Software Engineering Institute

Potential Use of Agile Methods in Selected DoD Acquisitions: Requirements Development and Management

Dr. Kenneth E. Nidiffer
Suzanne M. Miller
Dr. David Carney

April 2014

TECHNICAL NOTE
CMU/SEI-2013-TN-006

Client Technical Solutions

http://www.sei.cmu.edu

Carnegie Mellon

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-0000785

Table of Contents

List of Figures

List of Tables

Acknowledgments

The authors would like to express our appreciation to all those who took time out of their busy schedules to participate in one or more of the following: (1) structured interviews, (2) technical reviews, (3) collaborative discussions and (4) information exchanges via providing articles, graphics, blogs and encouragement. Their thoughtful and precise insights added great value, which we greatly appreciated. We extend our sincerest thanks to the following individuals:

- Dr. Alistair Cockburn, Signer of the Agile Manifesto
- Mary Ann Lapham, SEI
- Ceci Albert, SEI
- COL Pat Flanders, U.S. Army Project Manager, Army Enterprise Systems Integration Program
- Dr. Richard Turner, Stevens Institute of Technology
- Carmen Graver, USMC, Civilian
- Don Firesmith, SEI
- Colleen Regan, SEI
- Dick Carlson
- Dr. Dick Fairley, Software and Systems Engineering Associates
- Mike Janiga, MITRE
- Scott Anderson, Lincoln Labs
- Pete Modigliani, MITRE
- Jesse Fewell, LeadingAgile, LLC

There are always others who helped that we can't publicly acknowledge. We appreciate all your efforts on our behalf.

Executive Summary

Organizations in the U.S. Department of Defense (DoD), as well as other federal agencies, are beginning to make greater use of Agile methods to build and evolve software-reliant systems. Some Agile practices, especially those that relate to software implementation—design, code, and unit test—translate in a relatively straightforward manner into regulated settings like the Department of Defense. Other practices, particularly those related to how requirements are treated in Agile settings, are more challenging to adopt in regulated settings. Adding to the complexity of adopting Agile requirements practices in software implementation settings is the interaction between Agile software teams and systems engineering and other development teams in a complex software-reliant system development that includes custom hardware.

This technical note is one of several SEI technical notes that Carnegie Mellon's Software Engineering Institute has produced that investigate Agile adoption issues in DoD and other regulated settings. It focuses on challenges inherent in using Agile requirements approaches in DoD and federal software-reliant systems, and offers a few potential solutions when Agile practices are in use within a regulated setting. In truth, in our investigation we did not find as many success stories related to using Agile requirements approaches in software-reliant development settings as we have in some of the other topics we have investigated. We address some of the possible reasons for that in the technical note.

In this technical note we differentiate most Agile approaches to requirements from traditional approaches. We call traditional approaches "document-centric" approaches because the early development and management of requirements depends heavily on the creation and review of documents to communicate needs and intent. When contrasting directly with document-centric approaches, call Agile approaches "implementation-driven" approaches because they rely less on documents to communicate requirements and more on conversation that occurs as part of implementing less granular requirement or feature descriptions, often through an Agile concept called stories. This dichotomy of focus is at the crux of the challenge in adapting Agile requirements approaches into programs that are accustomed to traditional systems requirements development, management, and allocation.

In conducting this research, we used accepted qualitative research techniques, particularly a method called grounded theory, to move beyond traditional literature searches and anecdote collection. In grounded theory, we focus on gathering information across the broad boundaries of a topic, then affinitize the information from literature reviews, subject matter expert interviews, and practitioner interviews into the themes that we found in the collected information.

The following six themes were elicited from this study:

- Accepted DoD-level guidance for tracking progress
- Translating potential Agile measures of progress to earned value measures
- A risk-averse acquisition culture
- Work breakdown structure as a tool to manage task assignments
- Effect of requirements change on contracts

- Perception that reduced documentation is a cause for concern

For each of these themes, the document

- describes the issue that the interviews revealed, and why it poses a barrier to use of implementation-centric approaches
- characterizes both the implementation-driven and Agile aspects of the issue
- where possible, provides a sanitized example of how this was manifest in specific programs
- provides suggestions for potential solutions

This set of themes and how they manifest in DoD settings trying to apply Agile requirements approaches is the main focus of the technical note. We also provide a small amount of tutorial information on how typical Agile requirements approaches work in both small and larger settings.

Our most significant finding was that it will require a significant cultural shift for DoD to better align and change its existing Planning, Programming, Budgeting and Execution (PPBE), Joint Capabilities Integration and Development System (JCIDS), and Major Acquisitions System (MAS) document-centric acquisition support systems from a requirements top-down prescriptive delivery focus to a more agile, implementation-driven focus where requirement changes are more welcome across the entire acquisition life cycle [Lapham 2011].

Even though commercial industry has successfully made this transition and documents such as the 2010 and 2011 Defense Authorization Acts use wording that heads in this direction, progress has been slow, partly because this change represents a dramatic change in requirements development and management approaches and an associated cultural shift [DAA 2010, 2011]. Namely, a document-centric orientation focuses the various activities to generate and baseline the full set of requirements in a specification document and associated documentation required throughout the acquisition life cycle, while the implementation-driven orientation focuses on the delivery of required capabilities via executable software. In the former, governance is achieved through measurement of artifact production and activity completion, and in the latter, governance is achieved through measurement of incremental outcomes and progress and quality trends. This governance change represents a significant cultural shift in terms of DoD oversight responsibilities.

An increasing number of commercial organizations have embraced one or more of the Agile method approaches, producing software with less documentation under conditions of rapid change to the high satisfaction of the user. However, DoD is one of the largest and most complex organizations on the planet, which, for some good reasons, has instantiated a risk averse acquisition framework. As a result, DoD acquisition decision makers are perplexed about the risks involved in using Agile methods on large programs even though the potential benefits (as seen at least in unregulated commercial settings) are very attractive. In an effort to support DoD acquisition professionals in the use of Agile methods, this research investigation disclosed a set of requirement development and management acquisition issues that inhibit Agile method usage within DoD. Note that some of these requirements issues also plague acquisition practitioners in document-centric programs. The difference is that, over time, conventions have arisen for dealing with requirements development issues in traditional software-intensive acquisition environments that, though not covered by DoD guidance, are generally accepted in practice.

The acquisition issues we raise in this paper are most productively considered when a program is formulating its acquisition strategy, and we are seeing cases across the military services where projects that want to effectively use Agile methods are being given waivers to some regulatory constraints so that they have the opportunity to demonstrate their ability to deliver needed capabilities in an implementation-driven setting. As mentioned in relation to several of the themes, longer term solutions to the issues that negatively affect DoD adoption of Agile requirements development and management techniques suggest changes will be needed in DoD-level policy and guidance, and those who have embraced a risk averse acquisition culture will surely wait until those changes occur to change their own behavior. Early adopters such as the U.S. Food and Drug Administration and Department of Homeland Security have already published local guidance for use of Agile methods as an alternative to their traditional life cycle approaches. In addition, interaction with individuals from the SEI Agile Collaboration Group and interaction with other government staff who are implementing Agile in their settings make us hopeful that their successful use of Agile methods will speed the guidance transitions that will make Agile methods use easier for the programs to which they are most relevant.

Abstract

Adoption of methodologies that generally come under the umbrella of “Agile” in the software development community includes consideration of how those adopting Agile methods interface with elements of the acquisition community who provide the requirements and other constraints that govern how the software part of a system will be developed. Several Agile methods have expectations about how requirements are handled that are different from the typical approach used in government acquisition settings. This qualitative research study from the Software Engineering Institute explores issues that practitioners in the field who are actively adopting Agile methods have identified in our interviews about their experience in defining and managing requirements.

1 Introduction

1.1 Background and History

In recent years, the field of software engineering has witnessed a broad transformation in management philosophy that has produced significant successes in commercial software engineering practice. The term for this philosophy, embodied by a set of methods and principles, is *Agile*. The successes in using Agile methods have occurred largely in relatively unconstrained environments (i.e., unconstrained by regulation). Recently, Agile methods have started to be used in the DoD acquisition environment, initially in experimental fashion, more recently by larger, more mainstream programs.

Consequently, while there is considerable literature on most aspects of Agile (e.g., methods, tools), there is little written with respect to adopting Agile approaches in large government programs [DAR 2010, Larman 2010, MITRE 2012a], particularly with regard to software requirements, either their development or management. In this technical note, we report on an investigation into this issue, namely in what ways Agile methods relating to requirements development and requirements management, could be used by Department of Defense (DoD) acquisition professionals in selective settings. We argue that these methods could, in fact, prove of great value. However, achieving the possible benefits suggests that considerable change in DoD policy, practice, and culture will be needed.[1] A bibliography for readers who wish to investigate this subject matter further is provided at the end of this technical note.

The processes used to gather information in this investigation included:

- interviews with government and contractor staff who have been, or are currently engaged in, requirements definition or management activities using Agile principles
- discussions with experts in the application of Agile methods in the commercial space, some of whom also have significant experience applying them in highly regulated environments like the DoD
- literature reviews
- participation in government and industry groups that are attempting to provide guidance for Agile methods implementation in the DoD or other federal government agencies

1.2 Agile and the DoD

The DoD is becoming more dependent on using software to deliver needed capabilities both from an initial development perspective and in sustainment. Improvements in software life-cycle models and software best practices have followed a difficult journey that accelerated in the 1980s as the engineering roots of software management methods continued to fail to deliver acceptable software in a timely manner. The turn of the millennium also generated a turning point in software engineering and management philosophy that first produced success in relatively unconstrained

[1] Addressing some of the cultural disconnects between Agile and DoD culture is addressed in SEI 2011-TN-02, *Agile Methods: Selected DoD Management and Acquisition Concerns,* Lapham et al.

(at least by regulation) commercial software engineering environments but also started to be used on the edges of the DoD acquisition environment. The community term for this set of methods and principles is Agile.

This technical note provides results of an investigation into Agile methods with a special focus on requirements development and management in support of DoD acquisition professionals. It leveraged relationships with industry and DoD early adopters of Agile method approaches to address the effective application of requirements development and management approaches for large and small DoD programs, in addition to leveraging an extensive literature search.

A frequently cited document on this topic from the Defense Science Board Task Force, *Department of Defense Policies and Procedures for the Acquisition of Information Technology*, was published in March 2009. It highlighted several recommendations targeted at encouraging use of Agile methods in the DoD [DSB 2009]. The report also raised several concerns regarding existing requirements approaches in traditional DoD acquisition programs, including:

1. programmatic-centric "up-front" processes that only capture "user needs" as specific requirements in approved documents (100 percent solutions)
2. the lengthy Joint Capabilities Integration and Development System (JCIDS) requirements process for approvals and changes
3. user requirements specified by the Pentagon staff rather than operational users
4. independent requirements processes managed by each of the armed services, potentially resulting in conflicting processes, especially for joint programs

The net results of these existing approaches are that:

1. requirements are outdated even before they are approved
2. there is minimal support for commercial off-the-shelf (COTS) information technology (IT) solutions
3. interfaces among acquirers, developers, and users are insufficient
4. too many urgent requirements (time-urgent needs) are handled outside institutionalized processes

We assert that some of these issues can be addressed effectively by adopting Agile methods within particular programs. For example, using a method that includes significant face-to-face interaction among developers and real end-users could help to address Concern 3 and Result 4 above. Others will be far more difficult, and will require DoD policy-level agreement on changes to acquisition guidance. For example, in today's DoD, some types of requirements changes, even if agreed to by program stakeholders, still need additional approval to comply with JCIDS guidance.

