Software Engineering Institute
Carnegie Mellon University

Update 2016: Considerations for Using Agile in DoD Acquisition

Suzanne Miller
Dan Ward

Contributing Authors, 2010 Edition:
Mary Ann Lapham
Ray Williams
Charles (Bud) Hammons
Daniel Burton
Alfred Schenker

December 2016

TECHNICAL NOTE
CMU/SEI-2016-TN-001

Client Technical Solutions Division

http://www.sei.cmu.edu

This material is based upon work funded and supported by the 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
AFLCMC/PZM
20 Schilling Circle, Bldg 1305, 3rd floor
Hanscom AFB, MA 01731-2125

DM-0003307

Table of Contents

List of Figures

List of Tables

Acknowledgments

For this update, we would like to thank all those who have contributed to our knowledge of applying “other than traditional” methods for software system acquisition and management over the last six years. For acknowledgments of the interviewees and reviewers from the 2010 version, please see that version of the report. Many of the people mentioned there have continued to contribute to our understanding of Agile and Lean methods in government, particularly DoD, settings.

In addition, we would like to thank Mary Ann Lapham for her continuing support of this project, and Peter Capell, who took over some of the finalization when customer work became a bit hectic for the primary authors and threatened to delay publication.

Executive Summary

In 2009 the SEI was tasked by Mr. Blaise Durante, then Air Force Deputy Assistant Secretary for Acquisition Integration (SAF/AQX), to assess the state of the practice of Agile development in government software acquisitions. A team was assembled to complete that assessment, and the findings were published in 2010. Six years later, a small team set out to update the findings and incorporate new developments, both within the wider Agile community and the Department of Defense (DoD) itself.

The initial report aimed to debunk the prevalent myth that Agile and DoD practices are unworkable. The past six years have provided substantial support to this debunking, as new projects have successfully implemented Agile and updated policies have further embraced the principles and practices of Agile.

The audience for this report is

- 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
- 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
- software development contractors who are contemplating responding to a DoD request for proposal (RFP) with a proposal based on using Agile software development methods

Agile and the DoD

The formal definition of Agile was established in 2001 in the Agile Manifesto, but it was based on concepts, principles, and practices that had been around for decades [Beck et al. 2001]. Agile achieved its greatest initial success in small- to mid-sized commercial applications, but larger enterprises are actively incorporating Agile practices into their businesses. Federal agencies, including the DoD, have generally been slow to adopt Agile for a number of reasons, but in recent years a growing number of projects have begun to use Agile methods. Recently, even ACAT 1 (the class of large acquisitions that requires the most oversight from the Office of the Secretary of Defense [OSD]) programs have been seen incorporating Agile methods into their software development and Lean engineering methods that help to scale Agile beyond small software teams into their systems engineering approaches.

At the time the 2010 technical note was published, some DoD contractors had already started to build internal Agile capabilities and use Agile on DoD programs. Some DoD acquisition programs proposed and used Agile processes, attempting to take advantage of contractor capabilities, but without (as yet) any formal DoD guidance, templates, or best practices. In the ensuing six years, the cadre of Agile adopters has expanded. We have seen publications and presentations from all of the major DoD development contractors on their Agile approaches to government contracts.

Given this backdrop, it is increasingly clear that Agile can produce a better product developed within cost and schedule parameters when both the government and the development side understand how to effectively employ Agile principles and practices. Some barriers to widespread adoption across the DoD still remain, but they have become less substantial over the past six years. Our assessment of whether Agile can benefit the DoD resulted in a resounding, but qualified, "Yes." Agile *is* another tool that can provide both tactical and strategic benefits. The tactical benefits of lowering cost, being on schedule, and increasing quality are important; however, the strategic benefits of being responsive to operational needs and being able to adjust to the current situation more rapidly might be of even greater value. This could be a huge factor in today's defense acquisition world, where the DoD must get results faster and be better aligned with changing needs. In fact, reports[1] available about Agile are impressive. Even if experience provides savings for DoD programs on the low end of the spectrum, these savings can be significant over time. We also found that there are no prohibitions for using Agile in the DoD 5000 series.

The IEEE has published various standards that contain guidance on things such as producing documentation in Agile environments, as well as other topics of interest, and has other standards in work that contain Agile-related content.[2] The IEEE has also hosted an Agile Development Conference every year since 2010, with topics ranging from distributed teams to the role of trust, from automated test suites to system dynamics models. Conference proceedings are available in IEEE's digital library, Xplore. In parallel, other researchers have offered recommendations on how to apply and implement existing IEEE standards within Agile projects.[3]

During our research, we noted that in the current, traditional Waterfall method commonly employed within the DoD, there is an established practice that uses some form of controlled experimentation. Current Waterfall practices create experimental code or prototypes and then throw them away. Some adaptation of those practices to create evolutionary prototypes that are sufficiently robust for testing and deployment is a step towards Agile principles. Agile builds working, deployable software iteratively, refining or discarding portions as required to create increments of the product. The idea is to have some working code at the end of each iteration that "could" be deployed. We say *could* because the code available early in the program could be deployed to a sandbox type of area but will not have met the rigorous operational testing and cybersecurity activities that typically are required before a product can actually be fielded.

Embracing Agile Methods

Agile practices and principles are based upon good ideas derived from successful industry practices. We believe the DoD should embrace Agile for some programs and traditional Waterfall

---

1 Several results show that by using Agile methods, costs decrease from as little as 5 to as much as 61 percent, with schedule decreasing from as little as 24 to as much as 58 percent, and cumulative defects decreasing from as little as 11 to as much as 83 percent [Rico et al. 2009].

2 See *IEEE P1648: Working Group for Establishing and Managing Software Development Efforts Using Agile Methods* (https://standards.ieee.org/develop/wg/1648_WG.html) and *IEEE 26515-2012: Developing User Documentation in an Agile Environment* (https://standards.ieee.org/develop/wg/P26515.html).

3 For example, the paper *IEEE Std. 829- 2008 and Agile Process: Can They Work Together?* looks at IEEE's Software and Systems Test Documentation standard (http://worldcomp-proceedings.com/proc/p2013/SER3559.pdf).

methods for others, although policy and experience alike suggest the default should shift away from Waterfall and towards Agile. There is no "one size fits all" Agile process. Just like any set of practices, implementation of Agile must be tailored to fit the situation and context. For example, Agile teams responsible for developing high-risk core components of the software architecture might apply less-aggressive release schedules than Agile teams developing less critical pieces of the software system. Some Agile teams might pick a two-week iteration cycle where others might determine their optimum iteration cycle is three weeks. Agile is not a silver bullet but rather another "lead bullet" for the Program Management Office's (PMO's) and contractor's arsenal.

Sometimes a composite approach is the best for a particular program. This report examines a composite approach known as the Scaled Agile Framework (SAFe, which, anecdotally, seems to be the most widely adopted scaling framework for DoD contractors).

Moving to Agile requires considerable upfront work from the DoD entity (PMO, DoD, OSD, and perhaps Congress), and is not without hurdles, most notably acquisition lifecycle, team environment, end-user access, training and coaching, oversight, rewards and incentives, PMO team composition, and culture. We explore each topic below.

Acquisition Lifecycle

Each lifecycle phase (e.g., Materiel Solution Analysis, Technology Development, Engineering and Manufacturing Development, Production and Deployment, and Operations and Support) presents unique challenges and opportunities. Some phases lend themselves to the use of Agile better than others. You must consider the Agile processes and practices you want to use early in the acquisition lifecycle; it is of critical importance to make sure that contractually binding documents, such as RFPs and statements of work (SOWs) support those processes and practices. For example, if you embrace Agile, you need to determine how you will meet the standard milestone criteria such as Preliminary Design Review (PDR) and Critical Design Review (CDR). Typically, the types of documentation expected at these milestone events are not produced using Agile. Thus, you should create expectations and criteria that reflect the level and type of documentation that would be acceptable for those milestones and yet work within the Agile constraints. See the SEI report *Agile Methods and Requests for Change Observations from DoD Acquisition Programs* for particular insights on managing technical reviews in Agile DoD settings [Lapham et al. 2014].

Team Environment

A central concept to Agile is the small, dynamic, high-performing team. The challenge is this: *How do I provide an environment that fosters the creation of self-organizing teams in a culture that is accustomed to static, centralized organizational structures?* To complicate this further, consider that the software team might be a small part of an overall system procurement for something like a tank, ship, or plane. The system environment might call for centralized configuration management, highly defined legacy interfaces, and a predetermined architecture, all of which constrain the software. This environment, then, should be treated as a constraint by the Agile team and can provide boundaries within which the Agile team can operate. These system boundaries could act to encapsulate the Agile team environment.

End-User Access

Access to end users can be complex and difficult when dealing with any single service but it can be even more complex with multi-service programs. Agile developers need to have a single voice for the user and one that can commit to changes for the product being developed. In some Agile approaches, the "one voice" is a product owner or manager who brings decisions to the Agile team that have been made through collaborative interaction. Within the DoD, the acquisition organization typically is the proxy for the end users and only duly warranted personnel can commit the program. To mitigate these issues, end users should be invited to all demos where they can provide feedback that only becomes binding with PMO approval. The end users need to work closely with the PMO, as with any acquisition. Lack of relevant end-user interaction is one of the failure modes that we have seen that significantly reduces the effectiveness of an Agile approach.

Training and Coaching

While Agile concepts may not be new, the subtleties and nuances of each Agile method can be new to the uninformed PMO. To overcome this, train the PMO staff before starting and employ an experienced coach or knowledgeable advocate for the PMO to help guide them throughout the process. It is important to set funding for initial and ongoing training and support.

Oversight

Traditional methods have well-defined oversight methods. Agile oversight methods are less defined and require more flexible oversight to accommodate the fluidity of Agile implementation. The specific type of oversight needs to be resolved in advance. One aspect of the Agile management philosophy is that the primary role of *manager* is more of a team advocate than overseer. The management function of *roadblock remover* is critical to the success of an Agile team. Thought needs to be given to what day-to-day PMO activities might need to be altered to support this type of change. In addition, PMOs accustomed to large batch reviews of documentation and other program artifacts may not be prepared for the small batch, but more frequent, reviews typical of Agile programs. Adjusting to small batch oversight is a challenge, but it enables learning and course correction that is essential to a successful Agile endeavor.

Typically, projective documentation[4] is used by the PMO throughout the development cycle to monitor the progress of the contractor. Documentation produced using Agile methods is typically focused on as-built documents, and is just enough to meet the need and provide continuity for the team and for eventual sustainment of the system. This type of documentation is usually not sufficient for traditional capstone reviews. The PMO needs to create different ways to meet the same objectives for monitoring while leveraging the advantages of Agile.

Rewards and Incentives

Agile rewards and incentives are different from the typical structure of traditional methods. In the DoD environment, the challenge is finding ways to incentivize teams and individuals to support Agile goals such as innovation, rapid software delivery, and customer satisfaction. At the same

[4] Projective documentation is documentation that projects the intent of the author (e.g., requirements and design documents written before implementation to project a solution and its implementation).

time, we need to eliminate rewards that incentivize the wrong things. For example, rather than rewarding contractors for fixing defects, we may want to reward the developer for early delivery of beta software to a limited set of users in a constrained environment. This way the beta users get to test the product in the field sooner while at the same time providing feedback that helps to improve the quality of that iteration of the software. One other type of incentive that should be considered one that encourages a collaborative relationship between the PMO and the contractor's staff. We are starting to see more sophisticated incentive programs for Agile that involve incentives for meeting iteration goals at a certain level of quality and incentives for meeting the highest priority goals earlier in the development cycle.

PMO Team Composition

The composition of the PMO staff might look somewhat different to accommodate the use of Agile. The government should consider adding a knowledgeable Agile advocate or experienced coach to their team. End-user representatives are essential for Agile. These positions are difficult to fill in a timely and consistent manner. Some programs have used rotating personnel to fill these positions.

Another challenge is keeping high-performing Agile development teams together long enough for them to achieve peak performance. This is a challenge because developers change at the end of a contractual period of performance. One recommendation might be to look at putting key Agile technical developers or technical leads under a separate contract vehicle or hire them to work for the government organization itself.

Culture

The overall organizational culture needs to support the Agile principles that are in use. The Agile culture is counter to the traditional Waterfall culture in many areas, from oversight and team structure to end-user interaction throughout development. This will require a mindset change for the PMO and other government entities such as OSD. To employ any of the Agile concepts, the DoD organization will have to plan for them, anticipate the changes needed in its environment and business model, and apply the hard work to make the changes a reality. Organizational change management is essential during the transition to Agile. One exercise we perform when teaching Agile to government clients is to go through the 12 Agile principles and answer the questions "Which are already compatible with current DoD acquisition practice?" and "Which are not already compatible and will take work to change?" The typical class cites not more than three of the principles being "already compatible" with typical DoD acquisition practice.

Organization of This Report

This report is organized as follows:

Executive Summary (page iv)	contains highlights of this report, specifically Agile methods and their use in the DoD
Section 1, Overview (page 1)	describes the approach taken to develop this report, what is included, and what is excluded
Section 2, What Is Agile? (page 4)	provides a definition/history of Agile, a generic Agile example, and a short comparison of Agile to Waterfall
Section 3, Interview Observations and Findings (page 12)	presents the results from interviewing specific programs and includes pitfalls, issues, and potential solutions
Section 4, DoD 5000 Series and Agile—Potential Issues and Conflicts (page 27)	details the results of an analysis of the DoD 5000 Series (January 2015 version) and how it impacts Agile use within the DoD
Section 5, Considerations for Applying Agile in the DoD (page 37)	provides multiple considerations for applying Agile in the DoD; also includes discussions of various Agile concepts and addresses potential hurdles for implementing them within DoD
Section 6, Conclusion (page 48)	contains a summary of this report and suggestions for future research on using Agile within the DoD
Appendix A (page 50)	provides a variety of Agile methods and their definitions
Appendix B (page 53)	identifies and debunks common objections to using Agile
Appendix C (page 56)	details areas of concern and their considerations for using Agile in the DoD
Appendix D (page 59)	defines acronyms used throughout this report
Appendix E (page 61)	contains the text of the FIST Manifesto
Appendix F (page 63)	provides a collection of excerpts from the FAR and 2015 version of DoD 5000.02 thatsupport Agile practices
Bibliography (page 72)	contains a list of works referenced in this report

Abstract

This report, an update to a 2010 report, *Considerations For Using Agile In DoD Acquisitions*, addresses developments in commercial Agile practices as well as the Department of Defense (DoD) acquisition environment. It covers some previously unanswered questions and asks some new ones. It includes new research, examples of Agile in practice, and policy updates.

Continuing with the 2010 report's theme, this report updates the exploration of these questions: *Can Agile be used in the DoD environment? If so, how?* It includes lessons learned from DoD programs that have employed Agile and information gleaned from myriad articles and books on Agile. While this report does not pretend to cover every paper or thought published about Agile in the DoD world, it provides an updated overview of some challenges in using Agile, an overview of how some programs have addressed these challenges, and some additional recommendations on dealing with these challenges. The intended audience is policy makers, program office staff, and software development contractors who are contemplating proposing the use of Agile methods.

We hope this report stimulates new discussion about adopting Agile in the DoD world and equips practitioners with the information they need to make informed decisions.

1 Overview

Agile methods for software development have existed for many years. These methods have achieved their greatest success in small- to medium-sized commercial applications, although the commercial space is also finding the need to scale those methods to meet the needs of larger projects. Federal agencies, including the DoD, have been slower than the commercial sector to adopt Agile, but a growing number of projects are beginning to use Agile methods.

At the time the 2010 report was published, some DoD contractors had already begun to build internal Agile capabilities and initiate pilot usage efforts on DoD programs. Many DoD acquisition programs had also begun to propose and use Agile processes, attempting to take advantage of contractor capabilities; however, they did this without any templates, best practices, or formal DoD guidance. In the ensuing six years, the cadre of Agile adopters has expanded. We are observing anecdotally that all major defense contractors have Agile approaches to development that they routinely propose to government RFPs that are supportive of Agile in some form.

In 2009, the Carnegie Mellon® Software Engineering Institute (SEI) was tasked by Mr. Blaise Durante, then the SAF/AQX (Air Force Deputy Assistant Secretary for Acquisition Integration), to assess the state of the practice of Agile development in government software acquisitions. The initial team was assembled to complete that assessment.

This report updates the results of the 2009 SEI study of the utilization and future applicability of Agile for software development in DoD acquisitions. The original study was conducted in the latter half of 2009 and completed in January 2010. This updated report, like the original, is intended for

- 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
- 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
- software development contractors who are contemplating responding to a DoD request for proposal (RFP) with a proposal based on using Agile software development methods

1.1 Original Tasking and Approach

Our initial SEI team set out to document lessons learned and/or best practices in as many programs as we could find in the DoD acquisition community that were using or had used Agile for software development. For the original report, we made the assumption that we were dealing with software-only or software-intensive systems. Our purpose was to answer two questions:

- Is the use of Agile beneficial to the DoD; that is, can it produce a better end product developed within cost and schedule parameters?
- If the answer to the above question is "Yes," what are the barriers to using Agile in the DoD acquisition environment, and how might these barriers be addressed?

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

Our approach was to address both questions simultaneously because we believed that, regardless of any of the *theoretical* benefits of Agile (and it was quickly evident that there were many), it would only remain an academic interest if there was not solid experience available on the *actual use* of Agile within the DoD acquisition environment. Thus, we looked for current and recent DoD software development acquisitions that claimed to have used one or more of the methodologies generally accepted as Agile (eXtreme Programming, Scrum, Lean Software Development, and others).[5] At the time of the original report, we found only a small number of such programs willing to share their experiences; these were weapons systems programs including Joint Mission Planning System (JMPS), Single Integrated Air Picture (SIAP), Operationally Responsive Space (ORS), Virtual Mission Operations Center (VMOC), Space Radar, an Army tank program, and some other classified programs. The programs ranged in size from small to fairly large (based on budget). The amount of detail we were able to obtain from each program was a function of program constraints such as security. With this in mind, we conducted interviews with development team members, program office personnel, or other members of the SEI staff who had intimate knowledge of those programs to do the following:

- document lessons learned and/or best practices from the case study of the multiple programs using Agile, to include contractor capabilities, government management (strengths and weaknesses), and conflict points with standard DoD methods, etc.
- examine the viability of developing an approach that can be used within the DoD 5000 acquisition environment to take advantage of the benefits of using Agile methodologies with minimal need for policy waivers
- provide guidelines on how DoD technical milestone reviews (SSR, PDR, and Critical Design Review [CDR]) should be altered/augmented to account for Agile software development practices

After our interviews, we prepared an *Agile Study Lessons Learned* draft presentation and sent it to the interviewees for comment. We did this to assure that we understood the points that interviewees made and to lay out an initial argument addressing the primary questions. We incorporated the comments we received resulting in a final, annotated version of the presentation, which was given limited distribution. Finally, we created the report to document our findings and provide recommendations.

This report, an update to the 2009 study, is based on the knowledge gained from the several reports the SEI has published on Agile and related topics since 2009. Many of the topics in this report are treated in more depth in follow-on publications. We have kept the same general in/out-of-scope boundary so as not to overly bloat the update.

1.2 What Is Not Addressed

"Agile," in the context of software development, is a term that encompasses many different tools, techniques, and methods, all based on a single set of four tenets and 12 principles. Some of these methods are briefly described in Section 2. We give the reader context and awareness of Agile, but do not attempt to provide a comprehensive review or tutorial of specific Agile methodologies for application in software acquisitions. Rather, we provide questions to ask and guidance on how Agile could be useful and have relevance to DoD organizations.

[5] Appendix A contains several examples of Agile methods.

We have not attempted to address the question of whether DoD PMOs themselves could become "Agile" in their own internal operations. At the time of the original report, we decided that such a discussion would go too far beyond the current experiences of the interviewees. Since then, we have seen examples of Agile and Lean principles being applied to program office acquisition tasks. However, we chose to retain our Agile/PMO boundary in the update.

We have also not attempted to discuss the relationship between CMMI and Agile software development methodologies (a frequent question in the 2009 timeframe of the original study). In our view, CMMI is a framework of best practices that can be applied in any software development program (whether or not that program uses Agile). We recognize that some Agile advocates have equated CMMI with "traditional" and/or "Waterfall" software development approaches [Rico et al. 2009], but we believe that this is only a misunderstanding of CMMI on their part. Others have suggested that Agile and CMMI should be embraced together [Glazer et al. 2008]. There is a good body of knowledge and consulting practice available to those program offices who need to use Agile and CMMI together. One note—with the publication and use of CMMI-SVC in commercial IT settings, it may be useful to consider that model when approaching Agile implementation.

2016 update: In December 2012, at the request of OSD, the CMMI product suite and project were moved from the Software Engineering Institute to the CMMI Institute. For current information on CMMI-DEV, CMMI-SVC, or CMMI-ACQ, please refer to the CMMI Institute website (www.cmmiinstitute.com).

2 What Is Agile?

On the surface it seems that there is really nothing "new" about Agile. However, on close inspection, there are new components (ideas, practices, theories, etc.) and new combinations of those new components with "old" components. The explicit value statements (expressed via the Agile Manifesto tenets and principles) used within Agile are also new. However, in a way, Agile has simply swept up software development practices that have been used since the earliest days of software and added a few new twists, then consolidated them into a unified framework of principles. Philosophically, Agile also borrows heavily from approaches that have been successfully used in manufacturing throughout the world for decades, such as "just-in-time," Lean, Kanban, and work-flow-based planning. Another new development is that Agile is becoming codified, evolving from a collection of disjointed, separately created software development methods into philosophically coherent families of such methods.

