyle, when code analysis alone may provide insufficient direction. Specifically, our research focuses on understanding the right set of architectural models that can be used seamlessly within Agile software development 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. Decisions concerning architecture should be continuously analyzed and actively managed since they incur cost, value, and debt.

While SEI research focusing on opportunities for developing and quantifying 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 is being used:

- 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.

5. Applying Agile Principles in Strategic Planning Processes

Changing the practices of an organization is rarely easy, as demonstrated many times throughout the years of SEI research in process management. These challenges are not related to a particular model or method; instead, the further the new practices and their methods are from current practice, the more difficult, time-consuming, and resource-consuming the change is. As the appeal of Agile development methods in the software development community has grown, so have the complaints that organizational culture is impeding adoption and results. Aligning practices in related area—such as management and planning—can help to create a cohesive organizational culture [5].

Current SEI research is examining the value of bringing agility to the strategic planning process. Strategic planning is the process of defining an organization's plans for achieving its mission and is a critical foundation for executing work [6]. It sets the stage for division- and unit-level planning, as well as enterprise architecture, process improvement, risk management, portfolio management, and any other enterprise-wide initiatives. 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 [7]. 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 decisions or strategies that will serve the uncertain future well. Together, these two techniques help foster strategic thinking, which is 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 pursuing the four key values presented in the Agile Manifesto.

A strong process is important for effective strategic planning, but placing value on interactions between leadership and staff and face-to-face conversations enhances the value of the strategic planning process. This supports the first Agile principle of "Individuals and Interactions Over Processes and Tools," as both the CSF method and scenario planning rely heavily on individuals and interactions. These conversations themselves bring value to strategic planning.

A common criticism of strategic planning is that it over-emphasizes deconstruction of the past and present while creating the illusion that we can anticipate the future. In the second Agile principle, "Working Software Over Comprehensive Documentation," replacing "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.

The third principle, "Customer Collaboration Over Contract Negotiation," is not quite as pertinent since strategic planning does not involve contract negotiation. A strategic plan does, however, serve as an informal contract with the organization, and involving customers in scenario planning brings an external perspective that can be critical to getting quality results.

This fourth principle, "Responding to Change Over Following a Plan," offers perhaps the greatest potential for improving the way strategic planning is conducted and the results realized in implementation. 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 and in sync with an 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 (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 are affected when Agile development methods are introduced. Picking an Agile process and following it step by step without fully embracing the culture can provide some benefit. If being agile is the goal, however, then a culture of agility must be created. The culture goes beyond using an Agile software delivery process, it seeks to change what the team values, measures, and delivers. Understanding the challenges organizations have when their organizational cultures and management systems do not easily sync with Agile principles can make the difference in success and failure.

6. Defining Practical Guidance and Piloting Agile in the Department of Defense

Through applied research, the SEI is exploring the use of Agile approaches at scale in software-intensive systems developed by the DoD [8] [9]. 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. Through this work we have identified several issues and challenges when implementing Agile in a DoD environment. A few of the more broadly applicable 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 differences in 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. Expectations and criteria for acceptance must therefore be established at the beginning of the contract that meet both the contractual needs and allow for the use of Agile methods. Since Agile produces the final product iteratively, the expectations and criteria for acceptance must be compatible.

Another disparity is in end-user access and involvement, which in Agile often means end users are integral members of the iteration team. This approach is not always practical with joint programs and the myriad of stakeholders served by a large, software-reliant system. Some effects of this problem can be mitigated by agreeing on a proxy for end users day-to-day interactions and inviting end users to all demos. Other changes in team composition are often 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 answers 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 using Agile methods. Continuous integration and test of some form is done within Agile teams, which is often contrary to the traditional approach where 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. Continuous integration and test during development using Agile methods, however, should mean that there are less risks to overcome since more issues should be found earlier.

7. 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 (or other methods and models) can co-exist 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, but like agility in sports, it requires teamwork, strategy, training, planning, measurement, and discipline. The SEI's research and transition efforts have concentrated in recent years on enhancing and evaluating lightweight approaches and processes to address what's needed and required 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 mission-critical programs in other domains, including finance, telecommunications, energy, space exploration, and aviation.

