Software Engineering Institute

Agile Methods: Selected DoD Management and Acquisition Concerns

Mary Ann Lapham
Suzanne Miller
Lorraine Adams
Nanette Brown
Bart Hackemack
Charles (Bud) Hammons, PhD
Linda Levine, PhD
Alfred Schenker

October 2011

TECHNICAL NOTE
CMU/SEI-2011-TN-002

Acquisition Support Program/U.S. Air Force
http://www.sei.cmu.edu

Carnegie Mellon

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

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, the authors wish to thank Mr. Blaise Durante, Air Force Deputy Assistant Secretary for Acquisition Integration, for his continued support for the SEI and this project. This support allowed the authors to produce a second report addressing different topics of interest to DoD acquisition offices and development organizations that are currently pursuing or are contemplating pursuing acquisition strategies that employ one or more elements of a set of incremental development methods commonly termed "Agile methods."

In addition, the authors would like to express our appreciation for all those who took time to let us interview them. To those who reviewed the draft of this technical note, your insights added great value and improved the final version. We extend our sincerest thanks to the following organizations and people:

- Accenture Federal Services Agile Team, Mark Whelan, VP
- Dottie Acton, Lockheed Martin
- Glen Alleman, VP, Program Planning & Controls, Space & Defense, Lewis & Fowler
- Brent Barton, AgileEVM
- Stephany Bellomo, SEI
- Deborah Brey, Boeing Defense Systems
- Portia Crowe, Chief Engineer of the U.S. Army Defense Readiness and Projection Systems
- Noopur Davis, SEI Team Software Process (TSP) Initiative
- COL Pat Flanders, US Army Project Manager, Army Enterprise Systems Integration Program
- Hillel Glazer, Entinex, Inc.
- Shelley A. Goodman, Raytheon, IIS
- David Hussman, DevJam
- Suzette Johnson, Northrop Grumman
- John Klein, SEI
- Dr. Kathleen Mayfield, MITRE
- Carlton Northern, MITRE
- Patriot Excalibur (PEX) Program, Kelly Goshorn and team
- Alan Shalloway, CEO, NetObjectives
- Chris Sterling, AgileEVM
- Mike Stuedemann, Medtronic Field Services
- Warfighter's Edge (WEdge) Program, Lt Col Andy "Skipper" Berry and team
- David Zubrow, SEI

Executive Summary

Robert Gates, the United States Secretary of Defense, in a September 2008 speech, said, "Our conventional modernization programs seek a 99% solution in years. Stability and counterinsurgency missions—the wars we are in—require 75% solutions in months. The challenge is whether in our bureaucracy and in our minds these two different paradigms can be made to coexist" [Gates 2008]. This, and other similar statements by senior DoD officials, express a problem space that is also felt in commercial industry. In the commercial world, one challenge is how to get products to market faster than competitors do, while taking advantage of the latest technologies. In the DoD, the competitor is the adversary, and the consequences of providing competitive capabilities to warfighters too slowly are potential loss of life; not just loss of market.

In the commercial software development world, a potpourri of methods that explicitly address the need for getting valued capabilities to customers sooner have been in use formally for over 10 years. Prior to that, they were used as "lightweight" methodologies for over 30 years and some of the practices have been around since the 1950s. These methods generally are termed "Agile methods."

Agile methods are usually a set of practices. Most Agile methods are comprised of practices that compensate for each other. For example, minimal documentation is compensated for by practices like information radiators; no code reviews is compensated for by pair programming; and minimal documented requirements is compensated for by test-driven development. Chosen piecemeal, Agile practices can leave large gaps and introduce risks. Selecting an established method brings in a set of compensating practices, and reduces risk.

Interest in these methods within the DoD acquisition community has recently been increasing, and successful use of this class of methods has drawn attention. This interest is due to

- a need for an acquisition tempo that responds to operational tempos
- a need to obtain high-quality software within a dynamic environment
- a need to focus on value

This technical note (TN) is the second in a series[1] that addresses different topics of interest to

- Department of Defense (DoD) acquisition offices that are currently pursuing or are contemplating pursuing acquisition strategies that employ one or more elements of a set of incremental development methods commonly termed "Agile methods"
- development organizations for DoD that are currently pursuing or are contemplating pursuing development strategies that employ "Agile methods"
- members of DoD policy-setting organizations, both inside the services and at the level of the Office of the Secretary of Defense

The overall purpose of this technical note is *not* to champion any specific Agile method, but rather to provide acquisition and development personnel ideas about how to approach implementing Agile in their environments. The discussion will raise issues and concerns, and present potential

[1] The first report can be found at: http://www.sei.cmu.edu/library/abstracts/reports/10tn002.cfm.

solutions to those issues and concerns. In addition, we will summarize some background information necessary to understand how Agile *works*.

With this purpose in mind, we address several topics identified in the first TN, but not addressed there. These are:

- Why is DoD interested in Agile methods? There is an increasing awareness that the challenge in today's environment is to provide competitive capabilities to our warfighters in a timely manner to avoid potential loss of life. In addition, with Agile you are more likely to have a system that can continue to change/adapt over time. This should help reduce lifecycle costs.
- What does "being *Agile"* mean in the DoD? The Office of the Secretary of Defense (OSD) is working to streamline the acquisition process for business systems. Agile may be an answer to many of the ideas proposed for streamlining the process. However, adoption of any new acquisition lifecycle requires a change in the prevailing culture. Adopting Agile is not any different. There are differences in perspective on many elements such as organization structure, rewards system, communications, decision-making, and staffing model. To meet the challenges of adopting Agile, a program management office (PMO) can take specific actions that will assist in the adoption and even enable it. Terminology will need to be learned or relearned if terms have different meanings when using Agile. In order to employ any Agile concept, the DoD organization will need to plan for it, train for it, anticipate changes in the environment and business model, and apply the hard work to make the changes a reality.
- Managing and contracting for Agile programs. The management role in an Agile program takes on some added dimensions. Program managers (both acquiring and executing) do not only have to be leaders, they need to be coaches, expeditors, and champions. If not personally performing these roles, they will need someone within their organizations responsible for them. A particular concern is the selection and implementation of the appropriate contract vehicle that supports the Agile way of doing business. Among the people we interviewed for this technical note, the preferred methods were cost-plus or time and material. One corporate proponent for Agile stated that you could use any type of contract vehicle, but some were a lot easier to use than others.
- Technical milestone reviews in a DoD Agile acquisition context. A particular sticking point in employing Agile methods is how to accommodate large capstone events, such as the Preliminary Design Review (PDR), Critical Design Review (CDR), etc. There are many issues in this arena but the main thing to remember is the purpose and intent of holding these reviews in the first place. The purpose is to evaluate progress on and/or review specific aspects of the proposed technical software solution. Thus, expectations and criteria need to be created that reflect the level and type of documentation that would be acceptable for the milestone. This is not any different from *business as usual*. However, the key here is to define the level and type of documentation while working within an Agile environment.
- Estimating in a DoD Agile acquisition context. Estimation done on Agile projects is typically not the same as the traditional methods used on legacy systems within DoD. Agile estimates tend to be just-in-time with a high-level estimate that is refined to create detailed estimates as more is learned about the requirements. Traditional estimation tends to go to a more detailed level up front and details are modified as more information is obtained. Some of the tools within the traditional estimation community are now adding modules to address Agile

estimation. How the specific issues can be dealt with are highly contract specific. This section begins to discuss this subject area.

- Moving toward adopting Agile practices. Change is hard. Understanding the scope of the change is essential. Organizational change methods must be employed to help DoD organizations successfully adapt to implementing Agile. There are multiple adoption factors, such as business strategy, reward system, sponsorship, values, skills, structure, history, and work practices. Each of these must be addressed. Change-management best practices include understanding your adopter population, understanding the cycle of change, understanding your adoption risks, and building transition mechanisms to mitigate adoption risks. Some words of advice:
  - Find and nurture good sponsors for your adoption.
  - Understand the adoption population you are dealing with.
  - Conduct some kind of readiness assessment that addresses organizational and cultural issues.
  - Analyze what adoption support mechanisms you are likely to need for your context and build or acquire them before you get too far in to your adoption.

To address these topics, we conducted a literature search to see what information was available that could be adapted for a DoD environment. We also created an extensive list of topics that we used when interviewing personnel who are Agile corporate advocates, practicing Agile consultants, and personnel working on projects employing Agile methods. The projects ranged from 7-10 people to programs with 100 developers. Some staffs were co-located and others were distributed. Personnel interviewed were from both commercial and government domains. We combined the results to provide some anonymity for our interviewees.

We are planning to continue our exploration of Agile practices in DoD acquisition with future work including, but not limited to, a case study on the Patriot Excalibur program, creation of a contingency model (How do I determine if Agile is right for my project?) and creation of guidance that codifies key concepts and best practices.

Finally, remember that adopting Agile is not for everyone. It is neither a silver bullet nor a cure-all. Agile, like other methods, is better in some environments than other methods. Be sure you understand the potential benefits and risks of adopting Agile. This technical note addresses some of the concerns and issues that need to be overcome when adopting Agile but this is not an exhaustive list. Be prepared, ask for help, and obtain an expert advisor if possible.

Abstract

This technical note (TN), the second in an SEI series on Agile in the DoD, addresses some of the key issues that either must be understood to ease the adoption of Agile or are seen as potential barriers to adoption of Agile in the DoD acquisition context. These topics were introduced in the first TN of the series, CMU/SEI-2010-TN-002. For this TN, the SEI gathered more data from users of Agile methods in the DoD and delved deeper into the existing body of knowledge about Agile before addressing them. Topics considered here include: why DoD is interested in Agile methods; what it means to be *Agile* in the DoD; managing and contracting for Agile programs; technical milestone reviews in a DoD Agile acquisition context; estimating in a DoD Agile acquisition context; and moving toward adopting Agile practices. The authors hope that this report continues to stimulate discussion about and appropriate adoption of Agile in the DoD and federal agencies.

1 Introduction

In this section, we discuss what Agile is and why the DoD is interested in Agile, and we provide background for the report. This is the second report in a series discussing Agile methods within the DoD.

1.1 What Is Agile?

Nothing better reflects the culture and values of the Agile community than the Agile Manifesto developed by the Agile Alliance. This alliance was formed in 2001. Members were searching for an alternative to documentation-driven, heavyweight software development processes. In doing so, they expressed their allegiance to a set of values promoting organizational models based on people, collaboration, and the creation of the types of organizational communities they wanted to work in.

Jim Highsmith zeroed in on the importance of values and culture for succeeding with these Agile methods and wrote, tongue-in-cheek: "At the core, I believe Agile Methodologists are really about 'mushy' stuff about delivering good products to customers by operating in an environment that does more than talk about 'people as our most important asset' but actually 'acts' as if people were the most important, and lose the word 'asset'" [Highsmith 2009]. Therefore, in the final analysis, the meteoric rise of interest in and sometimes tremendous criticism of Agile methodologies is about the mushy stuff of values and culture.

The Manifesto for Agile Software Development (commonly referred to as the Agile Manifesto) states the following:

> *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
>
> - ***individuals and interactions*** *over processes and tools*
> - ***working software*** *over comprehensive documentation*
> - ***customer collaboration*** *over contract negotiation*
> - ***responding to change*** *over following a plan*
>
> *That is, while there is value in the items on the right, we value the items on the left more. [Agile Alliance 2001]*

In Agile terms, an Agile team is a self-organizing cross-functional team that delivers working software, based on requirements expressed commonly as *user stories*, within a short timeframe (usually 2-4 weeks). The user stories often belong to a larger defined set of stories that may scope a release, often called an *epic*. The short timeframe is usually called an *iteration* or, in Scrum-based teams, a *sprint*; multiple iterations make up a *release*. The team's progress toward completion of the iteration is measured via the team's *velocity*. While the code produced within an iteration is useable, it may not have enough functionality to be released to the end user until the multiple iterations that make up a release are completed.

In an environment employing Agile methods, working software is produced at the end of each iteration in an Agile project, and just enough documentation is produced to meet the needs of the team and its stakeholders. Many have speculated that the groundswell of interest in Extreme Programming, Scrum, and other Agile methods, is because the practices largely "define a developer community freed from the baggage of Dilbertesque corporations" [Agile Alliance 2001].

1.2 Why the DoD Is Interested in Agile Methods

Robert Gates, the United States Secretary of Defense, said in a September 2008 speech, "Our conventional modernization programs seek a 99% solution in years. Stability and counterinsurgency missions—the wars we are in—require 75% solutions in months. The challenge is whether in our bureaucracy and in our minds these two different paradigms can be made to coexist" [Gates 2008]. This, and other similar statements by senior DoD officials, express a problem space that is also felt in commercial industry. In the commercial world, the challenge is how to get products to market faster than competitors do, while taking advantage of the latest technologies. In the DoD, the competitor is the adversary, and the consequences of providing competitive capabilities to warfighters too slowly are potential loss of life, not just loss of market share. In addition, one of our reviewers stated that with Agile, one is more likely to get a system that can continue to evolve over time as the customer's needs change. The easier it is to evolve a system, the more likely it is that life cycle costs will be lower, which is important with today's budget pressures.

Gates's concern is reflected in statements by other DoD officials and by Congress itself [OSD 2010]. In December 2010, the Association for Enterprise Information (AFEI) sponsored a one-day forum on the use of Agile methods in the DoD, with a keynote by the Honorable Elizabeth McGrath, Deputy Chief Management Officer of the Performance Improvement Office of the Department of Defense. In her remarks, McGrath noted that the current *average* time from idea to production release for a DoD information technology (IT) system is 81 months. Her office has coordinated a response to Congress for improved acquisition performance for IT systems that includes recommendations favorable to many of the Agile approaches that we have seen used successfully in DoD programs.[2]

1.2.1 The Need for an Acquisition Tempo that Responds to Operational Tempos

These and other statements and activities in the DoD reflect recognition that we must successfully address the difference in the tempo of need (the tempo of the warfighter) and the tempo of provision (the tempo of the developer and the acquirer). A visualization of this challenge is illustrated in Figure 1.

[2] The report to Congress, *A New Approach for Delivering Information Technology Capabilities in the Department of Defense*, was written pursuant to Section 804 of the National Defense Authorization Act for Fiscal Year 2010.

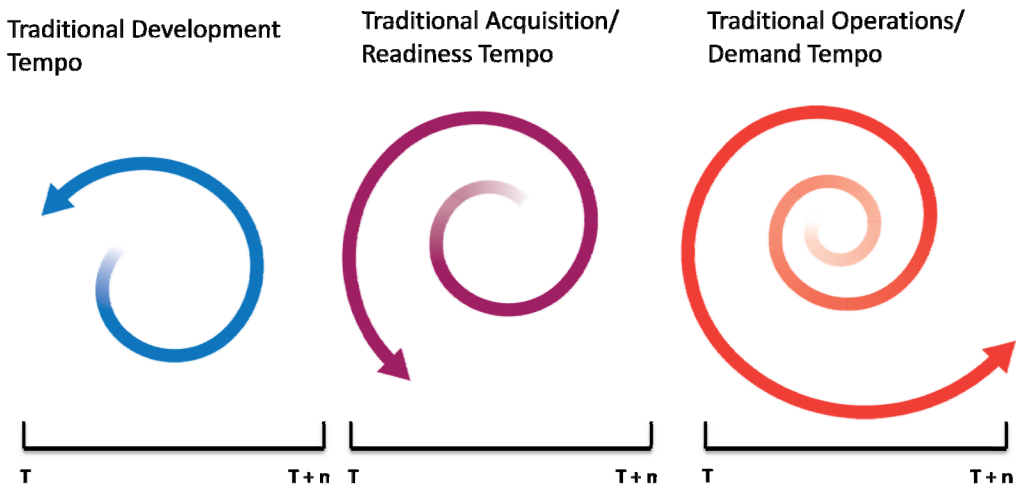


Figure 1: The Disconnect Among Warfighter and Acquisition Tempos [Boxer 2009]

Figure 1 shows the different tempos for traditional development, versus traditional acquisition/readiness, versus traditional operations/demand tempo. The "hotter" colors or larger spiral indicate higher tempo. For each, the timeline is the same but the amount that is accomplished varies as represented by the length of the spiral. This graphically depicts the differences in the amount of work that can be traditionally accomplished as opposed to the need or tempo that operations require. To increase the urgency of this problem, the current operations and demand tempo is accelerating to meet today's demands in the field. The DoD needs to get the tempo of work and tempo of operations more in sync. Slowing the operations tempo is not an option for this synchronization.

Addressing the disconnect between the warfighter/demand tempo and the acquisition/contracting tempo is not easy. The acquisition regulations, rules, and practices that have developed over the years to ensure that taxpayer dollars for DoD capabilities are being spent wisely mostly originated in a time when the U.S. was not engaged in such dynamic warfighting situations as today's. This same acquisition governance also reflects a time of building large, complex systems with minimal software reliance. Today, software-reliant systems are the norm instead of the exception.

Agile practices alone cannot solve the tempo issue. However, one of the common practices of Agile—to involve end users early and often throughout the development cycle and allowing them to change the priority of their needs—does address the tempo issue. By acknowledging that requirements are dynamic, not static, and by going directly to the end users who will be employing the provided capabilities, Agile helps to collapse the time lag between identification of a new threat or demand and its satisfaction. Agile also allows incremental software deliveries to the field as opposed to long delivery times associated with releasing all software at once. In Section 2, we will look more closely at Agile principles that affect tempo and their inherent challenges.

1.2.2 The Need for Rapid Development of Quality Software Systems Within a Dynamic Environment

Another issue that drives the attraction of Agile in DoD contexts stems from the recognition of a need to increase the tempo of acquisition and development while, at the same time, maintaining

high-quality software that ensures effective use of resources in providing needed capabilities. There have been many DoD initiatives that attempt to encourage the use of disciplined acquisition and development practices to obtain and maintain high-quality software—CMMI, Lean, and Six Sigma—are all examples that see both effective and ineffective use within the DoD's portfolio of projects.

Operational effectiveness, customer intimacy, and product innovation are the three strategies that market leaders in commercial industry pursue to achieve dominance. These are described in the book, *The Discipline of Market Leaders* [Treacy 1995]. Most methods used to improve high-quality software are focused on improving *operational effectiveness.*[3] Improving operational effectiveness generally focuses on improving the processes that are internal to the enterprise, as opposed to those that are focused on interactions with customers and end users.

Although Agile methods include very defined internal processes, their focus is actually on another dimension pursued by some market leaders—*customer intimacy*. Customer intimacy as a strategy focuses on deep understanding of a set of customer's needs and solution preferences, regardless of how well they fit with the performing organization's preferences. Operational effectiveness as a strategy focuses on optimizing the processes that produce the performing organization's products and services, with less regard for the deep understanding of customers. Gates's statement about needing a 75% solution in months reflects an acknowledgment that acquirers and developers who are not active in the operational space cannot be expected to provide complete solutions—the operational environment is not sufficiently static to support pre-definition of all the requirements. The Agile focus on direct involvement of end users throughout the development process is a direct reflection of this difference in strategy. At the AFEI DoD Agile Development Conference,[4] one of the recurring themes was how important the continual inclusion of end users was in successful projects using Agile. One of the authors has observed that outside the DoD, and even outside the U.S., organizations are finding that the use of Agile methods combined with other methods like CMMI is a powerful approach to achieving both customer intimacy and operational effectiveness.

When organizations like the SEI started addressing process discipline issues in order to obtain high-quality software in the 1980s, we often expressed a triangle made up of process, people, and technology, illustrated in Figure 2.

[3] In the context of Treacy's book, operational refers to the fundamental processes that produce the work products and services of an organization. Their context goes beyond the military viewpoint of operations to include acquisition and development operations.

[4] NDIA/AFEI. Program, DoD Agile Development Conference. NDIA/AFEI, December 14, 2010, Alexandria, VA.

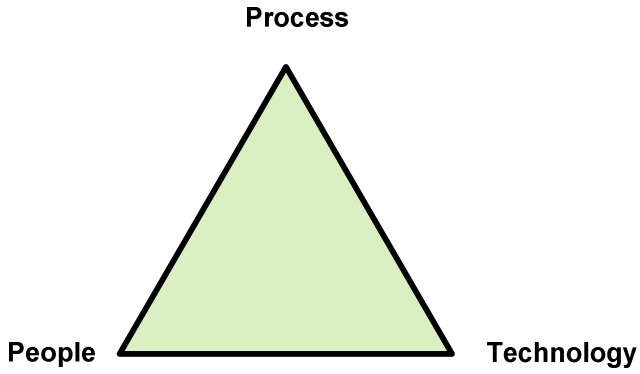


Figure 2: 1980s View of Process Discipline

As understanding of the role of process in supporting the key factors of market leaders—operational effectiveness, product innovation, and customer intimacy—evolved, a more accurate portrayal of the role of process discipline has evolved, as illustrated in Figure 3.

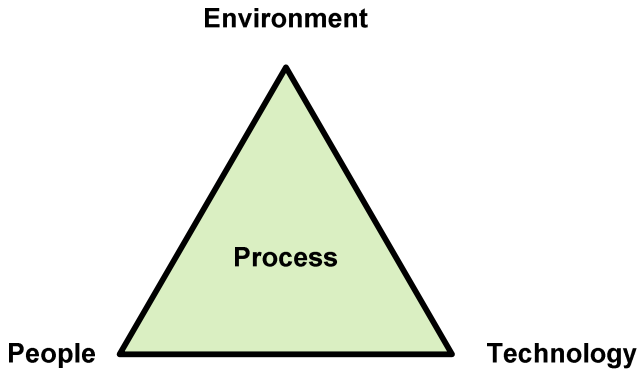


Figure 3: Process Triangle Including Environment [Garcia 2006]

This view of process sees process as an integrating function between technology, people, and their environment. When people and their skills change, the processes need to change; when technology changes, processes usually need to change too. And when the environment—the operational environment, the business or market environment—changes, then processes need to adapt to the new conditions. Incorporating the environment dimension as an explicit aspect to be accounted for in designing and adapting processes is consistent with the Agile view of the operational environment and its dynamism being the source of processes that are meant for adaptation.[5] Achieving high quality in the Agile context requires discipline in the process areas we are accustomed to focusing on, such as operational effectiveness, as well as a new focus on processes for customer intimacy.

[5] Watts Humphrey, one of the great proponents of process discipline and a consistent user of the original process triangle, commented in 2006 that this revised view of the influences on process solves some of the problems that he had experienced in communicating the benefits of disciplined processes.

1.2.3 Achieving More Value with Limited or Shrinking Resources

Historically, the project triangle, also known as the Iron Triangle, is a depiction of the three project attributes or constraints that must be balanced to achieve a successful project outcome: cost, schedule, and scope.[6] As shown in Figure 4, each attribute is shown on the corners of the triangle, implying that how the three attributes are balanced will determine the "shape" of the project's focus. If one attribute is changed, the other two attributes will also be affected. For example, increased scope typically means increased time and increased cost; a tight time constraint could mean increased costs and reduced scope, and a tight budget could mean increased time and reduced scope. Sometimes a fourth attribute, quality, is included and shown in the center of the triangle as it is the ultimate result of the three other attributes. Typically, projects use these three measures (scope, cost, and schedule) to determine the success of the project. Completion within cost and schedule and providing all the scope is the definition of a successful project.

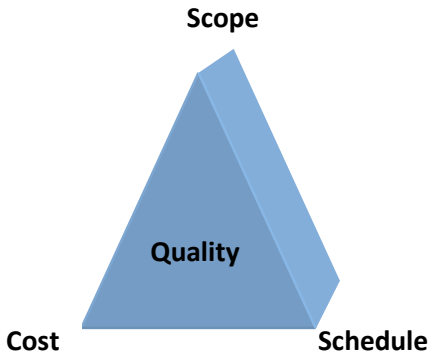


Figure 4: Classic Iron Triangle

However, software development projects often fail because the organization sets unrealistic goals for the Iron Triangle. An example of this came from one of our reviewers. If the government got a requirement to take a simple Hypertext PreProcessor (PHP)/mySQL-based forum type website that already exists in the .com and simply move it to the .mil, it could take $3-5 million and a year to complete. This would include, but not be limited to, documenting a new start, conducting a capabilities assessment, assigning a program manager, finding a host, doing the justification and approval, establishing contracts, getting the vendor and "approved" system for billing, briefing the required oversight groups, and so forth. If this type of requirement occurred within a commercial environment, it would take about two hours and less than $1,000.

"The fact that (particularly SIDRE [software-intensive innovative development and reengineering/evolution]) software development effort and duration cannot be estimated precisely means that it is unwise to try to lock a software project into simultaneously fixed budget, schedule, and feature content (as has been found in many fixed-price, fixed-requirements software development contracts)" [Critical Code 2010]. In the end, if the project team delivers at all, the quality of the delivered product suffers and the project is almost always late and over budget.

[6] The triangle is the historical representation of this idea as well as for process. The two triangles do not represent the same ideas but rather only use the same icon.

With the emergence of Agile, another view of the Iron Triangle has evolved. Jim Highsmith proposed the Agile Triangle as an alternative to measuring performance with the Iron Triangle because Agile is all about being flexible [Highsmith 2009]. Since value is based on capabilities that the users or stakeholders find valuable, scope is the cornerstone of the Agile Triangle. Scope should be considered first and cost and schedule should adapt to achieving the scope. This may or may not be possible, but it is the ideal. Because Agile processes and methods allow for flexibility, customers also gain more *innovation value* in that it is easier for them to be inventive or consider new ideas.

Highsmith has continued to evolve the initial Agile Triangle. The most important items to measure should be value and quality, within the constraints of the program (scope, cost, schedule). According to Highsmith, these are defined as

1. Value: Your project's value should be measured by the stakeholders and what they expect.
2. Quality: The quality part of the triangle means you can deliver a reliable product by adapting to the customer's needs.
3. Constraints: Here is where the three elements of the Iron Triangle appear—project scope, schedule, and cost.

The new Agile Triangle shown in Figure 5 illustrates these attributes.

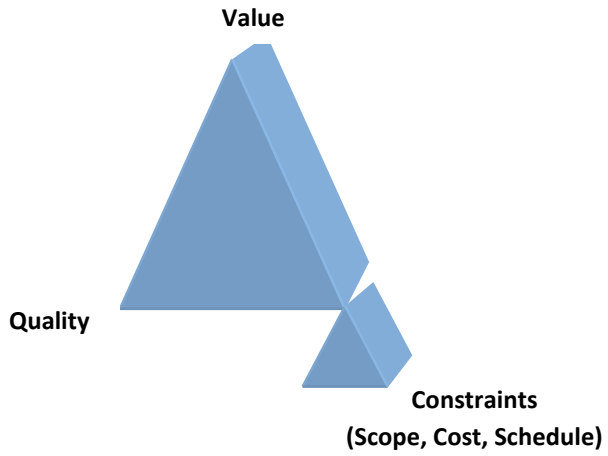


Figure 5: New Agile Triangle[7]

The new Agile Triangle changes the foundational trade-off elements to include value and quality, and keeps the old standards of cost, schedule, and scope in the constraints part of the triangle. This is another way in which Agile addresses Gates's need for a "75% solution in months." By putting the focus on value to end users through such approaches as continual end-user involvement, Agile's philosophy is poised to address explicit DoD needs.

[7] Adapted from Jim Highsmith (http://www.jimhighsmith.com/2010/11/14/beyond-scope-schedule-and-cost-the-agile-triangle/).

1.3 Background

1.3.1 Overall Approach

This report is based on both a literature review and interviews with many diverse programs and practicing Agilists. We conducted a literature search to see what information was available that could be adapted for a DoD environment. We also created a selective but not exhaustive list of topics that we used when interviewing personnel who are Agile corporate advocates, practicing Agile consultants, and personnel working on projects employing Agile methods. The projects ranged from 7-10 people to programs with 100 developers. Some staffs were collocated and others were distributed. Personnel interviewed were from both commercial and government domains. We combined the results to provide some anonymity for our interviewees. Table 1 depicts the general characteristics of our interviewees where available. We also gained information from several other sources that was not program specific but rather based on extensive consulting or tool usage.

Table 1: Characterization of Interviewed Programs

Characteristic	Program #1	Program #2	Program #3	Program #4	Program #5	Program #6
Size of Program in Dollars	$10-15M/yr	$13M/yr	$4.7M/yr	Multiple programs	Multiple programs	Multiple programs
Headcount of Developers	Not releasable	60	19	Varied	Varied	Varied
Headcount of Program Office	Not releasable	9	1-2	Varied	Varied	Varied
Type of Program (IT, C2, embedded weapons, other)	C2, IT, Modeling and Simulation	C2	C2	IT, C2	IT, C2	Commercial
Type of Contract	CPFF, T&M	T&M	CPFF	Varied, preferred T&M	Varied	In house
# Months Using Agile Methods	48-72 depending on program	96	42	Variable	Variable	Variable

Table 1: Characterization of Interviewed Programs (cont'd.)