This philosophical coherence—and the current energy driving advocacy of Agile—was the result of a remarkable meeting among thought leaders and consultants[6] in software development who would normally have been competitors. In February 2001, 17 people met to try to find common ground and ultimately produced the *Agile Software Development Manifesto* [Beck et al. 2001]. This manifesto detailed all of their commonalities—overlooking, for the moment, areas where they had differences of opinion.

2.1 Agile Manifesto and Principles—A Brief History

The self-named *Agile Alliance* shared allegiance to a set of compatible values promoting organizational models based on people, collaboration, and building organizational communities compatible with their vision and principles.

[6] The signatories were representatives from Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and others: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas.

Jim Highsmith asserts that "the Agile approaches scare corporate bureaucrats—at least those that are happy with pushing process for process' sake versus trying to do the best for the 'customer' and deliver something timely and tangible 'as promised'—because they run out of places to hide."[7]

Manifesto for Agile Software Development

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.

As the Agile Alliance noted, the four dichotomies listed in the manifesto ("individuals and interactions over processes and tools") are not intended to suggest that what is on the left is important and what is on the right is unimportant; rather, what is on the right, while important, is simply less important than what is on the left. For example, some believe that the Agile approach advocates providing no documentation other than the code itself. The Agile community would argue instead that documentation *is* important, but no more documentation should be created than is absolutely necessary to support the development itself and future sustainment activities. In fact, Agile emphasizes collaboration and the notion that when documentation replaces collaboration the results are problematic. Documentation should be the result of collaboration.

The Agile Alliance says the following principles underlie the Agile Manifesto:

> *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*
>
> *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*
>
> *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*
>
> *Business people and developers must work together daily throughout the project.*
>
> *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.*
>
> *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*
>
> *Working software is the primary measure of progress.*
>
> *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*

[7] See http://agilemanifesto.org/history.html.

Continuous attention to technical excellence and good design enhances agility.

Simplicity—the art of maximizing the amount of work not done—is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.[8]

From these principles, it is understood that Agile is really a *philosophy* or *development approach*, and it comprises a number of more specific methods, for example, eXtreme Programming (XP), Scrum, and Adaptive Software Development (ASD). (A synopsis of Agile methods is provided in Appendix A.) One of the things that makes adopting Agile challenging in DoD settings is that there is no canonical set of practices that *always* represent Agile. The focus on principles leaves open the possibility of combing existing methods and/or adding new methods that are consistent with the principles.

2.2 A Practical Definition

While this history provides a context for Agile, it does not provide a specific definition. We struggled with defining Agile. Since there are plenty of definitions to choose from, we picked one that closely reflects our intended use of the term "Agile" within this report and one that is more concise:

Agile: An iterative and incremental (evolutionary) approach to software development that 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 that meets the changing needs of its stakeholders.[9]

This definition is rather long but it covers our purposes. If a shorter definition is desired by the reader, Alistair Cockburn has said that Agile is

…early delivery of business value. That involves early and regular delivery of working software, a focus on team communications, and close interaction with the users.[10]

2.3 Example Agile Method

To provide the reader with further context of how Agile might be used, a simplistic generic example for a software development project might include the following.

Initial Planning[11]

---

8 See http://agilemanifesto.org/principles.html.

9 See http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm.

10 See http://bradapp.blogspot.com/2006/05/nutshell-definitions-of-agile.html.

11 For large programs, initial planning would be done during Iteration 0. Iteration 0 is a planning iteration only. Release planning, overall program planning, and high-level architecture creation are some of the tasks accomplished during Iteration 0.

- The overall scope of the project is examined. The business side sets the overall priorities and the development team members select and estimate work items.
- A fixed iteration length is determined (usually between one and four weeks; a two-week iteration appears to be common).
- The functional scope is broken down into a set of capabilities that initially are described in a coarse-grained manner (sometimes called epics). Prior to implementation within an iteration, the epics are broken into stories that are elaborated at a level detailed enough to allow each story to be implemented within a single iteration. They are prioritized, not by the developer, but by a representative of the user, often someone from the PMO (often called a Product Owner).
- The highest risk and highest priority "stories" are typically moved to the front of the queue for implementation in the development iterations. The "stories" at the front of the queue tend to be those with highest stakeholder value, which would include priority and risk.

Planned Iterations[12]

- Each iteration starts with the team planning session; stories are selected from the queue until a full iteration's worth of work is identified.
- Each story (capability) is refined further into specific tasks as noted above.
- Development work first begins with the writing of unit tests that will be used once the software is developed if using the Test Driven Design [TDD] method.[13]
- Coding does not begin until the unit tests have been written (if using TDD).
- At the end of the iteration, the output is an executable, testable product that could actually be used, and unfinished stories could slip into the next iteration. Typically, automated testing is used extensively in Agile.
- A retrospective is usually held at the end of the iteration. The retrospective gives team members an opportunity to reflect on the iteration and determine lessons learned (what went well and what needs to improve) and plan for how they will change their practices in the next iteration to improve their performance.

2.4 Example Agile Program: Coast Guard Logistics Information Management System

The Coast Guard Logistics Information Management System (CG-LIMS) is the primary maintenance software used to support the HC-144A Ocean Sentry aircraft fleet. It manages configuration, maintenance, supply chain, and technical data for the nation's maritime first responder service. Implementation of CG-LIMS began in December 2014 and was scheduled to complete by the end of 2015. The Coast Guard plans to expand the system to support all of its aircraft and some of its boats by the end of 2018.

CG-LIMS was explicitly and emphatically developed using Agile methods, making it a relatively rare breed among military projects. The CG-LIMS Program Manager, Captain Dan Taylor, wrote a blog during the project's early development phase, both to communicate with his team and to provide a historical

[12] Note that the customer or user is available for feedback throughout the iteration.

[13] For more information see: http://agiledata.org/essays/tdd.html or http://en.wikipedia.org/wiki/Test-driven_development.

record of the effort. The blog is no longer updated but it is still online and provides a blow-by-blow commentary of the program office's successful experiment with Agile development.

In a blog post dated August 8, 2012,[14] Captain Taylor shared some observations from the GAO report titled *Effective Practices and Federal Challenges in Applying Agile Methods* (GAO-12-681), pointing out

> *Everyone on the CG-LIMS team should take pride that we're doing (or trying to do) the ten practices that the GAO describes as being used and found effective by officials at five agencies they studied.*

The blog also includes CG-LIMS PowerPoint charts that show the project's specific approach to Agile, such as the chart in Figure 1:

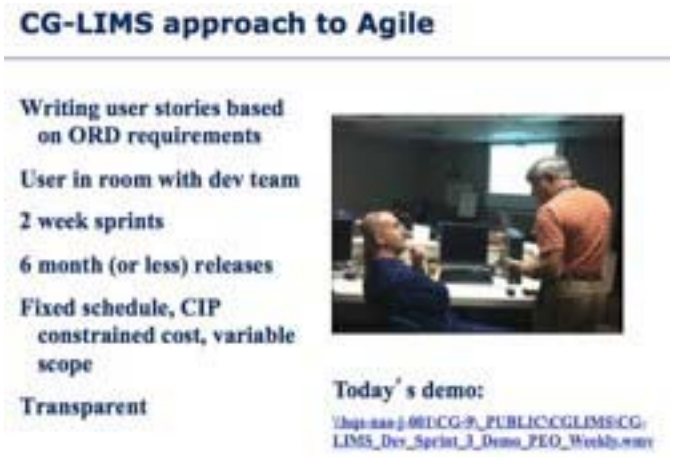

Figure 1: CG-LIMS Approach to Agile

It is worth noting that CG-LIMS adopted Agile despite being subject to the Coast Guard's decidedly non-Agile acquisition process, known as the System Development Life Cycle (SDLC) process. As Captain Taylor explained,

> *[SDLC is] still basically a waterfall process. To the extent we can tailor the SDLC process and help folks like our Asset Manager to change the SDLC process, it will be helpful to know the direction DHS is going with respect to Agile ….*

This is perhaps the most important lesson from CG-LIMS: Agile methods and practices can be implemented even within large government organizations that rely on the formal structures of a Waterfall process, given sufficient imagination, tailoring, and tenacity. That is, processes do not execute themselves. They require actual human beings to make decisions, use judgment, and take action. When confronted with a Waterfall-oriented process, the standard implementation and interpretation is not the only way to proceed. A dedicated leader who wants to use Agile can do so—and do so publicly—by following the Coast Guard's compelling precedent.

[14] See https://cglims.wordpress.com/2012/08/08/gao-on-agile/.

Agile project leaders should not expect to receive universal support and buy-in regarding this approach. However, the absence of unanimous support does not constitute an insurmountable barrier. As Captain Taylor wrote on his blog in July 2012,[15]

> *If it causes you some distress to know that Agile values and the principles of the Agile Manifesto are not embraced within the Acquisition community, know you're not alone. It saddens me too. But please don't let it slow you down. I was explaining to one of my fellow PM's … that I've long since given up on getting everyone's buy in on our approach. We'll need to keep moving with just enough approval and just enough support for what we're doing. As we deliver, we'll continue to make believers one at a time by showing results in the form of working software.*

In the years since that post was published, Taylor's prediction about making more believers has begun to come true. Agile methodologies are gaining significant traction and support, both within the Coast Guard and the wider Department of Homeland Security. As evidence, the DHS Chief Information Officer's office has drafted a guidebook, *DHS Agile Development and Delivery For Information Technology Guidebook* (MD 102-01-004-01) that is based in part on the CG-LIMS experience.

As the current draft explains, this DHS Agile guidebook aims to help program managers "build upon best practices and experience from industry, federal government, and recent DHS implementations of Agile methodologies. This guidance is intended to enhance understanding of why Agile is the preferred approach to federal IT development, and to provide a starting point for increasing DHS-wide application of and expertise in Agile methodologies." Accordingly, it not only constitutes another resource for practitioners who want to use Agile on military projects, it also signals a wider embrace of Agile within the formal leadership of the acquisition community.

2.4.1 Recent Developments: Scaling Agile and DevOps

Since 2010, two significant outgrowths of Agile have developed and taken root, expanding the practice beyond its original horizon. They are scaling frameworks like the Scaled Agile Framework (SAFe), which applies Agile principles and methodologies at the enterprise level (i.e., at scale), and DevOps, which brings Agile development techniques to bear on the operational side of business. There are other scaling frameworks besides SAFe, notably DSDM (Dynamic System Development Method), DAD (Disciplined Agile Delivery), and LeSS (Large Scale Scrum). Details of SAFe and these other frameworks are found in the SEI's forthcoming report on Agile and Scaling. SAFe is summarized below since it is, to our knowledge, the most adopted scaling framework in the DoD defense contractor community.

SAFe extends Agile to the program and portfolio levels, and introduces a broader, more strategic perspective to the practice. While Agile tends to be developer-centric, SAFe includes roles for executives and managers beyond the development team. The result is a roadmap that aims to help large, hierarchal entities realize the benefits of Agile, even on big, complex projects. One of the findings in the 2010 version of this report was that pure Agile does not fully address the needs and interests of large organizations working on large systems. SAFe and similar scaling frameworks may address that need.

DevOps, in contrast, extends Agile laterally and breaks through some of the silos that have historically divided the people who make software from the people who test, run, and maintain it. DevOps bridges these communities to foster more direct and timely communication, as well as to provide operators with

[15] See https://cglims.wordpress.com/2012/07/20/one-approach-to-agile/.

the same tools developers use. By blurring the lines between developers and operators, DevOps aims to not only help deliver valuable features faster, but also enhance the operating environment's stability.

A full examination of scaling frameworks and DevOps is beyond the scope of this report, but it is important for defense acquisition personnel to be familiar with both concepts. SAFe is particularly worth a closer look because it directly addresses some of the primary barriers to adopting Agile within a large enterprise such as the DoD as well as being a framework that program offices are likely to see proposed by defense contractors.

The U.S. Department of Veterans Affairs (VA) has already begun to use Agile at scale, such as on its post-9/11 GI Bill project. Launched in 2011, this system delivers "faster, more accurate payments to Veterans attending school under the Post-9/11 GI Bill."[16]

This was the VA's first experiment with Agile, and according to a 2010 GAO analysis, "the Agile process allowed the department the flexibility to accommodate legislative changes and provide functionality according to business priorities."[17] A subsequent study report from the GAO published in 2011 (GAO-11-663T) provides additional specifics on the results and observes that the VA "deployed the first two of four releases of its long-term system solution by its planned dates, thereby providing regional processing offices with key automated capabilities to prepare original and amended benefits claims." These GAO reports are generally positive on the value and utility of Agile itself while acknowledging that "[the] VA could make further improvements to these practices," hardly a surprising finding for an agency's first use of Agile.

On June 11, 2014, the VA's chief information officer (CIO) signed off on an updated Project Management Accountability Guide, which further emphasizes the importance of using Agile and addresses some of the GAO's earlier recommendations for improvement. Specifically, in paragraph 2.1.8, the guide explains that "the Agile development methodology is the method of choice." This endorsement of Agile signals a significant step away from the traditional Waterfall method.

The VA's results to date are encouraging. According to a fact sheet dated December 1, 2014, the VA now delivers more than two-thirds of its projects on time, compared to the previous rate of one-third. Prior to implementing Agile, development timelines ranged from three to seven years. The average is now 4.2 months.[18]

DevOps techniques are unfortunately harder for the military to adopt, because acquisition offices and operational units tend to be organizationally and geographically distant. This is not always the case and need not be the case. As explained on page 22 of this report, the United States Special Operations Command (USSOCOM or SOCOM) is an example of a military unit where ops and acquisition are tightly integrated already. The popular perception is that SOCOM possesses unique authorities and regulatory freedoms, which enable them to execute projects differently than the rest of the military. However, SOCOM acquisition personnel emphatically insist they do not have any unique freedoms and that their example can be followed by any other project within the DoD. The specific challenge of co-location is not an easy one for a typical project office to address, but with creative use of personnel, it is possible to create a team that

---

[16] See http://www.va.gov/opa/pressrel/pressrelease.cfm?id=2037.

[17] See http://www.gao.gov/products/GAO-11-115.

[18] See http://www.oit.va.gov/docs/oit_fact_sheet/VA_Agile_Development_Factsheet_12012014.pdf.

looks and behaves very much like a DevOps style organization. See the SOCOM story later in this report for more specifics.

3 Interview Observations and Findings

The results from interviews with Agile practitioners and other observations of Agile within the DoD environment from the original report are presented in this section. The results are lessons learned from actual application of Agile within the DoD environment. We reviewed the interview notes and other observations and found seven common topics, which we use to frame the discussion of the overall results. Each topic is structured to provide a context and the associated finding/observation. These topics are related to each other and there is overlap between their findings.

Note: Section 5 provides additional information about Agile concepts and how to apply them; Section 3 presents **only** what we learned from the original interviews and observations, with new observations and comments annotated as "*2016 update*." The list of original interviews is found in the 2010 version of this report.

3.1 Common Topics from Our Interviews

The following topics came up multiple times in our interviews. Each is treated in its own section below:

- Acquisition
- Knowledge of Agile
- Culture
- Oversight
- End-User Involvement
- Integration and Test
- Infrastructure

3.2 Acquisition

Overview

It is widely believed, both by program offices and DoD contractors, that the DoD 5000 series and other regulations and guidance documents limit the government and contractors from using a non-waterfall approach. A particular sticking point is that Agile does not readily accommodate large capstone events such as a CDR. However, the programs that have used Agile in software development have found that the DoD 5000 series has great flexibility and does not, in fact, preclude the use of Agile.

2016 update: The 2015 version of DoD 5000.02 includes at least one acquisition model diagram that is frequently interpreted as a support for Agile approaches to software development [DoD 2015].

Context

A strong belief that is prevalent across the DoD community is that the DoD 5000 series, and other acquisition policies, instructions, and regulations, are rigid in requiring a traditional Waterfall process for the

development of software. It is true that most acquisition personnel have been specifically trained in applying the Waterfall lifecycle to these acquisition regulations, irrespective of whether they are acquiring tanks and aircraft or acquiring software that might be used in stand-alone software applications, such as pure IT acquisitions (e.g., enterprise resource planning or personnel/pay systems). (The original findings and recommendations were created from the original data set. We use "*2016 update*" where the update authors make additional comments to the original material.)

Finding/Observation

Those programs that have used Agile in software development have found that the DoD 5000 series has great flexibility and does not in fact preclude the use of Agile. It appears that with careful review and some tailoring an alternate interpretation can be created so that Agile can be used on DoD programs.[19]

2016 update: The January 2015 version of DoD 5000.02 requires less interpretation than the 2008 version for programs wanting to use Agile. Software-only programs can start with Model 3, and even weapons systems have a "software-intensive" Hybrid B model to start from.

Context

An interesting corollary to the prevalent belief of using traditional methods is that many RFPs are written in such a way that a non-Waterfall response would appear to be or might be noncompliant. Most traditional RFPs require a full complement of Contract Data Requirements Lists (CDRLs) for documentation of progress.

2016 update: This is still an issue. Unless a program's RFP can account for incremental technical reviews and incremental documentation delivery, some aspects of typical Agile implementations will be less efficient.

Finding/Observation

This level of documentation is contrary to Agile precepts of creating "just enough" documentation. "Just enough" will vary from situation to situation depending on the needs and regulation requirements of the project. For Agile to become common place within the DoD, the acquisition organizations should encourage, and contractors should provide, a compliant proposal with suggested alternatives that use Agile. It is important for the acquirer to understand Agile benefits and to include project-specific guidelines[20] in RFP language for how Agile responses should be framed.

---

[19] DoD 5000 might not have addressed Agile back in 2010, but other documents did:
- The 2010 National Defense Authorization Act Section 804 allowed for
  - NEW ACQUISITION PROCESS REQUIRED —The Secretary of Defense shall develop and implement a new acquisition process for information technology systems
  - "… Be based on the recommendations in Chapter 6 of the March 2009 report of the DSB Task Force on DoD and Procedures for the Acquisition of Information Technology
  - Be designed to include—
    (A) early and continual involvement of the user; (B) multiple, rapidly executed increments or releases of capability; (C) early, successive prototyping to support an evolutionary approach; (D) a modular, open-systems approach
- See: http://www.acq.osd.mil/evm/resources/AgileSep2015/DOD%20AGILE%20DAY%201%20-%20CYBER%20UPDATE.pdf.

[20] As of publication, we are not aware of the existence of any guidance for how to frame Agile responses. However, a National Defense Industrial Association (NDIA) systems engineering working group is looking into the issue.

Context

A very specific acquisition issue and sticking point is that the Agile methodology does not accommodate large capstone events such as a 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.

2016 update: We have seen several different approaches to acquisition technical reviews, several patterns of which are documented in Lapham's *Agile Methods and Request for Change (RFC): Observations from DoD Acquisition Programs* [Lapham et al. 2014].

Finding/Observation

A software developer using Agile typically does not complete the design before beginning implementation of it, so the scale and comprehensiveness of a CDR is quite foreign to Agile development teams. Some experienced Agile providers have accommodated this issue by breaking the typical Waterfall-based CDRs into multiple Interim Design Reviews (IDRs), which is an example of the type of "flexibility" in implementing DoD 5000 requirements. These IDRs need to reflect the iterative nature of Agile, and they can be held more frequently and with tighter focus for only a few hours at a time, as opposed to the several days needed for CDRs. The entry and exit criteria for an IDR need to be dependent on the current iteration, and the results of a combination of all IDRs completed should be functionally equivalent to a CDR using Waterfall.

2016 update: The January 2015 version of DoD 5000.02 shows, in Model 3, an example of iterative reviews versus capstone events.

3.3 Knowledge of Agile

Overview

Agile methods have been developed and used most extensively in the non-DoD commercial sector with small- to medium-sized projects. Experience with larger projects has been swiftly accumulating in the commercial sector. As a result, fewer DoD acquisition professionals are familiar with the use of Agile or possess the necessary experience to effectively implement it. The DoD contractor community appears to be embracing Agile, which makes it imperative for DoD acquisition personnel to come up to speed.

2016 update: A recent change in the learning objectives of the Defense Acquisition University's (DAU) intermediate acquisition course resulted in the SEI being commissioned to develop new Agile practicum materials for inclusion in the course's capstone simulation. Inclusion of Agile content in DAU courseware is one of the signs that the acquisition community is taking Agile seriously enough to build the workforce's knowledge about it.

Context

A good understanding of the fundamentals of Agile development methods is required by both the government and contractor personnel. Without this understanding and knowledge, misunderstandings will certainly happen and could have disastrous consequences.

Finding/Observation

One example we found of a typical misunderstanding arose as early as contract negotiations. The contractor specified that it would deliver documentation in accordance with best Agile practices. The government included a list of required documents in the contract. The contractor understood that the Agile best practice meant that there would be minimal detail and documentation (what, when, and how much detail would be done) up front. Even though the contractor understood this, the government did not and still expected all the detail that was traditionally provided for the documents. The government had trouble accepting less detail and documentation even though the complete content would become available at a later date.

The issue was not only how many documents were produced but the expected level of detail in the documents and when the documents would be complete. One way to increase the amount of knowledge of Agile and avoid this type of misunderstanding is through education. For example, courses at the DAU and other institutions could be updated to include discussions of Agile, its pros and cons, and the challenges Agile presents to the PMO. One such pro is that Agile forces closure on requirements analysis for the iteration and flushes out problems early in each iteration; these are very desirable attributes. One potential con is the resistance to the amount of culture change that may be required to employ Agile.

2016 update: As suggested above, the DAU is starting to include Agile material in its acquisition core courses.

Context

The acquisition community's imperfect understanding of Agile might undermine the success of an Agile contractor. Agile is relatively new and has its genesis in the software development community itself, which is mostly isolated from acquisition concerns. As a result, relatively few acquisition professionals have direct experience with Agile, and such Agile-unaware PMO members might insist on the more familiar project plans and measures, but they will not fit into Agile.