One additional issue that arises regarding a DoD adoption of Agile methods is the concern that Agile is simply the latest silver bullet. And although there are many claims about the efficacy of Agile throughout many industrial domains, there are also many critics who doubt that Agile is as valuable as its proponents suggest. For instance, a recent and highly optimistic report from the Standish Group indicated that Agile projects show a success rate of 42% while traditional projects show a success rate of 14%.[2] However, this report also has been widely criticized; in an IEEE

[2] http://www.sds-consulting.com/blog/agile-projects-successful-3x-more-non-agile-projects

paper by J. Laurenz Eveleens and Chris Verhoef, they state that the figures in the Standish report are "misleading, one-sided, pervert the estimation practice, and result in meaningless figures."[3]

The authors of the present report are aware of this dilemma. But while it is true that there is little compelling empirical evidence, there is abundant anecdotal evidence that Agile methods can be effective in multiple settings of differing scales. There is also the clear recommendation of the Defense Science Board for adoption of methods with Agile properties by DoD. Hence, we have focused this report on acquisition professionals who are already planning on using Agile methods (whether voluntarily or as a result of a mandate) rather than trying to either justify or condemn the use of Agile methods.

1.3 The Document-Centric Approach to Requirements

One major characteristic of present DoD practice, labeled a "document-centric" approach, is that requirements are exhaustively defined, documented, and ostensibly frozen before any substantive development begins. To aid in this task, there is a complex and lengthy process, the Joint Capabilities Integration and Development System requirements process, through which approvals of and changes to the requirements documents must be made. As noted above, the length and complexity of this process was one of the chief concerns of the DSB Report on IT acquisition.

In truth, one reason for the DoD's desire to discourage requirements changes at any point after approval is that such changes usually affect contract terms (and usually negatively); this is a fate that has plagued and derailed many large projects. Another reason is that requirements changes, especially the addition of many new requirements, have often been associated with unnecessary wish lists; this phenomenon is referred to as "requirements creep." Hence, a methodical, formalized, document-centric approach that attempts to define and then freeze requirements is an attempt to prevent such occurrences. This need for stability of requirements is magnified in system development settings that involve custom hardware that must be manufactured for the system rather than procured off the shelf. But in so doing, the focus on completeness and stability of requirements has had unfortunate side-effects, most notably those called out by the DSB: user requirements specified by the Pentagon staff rather than operational users, and outdated requirements resulting from long delays in system delivery.[4]

1.4 The Implementation-Driven (Agile) Approach to Requirements[5]

By contrast, Agile methods adhere to a principle of *welcoming* changing requirements as developers and users learn more about what is feasible and desirable, and as they respond to ever-changing threat and mission environments [BENS 2009, NRC 2010, Tech America 2010]. Requirements changes are welcomed even late in development; this philosophy is typically discouraged in a document-centric approach.

---

[3] http://agile.dzone.com/articles/industry-report-project

[4] Note that the document-centric approach to requirements is undergoing considerable research in the business community; the European Commission is proposing its use in several "virtual enterprise" projects. See "A Document Centric Approach for User Requirements in BIVEE," Sinavi et al, 2013.

[5] This section provides only a brief overview of Agile methods; in Section 2 we provide a fuller description of the principles of Agile, and further details on its practice.

Agile methods specifically incorporate a system's real end users into the processes of establishing, prioritizing, and verifying requirements, which are typically formalized at the "last responsible moment." Precisely when that point occurs will differ across development contexts, but generally, requirements are formalized just prior to design and implementation. Agile proponents have a worldview that organizations are complex adaptive systems, in which requirements are *emergent* rather than *prespecified* [Boehm 2004].

Using an Agile method, a program would embark on a development project with a product vision that bounds the requirements, and a general roadmap of major evolutions towards the vision, but without explicitly documenting most of the *detailed* functional and nonfunctional requirements for the system. Agile methods have various mechanisms for specifying higher level product vision and operational needs, and we have seen multiple cases where system-level requirements are specified in the traditional way, with the software requirements not being detailed until close to design time. Above all, Agile relies on empowered teams to perform many of the tasks that, in current DoD practice, are performed early in the life cycle by acquisition personnel who are not themselves end users of the prospective system. The Agile teams determine which subsets of requirements are needed to start and how detailed the requirements need to be specified for the team to be able to begin its design and implementation work. That work explicitly includes frequent conversations with end users or knowledgeable surrogates to ensure developer understanding is sufficient to meet the users' needs. Teams produce working code as early as possible, and (at least informal) releases of code are frequent, even possibly daily (daily release, though not uncommon in industrial settings, is not something we have seen in our research in DoD settings).

In sum, the biggest shift in the Agile philosophy on requirements is that the development team is expected to directly include the end users, testers, and other stakeholders, in order to interpret and refine the requirements at a point in time that is close to the design and implementation activities of the software development. This is in stark contrast to a development team that simply implements a set of requirements approved earlier, conceivably years earlier, by a requirements approval authority with variable connections to the operational users.

1.5 Summary and Document Contents

Having described, albeit briefly, some key characteristics of both the document-centric approach to requirements and that of Agile, we need to consider in more detail the similarities and differences between them, and also to consider the strengths and weaknesses observed for each. Before doing so, however, we first provide in Section 2 a fuller description of how the Agile approach actually works. Following that, Section 3 will compare and contrast the two approaches, and Section 4 will discuss in detail some of the issues that might arise if DoD were to consider broadly using Agile methods, and speculate on the types of projects for which that approach would be optimal. These issues primarily arose via the interviews we conducted, the literature search conducted for the study, and the authors' experiences with both traditional and Agile development contexts.

In Section 5, we examine various ways in which Agile methods would fit with existing DoD acquisition structures. Section 6 describes the details from our interviews, and Section 7 is a summary of the entire document. In the course of our interviews, there was often discussion about some of the more fundamental requirements-related acquisition issues that software-reliant systems face. As a reminder that these are also part of the environment in which software acquisition

must operate, the appendix summarizes requirements-related acquisition issues that are neutral to Agile or document-centric approaches, but which are unique to developing software within a larger acquisition context.

2 The Agile View of Requirements

2.1 What We Mean by "Requirement"

The term *requirement*, as used in this document, connotes "a condition or capability needed by a user to solve a problem or achieve an objective." This definition is consistent with DoD definitions [DoD 2012]. Other definitions found in DoD policy and guidance documents include

1. Requirements are characteristics that identify the accomplishment levels needed to achieve specific objectives under a given set of conditions [Vannucci 2010].
2. A requirement states what the system is supposed to accomplish but does not specify how the system is to perform [Vannucci 2010].
3. Requirements are binding statements in a document or in a contract [Fairley 2009, Mistrik 2010].

There has been considerable research into requirements, both from the software as well as the larger system perspective. Requirements can be viewed from many different perspectives, e.g., user requirements, product requirements, system requirements. Requirements are often divided into functional and non-functional requirements [Leffingwell 2007].The latter class consist of such qualities as usability, dependability, (i.e., the "ilities"), and are often called "quality attributes," rather than non-functional requirements. Requirements can also be *derived*, that is, requirements that are implied or transformed from higher-level requirements. For example, a requirement for long range or high speed may result in a derived design requirement for low weight.[6]

Requirements are critical for engineers in general because they represent everything the engineer cannot change—everything else is in the engineer's design space [Larman 2003]. As a result, the important questions for an engineer at any point in the development life cycle are "What decisions are locked down?" and "Who can change those decisions?" From a software engineer's perspective in particular, there is a set of decisions at each level of decomposition of the software that resulted in the requirements for that level of decomposition. Those cannot be changed, but there is also a set of design decisions that need to be made to further the implementation. These design decisions will impose a set of requirements for the next levels of decomposition, and so forth.

2.2 The Agile Manifesto, Principles, and Life Cycle

To understand the Agile view of "requirement," it is necessary to refer to a foundational document for the Agile community, the Agile Manifesto, published in 2001. This document, and documents that succeeded it, lay out the key tenets and principles of the approach, together with a view of the life cycle it requires. The text of the Manifesto:

"We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

[6] *Systems Engineering Fundamentals,* Defense Acquisition University Press, 2001

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more." [Agile Manifesto 2001]

An important, but often overlooked statement in the manifesto that appears after the four tenets is: "Though we value those things on the right, we value the ones on the left more." This statement reinforces the perspective of the signators that elements like "working software" are not meant to *replace* "comprehensive documentation" but when faced with a choice of how to communicate progress, for example, working software would be the favored approach.

Based on these tenets, the following 12 Agile principles are

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

And finally, the Agile life cycle, necessarily an evolutionary development life cycle, has the following five characteristics from our viewpoint:

- *iterative*—elements are expected to move from skeletal to completely fleshed out over time, not all in one step
- *incremental*—delivery doesn't occur all at once
- *collaborative*—progress is expected to be made by stakeholders and the development team working collaboratively throughout the development time frame
- *parallel*—multiple self-organizing, cross-functional teams work concurrently on multiple product elements (e.g., requirements, architecture, design, and the like for multiple loosely coupled product components)
- *time-boxed*—relatively short-duration development cycles that permit changes in scope rather than changes in delivery time frame

These tenets, principles, and life cycle enable:

- adaptive planning
- iterative development and delivery
- a time-boxed iterative approach, and
- rapid and flexible responses to requirement changes

They provide a conceptual framework that promotes expected interactions in changing requirements, which are welcomed for either the customer's competitive advantage or the warfighter's combative advantage throughout the development cycle.

We will describe in a later section how applying these principles to DoD requirements development and management would necessarily bring about significant changes in current practice. For the moment, we will now examine how Agile development projects are conducted, both for small as well as large projects.

2.3 Requirements Development and Management in Agile SW Development

This section describes general practices related to requirements development and management in Agile settings. Our assumption is that readers have a basic understanding of common Agile methods, particularly Scrum and eXtreme Programming. For those unfamiliar with the basics of Agile development, the SEI's first technical note in this series, *Considerations for Using Agile in DoD Acquisition*, provides definitions of commonly used Agile terms and provides, for the DoD reader, a perspective on various general opportunities and issues associated with the use of Agile development in DoD settings.

Agile development was initially used primarily in smaller development projects, but in more recent years larger projects and programs in both commercial and government settings are adopting or adapting Agile methods. We describe Agile requirements in small projects first, since this is the most common context. We then describe the significant differences between requirements in small settings and larger Agile settings.

2.3.1 Basics of Agile Requirements Development in Small Projects

This section presents basic concepts of Agile software requirements development in small projects. In order to illustrate various aspects of the approach, we use selected features of two widely

used Agile approaches: *Scrum* (namely, the concepts of *product owner*, *product backlog* and *self-organized teams*) and *eXtreme Programming* (*epics* and *user stories*). These concepts are often used together on a single project.