Characteristic	Program #1	Program #2	Program #3	Program #4	Program #5	Program #6
# Iterations or Deliveries Using Agile methods	20-45 releases: up to 330 builds		207	8	Variable	Variable	Variable
Agile Method in Use	Scrum		XP/Scrum	Scrum	Scrum	Scrum	Scrum
Approx # of Users of Software/System Produced Using Agile Method	Not available		Greater than 670 adopting organizations	Not available	Variable	Variable	Variable

For this report, we will use one of the common definitions of Agile, as cited in our first report on Agile in the DoD:

> ***Agile**: An iterative and incremental (evolutionary) approach[8] to software development which is performed in a highly collaborative manner by self-organizing teams within an effective governance framework with "just enough" ceremony that produces high-quality software in a cost-effective and timely manner which meets the changing needs of its stakeholders [Ambler 2004].*

From a programmatic perspective, Agile should live within the traditional foundational triangle of scope, schedule, and cost. However, practitioners of Agile methods have taken a different approach towards the traditional project triangle as shown in Figure 5.

In order to leverage these new models for program execution, the acquiring organization and the developing organization must consider what changes need to be made to their current mental models and their current practices. The rest of this TN addresses factors that are likely to need change.

1.3.2 Audience for this Technical Note

The audiences for this report are

- senior DoD acquisition policy makers, to advise them on the practicality and policy pitfalls of encouraging the application of Agile software development methods in their programs;

[8] Note that the life cycles illustrated in DoD 5000.02 are also incremental in nature. Agile methods are not the exclusive expression of incremental methods. However, it is also true that a team claiming to use Agile methods must be using, by necessity, an incremental and iterative life cycle.

- members of DoD program offices who may be challenged to undertake a software development acquisition with a contractor who will be using Agile software development methods; and
- software development contractors who are contemplating responding to a DoD Request for Proposal (RFP) with a proposal based on using Agile software development methods.

1.3.3 Content of This Report

The reader may wonder how we selected the topics to address in this report. A subset of this author team published a technical note in 2010 (CMU/SEI-2010-TN-002) in which we acknowledged that the 2010 report only began to explore employing Agile within the DoD. We mentioned several potential future topics. This TN addresses most of them including

- Management—discussion and exploration of governance changes, management style for the Agile PM, and management structure for Agile projects (iteration, release, enterprise).
- Contracts and finance—discussion and exploration of costing and estimation for Agile programs, types of contracts and which works best with Agile, and incentives.
- Benefits from Agile—discussion about how Agile is viewed within the Agile community using risk and a variation of the cost, schedule, quality triangle.
- Organizational change management—discussion of what should be changed to work effectively within an Agile environment, how to go about instituting those changes, etc.
- Culture—definition of the Agile culture, what it relies on, how it is different from existing cultures, and how to bridge the gap.

There are a variety of other relevant topics that could and should be addressed. As we did our research, we created a list of these topics. The potential list of other topics is addressed in Section 7, Summary and Conclusion.

1.3.4 Organization of This Report

The organization and potential audience for each topic in the report is provided below.

Executive Summary contains highlights of this report.

Section 1, Introduction describes what is Agile, why DoD is interested in Agile methods, and the background for the report. All readers should review this section.

Section 2 discusses in more depth what it means to adopt Agile in a DoD context, especially as it concerns the organization's culture. The audience for this section includes policy makers, program managers, and development organizations contemplating moving towards Agile

Section 3 discusses contracting for and managing Agile programs. Its primary audience is acquisition program managers, with a secondary audience of policy makers and development organizations, as well as contractor personnel in various roles.

Section 4 discusses the effects of Agile on technical milestone reviews, particularly System Requirements Review (SRR), PDR, and CDR. Its primary audiences are acquisition program managers and policy makers who are reviewing, approving, and executing acquisition strategies.

Section 5 addresses cost and effort estimation in Agile projects. Its primary audience is acquisition program managers, with secondary audiences of development organizations and independent cost evaluators.

Section 6 provides insight into the use of organizational change management approaches for Agile methods adoption. Its audience includes policymakers, program managers, and development organizations, anyone who influences the environment for Agile adoption or who is substantively involved in the adoption.

Section 7 summarizes the TN and provides suggestions for future topics for detailed exploration.

Several appendices follow the main body of the report, providing more detail or pointers to resources related to the topics of the individual sections.

Appendix A defines acronyms.

Appendix B is the glossary.

Appendix C provides culture details.

Appendix D is a COCOMO factors list.

Appendix E describes an estimating process for Agile based on GAO best practices for estimation.

Appendix F provides details on the Satir Organizational Change Management Model.

Appendix G provides the Adler factors related to complexity and timing of change effort.

Appendix H provides notes from the field.

Appendix I provides selected Agile resources.

2 Implications of Agile Adoption for the DoD Acquisition Context

In this section, we begin with a brief description of Agile themes. This is followed by a discussion of various DoD processes and practices that are likely to need to change in order to successfully adopt Agile. We discuss impacts to the following areas: acquisition life cycle, DoD culture, PMO behavior, and planning practices. Sections that follow address particular subsets of these themes, especially contracting, management, and technical milestones in the DoD acquisition life cycle.

2.1 Agile Themes

Learning and value-seeking are two themes that were visible in our interviews and continue to be emphasized in the rhetoric on Agile methods. Together these themes support an Agile culture. Though they are not the only themes that support an Agile culture, they are ones that differ in their explicit use in DoD acquisition.

Through test-driven development cycles, nested iterations, continuing and frequent involvement of the users of the system being produced, and retrospectives on how well practices have worked, software developers involved in an Agile culture expend significant effort learning about the problem space of the user, the solution space, and the processes that work best for them. One of our interviewees, while reflecting on the importance of retrospectives, said, "I find this to be the *most* beneficial part of the entire process. We do this in the fighter aircraft world. We call it debriefing and we do it better than anyone. [It's] a chance to learn and improve. So this in my opinion is vital."

Communication amplifies individual learning into team learning through *information radiators*. Often, these take the form of big, visible charts that are posted in common areas of the workplace. They should display progress, obstacles, and actions, and be easy to update. Thus, they "radiate" information and support communication and rapid response to changing project conditions. In geographically distributed teams, other mechanisms besides physical wall charts are used for the same purpose, and some vendors of support tools for Agile methods incorporate tools specifically meant for this purpose. One of our reviewers commented on their use of the wall-chart style of information radiator: "[This was] one of the key practices used on [our program] increment 1, which brought about significant change in a short period of time."

Agile development is always value-seeking—concerned with delivering the best possible product to the customer within the available timeframe. Developers emphasize high-quality working software and take pride in providing value by having the flexibility to respond to changes in business direction, requirements, and new needs. "Practices that help the customer to determine what the better solution is and communicate that to the team correctly will deliver business value as well" [Elssamadisy 2009].

The two themes of communication and value-seeking get expressed in different ways within different Agile methodologies and for different product settings. The following discussion addresses these and other themes relevant to DoD acquisition from different perspectives. We address the acquisition context that Agile approaches would be expected to support, elements of

an Agile culture, differences between "doing" Agile versus "being" Agile, Agile success factors and corresponding program management office (PMO) enablers, Agile methods and the DoD culture, and the Agile philosophy related to planning.

2.2 Implications of Adopting Agile for the DoD Acquisition Life Cycle

The Defense Acquisition Guidebook states that "the Defense Acquisition System exists to manage the Nation's investments in technologies, programs, and product support necessary to achieve the national Security Strategy and support the United States Armed Forces" [Defense Acquisition Guidebook a 2011]. The DoD 5000 series (DoD Directive (DoDD) 5000.01 and DoD Instruction (DoDI) 5000.02) provides the basic guidance to implement the acquisition process. The overall life cycle framework is shown in Figure 6.

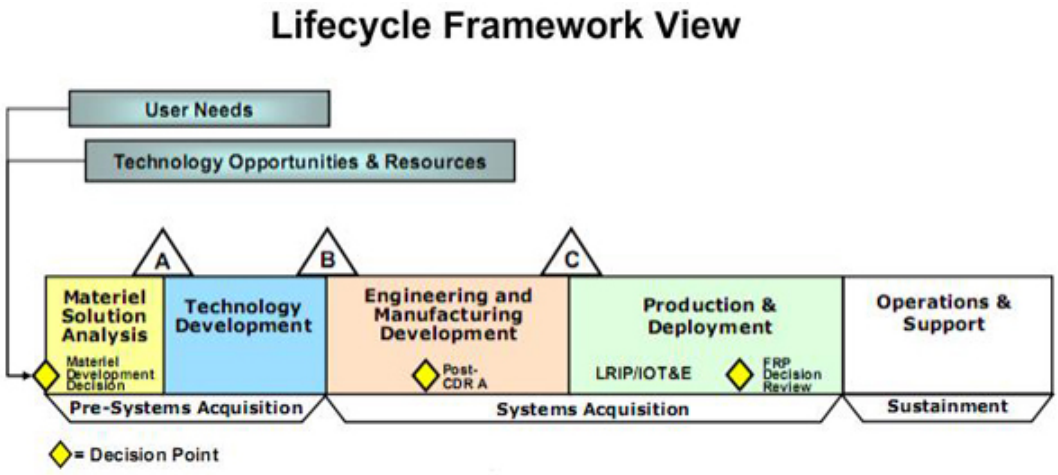


Figure 6: Acquisition Life-Cycle Framework [Defense Acquisition Guidebook 2011e]

In 2009 the Defense Science Board said, "The fundamental problem DoD faces is that the deliberate process through which weapon systems and information technology are acquired does not match the speed at which new IT capabilities are being introduced in today's information age" [Defense Science Board 2009].

The National Research Council has also studied DoD acquisition, stating, "DOD systems acquisition policies, expertise, practice, and culture—including those applied to IT systems—reflect the practices, policies, and cultural norms associated with large weapons systems programs…. With respect to IT, however, there is a longstanding reluctance to deviate from standard weapons system acquisition processes, and acquisition personnel are not trained or led to differentiate the unique aspects of IT systems acquisition" [Committee 2010]. One of our reviewers emphasized the burden this application of weapons system processes on IT has in their IT-focused part of a jet aircraft program: "We are constantly being forced to use the exact same processes as the jet itself and it continues to cost us non-value-added overhead."

These studies and others prompted Congress to include Section 804 in the National Defense Authorization Act for Fiscal Year 2010 (FY10 NDAA, PL 111-84). Pursuant to Section 804, OSD published a report providing updates on DoD's progress toward developing a new acquisition process for information capabilities [OSD 2010]. This new process "will leverage ongoing

Department efforts to streamline Defense Business Systems (DBS) acquisition and incorporate best practices garnered from engagement with industry and lessons learned from ongoing DoD efforts. The new process is intended to take full advantage of the speed of IT innovation from commercial industry to foster an environment for mission-focused and time-critical deliveries that support the full spectrum of IT applications within the DoD" [OSD 2010]. A new business capability life cycle is being created that will change the milestone and budgeting focus for IT systems that are not embedded or driven by technology development. The guiding principles for the new approach are

- delivery early and often
- incremental and iterative development and testing
- rationalized requirements
- flexible/tailored processes
- knowledgeable and experienced IT workforce

Agile methods seem to answer the need for many of the above principles. However, it should be noted that to embark on this new acquisition life cycle, the DoD workforce will need to adapt its culture to work within the new paradigm. It is also worth noting that some programs that are outside the IT systems mandate also use Agile methods, often when using Agile software development in the context of a larger traditional DoD weapons system.

2.3 Doing Agile vs. Being Agile

Agile is a mindset and a way of working that embodies the Agile Manifesto and Agile Principles. The mindset is often the differentiator between successful and unsuccessful Agile projects [Sidky 2009]. However, many organizations start their journey by "doing Agile," via adoption of the methodologies and practices commonly called Agile.

There are a number of Agile processes—Scrum, XP (Extreme Programming), RUP (Rational Unified Process), DSDM (Dynamic Systems Development Method), ASD (Adaptive Software Development), and others. Each provides potential benefits through its own practices' reflection of Agile principles and values.

A development team can start "doing Agile" by picking an Agile process and following it step by step without fully embodying the Agile culture described briefly in Table 2. This often provides incremental benefits through smaller and more focused delivery of operational software. However, adopting Agile for software development is more than learning a new process. "Being Agile" is more about creating and sustaining a culture of agility. This culture is founded upon key principles expressed in the Agile Manifesto. Rather than simply advocating a new software delivery process, these Agile tenets seek to change what the team values, measures, and delivers, (i.e., placing value on collaboration and personal interactions, working software, and adjustment to change).

Table 2 presents the principles (in bold) that tie to the Agile Manifesto, as well as Agile behaviors that support these principles, and possible government PMO actions and enablers that can support the use of Agile methods, where appropriate. This translation is intended to assist acquisition program managers to gain a better understanding of the Agile development teams they may be working with. Items in italics are direct quotes from the Agile Manifesto. Other behaviors listed

are ones that we have observed in our interviews that are consistent with the manifesto. The main cultural themes reflected in the table are leadership values, team values, values around incentives and reward systems, and values around development practices. Although these values may also be present in non-Agile development contexts (in fact, many of them are derived from pre-Agile experience of the Manifesto authors), they are behaviors that are iconic of Agile environments.

Table 2: Agile Manifesto Principles and Possible Program Office Enablers for Them

Agile Manifesto and Principles	Common Behavioral Expressions of the Agile Manifesto and Principles	PMO Actions & Enablers
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	Team performance is emphasized more heavily than individual performance. Candor and truth are rewarded, not punished. Distinction is clear between individuals' status on the development team (respect) and formal HR rewards. The team visibly rewards its heroes. Mentoring is rewarded.	At least some of the reward system is team-based. Provide rewards other than monetary (choice assignments, mentoring, training, etc) Downplay or equalize merit increases Associate career accomplishments and milestones with promotions. Let the development team naturally recognize its heroes. Include an appreciation step in iteration retrospectives.
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. - and - Working software is the primary measure of progress.	Early functioning software release is rewarded (vs. documents are "finished"). Focusing early on the parts of the system that have the most direct business or end-user value Use of user stories to communicate between developers and other stakeholders	Provide developers with access to real end users or to end-user surrogates with depth in operational use of the type of system being built. Architect system in a way that permits early delivery of working (although often prototype) software. Minimize time for delivery increments and base requirements choices around ability to get early feedback from operational use.
Business people and developers must work together daily throughout the project.	Vision is shared by sponsor and teams. Local sponsors respect team autonomy. Local sponsors (usually acquisition customer) visibly value team input in management decisions. Management willingness to be creative within the business constraints of the environment Support working well across boundaries Managers' first duty is to remove impediments to developer productivity.	Co-locate customer and end user(s) with developers Stakeholders, including the customer, participate actively in standup meetings. Establish award fee, CDRLs, and other contractual criteria that encourage small deliveries of working software rather than a "big bang" delivery. Work with relevant certification authorities to reduce lag time to end-user delivery while maintaining appropriate security.

Table 2: Agile Manifesto Principles and Possible Program Office Enablers for Them (cont'd.)

Agile Manifesto and Principles	Common Behavioral Expressions of the Agile Manifesto and Principles	PMO Actions & Enablers
Simplicity—the art of maximizing the amount of work not done—is essential. Keep everything—software, documentation, overhead—as simple as possible.	Bloated software and *gold-plating* are discouraged Minimizing waste is important.	Reduce distractions for developers and testers; consider physical facility adjustments like a team room, if feasible. Keep team on task; minimize team members being shared among multiple projects. Model appropriate product owner behavior in daily standup meetings.
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	Fail fast, learn fast is encouraged Time-boxed iterations (also called "sprints" in some methods) Scope management within release cycle(s)	Reduce risk via early working software that is releasable to users Consider use of time and material contracts where feasible—*software development as a service*
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	Co-located teams whenever possible Strong telepresence technology support for geographically distributed teams Developer isolation is frowned upon. Strong team-based values Information sharing vs. "knowledge is power"	Candor, communications, transparency Minimize barriers to use of collaboration technology when co-location is infeasible.
Continuous attention to technical excellence and good design enhances agility.	Peer reviews are embraced by development teams. Learning is valued. Shared team understanding of what "done" means	Encourage pair or small-team programming. Develop Agile coaches for both acquisition side and development side. Define acceptance test early to close in on definition of "done."
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Self-reflection via things like retrospectives is encouraged. Metrics used for course correction, not punishment of individuals	Formalize team reflection, via retrospectives or other mechanisms, as part of release cycles. Keep metrics simple enough to be gathered and still be useful. Focus on improving development team velocity.
Agile processes promote sustainable development. - and - The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Sustainable pace is rewarded. Personal heroics are not encouraged. Little overtime is seen. Uncompensated overtime is rare.	Focus on meeting shorter-duration delivery dates by modulating scope.

Table 2: Agile Manifesto Principles and Possible Program Office Enablers for Them (cont'd.)

Agile Manifesto and Principles	Common Behavioral Expressions of the Agile Manifesto and Principles	PMO Actions & Enablers
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Small amount of change occurring within an iteration, but a fair amount of reprioritization of requirements in-between iterations Product owner minimizes churn with the team inside an iteration, but brings up changes in the environment and end-user needs in-between iterations	Leave slack in each iteration for important, urgent defect removal or requirements change—how much depends on team velocity and character of software being developed.

2.4 Implications of Adopting Agile on DoD Culture

What can be observed in any organization? Every organization has common things that are visible and reflect its basic assumptions, shared values, espoused values, and artifacts as shown in Figure 7. Together these form a view of the organization's culture [Schein 2009]. Artifacts are the most visible elements of a culture. They include the products of the groups and everything an organization experiences, from the facilities and properties of the physical environment to the group's language, reports and documents, posters and presentation materials, myths, stories, and rituals.

"Information radiators" are a good example of an *artifact* of Agile culture. The big, visible version of these charts can take many forms, for example: a storyboard of user stories that highlights the iteration's status, burndown charts that show the number of requirements and stories that are completed, status of the continuous integration builds, roadblock charts that show obstacles and who has responsibility for working these, and so on. Distributed teams also need information radiators. One of our reviewers commented, "A large flat-panel screen in each lab/team room is a wonderful information radiator for distributed teams." In an Agile culture, it is critical that the artifacts are dynamic, actionable expressions of the project at work, not static documents (e.g., project mission statements that are not expected to change frequently). Cultural artifacts are easy to see but can be hard to interpret, and they alone do not represent the essence of a culture.

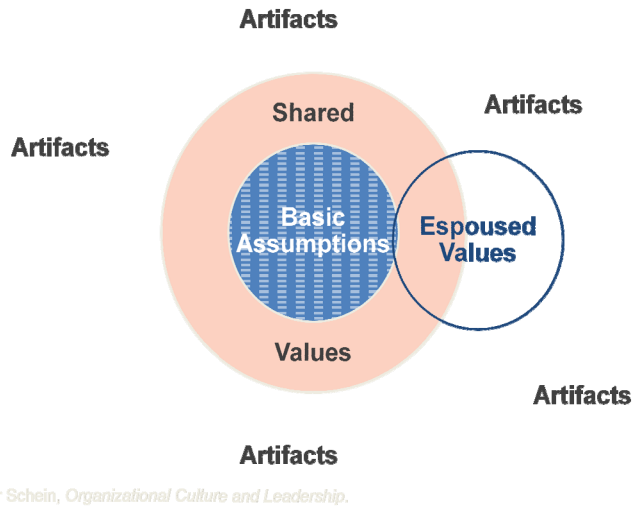


Figure 7: Key Elements of Culture[9]

The Agile Manifesto is a good example of the *espoused values* of the Agile Alliance, the originators of the term "Agile methods," and it generally reflects the values to which Agile teams aspire. Espoused values (things we say we do), norms, and rules make up the everyday operations of a workplace. Espoused values become *shared* when they have been proven as solutions and the group accepts the solution's value. Solutions (actions and values) that are repeatedly used with success eventually become *basic assumptions*—general inferences that are not grounded in substantiated research but are unquestionable [Schein 2009]. For example, continuous integration may start as an espoused value in an Agile organization (this is a rule we will follow), but until there are sufficient tooling mechanisms to support continuous integration, it will not become a shared value, a common practice that most of the organization would not readily abandon. Should continuous integration become automatically accepted within the group without its use being discussable, it would become a basic assumption for that particular culture.

The relationship between values and assumptions is important. "If the espoused values are reasonably congruent with the underlying assumptions, then the articulation of those values … can be helpful in bringing the group together, serving as a source of identity and core mission" [Schein 2009]. In situations where there is tight coupling or congruency, the Agile cultural identity will be very strong. In those situations where there is less congruence, evidenced by more explanation, rationalization, and aspiration around the values, the Agile cultural identity will be weaker.

Common manifestations of Agile principles, through shared values and behaviors, are highlighted in column two of Table 2. In some of the organizations we interviewed, we saw evidence of strong shared values related to Agile principles, particularly in programs where Agile had been successfully used for more than a couple of years. We saw other cases, particularly where the Agile adoption was relatively recent, of a weaker Agile cultural identity. In one program, with a strong Agile identity, prospective employees are likely to be interviewed by all their potential team members, not just the team lead. The team makes a judgment as to the fit of the employee,

[9] Adapted from Edgar Schein's *Organizational Culture and Leadership*, 1992.

not only in terms of software development skills, but also in terms of their perceived ability and willingness to adopt pair programming and test-driven development practices; both of these are anchors of Agile practices in that organization. In a more traditional project, full team involvement in interviewing and deciding on new members is not as common an occurrence. One reviewer, who shares the practice of team interviewing with this program, commented, "In almost all cases that I have seen, the team is better off without adding a team member versus adding one who does not jell with the rest of the team or work well in a team environment."

An expected, but still striking difference we saw in Agile programs versus more traditional software programs was the attitude toward requirements. In the more traditional programs that some of our interviewees had worked on, a tremendous amount of effort, energy, and attention was paid to refining requirements statements to a low level of detail and obtaining multiple levels of approval of them before doing any prototyping or visible design activities. The confidence of the development team in how accurate the requirements were in terms of the user's needs was low. However, they could not substantively move forward with the design until the complete systems requirements had been approved. A reviewer who had a similar experience said, "To date, I have not seen where the rigor applied to our requirements has brought any value to our product." The attitude toward requirements in the programs we interviewed using Agile was much more focused on a definition of only enough capability to bring acknowledged value to an end user, with much less focus on ultimate completeness of the requirements set. In the program using Agile, the team was confident that with the user involvement that they knew they could count on, they would discover additional requirements as the project evolved and that all the requirements they worked on would provide definable value to the customer.

The shared value of prioritizing end-user involvement to build confidence in requirements early allows both the acquirer and developer to permit appropriate ambiguity early in the project. This is contrary to what is more typical in a traditional acquirer/development team, which strives for *completeness* of a requirements specification prior to significant design work. Although this is a false completeness (almost everyone we spoke with acknowledged that the complexity of today's systems makes it impossible to completely know a system's requirements prior to design), the shared values of the acquirer/developer team are focused on the completion aspect of the requirements rather than allowing evolutionary learning about them.

Here is what one colonel tells his subordinate PMs about requirements and release planning:

> *Planning Release 1 … [should contain the] smallest fieldable amount with tangible ROI that the customer will agree to, and*
>
> - *should demonstrate that the system will work end-to-end … or a logical chunk of it that can be built upon.*
> - *Pick the easiest requirements that have the best quality data.*
> - *Fight for this. The customer wants it all in the first release … every time.*
> - *Give success earlier, validate assumptions, buy time to further analyze the harder things.*
> - *If budget cuts take everything except the release 1 money, you are still a success.*
>
> *Planning Release 2 thru 'n' … use a model.*

- *You must know: the interfaces required, the quality of the data required, the hardware required, and the software required ... with costs and estimates of complexity for each.*
- *Create a model based on the variables above ... use it for estimates going forward; continuously refine.*
- *If you have to go around the world to field each increment at each unit, it means many trips;[that must be] synced with [the] deployment schedule ...[which will cost lots of] $$$ ... think this thru carefully.*

Culture is ingrained in the organization, and is intertwined with everything that the organization is and does, including these dimensions:

- organizational structure
- leadership style
- rewards system
- communication and decision-making styles
- staffing model

Table 3 compares the above cultural dimensions as reflected in typical Agile and traditional DoD environments.[10] This is not a comprehensive characterization, but is intended for purposes of comparison. It also provides an informal way for a DoD organization to understand how closely its current culture reflects Agile cultural elements. Although this table may seem like it advocates Agile elements as superior to traditional DoD elements, for adopters of Agile, the point and the challenge is that the Agile adoption is likely to occur *within* the traditional DoD culture and steps will need to be taken to mitigate risks related to cultural conflict that is likely to arise.

An Agile culture runs counter to the traditional DoD acquisition culture in many ways, from oversight and team structure to end-user interaction throughout development. In our interviews with successful DoD Agile projects, we have consistently confirmed that adopting Agile methods requires a mindset change for government program management offices and other acquisition entities the projects interact with, such as the Office of the Secretary of Defense (OSD). The first step in effecting a culture change of this type is to understand the differences in assumptions, shared values, and artifacts that make up the cultures of Agile projects and those of more traditional projects as executed in the DoD [Schein 2009].

[10] These are generalizations and by nature will not be present as stated in every Agile or DoD traditional program situation. Some DoD program offices, for example, have created excellent collaboration structures through the judicious use of integrated product teams.

Table 3: Comparison of Agile and Traditional DoD Cultural Elements

	Agile DoD	Traditional DoD
Organizational Structure	Flexible and adaptive structures Self-organizing teams Collocated teams or strong communication mechanisms when teams are distributed	Formal structures that are difficult to change Hierarchical, command-and-control-based teams Integrated product teams that have formal responsibilities
Leadership Style	Facilitative leadership Leader as champion and team advocate	Leader as keeper of vision Leader as primary source of authority to act
Rewards System	Team is focus of reward systems Sometimes team itself recognizes individuals	Individual is focus of the reward system
Communications & Decision Making	Daily stand-up meetings, Frequent retrospectives to improve practices Information radiators to communicate critical project information Evocative documents to feed conversation "Just enough" documentation, highly dependent on product context	Top-down communication structures dominate External regulations, policies and procedures drive the focus of work. Indirect communications, like documented activities and processes, dominate over face-to-face dialogue Traditional, representational documents used by the PMO throughout the development life cycle to oversee the progress of the developer PMO oversight tools focused on demonstrating compliance vs. achieving insight into progress
Staffing Model	Cross-functional teams including all roles across the life cycle throughout the lifespan of the project Includes an Agile advocate or coach who explicitly attends to the team's process	Uses traditional life-cycle model with separate teams, particularly for development and testing Different roles are active at different defined points in the life cycle and are not substantively involved except at those times

Agile culture, by contrast, is the culture of just enough. "Just enough" is, of course, contextually defined. Just enough documentation for a mobile game application versus just enough documentation for a missile navigation system is likely to be quite different, both in volume and in character. An Agile approach would dictate just enough

- detail and process planning to ensure compliance throughout the program
- easily identifiable original work to tailor the process to the program
- review cycles to ensure the plan is well understood by all stakeholders
- reviewers to ensure the plan represents the desired level of compliance for the program

2.5 Implications of Adopting Agile for DoD Planning Approaches

Agile practitioners conceptualize a software development project as a value-seeking activity in which knowledge is continually being acquired about the product and the user needs that make it valuable. While most mature Agile development organizations recognize that a certain degree of upfront analysis is important, at the core of Agile is a fundamental belief that knowledge acquisition will and must progress throughout the course of a software development project. One group of interviewees emphasized that, unlike other programs they had worked on, they were allowed to accept that the requirements will change as a routine—rather than exception—condition. Agile encourages interaction among the entire stakeholder team to determine what is required to satisfy the current user needs. Thus, they do enough analysis so that user stories (their way of expressing requirements) assigned to the current iteration can be accomplished, but allow for continuing accumulation of data for stories that will be implemented at a later date. This approach also allows them to not waste effort on items that may change or might be interpreted incorrectly. Another group said that Agile allows them to get a product out in a timely manner because the user environment is constantly changing; they could never know all the requirements. As the users, environment, and technology change, updates and changes can be made to the original product. Non-Agile DoD programs have explicit mechanisms for addressing change, of course. However, generally speaking, those change processes, once a baseline has been posted, require a significant amount of time and effort to navigate prior to a change being actionable. This reflects a mindset of change as an exception, versus change as routine.