In our experience to date, we've found that striking the right balance between agility and discipline is essential for organizations seeking to implement Agile methods at scale [10]. 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 the research and evaluation of the performance outcomes methods provide, whether the methods 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.

References

[1] *Agile Research Forum*, SEI Webinar Series. Software Engineering Institute, Carnegie Mellon University, 2012. http://www.sei.cmu.edu/go/agile-research-forum/

[2] Nord, Robert. "Rapid Lifecycle Development in an Agile Context." SEI Blog. Software Engineering Institute, Carnegie Mellon University, 2012. http://blog.sei.cmu.edu/post.cfm/rapid-lifecycle-development-in-an-agile-context

[3] 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

[4] Bachmann, Felix; Nord, Robert L.; and Ozkaya, Ipek. "Architectural Tactics to Support Rapid and Agile Stability." *Crosstalk 25*, 3 (May/June 2012).

[5] Lapham, Mary Ann; Miller, Suzanne; Adams, Lorraine; Brown, Nanette; Hackemack, Bart; Hammons, Charles (Bud) ; Levine, Linda; & Schenker, Alfred. *Agile Methods: Selected DoD Management and Acquisition Concerns* (CMU/SEI-2011-TN-002). Software Engineering Institute, Carnegie Mellon University, 2011.

[6] Parker Gates, Linda. “Toward Agile Strategic Planning,” SEI Blog. Software Engineering Institute, Carnegie Mellon University, 2012. http://blog.sei.cmu.edu/post.cfm/toward-agile-strategic-planning

[7] Parker Gates, Linda; *Strategic Planning with Critical Success Factors and Future Scenarios: An Integrated Strategic Planning Framework* (CMU/SEI-2010-TR-037). Software Engineering Institute, Carnegie Mellon University, 2010.

[8] Lapham, Mary Ann; Williams, Ray; Hammons, Charles (Bud); Burton, Daniel; & Schenker, Alfred. *Considerations for Using Agile in DoD Acquisition* (CMU/SEI-2010-TN-002). Software Engineering Institute, Carnegie Mellon University, 2010.

[9] Lapham, Mary Ann. “DoD Agile Adoption: Necessary Considerations, Concerns, and Changes” *Crosstalk 25*, 1 (Jan/Feb 2012).

[10] Schmidt, Douglas C. “Balancing Agility and Discipline at Scale,” SEI Blog. Software Engineering Institute, Carnegie Mellon University, 2012. http://blog.sei.cmu.edu/post.cfm/balancing-agility-and-discipline-at-scale

Related Reading

Beck, Kent and Fowler, Martin. *Planning Extreme Programming*. Boston, MA: Addison-Wesley, 2000.

Boehm, Barry and Turner, Richard. *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston, MA: Addison-Wesley, 2004.

Humphrey, Watts; *The Watts New Collection: Columns by the SEI’s Watts Humphrey* (CMU/SEI-2009-SR-024). Software Engineering Institute, Carnegie Mellon University, 2009.

Jakobsen, Carsten and Johnson, Kent. "Mature Agile with a Twist of CMMI." Proceedings of the Agile 2008 Conference, Washington, D.C., 2008.

McConnell, Steve. *Professional Software Development: Shorter Schedules, Better Projects, Superior Products, Enhanced Careers*. Boston, MA: Addison-Wesley, 2004.

Poppendieck, Mary and Poppendieck, Tom. *Lean Software Development: An Agile Toolkit.* Boston, MA: Addison-Wesley, 2003.

Schwaber, Ken and Beedle, Mike. *Agile Software Development with Scrum.* Upper Saddle River, NJ: Prentice Hall, 2001.

McBreen, Pete. *Software Craftsmanship: The New Imperative*. Boston, MA: Addison-Wesley, 2001.

Sutherland, Jeff; Jacobson, Carsten; and Johnson, Kent. "Scrum and CMMI Level 5: A Magic Potion for Code Warriors!" Proceedings of the Agile 2007 Conference, Washington, D.C., 2007.