Agile development, as currently applied in industry and government, has always been well suited to small applications projects [VersionOne 2012]. This is the context in which Agile methods originated and it is still a context in which they excel. The approach assumes the presence of a knowledgeable customer/user who has a clear understanding of the prioritized needs to be satisfied by the system to be built. The approach also assumes that a number of small teams will exist, and that they are empowered and are largely self-organizing and self-managing. In accordance with the Agile Manifesto, self-organization is the optimum mechanism for incentivizing and motivating a team to produce the best results [Agile Manifesto 2001]. A common organizational construct in industry and government development settings, an integrated product team (IPT), could be configured to resemble an Agile development team. IPTs vary in purpose and implementation, particularly in the degree and approach to decision-making granted to the team. IPTs have been used in at least one government setting (Department of Homeland Security (DHS)), with explicit roles configured to accommodate Agile principles and practices. Where IPTs would not reflect the norms of Agile teams is when the members of the IPT do not have decision-making authority for making requirements changes (at the level agreed to by the organization's management or charter) during the development iterations. Agile teams include a number of defined roles. The role that is key to understanding how Agile approaches to requirements development and management work is the product owner.

The product owner's role is to provide the development team with significant insight into the user problem space. Though this individual often is an end user, this is not mandatory, so long as he or she has a sufficiently clear vision of user needs to get started on designing and building the solution [Beck 1999, Cockburn 2000, Cohn 2004].

At the beginning of a project, only the very high-level requirements are known, and high-level concept documents, or groups of known user and technical requirements, would be formulated into what is generally called a "product backlog." One of the tasks of the product owner, in conjunction with rest of the development team, is to determine which subsets of requirements are needed to start and how detailed the requirements need to be specified for the team to be able to begin its design and implementation work. An assumption in this work is that the product owner will be available to answer development team questions as they go through the design and implementation process.

Product backlog items could be traditional requirements statements at an appropriate level of abstraction. They are more frequently expressed as "user stories."[7] As shown in Figure 1, they are the basic unit of work. The technique is a popular way of collecting data and communicating about stakeholders' needs. An "epic" is a group of related user stories. Stories are a means to describe "features," which loosely parallel the notion of requirements: a unit of functionality that delivers specific business value. When planning an iteration of work, an Agile team decides on some number of features to be implemented, and defines the tasks necessary to complete them.

[7] User stories have also been adopted as a requirements specification mechanism by many teams using Scrum, even if those teams don't use other aspects of eXtreme Programming.

Note that Figure 1 is highly simplified; these relationships are commonly considerably more complex than shown here.

The team's objective is to define, build, and test some number of features within the time box of the iteration, typically one to two weeks. Each iteration has a short, sometimes intensive, development cycle. In Agile, the organization of the requirements and the organization of the team itself are not independent: teams organize around the requirements (the backlog) to optimize the efficiency of defining, building, testing, and integrating the code that delivers value to the customers. The entire team is involved in defining requirements, optimizing requirements and design tradeoffs, implementing them, testing them, integrating them into a new baseline, and then seeing to it that they are delivered to the customer.

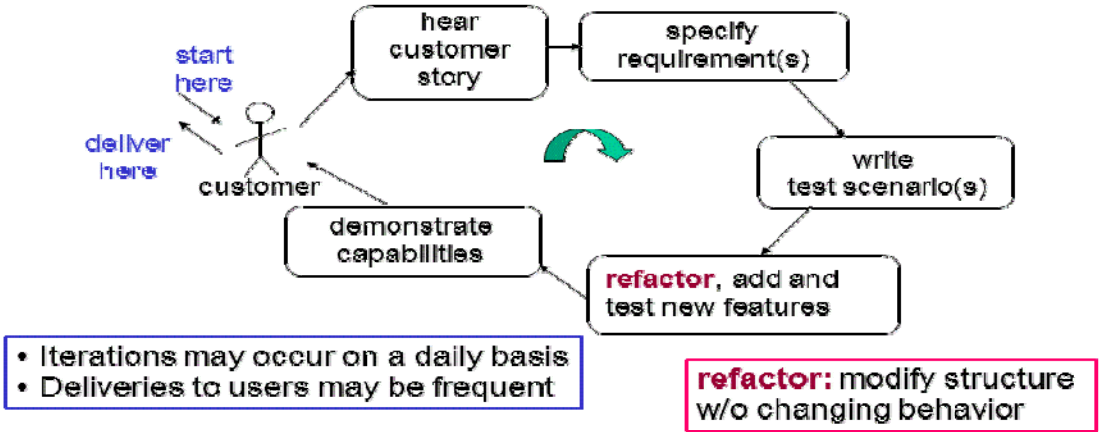

Figure 1: Agile Development for Small Projects[8]

For example, the requirements for a given Agile project could be to build an "official" demonstration version every week. Developers may do more frequent "sandbox" builds, thus several ongoing tasks are ready and in progress. Product versions (what story to take on next) are built according to the priority of requirements set by the product owner in conjunction with the team's knowledge of inter-task dependencies, which would have been discussed and accounted for in the iteration planning session.

Another common Agile practice is to explicitly use "information radiators" that provide detailed information on the status of the work to all stakeholders in the project. Often, these take the form of wall murals and sticky notes, but most online Agile support tools have electronic versions that perform the same function. The information radiator is a place where any team member or stakeholder can see the progress of requirements they are interested in. It also visually highlights stories that are blocked. Often this has a beneficial effect of getting engagement from the parties that are needed to resolve the blocked story so that progress can be resumed.

In Figure 2, it is easy to see which tasks and User Stories are Ready, In Process, Done, or Blocked (the tasks with italic burgundy sticky notes attached). Team members can pull from the board tasks that they are capable of performing from any of the tasks in the "ready" column.

[8] From Fairley, Richard. Managing & Leading SW Projects, Wiley, 2009. Used with permission.

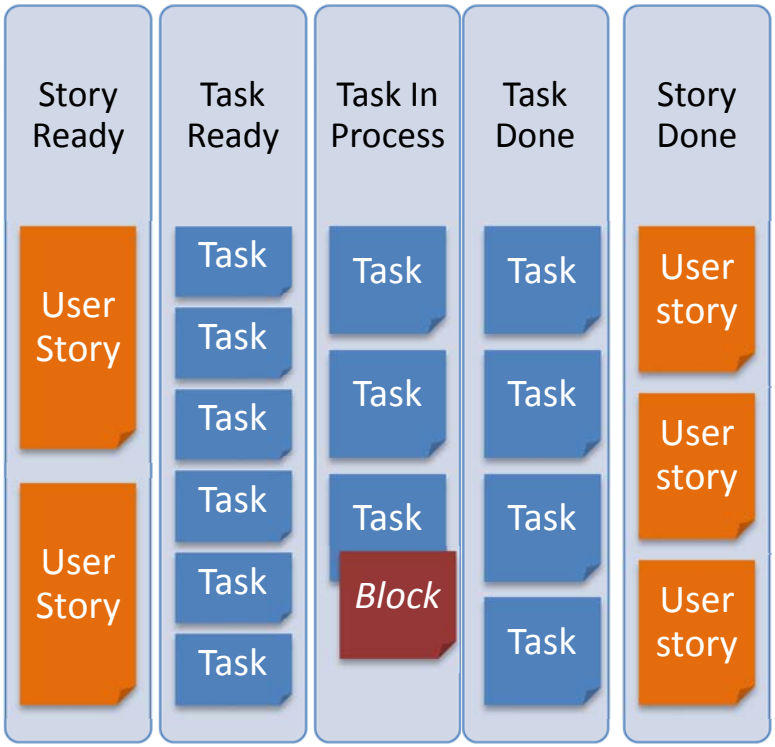

Figure 2: Dynamic View of Work in Progress via an Information Radiator

The integration of requirements analysis into each iteration is a hallmark of Agile methods and is relatively easily achieved in small team settings.

In a small acquisition context, the management of requirements via a product backlog and user stories can be accommodated by contracting constructs like task orders. Some of our interviewees are able to use task orders within a larger indefinite delivery, indefinite quantity (IDIQ) contract to specify the product vision for a release, which is then, as part of the task order, collaboratively decomposed into a product backlog. Frequent iteration reviews and daily interactions between the development team and product owner have proven to be a successful approach for oversight. In at least one case, a small United States Air Force program evolved over several years into a very successful, self-sustaining system using the requirements approach described above. The approach also scaled successfully up to a 100-person development organization.

2.3.2 Scaling Agile Requirements Analysis from Small Settings to Larger Contexts

In successful small Agile settings, the close collaboration between developer and end user visibly speeds the time it takes to move from product concept to working software. In larger settings, there are typically more teams, more layers of management between the teams and their stakeholders, and more communication complexity inherent in ensuring that relevant stakeholders have the information they need to make decisions that progress the product. Industry and government organizations alike have had challenges in retaining the close collaboration aspect of Agile approaches while managing the increased complexity of a large product design and implementation.

Several well-known authors in the Agile community have documented approaches to scaling. Two of these represent two ends of the thinking on how best to scale Agile practices and norms beyond a single team. Craig Larman uses a bottom-up approach, focusing on adding just enough structure at each layer of the architecture to ensure communication and stakeholder involvement, while minimizing non-value-added organizational structures and practices. [Larman and Vodde, 2010]

Dean Leffingwell and his team, the creators of SAFe, the Scaled Agile Framework, have envisioned more of a comprehensive organizational transformation that incorporates lean concepts like kanban[9] at the business/portfolio level, a construct called a "release train" that organizes teams to deliver an element of a product vision, and traditional iteration teams at the lowest layer. SAFe incorporates additional roles at the release train and portfolio levels of the organization, and explicitly addresses architecture as an ongoing aspect of the product life cycle. Figure 3 shows the Scaled Agile Framework as of this writing. Currently, it is in greater use in commercial industry than government settings. However, some of our interviewees had begun using SAFe to organize their larger Agile developments.