Agile practices focus on creating a development environment that acknowledges and supports continuous knowledge acquisition. For this reason, detailed upfront requirements specifications and detailed, long-range, task-level project plans are not used within Agile development projects. Agile projects generally reflect a multi-level approach to both requirements specification and project planning.[11]

Rolling wave planning, an element of earned value management, is a particular approach to resource planning and management that is prevalent in DoD acquisition [Defense Acquisition Guidebook b 2011]. A time period for a wave is defined for the project (often three months, but not always), and the next wave is planned in significant detail, while waves that follow it are generally planned at a much higher level. The thinking, which is similar to that in the Agile community, is that learning occurs as we go and detailed planning beyond a certain window is counterproductive. However, where rolling wave practice departs from Agile practice is in the foundation of each. Rolling waves assume a complete and static set of requirements that will be *worked off* in a pre-planned fashion. Agile practice assumes that, as the developers and users learn together, requirements priorities and even the need for them will shift, and each iteration provides an opportunity to realign the developer's priorities to the user's needs. Interviewees working this way did not provide direct cost comparisons between using baseline change requests (BCR) and shifting requirements priorities between iterations. However, those who had worked previously in more traditional programs did state that the decreased level of formality at least provided a sense of collaboration between user and developer that was missing in other programs they had worked.

[11] Note that multi-level, rolling wave planning is not unique to Agile projects; however, it is one of the practices that have become the basis of several Agile methods.

2.6 Summary

To sum up, Agile is a different culture, but one that can be incrementally adopted and tailored to the existing conditions within the DoD. Things like oversight, cross-acquisition/development team structure, and end-user interaction throughout development are areas where incremental steps must be taken in many settings, rather than trying to adopt all of Agile at once. In our interviews, we saw a variety of adoption approaches, several of which were explicitly incremental to ease the cultural transition. The use of Agile methods and the oversight of Agile programs require a mindset change, not just a practice change, for the DoD and other government entities. For example, a PMO that embraces the Agile principle that values operating code over extensive documentation may require a different set of CDRLs when formulating a contract. This not only requires a change in perspective, but also the creation of appropriate governance models, via tailoring DoD 5000.02 and CDRLs from such events as SRR, PDR, CDR, etc. The PMO involved may have to seek waivers from higher up the acquisition chain, and these higher-ups must also understand Agile methods if they are to understand what they are waiving. One of our reviewers cited a recent contract using Agile methods, in which they were bounded by an SDR milestone, but obtained approval to have IDRs (Incremental Design Reviews) beyond that time instead of the traditional PDR and CDR cycle.

At the recent AFEI Forum on Agile in DoD, acquisition strategies that promote Agile practices via tailoring of regulations and standards were seen as one of the ways that Agile methods could more usefully be employed.[12] One of the barriers to this kind of tailoring is the uncertainty of program managers as to whether the desired tailoring will be approved, since these types of tailoring reflect a different approach than is expected by DoD policy makers.

In programs that wish to take advantage of the benefits seen by other Agile programs in the DoD, both the government PMO and the contractor developer need to be aware that different skills sets or skill mixes will likely be needed in programs using Agile. Agile takes a lot of strong, focused team and management oversight at the middle and low levels. To achieve the benefits of any of the Agile methods commonly in use, the DoD acquisition organization attempting Agile methods for the first time will have to

- plan for them
- train themselves in Agile concepts as well as ensure their developers are adequately trained and skilled in Agile methods
- anticipate the changes needed in their environment and business model, especially as they relate to increased end-user involvement
- work hard to make the changes a reality

If the team can stay together, the work of sustaining Agile approaches is usually much less than the effort required to initiate them. These topics are discussed in Section 6, Moving Toward Adoption of Agile in DoD IT Acquisition.

[12] NDIA/AFEI. Program, DoD Agile Development Conference. NDIA/AFEI, December 14, 2010, Alexandria, VA.

3 Adopting New Management and Contracting Practices for Agile Programs

In this section, we discuss the characteristics and practices that are generally associated with contracting for and managing Agile projects. The viewpoint we take is primarily that of a project or program manager who is directly involved with oversight of an Agile development team. In an acquisition program, we saw examples where a government employee filled this role, as well as examples where the person in this role was a contractor employee. We also differentiate, where appropriate, managerial behaviors of a manager on the acquisition side who is not directly supervising a development team. The topics address the adoption of Agile using management and contracting practices best suited for use with Agile programs. Practices for continued execution are not specifically addressed in this report. This could be a topic for the future.

3.1 Common Agile Management Traits

The interviewees who provided source material for this report uniformly acknowledge that the Agile approach to project execution places demands upon all personnel that differ from other execution environments. The managerial role is uniquely affected by the features of the Agile approach.

The Agile program manager has more direct interaction with the development teams than is typical in a development project. This is in no small part due to the flatter organizations that comprise Agile development teams. In addition, the use of iterations demands that the program manager be more aware of and supportive of short-term planning. In large organizations, there may be more than one individual designated to fulfill the span of activities, in effect a management team. Note that in the role descriptions that follow, either multiple people could hold a role, or a single individual can hold one or more of the roles. This section takes inspiration from ideas advanced by Dean Leffingwell, then alters and expands those ideas to address inserting Agile practices into DoD acquisition [Leffingwell 2008].

3.1.1 Executing-Side Program Manager[13]

The managerial role for the organization executing a program exploiting Agile methods is distinguished by traits described below.

Leader. The program manager provides leadership and should spend more time with the team than behind the office desk. Consistent contact in this *high-touch* environment fosters team communication and cohesion. The "trust factor" is essential in the Agile environment, and the ability to delegate important tasks to others is essential. It is very helpful for the program manager to have a personality compatible with these kinds of practices.

Coach. The program manager acts as a coach, not a command-and-control supervisor. Instead of telling the team what to do, the coach seeds the team with ideas and allows the team to solve the

[13] The executing-side manager could be a development contractor or part of an organic government team, such as an Air Logistics Center team.

problem on its own. For instance, the coach might ask, "Have you thought about how to do this better?"

This departure is driven by three features of this project environment:

- There is a need to help new people learn new behaviors, especially if they have not been exposed to it before.
- Established members of such teams will tend to be disciplined, able to act as mentors to new people, and require less traditional supervision.
- There is a need for collaboration, among team members and across teams.

Expeditor. The program manager must be vigilant in establishing an environment that fosters successful execution of individual iterations and the overall project. The discipline of the time-boxed iteration magnifies the effect of organizational and operational impediments. The more intimate involvement of the manager in the day-to-day program execution provides a basis for identifying operational impediments that reduce the probability of success, either through personal observation or by receipt of timely feedback from the development team.

Champion. The program manager must deal with upper-level management and stakeholders. For some time, part of the manager's job will be to provide adequate visibility into an execution model that will be unfamiliar, if not foreign, to upper-level management and other managerial stakeholders. The program manager is likely to find it necessary to act as the translator to effectively communicate with this constituency. In at least one of the programs we interviewed, this was seen as one of the most critical elements of the managerial task. In fact, some have said performing the champion role will be the greatest challenge while adopting Agile development methods within DoD contractor organizations.

Ambassador. A key set of stakeholders are the prospective users and subject-matter experts whose sustained participation is necessary for successful execution. The program manager will need to cultivate relationships with these people and their leadership to ensure this participation.

These personas should be familiar to most program managers. The difference is the perspective from which the personas are implemented. Specific Agile training or certification would make it much easier to accomplish these roles in the Agile environment.

3.1.2 Acquiring-Side Program Manager

For the manager operating on the customer side, inside an acquisition organization, the above personas are still present, but with variations in emphasis.

Leader. The leadership role for the acquiring-side manager requires establishing and maintaining relationships with the leadership of the executing organization. The likely geographic distribution (not a particular attribute of Agile projects, but a fairly commonplace occurrence in today's programs) places additional demands on the acquiring organization. As such, the acquiring-side manager may delegate representative(s) to be on site with the executing organization to maintain adequate visibility into the emerging product(s). Electronic means are also effective but lack the immediacy offered by physical presence. In at least one of the programs we interviewed, the weekly presence of the acquiring-side manager was seen as a great benefit to the development

team and the end users, because that individual could bring insight from the field and from the policy-making parts of the program organization to the development team in a timely fashion.

Coach. The acquiring-side manager serves as a coach for those people who are the direct contacts with the program in execution, such as the testing community, information assurance personnel, operations personnel, etc.. That coaching spans the kinds of interactions the acquiring-side manager wants to have with the development team and the execution-side manager. It also includes defining the nature of the information and metrics the acquiring-side manager wishes to receive. The acquiring-side manager has a direct stake in the nature and quality of the interaction between subject-matter experts and the development team, so the coaching role extends there as well. Much of this coaching will involve helping existing personnel make the transition to the fast-tempo, high-interaction environment that typifies Agile projects.

Expeditor. The acquiring-side manager has a significant challenge in efficient deployment of the people directly interacting with the development team and its management. For DoD projects, the norm for larger programs is that the development team will be geographically dispersed. Identification of the best distribution of the available staff will be an ongoing challenge for the acquiring-side manager. Securing appropriate status information that does not unduly interfere with the tempo of Agile development will be a matter of negotiation and establishment of trust between the acquiring manager and the execution manager, as well as the staff doing the day-to-day work. In addition, the norm is that end users and developers do not have much, if any, direct interaction. Finding and getting access for developers to the right end users in a timely fashion within a culture accustomed to separating these two groups is a challenge for DoD teams using Agile methods.

Champion. The acquiring-side manager is responsible for maintaining buy-in by external funders and stakeholders. It is unlikely that these stakeholders will be familiar with the dynamics of Agile projects, which places an additional burden upon the acquiring-side manager to provide a portrayal of project status and accomplishments that is accurate and to bridge the cultural gap. In addition, the champion removes obstacles, rather than beating up the contractor for every risk that pops up or retracting award fees. As we said about the executing-side manager, some have said performing the champion role will be the greatest challenge while adopting Agile development methods within the DoD.

Ambassador. The acquiring-side manager must ensure the appointment of end users or subject-matter experts to work with the developers. These could be proxies if actual end users are not available.

3.1.3 Product Owner

A separate but equally important management role for the acquiring side is the product owner. The product owner will

- define the features of the product
- decide on release date and content
- be responsible for the profitability of the product (return on investment [ROI])
- prioritize features according to market value
- adjust features and priority every 30 days, as needed
- accept or reject work results [Kovatch 2009]

Thus, one of the product owner's roles includes adjudicating conflicting requests for change. The product owner is responsible for the mechanism that ensures that conflicting user requested changes are resolved and that the system does not stray from its vision through a thousand little changes.

3.2 Lessons Learned Implementing Agile

Given the above management behaviors that are needed for successful management of Agile projects, there are some lessons learned from our interviews characterized in Table 1 that particularly focus on management issues. We will discuss lessons in team building, increments/iterations in the larger scheme of the project, metrics, and geographic distribution.

3.2.1 Incentivizing Teams

A manager's role in team building, whether on the acquisition or execution side, is an important one and is reflected primarily in how the manager establishes trust with the development team and other stakeholders, and in the behaviors that the manager chooses to reward or punish. Honesty and open communication were key attributes for the manager that we heard over and over during interviews.

For example, in an Agile project, one of the goals is to understand the team's velocity with enough accuracy to plan iterations reliably. When this is achieved, not only does the team tend to deliver on time, they tend to do it without a lot of fuss. Rewarding the team effort that is exemplified by this case is appropriate as it reinforces one of the key tenets of Agile—collaborative teams. However, rewarding individual heroes who work abundant overtime to pull out a delivery, rather than addressing the root cause such as poor estimation of story points, will work against the team's ability to perform effectively. Many team members said they liked being able to provide estimates that were realistic and see the success of their estimates at the end of the sprint. One reviewer stated that their first Agile implementation had a 50% increase in productivity. In addition, the manager's annual employee survey showed amazing improvements. This manager gave the team the authority to make decisions and took on the attributes that were mentioned under the *executing-side manager* in Section 3.1.1.

Another aspect of team building that is sometimes difficult for managers experienced in leading traditional projects is giving the team more autonomy in the selection of goals and in the selection and application of rewards. Agile teams told us that they found that the entire team understood

and worked toward the selected goal. In addition, all team members understood and agreed to the expectations for each iteration. Many managers are not accustomed to allowing the team to decide how it will distribute rewards or to self-organize into the roles that need to be performed to accomplish the project goals. Successful Agile teams that we interviewed all said that a focus on team awards such as team lunches and fun activities provided positive reinforcement. One team also said that 20% of the raise pool was given to the team to allocate.

One impediment to facilitating team building in a DoD environment is the common perception that team building exercises are a "waste of taxpayer dollars," not reimbursable and thus counterproductive. However, forming productive self-supporting teams requires some amount of focused team building. Collaborative, self-organizing teams are a key tenet for Agile. Therefore, a strong argument can be made for the use of team building exercises when forming or even continuing the use of an Agile team.

3.2.2 Mastering the Iteration

In Agile projects, the time box for accomplishing a useful unit of work usually refers to a 2-4 week period (iteration) where a small set of the functional requirements (expressed as user stories) will be moved through design into implementation and initial testing. These iterations have a different technical and business rhythm than most traditional projects are accustomed to accommodating and executing. From a management viewpoint in particular, the movement of small elements of functionality/capability through the entire development life cycle in less than a month presents communication issues to middle/upper management accustomed to seeing more of a "*complete* the requirements, then *complete* the design, then implement" approach. During the interviews one of the most often-cited benefits of Agile projects, for both end users and development teams, is the ability to accomplish *something real* in a short time, get feedback, and make course corrections soon enough to affect the overall outcome. Team members and end users get a sense of accomplishment, have an increase in team morale, and see actual usable software at the end of each iteration. The users found that they were getting what they asked for quickly and this encouraged them to continue using the product. Some environments may have constraints such as rigorous configuration management, which can delay delivery to the customer. Even in these circumstances, demonstrations can be done to show the customers what they will be receiving.

DoD projects often have external requirements, like air worthiness certification or information assurance certification, that are not as easily accommodated within the iterations.[14] Thus, additional time is needed to meet the external requirements. Wherever possible, time between completion of a release's functionality and delivery to the end user is minimized. The projects we interviewed have used various methods for addressing this. Sometimes they add an iteration at the end of a release cycle to deal explicitly with certification. Sometimes the certification activities are performed outside the development iterations framework altogether. Whenever possible, certification activities are included as tasks within an iteration. Especially when users or user

[14] There are myriad commands, services, DoD regulations, guidance, requirements, and reports that need to be considered depending on the project. Examples include DoD Architecture Framework, Standard Financial Information Structure, Federal Finance Management Improvement Act, service test and evaluation agency rules, and DoD Inspector General.

surrogates are actively involved in the evolution of a capability, the delays that can occur due to external testing or certification, though necessary, may seem to be a hindrance.[15]

One interviewed program had a hybrid approach to certification. Some of the work was done before and after the iterations and some was done during the iterations. The sprint teams worked with the Information Assurance (IA) group before the iterations to define the latest security rules for their environment based on the latest guidance. Once the environment was defined, the teams would plan for their next iteration(s) within that environment.[16] Their plans would be reviewed by the IA team for compliance before the iteration started and after the iteration finished. If at any time during the iteration the development team had questions about staying within the defined IA environment, the IA team would come in to assist. This process was very successful for them and they also passed an external audit of this process.

From an acquisition manager's viewpoint, managing the rhythm of the iterations means negotiating windows of opportunity for planning actors such as user involvement, certification, and independent field evaluation so that those activities will contribute optimally to the tempo of the release. The acquisition manager also has a role in designing the technical milestones review schedule and its contents. The content of the technical milestone reviews have to be correlated to the content of the planned iterations. This particular area is sufficiently challenging that Section 4 is devoted to technical milestones in an Agile project.

3.2.3 Determine Appropriate Metrics

The old adage *what usually gets measured, gets done* is used to advantage in Agile projects. What gets measured in Agile projects, on the execution side, are

- elements in a product backlog (the master list of all functionality desired in the product usually grouped by user stories and epics—groups of related user stories) and the rate at which they are being addressed (typically called the "burndown rate"). While some of these items may never be implemented, this information could be used for trending data.
- the speed and effort related to completing one unit of work—usually a story point—an estimate of the complexity of individual stories that can be aggregated and used predictively. This is usually called "velocity" in an Agile project. However, there is a good bit of variation in exactly what is collected and how it is reported within the Agile project. This metric also varies per team, so care must be taken not to misuse or misconstrue the meaning of this metric. It is more appropriate for planning purposes.
- progress toward release—a variation of the burn down rate that looks at completed stories versus estimated stories for a particular release to the field or end users
- technical debt—loosely defined as the volume of lines of code that are poorly written, poorly refactored, do not follow coding standards, and are not supported with sufficient unit tests,

---

[15] The reader can perform an internet search on Microsoft Agile Security Development Life Cycle for an approach to certifications within an Agile life cycle. A detailed discussion of this topic is not included in this technical report.

[16] IA and other crosscutting special requirements define the environment in which a product is developed, integrated, and/or tested. They need to be dealt with ahead of the development effort.

and the amount of code duplication.[17] Less technical debt is better. If technical debt is allowed to grow too big; it could lead to unmaintainable code and eventually stop the entire development effort. The overall quality of the product is better with smaller technical debt. Measuring technical debt helps achieve good quality and avoid undesirable results.

Most Agile methods are supported by tooling that simplifies the collection of the raw data that goes into establishing the baselines for these measures. Some projects are small enough to use Excel spreadsheets, but most find that either the free tools widely available across the internet or commercially available licensed tools make the collection and use of appropriate measures efficient. For information beyond determining appropriate metrics see Section 5.4.4 which describes Agile release tracking.

A note of caution is appropriate for metrics. Within the DoD environment, OSD and the individual services[18] use models to justify the life-cycle cost estimates. These models may use function points, RICE (reports, interfaces, conversions, enhancements or extensions) objects, or lines of code. Thus, the metrics would be different from those obtained from the Agile project using stories. Therefore, the program manager would end up keeping at least two sets of metrics by necessity. This is another disconnect (one not addressed further here) that needs to be resolved for the Agile project to run smoothly.

As with any measurements, using the data to punish individuals or teams is rarely productive. This is especially true in Agile environments, where this practice goes against Agile principles, where trust is a hallmark of successful projects and appropriate use of metrics to gain insight into the development process is crucial.

A special case for measurement that relates to estimation is measurement that supports a basis of estimate. In our section on cost estimation, we will discuss a frequently used measurement method in DoD projects, Earned Value Management, and its application in Agile projects.

3.2.4 Overcoming Challenges with Geographic Distribution of Projects

Some agilists insist that collocation is an absolute requirement for successful Agile projects; however, experience in both industry and the DoD programs we spoke with contradict this viewpoint *if* geographic distribution is managed effectively. In some ways, all the good management advice related to any distributed project applies here: ensure multiple communication pathways, have guidelines for when to move discussions among different communication modes, provide an appropriate amount of face-to-face time to promote cohesive team building, and so forth. In addition, the information radiators,[19] which are unique to Agile implementations, provide a good source of communication. Of course, there will be times—such

---

[17] http://software.intel.com/en-us/blogs/2009/05/29/agile-best-practice-3-of-10-measure-and-frequently-repay-technical-debt/

[18] An OSD model is Cost Assessment and Program Evaluation [CAPE]. For the Army, the Deputy Assistant Secretary of the Army for Cost and Economics (DASA CE] uses the cost model.

[19] This term was coined around 2000 by Alistair Cockburn while standing in a Thoughtworks office, looking at all the paper on the walls around him. “Information radiator” refers to a publicly posted display that shows what is going on to people walking by. Information radiators work best when they are big, very easy to see (e.g., not online, generally), and change often enough to be worth revisiting. See http://alistair.cockburn.us/Information+radiator.

as for briefings to oversight groups—when the team or at least part of the team (PM, prime contractor, and user's representative) must be collocated.

Beyond the guidance that applies to any distributed project, there are some particular issues as described by some of our interviewees that could come up when using particular Agile methods:

- When using pair programming, try to avoid pairs that are geographically distributed. When they need to be distributed, supportive technology like webcams, telepresence software like Skype and GoToMeeting, and desktop sharing software, along with sufficient bandwidth to make it all useful, have been used in some of the programs we talked to.
- Assigning a bounded set of functionality to one location whenever possible is a useful strategy. This division of labor follows the division of functionality, which takes advantage of the "natural" boundaries, and containment found when developing separate functionality.
- Synchronize iterations so that there is time for remote groups to access integration and regression testing environments. Often an extra day needs to be added to allow transfer, processing, and return of data (programs, actual user data, test scripts, etc.). Figuring out the right points for movement back and forth of data between teams is sometimes needed to ensure everyone stays productive.
- Continuous integration and regression testing environments are one of the biggest challenges for distributed teams. The physical act of moving large amounts of data to a central location for integration proved to be an initial issue for at least one interviewed project. However, in their expeditor role, interviewed managers who manage distributed Agile projects were key resources for teams in getting the right infrastructure and bandwidth for each site. Of course how a program intends to implement continuous integration and regression testing could also have security issues depending on the tools used and the security posture of the program.
- One program has a periodic face-to-face users' conference of all the teams and stakeholders involved in the project. The program characterizes the conference as "an essential element of their strategic integration of the using community into their development process." The agenda includes updating everyone on the current state of the product, its user base, release plans, customer testimonials, etc. It is a different type of interaction between the users and the developers, but one that is greatly appreciated by the entire team. Though expensive (rental of conference site, travel, time of participants), the program believes that its motivational value outweighs the cost by a good measure. The overall conference not only provided benefit to the developers in that they meet the users, but the user community also gets to meet the developers and asks questions first hand.
- One reviewer also suggested that sometimes you might choose to distribute teams deliberately to accomplish a specific goal. For example, when merging two companies in two different locations you might deliberately compose teams of people from both locations to start forming a single new company culture.

Establishing management behaviors and infrastructure that is tuned to Agile methods is a challenge, but the managers we interviewed found the effort to be worthwhile. Many of these managers took risks with their own managers to *protect* the Agile culture that they were building. The payoffs in terms of user satisfaction and productivity were unanimously considered

successful. This "protecting the team and team culture" mentality is a critical part of being an Agile leader.

3.3 Contracting Issues and Solutions for Agile

One of our government reviewers said, "This is the single biggest barrier to the government operating like a commercial entity. We take 5 to 10 times longer to execute routine contract actions."

A particular management concern for Agile methods in the DoD, especially from the acquisition side, but also from the execution side, is the selection and implementation of appropriate contracting vehicles to support the types of practices that successful Agile projects exhibit. Due to the iterative nature of Agile and its propensity to accept (even welcome) change, many contracting vehicles present unique challenges for employing Agile methods. A particular issue is the reporting and milestone requirements often levied against DoD contracts. These may be especially difficult to meet using Agile methods. For example, one of the interviewed programs was entering PDR. The biggest issue was to ensure the government review team understood what was included in PDR and what was not. Requirements assigned to iterations in the future would not have any design detail available, as opposed to a traditional PDR where everything would have some type of design available at PDR. Section 4 discusses technical milestones in detail.

Another challenge is to construct any contract type so that it rewards working software and meaningful milestone review compliance rather than just traditional artifact production. Several of the interviewees stated that achieving rewards for working software that is balanced with sufficient documentation was a challenge.

This section is not meant to be an all-inclusive and exhaustive discussion of the contracting issues associated with employing Agile methods. Rather, it is only a brief introduction to the types of contracts and some associated issues. Four of the most common contracting vehicles are discussed in this section, along with comments about what we found in our interviews in relation to the use of these contracting vehicles. One interviewee specifically stated that any contract type could be used; some were just easier to work with than others.

3.3.1 Cost Plus Incentive Fee (CPIF)[20]

CPIF contracts are used for a wide variety of purposes in the DoD. In terms of Agile development, a CPIF vehicle is a reasonable contracting vehicle to use when (as is often the case with Agile projects) requirements will be quickly evolving and redirection of the tactical details of the software is expected. The incentive fees can be constructed around measures and motivators that support Agile methods, if the acquisition manager is cognizant of the kinds of management and measurement aspects, as we have described above.

The management team of one of our interviewees credits its CPIF contract structure as a key success factor. The program delivers multiple releases per year of ever-improving functional requirements despite having zero software requirements specified in the enabling contracts.

[20] For information on contracting vehicles discussed here, as well as other contract types typically used in the DoD, please refer to http://www.dtc.dla.mil/dsbusiness/Info/contracts1.htm or http://guidebook.dcma.mil/18/ContRecRevconttypes.htm.

This is possible because the contract is a services type contract for software engineering support. This allows the development team to avoid situations where a contractor is obligated to satisfy a requirement that has become irrelevant or obsolete. The team can respond to changing needs and priorities without onerous contracting actions.

3.3.2 Fixed Price (FP)

There are a variety of fixed price contracts: firm fixed price, fixed price incentive, fixed price with economic price adjustment, fixed price redeterminable, and fixed price level of effort.[21] The government prefers this type of contract because it encourages the contractor to contain costs. Fixed price contracts are used with Agile projects in industry, but usually after the product backlog and critical elements for each release are defined. In industry, the contracts are generally short in time span in comparison to what is typical in the DoD.

3.3.3 Indefinite Delivery Indefinite Quantity(IDIQ))/Delivery Orders

IDIQ/Delivery orders, or task orders, comprise the largest percentage of contract type in terms of the programs that we interviewed. Some of the benefits were that IDIQ vehicles provided the most flexibility for adjusting to changing operational needs and provided more opportunities for close working relationships between the developers and the end-user community they are serving. Although none of our interviewed programs expressed it exactly this way, IDIQ types of vehicles can be managed more like a service contract, where the service being provided is on-demand software evolution, rather than like a product contract. Some of the author team sees this framing of software development as a productive service for Agile projects in the DoD, one that is supported by the prevalent use of the IDIQ contract type.

However, just because IDIQ provides flexibility, the government still needs to require an estimate in cases where a release is mapped to (or the subject of) a task order. The releases closer to being performed should have better estimates than those further out in the schedule. That is, the immediate releases will have deliberate estimates and those further out will have planning estimates. If the estimates change, the contractor needs to provide a reason. Otherwise, the contractor may not look ahead further than a single increment.

One potential frailty of the IDIQ contracting model, particularly for a large overall program, is the management of multiple delivery order contracts running in parallel. Even if the art of the iteration has been mastered by the team(s) involved, there is a potential for the focus of program management to be limited to each of the delivery order contracts in isolation, particularly if metric reporting is limited to the scope of individual delivery orders. Make sure that oversight is across the entire program rather than focused on individual tasks, and keep a continuing focus on quality and integrity within product management rather than just looking at the project management detail. While this may seem obvious to the seasoned program manager, remember that your team is learning how to work with Agile and may need to be reminded of the basics.

[21] http://guidebook.dcma.mil/18/ContRecRevconttypes.htm

3.3.4 Time and Material (T&M)

T&M contracts are used when it is not possible to estimate accurately the extent or duration of the work or to anticipate costs with any reasonable degree of confidence.[22] From a government perspective, this type of contract requires significant oversight to assure that the contractor is performing efficiently and using effective cost control measures. From a contractor perspective, it fits rather well with the typical Agile planning and estimation process. T&M contracts may be the most successful of the contracting types discussed here for use with Agile. In fact, many of the interviewees, both contractor and government, commented that T&M contracts were the easiest to use.

3.3.5 Contracting Vehicles Summary

Different contracting vehicles can work for Agile methods. Acquirers must be savvy about the interpretation limits of different contracting regulations that affect the type of contract they are contemplating. In cases where we saw awareness of differences in the cultural norms of Agile and traditional acquisition teams, the contract vehicle chosen was not an impediment to getting the desired results using Agile. In other words, the contracting personnel were aware of how Agile methods worked: change is part of the norm, not an exception, they knew what would be delivered when using those methods, and they adjusted their expectations accordingly within the limits of the contracting rules and regulations.

One of the key management and contracting aspects in the DoD is the structure and requirements for technical milestones. The next section addresses technical milestones explicitly, within the context of DoD management and contracting.

[22] http://guidebook.dcma.mil/18/ContRecRevconttypes.htm#Time and Material

4 Technical Milestone Reviews

The DoD 5000 series provides the framework for acquiring systems. This framework includes a series of technical reviews including but not limited to System Requirements Review (SRR), System Design Review (SDR), Software Specification Review (SSR), Preliminary Design Review (PDR), and Critical Design Review (CDR) among others. Historically, documents such as MIL-STD-1521, Technical Reviews and Audits for Systems, Equipment, and Computer Software, or the United States Air Force Weapon Systems Software Management Guidebook are used as guidance for conducting these reviews.[23] Some smaller programs do not require this level of review; however, programs that do require them expect a certain level of documentation and rigor regardless of the development methodology employed.