Finding/Observation

All government and contractor personnel need to spend the time necessary before contract award to understand what it means to use Agile from all perspectives. The following are examples of such considerations:

- Which contractual phases can employ Agile?
- What are the milestone and deliverable details for each phase when using Agile?
- What contract changes would be needed?
- What changes to the approach of monitoring development progress will be needed?
- What type of staff members are needed on both sides (government and contractor)?
- Which of the 5000.02 process formalities will be tailored?

Context

Both government and contractor personnel need to acquire an appropriate skill set to support Agile use within DoD systems. The nature of Agile lends itself to a slightly different staffing model than the one the government is used to seeing with Waterfall.

Finding/Observation

From our interviews, we learned that there are subtle but critical differences:

- The contractor's program manager needs to be experienced in Agile. The government program manager should also be experienced in Agile, though at present this might be more than can reasonably be expected.
- Contractor personnel need to be trained and experienced in the Agile method to be used on the project.
- PMO personnel need specific training in the Agile method that the contractor is going to use, as well as a more general understanding of Agile. They need to develop an understanding that Agile is adaptive to each project or program.
- There needs to be an expert advisor/advocate for Agile in a position of authority in the PMO. Without authority, such an advisor/advocate becomes "just another opinion."

Additional Finding—2015

Context

While the mission of the DAU is to "Provide a global learning environment to develop qualified acquisition, requirements and contingency professionals," the DAU does not exactly *teach* Agile methods (which is appropriate, since they are focused on the *acquisition* aspects of systems development, not the engineering/development aspects). Rather, the curriculum includes a few brief modules that broadly introduce the concept and increase students' awareness of Agile. Interested students can then go on to pursue in-depth training through other outlets. At the end of 2015, Agile-related material can be found in the following courses:

- Intermediate Information Systems Acquisition (IRM 202): Approximately six slides, presented in the second week of this 10-day course.
- Advanced Software Acquisition Management (SAM 301): An Agile case study is presented as a four-hour block in this five-day course.
- Leadership in Engineering Defense Systems (ENG 301): One Agile block is presented in this 10-day course

Finding

Despite the inclusion of Agile content in DAU acquisition courses, many acquisition practitioners are still largely unaware of the basic principles and tenets of Agile. There is, therefore, a significant opportunity to add Agile to the common toolbox for acquisition professionals within the department by increasing general awareness of the concepts and benefits.

The question of how best to provide Agile training, education, and coaching is still open and merits further research. The DAU clearly has a role to satisfy at least a portion of the training requirement, while commercial and academic entities outside the department are also well positioned to help address the

need. The optimal solution will likely be a blend of in-house and external resources, but how best to assign these roles and responsibilities is currently undefined. A new continuous education course, CLE 076, an introduction to Agile in DoD, is in progress and is expected to be released in late 2016. This online course will hopefully begin to fill the need for Agile-specific education.

3.4 Culture

Overview

Culture is the customary knowledge, beliefs, behaviors, and traits displayed by an acquisition organization or contractor. The government is heavily invested in the use of Waterfall for acquisition of all equipment and systems, whether they are software intensive or not. As a result, a large segment of the DoD acquisition community (and that of long-time DoD contractors, as well) is more comfortable with Waterfall and skeptical about using Agile. There are many cultural norms that Agile embraces that are not traditional for DoD acquisition.

2016 update: The SEI document *Agile Methods: Selected DoD Management and Acquisition Concerns s* contains an entire chapter on cultural issues associated with moving to Agile.

Context

The government is heavily invested in the use of Waterfall for acquisition in general, and this has been applied to software acquisition as well. While other methods have been used, Waterfall and its accompanying precepts are most familiar to most PMOs.

Finding/Observation

Moving to Agile is challenging—many of the "old ways" and paradigms need to be modified using a fundamental culture change. As alluded to previously, the existing training in the interpretation of software acquisition requirements is skewed toward the Waterfall approach. Thus, a PMO employing Agile will need to be trained in Agile concepts.

2016 update: In addition to supporting increasing Agile content in DAU courseware, the SEI has developed Agile-related courseware that addresses the government side, as opposed to the development side, of Agile efforts. While commercial firms provide strong support for training on the development side, the market for the acquisition/government side is much smaller, and hasn't generally been taken up by the Agile training community. The SEI courseware fills that gap.

Context

One challenge regarding benefits for the DoD is that the acquisition community might not perceive that there is any benefit in using Agile. Many believe Agile is *ad hoc* and that it does not produce necessary documentation or apply any rigor to development.

Finding/Observation

The PMO specifically needs to realize that while Agile provides many benefits, many of the traditional Waterfall activities, documents, etc. will not be present. In some cases, the data will be present but not in the anticipated form.

Context

Since the type of management oversight is different for Agile than Waterfall, members of the PMO are likely to feel that they are losing control over the program.

Finding/Observation

Historically, the PMO's role is to ensure orderly development progress, but with Agile the PMO has to relinquish control over how low-level change is managed. Agile attacks high-value and/or high-risk user items first instead of making steady progress on all requirements. This difference in handling requirements can create unnecessary friction between the PMO and the contractor, sometimes leading to outright hostility.

Context

Both the PMO and the contractor need to be aware that different skill sets or skill mixes will possibly be needed in programs using Agile (as opposed to programs using Waterfall).

Finding/Observation

Agile takes a lot of strong, focused team and management oversight at the mid- and low-levels versus the high level, particularly if a new development project/program is using a merger of Waterfall and Agile. Furthermore, the reference estimates that PMO members have developed over time about the number of developers needed based on the size/volume of the code may not be valid in an Agile environment. The management oversight required in the developer's facility is more at a technical level than at the project/program management level when using Agile—what is needed are "iteration leads," "scrum leaders," etc. This different skill mix does not necessarily lead to a more costly management structure, but it does require a different "scorecard" to evaluate progress and troubleshoot development issues [Cockburn 2002, Boehm & Turner 2004].

The important point here is that the PMO must be prepared to deal with organizational change management issues, particularly related to overseeing small batch increments versus large batches.

3.5 Oversight

Overview

Tracking and measuring progress while using Agile in a way that is clear to and trusted by the government is actually expressive of Agile values, but contractual and other oversight norms make this issue a continuing challenge. The measures applied on past software acquisitions typically do not work well for Agile at best, and at worst do not work at all. Agile also does not innately support showing outside the team the kind of granularity of estimates and task detail that is typically shown across the entire project in an Integrated Master Schedule (IMS). Rather, Agile provides high granularity task-level estimates for just

the upcoming iteration and provides progress measures externally at a higher level. On the other hand, well-functioning Agile developers provide the opportunity for the program office to review working software as often as every two weeks, which has the effect of building trust and transparency over time.

Context

Traditional Waterfall provides significant oversight and insight into the implementation details of the program; this method is very structured so that it provides predictability, stability, and high assurance, given that the requirements and environment stay stable [Boehm & Turner 2004]. The execution of Agile is distinctly different from what the PMO has seen in the past on programs using Waterfall. The control and discipline comes from the Agile team itself rather than from control external to the team, that is, project and higher management. As a result, the PMO will see a different way that the development is controlled, executed, and viewed.

Finding/Observation

We learned from our interviews that the PMO has to be prepared to relinquish some level of control and oversight of the program to allow Agile to operate effectively. What is needed is a system of program measures that allows the PMO to have insight into the developer's priorities and the development progress being made on a day-to-day basis, and this will allow the PMO to achieve an optimal balance between insight and oversight.

Context

Forecasting the project schedule when using Agile requires an entirely different approach than when forecasting the project schedule during Waterfall. Agile depends on being able to determine the content of iterations on a just-in-time basis, to use very short iterations, and to respond quickly to customers' changing needs. The creation of a traditional detailed IMS with the content of each iteration for the entire project is not done with Agile.[21] Agile does not support the kind of granularity of estimates and detail that are typically shown in a traditional IMS for the entire project. Traditional IMS estimates and the corresponding constituent tasks are very detailed and require a great deal of effort to change or update. This is counter to the "just-in-time" philosophy used in Agile.

Finding/Observation

Our interviews indicated that the IMS can be maintained at a level that is compatible and appropriate for Agile. This may be more difficult than it appears because it requires a different perspective about when and to what level of detail the IMS should be developed.

2016 update: The choice of level of detail for an IMS in an Agile program is an ongoing issue. Although some programs have successfully navigated these waters, others get mired in expectations of fine-grained schedule management, resulting in one program we know of requiring one staff day of effort each week to update the master schedule, per team!

---

[21] Agile creates detailed schedules for the current iteration. Agile does not create detailed schedules for all iterations a priori.

Context

An additional impact is that the estimates at the iteration level in Agile are done by the iteration team, not just by management (team leads and higher level management)[22] as is the case in most Waterfall developments. Depending on the skill level and the amount of learning achieved within the development team from previous iterations, the estimates they produce might be much more coarse-grained than expected by the PMO.

Finding/Observation

If the project is not adopting Agile outright, then some compromise between the PMO expectation of a detailed IMS and the contractor's Agile management techniques will be needed to define a model that uses the best practices from both Agile and Waterfall. Some things that have been done on existing programs and some options the PMO could consider include

- Traditional progress measures such as earned value and percent complete might be possible to use. Because a detailed IMS is not realistic for Agile, these measures would need to be computed differently than for Waterfall.
- Progress could be measured by the number of stories completed, though for the PMO to find this useful, it would need to understand the full inventory of stories that the contractor projects for the development and be convinced that the sum of all the stories fully comprehends the project requirements.
- Progress could be measured by the accumulation of "user value" during development. Unless the PMO participates in establishing the use values, this might not satisfy the PMO unless they had fully concurred with the contractor-assigned story (capability) values.
- In many Agile developments, the contractors use Agile tracking tools to keep track of progress. In one of the projects we interviewed, the PMO did use those same tools in lieu of expecting paper progress reports and acquiring a progress-tracking tool of its own. This approach has two advantages:
  - The PMO learns and uses the contractor tools to follow progress and review designs, which reduces the work and cost for the PMO.
  - The contractor realizes cost savings because he or she does not have to do any translation of what he or she sees in his or her own tools to what the PMO expects.
- The contractor and the PMO need to negotiate common ground to define the needed hybrid model for the measurement system.
- The project progress measurement system to be used must be negotiated and agreed upon early in the project/program.

Context

Another form of oversight used in traditional programs is the production and review of documentation on a regular basis. At first look, Agile documentation might not meet DoD expectations and the perceived need for acquisition office oversight. Most PMO personnel expect a full complement of CDRLs that are

[22] Note that there is a difference in how estimations can be done at the iteration, release, and enterprise levels. At the iteration level, the team should always be involved. However, as the project gets bigger, the need for release- and eventually enterprise-level estimates may look more like those seen in Waterfall.

complete upon delivery and that are provided at regular, defined intervals or milestones using traditional methods.

Finding/Observation

A developer using Agile only creates the minimum documentation necessary to accomplish the tasks at hand, and the as-built documentation evolves over time into a final product. Thus if Agile is to be employed, the government PMO needs to agree to less-than-full-blown documentation, as this saves time and avoids abandoning expensive documentation later. Further, the government PMO needs to relax traditional CDRL-level documentation completion (as opposed to incremental delivery) at milestone events. Still, the parties need to negotiate documentation to ensure that important data represented in a minimal required set of documents (programmatic and technical) is gathered. This requires more software expertise of the PMO staff, who need to recognize that documentation versus functionality is a "zero-sum game:" if more documents are required, then less functionality will be delivered in the final system.[23]

Eliminating these documents and the related oversight is easy to say but requires trust that the contractor is doing it right; this requires some other mechanism that ensures the proper oversight such as the government being on site, frequent code reviews, and frequent process checks. The Agilist might argue that the iteration builds provide the visibility needed for government oversight. But until there is more government experience with Agile methods, it will be difficult for the PMOs to relinquish the current technical documents needed for oversight.

3.6 End-User Involvement

Overview

The close involvement of end-users in the development process, reviews, and demonstrations, upon which successful Agile implementations depend, is often extremely difficult to achieve with the many stakeholders typical of DoD acquisitions. In addition, the continuous availability of the end user is an issue in the DoD environment, as end users are usually in operational, not acquisition, organizations. Acquisition personnel tend to be isolated in acquisition organizations, rather than integrated into operational units.

Context

One of the fundamental principles cited in the Agile Manifesto is *customer collaboration*. In other words, Agile believes close interaction between the developers and end users is important. A basic Agile principle is "business people (users) and developers work together daily,"[24] but in the DoD acquisition environment this is rarely easy, and it may sometimes not even be possible. DoD acquisitions—especially joint service acquisition—involve many stakeholders with inherently conflicting needs. It is hard to get a single viewpoint from the *customer* because no one person truly represents the users. Plus, it is hard to get all the stakeholders (maintenance and sustainment personnel) involved in development decisions.

[23] This should not be construed to think that doing no documentation is an option.

[24] See http://agilemanifesto.org/principles.html.

Finding/Observation

From interviews with existing programs we found

- A single voice for the user/customer is essential. This could be accomplished through an input-filtering steering committee that documents decisions, insists that the user community speaks with one voice to the Agile developer (through requirements definition), and receives input from and gives direction to a single person representing the Agile developer.
- True users (not just PMO representatives) *must* attend demonstrations that are given specifically to get user information and feedback.
- A hybrid approach (something between "pure Agile" and DoD 5000 traditional methods) is needed for large systems to assure that agreements with multiple users are documented, external interfaces are documented and agreed to, and multiple contractual and programmatic constraints are honored.
- A strong emphasis on government/user participation in reviews and demonstrations is essential. These reviews and demonstrations will be of shorter duration and have a tighter focus with Agile, and this will result in more frequent reviews that reflect the nature of Agile development. For example, having eight two-hour reviews spread over time, as opposed to a single two-day event to cover the same material, would be the implementation of multiple IDRs instead of one CDR.

2016 Update—Example: End-User Involvement Example: SOCOM

USSOCOM is an example of a government entity with tight integration between acquirers and operators. The two functions are co-located at SOCOM headquarters, where the senior acquisition executive staff works down the hallway from their operational counterparts. This enables rapid collaboration and immediate communication and shows that such arrangements are possible. As U.S. Air Force Capt. Tim Troup explained in an article titled *Acquisition Truths from the Trenches* [Troup 2010], "layers of command structure have been removed and the JATF commander has direct contact with the CADs [Combat Acquisition Detachments]. This is proving to be a game-changer."

This functional collaboration arrangement is not typical within the military but is frequently cited as a key reason for SOCOM's impressive track record. This example of what *right* looks like serves as a useful guide to consider when setting up an Agile program office. Removing unnecessary organizational layers is consistent with Agile and the FAR (see FAR 35.002 in particular), and streamlining the bureaucracy should be one of the first steps when establishing a new program. SOCOM includes this concept in its list of foundational Acquisition Truths, stated this way: "More bureaucracy does not ensure a better product."

There is a widespread misconception that Special Operations units have special permissions unavailable to standard acquisition units, and thus the experiences in SOCOM are not imitable by non-Special Operations organizations. Captain Troup addresses this as well:

> *[Y]ou may think that USSOCOM does not have to follow the acquisition regulations and directives like the rest of the military, but that is not true. USSOCOM follows DoD 5000 series policy, with the same funding rules and new program starting rules as standard acquisition programs.*

It is not always possible to put acquisition and operations in the same organization or location, but the option is available much more often than it is implemented. Even when co-location is not possible, today's IT capabilities make geographically distributed teams easier to integrate than ever. Video conferencing and online collaboration tools provide mechanisms to maintain a richer level of interaction than previous

generations had. There is still no substitute for physical presence, but if a distributed team is interested in building closer linkages, there are a number of powerful tools available for them to use.

The principle is to increase direct, high-bandwidth contact and reduce bureaucracy. SOCOM shows it can be done. Other expressions of this principle may not look exactly like SOCOM's experience, but the Special Operations community provides a compelling example of what customer collaboration can look like in the defense acquisition business.

3.7 Integration and Test

Overview

Because integration and testing activities are part of Agile development iterations, the approach to these activities might significantly change from those used in Waterfall. The biggest change is that integration and test need to be done throughout the project as opposed to waiting until the end of the release cycle. This is another particularly challenging issue of culture change since, historically, integration and test organizations have been outside the development teams once you get beyond unit testing. We have observed several programs where developmental test has "moved to the left" inside the Agile release cycle, while operational test and cybersecurity certifications remain outside the Agile development.

2016 update: There is some interest from these communities in engaging more incrementally with Agile developments, but this is not a solved issue even six years after the original publication of this report.

Context

Test and integration are incorporated throughout the iteration lifecycle used within Agile as opposed to Waterfall, which puts it at the end. Testing can have a significantly different role in the project depending on which Agile method is used. The big advantage in Agile testing and integration is that testing can, and usually is, started earlier because of the short timeframes for iterations; this flushes out problems more quickly. Furthermore, gathering customer feedback during the development phase (in each iteration) provides an early look at the code capabilities and helps reduce risk at the time of system integration.

Finding/Observation

We learned from our interviews that

- Within the DoD, "sell-off" is the process used by the contractor to obtain formal acceptance of the developed product from the government, thus the government takes ownership as the contractor "sells off." The nature of "system sell-off" from the contractor's perspective is still the same as in Waterfall; there might be fewer sell-off risks because of the frequent interactions among all parties during the demonstrations.
- Software builds are completed much earlier with Agile since each iteration produces a usable build; because of this, more frequent test and integration work can be done.
- The government test community can (and should) be involved early.

- A lesson specific to integration is that the software integrators need to have access to the ultimate target environment. This reduces issues for the development teams when they get to system-level integration. (The degree to which this is an advantage depends on the target environment and the number of platforms that are involved.)
- Access to the developers for the testing/integration team can be an issue because of the short (typically two-week) development cycles; this puts intense time pressure on the development team and should be addressed during forward iteration planning as iteration cycles are completed. Further, this might suggest that the testing/integration team members need to be part of the development team.
- Government testing personnel need to understand the differences inherent in Agile versus Waterfall to adequately adapt staff and time requirements for testing when using Agile. The government testing personnel need to be engaged at the development iteration level; they should not wait until the entire system is completed to initiate their testing work.

3.8 Infrastructure

Overview

No matter whether one uses Waterfall or Agile, the group undertaking the project needs to have an infrastructure supporting it. This would include the organization of the team and the context within which the team operates. The overall organization of a project using Agile is different from the traditional program structure. The structure for an Agile project reflects Agile precepts and is reliant on the context in which it will be applied. In addition to organizational structure, the organization's technical infrastructure is also an area of challenge—there are many organizations in DoD that have security boundaries that prevent sharing of data without expensive solutions that are often insufficient in bandwidth to allow tool-level sharing between the government program office and its Agile contractors. Solutions in this area are starting to emerge but are not yet promulgated throughout the Agile DoD community.

Context

For large programs, there is the need for early decisions about the support structure, including shared assets. To help eliminate configuration management issues, the government usually dictates the shared assets of models across contracts and contractors on a large program. Common facilities (or shared assets), such as common logging (for example, automated logging of test messages), agreements on units of measure, data models, etc., will be used on all segments and components of large programs. Early decisions on such aspects can appear to be in direct contradiction to the tenets of Agile.[25] However, developers using Agile need to be aware of the larger, system-wide constructs during their iteration planning. Developers need to use these as inputs so that they can be accommodated during the Agile implementation.

Developers also need to understand that for DoD-type programs, the Concept of Operations (CONOPS) developed for the overall system by the government strongly influences and provides the context for the capability stories used in Agile development. Therefore, an up-front operational architecture needs to be

[25] This may be true for small-to-medium sized Agile projects. However, for large projects, Iteration 0 would be employed to work activities such as architecture.

defined as part of the CONOPS, and all developers of the overall system need to understand that conventional "use cases" and Agile "stories" are different in construction and application. Agile "stories" are less-formal constructs written as informal English descriptions. "Use cases" are formal constructs including preconditions, post conditions, and detailed interaction diagrams.

Finding/Observation

Interviewees had several potential organizational ideas for executing Agile in the DoD acquisition environment:

- Agile can be experimented with early in the acquisition lifecycle to try out what works and what doesn't. Possible times to experiment might be during analysis of alternatives, risk reduction activities, activities leading up to Milestone B, phases in which only coding is being produced, and Advanced Concept Technology Demonstrations (ACTDs).
- For programs just getting involved with Agile, one organizational structure that worked well involved customers on-site at the contractor's facilities using a two-week rotational schedule. The interviewees indicated that such a short rotational schedule benefits the contractor team because it typically provides much better access to real users, and it benefits the PMO team members because they have better insight into what is going on in development.
- To help get started with Agile, the contractors brought in an Agile expert who would be embedded in the team and then train his or her way out of the job. This way the Agile team learns by doing, not just from classroom training or books.
- The contractors created something concrete that behaves in a representative manner, such as an early version or prototype.

Large programs with multiple Agile teams had several more ideas:

- To coordinate project dependencies across multiple Agile development teams, the leaders of the development teams, who typically maintain control of the team through a daily "scrum," can themselves become members of a team of consisting of all the team leaders (a "scrum of scrums").[26]
- To maintain subject matter expertise and foster the cross-training of staff, team leads should be permanent, rotating the staff underneath them. This allows the cross-training of staff in all areas and maintains the team lead as subject matter experts—a "best-of-both worlds" approach.
- Planning for iterations was difficult with multiple Agile teams running in parallel and working on the same source tree. It was difficult to track feature predecessors. For example, if story A is needed before story B can be implemented, and story A was scheduled to be completed during the last iteration by another team, the development team needs to know if story A actually made it into that iteration before scheduling story B. This problem is made more difficult if both A and B are scheduled for the same iteration because each team can decide for itself which stories get bumped from a particular iteration. To preclude this type of behavior, one group made this particular topic part of their daily team lead standups. This is a particular problem when teams are organized around components rather than features.
- Dependencies across multiple Agile teams working on a common source tree need frequent coordination. The interviewees pointed out that one possible way of doing this would be to have the Agile team leader scrum (the "scrum of scrums") meet in a daily standup to track the interdependencies.