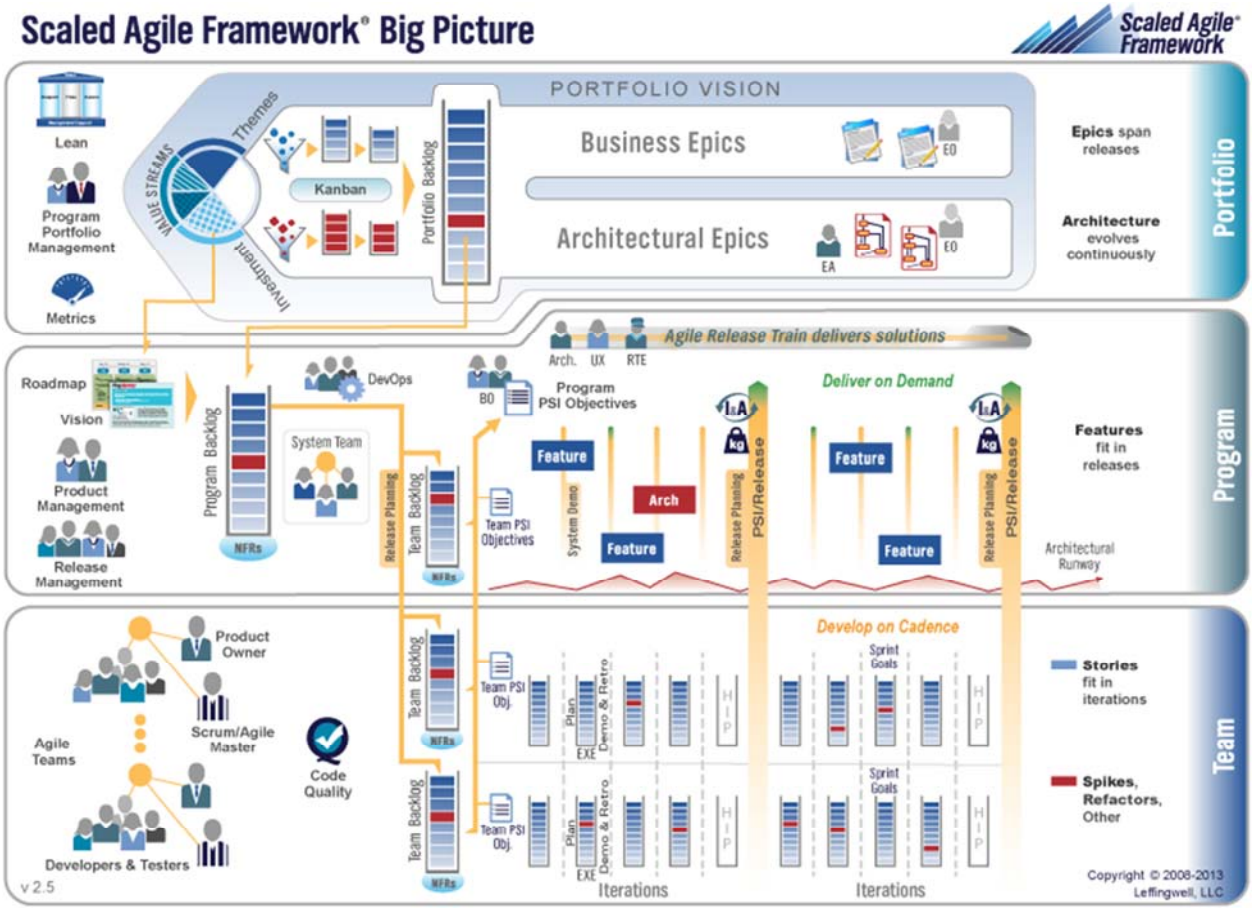

Figure 3. Scaled Agile Framework Big Picture. Used with permission per website guidelines.

[9] Kanban is a technique for managing workflow originating from the lean engineering methods pioneered by Toyota. [Reinertsen 2009]

SAFe in particular looks at the flow of requirements into Agile release trains and teams as a kanban-ish pull function. The concept is that a business (or operations unit) knows what themes it is willing to invest in, and using portfolio management approaches that incorporate lean thinking, the organization can construct the large-grained user and architecture stories that release teams can then organize around to deliver useful incremental deliveries of products that reflect the organization's mission and product vision.

Key differences in requirements development and management between small and large settings include the following:

- how the role of the product owner is organized
- how technical (i.e. architectural and platform), as opposed to business or user, requirements of large systems are accounted for
- how requirements are organized across multiple elements of an overall program or project

The first difference is about how the product owner role is organized. In small teams, the product owner is usually a single individual who is interacting with one or a few teams. In a large program with potentially dozens of development teams, a single product owner is impractical for multiple reasons. A product owner team is often the construct used to organize and coordinate the efforts of multiple product owners across teams.

The second difference relates to the increased importance of the role of system and software architecture, as well as technology platform, in constraining the software solution. Although some agilists favor the concept of allowing the software architecture to evolve via the implementation, several Agile authors (Ambler and Leffingwell among them) cite the importance of architectural roadmaps and/or architectural on-ramps as part of the overall system life cycle. Larger projects have the potential for accumulating large amounts of technical debt[10] and postponing addressing architectural issues can have significant implications for a large system, especially one with custom-built hardware that interacts with the software.

The third difference relates to how the software requirements are treated within the context of a larger system development, especially one with custom hardware. In some larger Agile settings, requirements dependencies among teams are treated via constructs like "scrum of scrums" meetings, where team-level facilitators raise issues to others facilitating teams and getting resolutions to potential conflicts that they take back down to the smaller teams. Others use organizational constructs like the product management team to pull together the product owners across multiple releases and ensure that the relevant dependencies are being dealt with at the right time. Some Agile support tools contain visualizations of dependencies across user stories and epics that make clear some of the stories that need to be handled earlier, before dependencies constrain the overall solution.

---

[10] Technical debt refers to the work that a team knows needs to be done to maintain the stability of the system or improve its performance that is deferred until after higher-priority user needs are implemented . For more information on technical debt, see the SEI's blog post, "Strategic Management of Architectural Technical Debt," http://blog.sei.cmu.edu/post.cfm/strategic-management-of-architectural-technical-debt

A struggle for scaling Agile requirements practices in industry and government is deciding how much documentation is enough. Requirements documentation has traditionally been used as a primary communication vehicle during the early evolution of a system concept, and is used in government to define acceptance criteria for the system, then as an approval gate to proceed further in the development. We use the term “document-centric” to describe approaches that follow this philosophy, while we use “implementation-driven” to describe approaches, particularly those from the Agile community, that favor producing working software as a communication vehicle and progress measure.

In the next section, we highlight the conceptual and philosophical differences between document-centric and implementation-driven requirements approaches. These differences underlie many of the issues that we saw in our interviews and other research activities.

3 Differences Between Document-Centric and Implementation-Driven Approaches with regard to Requirements Development and Management

In Section 1 we considered, at least briefly, the different ways that requirements are developed and managed in document-centric and Agile approaches. We can summarize those differences as follows:

For the document-centric approach:

- assigned stakeholders create formal documents as the expression of "what to build" that must be approved prior to use in further design and implementation
- verification and validation of the requirements occurs as a (generally) complete set prior to substantive design and implementation
- changing the requirements, regardless of source, is a time-consuming and expensive process designed to aggressively control change

For the implementation-driven, Agile approach:

- emphasize direct, face-to-face communication about requirements between the stakeholder group (end users, acquisition personnel, systems engineers, etc.) and the development team
- use devices such as user stories that focus on the user or business need to be met and provide a place to start a conversation versus a detailed specification of exactly what to build
- use the direct interaction of the development team and stakeholders to verify and validate that the requirements being worked are valued and relevant
- permit the stakeholders and development team to informally change what is being built to take advantage of learning about the system characteristics and evolution in the stakeholder's knowledge about the environment

3.1 Strengths and Weaknesses of each Approach

The document-centric approach to requirements in the DoD is a fundamentally hierarchical approach to capturing the user's operational requirements and codifying them via increasing detail in a requirements specification document or requirements databases. It provides the obvious benefits of visibility at a point in time when the documents are signed and placed under configuration control, thus providing a means to control requirements changes based on an identified baseline. Completion contracts can then be written against that baseline with some degree of certainty, thereby limiting the DoD's risk of incurring additional costs. If the requirements, and the environment in which the system is meant to operate, do not significantly change, the document-centric approach reduces cost and risk associated with requirements change management.

In sum, the strengths of the document-centric requirements approach are

- enables the comparability and repeatability that standardization provides

- enables a contractually verifiable definition of completed intermediate work products
- reduces risks by means of contractually assured baselines

Weaknesses of the document-centric approach include:

- the process drives measurement of compliance with itself as a primary measure of success (i.e., rather than measuring success as deploying a workable solution) [Fairley 2009]
- it depends on documents as the basis to verify and validate the requirements, the architecture, and the detailed design
- most of the requirements are completed before any code is written, thus extending development timelines

Agile methods employ short, iterative development cycles; actively involve users and other stakeholders to establish, prioritize, and verify requirements; and verify implementation of requirements in working software within the short iterations [Highsmith 2000]. The strengths of this approach include

- early insight by the users into the shape of the solution
- early course correction
- "fail fast" (If the early solution ideas turn out to be flawed, little time or money is spent before that learning occurs.)
- explicit understanding that the requirements are expected to evolve

Weaknesses of Agile requirements approaches (particularly in large acquisition settings) include

- more dependence on tacit knowledge (e.g., lack of explicit documentation) as the basis for decision-making than is comfortable for most acquisition organizations
- dependence on availability of actively engaged user/customers
- difficulty in aligning implementation-driven artifacts and measures with those of the larger traditional acquisition setting.

3.2 Problems Inherent in each Approach

The strengths of the two approaches detailed above would suggest that each has certain situations for which it is an especially appropriate choice. However, the weaknesses listed above also suggest that each approach will bring with it certain problems. We list the problems of each of these perspectives, particularly for large programs, in the table below.

Table 1: Document-Centric and Implementation-Driven Requirements Problems

Document-Centric Requirements Problems in Large Development	Implementation-Driven Requirements Problems in Large Development
Likelihood of disconnects between those eliciting and documenting requirements and stakeholders who will operate and sustain the system Different stakeholders in the process have more privilege, in terms of approval rights, than others, which can reduce the overall veracity of the requirements	Finding authoritative product owners who can adequately represent stakeholder needs and priorities Managing the conflicting priorities of multiple product owners across a complex system Getting the attention of verification and validation agents early and often to ensure early versions are workable

Document-Centric Requirements Problems in Large Development	Implementation-Driven Requirements Problems in Large Development
Fast pace of technology and environment evolution obviates some requirements prior to implementation	Creating a rhythm of deployment of the evolving system that is tolerable by operations and logistics stakeholders
Requirements management processes that inhibit change reduce likelihood that obviated requirements will be eliminated or replaced	Coordinating the rhythm and expectations of the different Agile teams with the systems engineering teams that they are interacting with
Extended review times for requirements in an attempt to assure their completeness, even though it is known that they cannot be perfect prior to implementation	

3.3 What Kinds of DoD Programs Would Benefit from Agile and Which Would Probably Not

From the discussions in Section 2 and Sections 3.1 and 3.2, a picture begins to emerge of the types of acquisition contexts for which each approach—document-centric or implementation-driven—might be more appropriate.

Based on our interviews and other research, the following contexts appear to be adequately served by the document-centric approach. An inherent advantage to the document-centric approach in today's DoD acquisition context is that this approach is consistent with the acquisition life cycle guidance provided in the DoD Acquisition Deskbook and its supporting documents.

- programs with stable requirements and environment, with known solutions to the requirements
- programs with a homogeneous set of stakeholders who communicate well via documents
- programs for which the technology base is evolving slowly (technology is not expected to be refreshed/replaced within the timeframe of the initial development)

The acquisition contexts that appear to be best served with an implementation-driven, Agile approach include:

- programs with volatile requirements and environment
- programs where solutions are sufficiently unknown that significant experimentation is likely to be needed
- programs for which the technology base is evolving rapidly
- programs with stakeholders who can engage with developers in ongoing, close collaboration

When looking at the above list, our team concluded that, in reality, no acquisition context that we have seen is "ideal" for either the document-centric or implementation-driven approach. Often, acquisition programs with volatile requirements have limited access to the stakeholders that they need to engage with to assure requirements validity. Further, programs with stable requirements are not only few and far between, but often do not have homogeneous stakeholders nor do they communicate well via documents. These observations argue that the requirements context is not a sufficient determinant alone of which approach is more suitable. The SEI's 2011 Technical Note, *Agile Methods: Selected DoD Management and Acquisition Concerns*, addresses many of the other factors that should be considered [Lapham 2011].

4 Potential Barriers to use of Agile Requirements Methods in DoD Acquisition

In the interviews we carried out with numerous acquisition professionals, a number of issues came to light that could (and often did) pose barriers to using Agile methods, particularly those relating to requirements, on DoD acquisition programs. Several of these issues are described below. They are not completely discrete, as is the case with many aspects of acquisition practice. Our goal is to shed light on both the policy issues that are currently under debate in the acquisition community (i.e., those related to Agile methods) and some of the challenges that practitioners in actual acquisition program offices are facing when trying to implement Agile requirement approaches.