These milestone reviews are system reviews. From a systems point of view, some would say these reviews are not part of software development methodology and that this is an incorrect use of the terms. For purposes of this report and in the view of the authors, software intensive systems typically are subjected to PDRs, CDRs, and other reviews. While the overall system may be a plane, tank, ship, or satellite, the software still must pass the PDR, CDR, and other milestones. If the system in question is an IT system, then the PDR, CDR, and other reviews apply directly, as software is the main component of the system (along with hardware to run it on). This section is aimed at readers who are contemplating using Agile methods and who are subject to the typical review activities prevalent in DoD acquisitions. The authors hope that this section will provide some useful guidance on how to approach Agile methods when following traditional technical milestones.

4.1 Milestone Review Issue

In the SEI report *Considerations for Using Agile in DoD Acquisition*, the authors stated: "A very specific acquisition issue and sticking point is that Agile methodology does not accommodate large capstone events such as Critical Design Review (CDR), which is usually a major, multi-day event with many smaller technical meetings leading up to it. This approach requires a great deal of documentation and many technical reviews by the contractor" [Lapham 2010]. The types of documentation expected at these milestone events are considered *high ritual* and are not typically produced when using Agile.[24] If the PMO intends to embrace Agile methods then it will need to determine how to meet the standard milestone criteria for the major milestones reviews, particularly SRR, SDR, PDR, and CDR. However, Agile can be adapted within many different types of systems. One of our reviewers said that "the Atlas V heavy launch guidance system is produced using eXtreme Programming and has all the DoD procurement program events." The

[23] Note that this report does not address the major milestone decision points, i.e., Milestone A, B, or C. Only technical milestones are addressed here.

[24] "High ritual" is a term used within the Agile community often interchangeably with "high ceremony." Alistair Cockburn defined *ceremony* as the amount of precision and the tightness of tolerance in the methodology [Cockburn 2007]. For instance, plan-driven methods such as waterfall are considered high ritual or ceremony as they require an extensive amount of documentation and control. Agile methods are generally considered low ritual as the amount of documentation and control should be "just enough" for the situation.

key here is knowing how long or how much effort is required to review the program in question (an hour or a week).

In addition, according to one of our reviewers who has monitored Agile-based projects from a government perspective:

> *[I] found that in projects using an Agile methodology, technical reviews are around delivery of major capabilities and/or leadership is invited to certain sprint reviews. [I] also found that technical reviews are not as needed as much because stakeholders buy-in to stories/development/schedule at the end of every sprint, which we found is more productive than large reviews every so often.*

In order to determine the type of criteria needed for any review, the intent of the review or purpose must be known. This raises a question: What is the purpose of any technical milestone review?

4.2 Intent of Technical Milestone Reviews

The Defense Acquisition Guidebook (DAG) says,

> *Technical Assessment activities measure technical progress and assess both program plans and requirements. Activities within Technical Assessment include …the conduct of Technical Reviews (including Preliminary Design Review/Critical Design Review Reports)…A structured technical review process should demonstrate and confirm completion of required accomplishments and exit criteria as defined in program and system planning…. Technical reviews are an important oversight tool that the program manager can use to review and evaluate the state of the system and the program, redirecting activity if necessary. [Defense Acquisition University 2011a]*

From a software perspective, each of these reviews is used to evaluate progress on and/or review specific aspects of the proposed technical software solution. Thus, expectations and criteria need to be created that reflect the level and type of documentation that would be acceptable for those milestones and yet work within an Agile environment.

4.3 Challenges

The intent of any technical milestone review is for evaluation of progress and/or technical solution. For PMOs trained and experienced in the traditional acquisition methods, evaluating program progress and technical solutions follows well established guidelines and regulations. Very specific documentation is produced to provide the data required to meet the intent of the technical review as called out in the program specific Contract Data Requirements List (CDRL). The content of these documents and the entry and exit criteria for each review is well documented. However, even in traditional acquisitions (using traditional methods), these documents, exit and entry criteria can be and usually are tailored for the specific program. Since the documentation output from Agile methods appears to be "light" in comparison to traditional programs, the tailoring aspects take on additional aspects. Some of the specific challenges for Agile adoption that we observed during our interviews that must be addressed are

- incentives to collaborate (see Section 4.4.1)
- shared understanding of definitions/key concepts (see Section 4.4.2)

- document content—the look and feel may be different but the intent is the same (see Section 4.4.3)
- regulatory language (see Section 4.4.4)

Many of these challenges are not specific to adopting Agile in DoD acquisitions but are also common to other incremental development approaches such as Rational Unified Process (RUP). In fact, RUP does address this level of milestone review. The reviews are called Lifecycle Architecture (LCA) Review and Lifecycle Objectives (LCO) Review. Some potential solutions to these challenges for Agile adoption are discussed in the following sections.

4.4 Agile Success Depends on Tackling Challenges

The PMO that is adopting or thinking of adopting Agile for an acquisition must set the stage to allow for the necessary collaboration required between the PMO and other government stakeholders and between the PMO and the contractor(s). The introduction of Agile into DoD acquisition can be considered yet another type of acquisition reform. The Defense Science Board has provided some cautionary words on any novel approaches for acquiring IT:

> *With so many prior acquisition reform efforts to leverage, any novel approach for acquiring IT is unlikely to have meaningful impact unless it addresses the barriers that prevented prior reform efforts from taking root. Perhaps the two most important barriers to address are experienced proven leadership and incentives (or lack thereof) to alter the behavior of individuals and organizations. According to the Defense Acquisition Performance Assessment Panel, '… current governance structure does not promote program success—actually, programs advance in spite of the oversight process rather than because of it.' This sentiment was echoed by a defense agency director in characterizing IT acquisition as hampered by the oversight organizations with little "skin in the Game." [Defense Science Board 2009]*

4.4.1 Incentives for Acquirers and Contractors to Collaborate

While this topic is relevant to all contracting issues, it is particularly important if the "world of traditional milestone review" collides with the "world of agile development." The authors have observed programs where collaboration was not yet optimal, which resulted in major avoidable issues when milestone review occurred. Thus, we emphasize them here.

We take the comment "skin in the game" to mean that the parties involved must collaborate. Thus, some form of incentive to do so must be created. (As the Defense Science Board noted, "the two most important barriers are leadership and incentives or lack thereof."). It is key that leadership convey clear goals, objectives, and vision for people to create internal group collaboration and move toward success. One of our interviewees likened an Agile program to a group of technical climbers going up a mountain: you are all roped together and if one of you falls, you all fall. If the Agile project is thought of in these terms, it is quite easy to have an incentive to collaborate. The key for DoD programs is to determine some incentive for both contractor and government personnel that is as imperative as the life and death one is for technical climbers.

Incentives need to be team based, not individual based. Agile incentives need to cross government and contractor boundaries as much as is practical, given regulations that must be followed. Each individual program has a specific environment and corresponding constraints. Even with the

typical constraints, the successful programs we interviewed had a *program identity*, which crossed the contractor–government boundary. The incentives to collaborate were ingrained in the team culture with rewards being simple things like lunches or group gatherings for peer recognition.

It is incumbent upon the PMO and the contractor to negotiate the appropriate incentives to encourage their teams to collaborate. This negotiation needs to consider all the stakeholders that must participate during the product's lifecycle, including external stakeholders. The external stakeholders (i.e., special review panels, other programs that interface with the program, senior leadership in the Command and OSD, etc.) need to be trained so they understand the methods being used, including, but not limited to, types of artifacts that are produced and how they relate to typical program artifacts. Without specific training, serious roadblocks to program progress can occur. Buy-in occurs with understanding and knowledge of Agile and the value it can bring. Defense Acquisition University (DAU) is beginning to provide some training on Agile methods and the associated acquisition processes.

Incentives should not be limited to just the contractor team but also encouraged for the government team as well. Government teams are often incented to grow their programs and follow the regulations. However, these actions should not be incented in an Agile environment unless the users' needs are met first.

The type of contract could also play into the incentive issue. One of the interviewees using a T&M contract stated that they had a limited structure for rewards mainly due to the structure of the contract, which provided limited motivation for the contracting company to provide rewards as the contract personnel were relegated to the role of body shop. With this said, we observed a very collaborative environment on this program.

In order to collaborate, all the parties must have a common understanding of terms and key concepts. According to Alan Shalloway, Agile "has to be cooperative [in] nature and [people] are not going to be convinced if they don't want to [be]" [Shalloway 2009]. We have addressed a set of Agile and DoD terms for which common understanding is required in Section 2.3.1. Some additional key concepts specific to supporting technical reviews in an Agile setting are discussed in the following sections.

4.4.2 Shared Understanding of Definitions/Key Concepts

While there are numerous key concepts involved with Agile development methods, the two we heard most about during our interviews were type and amount of documentation and technical reviews.

One of the key concepts related to Agile is the type and amount of documentation that is produced. Typically, Agile teams produce just enough documentation to allow the team to move forward. Regulatory documentation is still completed and may be done in a variety of ways. In our interviews, we saw several different approaches to producing regulatory documentation that was not considered to contribute directly to the development activities, but will be needed in the future (user manuals, maintenance data) or to meet programmatic requirements:

- A SETA contractor was hired by the program office to review the repository of development information (embedded in a tool that supported Agile methods) and produce required documentation from it.

- Technical writers embedded with the Agile team produced documentation in parallel with the development activities.
- Contractor personnel doing program controls activities produced required documentation toward the end of each release. The contractor program control personnel took the outputs from the Agile process and formatted them to meet the 5000 required documents.

Note that if the documentation is not of any value to the overall product or program, then waivers should be sought to eliminate the need.

Another key concept is that the whole point of using Agile is to reduce risk by defining the highest value functions to be demonstrated as early as possible. This means that technical reviews like PDR and CDR will be used to validate early versions of working software and to review whatever documentation had been agreed to in the acquisition strategy. Thus, additional documentation would not be created just to meet the traditional set of documents since that set would be tailored a priori.

4.4.3 Addressing Expected Document Content Mismatch

As stated above in Section 4.2, "A structured technical review process should demonstrate and confirm completion of required accomplishments and exit criteria as defined in program and system planning." Thus, the documents produced for any of these reviews should support the required accomplishments and exit criteria that were defined in the program and system planning, and reflected in the acquisition strategy chosen by the program.

As stated in Section 2.5, Agile planning is done at multiple levels, similar to a rolling-wave style of planning, though usually in smaller increments than is typical of rolling waves. The top-level overview of the system and architecture is defined and all releases and associated iterations (or sprints) are generally identified up front. The releases and associated iterations to be undertaken soon have more detail than those to be undertaken later—the further in the future, the less detail. Given this approach to planning, how does document production fit it? Typically using traditional methods, the same level of detail is provided for all sections of a document and then refined as time progresses. This approach does not work for programs employing Agile methods. Thus, the way documents are created will need to evolve just as the methods for development are evolving. Future methods could employ tools that examine the code to extract information to answer predefined and ad hoc questions.

For today's Agile-developed work, because the work is being accomplished using an incremental approach, one should expect the documentation will also be accomplished in that manner. However, as discussed in Section 4.4.2, several of the programs interviewed have used varying approaches to creating the documentation. None of the approaches are better than the others; they are just tailored for the specific situation found within the program.

The intent and content of each artifact should be considered and agreed upon by both the PMO and the developer. The current Data Item Descriptions (DIDs) applied to the contract can be used as a starting point for these discussions. Once the intent and content are determined, appropriate entry and exit criteria can be created for each document. Keep in mind the timing of the releases and iterations when setting the entry and exit criteria so that the content aligns with the work

being performed. Each program we interviewed took slightly different approaches to creating documentation. In all cases, the content and acceptance criteria were determined with the PMO.

4.4.4 Addressing Regulatory Language

While we do not know of any regulations that expressly preclude or limit the use of Agile, many in the acquisition community seem to fear the use of Agile because of their prior interpretations of the regulations [Lapham 2010]. Thus, care should be taken to ensure that all regulations are met and corresponding documentation is produced. At the very least, there should be traceability from the Agile-produced documentation to the required documentation. Traditional documentation as defined by CDRLs follow well established DIDs. Agile methods do produce documentation; however, the form is not the same as that prescribed in the DIDs, although the content may be the same. This difference tends to spark issues about documentation unless the contractor and the government PMO agree on some type of mapping. If needed, waivers can be requested to reduce or eliminate some of the regulatory implications.

Many of the current regulatory practices are at the discretion of the Milestone Decision Authority (MDA). In order for Agile to succeed, the MDA will need to be supportive of Agile processes. As one of our interviewees pointed out, ideally more responsibility would be delegated to PMs as they are more closely associated with Agile teams. In fact, this issue was one that a particular program continually ran into.

There are several endeavors underway to help define a more iterative or incremental acquisition approach for the DoD. These include the IT Acquisition Reform Task Force, which is responding to Section 804.[25] In addition, a smaller white paper effort is underway in response to Secretary Gates's solicitation for transformative ideas to improve DoD processes and performance, titled Innovation for New Value, Efficiency & Savings Tomorrow (INVEST).[26] The white paper submitted to INVEST originated within the Patriot Excalibur program at Elgin Air Force Base and suggests specific language changes to some regulations to make it easier for those choosing an acquisition strategy to understand how they could effectively use Agile methods.

4.5 Potential Solutions for Technical Reviews

Many DoD programs require technical reviews; however, there is very little written about accomplishing these types of reviews when employing Agile methods. Written guidance on performing such Agile-related reviews is beginning to emerge at a high level (e.g., the 804 response paper), and several programs have done or are doing these reviews. The information presented here is based on many interviews performed during the research for this technical note and on the ideas of the author team, which arose during spirited debates.

First, the underpinning for any solution has to be developed. This is the agreement the PMO and developer have made about how to perform their reviews. The reviews are planned based on the program and system planning. Given the incremental nature of Agile work, the most productive

[25] http://www.afei.org/WorkingGroups/section804tf/Pages/default.aspx

[26] http://www.defense.gov/home/features/2010/0710_invest/. At least one of the programs we interviewed submitted a paper to this contest sponsored by Secretary of Defense Gates.

approach to technical reviews will be incremental in nature also. We recognize that an incremental approach to reviews may not always be possible.

Our interviews showed that there tend to be two ways to accomplish the technical reviews:

1. Traditional technical reviews, like those that have existed on "waterfall" programs, have been added to an Agile process.
2. Technical reviews are an integral part of the Agile process and as such are a series of progressive technical reviews.

4.5.1 Traditional Technical Reviews

Where Agile methods are being used in a program to develop software, we saw two common approaches to dealing with technical milestone reviews:

- The program proceeded as though the software was being developed via waterfall as traditionally practiced, and the developer was responsible for ensuring that they produced the expected documentation and satisfied the entry and exit criteria.
- The program office or developer performed necessary reviews based on the Agile methods in use and then translated the Agile documentation into the expected artifacts for the review.

In either case, a translation step exists. In some instances, the developers are performing the translation internal to their program and presenting the standard artifacts one normally sees at the technical reviews. This approach has been called *covert Agile*. The developer and presumably the acquirer are still getting some of the risk reduction benefits from employing Agile but may be losing some of the potential cost savings due to creating the *translation layer*.

The other form of traditional technical reviews is one where the PMO is fully aware of the Agile method being employed by the development contractor. In this instance, the PMO does not want its developers dealing with the translation but rather hires a Systems Engineering and Technical Assistance (SETA) contractor to take the "raw" data produced from the Agile method and uses it to produce the required artifacts.

4.5.2 Progressive Technical Reviews

Progressive technical reviews use each successive wave of reviews to build on its predecessors. What does this mean? The program has only one PDR, CDR, etc., of record. However, when Agile is employed, the complete set of information for all requirements that is normally required for a PDR will not be available at that time. Thus, the available information will be used for the PDR with the understanding that multiple "mini" or progressive PDRs will occur as more information becomes available. One program we worked with is using this approach. Another program uses mini-PDRs and CDRs where they obtain signoff by the customer representatives. This is used to prove that the customer signed off, saying "they liked it." This type of sign-off can be used to document performance throughout the program. A third program calls these Incremental Design Reviews (IDRs). Currently, SDR is the only milestone besides IDRs that they use. At SDR, they establish the vision, roadmap, and release plan for the remainder of the project (which spans multiple contracts). At the end of each release, they have an IDR. Each IDR will deliver threads of functionality defined at SDR.

Some of the information available for the PDR will be far more detailed, as it will represent completed functionality from the iterations that are already finished. For example, in Figure 8 the relationship of a PDR to a release and its associated sprints is shown. This notional diagram shows that the PDR is programmatically scheduled for June and the CDR is scheduled for December (milestone timeline). The iteration timeline shows all iterations. Iterations 1-5 are completed by June. Thus, there are five iterations completed by the PDR date. The PDR would include data from iterations 1-5.

The PDR[27] also needs to demo a working, high-level architecture for the entire program implementing a critical set of functionality from the iterations completed by the PDR. This provides a demonstrable risk-reduction activity. In addition, PDR should also show how the requirements have been tentatively allocated to each release. For releases nearer in time (i.e., those closer to execution) the requirements will have been allocated to individual sprints. The PDR should also address at a high level the external interfaces and the infrastructure for the program. This information is used to provide context and enable the overall review.

The PMO should consider holding CDRs for each release, which would ensure the next level of detail is available and that the highest value capabilities are done or broken down to a level that is ready for implementation.

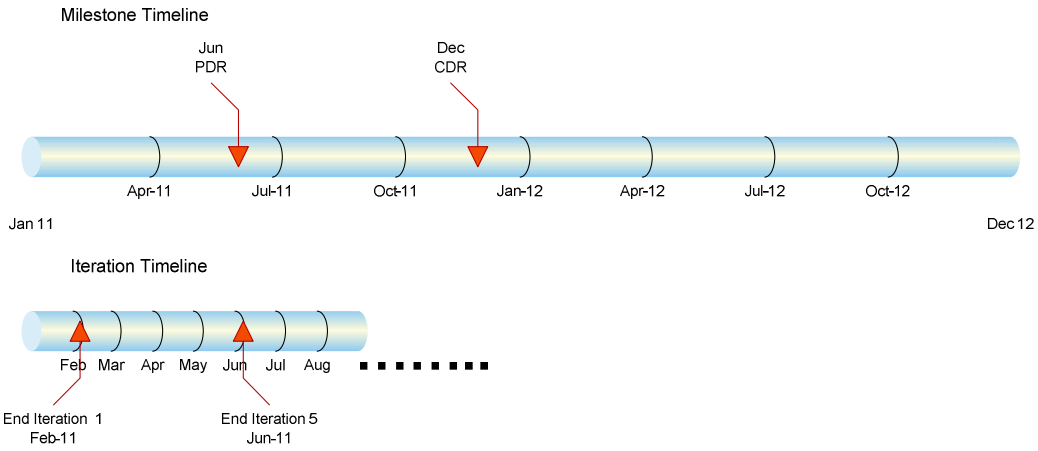


Figure 8: Relationship of Sprint to Release to PDR of Record

4.5.3 An Alternate Perspective on Milestones

Another way to look at milestones such as PDR is to see how they align to releases and the associated iterations. This discussion provides an alternate look at how the milestones could be achieved.

Each iteration within a release follows a standard process. However, before the iteration starts the development team and stakeholders determine which of the capabilities (functional or non-functional) will be accomplished within the upcoming iteration. These capabilities have already

[27] It should be noted that this discussion is redefining the terms PDR and CDR for use with Agile. This could lead to confusion and contribute to people "acting Agile" as opposed to "being Agile." Perhaps the milestones should be renamed. This renaming is left to future work.

been defined earlier in the program and listed in a backlog by priority and/or risk-reduction value. Once a capability is assigned to a release, it is decomposed into features as shown in Figure 9.

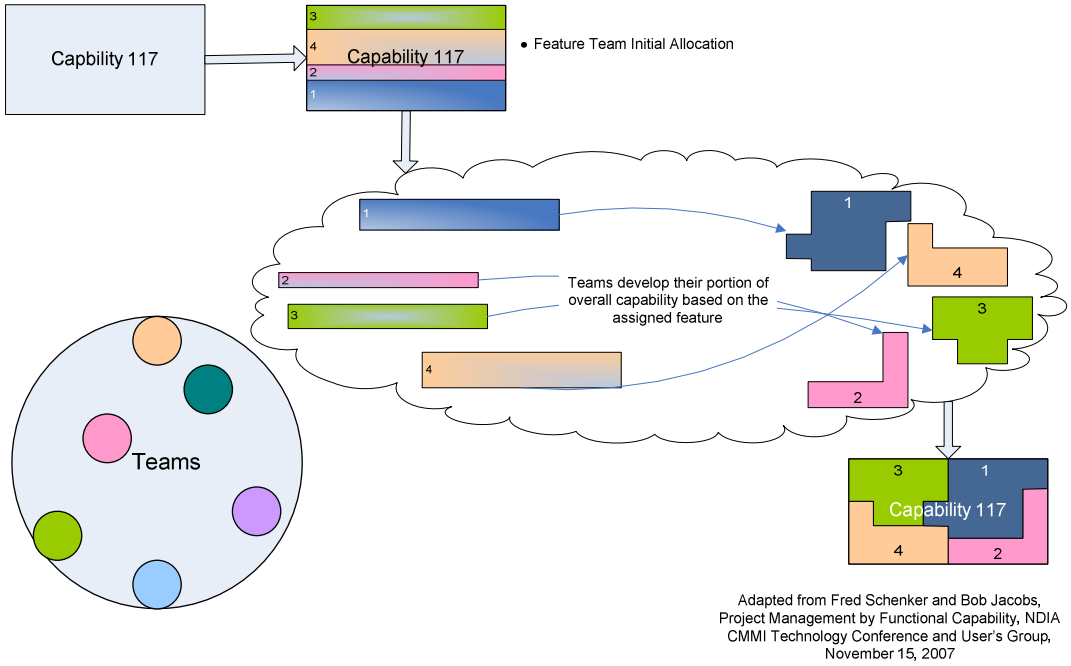


Figure 9: Capabilities Decomposed [Schenker 2007]

In this illustration, each feature of the capability is assigned to a different team for development during the iteration. The feature would be described using stories. The teams would be considered feature teams with one team singled out to be the capability guardian. Each feature team would then develop its portion of the overall capability using the process shown in Figure 10.

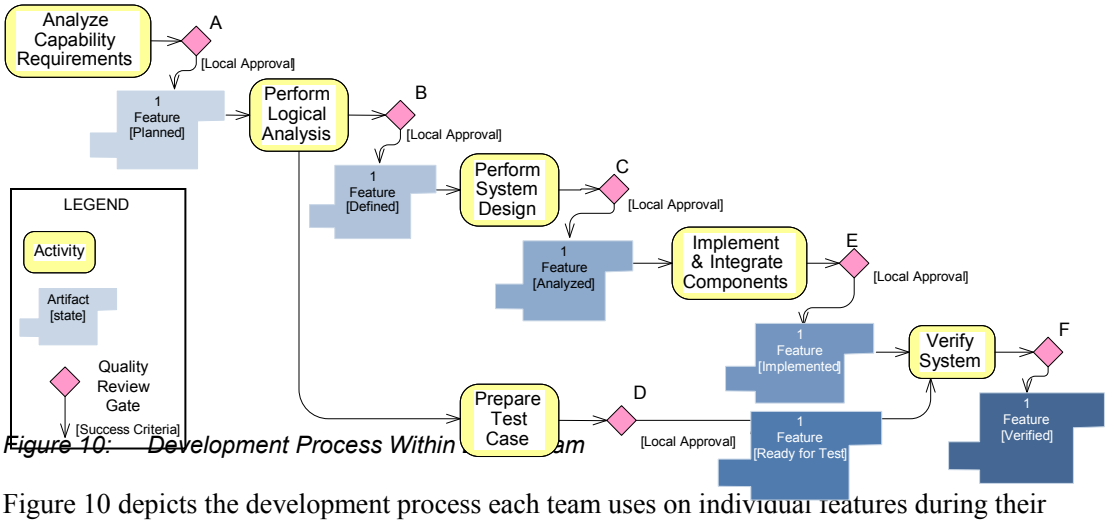


Figure 10: Development Process Within [illegible]am

Figure 10 depicts the development process each team uses on individual features during their iteration. For simplicity of comparison, the diagram looks very much like a waterfall process. However, all disciplines are represented on the team and they all work on the process together, with each person contributing in their discipline (analysis, coding, testing, etc.). Typically, the PMO is invited to participate at each Quality Review (QR) gate. This insight provides more touch

points for the acquisition agent, typically the government, to be involved in the progress of the program. The information gained by attending these QR gates can be used to augment the formal PDRs, CDRs, etc. The diamond shapes lettered A-F are QR gates which must be passed before the artifacts from the iteration can be considered done. Each QR can be considered "mini" PDRs, CDRs, etc. As you can see, analysis, design, implementation and integration, and test are all accomplished by the team within the time box allowed for the iteration. This type of PDR iteration would be expected to go beyond two weeks. Two weeks seemed to be a norm that is common in the interviewed programs that are not using formal technical reviews.

Now that we understand how the team functions, we can return to creating the capability shown in Figure 9. Capability 117 is decomposed into clear features that can be completed by a team within one iteration. Each feature is assigned to a team (1-4). This work is described in the work plan or scoping statement for the individual iteration. The teams get their individual features and proceed to develop their portion of the capability. Once all features are done, the capability is also completed.

This discussion of capabilities and features showing quality gates is another way to think about doing progressive milestones. It shows one possible detailed approach to solving the issue of technical milestones while using Agile methods.

It should be noted that this discussion does not address the difference between systems engineering/design and iterative development of the capabilities. These could be two very different items depending on the type of system. The systems engineering and design effort essentially feeds the product backlog. Iterative development addresses the creation of the actual products. The rhythm of each does not need to be the same. Further analysis and discussion of this area of concern are left to future work.

5 Estimating in Agile Acquisition

5.1 Introduction

Estimation activities occur throughout the DoD acquisition life cycle. Estimates are used by all DoD programs in a variety of ways, and they are generated and processed in a variety of ways. Books have been written, inside and outside the DoD, on cost estimation for large, complex, software-intensive programs [Stutzke 2005]. We cannot deal with all the many connections between DoD estimation practices and Agile estimation in this report. We address the issues that we have most frequently seen and discussed, in interviews and through reviewers. Also, in all of the generalizations we make below, it should be understood that the needs and constraints of a particular program could result in estimates being treated in similar or quite different ways than what we describe in this section.

Some general estimation activities that government program offices support on many (though certainly not all) programs include:[28]

- **Producing a Program Life-Cycle Cost Estimate.** Prior to Milestones A and B, the Acquisition Program Office (the government) must develop a program life cycle cost estimate (PLCCE). The PLCCE is presented to the program's Milestone Decision Authority (MDA) at each milestone. The PLCCE must look forward from the current program state to the end of the system's life, and assess the cost of the product or system over its entire life.
- **Program Monitoring.** During source selection, the Acquisition Program Office may want to gain insight into the manner by which the bidding contractors have prepared their estimates. During contract execution, the Acquisition Program Office is constantly reviewing the performance of the contractor with respect to the contractor's estimate. This is typically done by reviewing the contractor's earned value management (EVM) data, although there will be further opportunities to review the contractor's estimation process each time an engineering change is processed.[29]
- **Transition to Sustainment.** After the system is fielded and in sustainment, there is generally a two-year cycle of maintenance, technology refresh, and upgrade.[30] These system enhancements are estimated and budgeted by the relevant program office, and by the contractor, who may or may not be the same organization that originally developed the system.

To understand the varied uses of estimates by the program office staff throughout the acquisition life cycle, and how these uses may relate to the use of an Agile development process by a contractor, the reader must understand estimation practices in general, and Agile estimation practices in particular. The next section of this report (5.2) will begin to provide that insight and

[28] The specifics for what is and is not required are defined in the FAR and DFAR for EV programs.

[29] How this occurs and the various ways programs can implement this estimation process is covered in the earned value management system description (EVMSD) and its work instruction.

[30] The timing and nature of sustainment activities is, of course, ultimately dependent on the particular program contracting and technical characteristics. However, we have observed this pattern in many programs.

we will show how a government program office (including the program manager, the staff, the contracting officer's technical representative (COTR), and the procurement staff) could take actions with their cost estimating practices that would enable an Agile acquisition of a new system or sustainment of an existing system in the DoD.

5.2 Estimating to Support Request for Proposal (RFP) Preparation

The following discussion assumes that the government program office is acquiring software products (i.e., buying a system through a cost or incentive contract as covered by DFAR 234.201). We are not discussing acquiring software development capacity (i.e., software development expertise of a certain capability over a period of time), which is an alternate way we have seen government software needs being met. This model is more common in sustainment and operations and maintenance (O&M) programs. It generally consists of determining how many resources you can afford and how much capability those resources will allow you to build. This model is what some successful Agile programs have used, but it is not available to all programs.