[26] This is true when using Scrum techniques.

2016 update: Some of the issues described above have been addressed by scaling frameworks like SAFe and DSDM. Having ways to normalize development cadence and frequently synchronize across teams are critical elements of large Agile projects.

Context

Another issue that needs to be considered with multiple teams is code refactoring. "Code refactoring is the process of changing a computer program's internal structure without modifying its external functional behavior or existing functionality, to improve internal quality attributes of the software."[27]

Finding/Observation

We found that the Agile refactoring step can have unforeseen side effects. When refactoring of code is carried out as a standard step in Agile and the code involved is only part of a larger, interrelated software system, team members without comprehensive knowledge of all the interrelationships can inject serious defects that are not apparent to the development team and will not emerge until system integration. From our interviews we found that

- The Agile team needs to know at all times where their code fits with other teams' code, and what it is affecting across the entire system configuration, including the Work Breakdown Structure (WBS).
- The use of design patterns and team training in the overall architecture of the system can improve the learning curve for new team members and can help to constrain refactoring variability.
- Multiple teams working at the same time without knowing the overall program context should be cautious when refactoring solutions that simplify the code being developed. The refactoring can seem to provide an early value for code maintainability and modifiability, but ultimately not be scalable to the larger overall infrastructure.

On large, complex systems there may be a need for an entire iteration that is devoted to refactoring after integration of the various teams' code.

---

[27] See Wikipedia http://en.wikipedia.org/wiki/Refactoring.

4 DoD 5000 Series and Agile—Potential Issues and Conflicts

Several policies, instructions, and regulations were reviewed for the original version of this report to determine how they might impact the use of Agile on DoD programs. In particular, the DoD 5000 series was reviewed in depth. In all cited cases, we tried to determine if there could be an interpretation that might preclude or limit the use of Agile. In some instances, it appeared that the policy or regulation actually encouraged the use of Agile or at the very least some of the Agile concepts. The Department of Defense Directive (DoDD) 5000.01 provides supportive, challenging, and constraining policies that would need to be interpreted and applied to the specific program. Excerpts and considerations are provided in Table 1 for the areas discussed. The January 2015 version of the DoD 5000.02 does not explicitly mention the term Agile, but those who work in Agile settings recognize that some of the constraints encountered in earlier versions are no longer present.

2016 update: The term *Agile* is not present in the 2015 version of DoD 5000.02 (similar to the 2008 version). However, page 31 of 5000.02 discusses "incrementally deployed" software (Model 3: Incrementally Deployed Software Intensive Program), which opens the door to Agile-like processes. In 5000.02, the description for Figure 5 on page 11 states

This model is distinguished from the previous model by the rapid delivery of capability through multiple acquisition increments, each of which provides part of the overall required program capability. Each increment may have several limited deployments; each deployment will result from a specific build and provide the user with a mature and tested sub-element of the overall incremental capability. Several builds and deployments will typically be necessary to satisfy approved requirements for an increment of capability [DoD 2015].

4.1 Use of Agile Is Not Prohibited by DoDD 5000.01

Support

Flexibility, responsiveness, innovation, and collaboration are all terms that one might see when discussing Agile. These terms are in fact section headings in DoDD 5000.01. One could interpret these sections as encouraging the use of methods such as Agile. Other sections on Integrated Test and Evaluation, Professional Workforce, and System Engineering also support the use of Agile; at least they seem to be open to methods other than traditional Waterfall.

Challenges

There are other areas within the directive that provide challenges for the use of Agile. These are cost, affordability, and cost realism. In these areas, the policy requires the program to determine the total cost of ownership, which seems to be based on knowing all requirements at a detailed level up front. Agile does not necessarily support this concept well because all of the requirements are not known at a detailed level up front. However, cost as an independent variable is an inherent part of Agile, which starts out with a high-level estimate that can be, and is, refined as the program progresses. Agile allows the developers to provide an incremental total cost estimate at a detailed level as the iterations are performed.

The big challenge with moving to an incremental costing approach is that the contracting cycle takes too long for just one development iteration. So, the options are either to develop a more streamlined competitive bidding process that takes months instead of years to execute or estimate several Agile iterations based on a sample set of requirements. This requires a common understanding that the actual requirements delivered will vary depending on how the PMO and end user prioritize requirements.

Another challenging area is the Program Stability policy, which details when the Milestone Decision Authority (MDA) determines to fully fund an acquisition program; generally, this is when a system concept and design have been selected. Some might say that since the design within Agile is evolving throughout the program, it therefore does not support this policy. However, DoDD 5000.01 also states that "Evolutionary acquisition strategies are the preferred approach to satisfying operational needs." The practice of evolutionary acquisition, which involves incremental deliveries of operational capabilities, is precisely the sort of thing Agile supports.

DoDD 5000.01 goes on to establish an overarching policy that states "acquisition professionals shall continuously develop and implement initiatives to streamline and improve the Defense Acquisition System. MDAs and PMs shall examine and, as appropriate, adopt innovative practices (including best commercial practices and electronic business solutions) that reduce cycle time and cost, and encourage teamwork." This policy opens the door to Agile quite wide. Further, Agile does provide for an overall architectural framework (sometimes in Iteration 0) so even the Program Stability policy can be met using Agile. The PMO would need to work closely with the MDA on meeting this objective, but it is possible.

Constraints

There are other areas within the directive that provide constraints for using Agile on a program. These include independent operational test, information assurance, information superiority, and interoperability. These areas address the overall "environment" or context within which an Agile development project would need to operate. These policies and their implementations for the program are constraints within an Agile development effort.

Table 1 provides considerations for DoDD 5000.01 policies that the PMO should investigate before adopting Agile.

Table 1: Agile Considerations for DoDD 5000.01 Guidance

DoDD 5000.01 Guidance Excerpts	Considerations
4.3.1 Flexibility "There is no one best way to structure an acquisition program … MDAs and PMs shall tailor program strategies and oversight, including documentation of program information, … to fit the particular conditions of that program, consistent with applicable laws and regulations and the time sensitivity of the capability need."	**Support** This policy provides the foundation from which a program could adapt oversight suitable for Agile development. It also provides the high-level guidance for tailoring documentation such as CDRLs, which would be critical when using Agile methods.
Section 4.3.2 Responsiveness "Advanced technology shall be integrated into producible systems and deployed in the shortest time practicable. Approved time-phased capability needs matched with available technology and resources enable evolutionary acquisition strategies…. Incremental development is the preferred process for executing such strategies."	**Support** Using Agile might allow for deployment in the shortest time practicable. This policy certainly lends itself to Agile methods and is applicable in a software venue. The PMO would need to interpret it for their given program and apply appropriately.
Section 4.3.3 Innovation "MDAs and PMs shall examine and, as appropriate, adopt innovative practices (including best commercial practices and electronic business solutions) that reduce cycle time and cost, and encourage teamwork."	**Support** This policy also seems to be an invitation to use Agile. The PMO would need to interpret it for the given program and apply it appropriately.
Enclosure 1 Additional Policy, Section E.1.1.2 Collaboration "The DoD acquisition, capability needs and financial communities, and operational users shall maintain continuous and effective communications with each other by using Integrated Product Teams (IPTs)."	**Support** In the Agile environment, the iteration teams are cross-functional teams consisting of programmers, testers, and others as needed. Continuous and effective communication is one of the cornerstones for Agile. This policy supports the use of Agile.
Enclosure 1 Additional Policy, Section E.1.1.4 Cost and Affordability "The DoD Components shall plan programs based on realistic projections of the dollars and manpower likely to be available in future years."	**Challenge** This section deals with the reality of fiscal constraints and the notion that cost should be viewed as an independent variable. The MDA needs to address the total costs of ownership, and the user should address affordability in establishing capability needs. These items might need to be obtained in a different manner when using Agile as it does not involve a detailed determination of requirements nor identify all requirements in "concrete" at the beginning of the project. Rather, Agile refines the high-level requirements defined in the beginning of the project throughout the lifecycle. Thus, the costs are constantly being refined too. This policy is an issue needing attention by the PMO if Agile is to be used.
Enclosure 1 Additional Policy, Section E.1.1.5 Cost Realism "Contractors shall be encouraged to submit cost proposals that are realistic for the work to be performed.… Proposals shall be evaluated for cost realism in accordance with the Federal Acquisition Regulation."	**Challenge** The PMO would need to be convinced that the contractor is submitting realistic costs for an Agile project, given that the basis for estimating in Agile is different from the usual basis for Waterfall. Agilists tend to think in terms of fixed cost and floating requirements, concepts that are counter to traditional PMO thinking.

Enclosure 1 Additional Policy, Section E.1.1.8 Independent Operational Test Agency (OTA) "Each military department shall establish an independent operational test agency… to plan and conduct operational tests, report results, and provide evaluations of effectiveness and suitability."	**Challenge** The OTA needs to coordinate with the Agile team. The normal mode of doing business for the OTA will most likely have to change to accommodate Agile and the timing of available deliverable code. This should be coordinated in advance of the program start if at all possible since the test (OTA) personnel need to be part of the Agile team or at least an interfacing team. This would impact acceptance testing (scheduling of it, etc.)
Enclosure 1 Additional Policy, Section E.1.1.9, E1.1.10, and E1.1.13 Information Assurance, Information Superiority, and Interoperability, respectively.	**Constraint** These sections do not directly affect the use of Agile but do provide some constraints that need to be considered for the "bigger picture" or architecture of the entire program. Agile tends to develop small, focused pieces of functionality. However, these smaller pieces will need to fit into a bigger picture or architecture for the program, which will have outside constraints or overarching requirements that need to be met. The Agile teams need to consider how every Agile iteration fits within the bigger scheme of the program to avoid rework. One way to solve this problem is to use a composite approach that couples ideas from Agile and the traditional Waterfall to provide coverage for these areas. Another possible approach could be to include an Information Assurance (IA) expert as part of the Agile team. This may or may not be a full-time position but the expert will be needed certainly on a regular and consistent basis. Another possibility is to make sure that these requirements (non-functional) are emphasized in developing and prioritizing the backlog list.
Enclosure 1 Additional Policy, Section E.1.1.11 Integrated Teat and Evaluation "Test and evaluation shall be integrated throughout the defense acquisition process."	**Support** This fits into the Agile concept of test often and deliver a working product at the end of each iteration.
Enclosure 1 Additional Policy, Section E.1.1.19 Professional Workforce "The Department of Defense shall maintain a fully proficient acquisition, technology, and logistics workforce that is flexible and highly skilled across a range of management, technical, and business disciplines."	**Support** This section provides support for training government personnel in Agile if that is to be used on the program.
Enclosure 1 Additional Policy, Section E.1.1.21 Program Stability "The MDA shall determine the appropriate point at which to fully fund an acquisition program, generally when a system concept and design have been selected …"	**Challenge** This section discusses developing realistic schedules, investment plans, and affordability assessments. This suggests a priori design for the program which is counter to Agile. However, this may be an educational issue more than an Agile issue to resolve. A lot depends on the level of information needed to make this decision and the type of system being developed.
Enclosure 1 Additional Policy, Section E.1.1.27 Systems Engineering "A modular, open-systems approach shall be employed, where feasible."	**Support** This section requires the acquisition program to be managed using a systems engineering approach that optimizes total system performance and minimizes total ownership costs. In many respects Agile supports this concept and there should not be any issues if the PMO decides to employ Agile.

4.2 Regarding DoDI 5000.02

On January 7, 2015, the Department of Defense released an updated version of DODI 5000.02, *Operation of the Defense Acquisition System*, and rescinded the interim direction it had issued in November 2013. As with previous versions of this instruction, the barriers to adopting Agile in the DoD appear to be primarily cultural. That is to say, there is little in the way of regulation or guidance provided in DoDI 5000.02 that would prevent the use of Agile. This instruction does impose specific constraints on the acquisition office, but these constraints would be true of any development environment.

While this new instruction includes a figure depicting the "Generic Acquisition Phases and Decision Points" that is clearly a Waterfall-style visualization, it goes on to explain "MDAs have full latitude to tailor programs in the most effective and efficient structure possible, to include eliminating phases and combining or eliminating milestones and decision points, unless constrained by statute." This emphasis on tailoring is perhaps the most significant change in the new instruction. Rather than expecting every project to comply with a single standard process, DoD 5000.02 now strongly encourages modifying the process to fit the product. This opens the door quite wide to adopt and implement Agile methods where appropriate.

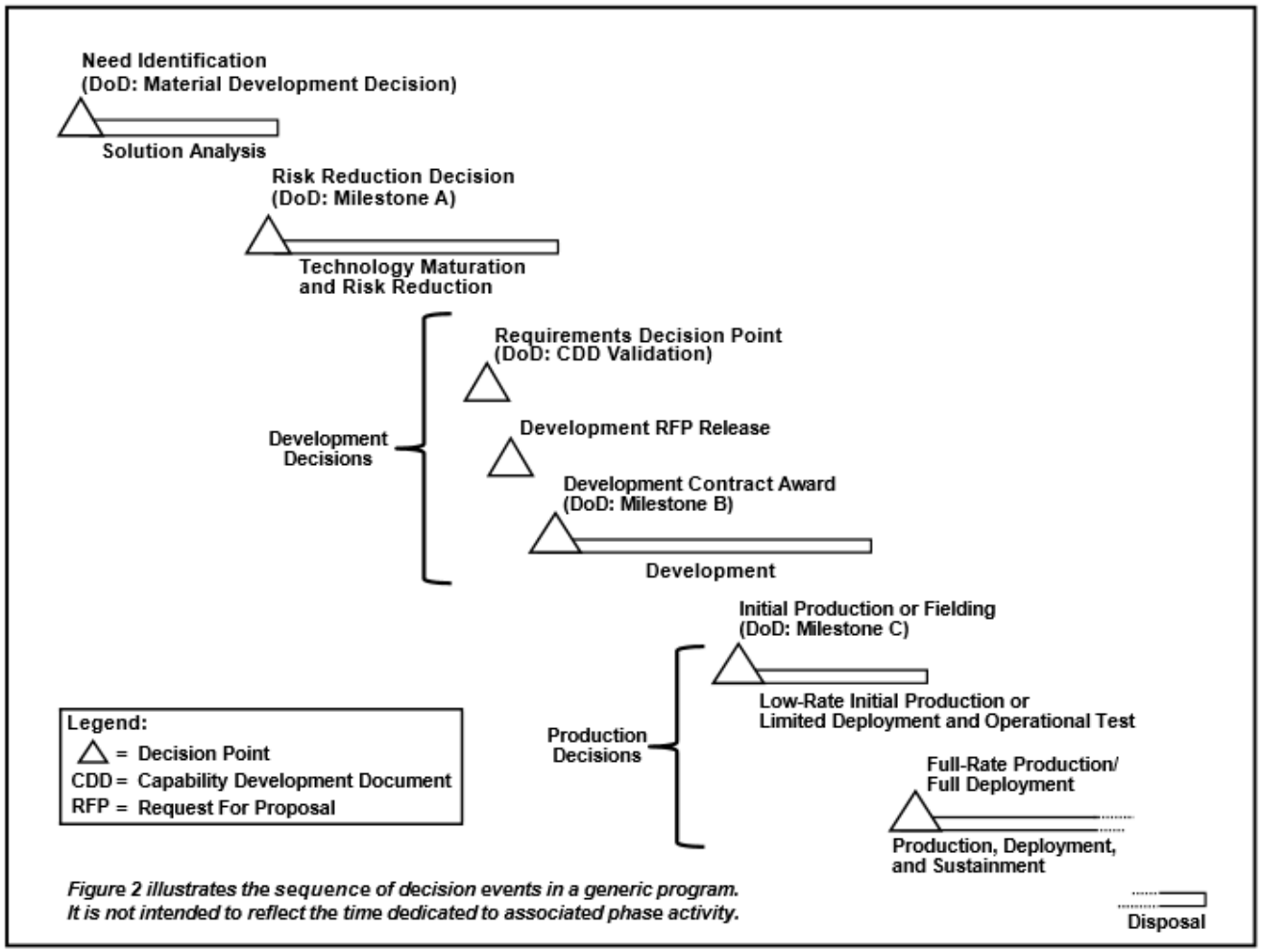

Figure 2: Figure from DoDI 5000.02: Generic Acquisition Phases and Decision Points

To help facilitate and encourage tailoring, 5000.02 presents a collection of acquisition models to choose from, each of which provides a starting baseline approach that program managers and program executive officers should further customize according to the specific needs of their effort. The program models include

- Hardware Intensive Programs
- Defense Unique Software Intensive Programs
- Incrementally Deployed Software Intensive Programs
- Accelerated Programs
- Hybrid A (Hardware Dominant) Programs
- Hybrid B (Software Dominant) Programs

While the Software Intensive model has some similarities with Agile, it seems to rely on predictive planning to a greater degree than is typical for an Agile program and appears less welcoming to changing requirements, particularly late in development.

The Incrementally Deployed model on the other hand is the closest to an Agile approach and is worth considering for DoD programs seeking to use Agile. The instruction explains that this model enables "the rapid delivery of capability through multiple acquisition increments, each of which provides part of the overall required program capability." After offering a caution that the Incrementally Deployed approach might introduce an overwhelming quantity of approval reviews, the instruction suggests a mitigating strategy in which "multiple activities or build phases may be approved at any given milestone or decision point…. An early decision to select the content for each follow-on increment (2 through N) will permit initiation of activity associated with those increments."

Related to this, the Federal Acquisition Regulations (FARs) seem to impose significant obstacles to collaborative endeavors. In fact, since the system tries to encourage fair competition, users often feel they are prevented from collaborating with system developers until late in the acquisition lifecycle. However, a closer reading of the FAR reveals greater flexibilities than are generally recognized. For example, FAR 15.306(d)(4) actually encourages collaboration with bidders in the pre-award stage of an acquisition program and provides guidance on how the government can and should help make bids more competitive. See Appendix F for a more detailed summary of this and other FAR sections.

Further, the mechanisms that are typically imposed by acquisition offices to monitor and control their system developers (such as earned value, or independent cost estimation) are significantly different when the developer is working in an Agile world. As stated earlier, the differences between using Agile and a more traditional method require different management approaches for the advantages of Agile to be fully realized.

4.2.1 Agile Impact to Acquisition: Scenarios

As a basis for discussing how Agile might impact a typical acquisition, let's look at two very simplified scenarios: a non-Agile software acquisition and an Agile software acquisition. These scenarios primarily relate to an acquisition initiated during the Engineering and Manufacturing Development phase of the acquisition lifecycle.

Non-Agile Software Acquisition

In this scenario, a capability document would be created and then a standard government contracting process would be used to select a contractor. The contractor would follow a standard development process that produces the requirements, specifications, and designs that would be reviewed and approved by the acquisition office. Typically, users participate in milestone reviews that would accompany the contractor's development phases. Once the reviews were complete and the requirements, specifications, and designs approved by the acquisition office, the contractor would begin the software implementation. Once the software development is complete, the system would be integrated and go through system and acceptance testing. The first time the acquisition office and users get to try the software out to see if it really works as they want it to is during this testing, which might be many months or even years after the contract was signed.

Agile Software Acquisition

This scenario would start out the same as the non-Agile situation. A capability document would be produced and used as the basis for selecting a contractor. However, from this point on, things would be different. Instead of the contractor producing a series of requirements, specifications, and design documents to guide the implementation, the government would provide a "user representative" to the contractor. This user representative would be a member of the development team and would support creation of a set of user stories[28] that describe the user capabilities in terms of simple features (that the users need). The user representative would then collaborate with the development team to prioritize these stories. The team would begin implementing the stories by first selecting stories with the highest user priority and those stories they could implement in a short (typically two- to four-week) iteration. At the end of each iteration, the team would produce a working system that implements all the user stories that have been completed to-date. The acquisition office and users would be able to try the software out at the end of each iteration, providing feedback to the development team. They could even add new user stories, change or delete existing user stories, and reprioritize all the user stories at the iteration boundary. This process of short iterations continues until all the user stories are completed or until the acquisition office and user agree that the system is good enough; then the system goes through final testing.

Some key differences between these two scenarios are

- Instead of producing a complete, detailed design up front, the Agile team begins with a skeleton architecture/design up front; the architecture evolves over the iterations.
- The Agile process produces a testable system at the end of each iteration that the users can try out (in contrast to non-Agile processes that typically do not provide a user testable system until software development is fully completed).
- The Agile process accepts changes at the start of each iteration. Coupled with the user's evaluation of the testable system at the end of each iteration, this approach keeps the Agile development team focused on what is most important to the users.

[28] This assumes that an overall release plan if needed for the project has already been created. The release plan could be at the feature or capability level.

4.2.2 Agile Within The Acquisition Lifecycle Phases

This section has been deleted from this 2016 update, as it addressed an obsolete version of DoDI 5000.02.

4.3 Foundational Concerns

While policies, regulations, and other governing documents are a large concern for anyone in a PMO thinking about adopting Agile, there are some other underlying concerns that form the basis for developing the application of Agile. *The most significant of these are culture, training, and customer interaction.*

Culture

Culture is inherent in any organization; in many ways it reflects the methodology being used to develop the product. It is a mindset, a way of thinking, and a way of doing business. Culture becomes ingrained into the organization and is usually intertwined with everything the organization does. This includes the organizational structure, the rewards system, the communication style, the decision-making style, and the staffing model (types of personnel, roles and responsibilities, team composition, etc.). The Agile culture is quite different from the traditional Waterfall culture. This in itself could be a huge obstacle to the adoption of Agile.