The six most significant issues that were raised were:

- accepted DoD-level guidance for tracking progress
- translating potential Agile measures of progress to earned value measures
- a risk-averse acquisition culture
- work breakdown structure as a tool to manage task assignments
- effect of requirements change on contracts
- perception that reduced documentation is a cause for concern

For each of these issues, the following sections:

- describe the issue that the interviews revealed, and why it poses a barrier to use of Agile methods
- characterize both the implementation-driven and Agile aspects of the issue
- where possible, provide a sanitized example of how this was manifest in specific programs
- provide suggestions for potential solutions

4.1 DoD Guidance

Issue: Accepted DoD-level guidance for tracking progress

Discussion: Current DoD acquisition guidance is manifestly targeted to a document-centric approach; in particular, all of the approved methods for tracking progress on a project are based on documentation. And though the need for an alternative to the current approach was stressed by many of the persons we interviewed, until Agile tracking methods are standardized and proven in a multitude of DoD environments, they will not be accounted for in DoD acquisition guidance. While Agile methods have demonstrated success in varied DoD environments, those methods are still evolving in many ways. This presents, in effect, a “chicken-or-the-egg” problem.

The obvious answer is that it is time for agreement on a widely-accepted Agile life-cycle approach that satisfies DoD’s need for tracking progress, preferably within the DoD 5000 series approach. Some of the thought leaders of the Agile community are sympathetic to this need. They

offer an increasingly wide variety of methods, all of which reflect the Agile tenets and principles. [Leffingwell 2010, Ambler 2013]

However, there are difficulties that must be overcome. The variety of methods and practices that are associated with the Agile principles makes it challenging to create a set of "generic Agile" progress measures and tracking mechanisms that can be acceptable for DoD acquisition programs. Partly this is because a fundamental theme underlying the Agile approach is trust. In contrast, the DoD acquisition guidance has been built (for a variety of reasons) upon the principle of verification, which has the unfortunate side effect of reducing trust.

Example: One successful Air Force program that has been using Agile methods is trying to move into a program-of-record status after several years of being outside the mainstream acquisition process. Its acquisition office wants to retain the high quality and high user satisfaction that the program has been able to achieve while meeting current DoD acquisition guidance. The trust level between the acquisition office, the development team, and the 600-plus user organizations for the software is very high, so many of the document-based verifications that might be useful in another environment represent wasted effort for this community. So far, compliance has been accomplished through waivers of certain prescriptive elements that the program has demonstrated are not necessary for it to meet the programmatic and user goals.

Potential solutions: All the programs the authors talked to and interacted with agreed that having appropriate DoD 5000-level policy guidance would make application of Agile methods easier for both the acquisition program office and the developers, whether in-house or contractor. Some programs we spoke with have been given approval by the systems engineering or other oversight groups to keep the requirements baseline at a higher level of abstraction than is typical, provided that they have sufficient user and stakeholder involvement during implementation (documented by informal meeting notes or even through digital photos of working diagrams on flip charts). These informal mechanisms assure those in oversight that developers get a sufficient understanding of both user needs and technical constraints.

The program cited above shows at least a partial solution: it has reviewed its oversight guidance and has made explicit requests for waivers of clauses that it knows, from its multiyear successful experience of using Agile methods, do not provide sufficient value add to their customers in comparison to the amount of work required to comply.

In sum, creating Agile-aware mechanisms for tracking progress that have the same level of sponsorship as those currently in DoD 5000.02 would provide the blessing that some programs feel is needed before actively embarking on Agile method adoption. Note that this solution eventually must go beyond just tracking progress, or requirements development and management, but is needed in many other project areas as well.

4.2 Translation of Progress Measures

Issue: Translating potential Agile measures of progress to earned value management

Discussion: In a wide variety of programs, earned value management (EVM) approaches as usually practiced have emphasized document-centric measures as the primary evidence of program progress, particularly in the early parts of a program. Reporting progress using EVM is an accepted practice for both developers and acquisition personnel. But in Agile methods, the lack of focus

on detailed requirements generation before design and implementation makes the traditional translation of progress into EVM (which is very often mandated) awkward and challenging. There are three aspects where Agile has a poor fit with EVM: communication, measurement, and decision-making.

In Agile methods, communication is achieved via frequent meetings and face-to-face (even if sometimes that face-to-face is over a network connection) dialogues. Similarly, Agile methods demand that measurement be team-focused and uses such devices as information radiators to communicate critical project information and decisions. Teams are expected to self-monitor and act based on what they learn from implementation and from measuring their progress. And finally, Agile decisions are team-based with frequent retrospectives[11] to improve practices. All of these manifest some variance with an EVM approach. The translation of the team-based measures and practices into the schedule-centric EVM approach typical in many programs requires additional work that is typically considered non-value-added by the team, and provides a distorted picture of the actual value delivered, since not delivering a feature could be considered a good thing if the development team and product owner agree that it is no longer needed.

Example: A government project underwent significant extra scrutiny, exception management meetings, etc. because their earned value data showed they were not delivering as predicted. However, the product owner, after seeing the demo of the functionality after the first iteration, had agreed that several of the items in the release backlog were not needed based on the implementation approach that the team had discovered. So the team was doing what the product owner wanted, but was showing "bad" performance from an EVM viewpoint.

Potential solutions: There is a growing area of research into applying the mathematical basis for EVM to the kinds of measures that emerge naturally from Agile processes. To be successful, an "Agile EVM" approach will necessarily include both a theoretical basis and a method for practical application. Should this research prove successful, it may prove possible for certain programs to rely on an Agile EVM method to define their needed progress measures.

Possible additional solutions could include establishing deliverables from the requirements phase that include prototypes and working software as acceptable elements of the documentation package.

And finally, as Agile EVM or equivalent measures are found that provide the predictive power desired by DoD acquisition oversight organizations, a potential longer term solution would be to incorporate appropriate Agile progress measures into Service- or DoD-level guidance on the use of Agile methods.

4.3 Risk Averse Culture

Issue: A risk-averse acquisition culture

Discussion: Among the bases of the current acquisition process is the commendable goal of reducing risk over the acquisition life cycle. Among the measures adopted to accomplish that goal

[11] Agile retrospectives are events that happen at the end of each iteration which are focused on the team analyzing their performance in the last iteration and deciding on candidate improvements to try (if within their scope of decision-making) or suggest to management (if they are things that need management approval) . See Esther Derby's book on retrospectives for details on retrospectives and their use [Derby 2006].

has been the incorporation of more and more activities to cope with individual lessons learned on specific programs, rather than limiting general guidance additions to address systemic, common problems. The result is that navigating the DoD's Planning, Programming, Budgeting & Execution (PPBE), JCIDS/requirements process, and MAS acquisition support systems is complex.

One unfortunate side effect of these lengthy processes is an overall environment that is inherently risk-averse; another is that entrenched behaviors are difficult to change. For example, the current cultural tolerance with the long length of time for acquisition programs to complete may have some relation to the fact that many in the government sincerely believe that it must take years to deliver capabilities because in the recent past, it always has taken a long time [Gilligan 2012b]. Another side effect is a tendency of larger programs toward larger increments and less rapid delivery. In their defense, program managers often start with the objective of rapid fielding only to be consumed by the mandated documentation and well-intended oversight activities that quickly erase any hopes of achieving the desired delivery tempo. The bottom line is that risk aversion often leads to behaviors that do not embody the trust that is a hallmark of well-functioning Agile cultures.

Example: A program hired a SETA contractor to produce the documentation required by their contract, even though the program office and developer had agreed that much of the documentation required was non-value-added and would not be useful, either during or after the initial development. However, the program manager was new to acquisition and felt that producing the documentation (using some of the cost savings he was experiencing through the close collaboration with his users and developer) was less risky from a program compliance viewpoint than trying to get waivers for the non-value-add documentation.

Potential solutions: Whether through guidance exception or guidance change, our interviews indicate a widespread belief that significant benefit would be achieved if a less risk-averse culture could be attained; our sources indicated further that such technologies as Agile methods, when successfully implemented, could help bring this about. But successful Agile methods adoption requires a move toward more trusting relationships among all stakeholders in acquisition. Potential solutions involve chartering and sustaining support for DoD-level integrated product teams (IPTs) that involve Agile-aware participants working toward the mission of achieving aligned, Agile-aware DoD policies across the existing acquisition processes. As authors who have observed and participated in several similar cultural shifts, we are well aware of the difficulties of achieving this goal.

4.4 Work Breakdown Structure

Issue: Work breakdown structures are not compatible with Agile methods.

Discussion: Within DoD acquisition programs, work breakdown structures (WBSs) are commonly encouraged as a way of organizing and communicating progress on work. In a document-centric systems engineering approach, a WBS creates a hierarchical decomposition of the system requirements, which constitutes an "is a part of" relationship and a system-level architecture. [12]

[12] This issue is of considerably greater scope than the other issues discussed in this section. Therefore, in the following section, where we focus on the relationships between Agile and the acquisition process, we will discuss more fully the question of the relationships between system engineering and software engineering, and their effect on software requirements, whether using Agile or otherwise.

For example, an airplane can be decomposed into a wing plus several other components, a wing can be decomposed into an aileron plus other components, and an aileron can be decomposed into a motor plus other components; thus, a motor is part of the aileron, the aileron is part of the wing, and the wing is part of the overall aircraft system. Furthermore, the accomplishment of work would be monitored and controlled based on the accomplishments of the work packages (i.e., via the earned valued management system) related to each component in the WBS hierarchy.

However, when we add the software element, then whether we take a document-centric approach or an Agile one, use of a WBS poses some difficulty for the software engineering processes. First, the systems engineer typically allocates the system's software requirements based on the "is a part of" decomposition (via a software requirements specification) and associated systems-level architecture. The software engineer then takes the allocated software requirements and creates a software architecture by which software requirements are satisfied, which is often based on an "is used by" relationship (e.g., some element of the software "is used by" the aileron and possibly by other elements of the system architecture).

As a result, the system-level, hierarchically decomposed architecture for the hardware does not directly map to the software architecture. This is because a software processor may obtain and provide information and have interfaces at different levels of the system as well as to components that are not directly linked from a physical perspective.

For example, the processor may pass information to the aileron motor; it may also pass information to the gyro navigation system and information from other components of the attitude control system, such as the rudder control system. In this case, information from this processor is used by components that are not at the same level in the work breakdown decomposition and not physically tied together by an "is a part of" relationship.

The differences between the system and software views of the system make communication difficult and imprecise between the users, the system engineers, and the software development teams. It also makes it difficult for the software engineer to effectively use the WBS as a project management tool, since the WBS is a hierarchical decomposition of the system based on "is a part of" relationships versus "is used by" relationships. As a result, the software engineer often gauges the productivity of the teams at the story board level as part of the estimation process and then must map these estimates into the appropriate project work breakdown elements as an extra, non-value added (to the software work) step.

And when we consider an Agile approach to software, Agile requirements management approaches generally do not rely on the WBS approach to define work. Instead, they rely on interaction among the users, stakeholders, and development teams to produce the larger scale commitments, with the Agile teams themselves performing work decomposition.

Example: A government program has explicitly limited the use of Agile methods to the implementation phase of software system development because they couldn't figure out how to incorporate the cross-functional team aspects of Agile development into the WBS for the architecture and design tasks. This reduced much of the utility of Agile methods in terms of fostering communication across the different system stakeholders.