During the RFP preparation phase of a new system acquisition, the government program office will make the decisions that are pivotal to enabling or disabling an Agile development contractor to bid and meet the program's needs. It is during this phase that the government program office will prepare its PLCCE, which will be based on the government's work breakdown structure (WBS). The prohibition during this timeframe against engaging with the development team when this is a competitive contract is a significant barrier to establishing the trust that is key to Agile project success; however, the considerations below could help to mitigate this issue.

If the program office wants to allow a developer using Agile methods to effectively compete, there are considerations that relate to both the acquisition strategy and its follow-on activities, as well as considerations related to execution of the Agile methods within the boundaries of the Program Management Baseline (PMB). From the acquisition perspective, the government program office must address how typical Agile methods artifacts fit into the traditionally specified artifacts of an acquisition, for example:

- The acquisition strategy should describe how the program office would interact with its contractor in order to provide the subject matter expertise needed on a continuous basis throughout the iterations of an Agile development.
- To ensure that the Agile acquisition strategy is enacted, the statement of work (SOW) or program work statement (PWS) must include language that allows the program office to provide subject matter experts with the ability to participate in the development of the software. This may be complicated by the hierarchical structure of contracts in a large system acquisition. The program life cycle cost estimate and budget must include funding for these subject matter experts throughout the development of the system. Because the SMEs usually come from government operational units, agreements must be crafted (e.g., memorandum of agreement [MOA], memorandum of understanding [MOU]) that make clear the expectations of participation of different stakeholders.
- The government program office must have a notional plan for what to do with the interim product releases that come from an Agile development process. Specifying these in the SOW is one way to emphasize the importance of working software being available in short iterations. There should be an evaluation environment established along with a feedback

mechanism in place that permits the end-user community to try out these interim releases in a safe, secure environment, while waiting for required acceptance and certification testing activities to take place.[31]

Generally speaking, the most visible element of a software product estimate in DoD programs is the estimate of product size.[32] Even though modern software development tools and techniques reduce dependence on handcrafted source code, size is still frequently expressed in source lines of code (SLOC) or in function points. In Agile development environments, the development team may use "story points" as an alternative to either of these. Story points can be problematic in acquisition settings accustomed to SLOC or function points because they are explicitly a relative measure of size, not an absolute measure. Therefore, when story points are used outside of the team that generated them, it is necessary, though not trivial, to make some translation between story points and, typically, function points. Some of the programs we interviewed acknowledged they made the translation from story points to product size to provide cost estimates to those outside the development team. We saw proprietary tools that address this translation, and the commercial vendors for estimating tools are starting to address this new market need. In acquisition settings where trust has already been established between the contractor and the acquisition program office, this dependence on an absolute, versus relative, measure may be reduced.

Most parametric cost-estimation models base their outputs on software size, so errors in the size estimate will propagate into the estimate of effort and schedule. According to the GAO *Best Practices Guide for Estimation*, the keys to producing a defensible software cost estimate are (1) to have a reliable method for estimating the size of the product and (2) to employ a method for transforming the size estimate into an estimate of cost and schedule demonstrated to be accurate on similar projects [GAO 2009].

One popular parametric cost-estimation tool is the constructive cost model (COCOMO). According to Boehm, "COCOMO is an algorithmic-based parametric software cost-estimation model for estimating a software project as an 'effort equation,' which applies a value to tasks based on the scope of the project (ranging from a small, familiar system to a complex system that is new to the organization). COCOMO II is the successor of COCOMO 81, incorporating more contemporary software development processes such as code reuse, use of off-the-shelf software components, and updated project databases" [Boehm 1981].

At the heart of the COCOMO II model are the cost parameters themselves. These parameters include five scale factors and seventeen effort multipliers . Scale factors represent areas where economies of scale may apply. Effort multipliers represent the established cost drivers for software system development. They are used to adjust the nominal software development effort to reflect the reality of the current product being developed.

---

[31] Note that certification and accreditation (C&A) issues within the DoD acquisition life cycle are currently being addressed on multiple policy and implementation fronts, all with the goal of reducing the time, usually spent at the end of a program, to get the software system certified and then accredited by the appropriate governance body. We are not dealing with the specific requirements of the DIACAP process in this report.

[32] Software size may not be the most reliable predictor of software effort and cost (see Capers Jones, for example, who cites programmer skill as a better predictor of software outcome than size, among other attributes).

It would be reasonable to assert that an Agile development process would have an impact on some of these parameters. For example, the COCOMO II model includes an effort multiplier for domain knowledge, or applications experience. The cost estimating multiplier based on the domain knowledge and capability of the software developer staff is called "application experience" (APEX). The rating for this cost driver is dependent on the level of applications experience of the project team developing the software system or subsystem. The ratings are defined in terms of the project team's equivalent level of experience with this type of application.

In an Agile development environment, there would be subject matter experts (users) participating with the system developers. The participation of users in the development process should improve the domain knowledge of the development team. The magnitude of the improvement can be assessed by changing the assignment of this effort multiplier, and observing the impact on the estimate.

A selected list of COCOMO II scale factors and effort multipliers is provided in Appendix D. Factors listed there that we would expect to be impacted by the use of an Agile development process include

- the development flexibility factor
- the architecture/risk resolution factor
- the team cohesion factor
- the analyst capability effort multiplier
- the programmer capability effort multiplier
- the application experience effort multiplier

Appendix D also contains information about Agile COCOMO, a 2004 prototype product that reflects some Agile estimation principles while relating back to concepts familiar to COCOMO users [Agile COCOMO 2011].

Among the many software estimation tools generally available (including Price-S, Software Lifecycle Management-Estimate [SLIM], and others) is the Software Evaluation and Estimation of Resources (SEER) model. It is one of those that actively updates its products to accommodate Agile estimation.

> *SEER for Software (SEER-Software Estimation Model [SEM]) is an algorithmic project management software application designed specifically to estimate, plan, and monitor the effort and resources required for any type of software development and/or maintenance project. SEER, which comes from the noun referring to one having the ability to foresee the future, relies on parametric algorithms, knowledge bases, simulation-based probability, and historical precedents to allow project managers, engineers, and cost analysts to accurately estimate a project's cost schedule, risk and effort before the project is started [SEER-SEM 2011].*

For Agile projects, SEER uses three kinds of estimates. These are planning, forecast, and working. The planning estimate is still used to determine how big the project will be and is usually based on analogies of previous projects of similar size. The forecast estimate is accomplished after you have built your backlog. Several things can be defined at this time, such as incremental delivery, release cycle, the length of the iteration, exit criteria for a deliverable, and negotiation for scope change requests. (Baseline change requests accomplish this in the DoD

acquisition cycle.) Finally, working estimates are done for all iterations after the first iteration is complete. This allows assessment of the team and customer as well as an understanding of the individual team velocities.

SEER, like COCOMO, uses a variety of parameters for their model. These parameters include

- requirements formality
- requirements volatility
- personnel capabilities – analyst and programmers
- familiarity with the process
- process maturity
- staffing complexity
- development system volatility
- automated tools usage
- testing level
- quality assurance participation
- infrastructure and tooling costs

Before the build, your estimate considers these parameters in relationship to your team. We recommend that you revisit your forecast estimate as your team changes.[33]

5.3 Source Selection

In the source selection phase of an acquisition, the program office will have to evaluate estimates that are prepared by the bidding contractors. In many cases, the program office will seek to understand the contractor's process for producing the estimate. It is very important for the program office to establish a high degree of confidence in the bidding organization's estimation process.

The following discussion focuses on a notional Agile estimation process from the development estimator's viewpoint. We have synthesized this description from our various interviews and include some clarification information from the Agile literature to help readers new to Agile methods relate the Agile approach to knowledge they already have from using traditional estimation practices. We hope that this approach will enable government program office personnel new to Agile approaches to gain insight into why estimates for an Agile project may look different from traditional ones.

5.3.1 Estimating from the Development Estimator's Viewpoint

We focus this section on the development estimator's viewpoint, which could either be for a government organization (such as an Air Logistics Center of the U.S. Air Force), or a commercial development contractor. In either case, the viewpoint is based on knowledge of the team that will be producing the software, knowledge of the tools and development environment that are available, as well as knowledge of the practices that are intended to be used. We also distinguish

[33] DeWitt, D. *Demystifying Agile Project Cost and Schedule Estimates*. Webinar. Galorath Incorporated, 2010.

between new software development estimation and sustainment-focused estimation, since the basis of each is different.

In the case of new software development (some new feature being implemented in software for the first time or a significant upgrade to existing software being treated as its own project), some initial work will need to be estimated for creating an overall architecture that will be the basis for the rest of the project. That architecture will determine some of the requirements prioritization, though not all of it. Overall system design is outside the normal scope of software development estimation, so some ideas of architecture and its implications may be established prior to estimation. In any case, working the initial aspects of the architecture and platform infrastructure is usually estimated separately from the actual requirements implementation, and in Scrum, the most commonly used Agile project management method, this is usually called "Sprint 0" [Ozkaya 2011].

Often, especially in the DoD programs we interviewed, the early iterations and stories are more about building the infrastructure needed to ensure a stable architecture than about delivering end-user functionality. If using the RUP as a framework for an Agile project, this kind of work is done during the Inception and Elaboration phases. In cases where this was necessary, some of the programs we talked to mixed infrastructure building with end-user functionality, so that end users received working software at least every other release. Others coupled architectural infrastructure elements to end-user functionality so they could deliver on just a piece of the architecture. In either case, the emphasis was on ensuring that end users saw progress quickly and frequently. Once a general pattern of releases was generated, a more detailed estimation of future releases and sprints occurred.

After user stories are generally prioritized, they become a product backlog. From the product backlog, releases are constructed that deliver evolving capability to the end users. Each release has a nominal set of user stories (based on team velocity, vital factors, and initially estimated story points). Up to this point, the estimation has been coarse-grained, since it is known that user priorities will change over the course of a project, especially one that is longer than one year.

From the product backlog reflected in the first release, the user stories for the next iteration within that release are selected (a process sometimes called "grooming" the backlog) and the team working on each story does more fine-grained estimates of the appropriate story points for that release. Based on the team velocity, an estimate of feasibility is made as to whether the proposed set of user stories can be built within the iteration timeframe.

In most Agile methods, the end users and other project stakeholders are present in the iteration-planning meeting where these issues are discussed, so that re-prioritization can occur if necessary. These meetings also enable an essential element of Agile methods: the development team and the end users decide on the character and timing of user/developer working sessions. This kind of joint decision making is one of the things that the programs we interviewed emphasized as being essential to their progress.

This rhythm of each iteration being estimated at a fine-grained level while releases and the overall project are estimated much more coarsely actually reflects the common practice in DoD cost accounting discussed earlier: rolling wave planning [Department of Defense 2011]. More detail on this part of the process is found in Section 5.4, Contract Execution and Monitoring. One

important aspect of rolling wave planning related to estimation is that the period of performance covered by the rolling wave must align with the iterations in the life cycle so that planning does not occur, for example, for only half of an iteration.

In the case of sustainment or enhanced legacy software, if the architecture is stable, then prioritizing the known requirements is a first step in estimating. In Agile methods, these are gathered as user stories—descriptions of discrete functionality known to be needed by a particular user segment that is part of the project's audience, and other stories that address infrastructure and quality attributes that are pervasive to the product (e.g., security or usability). Although user stories are generally constructed to be discrete and separable (one of the things that permits reasonable prioritization), they can often be bundled into a related feature set to be delivered, called an epic. It is not unusual for a release to be defined by the completion of one or more epics. Where the user stories come from (government operators or contractor subject matter experts), is highly dependent on the contracting vehicle and agreements that are in place for the effort.

5.3.2 Evaluating Estimates from the Acquirer's (Source Selection Team) Viewpoint

In this section, we change focus from what an Agile estimation experience looks like from the development estimator's viewpoint to what it looks like from the estimate evaluator's viewpoint, usually the source selection team or other members of the government program office.

The biggest difference between evaluating an estimate for an Agile project and a traditional project is that the Agile project admits up front that not all requirements can be known early in the project and so the overall estimate will be amended as more knowledge is gained. Where a contract vehicle has been constructed that allows these amendments to occur without having to process baseline change requests, (such as a time and materials contract type) the overall process has been easier for both acquisition personnel and the development contractor.

Estimates for near-term activities—usually through a single release—can be made more accurately than the typical traditional project because the period for estimation is usually less than four months. The four months is the equivalent of eight two-week iterations, an approach consistently used for several years on one of the programs we interviewed. The team's capabilities, in terms of how quickly they can typically address a story point's worth of work, are well understood after the first couple of iterations. This accuracy is dependent on knowledge of the team's progress characteristics.

In discussing government evaluation of development contractor estimates in Agile projects, we gleaned that the questions in the following list were considered useful by a variety of our interviewees. Not all programs used all questions; this list is a union, not an intersection, of the questions. Which questions apply in a particular acquisition situation also depends on the acquisition strategy decided upon and the contract vehicle used. Not all of these questions can be used for all contract types. Some of them (e.g., the first one) assume that the developer already understands and has worked in Agile projects, while others do not make that assumption. The questions different programs ask about an Agile project's initial development estimates include:

- If the project involves new software development, did the development team leave separate time for constructing the product's architecture and infrastructure needed (e.g., the continuous integration and test environment) to operate the project?

- Do the initial user stories adequately reflect the *known* end-user project priorities, tempered by any programmatic constraints that have been shared with the estimator? (Clearly they will not reflect those that are unknown at the time of estimation.)
- Has the team performing the work used Agile methods before as a team? If so, do they have evidence of their velocity on similar projects? (If they have not worked on similar Agile projects before, calculating velocity from their individual performances would be inappropriate and misleading.)
- If this team has not worked together before, how did they derive their velocity?
- If this team has not used Agile methods before, have they left some slack to account for a learning curve?
- Does the estimate include frequent opportunities for user feedback (e.g., pre-release demonstrations of working software at the end of each iteration)?
- Does the estimate include time for side-by-side working sessions with end users during iterations?
- Does the estimate characterize the "vital factors," such as distributed team, new project domain, complex operational processing, and other factors, and how they affect the estimate? [Bhalareo 2009]

These factors are somewhat different from the COCOMO II factors that estimate evaluators may be familiar with, but they bear some relationship and may be able to be resolved (though we have not run into this in interviews with Agile DoD programs so far).

Also important for evaluators to remember is that you will be receiving new estimates for each release or iteration depending on the project norms and contract vehicle. This gives you the opportunity and an obligation, as an acquirer, to reevaluate requirements priorities (via the product backlog) based on user feedback for the most recent releases. Depending on the project, releases for informal early adopter use, usually in sandbox environments, may happen as often as every two months.

Note that from an acquisition life cycle viewpoint, the releases we are talking about here are generally development releases, so the user community intended to receive them must be carefully selected. Our interviewees usually had subject matter experts on the development team who were knowledgeable about certification requirements for their software and who participated in identifying the appropriate user audience for different classes of release. Certification requirements are a type of constraint that can prevent early release of software, even on a development basis. The effect of these interim releases on estimation varies. Depending on the constraints of the contract, interim releases may be accomplished easily and often, or they may be almost as much work as a fully deployed release to fielding.

5.4 Contract Execution and Monitoring

When working with any development contractor, it is important to understand how the work is being planned and executed, so that the program office can understand how to interpret the progress data provided by the contractor. When working with an Agile contractor, it is especially important for the acquirer to understand the methods and techniques employed by their contractor,

as it is likely that the techniques used by the contractor will be new to the program office. This understanding provides the foundation for any discussions between the acquirer and the contractor. In addition, this understanding does not replace specific constraints or directions levied by the FAR/DFAR, but it allows the acquirers to understand the implications of their contract vehicle in the Agile environment. This section of the report provides insight into the techniques often used to plan Agile projects, mostly from the development team viewpoint, and into the techniques used to monitor and control Agile projects, mostly from the acquirer's viewpoint.

5.4.1 Story Point Estimation

In Agile projects, user stories and technical stories are typically estimated in story points. Story points are commonly used in several Agile methods for estimation at both the release and iteration levels. They do not use lines of code as their base unit of measure. Tasks, on the other hand, are generally estimated in hours and are used only for detailed iteration-level planning. Tasks are the activities that developers determine will be necessary to successfully complete the story. If you are evaluating developer estimates, being able to understand the source of the developer estimates can improve your ability to interpret them.

The following is a common definition of story points:

> *Story points are a unit of measure for expressing the overall size of a user story, feature, or other piece of work. ... The number of story points associated with a story represents the overall size of the story. There is no set formula for defining the size of a story. Rather a story-point estimate is an amalgamation of the amount of effort involved in developing the feature, the complexity of developing it, the risk inherent in it and so on. [Cohn 2006]*

One of our reviewers commented, "This [concept] is really important as it can thwart meaningful comparison and tracking of trends. It certainly can undermine the ability to do cross-team comparisons."

It is important to note that story point estimates are both *relative* and *local*. They are *relative* in that estimates are typically derived by comparing the size of one story to another or by assigning a point value to one or more reference stories, which are then used to calibrate the sizes of newly created stories. Story point estimates are *local* in that different teams may arrive at different sizing conventions. A story that is assigned five points in team A may, for example, be assigned three points in team B. One implication of this is that, for most DoD programs, at some point estimates must be converted from relative to absolute estimates, especially for programs using EVM (earned value management).

Story point estimation is typically conducted as a team-based activity and is guided by defined techniques. Two popular team-based estimation techniques are Planning Poker and the Team Estimation Game [Larman 2004]. In Planning Poker, stories may be assigned point values of 1, 2, 3, 5, 8, 13, 20, 40 or 100 (an adaptation of a Fibonacci series). Other Agile estimation techniques use similar scales. The spacing between the point values is designed to reflect both the principle that "we are best at estimating things that fall within one order of magnitude" and "greater uncertainty is associated with estimates for larger units of work" [Cohn 2006]. Stories planned for incorporation within an upcoming iteration will typically be assigned point values at the lower range of the estimation scale while stories coming later will be assigned point values at the higher

ranges, especially if they reflect a lack of knowledge until some of the earlier stories are executed. In addition to using story points to estimate effort, at least one Agile author (Larman) recommends that stakeholders independently estimate story point value at the same time developers are estimating effort, allowing for an explicit prioritization of effort for value [Larman 2004].

While story points are the most widely advocated metric for story and feature size estimation, some within the Agile community also advocate for the use of "ideal days" for this purpose. Similar to story points, ideal days are intended to be used as a sizing estimation metric, expressed as the number of days a story or feature would take to develop, assuming

> *The story being estimated is the only thing you'll work on.*
> *Everything you need will be on hand when you start.*
> *There will be no interruptions [Cohn 2006].*

It is important to note that when estimating in ideal days, as with story points, the estimate is intended to include the aggregated work required from all team members for all tasks required to successfully complete development (e.g., elaborate story details, write unit tests, design, code, build, execute acceptance tests, write required user documentation). In addition, as with story points, estimates in ideal days are a local metric. Once again, a story that is assigned five ideal days in team A may, for example, be assigned three ideal days in team B.

5.4.2 Velocity

As discussed above, both story points and ideal days are relative, local sizing metrics, rather than objective projections of effort and duration. Therefore, story points cannot be used directly for absolute estimation purposes. Rather, within Agile practices, story points provide input to the calculation of other measures like team "velocity," which is in turn used to derive estimates for releases and iterations. As stated by Mike Cohn, "…a key tenet of agile estimating and planning, is that we estimate size and derive duration"[34] [Cohn 2006]. However, unlike traditional projects, Agile projects estimate *relative* size, rather than *absolute* size. Current expressions of estimates within DoD programs use absolute estimates of size and duration, requiring translation from Agile estimation approaches, as we have mentioned previously.

"Velocity is a measure of a team's rate of progress. It is calculated by summing the number of story points assigned to each user story that the team completed during the iteration. If the team completes three stories, each estimated at five story points, their velocity is fifteen. If the team completes two five-point stories, their velocity is ten" [DeWitt 2011].

Mike Cohn describes three potential options for estimating the velocity of a given team.

> *Use historical values.*
> *Run an iteration.*
> *Make a forecast [Cohn 2006].*

[34] It is worth noting that deriving effort from size is a common way of estimating software projects in traditional methods as well.

Each of these activities takes place within a particular context and is based on specific assumptions, such as team skill and history with the domain. Velocity is sufficiently tied to the specific team's characteristics that cross-team velocity comparison can be misleading.

Running an iteration is a common approach to learning about a team's velocity in the commercial space. Depending on how the contract is constructed, this may or may not be an option within a DoD contract.

5.4.3 Agile Release Planning

The Iron Triangle of cost, time, and scope is fundamental to traditional release planning. An Agile perspective on this triumvirate is expressed by Dan Rawsthorne in the following equation:

```
Time x Capacity = Scope
```

where
`Time` = # of iterations * iteration length

`Capacity` = average velocity per iteration[35]

`Scope` = total # of story points that can be completed in the release

Using the above equation, a team with an iteration length of two weeks and an average velocity of 30 could complete 300 story points in approximately 10 iterations or 20 weeks. While seemingly straightforward, this equation must be understood within the context of the Agile approach to project scoping [Rawsthorne 2010].

Whether developing within a traditional or an Agile methods environment, the first step in planning any release is to establish the high-level goals and purpose of the release. That purpose may include delivering capabilities to a particular group of stakeholders, increasing customer satisfaction, or gaining market share. Once the goals and mission are established, the focus then turns to scoping the release contents.

On a traditional project, establishing scope for a release begins with the creation of a detailed requirements specification. Within an Agile project, establishing scope for a release begins by examining the product backlog. The product backlog is the name commonly used for the repository of stories associated with a given product or project. If the project is a completely new start, a new product backlog will have to be defined by delineating the user and technical stories relevant to that project. Elements within the product backlog may also include features, capabilities, and defects, as well as stories. It is a recommended (although not universally adopted) practice to attach story point estimates to all items within the product backlog [Cohn 2008].

For an Agile project, scope is often expressed in stories. The major activities involved in scoping and planning an Agile project include:

---

[35] Note that there is an implicit assumption that capacity does not vary. And while it is true that it varies less if the team is stable in terms of membership and type of tasks performed, capacity is quite likely to change when the domain, programming environment, or other significant environmental factors change, even if team composition does not.

- selecting features and stories from the product backlog for incorporation into the release (some features will already be expressed as stories, depending on their prior history)
- decomposing features into stories that reflect each feature's intended business value (in doing this, it may become clear that some stories must take precedence over others; stories that are not suitable for the current release get returned to the product backlog)
- decomposing larger stories into smaller stories that can be completed within a single iteration (again, after this step, some stories may be returned to the product backlog)
- assigning a story point value to each "iteration-sized" story (although story point values may have already been assigned within the product backlog these estimates will typically be re-examined and validated during release planning)
- prioritizing stories and assigning them to specific iterations

Whether all of these activities are done upfront at the start of the release or whether some are conducted on a per-iteration basis will depend upon the team, the project, and the associated program expectations and constraints. "Some teams in some environments prefer to create a release plan that shows what they expect to develop during each iteration. Other teams prefer simply to determine what they think will be developed during the overall release, leaving the specifics of each iteration for later. This is something for the team to discuss and decide during release planning" [Schenker 2007].

It is important to note that even after stories have been broken down and estimated, they generally are not specified to the degree that would be found in a traditional requirements document. This does not necessarily imply increased risk, because successful Agile teams rely on ongoing dialog with users, user proxies, and subject matter experts throughout the course of the release to gain insights needed to satisfy the users, usually better insight than could be gained from typical requirements-specification documents. If the user interaction that makes this dialog possible is missing, the benefits associated with user stories as an anchor for the requirements will be lost.

Prioritization of stories across iterations is another important aspect of release planning. Cohn identifies the following four factors as critical considerations during prioritization [Cohn 2006]:

- value of the story
- cost of developing the story
- knowledge generation, including
  - knowledge about requirements, the domain, and user needs
  - knowledge about the underlying product technology, architecture, and design
- risk, including
  - technology risk
  - business risk
  - schedule risk
  - cost risk
  - functionality risk

While a certain amount of prioritization will take place during initial release planning, on Agile development projects, prioritization is ongoing and stories are often reprioritized at the end of

each iteration. Successful adoption and execution of this dynamic approach to prioritization once again requires a close relationship and ongoing dialog with program stakeholders.

The role of the product owner (usually played by the acquisition program manager) in release planning cannot be underestimated. Product owners resolve the concerns of multiple stakeholders with conflicting priorities. They maintain the integrity of the product and ensure that it actually delivers the promised value to end users. In release planning, they often know the most about the programmatic constraints that must be met prior to release to the end user, and they are the people who will have to seek waivers or other relief if needed from processes that disable a project's intended Agile practices from working.

5.4.4 Agile Release Tracking

As discussed previously, Agile release planning relies upon the following three factors:

1. the team's estimate of their projected average velocity for the release
2. the set of stories selected for inclusion in the release (i.e., the stakeholders' estimate of the desired release contents)
3. the sum of the story point values for the stakeholder-selected stories

Referring back to the previous example, if the team estimates its velocity at 30 story points per two-week iteration and estimates the sum of the sizes of the stakeholder-selected stories at 300 points, then the release should take 10 iterations or 20 weeks.

Agile release tracking focuses on these same three factors and examines how closely the initial estimates are tracking to actual results. Agile release tracking, therefore, asks the following three questions [Rawsthorne 2010]:

1. Is the team's velocity tracking to its initial estimates (i.e., how many story points have been completed to date and how does this compare to the plan)?
2. Have the stakeholders added stories or removed stories from the release? If so, have these changes increased or decreased the sum of the story points for the release?
3. Has the team changed its point value estimates for any stories?

One of the most commonly used charts for tracking progress on Agile releases is the release burndown chart. On the release burndown chart, the x-axis is expressed in iterations, while the y-axis is expressed in story points remaining to be completed. Under ideal conditions, a release burndown chart for our sample project, with 300 story points and a velocity of 30, would appear as follows:

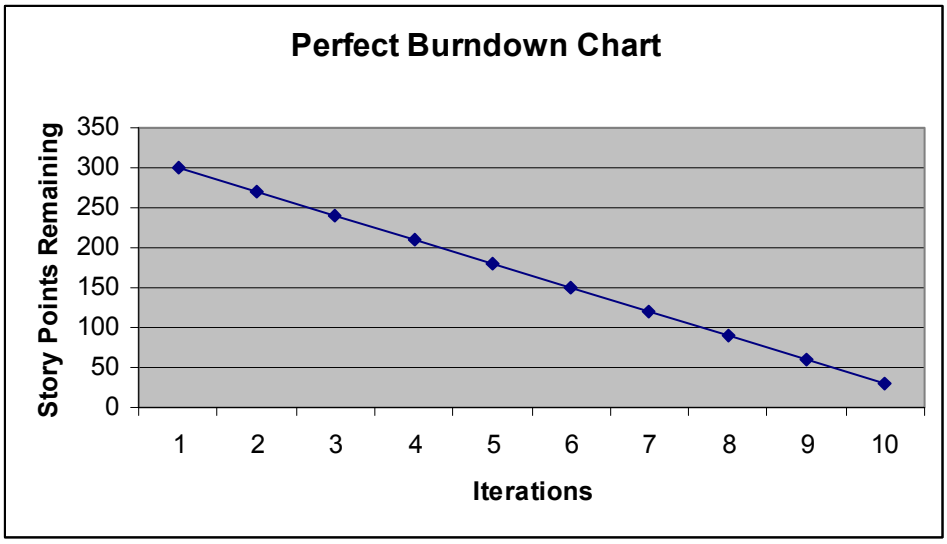


Figure 11: Perfect Burndown Chart

In reality of course, no project will ever execute in precise conformance to initial estimates. Therefore, it is more likely that by iteration 5, the release burndown chart for the project will look something like this:

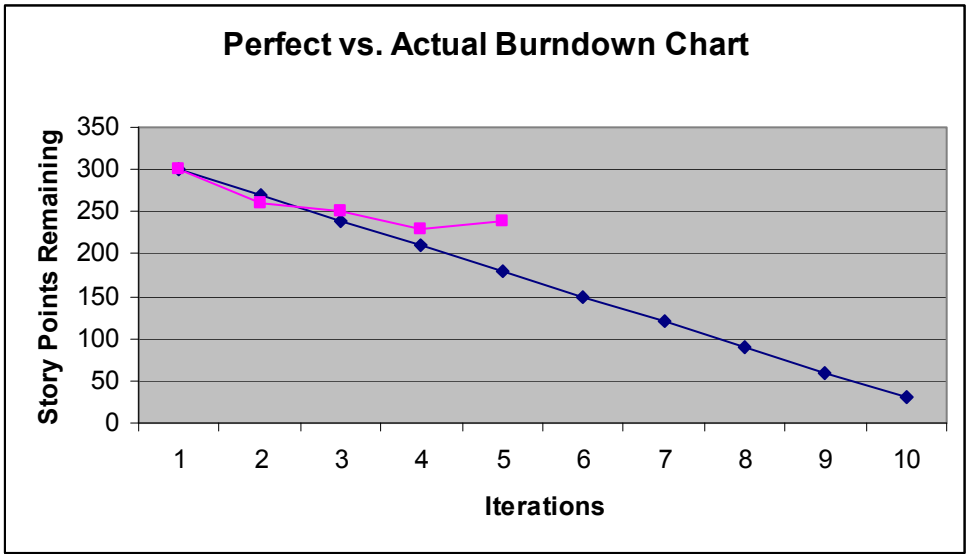


Figure 12: Perfect vs. Actual Burndown Chart

This chart for our sample project clearly shows that by the end of iteration 5, we have more stories remaining to complete than we had originally planned. However, the chart itself does not give an indication of why this is the case. Any of the three factors discussed above could be behind the discrepancy:

1. The team's velocity could be less than initially anticipated.
2. The project stakeholder may have added stories to the release.