Successfully adopting Agile requires a change in culture, not just procedure, and meaningful culture change is notoriously difficult. However, culture change is possible, even within a large, formal enterprise like the DoD, and certain strategies can help facilitate the change.

The diagram in Figure 3 first appeared in a 2013 paper titled "Changing Acquisition Culture: What and How," published by the Center for National Policy.[29] The article lays out a plan for introducing specific elements into the Pentagon's acquisition establishment to build a culture that places a premium on speed, thrift, and simplicity. The plan involves using the four influence channels in Figure 3 (Leadership, Literature, Education and Training, and Peers) to reinforce an enterprise-wide message about the desired culture. A similar strategy could be used to support a cultural shift towards Agile.

[29] See http://cnponline.org/p/changing-acquisition-culture/.

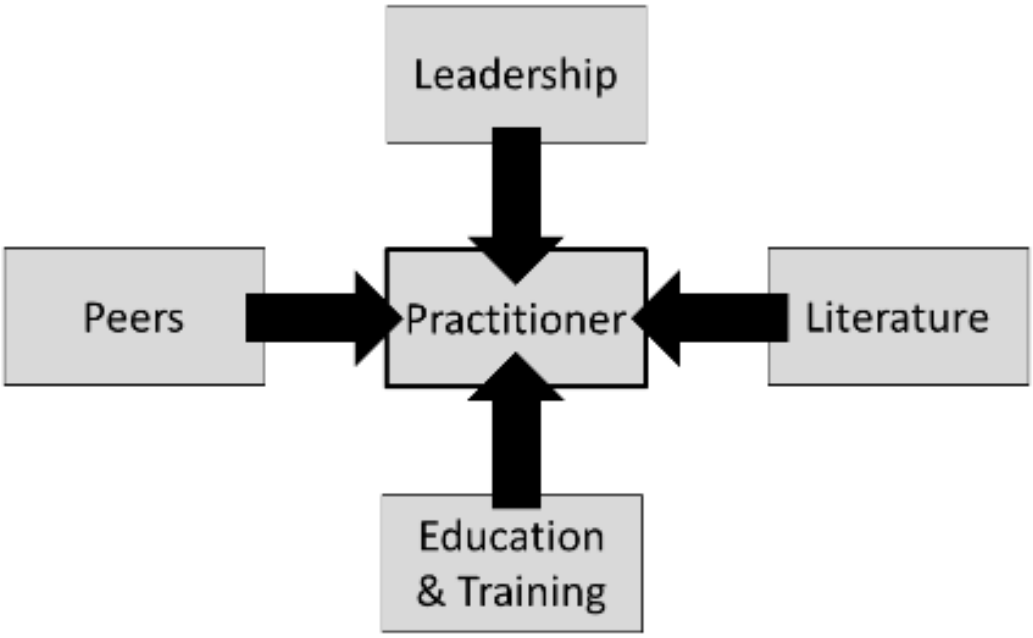

Figure 3: Cultural Influence Channels

The *Leadership* channel includes formal elements such as measures, incentives, and policies, as well as the informal messages and communications that leaders deliver. This channel is not limited to executive leaders at the top of the pyramid, who sometimes have less influence on culture than the front-line leaders, but instead includes the full range from project-level technical leads up to senior OSD officials.

The *Literature* channel is made up of professional journals, blogs, magazines, books, conference proceedings, and other sources of professional source material. These constitute a collection of fresh ideas, conventional wisdom, and best practices.

The *Education and Training* channel spans the widest range of time and origins, as it includes everything from undergraduate courses and four-year degrees to executive education programs and one-day seminars, from mandatory certifications to optional electives. Within the defense acquisition community, the main source of education, training, and certification is the DAU, which is augmented by the Air Force Institute of Technology and the Naval Postgraduate School. The Leadership channel directly affects this path by defining mandatory training and education requirements, establishing curriculum, and setting policy on certifications for various positions.

Finally, the *Peers* network is the least formal, hardest to influence, and arguably the most impactful. It consists of hallway conversations, peer-mentoring, rumors, and war stories that make up the daily information flow on "how things are done around here." This is where culture is largely defined, developed, and shared. It overlaps with several of the other channels, as colleagues tell each other about an article they read, a course they took, or a policy their boss instituted. Any attempt to foster a specific culture must account for the impact of this channel.

A comprehensive culture change strategy involves leveraging all four of these channels to reinforce the desired values and behavior. Establishing an Agile culture might include leaders publishing policies and tracking measures related to Agile while the schoolhouse incorporates Agile more fully into the existing

training curriculum or adds new courses to the pipeline. The culture would be further reinforced and established by creating opportunities for practitioners to come together, in person or online (or both), to connect and collaborate. Advocates may write or commission articles on Agile, or even create new outlets to publish and propagate Agile-related material. And the more shareable and effective all of those elements are, the more likely the peer network will spread them around.

Training

Training is essential when adopting Agile methods: while the constructs and principles seem readily apparent and easy to understand, the actual implementation is more difficult than one would think. Consider that the PMO will be asking people to change habits that have become ingrained over years or decades; PMOs will ask them to change the way they do business, conduct their work, and spend their days. Many times the staffing profile of personnel who thrive in an Agile environment is totally opposite to that of those who thrive in a Waterfall environment. People can adapt, but this type of fundamental change is not easy.

Further, there is a significant difference between training and education, so Agile training should not be limited to the classroom. While instructors delivering seminars and lectures are a useful first step in establishing a baseline awareness of Agile methods, real-world coaching and guided applications are necessary to help practitioners establish and develop these new skills. Finding an Agile coach to help the organization move to Agile is essential.

Customer Interaction

One of the key tenets of Agile is access to the customer—the end user; this is essential to the Agile way of doing business. In a government acquisition environment, access to the end user is not always possible. In many cases, there are multiple end users for the product. Staffing this position is problematic due to resource availability, representation of all users, and the type of personnel typically available for this type of interaction.

These concerns need to be addressed by the organization before it begins using Agile. Some ideas on how to do this are provided in Section 5, with common objections to Agile in Appendix B, and areas to consider when embarking on using Agile in Appendix C.

5 Considerations for Applying Agile in the DoD

> *Neither agile nor plan-driven methods provide a methodological silver bullet that slays Fred Brooks' software engineering werewolf.… Elements of both agile and plan-driven approaches may be characterized as lead bullets*. –Barry Boehm and Richard Turner [Boehm & Turner 2004]

For those who are looking to Agile to solve all their software woes, be aware—that particular nirvana will not be presented here. Agile is just another "lead bullet" in the arsenal of methods, practices, techniques, and procedures that can be used to help solve software woes. One statistical study concluded that "little empirical research had been conducted in establishing whether customer satisfaction in the use and results of Agile-driven software development methods was greater than the customer satisfaction in the use and results of plan-driven software development methods" [Buresh 2008]. This study went on to say that "both methods satisfy their respective customers under a wide range of different situations." Thus, like any technique or approach, Agile must be applied appropriately and will require discipline.

During the 2009 Agile Development Practices Conference, Alistair Cockburn said during his keynote speech that the concepts of Agile were not new. He went on to say that the concepts of Agile were ones that had been used successfully over the years, and the Agile Manifesto gathered and documented them.

While reviewing multiple references on Agile, we found that indeed, the concepts used in Agile are not new. Some were used as early as the 1950s and through the 60s and 70s, and on into the 80s [Rico et al. 2009]. The Agile Manifesto gathered and documented the ideas, and the Agile movement promoted them for the betterment of software development and added value to the end user.

Some might ask "If these concepts are not new, then what's the big deal?" Upon close inspection, there are new components (ideas, practices, theories, etc.) and new combinations of those new components with "old" components. The explicit value statements used within Agile are also new. In addition, the practice of Agile is new in that it is now becoming more widely employed with demonstrable benefits. From a DoD perspective—or that of any large organization for that matter—the paradigm behind using Agile is significantly different than "business as usual." Business as usual tends to be known as Waterfall or what Boehm includes under a broader definition as a plan-driven method. The mental models for using Agile or Waterfall are very different. For instance, Waterfall says to define all requirements in advance, but Agile says this is impossible and futile because users don't really know all their requirements until they see a system in operation.

To further differentiate between the two paradigms, plan-driven methods' goals are predictability, stability, and high assurance. These can be thought of as *strategic objectives*. On the other hand, Agile goals are rapid value and responsiveness to change. Agile can be thought of as having more *tactical objectives* [Boehm & Turner 2004]. However, this is no reason to prevent an organization from using Agile as part of its strategic approach to solving problems.

There is a culture that emerges around any methodology. The culture for plan-driven methods is different from that of Agile. Neither culture is better or worse than the other, just different. For those choosing to move to Agile, the first thing that must be understood is that it won't be "business as usual," and the PMO will need to change its collective mindset, its paradigm, and its culture.

Jim Highsmith, one of the Agile Manifesto signers, has said there will be barriers and impediments as an organization moves to Agile. For large companies, it can be a multi-year transformation. The PMO needs to determine if Agile will be a match for what it wants to do. Is it a strategy for your project, your division, your whole company? The PMO needs to determine how proficient it will be at change—organizational change [Highsmith 2009a]. Be prepared for organizational change management issues.

Since organizational change is always difficult, why would anyone want to embrace Agile? We did not perform an in-depth look at the various statistics on the benefits of Agile. However, the reports and literature about Agile's performance we did look at varied from extremely impressive, which some might think a little too good to be true, to moderately good. One example shows that by using Agile, costs decrease from as little as 5 to as much as 61 percent, with schedule decreasing from as little as 24 to as much as 58 percent, and cumulative defects decreasing from as little as 11 to as much as 83 percent [Highsmith 2009b].

Even if one is skeptical and believes only the lower end of these statistics, Agile beckons to be tried to reduce cost and improve benefits and quality for the DoD. Before jumping into the Agile world, take time to consider how Agile can benefit your program, what the issues will be, and if perhaps a hybrid or composite approach is best.

Some of the concepts that need to be considered when embarking on the use of Agile are discussed below. The discussion assumes the government will be contracting with a firm to actually do the development. Since the contractor will be creating the organization structure, it is important the government understands what it is and how they interact within that structure. The better the understanding, the less likely there will be inadvertent roadblocks or obstacles created to impede the progress of the Agile team(s). If the government is doing the development internally, some of the actions may differ and would be accomplished by the government. We considered the following concepts:

- acquisition lifecycle
- team environment, including the specific Agile method, team communication, distributed teams, size of the program, potential encapsulation
- end-user access
- training and coaching
- oversight, including milestone reviews, documentation, evaluation (measures)
- rewards and incentives
- team composition
- culture

Some of the discussion will sound familiar as it parallels feedback we obtained during our program interviews in Section 3. This is not surprising as the concepts were actual issues the programs dealt with during their use of Agile. The concepts discussed here overlap and are intertwined. In many cases, the concepts are mutually reinforcing.

5.1 Acquisition Lifecycle

The acquisition lifecycle consists of multiple phases: Materiel Solution Analysis, Technology Development, Engineering and Manufacturing Development, Production and Deployment, and Operations and

Support. Each of these phases presents unique challenges and opportunities. Some phases lend themselves to the use of Agile better than others. Agile was used on the programs interviewed, spanning all the lifecycle phases except the Materiel Solution Analysis phase. However, how Agile was employed varied from program to program. The PMO should determine how to best employ Agile in its program depending on its specific situation. In the following paragraphs, we propose questions to ask and identify issues to consider in building an Agile program.

If the PMO is doing an RFP, no matter which phase, ensure that the RFP contains language that allows the use of Agile. In many instances, traditional RFP language makes it difficult, if not impossible, to propose an Agile-based solution. One consideration is the types of reviews and documents required. If the PMO wants to employ Agile, be prepared to allow for "Agile style" document development (i.e., incremental development of documents and data for reviews that result from the individual iterations and/or releases). This might not seem much different from what the traditional methods provide, but consider the level of detail may be sparser using Agile in the earlier versions of the documents. Even final documents might not contain the amount of detail provided in traditional documents. The key here is not the volume, but the content. A necessary and sufficient criterion is that all important data required for operation and maintenance of the system are supplied.

5.2 Team Environment

Earlier in this report we discussed findings in an area called infrastructure. By infrastructure we mean the structure of the team and the context within which the team operates. Organization structure and environmental support structure both need to be established to support an Agile implementation. The context of a program and its inherent organizational structure are related.

For this report, we made the assumption that we were dealing with software-only or software-intensive systems. Many systems contain software and could be considered software intensive, but the software is only a small part of the overall system and certainly not the end item being procured. For large systems acquiring end items such as tanks, ships, planes, or satellites, the Agile software team may need to be encapsulated from the rest of the program.[30] This would entail determining the boundaries or interfaces to the rest of the system and using those as constraints to create the boundary for the Agile software project. These would become constraints for the software development and would be part of any working assumptions for the software environment. For instance, the software could be developed and tested within the Agile environment but then "delivered" to that full system for system test within the tank, missile, ship, etc.

Due to the size and complexity of most DoD programs, multiple iteration teams will be needed. The number is dependent upon the program and in some instances the locations of the contractor team. The larger the number of teams, the more complicated the communications and the greater the need for more users to be involved. In an ideal situation, each iteration team would have access to its own dedicated end user. However, that is not practical in the DoD environment, so alternatives need to be employed. Consider the use of proxies, rotating personnel every x weeks (x usually is two to four weeks), or perhaps a separate "team" of subject matter experts (SMEs) accessible by the iteration teams as needed.

[30] For other systems, such as AIS, this is usually not an issue.

The structure of the overall program team—especially the contractor team—is dependent upon which Agile method is chosen. Methods such as pair programming and scrums are just two examples of these different practices. Typically the contractor determines the "flavor" of Agile. However, the government team needs to be responsive and supportive of that method. Otherwise, using Agile will have less than optimal results.

The Agile team also must exhibit behavior reflecting the approach. Leffingwell describes seven practices observed to scale up to enterprise-level development projects, and we have adopted his terminology for this summary treatment [Leffingwell 2008]. A more detailed treatment of these practices is reserved for future work.

The Define/Build/Test Component

Three basic skills are combined in the component team: define, build, and test, operating cooperatively within a pre-defined period known as a *time box*. The juxtaposition of these skill sets into one team tends to run counter to some conventional methods employed in DoD programs, where these players are often separated by intent.

Two-Level Planning

Two-level planning is portrayed as providing both guidance of how software is to be inserted into the operational environment as well as allowing some flexibility to accommodate what is learned during development:

- The "top level" of the planning cycle is termed *release level planning*. This cycle of planning defines series of releases that broadly define the capability to be contained. This could be done at the feature set level.
- The "second level" of the planning cycle is termed *iteration level planning*, where iterations break the release into a set of iterations that can be time-boxed.

Mastering the Iteration

The ability of a team to reliably execute a sequence of iterations may well be the key behavior that distinguishes a team capable of exploiting Agile techniques in a large organization. If this capability is not present, the likelihood of success is minimal at best.

The iteration consists of the following key activities in a small time box:

- creation of complete, tested, working code that implements a set of features
- integration of the developed code into the working baseline within the timeframe of the iteration

The result of a given iteration is potentially releasable to the customer.

Producing Smaller and More Frequent Releases

It is clear that one natural effect of the expectations is to desire more frequent feedback from the customer and/or stakeholders to avoid large-scale course corrections. Shorter iterations will help to maintain more or less continuous feedback from the customer. In particular, for feature sets that may evolve due to improved customer understanding of needs, this model of short iterations offers a more timely alternative.

Concurrent Testing

Concurrent testing practices are based upon thorough testing of code both during development and during integration. The goal is that all code is tested. Gamma[31] and others advocate a "test first" developmental approach [JUnit 2015] where the unit tests for software are created prior to actual development. Gamma also advocates frequent use of the unit tests during actual development.

Continuous Integration

Continuous integration may well be the most useful and controversial practice advocated in the Agile community. The continuous integration model diverges from the usual V-shaped model advocated by traditional systems engineering practice employed in DoD programs. In the V-shaped model, requirements synthesis, allocation, and development are carried out in a top-down fashion. This is followed by a bottom-up sequence of integration and verification activities, leading to a product ready for use acceptance or sale.

Continuous integration of software is contingent upon the ability to concurrently execute two crucial activities: (1) collect incremental changes from multiple developers on a regular basis, ideally daily, and (2) perform the "nightly build" discipline, where all changes are brought together in an incremental software baseline, which is in turn compiled and tested with the available unit and regression tests.

Regular Reflection and Adaptation

Reflection and adaptation (sometimes called the *retrospective*) is the Agile version of "continuous improvement" that is highlighted in other methodologies. In keeping with the bottom-up discipline of Agile approaches, this introspection is driven down to the individual team level.

5.3 End-User Access

One of the tenets stated in the Agile Manifesto is "customer collaboration over contract negotiation." Agile implements this concept by having continuous contact with the end user. Typically, an end user or his or her representative is an integral part of the iteration team. This approach is not always practical in the DoD environment and can be complicated by the fact that some programs are joint programs involving more than one service. With multiple end users, all with different ideas of what the end product should be, it will be difficult to have a single voice for the end user. Also, the real end user is an operational person, not an acquisition person, so meeting this Agile requirement is challenging.

Traditional acquisitions try to have user inputs, with their success varying depending on availability and a host of other issues. Typically, the acquisition organization speaks for the end user. Thus, they become the proxy for them. In addition, due to contractual rules, only certain people are warranted to talk to the contractors—these are the people who can legally direct the contractor. In Agile, the end user who sits with the iteration team speaks for the program and has the authority to commit. This leads to potential

[31] Erich Gamma is a Distinguished Engineer at IBM Rational Software's Zurich lab. He is a coauthor of the first comprehensive book on design patterns [Gamma et al. 1994], was a key contributor to the development of the Eclipse software development platform, and led the development of the design patterns employed in the JUnit and related testing infrastructures.

constructive change issues within the DoD arena. It is important to note that no one we interviewed experienced this issue; we mention it here only as a caution and as a potential for changing the contracting officer's skill set.

Agile in its pure form insists on interactions with the real end user. This interaction will surface end-user disagreements earlier in the project and in the concrete context of demonstrable capability. To overcome this, the PMO and the contractor may have to consider surrogates or proxies. Depending on the PMO's experience, the use of surrogates may require a culture change—one that may or may not work well.

Another alternative is to use remote collaborative presentation capabilities—as well as wikis, blogs, and live chat—to keep travel costs from being overwhelming. The challenge for DoD programs is that some of these approaches are not always approved for DoD use, so the program may have to take the lead to get them approved. Another challenge for some organizations is the cultural shift from formal face-to-face review to collaborative, virtual meetings.

5.4 Agile, Constructive Change, and Ratification

In standard contracting, only a warranted contracting officer (CO) can give formal direction, telling the contractor to perform work or change their approach. Policies and regulations tend to discourage COs from granting after-the-fact approvals (known as ratification) to unofficial change requests that come through other channels. The result of this policy is that change requests are generally handled formally, via contract actions, which adds time, cost, and complexity to the overall effort. This formality is incompatible with Agile.

The primary reason for these restrictions is that engineering changes tend to have significant cost and schedule implications on defense acquisition programs. Limiting the ability of non-COs to direct the contractor aims to prevent undesirable cost increases. In practice, however, most contracting officers insist that even a change with no effect on cost or schedule requires their involvement, which is arguably an over-application of the principle.

The best COs are able to quickly resolve change requests, but even the best COs require some amount of time to do so, and the accumulated delays can be significant. This review and approval process tends to reduce the project's overall agility, particularly when applied to low-impact changes.

Preventing expensive rework, wrong-work, and waste is, of course, part of Agile's objective as much as the traditional approach, but the difference is that Agile actually welcomes requirement changes, even late in the development. Agile manages these changes organically and directly rather than through formal brokers like COs. It does this by making end users a key part of the team and empowering them to provide feedback and direction. This approach works because Agile programs test the software's performance early and often rather than all at the end. The team is therefore able to make a steady series of course corrections along the way. Keeping decision-making authority as close to the action as possible allows the development team to make substantive positive change to the effort without major impact to cost or schedule.

The DoD tends to be skeptical of Agile's approach to change management because under a Waterfall approach, change is expensive (and late change doubly so). However, it turns out that *not* changing is even more expensive and problematic, producing systems that are operationally irrelevant, technologically obsolete, or both. Agile suggests that program developers should direct their skepticism towards stability rather than change, and should not tolerate inflexible, frozen requirements that were established when the

people involved know the least about the effort. Agile also insists that these changes are best handled via conversations rather than contract actions.

Agile relies on a nuanced and mature understanding of what drives cost and schedule problems in modern software development efforts. It recognizes that ignoring or delaying user feedback causes more problems than allowing such feedback to flow directly to the contractor. To be sure, cost growth is a genuine risk that needs to be addressed, but restricting authority to warranted COs is not the only way to address it.

One promising approach that is consistent with Agile is to make sure the original contract is written with Agile in mind and contains sufficient flexibility to permit a wide scope of activity that could be modified as the situation develops. Agile program managers (PMs) could establish contract vehicles that allow for collaborative discussions to resolve and address dynamic developments over the life of the effort.

In a properly scoped Agile effort, these changes would not constitute an expansion or deviation, or a constructive change, and thus would not require a ratification. The challenge is to craft contract language and requirements with sufficient flexibility to accommodate the types of changes Agile welcomes. Defining specific contract clauses and organizational structures that allow program teams to "welcome changing requirements, even late in development" is beyond the scope of this report.

5.5 Training and Coaching

Training for Agile is essential. While the concepts of Agile are not new, the specific implementations contain subtleties and nuances that need to be explained. Additional training in the specific contractor method is also a must. Training before starting the project will help to avoid inadvertent roadblocks and prevent some of the more common issues from arising.

Many contractor organizations employ a coach to help them convert their processes to support Agile. A coach and/or Agile advocate who has "clout" within the PMO is a good addition to the PMO staff. Their presence can answer daily questions, help resolve issues before they become problems, and help to ensure the program runs smoothly from an Agile perspective. A word of caution: An Agile advocate or coach without authority is like not having an advocate at all; they get lost in the chorus of voices demanding to be heard. Keep in mind that the Agile coach for the PMO will have a different role than an Agile coach for the development team. The PMO Agile coach will be there to help the acquisition organization understand what the developers are doing and assist in making both the acquirers and developers work better together.

5.6 Oversight

The existing traditional structure is in place to provide predictability, stability, and high assurance [Boehm & Turner 2004]. The essence of the traditional structure is created to allow for close oversight and insight into the workings of a program. The structure requires immense amounts of documentation, which is evaluated at key milestones throughout the program. These documents and their review along with the accompanying capstone events (PDR, CDR, etc.) provide the government with a high level of "comfort" that the program is progressing the way it should. The traditional Earned Value Method System (EVMS) of measurement is also constructed to provide the government with a means to monitor the progress of the program. This system is rigid and monitors progress against the plans in both cost and schedule. These plans are reflected in the IMS.

Agile is very flexible and promotes the capability of moving tasks and functions from one iteration to another or even deleting them altogether. This fluid environment is very difficult to track using EVMS as implemented today. The fluid environment also makes it difficult to maintain a current IMS.

An analogy might be useful to understand the kind of oversight expected within Agile. In the military there is something called commander's intent:

> *Commander's intent describes the desired end state. Your intent statement provides a framework for the operation. It does not tell your soldiers what to do. It does give them the overall picture of what you say the company needs to accomplish to be successful. By making your intent a clear, concise, and focused statement, you greatly increase the chances that your soldiers will continue the mission, even when the operation doesn't go as planned.*[32]

One can think of the overall plan for an Agile program as its intent. If the initial plan does not work as thought, then the development team alters the plan with the intent still in mind. Allowing the Agile team to follow the intent without detailed direction is based on trust, collaboration, and relationship building.[33] These ideas are core to Agile.

One often hears that Agile is ad hoc and has no planning. Do not confuse formality with discipline. Agile teams tend to be less formal but are highly disciplined. Combining this with the above discussion means that Agile requires considerable planning if the program is to achieve its objectives. However, more of the planning is done at the mid and low levels of the program versus at the high management level on traditional programs.

The issue of how oversight will function on an Agile-based program must be resolved before you start the program. Both the government and contractor need to agree on the method to be used. EVMS can be used for Agile programs but it requires close coordination between those who monitor the EVMS system and those who maintain it. If a capability or task is swapped out for an equivalent task (equivalent in EVMS value) then it could be used. This becomes labor intensive. Other ways of monitoring Agile programs can be used, such as using completed stories or accrued value to the user as measures.

In his seminal 1970 paper on software development, Royce explained that his view on how much documentation to produce was to write "quite a lot." This stands in stark contrast to the Agile approach to documentation, which recommends producing "just enough" to meet the need and provide continuity for the team. This "just enough" approach is usually not sufficient for the capstone reviews, which tend to default towards Royce's perspective. This is another example of how culture shapes behavior and determines outcomes.

Interestingly, acquisition policy is agnostic on the question, although it increasingly leans towards Agile's minimalist preference. The current procedures and policies allow a team to produce "quite a lot" of documentation or to produce "just enough." The factor that determines which approach will be followed comes down to the cultural preferences of the people involved.

---

[32] See http://www.globalsecurity.org/military/library/report/call/call_98-24_ch1.htm.

[33] Further discussion of these topics is left to future work.

Remember that documents are evolved in increments within Agile and that this will have an impact when the complete document is available. Another documentation challenge in the DoD acquisition environment is maintaining enough documentation of the critical architectural information and decisions so that the knowledge can be effectively transitioned when personnel—military or contractor—leave the program. Thus, an understanding of the content and when a final version of the documentation will be available needs to be negotiated in advance. Some items that can influence these negotiations include the opportunity cost for how much time is spent on documentation versus creating working software and the impact of a feature review of the working software, which provides the PMO with an indication of progress instead of reviewing just the documents.

Capstone events, like CDR and PDR, are also issues in the Agile world. One of the programs we interviewed broke the capstone event into smaller IDRs, which cumulatively equaled the overall capstone event. Again, this work needs to be planned in advance for both technical and programmatic reasons.

5.7 Rewards and Incentives

In the traditional Waterfall methodology, typical rewards and incentives are individual based rather than team based. Contracts, team organizations, and other program structures are developed and interpreted to enforce and enable individual awards. The Agile environment is more team oriented and does not thrive well within the traditional reward structure. While this will not be a large concern for the government unless it is doing internal development, anything the government can do to incent the contractors and support the Agile culture is a major advantage.

The government may want to consider incentives that involve embracing and fostering change and sharing data at the enterprise level. One of the problems with making the cultural shift to Agile is that the right incentives are not in place to foster change. Personnel need to be incented to do significant adoption planning and develop a strategy for the technology shift and related business, legal, and operational aspects. If people are incentivized the right way, they will embrace change.

Reuse and information sharing across the enterprise are important measures according to Agile. This means that you want to incentivize doing things “for the good of all,” not just for the good of an individual program. The problem is that right now DoD programs are structured to compete with each other. This creates a culture of hoarding knowledge for competitive advantage. The DoD needs to think about how to incentivize collaboration across programs even between competing contractors; this will not be easy and may mean that the DoD might have to think about ways to fund programs other than funneling money to a single program.

5.8 Team Composition

One addition to the typical traditional DoD PMO is an Agile advocate. As described in training and coaching, the advocate is someone who can provide real-time answers for the immediate Agile issue. Another addition to the typical staff is an end-user representative who is empowered to make decisions that are binding for the development. Given the nature of government contracting, care must be given to ensure that this user representative has the legal authority to direct the contractor. We can envision a situation where constructive change could become an issue.

The background of the team members may be slightly different than the norm. An ideal Agile team would consist of experienced, high-performing Agile developers in all positions. It is a proven fact that Agile

teams are more successful with more experienced and skilled team members, so it is important that the government provide environments that are attractive to high-performing individuals. More experienced personnel would have more skills than just coding. They must be able to work from user stories to design, implement, and test features. The government also needs skilled Agile personnel to review the documentation and understand how the Agile software development approach works. Many traditional PMO teams do not have software representatives experienced with modern software development approaches. That could be more problematic in an Agile environment, where any shortfalls quickly become more visible.

Another challenge is keeping high-performing Agile teams together long enough for them to achieve peak performance. This is a challenge because developers can change at the end of a contractual period of performance. The continuity of an Agile team enhances the "tacit knowledge" of the program and this improves overall performance. One recommendation might be to look at putting key Agile technical developers or technical leads under a separate contract vehicle or hire them to work for the government organization itself.

5.9 Culture

One of the most frequent topics of discussion that was heard at the 2009 Agile Development Practices Conference dealt with culture. People talked about how Agile would not succeed if an organization's culture did not support it.

As we said before, culture is the customary knowledge, beliefs, behavior, and traits displayed by an acquisition organization or contractor. The government is heavily invested in the use of plan-driven methods for acquisition of all equipment and systems, whether they are software intensive or not. As a result, the culture of the DoD acquisition community (and that of long-time DoD contractors, as well) is comfortable with Waterfall and skeptical about Agile. Part of this comfort is how project management has been trained to manage change. Traditional project managers focus on following the plan with minimal change, but the Agile manager focuses on adapting successfully to inevitable changes [Highsmith 2009a]. Since neither Agile nor plan-driven approaches fit every problem, a key to changing the culture is to make it so that it is flexible enough to accommodate both Agile and Waterfall—and anything in between.

For the DoD to successfully employ Agile, it needs to embrace a culture change. The way it thinks about oversight, documentation, team structure, user interaction with the development team, and flexible change must be altered. Changing a culture—any culture—is difficult. It is even harder to change a culture that has strong motivations for control since mission-critical and life-critical systems are involved. A fear associated with safety or mission-critical systems is that Agile does not put enough focus on software engineering practices such as analysis and design necessary to achieve key quality attributes such as performance, security, availability, etc. This can be addressed by the architect and how the architectural requirement stories are prioritized within the team. In addition, some understanding of organizational change management and how groups change will be invaluable. Organizational change discussions are left for future work.

Our research shows that starting small and taking gradual steps into the Agile world helps the transition to the new Agile culture. By starting with a smaller project (thus smaller teams), experience is gained that can be applied to larger programs. In fact, we came across a group within the Air Force that adopted a modified set of Agile concepts and published its own method. It was initially called Fast, Inexpensive, Simple, and Tiny (FIST). We include the FIST Manifesto in Appendix E as an example of what is being done at the *grass roots* level within the DoD [Ward et al. 2008]. As the practice matured and expanded,

the acronym was changed to FIRE (Fast, Inexpensive, Restrained, Elegant) and published in book form by HarperBusiness. An updated FIRE manifesto is available at http://changethis.com.

The FIRE method was included in an Innovative Contracting Case Studies playbook,[34] published by the White House Office of Science and Technology Policy in 2014, as an example of how "Federal agencies are getting more innovation per taxpayer dollar—all under existing laws and regulations." Senators John McCain and Carl Levin included it in a compendium of acquisition reform proposals they published in 2014. FIRE was also cited by Microsoft in a profile of its Torque project[35] as contributing to its success in rapidly developing an app for Android Wear. This is as an example of how an "Agile-adjacent" effort that began in the DoD was able to spread beyond the military and U.S. government into the commercial sector.

[34] See https://www.whitehouse.gov/blog/2014/08/21/buying-what-works-case-studies-innovative-contracting-0.

[35] See http://www.microsoft.com/en-us/garage/profiles/profile-torque.aspx.

6 Conclusion

This small study on the current utilization and future applicability of Agile for software development in DoD acquisitions is meant to whet the appetite of those looking for another tool to solve the ever-present conundrum of obtaining good software quickly and as inexpensively as possible.

Agile methods have existed for many years. In fact, they are based on concepts that have been around for decades, but the methods have not been widely used, especially within the DoD. In recent years, Agile has matured, personnel have become more skilled, and some DoD contractors have started to build internal Agile capabilities and initiated usage on DoD programs. Since the 2010 SEI report *Considerations For Using Agile In DoD Acquisitions* was published, a growing number of military programs have used Agile methods to great effect, and it is beginning to be adopted into policies and guidebooks.

For the complete novice, we provided a short overview of Agile, including the Agile Manifesto and its underlying principles. There are multiple flavors of Agile, and we listed several of the more common ones for the reader's edification. Finally, we provided a limited comparison of Waterfall to Agile. This shows some high-level similarities and differences.

Several existing programs that we interviewed and existing literature we read expounded on the benefits and the pitfalls of using Agile. The general consensus is that Agile is another tool to be exploited, and this will provide benefit to the DoD in the correct environment.

If this is the case, then why isn't Agile used more? We distilled the information we found from interviews into seven areas that address this question. In each area we looked at the issues and some of the solutions that worked. The seven areas are

- acquisition
- knowledge of Agile
- culture
- oversight
- end-user involvement
- integration and test
- infrastructure

A review of the DoD 5000 series showed there are minimal barriers, and none of the challenges are show stoppers. A lot of these challenge areas depend on the interpretation. Unfortunately for Agile, most of today's interpretations lean towards the more traditional methods like Waterfall.

Finally, we looked at other concepts that need to be explored before employing Agile in the DoD environment. Most important from our viewpoint are end-user participation and culture. However, to employ any aspects of Agile, the DoD organization will have to plan for them, anticipate the changes needed in its environment and business model, and apply the hard work to make the changes a reality.

We acknowledge that this report only begins to explore employing Agile within the DoD. During the course of our original research, we touched on a lot of topics, many of which needed further research.

Some of the potential topics from the original report cited for future research, in no order of priority included

- technology—discussion and explanation of different Agile technical concepts and how they apply within the DoD. (Considerable literature and courses are already available on specific Agile methods.)
- 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 one works best with Agile, and incentives
- comparison of methodologies—methods, including plan-driven, Agile, and hybrids, that work best for different types of programs
- benefits from Agile—discussion about how Agile is viewed within the Agile community using risk and a variation of the cost, schedule, and 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

Several of these topics were addressed during the course of the Agile Adoption in Regulated Settings research initiative of the SEI. We hope that as more government programs use Agile, more findings, observations, and lessons learned will be shared. After all, that is what the retrospective is all about.

Appendix A: Examples of Agile Methods

There are many methods that fall under the umbrella of Agile. Some focus on the developer (e.g., XP) and others focus on managerial processes (e.g., Scrum). Most of these approaches are evolving and borrow from each other. Examples of specific Agile methods are listed below.

eXtreme Programming (XP)

> *…A software development discipline that organizes people to produce higher quality software more productively.… XP attempts to reduce the cost of change by having multiple short development cycles, rather than one long one. In this doctrine changes are a natural, inescapable and desirable aspect of software development projects, and should be planned for instead of attempting to define a stable set of requirements. Extreme Programming also introduces a number of basic values, principles and practices on top of the agile programming framework.*[36]

While a stable set of requirements is not defined up front, the overall requirements are defined and refined throughout the program based on user feedback.

Scrum

Scrum is a

> *'process skeleton' which contains sets of practices and predefined roles. The main roles in Scrum are: (1) the 'ScrumMaster,' who maintains the processes (typically in lieu of a project manager); (2) the 'Product Owner,' who represents the stakeholders; (3) the 'Team,' a cross-functional group of about 7 people who do the actual analysis, design, implementation, testing, etc.*
>
> *During each "sprint," typically a two to four week period (with the length being decided by the team), the team creates a potentially shippable product increment (for example, working and tested software). The set of features that go into a sprint come from the product "backlog," which is a prioritized set of high level requirements of work to be done. Which backlog items go into the sprint is determined during the sprint planning meeting. During this meeting, the product owner informs the team of the items in the product backlog that he or she wants completed. The team then determines how much of this they can commit to complete during the next sprint. During a sprint, no one is allowed to change the sprint backlog, which means that the requirements are frozen for that sprint. After a sprint is completed, the team demonstrates the use of the software.*[37]

[36] See http://en.wikipedia.org/wiki/Extreme_Programming.

[37] See https://en.wikipedia.org/wiki/Scrum_(software_development).

Adaptive Software Development (ASD)

ASD replaces the traditional waterfall cycle with a repeating series of speculate, collaborate, and learn cycles. This dynamic cycle provides for continuous learning and adaptation to the emergent state of the project. The characteristics of an ASD life cycle are that it is mission focused, feature based, iterative, time-boxed, risk driven, and change tolerant.[38]

Dynamic Systems Development Method (DSDM)

As an extension of rapid application development (RAD), DSDM focuses on Information Systems projects that are characterized by tight schedules and budgets. DSDM addresses the most common failures of information systems projects, including exceeding budgets, missing deadlines, and lack of user involvement and top-management commitment. By encouraging the use of RAD, however, careless adoption of DSDM may increase the risk of cutting too many corners. DSDM consists of: (1) Three phases: pre-project phase, project life-cycle phase, and post-project phase; (2) A project life-cycle phase subdivided into 5 stages: feasibility study, business study, functional model iteration, design and build iteration, and implementation.[39]

Crystal

The Crystal methodology is one of the most lightweight, adaptable approaches to software development. Crystal is actually comprised of a family of methodologies (Crystal Clear, Crystal Yellow, Crystal Orange, etc.) whose unique characteristics are driven by several factors such as team size, system criticality, and project priorities. This Crystal family addresses the realization that each project may require a slightly tailored set of policies, practices, and processes in order to meet the project's unique characteristics. Several of the key tenets of Crystal include teamwork, communication, and simplicity, as well as reflection to frequently adjust and improve the process. Like other agile methodologies, Crystal promotes early, frequent delivery of working software, high user involvement, adaptability, and the removal of bureaucracy or distractions. Alistair Cockburn, the originator of Crystal, has released a book, Crystal Clear: A Human-Powered Methodology for Small Teams.[40]

Feature-Driven Development (FDD)

FDD is a model-driven short-iteration process that consists of five basic activities: Develop Overall Model, Build Feature List, Plan By Feature, Design By Feature, and Build By Feature. For accurate state reporting and keeping track of the software development project, milestones that mark the progress made on each feature are defined.[41]

---

38 See http://en.wikipedia.org/wiki/Adaptive_Software_Development.

39 See http://en.wikipedia.org/wiki/Dynamic_Systems_Development_Method.

40 See http://www.versionone.net/Agile101/Methodologies.asp.

41 See http://en.wikipedia.org/wiki/Feature_Driven_Development.

Pragmatic Programming

Pragmatic programming follows the principles of *The Pragmatic Programmer* by Andrew Hunt and David Thomas [Hunt & Thomas 1999]. It is itself a kind of "umbrella" development approach, since it advocates that the developer not stick to any particular methodology but choose the methods and techniques that work best in the specific environment.

Lean Software Development

The term originated in a book by the same name, *Lean Software Development*, by Mary and Tom Poppendieck [Poppendieck & Poppendieck 2003]. The book presents the traditional Lean principles in a modified form (as well as a set of 22 tools) and compares the tools to Agile practices. Lean Software Development promotes seven core principles: eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build integrity in, and see the whole.

Other software development techniques have been mentioned by writers as belonging to the Agile family of approaches, including Kanban [Shalloway 2009], Rational Unified Process (RUP), Personal Software Process (PSP), Team Software Process (TSP), and Cleanroom [Boehm & Turner 2004].

Appendix B: Common Objections to Agile

There are common objections lodged against Agile we have encountered in conducting our professional activities, many of which are echoed in the relevant literature. The most pertinent objections and responses are discussed below.

Agile and DoD Software Practices Are Incompatible

There is a widespread perception that Agile is in rather stark conflict with DoD software development practices. There are a number of facets to this objection, which we will address in turn.

There is a perception among many that DoD development practices mandate a Waterfall approach to software development. A careful examination of DoDD 5000.01 and DoDI 5000.02 does not support the contention that this guidance requires a particular software development methodology. Nothing in the FAR appears to prohibit the use of Agile (see Appendix F for an overview of FAR sections that can be used in support of Agile methods). The same can be said of the more systems-oriented DoD Architecture Framework (DoDAF). It is fair to observe that a lot of the published examples and explanations, which are linear descriptions of the processes, can be read to suggest that Waterfall is strongly preferred.

We suspect that a lot of DoD acquisitions take the Waterfall approach rather than some iterative or Agile approach because it is consistent with program skill sets and is perceived as the path of least resistance. Nevertheless, there is no mandate to employ a strict Waterfall development methodology that we have been able to find.

Agile Is New and Risky

Even though Agile as a practice was formally documented in 2001 and had been successfully used for years prior to that, Agile is widely perceived as new and experimental, particularly in DoD circles.

The truth is, Agile is largely a codification of things that practitioners have been informally doing for some time. Iteration and experimentation have long been carried out under the umbrella of *trade studies*, which often are constrained by limited resource availability and time boxes. Trade studies, the development of breadboard or prototype elements are simply kinds of controlled experiments used to reduce risk. Agile is simply another way to orchestrate controlled experimentation and address uncertainty and risk (user requirements, operational environment, etc.).

Agile techniques have been widely employed in commercial software product development, and much of the literature surveyed offers examples based upon such projects. Agile techniques have been cited in some DoD programs (JMPS, SIAP, classified programs). Given that the literature and practice of Agile as a distinct set of methods spans two decades, it seems inappropriate for DoD software programs to dismiss it as it as new and untried. As of the writing of this report, the IEEE was pursuing the creation of standards for Agile development practices, which had yet to be published.

The issue of risk and Agile appears to be a red herring as well. The SEI has a rather unique perspective on the software-related failings of numerous DoD programs. It is fair to suggest that observed program problems and failures include the use of all the major documented software development methodologies and significant variants. With that experience, it is difficult to ascribe incrementally increased risk specifically attributable to the choice of a given software development methodology.[42]

Refactoring Is Incompatible with Stable Work Breakdown Structures (WBS)

One of the primary tactics employed in Agile development projects is refactoring, which is the restructuring of software behavior and structure as development unfolds. The goal of this tactic is to take advantage of improved insight into coupling, cohesion, and maintainability as more software structure is created and evaluated. In most DoD projects, the WBS is a key organizing artifact. Some observers have expressed the concern that refactoring could affect the WBS, which would introduce a level of turbulence in the WBS that might be unacceptable.

This objection is not compelling. It is typical practice that the WBS for major systems reflects deliverable physical end items, and software to be developed for a given end item is treated as embedded within the end item. If current systems engineering practice is pursued with regard to establishing external interfaces for physical end items (including the software aspects of the interfaces), refactoring will be confined to the end item, and any WBS elements addressing the software relevant to the end item can be structured to accommodate iterative development techniques. The remaining issue is to allow for refactoring in the software within the scope of the end item. The most straightforward mechanism to achieve this refactoring is to structure the software elements of the WBS around operational capabilities, not a functional decomposition that posits specific internal interfaces. A structure of this sort will enable refactoring, and disturbances to the WBS should be affected by software changes only when other considerations come into play as well, such as problems in critical item development.

Unwarranted Impact on the Contracting Officer

One difficulty that is relevant to the choice of Agile techniques is its impact upon the program office and the contracting officer that deals directly with software issues.

As has been noted elsewhere, Agile techniques tend to promote project planning and documentation practices that are not within the usual experience base of the program office staff. This is not an insurmountable problem and can be addressed with appropriate training.

The most critical impact on the contracting officer is the need to facilitate the high-tempo user/stakeholder involvement that Agile techniques depend upon to implement more frequent user evaluation than is customary. This model places an uncharacteristic burden upon the program office to make knowledgeable users/stakeholders available on a periodic basis to provide feedback, which in turn places a burden upon the user community to release such people for these assignments. This will ordinarily be a planning, personnel, and inter-organizational issue that will merit

[42] Jim Highsmith comments, “I always admonish people that in the end, politics always trumps methodology, any methodology.”

the creation of a program office to military service entity memoranda of agreement to secure people for these roles. If the program office is operating in a competitive situation, where two or more contractors are pursuing the same work, there is a potential risk of (1) not treating the two contractors identically, and/or (2) users or stakeholders inadvertently disclosing contractor proprietary information. The following are some obvious tactics that may be employed; some of these tactics are not new:

- Contracting officers should be present for all interactions between the contractor and users/stakeholders.
- A program office may form multiple user/stakeholder feedback teams and establish procedural and informational firewalls across those groups.
- Should a single user/stakeholder feedback team be employed in a competitive situation, it is critical it be trained and briefed on organizational conflict of interest policies employed by the program office; it is desirable that the participants have basic knowledge of DoD acquisition conventions (e.g. training in DAU or other acquisition courses).

Interpreting User Feedback as Constructive Change

For reference, *constructive change* is defined as an oral or written act or failure to act by an authorized government official construed by a contractor as having the same effect as a written change order. Such a change must involve (a) a change in performance beyond minimum contract requirements, and (b) word or deed by government representative that requires contractor effort that is not a necessary part of the contract, and (c) getting ratification.

Some have expressed concerns that user feedback as employed in Agile may be interpreted as constructive change. While this is plausible, it seems no more problematic than ordinary feedback and guidance offered to contractors during dry runs and formal presentations for major reviews (e.g., System Requirements Review [SRR], PDR, CDR, and Test Readiness Review [TRR]). As mentioned above, it is important for the program office and the contractor who elects to employ Agile to arrive at a careful agreement on ground rules. It is important that contractual provisions be incorporated that recognize the role of user/stakeholder feedback that may influence architecture/design evolution in future iterations of development.[43] Users/stakeholder teams must be educated about the limitations of their role in providing guidance to the contractor. Contracting officers must always be present at such feedback sessions to avoid problems. Feedback that is more than clarification, which does indeed alter the previously agreed scope, should be handled separately by contracting officers. While it is impossible to predict what happens once a program has degenerated to the point that litigation occurs, such measures should reduce the risk expressed by critics in this area.

Agile Is Too Hard

Agile techniques are different from conventional practice in many DoD and military service circles. However, to dismiss them as “too hard” strikes us as a far too pessimistic stance in view of

[43] SEI architecture evaluation teams currently encounter this issue when they elicit feedback in the form of stakeholder scenarios. They are important to the architecture team, yet need to be reconciled with contractual requirements.

the growing body of commercial experience with these methods. The main source of the perceived difficulty is simply a lack of familiarity (new things are hard to do the first time) rather than any intrinsic difficulty in Agile methods themselves. As with any tool or technique, mastery takes time. Those who have experience with Agile tend to view it as not only more effective than Waterfall, but actually easier. In view of the difficult, painful, and expensive legacy of cost, schedule, and performance problems on software-intensive DoD programs, there seems little to be lost by judiciously employing Agile to potentially improve contract performance.

Appendix C: Areas for Consideration

The following table provides topics that should be considered when embarking on using Agile in the DoD. We have summarized our observations and findings; however, this list is not complete and should be used only as a beginning guideline. All programs are different and will have unique requirements and issues to resolve.

Table 2: Areas to Consider for Using Agile in the DoD

Area of Concern	Consideration
Content of RFP	Consider using RFP language that does not preclude the use of Agile. This includes the type of reviews, and type and content of documents (CDRLs). At this time, we do not know of any model Agile RFP language.
Organization structure	Consider the use of a coach or advocate within the PMO to help understand the Agile structure.
End-user involvement	How will you provide access to end users? Is there more than one end-user group? How do you have a single voice for all groups? How often will the end user be available for discussion with the development group? Will end users attend demonstrations? What authority to commit the program will the end user have? Consider using proxies or rotating SMEs or a SME team. Consider a hybrid approach using the best practices of Agile and traditional Waterfall techniques.
Training and coaching	Has pre-award training in Agile been created? …given? Who will be trained? Does the training include which contractual phases will use Agile? Is a coach available to work with the team? What authority does the coach have? Has your program had training on the specific Agile method your contractor is employing?
Oversight including reviews	What type of oversight (e.g., EVMS or an alternative such as stories or "user value") will be employed? What type of capstone event (e.g., CDR or multiple IDRs) will take place? Do the entrance and exit criteria mesh with an Agile approach? How will the IMS be created and maintained? Is the PMO trained to use the contractor's Agile tracking tool? Is the PMO prepared to relinquish *total* control over how change is managed and which capabilities are developed in the short term versus the long term? Is the PMO aware of the mid- and low-level management focus used with Agile? Is the PMO aware of the potential change in "rules of thumb" for the number of developers and code size?
Rewards and incentives	What types of incentives are provided by the program? Do they support the use of Agile or undermine it? Does the type of incentive for both the contractor and PMO support the use of Agile?
Team composition	Have you adjusted to include an Agile advocate who has authority? Have the roles and responsibilities been updated to reflect the use of Agile?
Culture	Are you prepared to encourage, institute, promote, and sponsor the culture change and the associated issues? Is someone knowledgeable in organizational change management available to work with your team?
Staffing	Is the PMO aware of the different team compositions needed to support an Agile project? What type of Agile-knowledgeable PMO staff is needed for your program? Does the government PM have Agile experience? Are more Agile software-savvy personnel available to support the program? Consider how you will include integration and test personnel—in particular those involved in system acceptance and operational test.

Area of Concern	Consideration
Acquisition, regulations, policies, etc. used within phases	Review all application regulations and policies to determine any specific impacts to your program. Tailor items or obtain waivers as needed. In particular, pay attention to cost, affordability, and cost realism. Also look for potential constraints from independent operations test agencies, information assurance, information superiority and interoperability. Consider taking a hybrid approach or having an IA expert as part of Agile team.
Integration and test	Determine your approach. Encapsulate the “Agile work” so that sell off and final integration and test are traditional. Conduct early and frequent involvement of testing personnel (earlier than for traditional methods). If the team does continuous integration, determine how it may affect your test program. Determine if concurrent testing would conflict with the sometimes-mandated separation between test personnel and mission development personnel. Determine if access to the target environment will be given to the software integrators.
Infrastructure	Determine the shared assets (if any) for the program. Shared assets could be models, common facilities, agreement on measures, etc. used by multiple teams supporting the program. Sometimes shared assets become constraints for Agile teams.
Deliverables	Determine deliverable details for each phase using Agile. Ensure adequate documentation for use in operations and support (sustainment) is provided.
Contracting	Determine any contract changes required to support Agile.
Concept of operations (CONOP)	This still applies for Agile. Determine the influence and context the CONOP provides to the Agile stories. Ensure the operational architecture is provided up front.
Cost estimation	Determine how you will evaluate the costs based on Agile since cost and accounting techniques may be different. Determine what “full funding” means in an Agile sense.
Define/build/test component team	Do you have appropriate staffing to represent the end user? This could be a “product owner” who serves as the proxy for the users.
Two-level planning	Determine the contents of each release and iteration. These are typically done by the contractor (developer) with collaboration from the government, which will need to negotiate the government priorities. This is similar to the planning practices already in place with other methods (rolling wave).
Reflection and adaptation (retrospective)	Be prepared to participate in this activity. It is driven from the individual team not top-level management. Thus, some organization change management and a new business model will need to be employed.

Appendix D: Acronyms

Table 3: Acronyms Used in This Report

ACTDs	Advanced Concept Technology Demonstrations
ADM	Acquisition Decision Memorandum
AIS	Automated Information System
AoA	Analysis of Alternatives
APB	Acquisition Program Baseline
ASD	Adaptive Software Development
CAIG	Cost Analysis Improvement Group
CDD	Capability Development Document
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CMMI	Capability Maturity Model Integration
CONOP	Concept of Operations
COTS	Commercial-off-the-Shelf
CSDR	Cost and Software Data Reporting
CTE	Critical Technology Element
DAU	Defense Acquisition University
DoD	Department of Defense
DoDAF	DoD Architecture Framework
DoDD	Department of Defense Directive
DoDI	Department of Defense Instruction
DSDM	Dynamic Systems Development Method
EMD	Engineering and Manufacturing Development
EVMS	Earned Value Management System
FAR	Federal Acquisition Regulation
FDD	Feature Driven Development
FIST	Fast, Inexpensive, Simple, Tiny
HSI	Human Systems Integration
IDR	Interim Design Review
IEEE	Institute of Electrical and Electronics Engineers
IMS	Integrated Master Schedule
JMPS	Joint Mission Planning System
KPP	Key Performance Parameter
MDA	Milestone Decision Authority
MSA	Materiel Solution and Analysis
ORS	Operationally Responsive Space

OS	Operations and Support
OSD	Office of the Secretary of Defense
OTA	Operational Test Agency
PDR	Preliminary Design Review
PMO	Program Management Office
RFP	Request for Proposal
RUP	Rational Unified Process
SAF	Secretary of the Air Force
SEI	Software Engineering Institute
SME	Subject Matter Expert
SOW	Statement of Work
SRR	System Requirements Review
TD	Technology Development
TDS	Technology Development Strategy
TRR	Test Readiness Review
TSP	Team Software Process
VMOC	Virtual Mission Operations Center
WBS	Work Breakdown Structure
XP	eXtreme Programming

Appendix E: FIST Manifesto

–contributed by Dan Ward

THE FIST MANIFESTO (Fast, Inexpensive, Simple, Tiny)

System development projects should be done by the smallest possible team of talented people, using a short schedule, a small budget and mature technologies to deliver innovative solutions to urgent needs. This approach is called FIST: *Fast, Inexpensive, Simple, Tiny*.

Short timelines increase agility and stabilize requirements, technology, budgets and people. Short timelines also force accountability, ownership and learning. To maintain short timelines, a project must also exercise restraint over budgets, complexity, and size. Increases to the project's budget, complexity or size inevitably reduce its speed.

Accordingly, the FIST approach advocates the following:

- Minimize team size, maximize team talent.
- Use schedules and budgets to constrain the design.
- Insist on simplicity in organizations, processes and technologies.
- Incentivize and reward under-runs.
- Requirements must be achievable within short time horizons.
- Designs must only include mature technologies.
- Documents and meetings must be short. Have as many as necessary, as few as possible.
- Delivering useful capabilities is the only measure of success.

FIST Principles

- A project leader's influence is inversely proportional to the project's budget and schedule.
- Creative constraints foster creativity. Adding time and/or money generally does not improve outcomes.
- Fixed funding and floating requirements are better than fixed requirements and floating funding.
- Complexity is a cost.
- Complexity reduces reliability.
- Simplicity scales. Complexity doesn't.
- An optimal failure costs a little and teaches a lot. When FIST projects fail, they fail optimally.
- Iteration drives learning, discovery and efficiency. FIST is iterative.
- Talent trumps process.
- Teamwork trumps paperwork.
- Leadership trumps management.
- Trust trumps oversight.

—*Lt Col Dan Ward, USAF, Maj Gabe Mounce, USAF, J. Simmons, Founder Mach 30 Inc., Deji Badiru, PhD, Air Force Institute of Technology, Rolf C. Smith III, USAF, Lt Col Phil Garrant, USAF, Maj Rhet Turnbull, USAF, Richard A. (Dick) Field, Jr., OASD(HA)/TMA, Cynthia J. Wood, U.S. Corps of Engineers, Christopher R. Paparone, PhD, U.S. Army Command and General Staff College, Chris Gunderson, Research Associate Professor of Information Science, Naval Postgraduate School PI W2COG and NetCert projects, Andy Nulman, President and CMO of Airborne Mobile Inc., Rolf Smith II, John Palmer, PhD, Rick Brennan, Capt Pete Mastro, USAF*[44]

[44] FIST Manifesto signatories as of March 16, 2010.

Appendix F: A Reference Guide To Agilities, Flexibilities & Simplifications within the Federal Acquisition Regulation and DoD 5000.02, Operation of the Defense Acquisition System

contributed by Dan Ward

INTRODUCTION

People who can quote chapter and verse from the Federal Acquisition Regulation (FAR) are not only more convincing than those who cannot, but their proposed courses of action are more likely to be correct and allowable under current rules. However, the length and density of the federal regulation is often a barrier to familiarity for all but the most dedicated specialist. Further, the accumulated inertia of earlier, narrow interpretations discourages the adoption of new approaches.

This appendix aims to help reduce those barriers and make acquisition professionals aware of ways the government's official policy can help make acquisition programs faster, simpler, and less expensive, while also ensuring the technology satisfies the necessary quality and performance requirements. There is plenty of room for improvement within the FAR, but there is also more room for innovation than the casual observer might suspect.

The following pages present selected excerpts (emphasis added) from the FAR and DoD Instruction 5000.02, *Operation of the Defense Acquisition System*. Accompanying each excerpt is a brief commentary on potential ways to interpret and implement these regulations, as well as summaries of the underlying principles.

This appendix is not a comprehensive overview of the entire body of federal acquisition policy and regulation. Instead, it highlights specific portions from two key regulations which describe the simplifications, agilities, flexibilities and alternatives currently available to acquisition professionals.

This appendix is not an official opinion and does not constitute legal or contractual advice. Instead, this informal analysis aims to serve as an easy starting point for further discussion. The goal is to equip program managers, engineers, and other acquisition practitioners from government and industry alike with a quick reference guide to some of the more useful and empowering portions of federal acquisition policy.

THE FEDERAL ACQUISITION REGULATION

FAR 1.102-2(b)(2) Federal Acquisition Regulation System – Performance Standards

> (2) The System must provide uniformity where it contributes to efficiency or where fairness or predictability is essential. **The System should also, however, encourage innovation, and local adaptation where uniformity is not essential.**

In this introductory statement, the FAR makes it clear that the System is supposed to encourage innovation and variation. It acknowledges that uniformity in how we interpret, implement, or execute the FAR's guidance is occasionally necessary in the name of efficiency, fairness, or predictability, but **the intent is for the FAR to adapt to local needs, not the other way around**.

This is an important point to understand, as it sets the foundation for how the FAR should be interpreted and applied. The FAR is supposed to foster efficiency, fairness, and predictability, as well as innovation and adaptability to different contexts. These are not mutually exclusive objectives. In fact, **adapting to local needs actually makes the FAR fairer and more efficient**. In contrast, insisting that each federal agency behave exactly like every other agency tends to drive inefficiency into the process. And that is not what the FAR says to do.

FAR 13.003 Policy.

> (a) Agencies shall **use simplified acquisition procedures to the maximum extent practicable** for all purchases of supplies or services not exceeding the simplified acquisition threshold

Despite its reputation for complexity—or perhaps because of it—the FAR actually contains an explicit preference for using simplified procedures. **These simpler approaches do not merely exist. They are actually the preferred default** and should be used "to the maximum extent practicable."

The underlying principle—that **simpler procedures are both available and effective**—is an important one to keep in mind. Extremely high levels of complexity are neither inevitable nor desirable, and are certainly not required by the FAR.

Acquisition programs are full of opportunities and decision points where people get to choose between simple and complex alternatives. These alternatives are often stark and obvious, but while the FAR has an express preference for simplicity, many people feel compelled to adopt the more complicated approach due to organizational inertia or "the way we always do it around here." This is not necessary, and **acquisition professionals who opt for a simpler approach should know they do so with the full support of the FAR**.

FAR 15.306(d)(4)

> This section encourages the government to "suggest to offerors that have exceeded any mandatory minimums (in ways that are not integral to the design), that **their proposals would be more competitive if the excesses were removed** and the offered price decreased."

In yet another statement favoring simplicity, this FAR paragraph addresses the pre-award/source selection phase of activity. When evaluating an over-engineered proposal or one that exceeds

minimums unnecessarily, **the government's source selection team is encouraged to provide direct feedback to the offeror** and to suggest that their proposal would be more competitive if it was scaled down.

Source selection teams should neither scoff at nor pursue an expensive, over-engineered proposal that aims to deliver a Gucci solution to an every-day requirement. Instead, the FAR makes it clear that **it is in the government's best interest to provide feedback to such offerors and to invite them to simplify their proposals**.

The underlying principle is that an **over-reaching proposal drives up cost and reduces the quality** of the award competition. This is important for government and industry personnel alike to understand. The other principle is that **the government is free to tell offerors how to make their proposals more competitive, even in the early phase of the source selection.** In fact, the FAR encourages them to do so.

This communication between evaluators and bidders needs to be accomplished in accordance with the overall guidelines for source selection communications and should not be used to give anyone an unfair advantage (see the remainder of FAR 15.306), but the objective of these discussions, as explained in 15.306(d)(2) "… is to maximize the Government's ability to obtain best value …" To obtain best value, the government can invite offerors to remove excesses from within their proposals. This allows each offeror to put their best foot forward and fosters genuine competition between the best of breed from all contenders.

FAR 18.101 Emergency Acquisitions, General.

The FAR includes many acquisition flexibilities that are available to the contracting officer when certain conditions are met. These **acquisition flexibilities do not require an emergency declaration** or designation of contingency operation.

In addition to having a preference for simplicity, **the FAR is also strongly in favor of flexibility**. The previous sentence may come as a surprise to people who have only seen the FAR subjected to strict interpretations, but the regulation makes it quite clear that **flexibilities are indeed present, available, and preferred**.

The FAR does not intend these flexibilities to be seldom-used contingencies or reluctantly authorized departures from the norm. Instead, it explicitly encourages their use as a matter of course. It is worth noting that while many of these flexibilities are described in Part 18 (*Emergency Acquisitions*), their use is *not* limited to formally declared emergencies or other special occasions. In fact, they are available to any contracting officer when "certain conditions" are met. The remainder of Sub-Part 18.1 identifies a number of specific flexibilities relevant to various situations and explains those conditions.

FAR 35.002 Research & Development Contracting, General.

The contracting process shall be used to encourage the best sources from the scientific and industrial community to become involved in the program and must provide an environment in which the work can be pursued with **reasonable flexibility and minimum administrative burden**.

Here we again see the FAR expressing a preference for flexibility, but now we see the added emphasis on reducing the administrative burden. It may be fair to say the FAR is the very definition of "administrative burden;" we should also recognize that **the FAR goes to considerable length to minimize and reduce the administrative burden on acquisition programs**.

While this particular quote is from the FAR's R&D subpart, variations on that phrase are found in several other places throughout the regulation (for example, Subpart 16.202-1—Fixed-Price Contracts, or 4.1200 – Representations and Certifications). The general principle of reducing the administrative burden can be applied quite broadly.

Therefore **any enterprising acquisition professionals who seek to reduce the administrative burden for their particular project will find the FAR is on their side**. Attempts to maintain or increase the burden, on the other hand, are actually contrary to the FAR's direction.

FAR 35.008 Evaluation for award.

> (a) Generally, an R&D contract should be awarded to that organization, including any educational institution, that proposes the best ideas or concepts and has the highest competence in the specific field of science or technology involved. However, **an award should not be made to obtain capabilities that exceed those needed for successful performance of the work**.

This section echoes FAR 15.306(d)(4) in its preference for a restrained approach to contract award. That is, an offeror whose proposed solution exceeds the government's need should not be viewed more favorably than an offeror whose proposed solution merely meets the need. Again, this is an important principle for government and industry alike to understand.

The basic principle is that **the government should avoid paying a premium for capabilities it does not really require.** Source selection decisions should not chase after the latest shiny object, which tends to cost more and take longer, but instead should take a more restrained approach.

This subpart brings to mind the proverbial 70% solution, widely held up as preferable to the 100% alternative. The reason 70% beats 100% is that the more modest solution tends to cost considerably less, tends to be available much sooner, and tends to be a better fit with actual needs. The so-called 100% solution, in contrast, tends to be overkill, late-to-need, and overpriced.

The Department of Defense is quite specific on this topic. A memo signed on January 23, 2013 by ADM James Winnefeld, the vice chairman of the Joint Chiefs of Staff and chairman of the Joint Requirements Oversight Council, encourages program managers to request requirement relief whenever requirements (specifically Key Performance Parameters, or KPPs) "appear out of line with an appropriate cost-benefit analysis." The memo, titled *Key Performance Parameter Relief*, states

> *KPP relief should be considered especially appropriate in cases where significant cost savings may be achieved with marginal impact to operational capability (i.e., spending 15 percent of a program's budget to get the last 3 percent of KPP performance)…*

This is essentially identical to the principle in FAR 35.008. If the last 3 percent of capability is not really necessary and is consuming a disproportionate amount of the resources, then a more modest approach is clearly called for.

FAR 39.103 Modular contracting.

Modular contracting is intended to reduce program risk and to incentivize contractor performance while meeting the government's need for timely access to rapidly changing technology. Consistent with the agency's information technology architecture, **agencies should, to the maximum extent practicable, use modular contracting to acquire major systems (see 2.101) of information technology**. Agencies may also use modular contracting to acquire non-major systems of information technology.

The concept of modular contracting involves dividing large efforts into a series of smaller efforts. This can be done more often than it is done, and the FAR establishes an explicit preference for modular contracting.

While major information technology systems often appear monolithic and indivisible, a closer inspection often reveals hidden seams and opportunities to chunk, divide, and sub-divide the effort. **The FAR could not be clearer in its preference for reducing large IT systems into a series of smaller systems**.

The specific application of modular contracting in the FAR is for information technology, but the practice can often be applied on non-IT systems as well. **Not every program can be developed using this method, but modular contracting is clearly the default, preferred approach.** Organizations that want to use a non-modular approach to deliver a big, expensive, complex system should have to justify their preference and gain special authorization. Those that want to break a large effort into a modular series of smaller projects can look to this section of the FAR for support and should know they are doing precisely what they should be doing.

This has significant implications for the proponents of Agile methodologies in particular, because many of the benefits described in the box below speak directly to Agile practices.

(b) When using modular contracting, an acquisition of a system of information technology may be **divided into several smaller acquisition increments** that—

(1) Are **easier to manage individually** than would be possible in one comprehensive acquisition;

(2) Address complex information technology objectives incrementally in order to **enhance the likelihood of achieving workable systems** or solutions for attainment of those objectives;

(3) Provide for delivery, implementation, and testing of workable systems or solutions in discrete increments, each of which comprises **a system or solution that is not dependent on any subsequent increment in order to perform its principal functions**;

(4) Provide an opportunity for subsequent increments **to take advantage of any evolution in technology or needs** that occur during implementation and use of the earlier increments; and

(5) Reduce risk of potential adverse consequences on the overall project by **isolating and avoiding custom-designed components** of the system.

Having established the benefits of modularizing large IT systems, the FAR goes on to explain how to define each module and to illuminate what these increments should look like:

> (c) The characteristics of an increment may vary depending upon the type of information technology being acquired and the nature of the system being developed. The following factors may be considered:
>
> (1) To promote compatibility, the information technology acquired through modular contracting for each increment should **comply with common or commercially acceptable information technology standards** when available and appropriate, and shall conform to the agency's master information technology architecture.
>
> (2) The **performance requirements of each increment should be consistent with the performance requirements of the completed, overall system** within which the information technology will function and should address interface requirements with succeeding increments.

The key to successful modular design, in IT or other categories, is to have a well-defined architecture, complete with standard interfaces. This helps prevent optimization of a part at the expense of the whole, and ensures that new modules are compatible with the existing modules. And it is precisely what the FAR proposes.

The formal systems engineering principle of "high cohesion, low coupling" applies here as well. This principle ensures that changes to one module do not cause complexity-adding ripples throughout the rest of the system, and helps to reduce the cost, delay, and complexity of upgrading or replacing older modules.

Finally, this section addresses the importance of ensuring the performance requirements of each part are consistent with the performance of the whole. Mismatched performance can produce a fragile architecture rather than a robust one, as one segment produces more information than another can accommodate, or one piece operates at a slower rhythm than the rest.

> (d) For each increment, contracting officers shall choose an appropriate contracting technique that facilitates the acquisition of subsequent increments. Pursuant to Parts 16 and 17 of the Federal Acquisition Regulation, contracting officers shall select the contract type and method appropriate to the circumstances (e.g., indefinite delivery, indefinite quantity contracts, single contract with options, successive contracts, multiple awards, task order contracts). Contract(s) shall be structured to ensure that **the Government is not required to procure additional increments**.

This section is **yet another FAR paean to flexibility**. It points out that contracting officers have a number of contract types and methods to choose from. It lists them. It then encourages adopting and structuring the contract in such a way as to ensure the government maintains the flexibility to stop procurement between each increment. That is, the **contract structure should provide the opportunity—but not the obligation—to proceed from one module to the next.**

Maintaining this opportunity is a simple matter of including the appropriate contract clause. The idea of modularity means the overall effort is severable. While the initial plan may envision a series of 10 increments, we may discover that the requirement is satisfied after only 6 or 7. In such cases, we can declare success and call it a day. This is just one of the many benefits of the modular approach.

(e) To avoid obsolescence, a modular contract for information technology should, to the maximum extent practicable, **be awarded within 180 days** after the date on which the solicitation is issued. **If award cannot be made within 180 days, agencies should consider cancellation** of the solicitation in accordance with 14.209 or 15.206(e). To the maximum extent practicable, **deliveries under the contract should be scheduled to occur within 18 months after issuance of the solicitation**.

Information technology is changing at a terrific pace, and it shows no signs of slowing down. Accordingly, **the FAR encourages setting firm deadlines for both contract award and solution delivery** to reduce the likelihood of pointlessly delivering yesterday's technology tomorrow. In fact, the FAR suggests that **delays justify cancellation**. If we cannot do it in 180 days, we should not do it.

DoD Instruction 5000.02

January 7, 2015

While the Pentagon is subject to the FAR, it also has its own set of instructions on how to run an acquisition program: DoD Instruction 5000.02. This instruction is intended to complement the FAR and to provide military-specific guidance for the unique demands and challenges associated with defense-related technologies and services.

Like the FAR, 5000.02 can be interpreted and implemented in a variety of ways. Also like the FAR, the text itself provides instructions on how it is supposed to be interpreted: with an eye to tailoring, flexibility, innovation, and speed. The following pages provide a small number of excerpts intended to point defense acquisition personnel towards some of the opportunities and flexibilities inherent in the Pentagon's acquisition system.

5.a.(2) Procedures - Overview

The structure of a DoD acquisition program and the procedures used should be **tailored as much as possible** to the characteristics of the product being acquired, and to the totality of circumstances associated with the program including operational urgency and risk factors. (page 2)

There are two approaches to tailoring a policy or procedure. One is to maintain uniformity as a general rule and only tailor the procedure when necessary. Under this approach, the default answer to a tailoring request is *No*, and the burden of proof is on the person asking for the waiver or alternative path. The requestor must provide a strong and compelling justification for why they should be allowed to deviate from the standard procedure. This is not consistent with DoD 5000.02.

The other approach is to tailor as much as possible, which is what 5000.02 says to do. Accordingly, tailoring should be the standard procedure and **the default answer to a tailoring request should be *Yes***. Under a "tailor as much as possible" framework, there must be a strong and compelling reason to enforce uniformity. The burden of proof is on the person who seeks to deny the request rather than the one making the request.

(b) When there is a strong threat-based or operationally driven need to field a capability solution in the shortest time, **MDAs are authorized to implement streamlined procedures designed to accelerate acquisition system responsiveness**. (page 2)

When speed is needed, Milestone Decision Authorities are explicitly allowed to streamline and accelerate the process. This is an important authority for program offices to bear in mind, because they often have direct awareness of the operational need and thus are in a position to make the case for shortening the delivery timeline.

A program office must ensure they have a solid understanding of the user's time-to-need, and should be diligent to convey that information up the chain to the MDA. If the timeline is short, this information should be accompanied with a specific request to streamline the procedure.

4 – Program Decision Reviews and Milestones

(e) **Issues should be resolved at the lowest level possible**. When an issue cannot be resolved quickly at a lower level, the issue will be submitted to the MDA with complete and objective data necessary to support a decision (page 3)

In an echo of Peter Drucker's recommendation that "decisions should be made at the lowest possible level and as close to the action as possible," 5000.02 establishes a clear preference for resolving issues at the lowest level possible. This is important for people at all levels of the organization to understand.

(f) The documents prepared in support of the decision process (e.g., Acquisition Strategy, Systems Engineering Plan (SEP), Test and Evaluation Master Plan (TEMP), Life-Cycle Sustainment Plan (LCSP)) **should generally not be prepared solely for staff review and approval, but be intended primarily for use within the program as planning and management tools** that are highly specific to the program and tailored to meet program needs. (page 4)

This paragraph on program decision reviews shines a light on some of the tension inherent in many acquisition programs. Program offices are required to produce a large stack of documentation at various phases of the decision process, but 5000.02 explains that **the primary audience for these documents is the program office itself**, not the various staffs and functions who review the documents.

That is, while senior executives and functional experts review and approve the various planning documents, securing approval is not the sole—or even the primary—reason for producing them.

Defense Acquisition Program Models

This model is distinguished from the previous model by the **rapid delivery of capability** through multiple acquisition increments, each of which provides part of the overall required program capability. Each increment may have several limited deployments; each deployment will result from a specific build and provide the user with a mature and tested sub-element of the overall incremental capability. **Several builds and deployments will typically be necessary to satisfy approved requirements for an increment of capability**. The identification and development of technical solutions necessary for follow-on capability increments have some degree of concurrency, allowing subsequent increments to be initiated and executed more rapidly. (page 11)

The instruction provides several tailorable models for acquisition programs, and the third is named *Incrementally Deployed Software Intensive Program*. While this is distinct from the FAR's "modular contracting," it shares many of the same practices and benefits.

Specifically, it involves breaking a large effort into an iterative series of smaller efforts, each of which provides a fraction of the overall capability. It focuses on rapidly delivering valuable, working software on a short timeline, then using the lessons from the earlier phases to shape activities in subsequent phases.

The goal is to use speed to foster learning. The point is that the Instruction provides a roadmap for doing exactly that.

Enclosure 8: AFFORDABILITY ANALYSIS AND INVESTMENT CONSTRAINTS

3 – Lifecycle Affordability Analysis, 3.e (3) (page 124)

> The **metrics used for MDA-approved affordability constraints on procurement and sustainment costs may be tailored** to the type of acquisition and the specific circumstances of a given program. In addition to capability requirements tradeoffs approved by the requirements validation authority; prudent investments in research, development, and test and evaluation; **innovative acquisition strategies; and incentives to reduce costs can be used** to ensure that affordability constraints are achieved

Once again, we find an express preference for flexibility and tailoring, this time in the areas of measures, strategies, and incentives. That is, **5000.02 does not insist on every project conforming to a uniform approach**. Instead, it grants authority for different programs to use different measures, to adopt innovative acquisition strategies, and to provide a variety of incentives to reduce costs.

CONCLUSION

Regulations do not interpret themselves. Rather, it is up to human beings to read, understand, and apply the regulations. Both the FAR and DoDI 5000.02 contain guidance about how they are intended to be applied—with a preference for adaptability instead of uniformity and a preference for minimizing the administrative burden rather than expanding it.

These policies contain many explicit statements in favor of flexibility, agility, and innovation, many of which are described in this document. Widespread familiarity with such statements can help increase the efficiency and effectiveness of government acquisition programs.

It is hoped that this brief primer helps equip acquisition professionals with the policy support necessary to reduce the cost, time, and complexity of their projects.

Bibliography

URLs are valid as of the publication date of this document.

[Alleman et al. 2003]
Alleman, G. B. et al. Making Agile Development Work in a Government Contracting Environment: Measuring Velocity with Earned Value. *Agile Development*. June 25-28, 2003. http://www.informatik.uni-trier.de/~ley/db/conf/agiledc/agiledc2003.html

[Beck et al. 2001]
Beck, Kent et. al. A*gile Software Development Manifesto*. 2001. http://www.agilemanifesto.org/

[Beck & Boehm 2003]
Beck, K. & Boehm, B. Agility Through Discipline: A Debate. *IEEE Computer.* Volume 36. Number 6. June 2003. http://www.computer.org/portal/web/csdl/doi/10.1109/MC.2003.1204374

[Bingue & Cook 2009]
Bingue, E. W. & Cook, D. A. *The Art of Applying Commercial Best Practices in the DoD*. 21st Annual Systems and Software Technology Conference 2009: Technology: Advancing Precision. 2009.

[Boehm 2002]
Boehm, B. Get Ready for Agile Methods, With Care. *IEEE Computer.* January 2002.

[Boehm & Turner 2004]
Boehm, B & Turner, R. *Balancing Agility and Discipline – A Guide for the Perplexed*. Addison-Wesley. 2004.

[Boehm & Turner 2005]
Boehm, B. & Turner, R. Management Challenges to Implementing Agile Processes in Traditional Development Organizations. *IEEE Software*. Volume 22. Number 5. September 2005. http://portal.acm.org/citation.cfm?id=1092725

[Buresh 2008]
Buresh, D. *Customer Satisfaction and Agile Methods.* VDM Verlag Dr. Muller Actiengesellschaft & Co. KG. 2008.

[CAI 2009]
Computer Aid Inc. *An Agile Developer's Guide to Lean Software Development*. 2009.

[Cianci 2009]
Cianci, K. A. *Agile System Development*. 21st Annual Systems and Software Technology Conference 2009: Technology: Advancing Precision. 2009.

[Clark & Johnson 2009]
Clark, J. O. & Johnson, S. *Agile Systems Engineering and Software Engineering*. 21st Annual Systems and Software Technology Conference 2009: Technology: Advancing Precision. 2009.

[Coats & Koehnemann 2009]
Coats, M. R. & Koehnemann, H. *Experiences Applying Agile Practices to Large Systems Development*. 21st Annual Systems and Software Technology Conference (STSC) 2009: Technology: Advancing Precision. 2009.

[Cockburn 2002]
Cockburn, A. *Agile Software Development*. Addison-Wesley. 2002. http://alistair.cockburn.us/Agile+software+development:+the+people+factor

[Cockburn 2007]
Cockburn, A. What Engineering Has in Common With Manufacturing and Why It Matters. *Crosstalk*. April 2007. http://www.crosstalkonline.org/storage/issue-archives/2007/200704/200704-0-Issue.pdf

[Daconta 2009]
Daconta, M. 6 Tech Trends Government IT Managers Should be Wary of. *Federal Computer Week.* August 2009. http://fcw.com/articles/2009/08/10/reality-check-it-fads-not-fit-for-government.aspx

[DeMarco & Boehm 2002]
DeMarco, T. & Boehm, B. The Agile Methods Fray. *IEEE Computer.* Volume 35. Number 6. June 2002. http://portal.acm.org/citation.cfm?id=622005

[Derby 2007]
Derby, E. Collaboration Skills for Agile Teams. *Crosstalk.* April 2007. http://www.crosstalkonline.org/storage/issue-archives/2007/200704/200704-0-Issue.pdf

[Dobbins 2009]
Dobbins, J. H. *Agile Acquisition Within the Current Policy Framework*. 21st Annual Systems and Software Technology Conference 2009: Technology: Advancing Precision. 2009.

[DoD 2007]
Department of Defense. *Department of Defense Directive (DoODD) 5000.01*. November 2007. https://acc.dau.mil/CommunityBrowser.aspx?id=37343

[DoD 2008]
Department of Defense. *Department of Defense Instruction (DoODI) 5000.02*. December 2008. https://acc.dau.mil/CommunityBrowser.aspx?id=37343

[DoD 2015]
Department of Defense. DoD 5000 Series. *Acquisition Community Connection Website*. January 2015. https://acc.dau.mil/CommunityBrowser.aspx?id=18532&lang=en-US

[Dwyer 2009]
Dwyer, M. *Updating "Software Engineering" for the 21st Century*. 21st Annual Systems and Software Technology Conference (STSC) 2009: Technology: Advancing Precision. 2009.

[Dybå & Dingsøyr 2009]
Dybå T. & Dingsøyr, T. What Do We Know About Agile Software Development. *IEEE Software*. September/October, 2009.

[Gamma et al. 1994]
Gamma, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley. 1994. http://www.informit.com/store/product.aspx?isbn=0201634988

[Glazer et al. 2008]
Glazer et al. *CMMI or Agile: Why Not Embrace Both!* CMU/SEI-2008-TN-003. Software Engineering Institute, Carnegie Mellon University. 2008. http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=8533

[Highsmith 2004]
Highsmith, J. *Agile Project Management: Creating Innovative Products*. Addison Wesley. http://www.informit.com/store/product.aspx?isbn=0321219775

[Highsmith 2009a]
Highsmith, J. *Advanced Agile Project Management Seminar*. Agile Development Practices Conference. 2009.

[Highsmith 2009b]
Highsmith, J. *Beyond Scope, Schedule, and Cost: Measuring Agile Performance*. Agile Development Practices Conference. 2009.

[Highsmith 2009c]
Highsmith, J. Beyond Scope, Schedule, and Cost: Measuring Agile Performance [blog post]. *Cutter Blog.* September 2009. http://blog.cutter.com/2009/08/10/beyond-scope-schedule-and-cost-measuring-agile-performance/

[Hunt & Thomas 1999]
Hunt, A. & Thomas, D. *The Pragmatic Programmer*. Addison-Wesley Professional. 1999.

[JUnit 2015]
JUnit. JUnit.org Resources for Test Driven Development. *JUnit.org Website*. 2015. http://www.junit.org

[Lapham et al. 2011]
Lapham, M. A. et al. *Agile Methods: Selected DoD Management and Acquisition Concerns.* CMU/SEI-2011-TN-002. Software Engineering Institute, Carnegie Mellon University, 2011. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=9769

[Lapham et al. 2014]
Lapham, M. A. et al. *Agile Methods and Request for Change (RFC): Observations from DoD Acquisition Programs.* CMU/SEI-2013-TN-031. Software Engineering Institute, Carnegie Mellon University. 2014. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=77732

[Lascano 2009]
Lascano, J. E. *eXtreme Programming (XP) May Be Embedded Inside Scrum*. 21st Annual Systems and Software Technology Conference (STSC) 2009: Technology: Advancing Precision. 2009.

[Leffingwell 2008]
Leffingwell, D. *Scaling Software Agility: Best Practices for Large Enterprises*. Addison-Wesley. 2008. http://www.informit.com/store/product.aspx?isbn=0321458192

[Miller 2005]
Miller, G. Agile Software Development For The Entire Project. *Crosstalk*. December 2005. http://www.crosstalkonline.org/storage/issue-archives/2005/200512/200512-Miller.pdf

[O'Brien 2009]
O'Brien, W. P. *Agile Integration of Complex Systems*. 21st Annual Systems and Software Technology Conference 2009: Technology: Advancing Precision. 2009.

[Patton 2009]
Patton, J. *Using Kanban Techniques to Control Incremental Development*. 21st Annual Systems and Software Technology Conference (STSC) 2009: Technology: Advancing Precision. 2009.

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

[Rally 2009]
Rally Software Development Inc. *Iteration Planning Guide*. 2009. https://www.rallydev.com/content-request?nid=729

[Rico et al. 2009]
Rico et al. *What Is the ROI of Agile vs. Traditional Methods?* J. Ross Publishing. 2009. http://www.jrosspub.com

[Rico 2009]
Rico, D. F. Business Value of Agile Methods: Using ROI & Real Options. *PMI Baltimore Chapter Website*. 2009. http://www.pmibaltimore.org/events/event_details.php?id=452

[Scumniotales et al. 2009]
Scumniotales, J. et al. Why Scrum Isn't Enough for Agile Success. *Agile Journal Website*. July 2009. http://www.agilejournal.com/news-a-events/events/details/15-webcast-why-scrum-isnt-enough-for-agile-success

[Serena 2009]
Serena Software. *Comparing Extreme Programming, Scrum, and Lean Software Development in Agile*. October 2009.

[Shalloway 2009]
Shalloway, A. *Kanban: A True Integration of Lean and Agile*. Agile Development Practices Conference. 2009. http://www.sqe.com/ConferenceArchive/AgileDevPractices2009/ConcurrentThursday.html

[Sliger 2009]
Sliger Consulting & Computer Aid Inc. *Selling Agile*. 2009.

[Solomon 2009]
Solomon, P. J. *Agile Methods with Performance-Based Earned Value*. 21st Annual Systems and Software Technology Conference 2009: Technology: Advancing Precision. 2009.

[Troup 2010]
Troup, T. Acquisition Truths From the Trenches. *Defense AT&L.* May/June 2010. http://www.dau.mil/pubscats/ATL%20Docs/May-Jun10/troup_may-june10.pdf

[Turner 2009]
Turner, R. *Evaluating the Effectiveness of Systems and Software Engineering Methods, Processes and Tools for Use in Defense Programs*. 21st Annual Systems and Software Technology Conference 2009: Technology: Advancing Precision. 2009.

[Turner 2007]
Turner, R. Toward Agile Systems Engineering Processes. *Crosstalk.* April 2007. http://www.crosstalkonline.org/storage/issue-archives/2007/200704/200704-0-Issue.pdf

[Ward & Quaid 2006]
Ward, D. & Quaid, C. FIST, Part 5, Putting the Pieces Together. *Defense AT&L.* May 2006.

[Ward et al. 2008]
Ward et al. *The FIST Handbook*. Rogue Press Book. 2008.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	December 2016	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Update 2016: Considerations for Using Agile in DoD Acquisition	FA8721-05-C-0003

6. AUTHOR(S)

Dan Ward and Suzanne Miller

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2016-TN-001

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFLCMC/PZE/Hanscom Enterprise Acquisition Division 20 Schilling Circle Building 1305 Hanscom AFB, MA 01731-2116	n/a

11. SUPPLEMENTARY NOTES

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

13. ABSTRACT (MAXIMUM 200 WORDS)

This report, an update to a 2010 report, *Considerations For Using Agile In DoD Acquisitions*, addresses developments in commercial Agile practices as well as the Department of Defense (DoD) acquisition environment. It covers some previously unanswered questions and asks some new ones. It includes new research, examples of Agile in practice, and policy updates.

Continuing with the 2010 report's theme, this report updates the exploration of the questions: Can Agile be used in the DoD environment? If so, how? It includes lessons learned from DoD programs that have employed Agile and information gleaned from myriad articles and books on Agile. While this report does not pretend to cover every paper or thought published about Agile in the DoD world, it provides an updated overview of some challenges in using Agile, an overview of how some programs have addressed these challenges, and some additional recommendations on dealing with these challenges. The intended audience is policy makers, program office staff, and software development contractors who are contemplating proposing the use of Agile methods.

We hope this report stimulates new discussion about adopting Agile in the DoD world and equips practitioners with the information they need to make informed decisions.

14. SUBJECT TERMS	15. NUMBER OF PAGES
acquisition, Agile methods, lessons learned, software development, DoD, Department of Defense	90

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