Potential solutions: Some observers to whom we spoke took the most negative position possible: Agile is simply not suited for use in a system such as the airplane example described above. We do not concur with the position, though we are aware of the inherent misfit between Agile and a complex, hardware-driven WBS.

Our view is that while a "pure" Agile approach may not be reasonable, there are many aspects of Agile that could usefully be borrowed on such a project. In some programs we have seen, Agile-type user stories are accepted as a form of intra-program communication, and some systems engineers have started to think of them as a subset of the "concept of operations," (CONOPS). It is helpful (from an Agile perspective) that user stories can often be associated with different parts of the physical elements, making clearer the conflict between "is used by" and "is a part of" concepts.

Some programs in the sustainment world that we interviewed who are using Agile methods have married the task order structure they use for formulating Agile project scope with the WBS. The task order becomes the WBS item, and below that level the team self-manages using the Agile methods common to their development organization.

However, in the wider sense, as noted above, the WBS approach has always been problematic for software development, since software functionality often is specifically relied on to cross component and system functionality boundaries. As Agile systems engineering approaches become more prevalent, structures other than traditional WBS may become more prevalent. Perhaps they will be structures that focus on capabilities versus components. Research is ongoing on the evolution of systems engineering practices to encompass Agile methods and is the topic of a future SEI technical note in our Agile Adoption series. Even the International Council on Systems Engineering has convened a working on group on the applicability of Agile principles and methods to systems engineering practice.

4.5 Effect of Requirements Change on Contracts

Issue: Requirements change often affects contract terms and conditions.

Discussion: As was pointed out in an earlier section, one principal reason for discouraging requirements changes at any point after initial approval is that such changes almost definitely will affect contract terms and conditions. Historically, the outcome of this is most often from terms favorable to the government to terms more favorable to the developer. This is frequent even on large government projects.

Hence, there is an understandable desire to make a painstaking effort to produce a set of requirements that will withstand attempts to change them. In fact, the acquisition community has an abundance of expressions (e.g., "nail down the requirements") that reflect this thinking.

However, the shared experience of decades of software-intensive programs, in fact the very experiences that led to the Agile movement itself, is that attempting to "nail down the requirements" is too often a hopeless task. Agile methods accept as a given that building a system is often accompanied by greater insight into what the system really should do, that people change their minds, that projects are often overtaken by changing technology, that different people in leadership of a program have different ideas about what should be built, and so forth.

Hence, if requirements are written initially at too low a level of abstraction, and then are impossible (or nearly impossible) to change, then the power of Agile methods—to change downstream requirements as more is learned about the system from users, testers, and other stakeholders—is lost.

Example: A joint program created a new type of requirements document to ensure that requirements that had to go through the formal change control processes were not written at too detailed a level. Documents that derived from the higher-level requirements document could be changed with the agreement of the product owner and did not need to go through an Engineering Change Proposal (ECP). This eliminated tremendous schedule and dollar costs on the program.

Potential solutions: In our interviews, we witnessed several programs that are pursuing adoption of Agile methods. Almost all of them are affected by this issue, and the resultant difficulties it poses. One possible solution might be to view software development as a service to the larger system. If the acquisition program office is sufficiently skilled to manage a software time and materials contract, then Agile requirements management approaches like collaborating on a prioritized backlog are feasible. At least two Air Force programs manage their Agile projects this way. One is more sustainment focused, but the other started out as a major system upgrade. Having an appropriate contracting mechanism for Agile projects can also be useful in mitigating issues related to work breakdown structures.

A longer term solution, as mentioned in previous issue discussions, would be a change in acquisition guidance to recognize that the continuing evolution of requirements could make dealing with requirements change easier, particularly for programs using Agile methods.

4.6 Perception That Reduced Documentation Is a Cause for Concern

Issue: Reduced documentation is a cause for concern

Discussion: Since much of the requirements documentation in the documentation-driven approach has the purpose of supporting decision-making, it is understandable that any reduction in documentation can be viewed as a cause for concern by those who have program oversight responsibilities.

We argue, however, that this need not be so. Engineering decisions made solely on the basis of documentation often fail to account for the tacit knowledge of the end users and other stakeholders. Further, understanding which documentation is really necessary to support downstream stakeholders (e.g. sustainment-focused, user-focused documentation), and which should be used to record early decisions is often a difficult challenge. Even realizing that these two classes of documentation exist is a helpful first step in establishing the right level of documentation.

It is the latter focus of documentation—the early decisions in the early days of the project—that is deemed not as necessary in Agile projects. Developers are actively communicating at the detailed level with end users, but only on the details of the requirements for the bounded portion of the system being worked on during an iteration. This can succeed only when the appropriate end users are made available to answer developer questions, and to participate in requirements validation. And it can only work when the program office is a willing participant in the process.

Potential solutions: In several Agile implementations we have observed, documentation was definitely a significant part of the process, but the generally agreed goal was "just enough documentation." Documentation is still used to capture knowledge that needs to be explicit and codified, such as architecture views, end user documentation, maintenance checklists, and training manuals. Documentation that will be used in operations and sustainment tends to be regarded as extremely important. But where appropriate, conversation often can replace documentation to support some issues of technical decision-making.

One program we participated in provided special training to stakeholders and oversight staff on what to expect from the software part of a program using Agile methods as the preliminary design review approached. This included a description of the levels of abstraction and degree of "doneness" they could expect in the documents that were provided.

A potential longer term solution could focus on adding a specific Agile-aware life cycle choices section in DoD-level acquisition guidance. As experience with Agile projects increases, changes in the norms for how acquisition decisions can be productively made in government settings using Agile methods will be proposed. One program successfully uses a user conference concept to bring end users of their widely deployed software together every one to two years to review and prioritize product backlogs, learn about the strategic direction of the products, and learn from each other tips and tricks for optimizing use in operation. User support as the basis for decision making is a powerful basis for moving forward with implementing requirements.

4.7 Moving Beyond Document-Centric Requirements Definition and Management

The above discussions focus equally on systems and software acquisition. A key issue that contributes to all of those cited above is that the same acquisition approach is being used to acquire both hardware (software-reliant/embedded/weapon systems) and software-reliant systems (information systems and business systems). Over several decades now, this approach has proven to be ineffective [DSB 2009]. We continue to need an approach that allows for the use of appropriate software development techniques, such as Agile methods, within a large software-reliant systems context.

Jack Ferguson and Michael DeFiso provided the following recommendations with respect to looking at the requirement phase of the acquisition life cycle in their SEI special report titled *Software Acquisition: A Comparison of DoD and Commercial Practices* [Ferguson 2004]. Their recommendations are relevant to several of the issues discussed in this section.

1. Reduce ambiguity by extensive simulation and prototyping.
2. Specify data model and global standards.
3. Maintain prototypes as the baseline throughout the development to quickly analyze changing requirements.
4. Eliminate delay and improve quality and efficiency by specifying standard interfaces to minimize data manipulation.
5. Perform functionality/cost tradeoffs early in the requirements phase to determine if dramatic time or cost savings can be obtained with relatively small changes to requirements.
6. Maximize flexibility by stating system performance and functional requirements as broadly as possible, consistently with supporting the intended mission.

7. Tailor documentation requirements to emphasize those items needed by the end user to understand, use, and maintain the final software product and minimize the amount of (often unused) documentation associated with recording each step of the design process.

The above recommendations are consistent with supporting Agile requirements approaches, and although they are generally at a lower level of detail than prior potential solutions, they could contribute to solving some of the issues cited in this section.

In the next section, we will provide a summary of actual interview results, followed by a summary and an Appendix.

5 Interview Results: Effective Application of Requirements Development and Management Approaches for DoD Programs

This section provides a summary of the questions and comments used in the interview portion of this research study. No details on the participants are provided as part of our agreement for confidentiality.

Participants in the interviews included both government and industry representatives. All of the participants had working knowledge of one or more Agile acquisition or development methods and experience in acquiring or developing large, complex software systems used by the DoD. The purpose of each interview was to collect their experiences and perceptions about the effectiveness of the requirements development and management processes reflected in Agile methods as they might apply to the large, complex systems acquired and developed by the DoD. The discussion questions and representative summary-level responses are presented in Table 2. These interview results contributed in large part to our characterization of the issues found in Section 4. The questions in the table are the general questions that we used for getting conversations on various topics started; however, often an interviewee answered in such a way that their comments contributed to understanding of more than one question. When that was the case, we put the responses in line with the question that would have normally elicited that response.

Table 2: Interview Results

Representative Interview Questions	**Representative Responses**
Thrust: Scope of Requirements and Role of Documentation Can a project be reasonably started without explicitly documenting all the functional and nonfunctional requirements for the system?	The overall answer is yes. Requirements will change as the system is decomposed from the top-level requirements to the lower level requirements. In general, it is better if users initially state what they want in terms of capabilities or objectives than if they provide detailed requirements. Agile does not mean documentation is not required with respect to requirements development and management. It does mean just enough documentation. The systems requirements should be stable and documented prior to passing the software-allocated requirements to the software team. The software requirements required for each iteration need to be defined for the pending deliverable software at the end of each iteration. Working software and any documentation that is required for that software iteration are deliverable work products.

Representative Interview Questions	Representative Responses
Thrust: Tailoring of Requirements and Leveraging of Commercial Practice Given your sense of how Agile approaches are being used to develop IT and software-reliant systems in commercial industry, can aspects of their approaches be tailored for use in DoD?	The overall answer is yes. The development of IT systems is leading the way with respect to Agile method implementation and tailoring based on the needs of their customers. Similar warfighter needs exist (e.g., faster delivery of capabilities) within DoD and should drive DoD in implementing Agile method approaches by leveraging the experience of commercial practice. A review of the 12 Agile principles indicate that only a few apply directly to software; the remaining are good business principles that are either being applied or could be applied on DoD programs.
Thrust: Requirements Evolution and Learning Is it best to evolve the requirements as more is learned about the system and deliver incremental capabilities based on this knowledge versus the more prescriptive approach of defining and satisfying all the requirements in one major delivery?	The overall answer is yes. Modern software systems are becoming more complex, and the environment where these systems operate is becoming more and more dynamic. A consequence of this trend is that the average number of requirements for a software system increases and changes continually as more knowledge about the system and user needs is gained. Requirements, design, and development phases are mechanisms for "knowledge acquisition." A concern was raised that the type of contract can be a business factor. Fixed price contracts with firm requirements limit requirements evolution and learning. A more agile contracting approach is needed, versus the current document-centered approach for the engineering of product requirements to rapidly respond to and exploit changing conditions in incremental deliveries. The Agile approach to requirements must be systematic, especially with respect to accommodating legal and nonfunctional requirements.
Thrust: Incremental Software Requirements Decomposition and Delivery Schedule Impacts Question 1: Given stable system requirements, can software requirements for large projects be broken down such that a program can incrementally develop capabilities? Question 2: Can this incremental requirements decomposition approach be used and still deliver all the capabilities in less time than current prescriptive methods?	The overall answer is yes to Question 1 and is "it depends" to Question 2. On prescriptive software development projects, we talk about capturing the requirements in documents. What is actually meant by this phrase is producing documents that specify a system in sufficient detail that software development can proceed by reference to these documents alone. Four general points were made with respect to this line of reasoning: (1) Although it is theoretically possible to capture user knowledge about requirements in a document from which the software engineer can extract the same knowledge without distortion, this may not be a practical or efficient way to communicate requirements for complex software systems. (2) Requirements can and are usually decomposed into