3. The point estimates for certain stories may have increased as the team gained further knowledge of the technology and the domain.

The chart therefore provides an early indicator of potential future issues, but only discussions with the development team will reveal the reason for the discrepancy and what actions, if any, need to be undertaken. Other visualizations can increase the insight into reasons behind a particular burn down phenomenon, which are discussed in detail in Cohn's *Agile Estimation and Planning* [Cohn 2006].

As with any other progress tracking method, using user stories to generate velocity measures can lead to some anomalous results. For example, if the user stories are more than an order of magnitude sizing difference during an iteration, velocity could appear lower than is warranted. This sizing difference could also result in one story taking an inordinately long time to complete, possibly even resulting in a velocity of 0. A development team should develop its own norms in terms of the relationship of the number of stories to the number of team members to iteration duration.

5.4.5 Verification & Validation

The amount of total effort estimated for verification and validation (V&V) activities may not be that different in amount when comparing Agile and traditional projects. However, the timing of verification and validation activities is expected to be different and that should be reflected in the way the CDRLs are handled for the contract. Most V&V estimates for traditional projects show a bimodal distribution of effort—high at the beginning when test plans and environments are being determined, low in the middle during design and implementation, high at the end during execution of verification and validation activities. However, most Agile methods involve some type of continuous integration and testing, and some methods, like test-driven development, actually demand that test cases be written before designs are implemented in code. Thus, the profile of V&V activities may well look more like a steady level of effort than a bi-modal distribution of effort.

Some of the projects we interviewed included a separate iteration for acceptance testing, including, if appropriate, some of the information assurance (IA) testing that is required for certification. (Note that although information assurance is a specialty engineering discipline that is involved throughout the project, there is a certain amount of testing for IA that usually occurs as part of the overall V&V effort.) Others considered acceptance, certification, and other operational testing to be outside of their Agile life cycle and their delivery to those testing environments was the completion of their Agile project life cycle, other than rework that was required to address defects found in the acceptance test cycle. The decision about how to treat V&V is a contract-specific issue and, as can be seen from some of the variants expressed here, the effects on estimation will be determined by which process and method selections are made.

5.4.6 Agile EVM (Earned Value Management)

Earned value is one of the primary tools that the Department of Defense uses to measure contractor performance. For programs valued at more than $20 million, an earned value management system (EVMS) is required to be used, and for programs more than $50 million, a *validated* EVMS must be used. "EVM techniques, however, assume complete planning of a

project to discrete work package levels, then assigning cost and duration to these packages" [Sulaiman 2006].

It should also be noted that the application of traditional EVM methods within the DoD acquisition process is currently being reexamined.

> *EVMS has experienced a number of issues, notably with contractor implementation and data quality. However, for the Panel's purposes, the most significant limitations are that EVMS only measures the performance of a contractor, not of the organization which is managing the acquisition. Furthermore, EVMS would generate no negative information about a contractor performing on cost, on schedule, and meeting all contract requirements even if (or perhaps especially if) the contract in question had a wildly inflated price or a schedule or set of contract requirements that utterly failed to meet warfighter needs. Thus, EVMS, while a valuable tool, is not sufficient to fulfill the Panel's recommendations [House Armed Services Committee 2010].*

Accommodating the Agile principles of incremental and adaptive planning, and embracing change in the pursuit of value, can be challenging, especially when faced with the significant implementation guidance related to EVM that mentions nothing about its use in Agile projects. AgileEVM is a new, exploratory practice area within the Agile development community. Proponents suggest that EVM may be applied usefully and validly to Agile software development projects. Proponents also believe that AgileEVM addresses some of the above-mentioned shortcomings of traditional EVMS. However, for AgileEVM to work, it is important that tasks are small and that iterations are short. The most comprehensive treatment to date of AgileEVM may be found in an *IEEE Software* 2006 article, entitled "AgileEVM—Earned Value Management in Scrum Projects" [Sulaiman 2006]. The described method computes AgileEVM for a single release of software and makes use of story points as the fundamental units of work and the fundamental units of earned value.

The above-referenced method of calculating AgileEVM requires the development team / contractor to supply the following data prior to the start of development:

1. performance measurement baseline (PMB)
   (expressed as total number of story points planned for the release)
2. schedule baseline
   (expressed as total number of sprints planned for the release * length (in time) for each sprint)
3. budget at completion
   (expressed as the total budget planned for the release)

During project execution, the following data is collected on a per-iteration basis and used to generate updated AgileEVM calculations:[36]

1. story points completed
2. story points added
3. iteration cost

The above-described method covers the generation of all standard EVMS equations. The assertion that AgileEVM addresses shortcomings within traditional EVMS is based upon the following: AgileEVM calculations are based upon delivery of completed, tested units of functionality. No *credit* is given for delivery of intermediate work products. Therefore, AgileEVM may be seen as incorporating quality standards into the metric and may be seen as providing stricter evidence with respect to delivery of value.

Because the performance measurement baseline (PMB) is expressed as "number of story points planned" rather than at the level of specific tasks, it allows course corrections to be made without disruption or re-baselining of the PMB. This addresses the criticism expressed in the Defense Acquisition Reform Findings and Recommendations (DARFAR) report regarding the inability of traditional EVMS to identify issues related to "contract requirements that utterly failed to meet warfighter needs" [House Armed Services Committee 2010].

5.5 Sustainment

Sustainment of existing software-intensive systems—corrective maintenance and evolution of capability—is a large part of the software activity performed by or on behalf of the US Department of Defense [Defense Acquisition University 2011c]. Agile methods have been successfully used in sustainment as well as new developments in commercial industry, and in fact, some of the program offices that we interviewed either started as sustainment projects or transitioned into sustainment projects during the course of the project's life cycle. One of our reviewers commented, "Agile is perfect for continuous maintenance, [including] many of the NASA Deep Space systems." Among other benefits, one reviewer commented that, for programs they had worked in an Agile fashion for both development and sustainment, "…there is very little change in process or planning artifacts when a product transitions from development to sustainment. This can save an enormous amount of time and money." There is also, generally, alignment between a sustainment effort's periodic releases for patches and the short iterations used in Agile methods.

In sustainment contexts for IT systems with long life, contracting mechanisms tend toward service contracts, in which the contracted element is the staffing of a set of skills anticipated to be needed to sustain the software at a certain capacity. Projects we interviewed in these kinds of sustainment contexts found estimating and tracking using Agile methods and measurements to be useful to the customer as well as the development team. This is because of the strong communication between

[36] One of our reviewers who has used AgileEVM noted, "This is fine as long as the iterations are short. When an iteration is longer than about three weeks, it will be important to calculate percent complete of an iteration based on percent of story points planned for the iteration that are complete to this point in the iteration. This is a type of "information radiator" that can be implemented that basically shows current percent complete of the iteration."

end users and the development team that resulted in a deep understanding of the priorities of the user community being served and a commitment to providing as much value as possible.

In service contracts of less than $20 million EVM threshold, where Agile methods were in use, the use of story point estimation and the formulating of iterations based on product backlog (often consisting of requests and defect reports from the field) were consistently in use. Much of the content in Section 5.4, Contract Execution and Monitoring, applies equally well to sustainment situations as to new start situations.

The biggest difference in sustainment is that an architecture for the system has been defined and implemented. Depending on how well it has been communicated to the sustainment team, there may be constraints on the team's ability to evolve the product to serve end-user needs. This is because the team also needs to adhere to the architectural constraints that are often in place to meet security or other non-functional requirements that may not be obvious to a team taking over an implemented product. In this situation, there may be iterations that are needed from time to time that are expressly focused on evolving the architecture to address new infrastructure or quality attribute requirements. From an estimation process viewpoint, these iterations are likely to be estimated using either *non-user* technical stories, or some other estimation method, such as ideal days.

6 Moving Toward Adoption of Agile in DoD Information Technology Acquisition

6.1 Why Change to Agile Methods?

Changing the practices of an organization is rarely easy. The further the new practices and their methods are from current practice, the more difficult, time-consuming, and resource-consuming the change is.

In Section 1, we pointed out several of the reasons that acquisition and development organizations are adopting Agile methods. Our interviewees exhibited a wide variety of reasons for considering Agile methods. The ones who experienced the greatest success were those for whom the change was important in a particular, rather than a general way. Where the motivation for change was less particular and more general, the changes did not stick as well, nor were the expected benefits realized. The two biggest reasons we have seen within DoD for moving to Agile are

1. a *burning platform:* If the program does not change its current development practice to improve outcomes, it is likely to be cancelled.
2. urgency of delivery: An operational need that cannot wait for traditional delivery times is mission-critical enough to warrant a different acquisition approach.

In the programs that we interviewed, there were other motivations for beginning the change to Agile practices. For the organizations that have been using Agile methods for some time, we heard some common themes that characterized their continuing motivation for change, including

- a sense of true accomplishment when they delivered a release that they knew incorporated functionality the end user needed
- a short time span for seeing the differences their work made to their end users
- encouraging (often laudatory) user feedback that clearly communicated the value of their approach
- consistent ability to meet or exceed user expectations
- previous inability to deliver value within agreed time spans and costs

The above themes were seen among both developers and acquirers. However, all the organizations agreed that their culture and practices needed to change if they wanted the benefits they were seeking from adopting Agile methods. The next section discusses a way to gauge the size of the needed adjustment.

6.2 Understanding the Scope of Change

6.2.1 How Big a Challenge is Your Adoption of Agile Practices?

Paul Adler, a well-known researcher in the field of organizational behavior, has postulated a continuum of change that predicts how difficult it will be for an organization to change its practices. In its simplest version, it looks at factors that indicate increasing level of difficulty, and correlated with that, time to change. The least impacting change tends to be a skills change, followed by procedural change, structural change, strategy change, and the most difficult and

time-consuming, culture change. See Appendix G for elaboration of these factors and their effect on time and complexity of change efforts.

It is clear in our interview population and other experience that different organizations adopt Agile practices to different levels in terms of Adler's progression. All projects we interviewed at least had skill and procedure changes. Many of them incorporated structure changes due to different kinds of interactions with users than in the past. Some also changed their business strategy, and several of them changed their culture to be more amenable to Agile practices. In the book *Becoming Agile*, the authors differentiate "doing agile" versus "being agile" [Sidky 2009]. From Adler's viewpoint, they differentiate "being Agile" as involving the culture change to adopting the values of the Agile manifesto published in 2001 [Adler 1990]. We discussed Agile cultural norms in Section 2. In that section, we introduced some of the differences between Agile culture and more traditional development cultures. We will address that knowledge, as well as other elements that affect the pace and success of change, in this section.

6.2.2 Adoption Assumptions Table for Agile Methods

As part of a readiness and fit assessment (RFA) for adopting Agile methods, a series of adoption factors that include those from Adler have been articulated. RFA can help you to understand how far away your own organization's practices, values, and culture are from what is typical in an organization that has embraced Agile practices and culture [Garcia 2006].

Cultural expectations related to an environment that has successfully adopted Agile methods are divided into two columns. The first lists expectations of what would be seen in a *development* organization adopting Agile methods. The second lists expectations of what would be seen by customers (or in the DoD case, *acquirers*) of a solution being built using Agile methods. This is a starting point list based on our observations of Agile projects in the DoD context. The kinds of behaviors and attributes that are listed in Table 5 are ones that we have seen in at least some of the DoD programs that were adopting Agile methods. Some of the content of this table reflects that of Table 2, which focuses on the cultural aspects explicitly. However, Table 2 only focuses on acquisition personnel. By providing both developer and acquisition personnel expectations, we hope to help acquisition program offices understand the cues they are getting from their development contractors.

As more DoD projects adopt Agile practices, we expect Table 4 to evolve, since particular instantiations of these factors depend on both the product and environmental context.

Refer to Section 6.2.3.3 for information on how to use Table 4 in a readiness and fit assessment.

Table 4: Adoption Expectations for Agile Culture, Practice, and Skill for Developers and Acquirers

Adoption Factor	Agile Culture Expectations—Developer	Agile Culture Expectations—Customer/Acquirer
Business Strategy (What types of business strategy are typical in this environment?)	-customer-centric vs. technology-centric business strategy -value to customer is paramount	-incremental delivery is supported -ongoing solution vs. single point product is needed/supported
Reward System (What are the types of behaviors that are rewarded or punished?)	-team performance is emphasized more heavily than individual -mentoring is rewarded -candor/truth is rewarded, not punished -solving problems that are not directly tied to your iteration goals is NOT rewarded -heroics are not encouraged	-early functioning software release is rewarded vs. documents being "finished"
Sponsorship (What level and types of sponsorship behaviors are needed for adoption to succeed?)	-requires local engineering management sponsorship and senior management buy-in that is sufficient to allow changed practices -vision for change is shared by sponsor and teams -engineering and senior management are willing to adopt "time box" approaches to managing	-willingness to permit access to end-user subject matter experts throughout the development life cycle -willingness to be an active partner in the co-creation of the software solution
Values (What cultural norms are exhibited in the activities and decisions of the organization?)	-keep everything —software, documentation, overhead—as simple as possible -sharing of information vs. "knowledge is personal power" -learning is valued -fail fast, learn fast is encouraged -self-reflection via activities like retrospectives are encouraged -strong team-based values -shared team understanding of what "done" means -developer isolation is frowned upon	-metrics used for course correction, not punishment of individuals -"gold-plating" is discouraged -bloated software is discouraged -minimizing waste is important

Table 4: Adoption Expectations for Agile Culture, Practice, and Skill for Developers and Acquirers (cont'd.)

Adoption Factor	Agile Culture Expectations—Developer	Agile Culture Expectations—Customer/ Acquirer
Skills (What skills and knowledge are prevalent in the individuals in the organization?)	-working well with peers (especially for pair programming) -translating user scenarios (stories) into test cases -working directly with end users or surrogates -different roles working together across boundaries (i.e.., testers and developers working together from the beginning of the cycle) -strong verbal communication skills -ability to move on when product is sufficiently "done"	-strong verbal communication skills -facilitative leadership skills -conflict resolution skills when dealing with diverse stakeholders -ability to recognize when product is sufficiently "done" to move on -working effectively with end users or surrogates
Structure (What kinds of structural mechanisms are used that contribute to the organization's success?)	-colocation; use of collaboration technology to compensate when co-location is not feasible -team roles are flexible -layering of teams when projects get too large for a single team; sometimes this becomes a hierarchy, sometimes it is a set of peer teams	-porous boundaries between acquisition and development staff -support for a strong user (surrogate) presence throughout development iterations and at release
History (How necessary is successful change history to the adoption of this set of practices?)	-history of successful management changes is helpful in getting teams sufficient time to demonstrate new approach to senior management -"burning platform" motivation is often useful in allowing teams to try something different from past practice	-understanding that "silver bullets" don't work is useful
Work Practices (What work practices distinguish this technology or methodology from others?) Note: Not all of these work practices are present in all variations of Agile methods.	-test-driven development -user scenarios creation -pair programming -daily builds; continuous integration -strong version control -project retrospectives -iteration-based release management -daily standup meetings -collaborative planning among team, customers, and management -use of velocity as a primary measure of team progress	-active participation in construction and evolution of a product backlog -active participation in progress reviews and demonstrations -active participation in construction and evolution of user stories -collaborative planning among team, customers, and management -construction of contracting vehicles that support practices (see above) and deliverables typical of Agile methods

In earlier sections of this document, we explored some of the above attributes as well as others that play a role in adopting Agile within a DoD context. As more development organizations adopt Agile practices (some of the literature argues that Agile development has already reached a tipping point in the software industry and is becoming the dominant approach), more DoD acquisition organizations are likely to try to benefit from Agile [Leffingwell 2008]. The recent report to Congress responding to the 804 section requirements of FY 2010, although it does not explicitly call out Agile, embraces many of the tenets of Agile, as we have discussed in the early sections of this report [OSD 2010]. The further organizations are from the attributes in the adoption expectations table above, the more difficult the transition to Agile practices will be. The 804 Implementation Task Force has started studying the DoD and service-level acquisition regulations to find or establish better support for responsive, iterative acquisition practices. Although the task force has found some supportive rhetoric, it also has found regulations that are more easily *interpreted* to prohibit some of the Agile attributes listed above; however, these interpretations are not codified into regulations.[37] For IT systems at least, changes in regulations that make it easier to adopt Agile methods are being contemplated, according to the 804 report [OSD 2010].

6.2.3 Approaches for Successfully Adopting Agile Practices

In interviewing projects that have adopted Agile, we consistently asked, "Did your approach to adopting Agile methods have to be different from your approach to adopting other new practices (e.g., new testing methodologies)?" The consistent answer to this question was, "No, the same things you have to pay attention to for any adoption must be paid attention to for adopting Agile practices." In this section, although we will address some more general organizational change management approaches, we will include, where we have seen it, how these approaches were expressed in our interviewed projects.

There were a variety of approaches that the programs we interviewed took for adopting Agile practices. Most exhibited awareness of cultural issues, adopter populations, and finding or building the right adoption support mechanisms for their environment, even though they might not have expressed their journey in exactly those terms.

The change management topics we will focus on below are typical of the things that must be dealt with in adopting Agile practices:

- understanding your adopter population
- understanding the cycle of change
- understanding your adoption risks
- building transition mechanisms to mitigate adoption risks

6.2.3.1 Understanding Your Adopter Population

When adopting new practices, it is crucial that you understand the characteristics of the people—both as individuals and groups—you are trying to influence to adopt. Geoffrey Moore, a marketing researcher and business consultant, leveraged prior research that indicated that groups are differentiated in terms of their propensity to adopt a new technology, and can be characterized

[37] Boston, J., et al. EX INVEST proposal to Office of the Secretary of Defense. 2010.

in terms of the types of communication and implementation mechanisms they need to feel comfortable with the technology [Rogers 2003]. He noted that, in conjunction with the adopter groups that had been previously defined, a "chasm" appeared to exist between two of the groups, from the viewpoint that getting a technology adopted by one group was more than incrementally different from the next [Moore 2002]. Figure 13 shows Moore's version of the adopter populations curve.

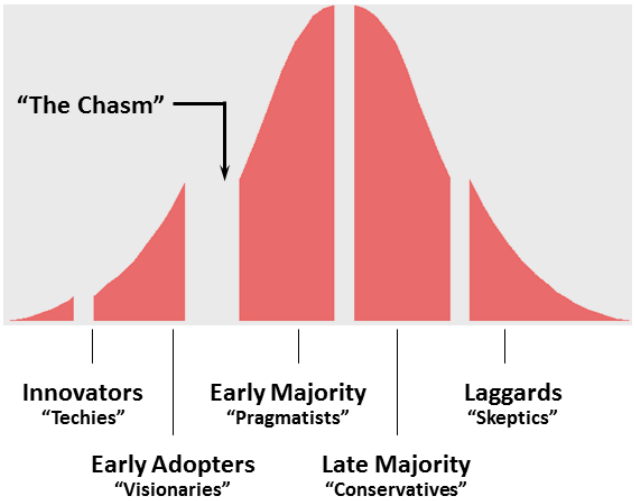


Figure 13: Geoffrey Moore's Adaptation of Rogers' Adoption Populations Curve[38]

The chasm appears between the Early Adopters and Early Majority groups. These two groups are very different in terms of the kinds of communication and implementation mechanisms that will influence them to adopt the new practices or technology. We observed most of the current DoD adopters of Agile to be Early Adopters, although when some programs began their adoption, they were even farther to the left. (For example, some of them were willing to put together their own tools for managing their product backlog and to measure and monitor their velocity.) Where we saw Early Adopter attitudes in the development staff and the acquisition staff, Agile methods had sufficient support to achieve successful use.

The Early Majority adopters, on the other hand, are looking for a more explicitly *proven* benefit and prefer technologies or practices that are supported by training and tools. However, they are still willing to tailor practices to their own environment or domain.

Where we have seen the most conflict in DoD programs in transition toward Agile methods is one where Early or Late Majority acquisition staff had been rotated into a program that had been previously staffed with Early Adopter acquisition and development staff using Agile methods. In particular, plans that had been made for iterative mini-PDRs were reversed by the acquisition staff to a more traditional "one big PDR" event. As some of the recommendations from the 804 Report to Congress are implemented, we hope that more communication and implementation support mechanisms will be made available to Early and Late Majority acquisition staff.

[38] The y-axis, not shown in these figures, represents the percentage of people typically in each group.

What this meant to our interview population is that they have been instrumental in terms of finding ways to "work Agile" in an environment that demands artifacts and evidence based on *working traditional*. In other sections, particularly the Technical Milestones section of this report, we have elaborated some of the ways in which Early Adopters have made Agile methods successful in the DoD acquisition context. However, successful adoption across a wide spectrum of appropriate DoD programs will not occur until more communication and implementation support mechanisms are available to the Early Majority adopters (e.g., DAU courses on how to implement Agile methods in an acquisition program, or other similar guidance).

Late Majority (or Conservative) adopters will wait until it is inconvenient to not adopt a new set of practices. The Agile methods proponents are still several years away from penetrating the Late Majority software development community. Evidence of this shift will be when there are Agile variants specific to different domain sectors (like banking) and when Agile methods are the dominant approach being taught at the undergraduate level in software engineering.

All of the programs we interviewed exhibited either Innovator or Early Adopter characteristics in their leadership teams, where adoption style will make a huge difference. We are starting to encounter programs that have more traditional middle management even though the senior leadership is interested in adopting Agile methods. Where middle management exhibits more early- or late-majority characteristics, versions of Agile methods with more *packaging* will have to be used to help them make the transition. For example, an appendix to 5000.02 or Air Force Instruction (AFI) acquisition life cycle regulations that provides explicit guidance on contents and sequencing of technical milestones in a program using Agile would be a part of such packaging, along with tailoring guidance for known variations of program types. Support in terms of policies that visibly drive toward Agile methods is another example of more packaging for Agile approaches.

6.2.3.2 Understanding the Cycle of Change

A precept of organizational change is that any change—positive or negative—takes effort and time to incorporate into an individual's mental models and into a group's behavioral norms. Even for changes that we perceive as positive, there is a series of steps that we invariably go through (usually faster for positive changes) before achieving the results that are desired from a change in behavior.

Virginia Satir's elaboration of this change cycle has been adopted by well-known consultants and change practitioners in the software industry and is cited explicitly in some Agile literature [Weinberg 1997].

Figure 14 illustrates the basic phases of Satir's change cycle.

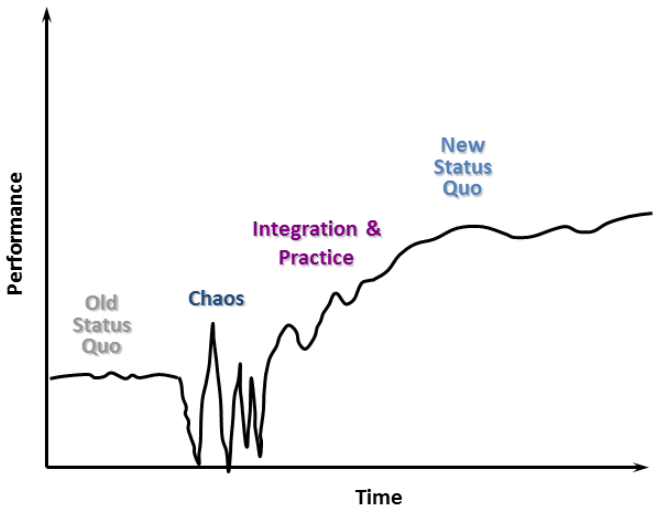


Figure 14: Satir Change Model

The Old Status Quo is disrupted by the introduction of a *foreign element*—a change that will require different skills, procedures, etc., and will presumably improve overall performance. This precipitates a stage of highly variable performance, as the individual or group tries to accommodate the change within its current frame of reference (for a positively perceived change) or tries to reject it (for a negatively perceived change). Chaos continues until the individual or group discovers a *transforming idea*. The transforming idea is something that allows the individual to perceive some intended benefit from the change and incorporates some idea of how to connect the change to his or her own behavior and values.

It is easy to think that, at this point, the New Status Quo, complete with improved performance, will be reached. However, a stage of integrating and practicing the new behavior, and fine-tuning it to meet the individual's context, will be needed before the New Status Quo, with its associated benefit in performance, can actually be reached.

As a whole, DoD's adoption of Agile is centered in the Old Status Quo and Chaos sections of the graph, with a few programs that are in Integration and Practice, and a handful at best where Agile is the New Status Quo. In our interviews, we saw evidence of this cycle. It was common for organizations to phase in their adoption of Agile methods over at least a couple of years, allowing staff to get accustomed to a new set of practices (e.g., pair programming) before adding in another set (e.g., continuous integration). When this kind of phasing is done, while conscious of where the individuals generally are in relation to the Satir cycle, the Chaos stage can be minimized. In most organizations, the optional point to add in a new practice is shortly after a New Status Quo has been achieved. At that point, the new behaviors are fairly embedded, but have not become so ingrained that they are difficult to modify if needed. However, piecemeal adoption of Agile methods, if not paced wisely, has also resulted in lack of full adoption and lack of benefit realized.

In some cases, we saw an organization that was well on its way into Integration and Practice but was thrown back into Chaos when a new foreign element was introduced. In particular, backsliding was seen when they were asked to continue achieving the success they were seeing with Agile practices while maintaining the familiar technical milestone events of a single PDR

and single CDR. From a Satir viewpoint, the solution to this dilemma is to find a transforming idea that allows both approaches (Agile and traditional PDRs) to coexist. If there is insufficient trust between the different parties advocating for the conflicting practices, finding that transforming idea could easily fail. We observed that failure firsthand in one of the programs with which we were involved.

6.2.3.3 Understanding Your Adoption Risks

Once you have some ideas about your motivation for change, the scope of the change, and the population you want to adopt the change, you can get more specific about identifying the particular adoption risks you are likely to face when adopting Agile development and acquisition approaches. One effective way of identifying adoption risks is to use Table 4, introduced in Section 6.2, as a basis for analyzing how closely your current organization exhibits the kinds of attributes that are typically seen in Agile environments. This method has been successfully used in other areas, including CMMI adoption and complex commercial off-the-shelf software adoption (like enterprise resource planning (ERP) software) [Garcia 2006]. We briefly summarize below an organizational evaluation method called Readiness & Fit Analysis. For additional detail on applying Readiness & Fit Analysis (RFA), refer to Chapter 12 in *CMMI Survival Guide: Just Enough Process Improvement.*

By surveying or interviewing potential Agile team members, you can get an estimate of how *close* the organization's attributes are to the Agile attributes in Table 4. Usually a 1-to-5 scale is used. The results can be aggregated by group and reported via a histogram or kiviat (radar chart) diagram to get an overall perception of fit. Beyond the summary visualization, it is extremely valuable to ask participants, via either a survey or workshop or interview, for specific issues they see in a particular topic area that would impede adoption or, conversely, specific strengths they see that would improve the chances of successful adoption. These opportunities for leverage, issues, and risks (participants will naturally identify some of each) can be grouped into similar themes (which may or may not reflect the categories in the expectations table—often they do not), and mitigation planning can be done to address the issues.

There are dozens of adoption support mechanisms that can be acquired or developed to help an organization through an adoption of new practices. The mechanism used across the majority of our interviewed programs was external training and consulting. Sometimes this was conducted by the iconic agilists in the field (one group hired Ken Schwaber, a well-known and prolific agilist, as its consultant/trainer). Agile consultants and trainers usually addressed four particular areas of the assumptions table for Agile methods: practices, skills, sponsorship, and values. Other transition mechanisms used included facility changes (substitution of team rooms for individual cubicles), changes to program-level management directives, formation of Agile methods communities of interest, and lunch-and-learn activities where individuals from the team chose a topic about which to become more knowledgeable.