Representative Interview Questions	Representative Responses
	incremental capabilities (e.g., computer software configuration items). (3) In determining a requirements process, it is important to remember that the primary purpose of a development project is to deliver a software system that generates value to a business and/or warfighter. (4) Although the time to deliver the entire set of required capabilities may or may not be longer using Agile methods, an incremental set of capabilities delivered in a timely manner is usually more satisfying from a cost-benefit perspective to the user than waiting for the entire set of required capabilities to be delivered.
Thrust: Requirements and Architecture Evolution Considerations Given that requirements evolve during development, is it important not to lock down a requirement until it becomes architecturally or functionally significant to do so?	The overall answer is yes. Requirements drive the architecture. Requirements evolve for a number of reasons, including the changing vision of the user for the system. As a result, architectures evolve. There needs to be sufficient definition of requirements at any point in time so that the resultant architecture is everything that needs to be consistent throughout a software system. Stated another way, the architecture is everything that is expensive to change later. There are risks associated with requirement and architecture evolution versus having all the requirements defined and a fixed architecture defined upfront; however, it is usually not realistic to assume that all the requirements can be defined. The functional requirements are easier to manage than the nonfunctional or quality requirements. As a result, it is important to identify the most important nonfunctional requirements as soon as possible so that they can be managed and incorporated into the architecture as appropriate.
Thrust: Agile Shareholder Requirement Communication Relative to the communication of requirements among stakeholders, does the use of Agile methods make communication clearer among the users, the system engineers, and the software development teams compared to prescriptive methods?	The overall answer is “it depends.” Communication is one of the fundamental values of Agile methods, although it would be more accurate to say that effective communication is what Agile methods deem critical to a developer’s and acquirer’s success. In Agile methods, developers and users must communicate if they are executing the practices. Given that DoD’s acquisition processes are centered on weapon system acquisitions where the ongoing communication of the user and developer is not designed into the acquisition approach, introduction of new Agile approaches that require ongoing communication will meet with cultural resistance. There is a fundamental communication barrier among software engineering and other forms of engineering,

Representative Interview Questions	Representative Responses
	such as systems engineering, due to the inherent nature of the software (e.g., software's invisibility, changeability, complexity for its size, need to conform to its host environment). Stated another way, software engineering in unlike other forms of engineering as other forms of engineering are like unto themselves. These inherent attributes of software lead the software engineer to approach requirements development, architecture (logical design), and physical design differently than other forms of engineering. These two different approaches among the engineering communities can lead to communication issues independent of the use of Agile methods. Introducing some new approach may lead to confusion, but this is not likely since software engineers have been using Agile methods for a long time and have found mechanisms to report results regardless of the acquisition approach.
Thrust: Perception of Prescriptive Process as It Relates to the Length of Time It Takes from Requirements Formulation to Delivery of Capabilities Does it take too long to deliver IT system capabilities to the warfighter largely because the current DoD acquisition, requirements, and budgeting approaches do not adequately support the delivery of incremental capabilities?	The overall answer is yes. There needs to be a better alignment among the three major DoD decision support systems (i.e., PPBE, JCIDS, MAS) in order to reduce the acquisition cycle time. Although the Defense Acquisition System gets most of the blame for the excessive acquisition delays, the other components, such as the JCIDS requirements process, are also significant contributors. One of the significant factors in fully implementing Agile methods within the DoD is culture change with respect to the three major DoD decision support systems. When the decision makers feel that the operational requirements are urgent due to an impending threat, the three major DoD decision support systems can be modified to permit a more Agile environment; however, this situation is rare in today's environment.
Thrust: Perception of Cultural Change Needed Within DoD to Implement More Effective Requirements Formulation to Delivery Acquisition Life-Cycle Processes Based on Agile Methods Can a large DoD program use components of the Agile methods, processes, and tools where (1) authority for making decisions is further from project execution (i.e., JCIDS), (2) decisions take longer to make, and (3) the people who are doing the work are geographically dispersed across multiple companies?	The overall answer is yes. Currently there are some ongoing Air Force program pilots that are successfully delivering capabilities. These programs are tailoring their Agile method approaches based on the developer's and acquirer's needs. The former issues associated with geographically dispersed teams are decreasing in importance in implementing Agile methods due to maturation of communication tools.
Thrust: Requirements Satisfaction and Technology Transfer Relative to initial demonstration of requirements satisfaction, are there types or classes of systems that should be considered first for the transfer of Agile methods: (1) IT, (2) business systems, (3) weapon systems, (4) cyber systems, (5) large systems, (6) small systems, (7) systems of systems, and (8) other types of systems?	The overall answer is yes. Based on the interviews, the following types of systems should be considered first for the transfer of Agile methods: information systems, cyber systems, and small systems. One of the key discriminators was the ability of these systems to adapt commercial capabilities and technologies for military needs and the relatively lower

Representative Interview Questions	Representative Responses
	acquisition cost.
Thrust: Requirements Satisfaction and Business Effectiveness From a business perspective, should the use of Agile methods be determined based on a trade-space analysis of the organization's desire for productivity and the organization's tolerance for ambiguity, flux, and uncertainty?	The overall answer is no. OSD is a huge organizational enterprise with formal structures that are difficult to change and its acquisition practices are risk adverse. While this type of trade-space analysis may be desirable for commercial industry, the DoD cultural limitations inhibit its effectiveness. It may be better to encourage DoD Agile method advocates to encourage decision makers to first prototype systems where the benefits of Agile methods on DoD programs can be demonstrated and provide the results to senior decision makers.

Although there were nuances in perspective among the interviewees, the overwhelming view was that acknowledging the incompleteness of requirements at the beginning of development and using implementation-driven methods to learn as you build the system are both productive steps toward improving the timing and quality of software-reliant system delivery. The associated viewpoint was that making this shift is one of the more difficult aspects of Agile implementation in the DoD.

6 Summary

This technical note has provided results of an investigation into Agile methods with a special focus on requirements development and management in support of DoD acquisition professionals. It leveraged relationships with industry and DoD early adopters of Agile method approaches to address the effective application of requirements development and management approaches for large and small DoD programs, in addition to leveraging an extensive literature search. Our most significant finding was that making a significant change in DoD to better align its existing PPBE, JCIDS, and MAS document-centric acquisition support systems from a requirements top-down prescriptive delivery focus to a more agile, implementation-driven focus where requirement changes are more welcome across the entire the acquisition life cycle is one that, if pursued, will require a significant cultural shift [Lapham 2011].

Even though commercial industry has successfully made this transition and recent federal laws like the 2010 and 2011 Defense Authorization Acts use wording that heads in this direction, progress has been slow, partly because this change represents a dramatic change in requirements development and management approaches and an associated cultural shift [DAA 2010, 2011]. Namely, a document-centric orientation focuses the various activities to generate and baseline the full set of requirements in a specification document and associated documentation required throughout the acquisition life cycle, while the implementation-driven orientation focuses on the delivery of required capabilities via executable software. In the former, governance is achieved through measurement of artifact production and activity completion, and in the latter, governance is achieved through measurement of incremental outcomes and progress and quality trends. This governance change represents a significant cultural shift in terms of DoD oversight responsibilities.

It is not just the requirements development and management activities that must be done more quickly to achieve faster deliveries; contracting, testing, and project oversight decisions will need to align with this objective. Although those activities are not the focus of this technical note, we concur with our interviewees that a holistic, systematic cultural change approach will need to occur that has full stakeholder commitment to achieve widespread Agile methods implementation.[13] DoD's history in making large scale cultural changes is variable, making it understandable that senior officials and program office personnel alike are hesitant to embark on such a change. This is not to say that Agile methods have not or cannot be implemented within the current environment, as some programs within OSD have already proven with their successful implementations of Agile methods. Communications with some of these early adopters indicate they have overcome some resistance but still face varying levels of resistance to implementation-driven approaches.

An increasing number of commercial organizations have embraced one or more of the Agile methods approaches, producing software with less documentation under conditions of rapid change to the high satisfaction of the user. However, DoD is one of the largest and most complex

[13] Other technical notes from the SEI focus on some of these issues, and future research is planned for topics like testing and contracting.

organizations on the planet, which, for some good reasons, has instantiated a risk averse acquisition framework. As a result, DoD acquisition decision makers are perplexed about the risks involved in using Agile methods on large programs even though the potential benefits (as seen at least in unregulated commercial settings) are very attractive. In an effort to support DoD acquisition professionals in the use of Agile methods, this research investigation disclosed a set of requirement development and management acquisition issues that inhibit Agile method usage within DoD. Note that some of these requirements issues also plague acquisition practitioners in document-centric programs. The difference is that, over time, conventions have arisen for dealing with requirements development issues in traditional software-intensive acquisition environments that, though not covered by DoD guidance, are generally accepted in practice.

The acquisition issues we raise in this paper are most productively considered when a program is formulating its acquisition strategy, and we are seeing cases across the military services where projects that want to effectively use Agile methods are being given waivers to some regulatory constraints so that they have the opportunity to demonstrate their ability to deliver needed capabilities in an implementation-driven setting. As mentioned in relation to several of the issues, longer term solutions to the issues that negatively affect DoD adoption of Agile requirements development and management techniques will require DoD-level policy and guidance changes, and those who have embraced a risk averse acquisition culture will surely wait until those changes occur to change their own behavior. Early adopters like the U.S. Food and Drug Administration and Department of Homeland Security have already published local guidance for use of Agile methods as an alternative to their traditional life cycle approaches. In addition, interaction with individuals from the SEI Agile Collaboration Group and interaction with other government staff who are implementing Agile in their settings make us hopeful that their successful use of Agile methods will speed the guidance transitions that will make Agile methods use easier for the programs to which they are most relevant.

Appendix: The DoD Acquisition Challenge

The DoD has the acquisition challenge to deliver to the warfighter new capabilities based on user requirements with speed, efficiency, and effectiveness [NAS 2010]. DoD also has challenges to ensure equitable distribution of funds and protect skills in the industrial base. The current prescriptive Acquisition Management System and associated major support systems, the DoD requirements system (Joint Capabilities Integration and Development System [JCIDS]) and budgeting system (Planning, Programming, Budgeting, and Execution [PPBE]) are currently not agile enough to effectively and efficiently meet these challenges, based on government reports [DSB 2009].

There is a difference of views on how to tailor the current major acquisition support systems to meet DoD needs; however, there is no disagreement that the current approach needs to be changed [DSB 2009]. As a result, DoD organizations have been mandated to develop alternative life-cycle approaches in order to deliver IT capabilities to the end user more effectively and efficiently [NDAA 2011]. Currently there is neither supporting policy interpretation nor sufficient supporting transition mechanisms for alternative life-cycle methods, such as Agile approaches [DoD 2010]. A fundamental area of concern is in the area of requirements development and management. There is skepticism among some DoD decision makers that some aspects used in current Agile methods for requirements development and management approaches used on small projects will scale for usage on the larger, complex DoD developmental efforts. As a result, the views of some key DoD stakeholders are that the lack of documented benefits of using Agile method approaches on large DoD systems and the associated lack of documented guidance on appropriate implementation mechanisms for addressing scalable Agile approaches are showstoppers for Agile adoption [Stober 2010]. On large programs the OSD Office of Cost Assessment and Program Evaluation and service-level costing organizations can go deep in their costing methodologies where it is impossible to get an approved cost position (which is required) without doing a deliberate design up front, which can take an extensive amount of time. Other dimensions that make Agile adoption difficult in DoD are addressed in other SEI technical notes.

References/Bibliography

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

Primary References

[Agile Manifesto 2001]
Cockburn, Alistair et al. *Manifesto for Agile Software Development*. 2001. http://www.agilemanifesto.org

[Ambler 2013]
Ambler, Scott. *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise,* IBM Press, 2013.

[Beck 1999]
Beck, K. *eXtreme Programming Explained: Embrace Change*. Addison-Wesley, 1999.

[BENS 2009]
Business Executives for National Security (BENS) Task Force on Defense Acquisition Law and Oversight. *Getting to Best: Reforming the Defense Acquisition Enterprise*. July 2009.

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

[Cockburn 2000]
Cockburn, Alistair. *Writing Effective Use Cases*. Addison-Wesley, 2000.

[Cohn 2004]
Cohn, Mike. *User Stories Applied: For Agile Software Development*. Addison-Wesley, 2004.

[DAA 2010]
U.S. Congress. Defense Authorization Act 2010.

[DAA 2011]
U.S. Congress. Defense Authorization Act 2011.

[DAR 2010]
House Armed Services Committee Panel on Defense Acquisition Reform. “Interim Findings and Recommendations,” *DAR Interim Report.* March 4, 2010.

[Derby 2006]
Derby, Esther & Larsen, Diane. *Agile Retrospectives: Making Good Teams Great.* The Pragmatic Programmers, LLC, July 1, 2006.

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

[DoD 2010]
Department of Defense. *A New Approach for Delivering Information Technology Capabilities in the Department of Defense* (OSD Report 13744-10). December 9, 2010.

[DoD 2012]
Department of Defense. Definitions, *DoD Instruction 1035*. April 4, 2012.

[Fairley 2009]
Fairley, Richard. *Managing and Leading Software Projects*. Wiley, 2009.

[Ferguson 2004]
Ferguson, Jack & DeRiso, Michael. *Software Acquisition: A Comparison of DoD and Commercial Practices* (CMU/SEI-94-SR-009). Software Engineering Institute, Carnegie Mellon University, 2004.
http://www.sei.cmu.edu/library/abstracts/reports/94sr009.cfm

[Gilligan 2012b]
Gilligan, John. *A Roadmap for Reforming the DoD's Acquisition Information Technology Programs, Defense (DoD) Information Analysis Center (IAC)*. April 2012.

[Highsmith 2000]
Highsmith, Jim. *Agile Project Management: Creating Innovative Products*. Pearson Education, 2000.

[Lapham 2011]
Lapham, Mary Ann, Miller, Suzanne, Adams, Lorraine, Brown, Nanette, Hackemack, Bart, Hammons, Charles (Bud), Levine, Linda, & Schenker, Alfred. *Agile Methods: Selected DoD Management and Acquisition Concerns* (CMU/SEI-2011-TN-002). Software Engineering Institute, Carnegie Mellon University, 2011.
http://www.sei.cmu.edu/library/abstracts/reports/11tn002.cfm

[Larman 2003]
Larman, Craig. *Agile and Iterative Development: A Manager's Guide*. Pearson Education, 2003.

[Larman 2010]
Larman, Craig & Vodde, Bas. *Practices for Scaling Lean and Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum*. Addison-Wesley, 2010.

[Leffingwell 2007]
Leffingwell, Dean. *Scaling Software Agility: Best Practices for Large Enterprises*. Pearson Education, 2007.

[Leffingwell 2010]
Leffingwell, Dean. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Addison-Wesley Professional, 2010

[Mistrik 2010]
Mistrik, Ivan, et al. *Collaborative Software Engineering*. Springer, 2010.

[MITRE 2012a]
MITRE. *DoD Agile IT Handbook*. March 2012.

[NAS 2010]
National Academy of Science. *Critical Code—Software Producibility for Defense*. Report of Committee on Advancing Software-Intensive Systems Producibility (ASISP), 2010.

[NDAA 2011]
National Defense Authorization Act for Fiscal 2010. "Section 804: Implementation of New Acquisition Process for Information Technology Systems." January 7, 2011.

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

[Reinertsen 2009]
Reinertsen, Donald. *The Principles of Product Development Flow: Second Generation Lean Product Development*. Celeritas Publishing, May 29, 2009.

[Stober 2010]
Stober, Thomas & Uwe, Hansmann. *Agile Software Development: Best Practices for Large Software Development Projects*. Springer, 2010.

[Tech America 2010]
Tech America. *Recommendations on Information Technology Acquisition Reform*. Report to the Defense Acquisition Reform Panel of the House Armed Services Committee. February 23, 2010.

[Vannucci 2010]
http://www.dtic.mil/ndia/2010systemengr/ThursdayTrack1_10812Vannucci.pdf

Version One 2012]
Version One. *7th Annual State of Agile Dev Survey*, Version One, 2012.

Bibliography

Adolph, Steve, Bramble, Paul, Cockburn, Alistair, & Pols, Andy. *Patterns for Effective Use Cases*. Addison-Wesley, 2002.

Anderson, David. *Kanban*. Blue Hole Press, 2010.

Black, S. E., Boca., P. P., Bowen, J. P., Gorman, J., & Hinchey, M. G. “Formal Versus Agile: Survival of the Fittest.” *IEEE Computer 42,* 9 (September 2009): 37-45.

Blank, Steven Gary. *The Four Steps to the Epiphany: Successful Strategies for Products that Win*. Cafepress.com, 2005.

Brooks, Fred. *The Mythical Man-Month: Essays on Software Engineering.* Addison-Wesley, 1975.

Brown, Nanette, Nord, Robert, & Ozkaya, Ipek. “Enabling Agility Through Architecture.” *CrossTalk* Nov/Dec (2010): 12-17. http://www.crosstalkonline.org/storage/issue-archives/2010/201011/201011-Brown.pdf

Chow, T. & Cao, D.-B. “A Survey Study of Critical Success Factors in Agile Software Projects.” *Journal of Systems and Software, 81,* 6 (2008): 961-971.

Clements, Paul et al. *Documenting Software Architectures*. Addison Wesley, 2010.

Cockburn, Alistair. *Crystal Clear: A Human-Powered Methodology for Small Teams*. Addison-Wesley, 2004.

Cockburn, Alistair. *Agile Software Development: The Cooperative Game*, 2nd ed. Addison-Wesley, 2008.

Cohn, Mike. *Succeeding with Agile: Software Development Using Scrum*. Addison-Wesley, 2009.

Cooke, N. J., Salas, E., Cannon-Bowers, J. A., & Stout, R. “Measuring Team Knowledge.” *Human Factors 42,* 1 (Spring 2000): 151-173.

Dettmer, H. William. *Goldratt's Theory of Constraints: A Systems Approach to Continuous Improvement*. ASQ Quality Press, 1997.

Gilligan, John, McCoy, Diann, & Heitkamp, Kenneth. *Rapid IT Acquisition: A New Model for Acquiring Government Information Technology*. Acquisition Solutions, November 2009. http://www.asigovernment.com/documents/RAPID_IT_SPECIAL_Advisory.pdf

Gilligan, John. *Reforming Acquisition of DoD Information Technology – Now*. August 29, 2012.

Goldwater-Nichols Act. *DoD Re-Organization Act*, 1986, Public Law 99-433.

Hantos, Peter & Kern, Nancy. *Life Cycle Alignment Concerns in Acquisition of Software Intensive Systems*. 2010 COCOMO International Forum. Aerospace Corporation, 2010.

Henderson, Derek E. & Gabb, Andrew P. *Using Evolutionary Acquisition for the Procurement of Complex Systems*. Technical Report # DSTO-TR-0481. Salisbury: DSTO Electronics and Surveillance Laboratory, 1997.

Herbsleb, James D. & Mockus, Audris. “An Empirical Study of Speed and Communication in Globally Distributed Software Development.” *IEEE Transactions on Software Engineering 29,* 6 (June 2003): 481-494.

Kennedy, Michael N. *Product Development for the Lean Enterprise: Why Toyota's System is Four Times More Productive and How You Can Implement It*. Oaklea Press, 2010.

Lapham, Mary Ann, Williams, Ray, Hammons, Charles (Bud), Burton, Daniel, & Schenker, Alfred. *Considerations for Using Agile in DoD Acquisition* (CMU/SEI-2010-TN-002). Software Engineering Institute, Carnegie Mellon University, 2010. http://www.sei.cmu.edu/library/abstracts/reports/10tn002.cfm

Larman, Craig &Vodde, Bas. *Practices for Scaling Lean and Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum*. Addison-Wesley, 2010.

Leffingwell, Dean. *Scaling Software Agility: Best Practices for Large Enterprises*. Addison-Wesley, 2007.

Martens, Ryan. Ch. 7, "The Essence of Agile with Ryan Martens." *Scaling Software Agility: Best Practices for Large Enterprises*, Addison Wesley, 2007.

McMaster et al. "The SEI's Capability Maturity Model: A Critical Survey of Adoption Experiences in a Cross-Section of Typical UK Companies." 319-334. *Proceedings of IFIP TC8 WG 8.6 International Working Conference on Diffusion*, 1997 .

National Institutes of Standards and Technology (NIST). *Guide for the Security Certification and Accreditation of Federal Information Systems*, NIST Special Publication 800-37. September 24, 2010.

Naur, Peter. "Programming as Theory Building." Jan 12, 2001.

National Defense Industrial Association. *Top Five Systems Engineering Issues Within DoD and Defense Industry.* Systems Engineering Division Task Group Report, 2006.

Office of the Joint Chiefs of Staff. *Joint Capabilities Integration and Development System (JCIDS) Manual.* February 19, 2012.

Pontius , Ronald W. *Acquisition of Information Technology*. Director, Command and Control, Programs and Policy (OSD), 2012.

Poppendieck, Mary & Poppendieck, Tom. *Lean Software Development: An Agile Toolset.* Addison-Wesley , 2003.

Reinertsen, Donald G. *Managing the Design Factory, A Product Developer's Toolbox.* Free Press, 1997.

Ries, Eric. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses.* Crown Business, 2011.

The Standish Group , *CHAOS Manifesto 2011*. March 3, 2011.

Steinberg, Daniel H. & Palmer, Daniel W. *Extreme Software Engineering*. Pearson Education, 2003.

United Kingdom Consortium. *Dynamic System Development Method*. 1995.

Waters, Kelly. “10 Key Principles of Agile Development.” http://www.allaboutagile.com (February 10, 2007).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	April 2014	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Potential Use of Agile Methods in Selected DoD Acquisitions: Requirements Development and Management	FA8721-05-C-0003

6. AUTHOR(S)

Dr. Kenneth E. Nidiffer, Suzanne M. Miller, Dr. David Carney

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2013-TN-006

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)

Adoption of methodologies that generally come under the umbrella of “Agile” in the software development community includes consideration of how those adopting agile methods interface with elements of the acquisition community who provide the requirements and other constraints that govern how the software part of a system will be developed. Several Agile methods have expectations about how requirements are handled that are different from the typical approach used in government acquisition settings. This qualitative research study from the Software Engineering Institute explores issues that practitioners in the field who are actively adopting Agile methods have identified in our interviews about their experience in defining and managing requirements.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Agile, acquisition	58

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