The most obvious of the four areas focused on by Agile consultants is practices. Training is a standard way of transferring knowledge about what practices are relevant to a particular situation and how to execute them. Successful use of practices depends, of course, on appropriate skills. The skills needed to adopt Agile methods, in terms of individual techniques, can be acquired through basic training. However, a more complete training series that includes education on Agile culture and values provides a basis for helping team members on an Agile program make

decisions that are consistent with the Agile culture even if the situation was not exactly the same as those covered in the class. At least one of our interviewed organizations took this extra step with its training approach.

Another program used team coaching from the consultant during a real project as a way to help the team recognize, transition, and apply the values of an Agile culture. Both approaches worked for the groups involved.

The other role a good Agile consultant played in some of the programs we interviewed was as coach to the sponsors of the Agile methods adoption. Early-adopter sponsors often understand the broad concepts and goals of a set of methods like Agile methods, but may not be as certain as to how their typical behaviors will enable or create barriers for adoption. A good Agile consultant can be very useful in pointing out alternative behaviors for the sponsors that will be more productive in an Agile setting. At least one of the programs successfully used its consultant in that role.

6.2.3.4 Building Transition Mechanisms to Mitigate Adoption Risks

We introduced the concept of communication and implementation support mechanisms in section 6.2.3.3. They are both examples of the more general category of transition mechanisms. Deciding which mechanisms must be built, and at what point in an adoption of new practices, is a common issue, regardless of the technology or practices being adopted. The timing of training, in particular, can have an effect on adoption pace. Training delivered too early cannot be acted on before the skills dissipate. Training delivered too late rarely is successful in refocusing negative impressions of the practices, especially ones that require new skills.

A model we have found useful for almost two decades in figuring out *what* adoption support mechanisms to build *when* is based on the adoption commitment curve, illustrated in Figure 15.

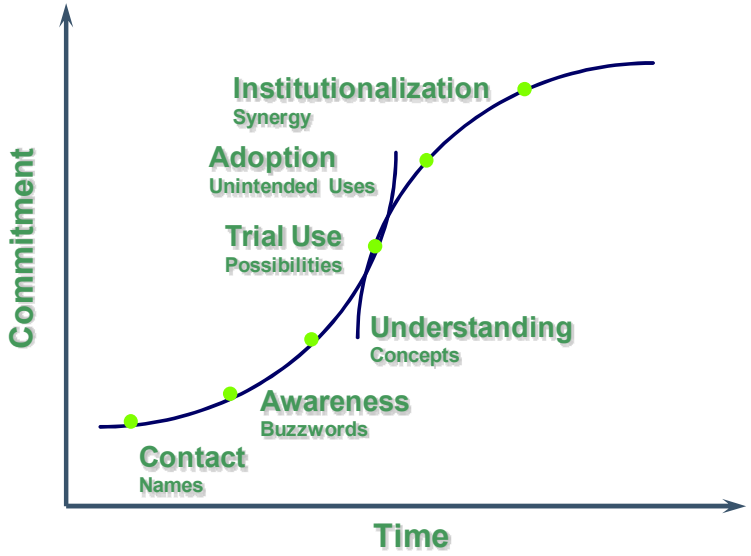


Figure 15: Adoption Commitment Curve [Conner 1983]

This curve expresses, in general, the difference in commitment that is needed by an individual or group to adopt a new technology or set of practices. The early, "easy" stages—contact, awareness, and understanding—primarily rely on communication mechanisms to help people move forward. The latter, "hard" stages—trial use, limited adoption, and institutionalization—rely on implementation support mechanisms that tend to be more specialized to the technology and the organization than the types of communication mechanisms that are used earlier in the adoption.

For Agile methods, some candidate transition mechanisms that might be useful at each stage of adoption in the DoD are proposed in Table 5. These are based on our team's interviews with DoD programs and our knowledge of Agile patterns of adoption, and have not been reviewed for their ease or difficulty to implement in DoD or service regulations and directives. They are suggestive, more than exhaustive or complete.

Table 5: Candidate Transition Mechanisms for DoD Adoption of Agile Methods Keyed to Adoption Commitment Curve Stages

Contact/ Awareness	Understanding	Trial Use	Adoption	Institutionalization
Articles in ***Crosstalk***, ***Defense Acquisition News***, and other publications on programs successfully using Agile methods	Course modules in DAU acquisition courses discussing when and how to use Agile methods in DoD acquisition	Standard language for Milestone Decision Authorities to use to except programs using Agile methods from waterfall practices that conflict with Agile methods	Standard reporting templates for oversight reports expected in an Agile acquisition	Language changes to DoD 5000.02 and service regulations that make clear when to use Agile methods and provide guidance on making Agile methods effective within an acquisition
Conference tracks and workshops that highlight the benefits and risks associated with adopting Agile practices	Conferences and workshops that focus on implementing Agile concepts in a DoD acquisition environment	Provisions for programs attempting Agile methods for the first time to get continual direct access to their end-user communities	Guidance on use of story points and other Agile estimation mechanisms in GAO Estimation Best Practices Guide	Citation of the successful use of Agile methods in programs that are held up as models for DoD acquisition success (obviously only if the use of Agile methods is true!)

The adoption commitment curve helps to explain why training can sometimes be too early in an adoption cycle. Training is primarily used as an awareness-building mechanism (one-day Agile methods overview, for example), or as an understanding-building mechanism (five-day Agile methods course where students exercise all the skills associated with that particular method on case study projects or their own). When skills-focused training is used as an initial event, it is usually overkill—the potential adopters have not had sufficient exposure, contact, and awareness to the practices or technology to "make it their own." Most of the programs we interviewed used external training, primarily of the skill-building type, to jumpstart their Agile teams. Some of them used general training as a way to build awareness of the coming change with both managers and development staff. Others used conferences, readings, and guest speakers at meetings as a way to build awareness of Agile methods before embarking on skill-building training.

Coupling the adoption risks list generated via RFA with the adoption commitment curve is a powerful way of planning "just in time" adoption support mechanisms for your Agile adoption. As we gain more insight into adoption of Agile methods across a broader range of DoD programs,

it may be possible to better characterize typical mechanisms that have been successful in different acquisition settings.

6.2.4 Organizational Change Management Summary

Although it is easy to say that Agile methods do not require any different methods for addressing organizational change, it is not so easy to actually apply those methods in demanding project settings, just as it is not easy to do so for other practices or technologies. Someday there may be a "GAO Best Practices Guide for Adopting Agile Methods," similar to the GAO Guide for Estimation. In the meantime, we leave you with the following advice:

- Find and nurture good sponsors for your adoption.
- Understand your adoption population.
- Conduct some kind of readiness assessment that addresses organizational and cultural issues.
- Analyze what adoption support mechanisms you are likely to need for your context and build or acquire them before you get too far into your adoption.

There is a starter list of resources for Agile adoption in Appendix 1, several of which are focused on organizational change management.

7 Summary, Conclusion, and Future Planned Work

This report is the second in a series funded by SAF/AQX. At the end of our first report, we provided a list of topics that we thought needed further research. Many of those have been presented here. However, we also acknowledge that our treatment of these subjects is by no means complete as the body of knowledge on these topics continues to evolve.

Since the publication of the first report and while we have been pursuing the contents for this report, there has been considerable movement within the government and DoD to identify and implement a new acquisition process that can take advantage of Agile methods. This movement extends from recognition by the former Secretary of Defense, Robert Gates, that conventional modernization programs may not meet the existing demand in today's environment, to Congress' inclusion of Section 804 in the National Defense Authorization Act for Fiscal Year 2010. Pursuant to Section 804, OSD published a report providing updates on DoD's progress toward developing a new acquisition process for information technology capabilities [OSD 2010].

This recognition shows an understanding that the acquisition tempo must respond to the operational tempo. In addition, there is still a need for process discipline that ensures effective use of resources in providing needed capabilities. While some may say that Agile does not have the appropriate discipline for use within the DoD, Agile does require adherence to processes and there is evidence that using Agile methods in conjunction with other methods like CMMI is a powerful approach to achieving the needed effectiveness. In addition to effectiveness, a focus on value must be maintained. Agile practitioners have evolved the classic iron triangle to include value.

Even though value is included, those within DoD that have adopted Agile methods have learned that a change in mindset and culture for the PMOs and other acquisition entities is required. In order to change culture, you need to understand your current culture, the assumptions, shared values, and artifacts that make up your current culture, and what the differences are with a new Agile culture. Table 2 shows a comparison of the Agile and traditional DoD cultural elements. This table should help those who want to adopt Agile methods understand some of the differences and changes they will need to make. Table 3 provides some potential PMO actions and enablers that can support the use of Agile methods. The end goal of many of these cultural shifts is to enable partnership and shared understanding of what value means from a customer perspective. When the acquirer, developer, and end user all share the same understanding of value, it is much easier to agree upon priorities.

In order to make the transition to Agile, one must understand the terminology and the differences between those of Agile practitioners and DoD acquisition personnel. This is only the beginning of "being Agile." "Being Agile" is more than adopting just another methodology. The key to "being Agile" is embracing change.

Another part of becoming Agile is learning the common traits of Agile managers. Agile managers are leaders, coaches, expeditors, and champions. Agile managers must be good team builders as the team within the Agile culture is the cornerstone of the thought process and method. Managers also need to master the time box, understand what contract types work best, learn which metrics apply to Agile programs, and deal with distributed teams in an Agile manner.

One of the main sticking points with using Agile methods in DoD acquisition is how to accommodate large capstone events such as CDR. There are other challenges that must be addressed such as incentives to collaborate, definitions, and regulatory language. While there are many issues in this arena, the main thing to remember is the purpose and intent of holding these reviews in the first place. The purpose is to evaluate progress on and/or review specific aspects of the proposed technical solution. Thus, expectations and criteria should be created that reflect the level and type of documentation that would be acceptable for the milestone. This is not any different from business as usual. However, the key here is for the government PMO and the contractor to define the level and type of documentation needed, while they work within an Agile environment that is unique to each program.

Estimating for Agile acquisition is another area that required considerable exploration. Estimation in Agile is different from traditional estimation. In Agile, the estimates tend to be just-in-time with a high-level estimate refined to create detailed estimates as more is learned about the requirements. Traditional methods are more detailed up front with the details being refined as more is learned. There are several parametric tools and even an AgileEVM tool that can be used in the Agile environment. Both the estimator and the reviewer need to be aware that estimation within Agile is different from traditional estimation and should act accordingly. How specific issues can be dealt with is highly contract-specific but general guidance is provided within Section 5.

We addressed the road to Agile adoption in DoD software acquisition. Our interviews revealed two main reasons to adopt Agile—moving from a burning platform and an operational need that cannot wait for traditional delivery times. Change is hard. Understanding the scope of the change is essential. Table 4 addresses different adoption factors and expectations for the developer and the customer/acquirer. Lastly, Table 5 provides some candidate transition mechanisms for DoD adoption of Agile methods. Remember to find and nurture good sponsors for your adoption, understand the adoption population, conduct a readiness assessment, and determine what adoption mechanisms you will need and have them on hand early in the adoption process.

Our journey into finding ways for Agile methods to be adopted within DoD has been challenging and exciting, and has taken unexpected turns. With Congressional direction and subsequent involvement of OSD policy makers, we have come to realize the need for practical guidance for the adoption of Agile. Our next endeavor will be to outline and then create a guidebook that DoD users can employ to help them first understand if their program is a good candidate for using Agile and then understand how to go about adopting Agile. In the meantime, we are working with the PEX program (whom we interviewed) to jointly publish a paper outlining some potential changes to Air Force regulations that would ease the adoption of Agile methods.

These additional potential topics for future work were identified during the review of this document:

- Conduct a more in-depth treatment of contracting types.
- Explore views from the lean software development community that it is more important for the contract to define how to negotiate changes than to define what will be built [Poppendieck 2003]

- As one of the reviewers pointed out, Agile methods should be unencumbered by regulation, policy, and law. Agile must put *people* in charge over bureaucratic processes or it will fail. The development teams can be as Agile as they want and they can have a total understanding of the customers' desires (which helps a lot for requirements analysis and trade off) but if the *touch points* (listed below) to the rest of the world are not fixed, we will not see much difference. Thus, there is a need to explore how the following elements impact and need to work with Agile methods
  - funding (colors of money, time limits on expiration of funds)
  - funding approval (Investment Review Board [IRB], $250,000 limit on sustainment enhancements, standard financial information structure [SFIS], Federal Financial Management Improvement Act [FFMIA])
  - architecture requirements approval (Department of Defense Architecture Framework (DODAF), common logistics operating environment [CLOE])
  - documentation (capability production document (CPD), test and evaluation master plan (TEMP), information support plan (ISP), economic analysis (EA), capabilities-based assessment (CBA), business case)
  - data center lead times
  - contract lead times (justifications and approvals [J&As] for add-on products by a vendor whose proprietary stack you already own)
- The relationship of system engineering and Agile development methods determines how to address the naming and content of technical milestones within an Agile context

As should be clear from the above list, there are still significant implementation areas for using Agile methods in DoD acquisition that warrant study. Some of these topics will be covered in our planned work.

Appendix A: Acronyms

ACAP	Analyst Capability
AFB	Air Force Base
AFEI	Association for Enterprise Information
AFI	Air Force Instruction
APO	Acquisition Program Office
APEX	Application Experience
AsD	adaptive software development
BCR	baseline change request
CBA	capabilities based assessment
CDR	Critical Design Review
CDRL	contract data requirements list
CEO	chief executive officer
CLOE	common logistics operating environment
COCOMO	Constructive Cost Model
COTR	contracting officer's technical representative
CPD	capability production document
CPFF	cost plus fixed fee
CPIF	cost plus incentive fee
CMMI	Capability Maturity Model Integration
DAG	Defense Acquisition Guidebook
DARFAR	Defense Acquisition Reform Findings and Recommendations
DASA CE	Deputy Assistant Secretary of the Army for Cost & Economics
DAU	Defense Acquisition University
DBS	defense business systems
DFAR	Defense Federal Acquisition Regulation
DIACAP	DoD Information Assurance Certification and Accreditation Process

DID	data item description
DoD	Department of Defense
DoDAF	Department of Defense Architecture Framework
DoDD	DoD Directive
DoDI	DoD Instruction
DSDM	Dynamic Systems Development Method
DT&E	development test and evaluation
EA	economic analysis
EVM	earned value management
FAR	Federal Acquisition Regulation
FCs	functional capabilities
FFMIA	Federal Financial Management Improvement Act
FFP	firm fixed price
FLEX	Development Flexibility
FP	fixed price
FRP	full rate production
FY	fiscal year
GAO	Government Accountability Office
IA	information assurance
IDIQ	indefinite delivery indefinite quantity
INVEST	Innovation for New Value, Efficiency, and Savings
IOT&E	initial operational test and evaluation
IRB	Investment Review Board
ISP	information support plan
IT	information technology
J&As	justification and approvals
LCA	life cycle architecture
LCO	life cycle objectives

Lt Col	Lieutenant Colonel
LRIP	limited/low rate initial production
MDA	Milestone Decision Authority
MOA	memorandum of agreement
MOU	memorandum of understanding
NDAA	National Defense Authorization Act
O&M	operations and maintenance
OSD	Office of the Secretary of Defense
OT&E	operational test and evaluation
PCAP	Programmer Capability
PDR	Preliminary Design Review
PEX	Patriot Excalibur
PHP	Hypertext PreProcessor
PL	public law
PLCCE	program life cycle cost estimate
PM	program manager
PMB	performance measurement baseline
PMO	program management office
PWS	program work statement
QR	quality review
RESL	Architecture / Risk Resolution
RICE	reports, interfaces, conversions, enhancements, or extensions
RFP	request for proposal
RUP	Rational Unified Process
SAF	Secretary of the Air Force
SDR	System Design Review
SEER	Software Evaluation and Estimation of Resources
SEER-SEM	Software Evaluation and Estimation of Resources – Software Estimation Model

SEI	Software Engineering Institute
SETA	Systems Engineering and Technical Assistance
SFIS	standard financial information structure
SIDRE	Software Intensive Innovative Development and Reengineering/Evolution
SLIM	Software Lifecycle Management-Estimate
SLOC	source lines of code
SOW	statement of work
SRR	System Requirements Review
SSR	Software Specification Review
TEAM	Team Cohesion
TEMP	Test and Evaluation Master Plan
TN	technical note
TSP	Team Software Process
V&V	verification and validation
WBS	work breakdown structure
XP	eXtreme Programming

Appendix B: Glossary

Backlog
An accumulation, especially of unfinished work or unfilled orders.[39]

Done

1. Having been carried out or accomplished; finished.[40] Author's note: In an Agile context, the definition of done can include software, documentation, testing, and certification being complete or any subset of this list being completed. The developer and product owner must agree on what is included in "done". With this in mind, another definition: 2. The useful definition of **doneness** stresses the goal of all Agile iterations: the product must remain shippable.

- All visible features work
  - as advertised
  - within the expected environment
  - in any combination
  - without degradation over time
  - with graceful handling of errors
- Hide all broken or unfinished features

This definition of doneness emphasizes this result: we want a stable app at all times. When we start the app, we know what is expected to work because we can see it and try it. We can prioritize new features by seeing how they must be reconciled with already-visible features.[41]

Epic
A connected or bundled set of stories that result in a definable (in the case of software, desirable) capability or outcome. An epic is a large user story. It is possible to break up an epic into several user stories.[42]

Iteration
In Agile software development,[43] a single development cycle, usually measured as one or two weeks. An iteration may also be defined as the elapsed time between iteration planning sessions

Just enough
Combining the two dictionary definitions of "just" and "enough" you get "exactly sufficient." Within the Agile community, this is an appropriate definition. Thus: just enough to be successful, to get started, support the user story queue, accomplish our goal.

---

[39] http://www.thefreedictionary.com/backlog

[40] http://www.thefreedictionary.com/done

[41] http://billharlan.com/pub/papers/Agile_Essentials.html

[42] http://www.targetprocess.com/LearnAgile/AgileGlossary/ThemeEpic.aspx

[43] http://searchsoftwarequality.techtarget.com/definition/iteration

Pattern
1. A form of knowledge management. It is a literary form for documenting a common, successful practice. It articulates a recurring problem, as well as the context of the problem and the conditions that contribute to creating it. Likewise, the solution, the rationale for the solution, and consequences of using it are given.[44] 2. A way to capture expertise. Patterns document good ideas—strategies that have been shown to work well for a variety of people in a variety of circumstances.[45]

Product Backlog
The master list of all functionality desired in the product.[46]

Release
5a. The act or an instance of issuing something for publication, use, or distribution. 2. Something thus released: a new release of a software program.[47]

Sprint
A set period of time during which specific work must be completed and made ready for review.[48] Often used as a synonym for iteration.

Story
In **Agile software development**, a story is a particular business need assigned to the software development team. Stories must be broken down into small enough components that they may be delivered in a single development **iteration**.[49]

Story Point
According to Cohn, "Story points are a unit of measure for expressing the overall size of a user story, feature, or other piece of work …The number of story points associated with a story represents the overall size of the story. There is no set formula for defining the size of a story. Rather a story-point estimate is an amalgamation of the amount of effort involved in developing the feature, the complexity of developing it, the risk inherent in it and so on."[50]

Technical Debt
Technical debt and *design debt* are synonymous, neologistic metaphors referring to the eventual consequences of slapdash software architecture and hasty software development. **Code debt** refers to technical debt within a codebase.

---

[44] www.eberly.iup.edu/abit/proceedings%5CPatternsAPromisingApproach.pdf

[45] Fearless Change, Patterns for Introducing New Ideas, Mary Lynn Mann, Linda Rising, Addison-Wesley, 2005, Pearson Education, Inc

[46] http://www.mountaingoatsoftware.com/scrum/product-backlog

[47] http://www.thefreedictionary.com/release

[48] http://searchsoftwarequality.techtarget.com/definition/Scrum-sprint

[49] http://searchsoftwarequality.techtarget.com/definition/story

[50] Cohn, M. , Agile Estimating and Planning, P. 36

Ward Cunningham first drew the comparison between technical complexity and debt in a 1992 experience report:

> *Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite... The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt. Entire engineering organizations can be brought to a stand-still under the debt load of an unconsolidated implementation, object-oriented or otherwise [Ozkaya 2011].*

Timebox

A fixed amount of hours or days in which to accomplish something.[51]

Timeboxing

A planning technique common in planning projects (typically for software development), where the schedule is divided into a number of separate time periods (**timeboxes**, normally two to six weeks long), with each part having its own deliverables, deadline, and budget.[52]

User Story

Descriptions of discrete functionality known to be needed by a particular user segment that is part of the project's audience, and other stories that address infrastructure and quality attributes that are pervasive to the product (e.g., security or usability).

Velocity

Velocity is a measure of a team's rate of progress. It is calculated by summing the number of story points assigned to each user story that the team completed during the iteration. If the team completes three stories each estimated at five stories, its velocity is fifteen. If the team completes two five-point stories, its velocity is ten.[53] Velocity, in the Agile community, refers to the amount of capacity of a *particular* team to produce working software. It does not have a general analogue in traditional DoD projects.

---

[51] http://www.agileadvice.com/archives/2006/02/timeboxing_a_cr.html

[52] http://en.wikipedia.org/wiki/Timeboxing

[53] Cohn, M. *Agile Estimating and Planning*, p 38.

Appendix C: Culture Details

This appendix adds a bit of detail on some of the research basis for the approach to cultural issues seen in Section 2.

Dimensions of Culture

Formality and formal communication define the absolutes of a group or organization, and breaking these rules creates anger. This is because there is often emotion around what is formal and rule-based. Informality describes the latitude or surroundings that exist around the absolutes and pushing the limits of the informal can generate anxiety. Informal understanding is often tacit; it is what you can gather from mentors, role models, and exemplars in the organization. It includes

- unwritten rules
- everyday behavior
- common sense approaches
- common courtesy
- what makes people angry
- what is insulting, admirable, or praiseworthy
- interactions with others
- where people spend their time

Operational concerns are made up of formal and informal elements, and changes in this area are often seen as bothersome or irritating [Hall 1980]. The operational or *intentional* aspects of the culture have been codified and people can talk about them. Operational aspects include:

- policies enforced
- teaching/training mechanisms
- rites of passage; myths and legends
- rituals
- celebrations

These dimensions of culture are different from actual communication styles, which can also be described as formal or informal. Because of the emphasis on flexibility, people interactions, collaboration and working software (as opposed to processes, tools, plans, and documentation), Agile styles are seen as informal. The DoD context is often described as the reverse—as formal. But it is important to pause here: in looking at dimensions of culture and communication styles, it is essential to consider our own unconscious assumptions. For example, someone who believes that people need controls, authority, and tight structures in order to be productive, is likely to interpret the informality and flexibility of an Agile environment as reflecting laziness, or a lack of focus and discipline. Our own unconscious assumptions can blind us from understanding the values, norms, and rules and practices of another culture.

Appendix D: COCOMO Factors List

One popular parametric cost-estimation tool is the COCOMO model. First published by Dr. Barry Boehm in his 1981 book, *Software Engineering Economics*, COCOMO (Constructive Cost Model) is an algorithmic-based parametric software cost-estimation model for estimating a software project as an "effort equation," which applies a value to tasks based on the scope of the project (ranging from a small, familiar system to a complex system that is new to the organization). COCOMO II is the successor of COCOMO 81, incorporating more contemporary software development processes, such as code reuse, use of off-the-shelf software components, and updated project databases [Boehm 1981].

At the heart of the COCOMO II model are the cost parameters themselves. These parameters are scale factors (5) and effort multipliers (17). Scale factors represent areas where economies of scale may apply. Effort multipliers represent the established cost drivers for software system development. They are used to adjust the nominal software development effort to reflect the reality of the current product being developed.

It would be reasonable to assert that an Agile development process would have an impact on some of these parameters. The following scale factors and effort multipliers, pulled from COCOMO II, might be impacted by the use of an Agile development process:

Development Flexibility (FLEX) Scale Factor

Definition: The FLEX scale factor is related to the flexibility in conforming to stated requirements.

Rationale: The participation of the user in the Agile development process, coupled with an iterative approach to building, should lower cost and schedule variance, because appropriate use of the methods assures continual communication as situations change. This permits appropriate reprioritization when needed.

Architecture/Risk Resolution (RESL) Scale Factor

Definition: The RESL scale factor is related to early, proactive risk identification and elimination. The goal is to eliminate software risk by Preliminary Design Review (PDR). This factor is also related to the need for software architecture.

Rationale: Although there is opportunity to tackle high-risk items early in the product lifecycle with an Agile approach, there is no guarantee that this will actually happen. The lack of clear guidance regarding how to accomplish a milestone review in an Agile development process, and the general lack of consensus in the Agile community on the need for or approach to developing a viable architecture, could increase the cost estimate.

Team Cohesion (TEAM) Scale Factor

Definition: The TEAM scale factor accounts for sources of project turbulence and entropy because of difficulties in synchronizing the project's stakeholders (e.g., users,

customers, developers, maintainers, and interfacers). These difficulties may arise from differences in stakeholder objectives and cultures, difficulties in reconciling objectives, and stakeholders' lack of experience and familiarity with operating as a team.

Rationale: The Agile culture, in addition to the frequent interchanges between the user and the developers, should provide plenty of opportunity to improve team cohesion and should lower the cost estimate.

Analyst Capability (ACAP) Effort Multiplier

Definition: Analysts are personnel who work on requirements, high-level design, and detailed design. The major attributes that should be considered in this rating are analysis and design ability, efficiency and thoroughness, and the ability to communicate and cooperate.

Rationale: The participation of users in the development process should improve the knowledge of the analysts that elaborate the requirements and produce the software design. The impact of the improvement should lower the cost estimate.

Programmer Capability (PCAP) Effort Multiplier

Definition: Current trends continue to emphasize the importance of highly capable analysts. However, the increasing role of complex COTS packages, and the significant productivity leverage associated with programmers' ability to deal with these COTS packages, indicates a trend toward higher importance of programmer capability as well. Evaluation should be based on the capability of the programmers as a team rather than as individuals. Major factors that should be considered in the rating are ability, efficiency, and thoroughness, and the ability to communicate and cooperate.

Rationale: The participation of users in the development process should improve the knowledge of the programmers who write the software code. This is the factor that most relates to the Agile measure of velocity. The impact of the improvement should lower the cost estimate.

Application Experience (APEX) Effort Multiplier

Definition: The cost-estimating multiplier based on the domain knowledge and capability of the software development staff is called APEX. The rating for this cost driver is dependent on the level of applications experience of the project team developing the software system or subsystem. The ratings are defined in terms of the project team's equivalent level of experience with this type of application.

Rationale: The participation of users in the development process should improve the domain knowledge of the development team. The impact of the improvement should lower the cost estimate.

Appendix E: Estimating Process for Agile Based on GAO Best Practices for Estimation

In the report, *GAO Cost Estimating and Assessment Guide*, the GAO has provided a useful guide for helping a program office understand the mechanics of cost estimation. Figure 16 shows the generalized estimation process recommended by the GAO. This section will overview the steps in the GAO's process, and highlight areas where particular approaches or issues come up when discussing estimation of projects that use Agile methods [GAO 2009].

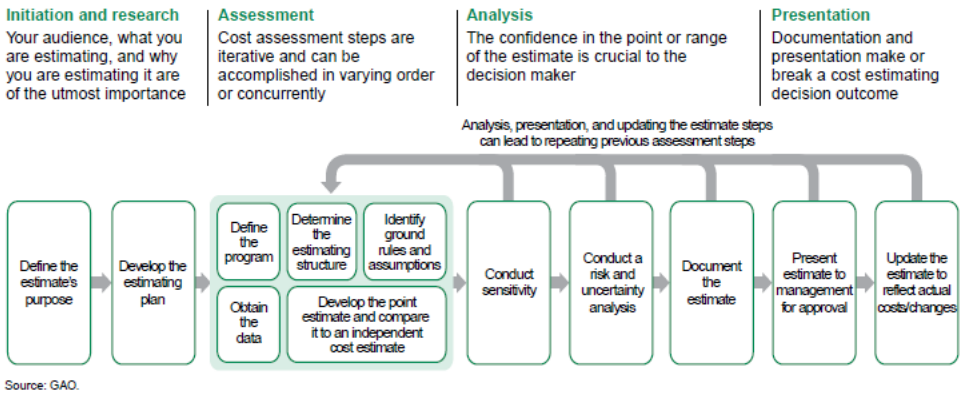


Figure 16: GAO-09-3SP Estimation Process Diagram

The process of estimation laid out by the GAO is applicable to a project using Agile methods as well as to any other kind of project. An Agile project's estimation is likely to look different from a traditional project's in terms of the estimating structure, ground rules and assumptions, and the data that is gathered to support the estimate.

The sections that follow will address these issues from the viewpoint of the team that is making the estimate (in DoD acquisitions, often a contractor). It is our belief that an understanding of the estimating team's process will make it easier for the team evaluating the estimate (typically the government program office) to judge the appropriateness of the estimate.

Estimating Structure

Steps in the GAO guide for determining the estimating structure include:

- Define a work breakdown structure (WBS) and describe each element in a WBS dictionary (a major automated information system may have only a cost element structure).
- Choose the best estimating method for each WBS element.
- Identify potential cross-checks for likely cost and schedule drivers.
- Develop a cost-estimating checklist.

Agile projects we interviewed had used multiple estimating structures, including product-oriented ones similar to traditional programs, as well as WBS that were based on releases and sprints (short, usually two- to four-week iterations), a typical construct of Agile projects. Where projects

were able to build the estimating structure to mimic the project approach (i.e., the releases/sprints approach), estimates were easier to construct and communicate.

Ground Rules and Assumptions

The steps in the GAO guide for identifying ground rules and assumptions include:

- Clearly define what the estimate includes and excludes.
- Identify global and program-specific assumptions, such as the estimate's base year, including time phasing and life cycle.
- Identify program schedule information by phase and program acquisition strategy.
- Identify any schedule or budget constraints, inflation assumptions, and travel costs.
- Specify equipment the government is to furnish as well as the use of existing facilities or new modification or development.
- Identify prime contractor and major subcontractors.
- Determine technology refresh cycles, technology assumptions, and new technology to be developed.
- Define commonality with legacy systems and assumed heritage savings.
- Describe effects of new ways of doing business.

This is the step where communicating about an Agile program's intended use of Agile methods is important, because the typical assumptions that underlie traditional estimates are likely to be different.

Some of the above items (like specifying equipment) are not specifically tied to use of Agile methods. This is a case where the ground rules come from the acquisition program, and the assumptions are provided by the estimator.

There are three specific areas where discussions about Agile methods can easily be called out—identify program schedule information by phase and acquisition strategy; identify schedule or budget constraints; and describe the effects of new ways of doing business.

When discussing program schedule in terms of acquisition strategy, discussing the Agile practices related to evolving functionality over multiple releases, and not completely specifying requirements at the beginning of the project, are relevant discussion points. For example, some programs can tolerate having an approved list of capabilities, versus a highly specific list of parameterized requirements. The level of abstraction of capabilities and requirements should be specified in the SOW so that there is no misunderstanding as the effort proceeds.

In terms of describing effects of new ways of doing business, this is where a program office can lay out its expectations in terms of releases, sprints, types of stories to be supported, continuous integration/test environments expected to be provided, and other practice and technology elements that may be involved with the particular Agile methods they wish to support. Estimators need to reveal their own methods and assumptions and how those are expected to affect how business will be conducted.

Contractors using Agile methods that we interviewed emphasized the importance of creating a shared mental model of ground rules and assumptions between the developer and the program office. Where a program office wanted to both use Agile methods on the project and apply traditional cost estimation (having their cake and eating it too), contractors tended to pick up the effort of translating their estimates based on Agile ground rules into estimates that reflect more traditional size or size surrogate-based approaches. Some of this is helped by methods like AgileEVM, which is discussed in Section 5.

Data

The steps in the GAO report for obtaining data include:

- Create a data-collection plan with emphasis on collecting current and relevant technical, programmatic, cost, and risk data.
- Investigate possible data sources.
- Collect data and normalize them for cost accounting, inflation, learning, and quantity adjustments.
- Analyze the data for cost drivers, trends, and outliers and compare results against rules of thumb and standard factors derived from historical data.
- Interview data sources and document all pertinent information, including an assessment of data reliability and accuracy.
- Store data for future estimates.

Agile projects are actually quite data-centric. However, the data that is gathered is a bit different from traditional projects, and its use in estimation is typically different as well.

A Nominal Agile Estimation Process in a DoD Context

Based on Mike Cohn's Agile Estimating and Planning, three basic concepts are key to creating and using estimates in Agile projects [Cohn 2006]:

- Estimation of overall size can only give a high-level estimate for the work item, typically measured using a neutral unit such as story points.
- Velocity is a measure of how many points this project team can deliver within an iteration.
- Estimation of effort for a work item translates the size (measured in points) to a detailed estimate using hours for each subtask. This estimation is usually undertaken at the beginning of a sprint or iteration, and is based on the velocity and whatever heuristics have been used to characterize the user stories that are the basis for the work item.

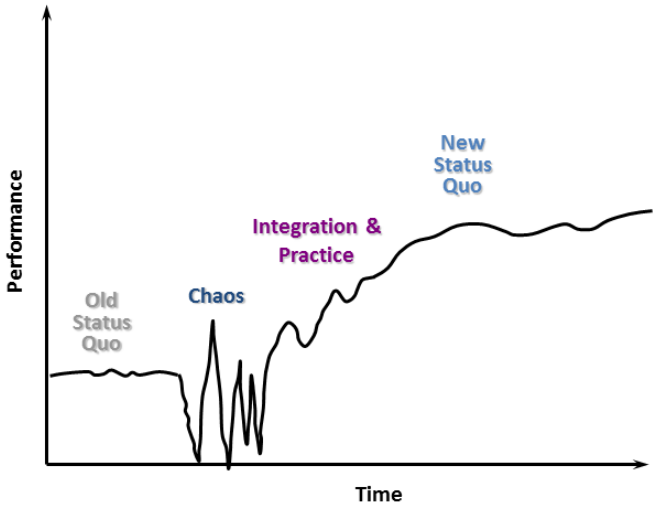
Agile estimation processes, like Agile methods in general, focus on near-term requirements and activities in more detail than longer-term requirements do. Therefore, the first step in Agile estimation is to elicit the requirements that are known at the time of the project start. Many Agile projects are not *fresh starts*, they build on existing legacy software baselines. *Requirements* may come in the guise of defects that need to be corrected or enhancements that are sometimes phrased as defects ("I want the software to do X, but it does Y" when it was never expected to have to do Y). The difference between the two conditions is whether there is an existing architecture for the product.

In summary, as with most other aspects of Agile development, existing practices, such as the GAO estimation practices, can be readily adapted for use in programs using Agile, with appropriate knowledge of both the Agile side and the traditional side of the topic.

Appendix F: Details on Satir Organizational Change Management Model

Satir Change Cycle Details

Remembering the cycle charted in Figure 14 (reproduced below), it is useful to think about how an individual or group actually makes their way through the cycle (or does not, in the case of an unsuccessful change).



The flowchart version of the Satir cycle emphasizes that there are multiple points at which an individual or group can backtrack into a prior stage if they are unsupported in making each shift [Weinberg 1997].

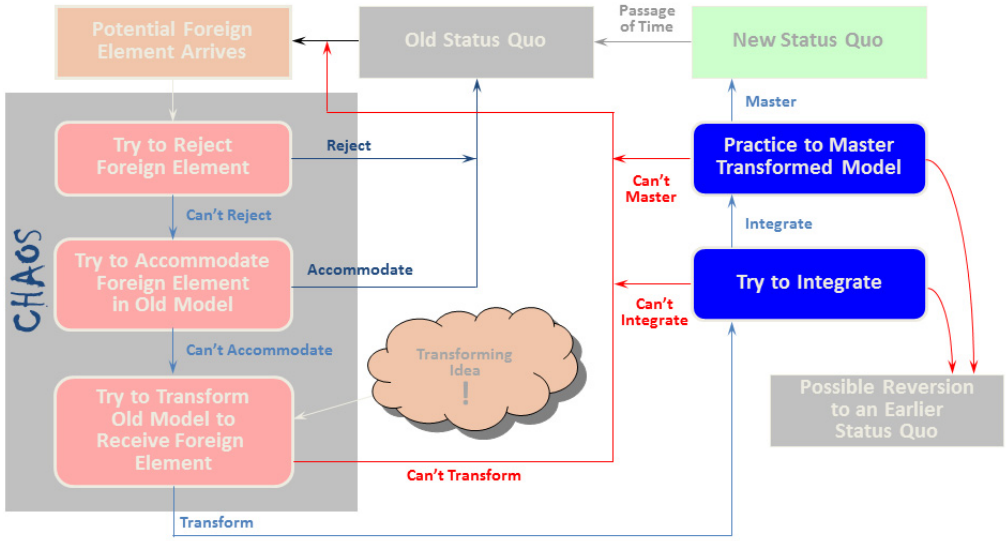


Figure 17: Flowchart Version of Satir Change Model

This model can be very explanatory when observing the behavior of an organization trying to adopt Agile methods. For example, when a program manager tries to use the same technical review agenda for an Agile project that they use for other types of projects, they are trying to accommodate the foreign element (new customer behaviors required by Agile methods) in their old model. Organizations that *all of a sudden* go back to traditional methods after using Agile methods for a period of time are often ones who were not given sufficient opportunity to integrate and practice the new behaviors to the point where they could be (at least) as successful as before.

Appendix G: Adler Factors Related to Complexity and Timing of Change Effort

In the main body of this technical note, we introduced a set of factors that help to determine the scale and scope of an organizational change. The factors are explained in a bit more detail here [Adler 1990].

Skills: The least impacting change involves something where the only thing that needs to change is the skills of the people adopting the new practices. The caveat here is an assumption that the new skill has some grounding in other skills the adopters are likely to have.

Procedures: Next higher on Adler's scale is procedures. When procedures need to change, there is usually a chain of command that must be brought in to the decision. Sometimes the management of an organization is unaware of the procedural changes that adopting a new technology (e.g., an electronic health record system) will have. Changing procedures sometimes also involves changing where power resides within an organization, leading to conflict.

Structure: Beyond procedures, structure is the next higher item on Adler's scale. Structural changes almost always involve changes in power structures, which gets the attention of people who are not necessarily actual adopters of the new practices, but are affected by the adopters, or are affected by the outcomes of new procedures. Any time power and its exercise are involved, passions will run high and resistance to changing the status quo is likely.

Strategy: Strategy goes beyond structure to touching the senior decision-makers in an organization. When business strategy changes, it often means that shifts in the markets are paid attention to, and there are implications for all the factors below it on Adler's scale.

Culture: Culture is at the top of the change difficulty scale from Adler. When the culture is expected to change, it impacts people's values and their assumptions about what behavior is acceptable and not acceptable within the organization. Often these assumptions and values are not explicit; they are discovered primarily by violating one or more of the organization's norms.

Appendix H: Notes from the Field

In the course of conducting interviews and reviews for this document, several of our interviewees and reviewers had their own ideas about what to change or how to change some particular aspect of DoD culture to successfully adopt Agile methods. The following are quotes from either an interviewee or a reviewer, framed in terms of the topic they were discussing. Their roles within DoD acquisition, though not their names, are included.

With regard to delivering more value with limited or shrinking resources, a senior program manager stated:

> *If you have a schedule and you are three months behind … but need ATEC testing … and a test unit for your IOT&E … there is NO way you can postpone to improve quality. If you try, you lose your test unit (because they will likely have to deploy/train to deploy), you lose your funding (because it expires or gets taken away for not spending it when you were supposed to, and you have to go explain to a myriad of oversight bodies what your problems are … and the multiple differing bodies do not agree … and want updated documentation (CBA, EA, ROI, net present value).*
>
> *Bottom line is that you can't stop to take the time to improve quality … you go live with what you have, you skip testing, you field the poor-quality system … and you try to fix it after go live in sustainment.*
>
> *Additionally, you are incentivized to get into sustainment (post MS C) on ACAT 1D programs to remove the constraint of OSD oversight … to try and get MDA delegated to the component.*

On the topic of system relationship to software:

> *This discussion (and virtually all Agile methodology discussions) fails to address the system design/architecture analysis phases, which identify and specify SW components that need to be built (or acquired) from the actual SW development methodology used to actually develop the one to many software components (or products). For large systems, there may be several layers of architectural analysis performed (system to segment, segment to element, element to component) before the SW components are identified and specified. The specification of a software component would be identifying the stories and epics for that component. As soon as there is an initial set of stories that are "good enough" (this is the product backlog), then the iterative/Agile SW component development can begin. All of the continued learning and updates to the "system," "segment," "element" engineering may result in updates being made to the component (or product) backlog over time. The rhythm of system/segment/element engineering does not have to be the same as the rhythm of the SW component Agile development.*
>
> *For large systems (such as those often acquired by the DoD) this is a very important discussion and distinction.*

On the topic of Agile methods and each Service:

> *There should be an organization in each Service that has competency in these things … and repeatable processes. That is NOT subject to the existing 5000. Where the software PMs have authority sufficient for responsibility.*

Things the organization will need:

- *dedicated architects that are RESPONSIBLE to the PM for what gets built (not above the PM) drawing cartoon compliance architectures*
- *dedicated lawyers*
- *dedicated KOs for each program*
- *an infrastructure team that is RESPONSIBLE for actual implementation of ALL programs (not one). This means they are responsible for its stand-up, not just its oversight*
- *dedicated functional experts; including FINANCE people that understand the proprietary government rules (requirements) that have to be built into all software*
- *dedicated cost estimators that do it for ALL programs the same way, with the same tools and cost elements*

On the topic of DoD culture of guidance that inhibits common sense:

With regard to enterprise software buys:

Seven programs with licenses bought from Company "X" with strings attached. Users of all seven systems require seven different licenses for the same product (e.g., a user of these seven systems has seven licenses for the SAME PRODUCT being used in seven different system stacks).

Proposed Solution: Enterprise license that allows access to all seven systems with one license … based on a package of Company "X" products that gives every program what they need … and then some … for 50% less.

The programs cannot mix funding (different colors, appropriated vs. non-appropriated) and it is against the law to "augment the funding of one program with that of another"… it gets stupidly hard … PMs give up … and the government spends lots more money.

On the topic of trying to use rolling wave planning as an Agile support:

How does this fit with the requirements for 25-year lifecycle costs, independently validated by the Service and DoD … and EVM attached to an approved baseline and APB that must not change by more than 10%?

Don't forget that you must know the different colors of money … so, you have to "guess" what color you need before you know what you really need (can DISA provide that unique Teradata hardware/software as a managed service … or do you have to buy it as a capital investment and just have DISA run it?) … color of money matters.…

… and there is a $250k limit on using OPS money for "development" … or "fixes"

… and anything more than $1M for a "business system" must go to the DOD Investment Review Board (IRB) … after it goes through the component and DCMO … and functional proponent.

... and trading partners of that business domain system are in the "warfighting domain" ... with different governance.

A labyrinth of bureaucracy.

On the topic of cultural themes supporting Agile methods:

If I were to pick a third theme, it would be teamwork. Agile is all about working as a team, including the customer as a valued team member. That change from us versus them to all of us working together to solve the customer's problem is a much bigger deal. And it may be one of the biggest hurdles that DoD has to overcome to transition to Agile. We have to put trust and goodwill back into the contractual relationship.

On the topic of embracing change and its importance in an Agile culture:

Embracing change is also the reason that Agile teams build quality SW. They know that if future changes are to be cost effective, the SW has to be clean. This is a fundamental difference between Agile and traditional. With traditional approaches, you often try to enable future change with infrastructures, hooks, etc. Agile takes a fundamentally different approach. Rather than anticipate what will change, they try to build SW that is amenable to change—few dependencies, clear code, safety net for future changes, etc.

On the topic of the importance of adopting Agile values, not just Agile practices:

Agile methods are not an option, they are what we do, how we behave. Having to justify Agile principles on every program basically keeps us from every actually becoming Agile; we simply act Agile from time to time. This is not where we want to be.

On the topic of the challenges in changing DoD culture to accommodate appropriate use of Agile methods:

Changing a culture is a large-scale change effort that will take time to implement. It is not critical that the culture be changed in order to start adopting Agile methods, but it is for them to be successful over the long term.

On the topic of embracing change:

The current DoD processes are change resistant—plan driven—and I think it is going to be a major mental/cultural shift to embrace change.

Trust and willingness to make mistakes are two more biggies with Agile. These are two things that (in my experience) are not encouraged or tolerated in many DoD projects.

Mastery comes with experience and feedback that a process works. Trust is built over time, so it is critical to start taking baby Agile steps and not wait until you have the culture fixed first.

On the topic of "doing Agile" versus "being Agile":

One of the first things that I tell people when I am coaching them about Agile is that Agile is not just something that you do by following a process, but it's something that requires you to think differently. This is where a lot of teams fail using Agile.

Appendix I: Selected Agile Resources

The Program Management Institute has started an Agile certification program. The following is the institute's current (as of publication of this TN) list of references:

Agile Retrospectives: Making Good Teams Great
Esther Derby, Diana Larsen, Ken Schwaber
ISBN #0977616649

Agile Software Development: The Cooperative Game – Second Edition
Alistair Cockburn
ISBN #03214827

The Software Project Manager's Bridge to Agility
Michele Sliger, Stacia Broderick
ISBN #0321502752

Coaching Agile Teams
Lyssa Adkins
ISBN #0321637704

Agile Project Management: Creating Innovative Products – Second Edition
Jim Highsmith
ISBN #0321658396

Becoming Agile: ...In an Imperfect World
Greg Smith, Ahmed Sidky
ISBN #1933988258

Agile Estimating and Planning
Mike Cohn
ISBN #0131479415

The Art of Agile Development
James Shore
ISBN #0596527675

User Stories Applied: For Agile Software Development
Mike Cohn
ISBN #0321205685

Agile Project Management with Scrum
Ken Schwaber
ISBN #073561993X

Lean-Agile Software Development: Achieving Enterprise Agility
Alan Shalloway, Guy Beaver, James R. Trott
ISBN #0321532899

See www.pmi.org/Certifcation/New-PMI-Agile-Certifcation.aspx for updated information.

References

[Adler 1990]
Adler, P. S. & Shenhar, A. "Adapting Your Technological Base: The Organizational Challenge." *Sloan Management Review 32*, 1 (1990): 25-37.

[Agile Alliance 2001]
Agile Alliance. *History: The Agile Manifesto*. http://agilemanifesto.org/history.html (2001).

[Ambler 2004]
Ambler, Scott W. *Disciplined Agile Software Development: Definition.* http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm (2004).

[Bhalareo 2009]
Bhalareo, S. "Incorporating Vital Factors in Agile Estimation through Algorithmic Method," *International Journal of Computer Science and Applications 6*, 1 (2009) 85-97.

[Boehm 1981]
Boehm, B. *Software Engineering Economics*, Prentice Hall, 1981.

[Boxer 2009]
Boxer, P. & Garcia, S. *Limits to the Use of the Zachman Framework in Developing and Evolving Architectures for Complex Systems of Systems*. SATURN Conference, May 2009. Software Engineering Institute, 2009. http://www.sei.cmu.edu/saturn/2009/images/Limit_use_Zachman_Framework.pdf

[CASISP 2010]
Committee for Advancing Software-Intensive Systems Producibility, National Research Council. *Critical Code: Software Producibility for Defense*. The National Academies Press, 2010.

[CIPP 2010]
Committee on Improving Processes and Policies for the Acquisition and Test of Information Technologies in the Department of Defense, National Research Council. *Achieving Effective Acquisition of Information Technology in the Department of Defense.* The National Academies Press, 2010.

[Cockburn 2007]
Cockburn, A. *Agile Software Development: The Cooperative Game*, 2nd ed. Addison-Wesley, 2007.

[Cohn 2006]
Cohn, M. *Agile Estimating and Planning*. Addison-Wesley, 2006.

[Cohn 2008]
Cohn, M. "When Should We Estimate the Product Backlog." *Mike Cohn's Blog – Succeeding with Agile* (March 16, 2008). http://blog.mountaingoatsoftware.com/when-should-we-estimate-the-product-backlog

[Conner 1983]
Conner, D. & Patterson, R. "Building Commitment to Organizational Change." *Training and Development Journal* (April 1983):18-30.

[CSSE 2011]
Center for Systems and Software Engineering, University of Southern California. *Agile COCOMO II*. http://csse.usc.edu/csse/research/AgileCOCOMO/ (2011).

[Defense Acquisition University 2011a]
Defense Acquisition University. *Defense Acquisition Guidebook*, July 2011. Defense Acquisition University, 2011.

[Defense Acquisition University 2011b]
Defense Acquisition University. "Foreword." *Defense Acquisition Guidebook.* Defense Acquisition University, 2011. https://acc.dau.mil/CommunityBrowser.aspx?id=314709. Accessed July 13, 2011.

[Defense Acquisition University 2011c]
Defense Acquisition University. Ch. 4.3.6, "Evolutionary Acquisition Programs," 280-281. *Defense Acquisition Guidebook.* http://at.dod.mil/docs/DefenseAcquisitionGuidebook.pdf. Accessed Sep 12, 2011.

[Defense Acquisition University 2011d]
Defense Acquisition University. Ch. 3.1.2, "Life-Cycle Cost Categories and Program Phases," *Defense Acquisition Guidebook.* https://acc.dau.mil/CommunityBrowser.aspx?id=314767#3.1.2. Accessed July 13, 2011.

[Defense Acquisition University 2011e]
Defense Acquisition University. *Welcome to the Defense Acquisition Guidebook Home Page.* https://acc.dau.mil/CommunityBrowser.aspx?id=289207&lang=en-US. Accessed July 13, 2011.

[Defense Science Board 2009]
Defense Science Board. *Report of the Defense Science Board Task Force on Department of Defense Policies and Procedures for the Acquisition of Information Technology.* Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, 2009.

[Department of Defense 2011]
Department of Defense. Section 2.5.2.4.1, "Guidance," 57. *Earned Value Management Implementation Guide.* Department of Defense, 2006. https://acc.dau.mil/CommunityBrowser.aspx?id=386074&lang=en-US. Accessed July 13, 2011.

[Elssamadisy 2009]
Elssamadisy, A. *Agile Adoption Patterns: A Roadmap to Organizational Success.* Pearson Education, Inc., 2009.

[GAO 2009]
Government Accountability Office. *GAO Cost Estimating and Assessment Guide: Best Practices for Developing and Managing Capital Program Costs* (GAO-09-3SP). United States Government Accountability Office, 2009.

[Garcia 2006]
Garcia, S. & Turner, R. *CMMI Survival Guide: Just Enough Process Improvement*. Addison-Wesley, 2006.

[Gates 2008]
Gates, R. M. *Speech to National Defense University (Washington, D.C.) Monday, September 29, 2008.* http://www.defense.gov/Speeches/Speech.aspx?SpeechID=1279 (2008). Accessed July 13, 2011.

[Highsmith 2009]
Highsmith, J. *Agile Project Management: Creating Innovative Products,* 2nd ed. Addison-Wesley, 2009.

[House Armed Services Committee 2010]
House Armed Services Committee. *House Armed Services Committee Panel on Defense Acquisition Reform Findings and Recommendations* (DAR Final Report [3-23-2010]). United States House of Representatives, 2010.

[Jones 1995]
Jones, C. *Patterns of Software System Failure and Success.* International Thompson Computer Press, 1995.

[Kovatch 2009]
Kovatch, D. *Roles & Responsibility of the Product Owner.* Scrum Gathering, Orlando, FL, 2009. Scrum Alliance, 2009. http://www.scrumalliance.org/resources/617

[Lapham 2010]
Lapham, M.A.; Williams, R.; Hammons, C.; Burton, D.; & Schenker, A. *Considerations for Using Agile in DoD Acquisition* (CMU/SEI-2010-TN-002). Software Engineering Institute, Carnegie Mellon University, 2010. http://www.sei.cmu.edu/library/abstracts/reports/10tn002.cfm

[Larman 2004]
Larman, C. *Agile and Iterative Development: A Manager's Guide*. Addison-Wesley, 2004.

[Leffingwell 2008]
Leffingwell, D. *Scaling Software Agility*. Addison-Wesley, 2008.

[Moore 2002]
Moore, G. *Crossing the Chasm: Marketing and Selling Disruptive Products to Mainstream Customers*. Harper Business Essentials, 2002.

[OSD 2010]
Office of the Secretary of Defense. *A New Approach for Delivering Information Technology Capabilities in the Department of Defense, Report to Congress,* November 2010, Pursuant to Section 804 of the National Defense Authorization Act for Fiscal Year 2010. United States Department of Defense, 2010. http://dcmo.defense.gov/documents/OSD%2013744-10%20-%20804%20Report%20to%20Congress%20.pdf

[Ozkaya 2011]
Ozkaya, I.; Brown, N.; & Nord, R. Ch. 3, "Communicating the Value of Architecting within Agile Development," 11-22. *Results of SEI Independent Research and Development Projects (FY 2010)* (CMU/SEI-2011-TR-002). Software Engineering Institute, Carnegie Mellon University, 2011. http://www.sei.cmu.edu/library/abstracts/reports/11tr002.cfm

[Poppendieck 2003]
Poppendieck, M. & Poppendieck, T. *Lean Software Development.* Addison-Wesley, 2003.

[Project Management Institute 2008]
Project Management Institute. *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*, 4th ed. Project Management Institute, 2008.

[Rally Software 2010]
Rally Software. *Agile Adoption Trends and Their Implications for Your Company.* Rally Software, Inc., 2010. http://www.slideshare.net/rallysoftware/agile-adoption-trends. Accessed July 13, 2011.

[Rawsthorne 2010]
Rawsthorne, D. *Agile Release Planning and Monitoring*. CollabNet, 2010. http://www.open.collab.net/media/pdfs/SBU_ReleasePlanning.pdf. Accessed July 13, 2011.

[Rogers 2003]
Rogers, E. *Diffusion of Innovation,* 5th ed. Simon & Schuster, 2003.

[Ruhe 2009]
Ruhe, Guenther. *Product Release Planning: Methods, Tools and Applications*. Auerbach Publishing, 2009.

[Schein 2009]
Schein, Edgar H. *Organizational Culture and Leadership,* 3rd ed. Jossey-Bass Publishers, 2004.

[Schenker 2007]
Schenker, F. & Jacobs, R. *Project Management by Functional Capability.* Presented at the 7th Annual CMMI Technology Conference and User Group, Denver, CO, November 2007. www.dtic.mil/ndia/2007cmmi/Thursday/3amSchenker.pdf

[SEER-SEM 2011]
"SEER-SEM." http://en.wikipedia.org/wiki/SEER-SEM. Accessed July 13, 2011.

[Shalloway 2009]
Shalloway, A.; Beaver, G.; & Trott, J. R. *Lean-Agile Software Development: Achieving Enterprise Agility*. Addison-Wesley, 2009.

[Sidky 2009]
Sidky, A. & Smith, G. *Becoming Agile in an Imperfect World*. Manning Publications Co., 2009.

[Stutzke 2005]
Stutzke, R. *Estimating Software-Intensive Systems: Projects, Products, and Processes.* Addison-Wesley, 2005.

[Sulaiman 2006]
Sulaiman, T; Barton, B.; & Blackburn, T. "AgileEVM – Earned Value Management in Scrum Projects," 10-16. *AGILE '06 Proceedings of the Conference on AGILE 2006*. Minneapolis, MN, July 2006. IEEE Computer Society, 2006.

[Treacy 1995]
Treacy, M. & Wiersema, F. *The Discipline of Market Leaders: Choose Your Customers, Narrow Your Focus, Dominate Your Market*. Perseus Books, 1995.

[Weinberg 1997]
Weinberg, G. *Quality Software Management: Anticipating Change*. Dorset House, 1997.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	October 2011	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Agile Methods: Selected DoD Management and Acquisition Concerns	FA8721-05-C-0003

6. AUTHOR(S)

Mary Ann Lapham, Suzanne Miller, Lorraine Adams, Nanette Brown, Bart Hackemack, Charles (Bud) Hammons, PhD, Linda Levine, PhD, Alfred Schenker

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2011-TN-002

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	CMU/SEI-2011-TN-002

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

This technical note (TN), the second in an SEI series on Agile in the DoD, addresses some of the key issues that either must be understood to ease the adoption of Agile or are seen as potential barriers to adoption of Agile in the DoD acquisition context. These topics were introduced in the first TN of the series, CMU/SEI-2010-TN-002. For this TN, the SEI gathered more data from users of Agile methods in the DoD and delved deeper into the existing body of knowledge about Agile before addressing them. Topics considered here include: why DoD is interested in Agile methods; what it means to be Agile in the DoD; managing and contracting for Agile programs; technical milestone reviews in a DoD Agile acquisition context; estimating in a DoD Agile acquisition context; and moving toward adopting Agile practices. The authors hope that this report continues to stimulate discussion about and appropriate adoption of Agile in the DoD and federal agencies.

14. SUBJECT TERMS	15. NUMBER OF PAGES
agile methods, agile acquisition	119

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL