Software Engineering Institute

Agile Methods in Air Force Sustainment: Status and Outlook

Colleen Regan
Mary Ann Lapham
Eileen Wrubel
Stephen Beck
Michael Bandor

October 2014

TECHNICAL NOTE
CMU/SEI-2014-TN-009

Software Solutions Division Client Technical Solutions

http://www.sei.cmu.edu

Carnegie Mellon University

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-0001535

Table of Contents

List of Figures

List of Tables

Acknowledgments

The authors wish to thank the U.S. Air Force for funding this investigation.

In researching this topic, we reached out to a variety of professionals associated with Air Force programs in sustainment. We extend our thanks to

Deborah Brey, Boeing

Dick Carlson, Agile & Lean Education Associates

Robert Frisch, Department of the Air Force, 578 Software Maintenance Squadron / FLT F

Michael D. Hancock, AFLCMC/HIQD

Sarah Sheard, PhD, the Software Engineering Institute

And many others who wish to remain anonymous.

Last but not least, we extend our gratitude to our ever vigilant and extremely helpful editor, Jerry Miller.

Executive Summary

The Software Engineering Institute has been conducting a multiyear exploration of the applicability of Agile software development techniques in Department of Defense (DoD) programs and other highly regulated environments. This paper examines using Agile techniques in the software sustainment arena—specifically on Air Force programs. The intended audience is the staff of DoD programs and related personnel who intend to use Agile methods during software sustainment. Some of the findings may be useful to those working in other highly regulated environments. However, the specific domain studied was DoD projects, particularly Air Force programs.

To ensure our audience starts with the same understanding, we define Agile, traditional methods, software maintenance, and software sustainment. There are plenty of definitions of Agile, but for purposes of this series of reports we use one that is paraphrased as an iterative and incremental approach to software development performed by highly collaborative teams with just enough ceremony to produce high-quality software.[1] Historically DoD software development organizations have used some form of the engineering "V" model or waterfall process to develop software. These methods we term traditional methods. Software *maintenance* consists of correcting faults, improving performance or other attributes, adapting to a changing organization and technical environment, and performing preventive maintenance. Software *sustainment* includes these items but addresses other issues such as documentation, operations, deployment, security, configuration management, training, help desk, commercial off-the-shelf product management, and technology refresh.

Software sustainment is a priority for the Air Force in that operations and sustainment costs can easily reach 60% to 80% of a weapon system's total lifecycle costs [Taylor 2012]. During our exploration of the viability of Agile software development methods within DoD programs, we have learned that software sustainers are increasingly leveraging Agile methods. However, we have not seen many, if any, "pure" instantiations of Agile methods. Rather, we have seen many different hybrids that combine practices from traditional and Agile methods to create a software sustainment method that works well within the given organization's environment. A preponderance of respondents reported significant benefits of using "Agile-like" (hybrid) implementations including substantially lower costs, early deployment, and transparency of action. Trust is reported to have increased among users when correct implementations are manifested within the contract and client organizations (to include stakeholders). Cultural shifts in thinking were reported to be vital to the success of Agile implementation, and while training personnel is highly valued, the availability of appropriate resources or investments in training appears to be less than optimal for all stakeholders.

[1] Agile: "An iterative and incremental (evolutionary) approach to software development which is performed in a highly collaborative manner by self-organizing teams within an effective governance framework with 'just enough' ceremony that produces high quality software in a cost effective and timely manner which meets the changing needs of its stakeholders."
http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm

In investigating the role of Agile in Air Force software sustainment organizations, we identified two types of Air Force sustainment teams. They are organic and contractor based.

Scrum is the most commonly used Agile method so we used it to organize the paper. The Scrum approach implemented by organic sustainment teams was distinctly different from what Scrum meant for a contractor sustainment team. The organic sustainment teams we interviewed appeared to have separate teams of system engineering, software development, and test and evaluation. The contractor organizations we interviewed work to allocate system architects, system engineers, software developers, and testers to every Scrum team. The contractor Scrum organizations applied this multidisciplinary team across subsystems being sustained.

We discuss at length the various constructs within Scrum to illustrate how the organic and contractor sustainers implemented these constructs while performing their sustainment activities. The topics included in the paper consist of

- backlog—product, release, and sprint/iteration. Of the sustainment organizations we spoke with, 75% appear to enter the lifecycle when the product backlog is approved and funded. The sustainment strategy (e.g., contract, organic, federally funded research and development center, or other organizations and combinations) has already been identified and at least some of the adaptive, corrective, perfective, and technology refresh tasks and priorities have been developed, typically without any input from the development teams. This separation from the development teams represents a departure from typical Scrum implementation in which the development team is engaged in creating the backlog.
- sizing backlogs—Our contractor respondents often see the release planning completed by the program office and end using commands with limited input from the contractor. Conversely, organic organization respondents are actively engaged in determining release content and use a known sizing and price model that has typically been very accurate in the past.
- sprint cycles—The interviewees said they worked in two- to five-week iteration cycles. Both the organic and contractor sustainers described this process as a standard implementation of an Agile sprint cycle.
- daily standups—The daily standups occur at the same time each day and last no more than 15 minutes. However, not all interviewees strictly followed this daily standup meeting rule of engagement.
- Scrum-of-Scrum meetings—Larger programs implement a Scrum-of-Scrum standup meeting. This Scrum-of-Scrum meeting is also a communication tool.
- potentially shippable product increments—Universally each organization interviewed agreed they build the release plan to meet a set of needs or requirements and this is a shippable product once test proves the level of quality needed was achieved.
- sprint reviews/demonstrations—Both the organic and contractors attend user-group meetings because this is an opportunity to solicit the “voice from the field.”
- retrospectives—Many of our respondents used the sprint retrospectives to decompress, where many meet for a working lunch to perform the retrospective. The team notes what went well, what went badly, and what needs to improve.

There are other considerations that impacted the Agile sustainers. Hardening sprints are used to address issues resulting from regulatory mandates, policy changes, and information assurance certification and accreditation. Another area of concern revolves around the impacts of DoD personnel changes to the project. These personnel changes impact the understanding and continued use of Agile methods, as the training for using these methods is not at this time included in the traditional career field training available to personnel. Personnel have a natural tendency to revert back to what they know and may even resist the different culture surrounding the employment of Agile methods. To counteract the impact, training in Agile processes needs to be part of the culture in each organization.

During our research we learned that a newer construct for the DoD is starting to emerge due to the growing pressure to reduce the software delivery cycle time while improving the reliability of software development. The idea is sometimes referred to as "delivery on demand and develop on cadence."[2] The construct blends development and operations to create a new term: DevOps. DevOps basically extends continuous integration mechanisms to ensure the software development and operation teams interact much earlier, preferably during the design of features. When an organization uses DevOps concepts, it introduces formal and operational tests earlier in the development lifecycle so that more defects are found "in phase." Adoption of continuous delivery becomes a forcing function to check out the software build on a continuous basis in a production-like environment. Within DoD programs, the degree to which continuous integration extends to continuous delivery is not clear.

Implementing a DevOps culture and continuous delivery principles requires an adaptation to how large DoD programs operate—for these principles to be effective the software development, formal test, and operations teams would need to remove communication barriers and prepare for test and operations personnel to build scripts and use tools earlier in the typical workflow. For example, both formal test and operations teams need to work in the production-like environment "owned" by software development prior to a handoff for formal testing.

Continuous delivery in DoD programs does not appear to be widespread, and if used it is being implemented using "stealth mode" practices. Federal IT teams may use continuous delivery, but none were available for interviews so this statement cannot be substantially verified. DoD and federal IT programs will find introducing formal and operational tests into the software development production-like environment is a forcing function that changes how testers look at boundary tests and the complementary data.

DoD network systems or service-oriented applications connected to the DoD network or federal IT systems should see a benefit from implementing continuous delivery mechanisms and process improvements. Mission capability systems that are designated ground systems supporting airframes, spacecraft, or satellites will need to deploy to pre-production environments much like a DoD network system or service-oriented application. After the checkout in the pre-production environment confirms the system is ready, then cutover to production occurs. Mission capability systems that are designated support equipment or support systems to airframes could implement DevOps at least from software development through formal testing. The DevOps pattern will need

[2] A phrase that is used in discussing the SAFe model consistently but aptly describes what DevOps is about..

to stop short of automating the deployment if the system connects and pushes data to airframe operational flight programs due to operational constraints.

Abstract

This paper examines using Agile techniques in the software sustainment arena—specifically Air Force programs. The Software Engineering Institute has researched the viability of Agile software development methods within Department of Defense programs and barriers to the adoption of those methods for several years. How software sustainers leverage Agile methods and avoid barriers to using Agile methods are addressed in this paper. In addition, the potential use of a construct called DevOps, a blending of development and operations, is discussed.

1 Introduction

This report continues the Software Engineering Institute's (SEI's) multiyear exploration of the applicability of Agile software development techniques in Department of Defense (DoD) programs and other highly regulated environments. Previous papers are listed in Appendix C. In addition, several podcasts and blog posts from the SEI have addressed additional topics. They can all be found on the acquisition research page of the SEI website.[1]

1.1 Defining "Sustainment" in the DoD

Within the DoD, the term "sustainment" has a certain meaning for all programs, whether it is a weapon system (aircraft, missile, or the like), an IT system, space system, or other system. The Defense Acquisition University (DAU) provides the following definition:

> *SUSTAINMENT: Sustainment involves the supportability of fielded systems and their subsequent life cycle product support—from initial procurement to supply chain management (including maintenance) to reutilization and disposal. It includes sustainment functions such as initial provisioning, cataloging, inventory management and warehousing, and depot and field level maintenance. Sustainment begins when any portion of the production quantity has been fielded for operational use. Sustainment includes assessment, execution and oversight of performance based logistics initiatives, including management of performance agreements with force and support providers; oversight of implementation of support systems integration strategies; application of diagnostics, prognostics, and other condition based maintenance techniques; coordination of logistics information technology and other enterprise integration efforts; implementation of logistics footprint reduction strategies; coordination of mission area integration; identification of technology insertion opportunities; identification of operations and support cost reduction opportunities and monitoring of key support metrics* [DAU 2013].

DAU also provides the following context and definition for software sustainment, adapted from our 2006 report:

> *Software maintenance consists of correcting faults, improving performance or other attributes, and adapting to a changing organization and technical environment. To be complete, there is usually a fourth category of maintenance activities focused on anticipated problems, or preventive maintenance.*
>
> *Software sustainment addresses other issues not always an integral part of maintenance such as documentation, operations, deployment, security, configuration management, training (users and sustainment personnel), help desk, COTS [commercial off-the-shelf] product management, and technology refresh. Successful software sustainment consists of more than modifying and updating source code. It also depends on the experience of the sustainment organization, the skills of the sustainment team, the adaptability of the customer, and the operational domain of the team. Thus, software maintenance as well as operations should be considered part of software sustainment* [Lapham 2006, DAU 2011].

[1] http://www.sei.cmu.edu/acquisition/research

1.2 Software Sustainment as a Priority for the Air Force

Decades of experience have shown that the majority of defense system lifecycle costs are incurred after the system has entered operations. Operations and sustainment costs can easily reach 60% to 80% of a weapon system's total lifecycle costs, depending upon the type of system and the duration of its employment [Taylor 2012]. A 2011 National Research Council (NRC) report on aircraft sustainment needs indicated that "…sustainment of weapons system software has the potential to be the largest single growth item in the ALC depot maintenance portfolio" [NRC 2011]. A 2013 NRC workshop report from the Committee on Zero-Sustainment Aircraft for the U.S. Air Force indicated that the Air Force's weapon system sustainment costs were growing by 4% per year, even in the face of downward budget pressure [NRC 2013]. The aircraft fleet is aging, operations tempos are increasing, life extension of airborne platforms is of critical importance, and sequestration has compressed budgets, making it all the more necessary to ensure high-quality, efficient sustainment programs. Software is critically enmeshed in these requirements from supporting automated logistics systems, to the Air Operations Center Weapon System, to replacing and upgrading sensors and systems that provide situational awareness, precision weapons targeting, and more [Air Force 2013].

1.3 How Does Agile Fit In?

Before we describe how Agile fits in, a brief definition of Agile and traditional methods is in order. For purposes of this technical note, we continue to use the Agile definition we introduced in Lapham [Lapham 2010]:

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

By traditional methods, we mean those derived from what is now the engineering "V" model with historic roots in the waterfall method. Essentially the basic concepts are that system requirements are determined up front, system analysis precedes design, system design precedes construction, and system construction precedes deployment [Palmquist 2013].

We have explored the viability of Agile software development methods within DoD programs and barriers to the adoption of those methods for several years. Our experience indicates that software sustainers, more and more, are leveraging Agile methods. On the surface, this seems like a natural fit: Agile methods tend to time-box software delivery using short "sprints" that produce potentially shippable product at the end of each iteration. Bundling these increments into deployable software releases might be a natural fit for adaptive, corrective, and tech refresh maintenance activities on programs and platforms that have already passed critical milestones like preliminary design review (PDR) and critical design review (CDR). We surmised that sustainers might have more flexibility at the local level in tailoring process and acquisition activities so that the battle rhythm of the two could be well aligned for effective, efficient delivery of software using Agile methods. The Acquisition Integration office in the Air Force (SAF/AQX) authorized us to explore the use of Agile in sustainment environments, to identify successes and opportunities to drive further efficiency in software sustainment activities.

As sustainment organizations leverage or transition to Agile methods, they need to think about how the six principles essential to modern project management can help the organization transition to Agile methods:

- "increase return on investment *by making continuous flow of value our focus*
- deliver reliable results *by engaging customers in frequent interactions and shared ownership*
- expect uncertainty and manage for it *through iterations, anticipation and adaptation*
- unleash creativity and innovation *by recognizing that individuals are the ultimate source of value and creating an environment where they can make a difference*
- boost performance *through group accountability for results and shared responsibility for team effectiveness*
- improve effectiveness and reliability *through situationally specific strategies, processes and practices"* [Anderson 2005]

Organizations can apply Agile methods to virtually any piece of work. Frequently people forget that Lean was created to streamline the automotive industry, a predominantly hardware world. Learning to be Agile (or Lean) is not unique to the software development lifecycle. Any task, (e.g., development of a certification and accreditation package) can be dissected into lower level tasks; predecessor and successor relationships can be attached to tasks to find the critical path; priority can be applied to the non-critical path items. All these lower level tasks can be placed in a product backlog, and the work can be planned out in two- to four-week intervals. Our Air Force respondents suggested that Agile methods are currently being applied only to the software development lifecycle or teams that directly impacted software development. Agile methods are broadly applicable across many domains and any functional group on a program, but the organization will need to be open to process innovation [Johnson 2012].

1.4 Research Method

Our research for this report took three avenues: written literature, surveys, and interviews.

First, we engaged in an extensive literature search on the use of Agile methods in software operations, management, and sustainment, both in the DoD and the commercial sector.

Next, we reached out via a variety of mechanisms (including our Agile Collaboration Group and the Air Force Software Improvement Program Working Group) to issue a call for participation in a survey and/or interview. We sought out personnel at both contractors and organic software development organizations who had experience using Agile methods on sustainment programs. We provided both an online survey and contact information so that individuals could provide data anonymously or via teleconference interviews with our research team. (In all cases, interviews were conducted on a non-attribution basis; no identifying information is presented in this report to map any individual or specific program to any response.) Table 1 depicts the summary of characteristics elicited from 28 respondents interviewed or surveyed. Appendix A contains complete details on the characteristics. The surveys and interviews sought information about various aspects such as the contract vehicle, mix of the workforce, support for using Agile methods at the government program office level or within the sustainment organization, use/tailoring of typical Agile methods and techniques. This report characterizes those results. The survey questions are available in Appendix B.

Table 1: Respondent Profiles

Team Size	Type of System	Acquisition Category (ACAT) Level	Type of Sustainment	Contract Vehicle
Ranged from 5 - 10	Safety of life	I	Software maintenance	Single contract
	Weapons systems	II	Contractor support	Multiple-award task order
	Mission capability	III	Corrective maintenance plus minor enhancements for existing system	Task order under existing multiple award
	Don't know	Does not matter	Corrective maintenance, minor enhancements, and major upgrades for existing system	Interdepartmental transfers
	Prefer not to answer			Cost plus fixed fee
				Cost reimbursement (time and materials, T&M)
				Cost plus incentive fee
				Fixed price
				Hybrid firm fixed price (FFP) and T&M
				Organic
				Don't know or prefer not to answer

1.5 DevOps: A New Frontier

An increasing trend in the commercial sector is the use of DevOps groups that blur the lines between software development and operations teams, pushing continuous integration even earlier in a product/system lifecycle. DevOps extends continuous integration and test automation mechanisms to ensure the software development and operation teams interact much earlier, preferably during the design of features. The result is not just continuous integration but also continuous delivery. As part of our exploration, we discuss the roots of DevOps concepts in the commercial world and their potential applicability within the DoD sustainment community. DevOps models appear at this time to be promising potential options for use in IT systems and weapon and logistics support systems. For a complete discussion of DevOps, see Section 4.

1.6 Audience for This Technical Note

The intended audiences for this report are

- software development teams (both government and contract) who are using or contemplating the use of Agile to execute software sustainment on a DoD program
- members of DoD program offices who may be challenged to undertake software sustainment efforts with a developer who will be using Agile
- senior DoD acquisition decision and policymakers, to advise them on the practicality and viability of encouraging the employment of Agile in software sustainment strategies.

1.7 Organization of This Technical Note

Section 2 of this paper provides a discussion of Agile in DoD sustainment, including why Agile implementations should be considered, trends with the use of Agile in DoD, and optimizing success and minimizing failure.

Section 3 discusses Agile software sustainment efforts in the Air Force using the Scrum method constructs to illustrate and discuss the information discovered during our interviews, survey, and research.

Section 4 describes DevOps. This is a continuous integration and deployment idea that is taking root in the commercial space. This could be one way to reduce the software delivery cycle time.

Section 5 provides conclusions from our research and exploration of Agile and software sustainment.

Appendix A provides respondent characteristics.

Appendix B contains a copy of the survey questions.

Appendix C provides a list of SEI published Agile papers.

Appendix D contains a reading list of various Agile-related topics gathered from the Scrum Alliance, the Program Management Institute, and the Lean System Society.

2 Agile in DoD Sustainment

Agile is a philosophy that is expressed in the Agile Manifesto and its 12 related principles. The manifesto and principles were created in February 2001 by 17 leaders and consultants in software development who would normally have been competitors. The self-named Agile Alliance shared allegiance to a set of compatible values promoting organizational models based on people, collaboration, and building organizational communities compatible with their vision and principles [Agile 2001]. [2]

Prior to and concurrently with the creation of the Agile Manifesto, numerous software development processes formed or were forming that were Agile in their characters (e.g., Evolutionary Project Management [EVO], Competitive Engineering, Rational Unified Process, Scrum, Crystal Clear, Extreme Programming [XP], Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method [DSDM)]). Arguably, Agile methods (incremental development) were used as early as 1957.[3] Since then, some of the methods have become more commonly used and tend to dominate the Agile implementation space. According to the *8th Annual Survey on the State of Agile* by VersionOne, Scrum followed by a Scrum/XP Hybrid are the two most common. See Figure 1, which was created using the survey data.

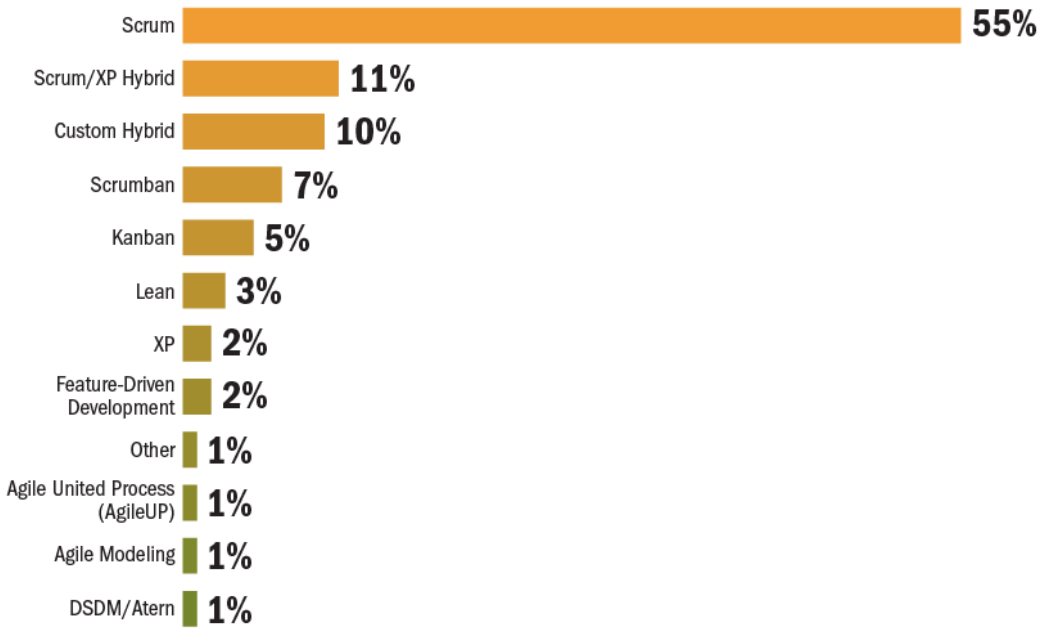


Figure 1: Most Common Agile Methods [VersionOne 2014]

[2] http://agilemanifesto.org/history.html

[3] Several references can be found for this 1957 date: http://www.stephenblower.co.uk/blog/26/04/2013/a-brief-history-software-development/ and "Agile Processes in Software Engineering and EXtreme Programming:" 9th International Conference, XP 2008, Limerick, Ireland, June 10-14, 2008 (Proceedings, Google eBook).

However, upon close inspection, many implementations in execution appear to be hybrids. The hybrids tend to be a mix of selected Agile practices, sometimes from multiple methods (such as Scrum and XP), with some practices from the traditional development world. There is even an instance where such a hybrid has a specific name, Water-Scrum-Fall.[4]

The authors believe hybrids are appearing at a much higher incidence due to the complexity of the environments in which programs operate, the regulations with which many environments are mandated to be compliant, and the inconsistent guidance on how best to organize for competitiveness within those environments. The implementations are defined by their particular hybridizations (adaptations) and they are tested within their environments for efficacy and agility. Many of our interviewees told us that they picked the Agile practices that enhanced their current methods to improve their processes. Few, if any, strictly adhered to one particular Agile method.

2.1 So Why Use Agile Implementations at All?

The environments (laws, regulations, directives, stakeholders, cost/schedule/performance pressures, knowledge gaps, emerging techniques, human capital turnover, classification requirements, etc.) in which we develop software are becoming more complex. Without a significant breakthrough, one may assume this trend will continue. In addition, the difference in the tempo of need (the tempo of the warfighter) and the tempo of provision (tempo of the developer and acquirer) needs to be addressed. As shown in Figure 2, the different tempos for development, acquisition/readiness, and operations/demand are shown side by side over a similar time scale. This depiction shows the amount of work (the bigger the spiral, the more work done) that can be typically accomplished as opposed to the need or tempo that operations require [Lapham 2011]. Therefore, change is the one constant we should expect, and agility as a response to that constant is vital to success when change and uncertainty dominate the environment.

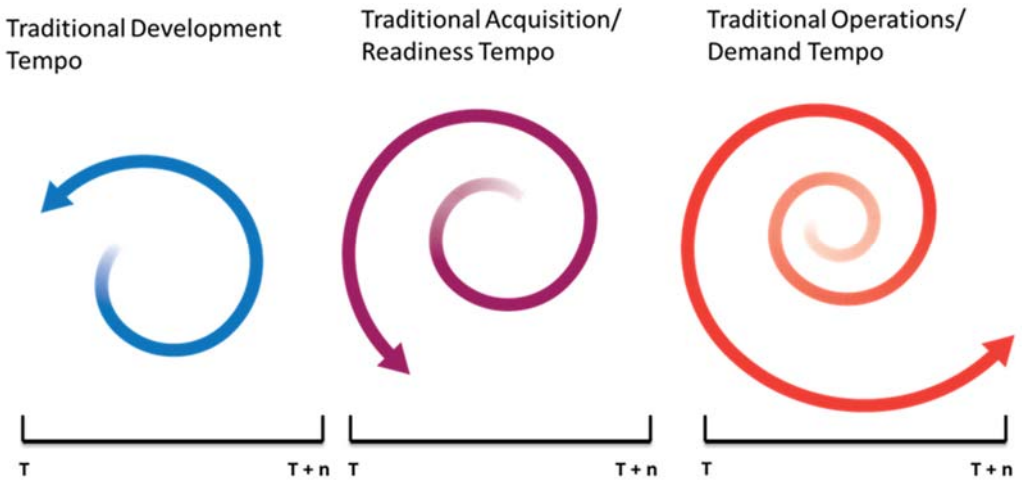


Figure 2: The Disconnect Among Warfighter and Acquisition Tempos [Boxer 2009]

[4] "A flex ble approach that embraces both traditional and Agile development principles allows development teams to use whatever practices and techniques best meet the needs of the problem being solved. Many organizations use Agile principles and Scrum communication techniques in their day-to-day product development but employ traditional waterfall methodologies for planning, budgeting or documenting the project's progress." http://whatis.techtarget.com/definition/WaterScrumFall-water-Scrum-fall

One could rather confidently say that business environments (civil and government) have changed so fundamentally with respect to competition that all that is left to change is one's ability to be agile: to adapt, to learn, and to make uniformly correct decisions at a rapid rate within those environments. One important character trait of agility is speed—speed of learning, speed of thought, speed of decision. Also within the character of agility is the correctness of action or the correctness of the decision based upon rapid learning within highly dynamic and often uncertain environments. Agile implementations for the development of software (and other employments) may be viewed as an adaptive option for gaining and maintaining competitive advantage for organizations implementing it and for the stakeholders seeking competitive advantage. Agile may be the breakthrough that succeeds in niches that traditional software development methodologies have been hard pressed to successfully fill.

2.2 Trends in the Use of Agile Principles in DoD

Agile implementations, as one interviewee within this study suggests, are "not a passing fad." Let's assume this interviewee is correct. Agile in its numerous forms is here to stay and the sooner we develop an understanding of where and when it is most optimally used, the sooner we will begin to benefit greatly from its practice. As important, we must understand when and where it may not be advisable to extensively employ Agile principles in order to limit the waste of precious resources while attempting an ill-advised implementation. While Agile has been evolving over a few decades, the time for its widespread use has come because, as suggested earlier, the environment is practically demanding it.

Trends discovered in the course of writing this technical note indicate the use of Agile principles is high among younger respondents. We attribute this to the fact that current curricula within higher institutions of learning provide a well-developed discussion and exercise on the use of Agile. Alternatively, a trend of overt skepticism is reported to exist among those trained and grounded in the more traditional methods of software development. This finding is not unexpected and it will no doubt contribute to an unnecessarily long implementation and acceptance phase we expect Agile to go through before being so well understood as to become a new traditional way of doing business.[5] The time will come, however, because the benefits can be significant.

Most respondents reported significant benefits of using Agile-"like" (hybrid) implementations including substantially lower costs, early deployment, and transparency of action. Another benefit is the reported increase in trust among users when correct implementations are manifested within the contract and client organizations (to include stakeholders). However, the most significant benefit of implementing Agile mentioned by the majority of respondents was the reduced time to market or how much faster the product made it into the hands of the client (warfighter). These benefits are profound and they reflect those found in the civilian sector, which has mastered in large part Agile implementations. But drawbacks and setbacks are not uncommon.

Cultural shifts in thinking were reported to be vital to the success of an Agile implementation. And while training personnel is highly valued, the availability of appropriate resources or investments in training appears to be less than optimal for all stakeholders. In fact, on-the-job training

[5] See Lapham Appendix F for a discussion on organizational change and the Satir management model [Lapham 2011].

(OJT) appears to be the norm across numerous organizations. Using OJT almost certainly yields a less than optimal result in the management and leadership of Agile efforts. A number of respondents reported a significant lack of training, which results in a "learning as you go" mentality. This OJT is common across the stakeholders (contractors, DoD, and other stakeholders). A lack of understanding most often results in misallocation of resources, delays, cost overruns, and scope creep by stakeholders, all of which eat away at the efficiencies gained through the use of Agile implementations. Numerous respondents reported the efficiencies they gained were consumed and ultimately outpaced by the client demanding new features be added, over and over again. The ability to respond to this demand is a value that Agile contributes to the sustainment arena, but if it occurs without appropriate controls, it's not as effective for either developers or customers.

Another problem is culture clash. When the contractor is well versed in Agile but the client is not, numerous issues may arise. One issue may be characterized as having a contractor who is doing well from an Agile perspective but the government client is not trained as a product owner and government personnel have little to no knowledge of how to provide adequate oversight to an Agile implementation. Communication is limited and often confused because the client focuses on incongruent and/or meaningless measures when compared to the Agile implementation, which in turn causes the contractor to spend time and resources to answer inconsequential questions or requests. Another drawback (or benefit depending upon one's perspective) is the case where a contractor is "forced" to implement Agile practices in "stealth mode" as a risk mitigation strategy. Only when the practices have proven utility is the contractor able to inform the client about how it executed. One respondent reported participation in a "stealth" implementation; the team's success benefited them and the client was very pleased with the results. This approach is risky and we would not recommend its use because it violates one of the principal tenets of Agile—transparency. So what can be done to maximize opportunities for success and minimize the pitfalls of an Agile implementation?

2.3 Optimizing Success and Minimizing Failure

Understanding the sustainment programs in which an Agile implementation will bring success is an important launching point. Not all programs are suited for Agile implementations. How to make this determination, however, is beyond the scope of this paper.[6] After the determination has been made, a successful Agile implementation begins with buy-in. All organizational change of this magnitude requires organizational and individual buy-in to be successful. We understand that no institution that has long endured will be able to easily accept such a profound change without a degree of upheaval or resistance. The people of an organization are the most important resource the organization has. Therefore, investing in people must be the first step to arriving at a unified Agile implementation. Training and certification[7] of people throughout the organization is imperative to an effective Agile implementation. Once people are trained, their training should be reinforced continually within the organization through mentoring and coaching. Following the institutional adoption of Agile and the training of people for particular roles within Agile, the next

[6] See *Readiness and Fit Analysis* (October 8, 2012) for information on a model for understanding adoption risks related to Agile implementation (http://blog.sei.cmu.edu/archives.cfm/author/suzanne-miller).

[7] Several Agile training courses offer certification, such as ScrumMaster and certified Agile tester; even PMI has an Agile PM certification.

step that should never be overlooked is organizing for success. Moving toward adoption of Agile is not an easy undertaking. Paul Adler, a well-known researcher in the field of organizational behavior, has postulated a continuum of change that predicts how difficult it will be for an organization to change its practices. The least impacting change tends to be a skills change, followed by procedural change, structural change, strategy change, and the most difficult and time-consuming, culture change. There are many factors that impact adoption including reward system, sponsorship, values, skills, structure, and history, to name a few. There are also other concepts such as determining where you are in the cycle for change and where your team is in the adoption process. There is an entire body of knowledge in each of these areas that can be used to help with Agile adoption. For more information on moving toward adoption of Agile, see Lapham and colleagues' *Agile Methods* [Lapham 2011, Section 6].

Agile implementations succeed and fail rarely on the technical abilities of the teams but instead based upon the organizational design and—by extension—how they execute their Agile implementation.[8] When people are "dual hatted" in roles and responsibilities, confusion may ensue. We believe one of the most important roles in the success of an Agile implementation in sustainment is that of the product owner. In commercial industry, the role of the product owner is well understood. But what about in the context of DoD sustainment programs?

When one moves into the domain of the DoD, the product owner role becomes problematic for a number of reasons. First, DoD personnel appear to not understand what a product owner is since there is no analogous role in traditional acquisition processes; thus they have no frame of reference for the importance of this role in the success of Agile efforts. Second, DoD personnel often falsely assume they do not have the latitude to assign a person to this full-time position working shoulder to shoulder with the contractor. Some may be reluctant to work shoulder to shoulder with the contractor, as they fear constructive change. We have learned from our interviews and research that it is possible to have this working relationship without creating constructive change in programs [Lapham 2010]. Third, due to military and civilian personnel transfers, DoD program offices may not have access to an individual with the technical depth and training to perform in this vitally important role. Planning for and overcoming these biases and perceived obstacles will prove to be an important act that the DoD can take in partnering with industry for successful Agile implementations.

A potential alternative based on the authors' observations would be for the DoD to hire a person with the technical depth and the proper Agile training to be an advisor or coach to the program office and possibly act on the government's behalf. This advisor or coach, who could be called an independent product owner agent (IPOA), may be an acceptable approach for DoD when implementing Agile. Often the misunderstandings and misconceptions of DoD personnel lead to serious problems in communication with the contractor and in attempts to understand the current state of the program. Having an IPOA hired by the government would permit the government to have an independent agent with the technical depth to perform the work as a product owner peer, shoulder to shoulder with the contractor-assigned product owner. The IPOA could then bridge the gap between DoD and the contractor and help both organizations understand each other's concerns and issues. While the IPOA would be clearly ingrained into the contract organization in day-to-day

[8] Scaling Agile: An Executive Guide, Scott Ambler, IBM, February 2010.

operations, he or she also would help create the mandated acquisition documentation required by DoD agencies. No one would be more attuned to the inner workings and progress of the contractor than the IPOA. In this regard, this role for the DoD could be second only to the program manager.

All roles in Agile implementations are important. Clearly defining and staffing each role and then training personnel within each role is important to success; all roles should be reinforced through the ample use of polyskilling,[9] as suggested by one of the interviewees. The contractors make strong use of polyskilling but DoD is not as adept at it. The DoD is often understaffed and overwhelmed so just committing the time and expense it would take to develop a strong polyskilling capability would be enormous. For example, the contracting officers by law must be warranted. They will probably not cross train as financial managers. However, they can be trained on what it means to operate a technical program in an Agile way so that they can think about contracting from an appropriate perspective.

For assignments to acquisition roles in which Agile implementations are used, personnel should be trained and certified prior to arrival at their new commands. A well-developed Agile implementation indoctrination should then be conducted to orient them on the intricacies of the particular implementation within which they will be working.

[9] Some including Scott Ambler would call this a generalizing specialist, someone who has one or more technical specialties; has at least a general knowledge of software development; has at least a general knowledge of the business domain in which they work; and actively seeks to gain new skills in both their existing specialties as well as in other areas, including both technical and domain areas. (http://www.agilemodeling.com/essays/generalizingSpecialists.htm)

3 Agile Software Sustainment Efforts in the Air Force

Given the definition of sustainment in Section 1.1, software sustainment addresses other issues that are not always an integral part of maintenance—such as documentation, operations, deployment, security, configuration management, training (users and sustainment personnel), help desk, COTS product management, and technology refresh. Successful software sustainment consists of more than modifying and updating source code. Successful software sustainment also depends on the experience of the sustainment organization, the skills of the sustainment team, the adaptability of the customer, and the operational domain of the team. Thus, software maintenance as well as operations should be considered part of software sustainment [Lapham 2006].

This section of our technical note provides insight into the state of Agile practices in the Air Force sustainment community. First, we explore the types of software sustainment organizations within the Air Force; then we walk through a common Agile model employed by sustainers and document their efforts and challenges in using this model. The note finishes with a discussion of how the software sustainment efforts interact with the larger system (e.g., aircraft, ship, satellite, weapons system, and the like).

3.1 Types of Software Sustainment Organizations Within the Air Force

There are two types of sustainment organizations employed within the Air Force. They are organic or contractor based. In some instances, a mix of the two is employed.

3.1.1 Organic Sustainment Organizations

The Scrum approach implemented by organic sustainment teams was distinctly different from the Scrum approach implemented by a contractor sustainment team. The organic sustainment teams interviewed, ranging in size from seven to 50 developers, appeared to have separate teams of system engineering, software development, and test and evaluation (T&E) that operated using Scrum methods to

- manage functional team work in process (WIP)
- use Scrum-of-Scrums to determine handoff points to another functional team or determine when hardening sprints are needed for a specific functional team
- use collaboration meetings between teams to determine definition of done and time frame when work products are needed

The net effect for one organization was that a Scrum/Team Software Process (TSP)/Personal Software Process (PSP) work environment was able to operate using a Scrum-waterfall (also known as Water-Scrum-Fall)[10] method. Another organization found applying the Scrum standup meetings ensured the team understood its velocity, and the self-directed team could apply the corrective actions needed to help team members who were struggling to meet the agreed-to velocity.

[10] http://whatis.techtarget.com/definition/WaterScrumFall-water-Scrum-fall

3.1.2 Contractor Sustainment Organizations

Contractor organizations we interviewed work to allocate system architects, system engineers, software developers, and testers to every Scrum team; they worked to keep the team size to 10 or fewer. The contractor Scrum organizations applied this multidisciplinary team across subsystems being sustained. The contractor organizations used Scrum-of-Scrum meetings to determine how to integrate the work products to

- manage an enterprise-wide functional team WIP
- determine handoff points from a Scrum team to another non-Agile functional team (i.e., specialty engineering teams, security, or system integration teams)
- determine trigger points for hardening sprints[11] so one non-agile functional team was not impacted by another non-Agile functional team (e.g., a security team impacting an integration team)—While this is not a "pure" Agile environment, this type of activity is more common than not and should be considered a hybrid situation.
- determine definition of done and time frame when work products are needed

3.2 Agile Applied by Air Force Sustainers

Scrum is the most commonly used Agile method. As shown in Figure 3 it has minimal roles, artifacts, and ceremonies. The roles consist of a product owner, ScrumMaster, and the team (not shown on diagram). There are three artifacts: product backlog, sprint backlog, and burndown charts (not shown in diagram). The ceremonies consist of

- release planning
- sprint planning
- sprint review/demonstrations
- sprint retrospective
- daily Scrum meeting

We adapted the original figure from Mountain Goat Software to include the first four ceremonies of the above list. More information on Scrum can be found from various sources including the Scrum Alliance and books and publications by Mike Cohn and others. See Appendix D for a suggested reading list.

[11] "[H]ardening activities provide time for final product validation, system and performance testing, standards and security validations, user acceptance testing, and any other Release Readiness activities which are not feas ble or economical every sprint." From Scaled Agile Framework, http://scaledagileframework.com/hardening-innovation-planning/ (2013).

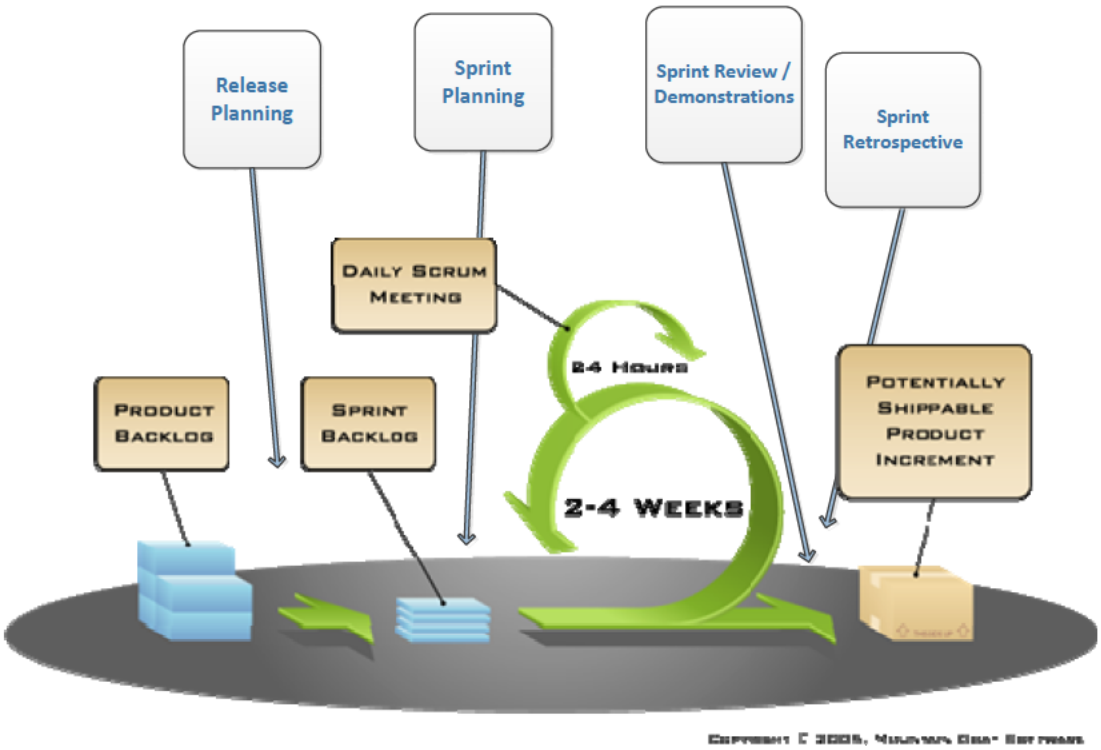
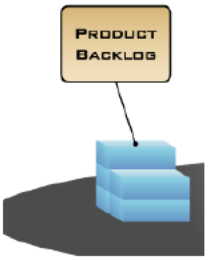


Figure 3: Key Elements of Scrum (reproduced with permission from Mountain Goat Software and Mike Cohn; adaptations added in unshaded boxes) [Mountain Goat Software 2005]

Every organization we interviewed or surveyed uses some tailored variant that fits into the commonly accepted Scrum model in Figure 3.[12]

The remainder of this section will use the overall constructs of Scrum to illustrate and discuss the information discovered during our interviews, survey, and research. As we discuss the different pieces of the Scrum model, we will show part of Figure 3 related to the discussion.

3.2.1 Sustainment Teams and Product Backlogs



Seventy-five percent of the sustainment organizations we spoke with appear to enter the lifecycle when the product backlog is approved and funded. The sustainment strategy (e.g., contract, organic, federally funded research and development center [FFRDC], or other organizations and combinations) has already been identified and at least some of the adaptive, corrective, perfective, and technology refresh tasks and priorities have been developed, typically without any input from the development teams. This separation from the development teams represents a departure from typical Scrum implementation where the development team is engaged in creating the backlog.

[12] This illustration is part of a presentation that is licensed under a Creative Commons Attribution 3.0 United States License. The source is acknowledged as Mountain Goat Software and Mike Cohn. http://www.mountaingoatsoftware.com/agile/scrum/a-reusable-scrum-presentation. Additional square unshaded boxes have been added to show more ceremonies.

A backlog, at its most basic, is “a list which the team maintains of features or technical tasks which, at a given moment, are known to be necessary and sufficient to complete a project or a release” [Agile Alliance 2013]. It is expected that as product users, owners, and developers learn more about the product and what features are necessary and sufficient to meet the need that the backlog will evolve over time. The backlog “is the primary point of entry for knowledge about requirements, and the single authoritative source defining the work to be done” [Agile Alliance 2013]. A backlog is not required to be stored or preserved in a specific format.

Backlog, though, is not a term that has been typically associated with the wants and needs of defense acquisition programs. Users and acquirers may not use typical Agile terminology such as “backlog” in their day-to-day lexicon. While a backlog should not be confused with a requirements specification (e.g., a software requirements specification), requirements and defect lists, databases, matrices, spreadsheets, and the like, all contain elements of the backlog for the product or system. The backlog must ultimately be a single source that can contain prioritized elements that are vetted as part of the vision for the overall product, sprint, or release. When working with acquirers or users, sustainers need to look for the tools or artifacts that fill the purpose of the backlog, whatever they may be called. For example, we are familiar with one organization that uses a requirements traceability matrix (RTM) that serves as the backlog. All items included in the RTM have been vetted by the stakeholders.

In the remainder of this section, we discuss the different types of backlogs—product (roadmap), release, and sprint (or iteration) shown in Figure 3. The product backlog contains all the requirements for the product in question. These are “broken up” and assigned to multiple releases starting with the highest priority requirements going to the first release. In turn, the release requirements are broken up into iterations or sprints. As shown in Figure 4, multiple iterations (sprints) comprise a release, and multiple releases comprise the product (roadmap). Icons from this figure will be used in the following sections discussing backlogs.

Our respondents described the development, contents, and ownership or management of backlogs, regardless of the day-to-day terminology used by the acquirers, users, or sustainers themselves. Their responses are summarized within the discussion.

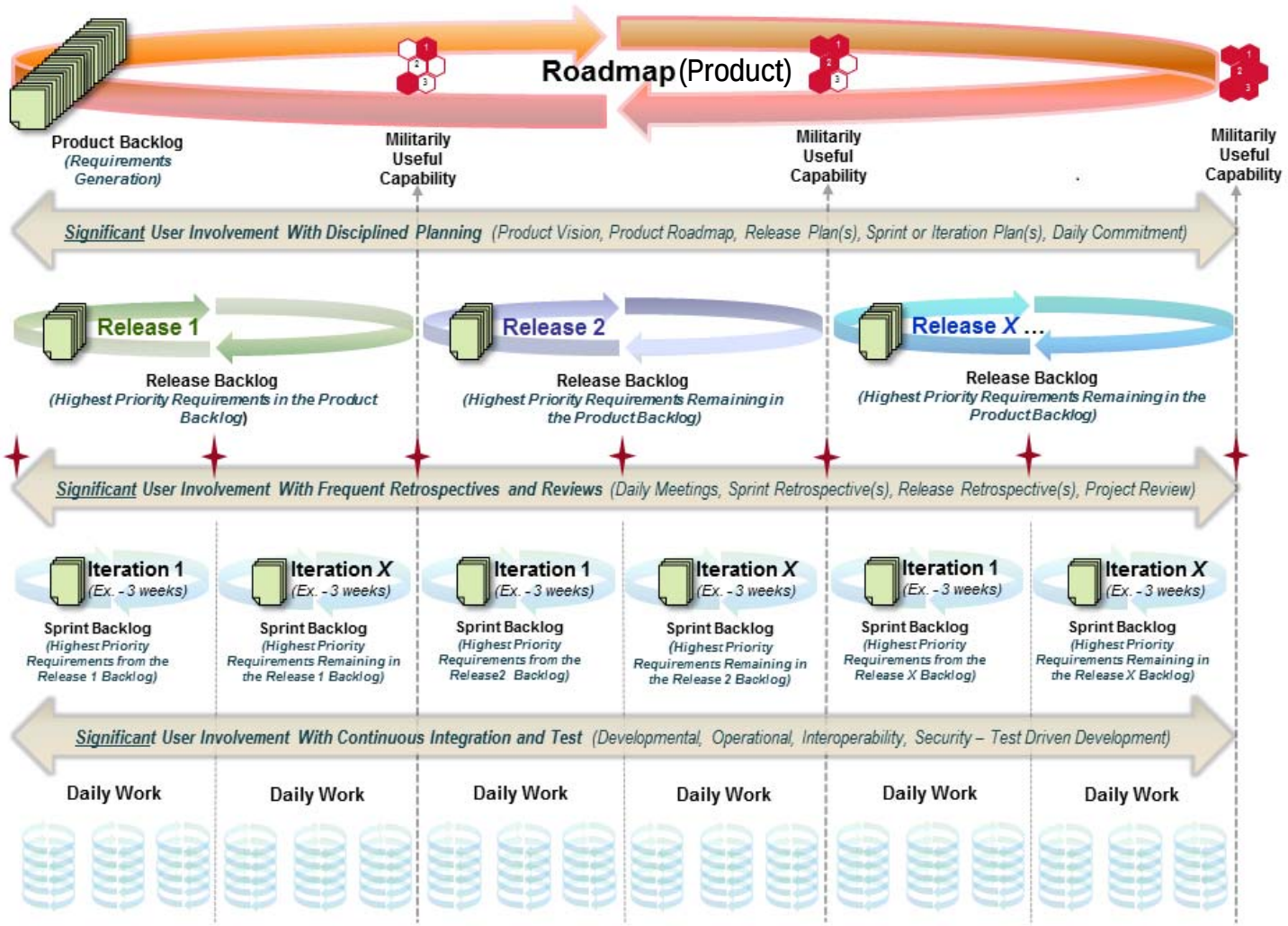
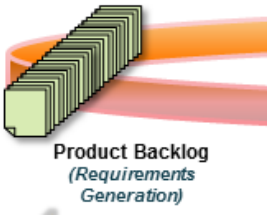


Figure 4: Types of Backlogs [Palmquist 2013]

3.2.1.1 Product Backlog



The term "product backlog" refers to the backlog for the overall product or system (versus the backlog for a particular development iteration or a particular software release).

In many organizations the acquiring and using commands determine the features, functions, defects, or content for the product backlog. The acquiring and using commands are the actual product owners (contrasted with the sustainment personnel who may act as surrogates or proxies for them) because they own the budget and schedule and have the actual need. These actual product owners will often delegate the technical work related to reviewing an impact analysis to the sustainment organization and to program managers within the sustainment program office.

Each respondent recognized there was a product backlog, where the changes to the system product are defined—these changes ranged from defects or technical debt[15] to minor enhancements or major upgrades. The program office and end user determine the priority of the need. The program office works with the sustainment organization to size the selected changes for system changes, which are planned 12, 18, or 24 months into the future. When the sustainment organization is engaged, the technical discussion often centers on the impact assessment completed to build the initial assessment and, where possible, any known dependencies for the change. In some organizations this product backlog is reviewed annually or even semi-annually based on the frequency of user-group meetings with the program office. Most respondents indicated that the release duration is long: 12- to 27-month durations are common. Milestones like system/software specification review (SSR) and test readiness review (TRR) are still planned into the schedule.

Both the organic and contractor teams receive a product backlog of sorts. Some organic teams we spoke with refer to the product backlog as a "list of candidates" but the data is essentially a product backlog. In the case of contract sustainment efforts, both organic and contractor sustainment organizations develop a rough order of magnitude (ROM) or provide definitization and costing information needed to get the work on contract. This negotiation process appears to occur semi-annually in some organizations, or at least as an annual process. Some organizations have user-group meetings with the using command, and the acquiring command will use this user-group meeting as an opportunity to complete a priority check on features, functions, or defects in the product backlog or to check affinities between new features or functions, prioritize the current defects, or determine where software changes will occur to reduce the number of times a piece of software is handled per release.

As noted above, most sustainers we interviewed indicated that by the time they were engaged in the process, the product backlog had already been established. Some organic organizations had more success in influencing this process with discussion of technical debt and bundles of related features and functionality.

---

[15] The technical debt metaphor was coined by Ward Cunningham in a 1992 Object-Oriented Programming, Systems, Languages and Applications (OOPSLA) experience report. (http://c2.com/doc/oopsla92.html)

Many sustaining organizations noted the product backlog provided by the acquiring command needed additional grooming—it may provide the priority of the feature or target a need date, but the technical teams need to complete the impact analysis or refactoring to develop the ROM for the proposed change. The individual items in the product backlog may or may not all be funded and approved at the time the backlog is developed.

3.2.1.2 Release Backlog

The release backlog is a step into further granularity of product definition.

The release backlogs are used to create the list of items to be potentially released in the next version of the product and as the source for sprint planning and the resulting sprint backlogs. The release backlog typically has detailed planning for the current and next sprint; intermittent planning may start for critical features with longer planning three or more sprints out from the current sprint. Our interview respondents reported using a variety of techniques to develop estimates for the effort required to execute backlog items, which were typically variants on "Planning Poker."[16] The estimating process varies due to the diversity of the organizations or level of adoption by organizations interviewed. The list below summarizes the nuances described about the release estimating processes:

- Some of the respondents still operate with multiple functional teams—system engineering, software development, test and evaluation, and operations—and each team relies upon a system software specification requirement document or change pages to a stable document. Once the impact analysis is complete, the change includes size and cost, and the Scrum teams are released to start the work.
- On many of the respondent teams, the system engineers need to complete their work before the Scrum team will see a change enter the release backlog. The Scrum teams will develop their detailed plans for implementation in two to three sprints using a time box pattern. The volume of content is based on average team velocity, often depicted by a number of story points.[17]
- One respondent noted that the team set up their story points as a "closed loop" function so that at the end of a release the end user received all the targeted functionality. In another organization they use rolling wave planning to plan a 12-month time box, typically based upon the fiscal year. This organization integrated rolling wave planning as described in the Scaled Agile Framework Principles for Lean/Agile Leaders (SAFe),[18] in which the whole team

---

16 Planning Poker® is a consensus-based estimating technique. Agile teams around the world use Planning Poker to estimate their product backlogs. Planning Poker can be used with story points, ideal days, or any other estimating unit. Planning Poker is a registered trademark of Mountain Goat Software, LLC. http://www.mountaingoatsoftware.com/agile/planning-poker

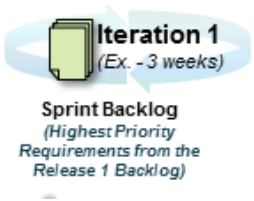
17 "Story points are a unit of measure for expressing the overall size of a user story, feature, or the piece of work ...The number of story points associated with a story represents the overall size of the story. There is no set formula for defining the size of a story. Rather a story-point estimate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

18 http://scaledagileframework.com/

works to build the release plan from an economic viewpoint focusing on the cost of delay.[19] The detailed planning appeared to use story point sticky notes to determine the change affinity of the list of candidates within the rolling wave plan where necessary. The team reviews the user stories, sizes the user stories using a Fibonacci sequence for relative sizing, and calibrates the sprint content to the team velocity, where the team skill at delivering story points is captured as the team's average story points completed per sprint.

- One organic sustainment organization doesn't use Planning Poker but relies on a candidate planning process based on past performance history and team velocity measures that evolved over the years. This candidate planning process is seen as providing a realistic cost and schedule, and the interviewees do use Agile planning steps to work with the customer to size and ROM the candidates, such as identifying the size (small-medium-large), the volume of changes, the complexity factor, and the number of change pages. However, they do not perform a check that the candidate fits into the customer's budget. This sustainment organization's candidate planning process is usually within 5% of the total cost in the customer's budget.

3.2.1.3 Sprint/Iteration Backlog



Both the organic and contractor sustainment organizations pick items or candidates from the release backlog and allocate them to a sprint. It is important to note that sustainers—both organic and contract—can only execute on backlog items that have been approved via the contract or memorandum of agreement/memorandum of understanding (MOA/MOU) in place with the acquiring organization. If the entire release backlog has already been funded, then the process of allocating these items into various sprints can be executed with little or minimal guidance and involvement from the acquirer. If, however, the sustainer is recommending the inclusion of nonfunded tasks (e.g., defect correction to address technical debt), then negotiation must occur between the sustainer and the acquirer to approve the work at hand and modify any relevant agreements accordingly. Generally, once an item is placed into the sprint backlog, it will not be removed without this type of negotiation.

Both organic and contractor sustainment organizations groom the sprint backlog so that the content for the current sprint and the next one or two sprints is understood. The sprint backlog grooming, or refinement, is completed using multidisciplinary teams, so any organization that maintains silos based on functional discipline needs to collaborate with the development team. All organizations we interviewed clearly see the need for multidisciplinary team collaboration, but this will not always mean the Agile team is a multidisciplinary team.

[19] "If you only quantify one thing, quantify the cost of delay." – Don Reinertsen, Economic Prioritization with Weighted Shortest Job First (WSJF) Abstract. The SAFe Framework is intended for application in situations where a number of teams are engaged in ongoing, continuous development—a flow. http://scaledagileframework.com/?s=reinertsen

3.2.2 Sizing Backlogs

Our contractor respondents often see the release planning completed by the program office and end using commands with limited input from the contractor. The contractor needed to provide ROM estimates or tee-shirt sizing (i.e., small, medium, large, and extra-large) where these sizes had an agreed-upon cost and time range. Conversely, organic organization respondents are actively engaged in determining release content and use a known sizing and price model that has typically been very accurate in the past.

These sustainment organizations typically use variants of Planning Poker to estimate the feature user stories, then determine which feature user stories are allocated to a sprint based on priority and total sprint story points. Some organizations indicated that they will use small-medium-large to define user story ballpark size for the release, and then during the sprint backlog grooming they work to apply more finite planning with Planning Poker.

Identifying a candidate as an epic[20] and recognizing the need to split it into multiple user stories can occur when grooming the product backlog, release backlog, or the sprint backlog.

3.2.3 Sprint Cycles

The interviewees indicated they worked in two- to five-week iteration cycles. Both the organic and contractor sustainers described this process as a standard implementation of an Agile sprint cycle. It seemed the only issue with performing in this manner was focused on the time duration and its relationship to earned value (EV). Most acquiring organizations appeared to want the sprint duration to align with the month so the EV metrics would correlate with planned versus actuals in a sprint. This is due to the fact that at the end of the sprint, there was a finished product that could be considered 100% done and thus earns the value assigned to that piece of the work.

3.2.4 Scrum Meetings/Daily Standups

Daily standups typically occur at the same time on each day and last no more than 15 minutes. Not all interviewees strictly followed this daily standup meeting rule of engagement. The daily standup (if held) lasted for 5, 15, or 30 minutes based on the size of the team. In all cases, they used the standard questions to frame the standup meeting:

1. What did I accomplish yesterday (or since the last meeting)?
2. What will I do today (or by the next meeting)?

---

[20] "Epic" is just a label we apply to a large story. Calling a story an epic can sometimes convey additional meaning. Suppose you ask me if I had time yesterday to write the user stories about the monthly reporting part of the system. "Yes," I reply, "but they are mostly epics." That tells you that while I did write them, I didn't get the chance to break most of them down into stories that are probably small enough to implement directly. http://www.mountaingoatsoftware.com/blog/stories-epics-and-themes

3. What obstacles are impeding my progress?

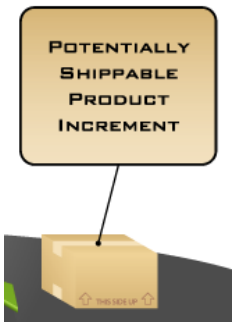
These Scrum daily standups are used as a communication tool. Impediments are identified to the ScrumMaster in order for action to be taken to eliminate the issue. If necessary, depending on the size of the program, these impediments will be escalated to the Scrum-of-Scrum meeting for resolution.

3.2.5 Scrum-of-Scrum Meetings

The size of the programs interviewed affected whether there was a need for a Scrum-of-Scrum standup meeting, so the intent here is to show that larger programs implement a Scrum-of-Scrum standup meeting. The significant change is the number of people at the meeting will often exceed 10; the members are typically integrated project team (IPT) leads or functional leads. The time allocation per person is still only a few minutes, and they change the word "I" and "my" in the standard questions to "the team." The meeting duration will likely exceed 30 minutes but will not exceed one hour, and it is not often a daily meeting.

This Scrum-of-Scrum meeting is also a communication tool. The ScrumMaster, IPT lead, or functional group lead will collect the impediments at the team level, then move to the Scrum-of-Scrum meeting and report the impediment. Frequently before the Scrum-of-Scrum meeting, these middle management or front-line managers will arrange a collaboration meeting with each other to discuss the impediments and determine what is causing the impediment or teams impacted by the impediment. These meetings are intended to stimulate conversation and proactive courses of action that can be weighed and picked as a path forward. These ScrumMasters, IPT leads, or functional group leads are front-line or middle management and often technical people promoted into this position. This level of the workforce is best suited to understand the low-level technical challenges and constraints; to build the reasonable courses of action for a path forward; and to see where process improvements or innovations are needed to remove process-related roadblocks.

3.2.6 Potentially Shippable Product Increments



Release cycles ranged from quarterly, semi-annually, every nine months, annually, to every 27 months. Release cycles and the release content were often predetermined by the acquiring command and using command. However, not all sustainment personnel were represented at these collaboration meetings. Additional collaboration meetings with the sustaining organization did occur to ensure the sustaining organization understood the product backlog that was pre-decided. When the sustaining organization completed an impact analysis or if refactoring the analysis identified other functions, features, or defects that could be worked with the change request, the acquiring command or the using command would make the determination about whether these additional changes could be added to the release. This approach tends to add work, which might be avoided by involving the sustaining organizations in the original meetings for the non-budgetary discussions.

Universally, all organizations interviewed agreed that they build the release plan to meet a set of needs or requirements and this is a shippable product once testing demonstrates that the level of quality needed has been achieved. Each sprint the interviewees gauge if the product is shippable

based on what from the product backlog is done and what is still being worked. Programs with long release cycles (12, 24, or 27 months) might ask the following questions:

- Is there a shippable product sooner?
- Can the acquirer reduce the time box of a release?
- What kinds of impediments need to be removed?

Many of the organizations interviewed can produce an emergency fix within the time allocated on their contract or agreement. However, the pace to do emergency fixes was just what they were called—emergency—which means faster than the usual cycle time and thus not sustainable.

3.2.7 Sprint Review/Demonstrations

Both the organic teams and contractors attend user-group meetings because this is an opportunity to solicit the "voice from the field." These events offer an opportunity for the contractor sustainment organizations to demonstrate features or functions to the subject-matter experts (SMEs) or the using command to ensure the changes completed are rendering the expected results.

The contractors will frequently solicit input from their SMEs by releasing, at the end of a sprint or release depending on the negotiated cadence, a demonstration for their environment to ensure the feature, function, or defect correction is operating as expected. In either case, when the SME or using command provides clarifications, the sustainment organizations add the changes or rework to the sprint backlog. In contrast, the organic sustainment organizations may demonstrate models, but they do not do development demonstrations per se—the system engineering models are usually adequate to solicit input from the customer to refine or clarify misunderstandings about a candidate (feature, function, or capability on the product backlog).

3.2.8 Retrospectives

Many of our respondents used the sprint retrospectives to decompress, where teams meet for a working lunch to perform the retrospective. As the team settled into their seats with food the Scrum Master or Agile coach would ask the team to use the provided sticky notes and markers to record the lessons learned about the following:

- What went well?
- What went poorly?
- What needs improvement?

Then the Agile coach/ScrumMaster aggregated data into categories, gained concurrence, and worked to identify process improvements as needed. Some Agile coaches and ScrumMasters used the results of the retrospective to not only update the day-to-day working processes but to address the overall organizational charter and culture for the program and organization.

Organizations often built a charter to define their effectiveness and reliability goals relying on higher level strategic goals, corporate processes, and practices—this charter needs to be refreshed intermittently to reflect changes in higher level strategic goals. This charter refresh may trigger changes in processes and practices about metrics collected or methods used by the team to increase product efficiency.

3.2.8.1 Sprint Retrospectives

One organic organization uses "sprint post-mortem" across the hybrid organizations including system engineering, software development, integration, test/evaluation, and technical order teams. Thus all teams participate as a multidisciplinary team. The lessons learned are put into a historical database to see how to improve future estimates. For this organization the primary objective for both release and sprint retrospective is to check cost and schedule estimates in order to improve reporting of EV in an Agile setting. This organization demonstrates a near perfect level of quality with a defect rate of 0.02 defects per KSLOC per release. Its primary fixable challenge is predicting EV with more precision.

One organization interviewed uses the sprint retrospective to fix or change peer reviews or process breaks. For example, the team identified a need to incorporate hardening sprints because they found test coverage was not adequate. They learned to think about not only story points for user stories but tasks such as refactoring test automation.

3.2.8.2 Release Retrospectives

The target audience for release retrospectives determines whether there is a release retrospective. Some interview respondents indicated they meet with the customer and conduct a technical interchange meeting (TIM) and a product demonstration. Release retrospectives are recognized as a valued meeting in which the program office and end user provide a feedback loop to the organization. In one contractor organization this feedback loop occurred every 6 to 7 months—so feedback occurred about two times per fiscal year. Another contractor organization only met annually with its customer and the demonstrations with the customer appeared to be a substitute for user acceptance testing. One contractor organization resorted to implementing a "stealth feedback loop" through its field representatives with the end user. Both feedback interchanges were aligned with a release or sprint close to the release completion. Some organizations interviewed would welcome their customer or senior leadership at the release retrospective, but the participation is intermittent.

One interviewee recalls one release retrospective on his project. The meeting was two-part—one was held internally without the customer. The internal meeting appeared to be a dry run for the customer event, and where the team could speak freely. Another difference between the two meetings was the focus on topics—the internal meeting was focused on what was important to the sustainment team (what did it want to fix) while the external meeting with the customer was focused on topics of interest to the customer.

In the case where there was low customer participation, the interviewees indicated there was a real need to increase the number of face-to-face meetings with the customer and the release retrospective was one opportunity to meet and discuss the product.

3.2.8.3 Retrospectives and Incentives

Most sustainment organizations appear to use sprint and release retrospectives to recognize individual and team innovations and contributions. Not all rewards need to be monetary to affect the team proactively. The contractor sustainment situations may use a firm fixed price contract with incentives or award fee—the retrospective provides one avenue to collect input about team performance from the acquiring command for use in incentive or award fee mechanisms.

3.3 Other Considerations Impacting Agile Sustainers

While the organic and contractor sustainment organizations are using the Agile methods successfully, there are still issues not directly related to the Scrum model above that need to be addressed. Issues uncovered during our research include addressing policy changes and guidance, and government staffing changes. There may be, and most likely are, other issues that did not surface during our look at this topic.

3.3.1 Policy and Guidance

Both the organic and contractor sustainment teams were asked pointed questions about the impact of regulatory mandates, policy changes, information assurance, and certification and accreditation (C&A) on sprints and how the team operates. In most cases, the organizations reported that they added one or two hardening sprints to address these issues. During these hardening sprints, work is either completed by the Scrum team, or the Scrum team releases specific work products to another team, which will complete the required regulatory data needs, regulatory training, or information assurance or C&A materials required by the designated approving authority (DAA). In the cases where technical orders (TOs) are part of the standard release, these hardening releases are used for the Scrum team to provide the necessary work products that are needed by another team developing the TOs. The challenge across all organizations was to know when to trigger the hardening sprint for the development team so that both the development team and parallel engineering teams can maintain their team velocity.

One potential solution to knowing when to trigger hardening sprints has been identified within SAFe. Descriptions of the SAFe model include discussion on hardening, innovation, and planning iterations (HIPs) (or sprints). The HIP sprint usually is the last sprint before a release and "provides time for final product validation, system and performance testing, standards and security validations, user acceptance testing and any other Release Readiness activities which are not feasible or economical every sprint" [Leffingwell 2013]. Thus, the hardening sprint is part of the planned Agile release and is always the last sprint before the release is scheduled.

3.3.1.1 Acquisition Objectives

Both organic and contractor interviewees identified "acquisition hoops" that impacted their ability to "be Agile." These "acquisition hoops" often precede the development or sustainment process and will change as necessary over the life of a system. One interviewee responsible for planning an acquisition noted implementing Agile best practices within a traditional acquisition process remains a challenge. Completing work products that support regulatory mandates could be done using Agile methods but the coordination and approval processes will not be streamlined. The Agile team should be able to break up the workload into two-week or four-week parts so a complete suite of documents is planned as tasks in the product backlog. The team can remain Agile if the product backlog includes a list of tasks with dependencies, predecessors, or successors so it is clear when the work product is being worked externally to the team.

3.3.1.2 Supporting Documentation

This Agile approach to developing specification documentation in support of regulatory mandates seems reasonable or even logical until the work products enter the traditional pipeline, and the team can no longer proceed due to external dependencies. In some cases, the regulatory mandate

has a pattern to follow, but some regulatory mandates are new and there is no pattern to follow for execution or the effects of the mandate on the system are not understood. The team will need to decide if accepting the regulatory change without understanding the full impact on the system is desirable—it may make sense to accept the regulatory change and develop a program risk that describes the challenges the system must resolve during the material solution analysis or technology development phases, or at some decision point during the engineering manufacturing development phase. For any Agile team within the acquiring command, there will always be a handoff to a complementary traditional team. Therefore the Agile team needs to decide what work can be completed while their current work product is in coordination with the work in a product backlog. The future challenge is how quickly can the traditional team pick up the work products under coordination and work them to completion. If the sprint cycle is a two-week interval, this challenge should be manageable.

3.3.1.3 Information Technology Policies

One interviewee discussed a regulatory mandate's effect on the development organization. There was an information technology (IT) policy change in his organization that mandated removing all compilers and executable files from development environments. This regulatory change impacted the Agile teams' velocity on the sprint. This situation was avoidable if the IT staff had completed some planning before implementing an IT policy change. The interviewee noted an Agile team can often adjust to impacts that are a few hours in duration, by working lower priority tasks while waiting for the development environment to return to a known state. In this situation, the paperwork to remove a development environment from the policy took days, and then reversing the change took additional time—in all the Agile team was affected for days by a regulatory mandate.

3.3.2 Government Staffing Changes

Both the organic and contractor sustainment teams identified a similar challenge—when the acquiring command experiences permanent assignment or station changes in military personnel, the sustainment team needs to implement a training course about the program's adaptation to Agile processes, or else risk reversion back to traditional methods. In some respects, this is like the Hasbro game of Chutes and Ladders[21] where you move along ladders nicely until you hit the square where you get sent down a chute back to the beginning. In the case of a program office, the staff may be trained in Agile methods and move along well until one day someone gets transferred and the new person coming in is sent back to the beginning to learn about Agile.

The organic teams may be able to adjust more quickly to personnel changes in the acquiring command's program office, but both organic and contractor organizations find there are challenges with personnel changes in key roles. Sustainment organizations may find that engaging with the outgoing personnel to ensure that the replacement is known and then getting on the incoming personnel's calendar quickly so that familiarization training can occur might alleviate some of the challenges with transition. In fact, our interviewees did try to accomplish just this activity but nearly every organization needed to defend or justify their use of Agile methods. Invariably as new government staff joined the program, the rules of engagement changed between the government and contractor and near-heroic measures were implemented to ensure Agile methods were

[21] Chutes and Ladders and all related characters are trademarks of Hasbro.

maintained. In one contractor situation the challenge was merely a process training challenge that was easily achieved, but another contractor explained significant process reversion to traditional or Water-Scrum-Fall methods after making great strides removing the hybrid nature of their Scrum implementation.

Another issue that can have similar results—reversion back to traditional methods from Agile—is changing the product owner or product owner proxy. Agile implementations need a consistent product owner or product owner proxy. Many programs have multiple stakeholders with different views on the priority of the requirements. This consistency is at risk when personnel are changed out without proper understanding of the methods in use.

Even though delivering more functions, features, or defects on a regular interval sounds promising, it is a difficult for some people to transition to an unknown pattern. This Agile-based sustainment organization may behave very differently from what a newcomer expects, based on experience at other programs. If the sustainment organization is not prepared to provide refresher training to new government leadership, then the team may experience some clashing and an ultimate move to another more traditional development and delivery process.

One contractor sustainment organization interviewed resisted the reversion back to traditional methods by implementing a training program for senior leadership, managers, and the Agile team. The contractor sustainment organization found the initial training with senior leadership and all levels of the team—both contractor and government—took approximately eight hours over two days. This eight-hour course best fit the needs of the technical Agile team, but it was too much detail for the management teams. After adjusting the time duration based on the target audience, the contractor's Agile coach worked to identify the frequency of the training. For the contractor, refresher training is triggered by the volume of change in the team composition or new managers to the contractor organization. Additionally, the trigger for government team members' refresher training is based on volume of change in program office personnel or signs that the program office appears to be reverting or moving to an anti-pattern that adversely impacts the team velocity. The training program changed to approximately four hours for middle management and the Agile team members. Senior leadership training transitioned to a two-hour course. This revised training program, shortened material, and attendant frequency appear to be having positive results. Constant adaptation is a key principle of Agile, and adapting the local-level training activities to ensure competence of all stakeholders is a primary example. For organizations employing Agile methods, training on Agile for the government either needs to be in the original contract or negotiated between the contractor and government with visionaries on both sides seeing the need and benefit.

4 DevOps

Sustainment and software development organizations have a growing pressure to reduce the software delivery cycle time while improving the reliability of software deployment. Organizations are finding routine operations of daily builds, release retrospectives, iteration acceptance testing, and the like become easier to manage with continuous integration extended to deployment operations. This change of daily and weekly activities helps organizations avoid the "death march" experiences at the end of each release cycle. The Scaled Agile Framework describes this pattern as "delivery on demand" and "develop on cadence" patterns [Leffingwell 2007]. This need to reduce software delivery cycle time is not unique to DoD and government agencies, but few organizations have thought about how to extend the existing continuous integration processes to include deploying software. This section does not encompass all the technical aspects of DevOps but rather works to explain what DevOps is, why we choose to discuss DevOps in the context of software sustainment, and considerations for organizations planning to implement DevOps design patterns.

For organizations that deliver product with frequencies such as semi-annually, every nine months, annually, or every 27 months as reported by our respondents, the Agile team needs to think about how each sprint will demonstrate continuous flow of value. With deliveries widely spaced, the team still needs to express the value of each sprint to ensure the product owner can express the continuous flow of value. Some DoD programs find that releasing more than annually is cost prohibitive or not possible due to constraints on hardware availability (e.g., aircraft block upgrades). The cost of completing regulatory data, training certification, or numerous information assurance tests and C&A material multiple times per year may not be initially included in the planning data. It may take the next program objectives memorandum (POM) planning cycle before the program can implement the cost and schedule change needed to make multiple deliveries in a year viable. The sustainment organizations we interviewed reported meeting with the acquiring and using command intermittently during a fiscal year—in most cases about twice per year. These interactions generally include a demonstration of the work product or models to ensure the acquiring and using command gain shared ownership of the product and provide a feedback loop that will ensure customer satisfaction at product delivery points. When the delivery cycle reduces from annually or bi-annually, the number of interactions with the acquiring and using command may need to change to quarterly to ensure there is timely feedback for each release.

4.1 What Is DevOps?

DevOps is a blending of the words development and operations. Industry weblogs and published books define DevOps as a balance of development and operation concepts with the ultimate goal of changing cultural mindsets and leveraging technology more efficiently [Swartout 2012]. The culture shift seeks to change the dynamics between software development and operation teams, and the leverage of technology refers to the application of continuous integration to the automation of operational processes. For a practical guide for the technical staff, the book *Continuous Delivery,* by Jez Humble and David Farley, is endorsed by industry practitioners; Paul Swartout's *Continuous Delivery and DevOps: A QuickStart Guide* provides a quick read and practical guide on what DevOps is and is not for leadership or team members [Humble 2010, Swartout 2012].

The weblog *InfoQ* (http://www.infoq.com/devops/) provides articles on what worked and the lessons learned by various industries—few instances are documented for government at either a municipal or federal level. For years, industry and DoD organizations have automated portions of configuration control and build processes; DevOps seeks to implement the processes continuously, which heavily depends on appropriate automation.

DevOps basically extends continuous integration mechanisms to ensure the software development and operation teams interact much earlier, preferably during the design of features. By extending continuous integration to operations and discussing the needs and constraints from the operations team's point of view during the design, the feature design is more complete. Early commercial adopters used Agile methods, such as test-driven development or behavior-driven design, to capture the feature needs and to leverage continuous integration during software development. These adopters wanted to increase the number of feedback loops for the customer and stakeholders involved. Early adopters changed the development release cycle by relying on tool chains or a collection of complementary tools to automate an end-to-end delivery or deployment process. These tool chains enable lifecycle-based automation and rapid responses to changing business conditions and leverage dynamic, changeable constraints by leveraging programmatic interfaces [Haight 2010]. Early adopters realized they applied Agile methods to software development and test processes, but they did not extend their thinking to automating operations—when the automated development lifecycle processes were optimized, these adopters shifted to thinking about how to address processes in operations. For DoD programs, operational efficiency for deployment is a primary goal for program success, and contractors have struggled to deliver products within the maintenance windows available. When a contractor is able to extend its Agile methods to tasks typically allocated to operations teams, it will often look at continuous integration success in software development and apply those lessons learned.

Humble and Farley's *Continuous Delivery* describes a delivery pattern for moving software from development to release [Humble 2010]. This pattern, the *deployment pipeline,* is an automated implementation of an application or system's build, deploy, test, and release process. Generally, no two organizations have the same process, but the continuous delivery principles that govern deployment do not vary. Humble asserts that if an organization uses a value stream for each application or system to map out how it releases software, the build, deploy, test, and release processes related to each system would have variances. When the organizations then moved to the deployment pipeline depicted in Figure 5, they found a common ground that allowed an unvarying view for delivery.

Figure 5: Continuous Delivery Deployment Pipeline [Humble 2010]

The deployment pipeline is based on continuous integration principles. This concept is supported in the literature found on www.infoq.com/devops/ and within the books cited earlier—these sources agree that continuous delivery extends continuous integration to its logical conclusion.

One way to describe the deployment of an automated pipeline is as follows:

> **Epic: Operationalize Deployment.**
>
> Every change to the applications' configuration, source code, environment, or data triggers the creation of a new instance of the pipeline. In addition to creating a new instance of the pipeline, the build installer process creates binaries and installers. The pipeline runs a series of tests to prove the applications' release candidate can be released. Each test the release candidate passes increases confidence that this combination of binary code, configuration information, environment, and data works as specified. When the release candidate passes all the tests, the release is release ready.

Continuous delivery relies on automation throughout the deployment pipeline to ensure that application readiness for release is fully realized. The process of building the release will verify the syntax of the source code. Automated unit tests will ensure the code behaves as specified and designed. The analysis component of this step ensures that quality criteria (such as test coverage) and metrics are collected. Automated testing for functional qualification and functional acceptance are needed to ensure the application conforms to the business acceptance criteria and delivers the business value intended. Similarly, the automated nonfunctional tests check that the application performs sufficiently in terms of capacity, availability, security, and the like while meeting the user's needs. The application will proceed through exploratory testing and demonstration to the customer and selected users. This exploratory test is typically a manual test in a production-like environment where the product owner might identify missing features or bugs that require fixing, or changes needed in the automated test to capture defects in regression testing. All testing is completed in a production-like environment to ensure environmental settings do not affect the applications' ability to function correctly.

For this shift in culture to be successful, the non-development teams need to be diligently collecting or developing the acceptance test for each epic and user story. This will help increase confidence in system results and ensure all points of view—development, test, and operations—are being addressed both in the automated and manual testing programs. Commercial vendors for years have tested their hardware and software products in homogenous and heterogeneous environments to ensure the performance measures stated in product warranties and service-level agreements (SLA) applied by their customers will not cause revenue loss and in turn cause warranty breaches. The due diligence used by commercial vendors to protect the advertised performance measures promotes adoption of methods like continuous delivery, DevOps, or automated deployment. The weblogs and literature survey do not identify continuous delivery as a new concept, but rather the application of automated tools or lean systems thinking to all disciplines or processes prior to and within the deployment pipeline is seen as an affordable method to increase quality and is necessary to operate at the speed of their business.

Humble uses anti-patterns to demonstrate why continuous integration and continuous deployment ensure that application errors or defects are found "in phase." He also shows how product quality increases when continuous integration includes testing the deployment of the system [Humble 2010]. Defects found prior to the manual test block, where the customer and end user are present, are considered found while development still controls the source code.

The literature survey and weblogs are consistent: committing every release into a production-like environment is required. Committing integrating testing, deployment, and release activities into the development process will increase collaboration and will reduce deployment risk because the teams have executed numerous deployment rehearsals. Commercial adopters that have organizations where downtime is considered loss of revenue use pre-production systems that are replicas of the operational system. This environment is where the deployment rehearsals occur to test data conversion and migration, cutover to the pre-production environment, and failback procedures prior to the actual cutover to production. These deployment rehearsals, or installation tests, are scheduled into the project as necessary steps to ensure the updated products integrate so that problems are worked out before release to the customer [Crispin 2009]. These rehearsals or installation tests ensure there is no unplanned loss of service or an excessive volume of help desk tickets that impact the SLA in place for a product line. It is not important what adopters call this activity—rehearsal, installation test, or acceptance test—the net effect is that these dress rehearsals or installation tests demonstrate the increased importance of continuous delivery that relies on continuous testing of application features and the deployment processes.

The deployment pipeline is resource intensive, especially once the comprehensive automated test suite is built. This automation optimizes how human resources are used—the people become less constrained and are then free to work the more interesting, challenging, or value-added tasks while leaving the repetitive low-level tasks to machines. In commercial industry, an application program typically achieves 70% to 95.25% efficiency in finding and removing defects prior to delivery, whereas military programs typically achieve 93% efficiency in finding and removing defects prior to delivery [Jones 2008]. Continuous delivery is another method organizations or programs can use when striving to achieve higher levels of efficiency in finding and removing defects prior to delivery. In the case of DoD programs, there are higher levels of efficiency in finding and removing defects prior to product delivery, but the time to market is typically seen as excessively long from a commercial perspective. Any organization that moves from its current state of development, sustainment, and operations to implement continuous integration extended to DevOps will want to maintain or improve the level of product quality—this goal is no different for DoD or federal IT programs. The significant difference DoD or federal IT programs would strive to achieve through continuous integration is more rapid delivery of products while maintaining or even improving product quality. A DevOps approach will not, however, reduce the time needed to develop the regulatory documentation related to a product.

4.2 Why Are DevOps and Continuous Delivery Discussed Here?

Industry provides tool sets that support automated deployment and extend continuous integration concepts—test automation can be applied to every level of testing. This does not mean every level of test merely exploits the automated testing developed during software integration and test (SWIT). Test automation can run autonomously or semi-automated, and organizations use automation for both formal tests and user acceptance tests. This will not preclude end users from conducting exploratory tests—often ad hoc testing—that focus on how the end user does his or her daily job. When an organization uses DevOps concepts, it will introduce these formal and user acceptance tests earlier in the development lifecycle so that more defects are found “in phase.” Adoption of continuous testing approaches such as those developed for continuous delivery allows the tester to check out the software build on a continuous basis in a production-like environment.

Within the DoD programs researched, the degree to which continuous integration extended to continuous delivery is not clear. It is very possible that contractors have embraced DevOps, continuous delivery or automated deployment where an automated script builds up and tests the production-like or production environments, but these terms are not applied to the effort. As with many Agile practices, this type of work may be done in what we typically call "stealth mode." For example, DoD programs in research, development, test, and engineering (RDT&E) often find test automation for formal functional and nonfunctional test is not available during the initial build-up of an environment or program. But in recent years DoD and federal IT have worked to ensure that programs in sustainment leverage Agile and Lean concepts in their software development, test and operations teams. Where fiscally feasible, organizations will reuse or leverage test automation scripts from formal functional and nonfunctional tests, and they will use acceptance tests as the starting point for regression tests during sustainment checkout. An organization or program in sustainment may not find the continuous delivery concept a huge paradigm shift—rather the shift is making time to complete the refactoring of the test and leveraging test products earlier. Our respondents did not appear to implement these automated deployment or continuous delivery concepts on their programs. However, the discussions generated within the interview process on this topic appeared to trigger almost universal recognition that DevOps and continuous delivery principles stand as another type of process improvement that can help identify system defects earlier in the process, and one that may provide significant time savings as evidenced by commercial adopters.

Implementing a DevOps culture and continuous delivery principles will require an adaptation to how large DoD programs operate. For these principles to be effective the software development, formal test, and operations teams would need to remove communication barriers and prepare for test and operations personnel to build scripts and use tools earlier in the typical workflow. For example, both formal test and operations teams need to work in the production-like environment "owned" by software development prior to a handoff for formal testing. Teams such as formal test and operations often find it difficult to gain access to their software development teams' production-like environment. To succeed, the software development team needs to open up "their" production-like environment to multidisciplinary teams so more defects can be found in-phase. Many large organizations and DoD programs will find this paradigm shift a nontrivial challenge. When large organizations and programs remove organizational barriers and operate as a cohesive team, the survey literature indicates the defects found in phase are corrected more quickly. Repeated user stories online[22] reported the transition to a multidisciplinary team—software engineering, software development, test, and operations. This multidisciplinary team located and resolved defects typically found late in the software development lifecycle. In addition, these teams developed into a cohesive unit that produced a level of consistency not possible in the past organizational structure. For DoD and federal IT programs, engaging formal test and operations teams while the software is still in development will enable the team to discover defects typically not found during configuration item or software integration testing. This shifting to end-game thinking, where end-to-end, systemic functional, and nonfunctional testing concepts are tested while the software development teams are responsible for the production-like environment ensures the software developer recognizes the defects and resolves the defect in the next sprint.

[22] http://www.infoq.com/devops/

Although continuous integration principles always included the deployment pipeline, the culture shift for continuous delivery engages the software development team within the automation of processes needed for formal test or operations. Typically, software development will *own* the production-like environment, and programs that implement continuous delivery demonstrate that environment ownership is not important when every team is responsible for delivery. A multidisciplinary, cohesive team will not focus on environment ownership—members focus on product quality and completing the necessary work before product release. When an organization is dedicated to producing releasable software on every code commit, then the patterns used to develop software will change. For example, feature toggles can be very useful for committing code early, allowing developers to release a feature before it is ready for use by end users. Clearly, the testers need to verify that the feature toggle works before the product release. Note that this approach is frowned on in principle by the information assurance (IA) community because it results in inactive code in an operational product. This is an example of a barrier that will have to be addressed if this approach is used.

Another example is branching from the code trunk. Although configuration management SMEs discourage this practice, branching can be adapted to fit within continuous delivery principles. Some organizations have a requirement to support multiple variants of the same code base, which means that the configuration management (CM) tool must have the capability to support complex branching and merging [Aiello 2011]. It is worth noting here that continuous delivery SMEs discourage excessive use of branching from the code trunk, but both continuous delivery and CM experts accept code branching and merging will occur therefore they promote merging back to the code trunk using a simple branching strategy. An organization using continuous delivery and continuous testing needs to ensure that the emergency patches or code on different code branches do not introduce new defects. Every organization recognizes emergency patches are sometimes needed, and the configuration management and build processes adapt by introducing a branch and merge process that has enough process to maintain consistency and allows the organization to implement just-in-time improvements needed for efficient, reliable, and repeatable processes while reducing unnecessary steps [Aiello 2011].

4.3 Is DevOps Being Used in the DoD, and How Could It Be Used?

As noted earlier, continuous delivery in DoD programs does not appear to be widespread, and if used it is likely to be implemented using stealth mode practices. As noted previously, continuous delivery allows an organization to continuously test software builds in the production-like environment and find defects prior to product release. DoD and federal IT programs will find introducing formal and user acceptance tests into the production-like environment changes how testers look at tests and the complementary data needed to fully test the system. For example, when an RDT&E program implements continuous integration with continuous delivery concepts, the testers may find test automation of functional and nonfunctional tests can occur earlier in sprints or releases and the results are highly useful. In turn, when these programs enter sustainment it is easier to embrace continuous delivery and leverage existing test automation scripts from functional, nonfunctional, and acceptance tests for regression or retest during environment check-out. As noted earlier, for sustainment teams DevOps or continuous delivery is not a huge paradigm shift—these teams already work to allocate time in the schedule to refactor test products and leverage existing test products earlier into the process.

4.4 Which Sustainment Organizations Need to Consider Implementing DevOps?

DoD network systems or service-oriented applications connected to the DoD network or federal IT systems should see a benefit from implementing continuous delivery mechanisms and process improvements. Mission capability systems that are designated ground systems supporting airframes, spacecraft, or satellites will need to deploy to a pre-production environment much like a DoD network system or service-oriented application. After the checkout in the pre-production environment confirms the system is ready, cutover to production occurs. Mission capability systems that are designated support equipment or support systems to airframes could implement DevOps at least from software development through formal testing. The DevOps pattern will need to stop short of automating the deployment if the system connects and pushes data to airframe operational flight programs (OFPs).

This continuous delivery concept does not need to target only sustainment. Programs in RDT&E could implement DevOps to save schedule and cost while the system iterates through development and test prior to the transition for deployment. DevOps, continuous delivery, or automated deployment can be implemented prior to sustainment or during the extended sustainment period—programs that deploy to multiple sites may have multiple environments in test at the same time—and using automated deployment to roll out software to these environments under test demonstrates how the process will work from initial operational capability through sustainment. Programs in sustainment with a plan to "sunset" or have an "end of life" within the next two years will not likely find implementing a DevOps approach reasonable. This DevOps approach scales from small projects to large programs and the volume of cost, resource savings, and time to-market savings are being learned about across commercial organizations and economies [Aiello 2011]. DoD and government IT programs that transition to DevOps, continuous delivery, or automated deployment should see enough savings to justify expending the time and effort for this process improvement.

It would be naïve to think that a program that uses the DevOps pattern for deployment would have no other issues to be addressed. This continuous deployment approach will need to be tailored to work within existing delivery mechanisms—therefore, expect there to be governance and program process changes needed. Some of the challenges DoD trailblazers may encounter when adopting DevOps include but are not limited to the following:

- What governance is needed to manage the infrastructure changes, to ensure the correct versions are installed, and to manage the corrections to configuration and environment settings installed?
- What additional tests or verification, validation, and accreditation (VV&A) are needed to ensure the tools used do not trigger IA or C&A concerns?
- Are there architectural tenets needed to implement DevOps, continuous delivery, or automated deployment? How much refactoring of the system architecture is really needed?

This list is far from exhaustive but demonstrates that the trailblazers will need patterns—these patterns are likely to be available from industry, but tailoring will be needed for DoD or federal IT programs. As changes to patterns occur, the trailblazers need to think about applying Agile or Lean concepts and remove redundant or non-value-added steps to streamline the processes.

5 Conclusion

The authors undertook this paper thinking that Agile methods were very conducive to use in sustainment. Using Agile seemed obvious since the nature of fixing defects was to create a list of defects, prioritize that list, and work the list in the order provided. This process seemed to provide a ready-made backlog as it mirrors how Agile backlogs are created. However, our interviews and surveys revealed that most sustainment organizations already had well-operating sustainment methods knowingly or unknowingly based on Agile concepts and principles. We learned that these groups were always looking to improve their processes and had looked at Agile principles and practices, adopting certain practices as warranted to improve their existing methods. Thus, most of the processes used in the sustainment arena are some form of hybrid—a mixture of traditional and Agile practices. With this in mind, we looked at how the sustainment organizations implemented Agile principles and concepts using the Scrum method as a model since most if not all were using parts of the Scrum methodology.

During our research we uncovered a fairly new concept that is taking hold in the commercial arena. It is an extension of continuous integration into continuous deployment. This new concept is called DevOps. DevOps is new and may or may not find a home in DoD acquisitions. There is a need for some type of process like DevOps as it will save time; however, it will be a challenge to combine software and sustainment developers with operations. Historically acquisition and operations are two separate domains, and the two organizations meet closer to the end game to determine operational readiness. If the software development and sustainment organizations can combine continuous integration and deployment earlier in the system lifecycle, then organizations will see software delivered to the end user (the warfighter) faster. Further research and exploration needs to be done in this area as more programs adopt DevOps.

For readers who desire further insight on Agile, sustainment, and DevOps, we have included

- Appendix C, a comprehensive listing of the SEI's published work on Agile
- Appendix D, a combined reading list for the benefit of the reader from several well-known Agile resources

Appendix A Respondent Characteristics

The following tables and figures depict the overall characteristics of the respondents. They include characterization of

- contract vehicle
- system classification
- type of sustainment
- Agile methods used
- role on team
- requirement changes during release
- team knowledge about product roadmap
- knowledge of time interval into future product roadmap
- team perception of team pace
- trust between government and team
- Agile effectiveness in sustainment

Table 2: Respondent Characterization of Contract Vehicle

Type of Contract Vehicles	**Survey [20]**	**Interview [8]**
Single contract	6	
Multiple-award task order contract	2	3
Task order under existing multiple-award contract	3	2
Interdepartmental transfers or purchase request	4	2
Cost Plus Fixed Fee	2	2
Cost Plus Incentive Fee	1	
Cost Reimbursement, Time and Materials (T&M), labor hour, or variations	1	2
Fixed Price (FP)	2	2
Other-hybrid of FFP and T&M	1	
Organic Support	1	
I do not know	14	3
I prefer not to answer	2	

Table 3: Respondent Characterization of System Classification

Type of System Classification	**Survey [20]**	**Interview [8]**
Safety of Life	1	1
Weapons Systems	5	6
Mission Capability	5	1
I do not know, I prefer not to answer	9	
Total	20	

Table 4: Respondent Characterization of Type of Sustainment

Type of Sustainment	Survey [20]	Interview [8]
Software maintenance	8	8
Contractor support	1	
Corrective maintenance only on an existing system	0	
Corrective maintenance plus minor enhancements on an existing system	1	2
Corrective maintenance, minor enhancements, and major upgrades for an existing system	9	4
I do not know	0	
I prefer not to answer	0	

Table 5: Respondent Characterization of Agile Methods Used

Agile Method Characterization	Survey [9]	Interview [8]
Scrum	4	4
Hybrid methods (i.e., Scrum and Waterfall)	2	
Hybrid methods (i.e., Scrum and Kanban)		1
Hybrid methods (i.e., Scrum, Kanban-modified Feature Driven Design, Scalable Agile Framework, and DSDM)		1
Hybrid methods (i.e., Scrum, TSP, and PSP)		2
Kanban	2	
I do not know	1	
I prefer not to answer	0	

Table 6: Respondent Characterization of Role on Team

Role on Team Characterization	Survey [5]	Interview [8]
Advocate	3	6
Scrum Master/Product Owner		1
Product Owner	1	
Developer/Tester		1
Other - Unspecified	1	
I do not know	0	0
I prefer not to answer	0	0

Table 7: Respondent Characterization of Requirement Changes During Release

Characterization of Requirement Changes During Release	Survey [5]	Interview [8]
Less than 1 per month	2	2
2-3 times per month	3	1
Response not quantifiable		5
I do not know	0	0
I prefer not to answer	0	0

Table 8: Respondent Characterization of Team Knowledge About Product Roadmap

Characterization of Team Knowledge About Product Roadmap	Survey [5]	Interview [8]
Rarely	1	
Sometimes	1	
Most of time	2	5
No response recorded		3
I do not know	1	
I prefer not to answer	0	0

Table 9: Respondent Characterization of Time Interval into Future Product Roadmap Is Known

Characterization of Product Roadmap Forecast	Survey [5]	Interview [8]
Less than 3 months	3	
6-12 months	2	3
No response recorded		5
I do not know	0	0
I prefer not to answer	0	0

Table 10: Respondent Characterization of Government Versus Team Perception of Team Pace

Characterization of Team Pace	Survey [5]		Interview [8]	
	Government Perception	Team Perception	Government Perception	Team Perception
Neutral	1	1	2	
Somewhat slow	1		1	3
Slow			1	2
Fast		1		
Somewhat fast	2	1		
Very fast	1	2		
No response recorded			4	3
I don't know	0	0	0	0
I prefer not to answer	0	0	0	0

Table 11: Respondent Characterization of Trust Between Government and Team

Characterization for Level of Trust Between Government and Team	Survey [5]	Interview [8]
Very good	1	1
Good	3	2
Low	1	1
No response recorded		4
I don't know	0	0
I prefer not to answer	0	0

Table 12: Respondent Characterization of Agile Effectiveness in Sustainment

Characterization of Agile Effectiveness in Sustainment	Survey [5]	Interview [8]
Neutral	2	1
Helpful	2	3
Very helpful	1	1
No response recorded		3
I don't know	0	0
I prefer not to answer	0	0

Appendix B Survey Questions

This appendix contains a copy of the actual survey questions.

Sustainment and Agile

Survey respondents should know the following:

This survey is part of a research study on the barriers and enablers to adoption of agile methods in DoD and other highly regulated settings, specifically as they relate to the interface between Agile software development and sustainment operations.

- Participation in this survey is voluntary and participants may withdraw from the study at any time. You may always select "I prefer not to answer."
- No identifying information will be collected in association with survey responses.
- There are no risks to participants associated with participation in the survey.
- Benefits to participation revolve around sharing of insights into barriers and enablers of Agile adoption in acquisition and development settings.
- Participants must be 18 years of age or older.

Characterize Sustainment and Agile Contract Vehicle

Q62 1. Identify type of Acquisition Category (ACAT) I program you support that complies with Major Defense Acquisition Programs (MDAP) (section 2430 of Reference (k)) or Milestone Decision Authority (MDA) designation

Q66 5. Identify the type of contractual arrangement used for your sustaining product:

Q67 6. Identify the type of contract vehicle used for engagement:

Q49 7. What is the criticality of the system under sustainment?

Q74 8. Does your team use or are they considering to use Lean or Agile development methods such as Scrum, Kanban, eXtreme Programming?

Characterize Operations and Sustainment Planned

Q8 9. Identify the depot level maintenance planned, where according to 10 U.S.C. §2460, the term "depot-level maintenance and repair" means the overhaul, upgrading, or rebuilding of parts, assemblies, or subassemblies. This includes all elements of:[23]

[23] From 10 U.S.C. §2460: In this chapter, the term "depot-level maintenance and repair" means (except as provided in subsection (b)) material maintenance or repair requiring the overhaul, upgrading, or rebuilding of parts, assemblies, or subassemblies, and the testing and reclamation of equipment as necessary, regardless of the source of funds for the maintenance or repair or the location at which the maintenance or repair is performed. The term includes (1) all aspects of software maintenance classified by the Department of Defense as of July 1,

Q9 10. Which of the following best characterize your situation?

Q11 11. Characterize the makeup of the sustainment development team, use the slider bars to identify percentage of support personnel used.

Characterize the Agile Training Needs

Q15 12. Identify how easy or difficult it was to "sell" Agile development to staff in the government Program Office.

Q42 13. What kind of sponsorship was provided when you began adoption of Agile or lean methods in operations and sustainment?

Q22 14. Identify sprint durations and frequency of releases:

Q17 15. Explain how your government team "learned to be Agile" or "shifted to the Agile paradigm."

Q43 16. What level of training/experience using the selected Agile methods does the sustainment developer team have?

Q18 17. Do your integrated project teams (IPTs) use the following Lean or Agile methods?

Q21 18. Identify progress and completion data reported to PMs and the interval of the reporting

Q23 19. Is the deployment of software dependent upon a technology refresh schedule?

Characterize the Agile Methods Used

Q36 20. Explain how Agile methods improved (or did not improve) your team's ability to meet their program goals.

Q81 21. Does using Agile methods provide cost or schedule savings? If yes, explain the type(s) of savings realized. Conversely, if no, explain the type(s) of overruns encountered and why.

Q69 22. What is your role on the Agile sustainment team?

Q59 23. How well is the product roadmap known by the sustainment staff?

Q71 24. What time interval into the future is the sustainment team aware of system changes?

Q70 25. How well do the program goals reflect known stakeholders' concerns? For example, how well does the software part of the program align with the stakeholders goal, are there disconnects or are they well aligned?

1995, as depot-level maintenance and repair, and (2) interim contractor support or contractor logistics support (or any similar contractor support), to the extent that such support is for the performance of services described in the preceding sentence.

(b) Exceptions.- (1) The term does not include the procurement of major modifications or upgrades of weapon systems that are designed to improve program performance or the nuclear refueling or defueling of an aircraft carrier and any concurrent complex overhaul. A major upgrade program covered by this exception could continue to be performed by private or public sector activities.

(2) The term also does not include the procurement of parts for safety modifications. However, the term does include the installation of parts for that purpose.

Q74 26. In what ways do program oversight mechanisms align with agile principles? (For example, good alignment might be indicated by active involvement in the program office oversight staff as Product Owners including active attendance at sprint/iteration reviews, and minimization of non-value adds compliance documentation.)

Q44 27. Where are the sustainment team members located?

Q45 28. How often are requirements changes that are discovered during sustainment activities addressed?

Q46 29. What describes the sustainment team's perception about the effects of Agile methods on their ability to accomplish their work?

Q47 30. What is the level of trust between the government and sustainment development team members?

Q48 31. How is the pace of development perceived by the government Program Office?

Q59 32. How is the pace of development perceived by the sustainment development team?

Q24 E2. Explain the Agile practices used for release planning.

Q25 E3. Explain the Agile estimation process used for release planning.

Q26 E4. Explain the use of release retrospectives.

Q27 E5. Explain the Agile practices used for sprint planning.

Q28 E6. Explain the Agile estimation process used for sprint planning.

Q29 E7. Explain the use of sprint retrospectives.

Q30 E8. Explain how end users or end user representatives engage in your Agile process?

Q14 E9. Explain what motivated your program/organization to use Agile (or lean) methods in your sustainment efforts.

Q16 E10. Explain the training needed for your sponsor or your management chain to understand the Agile methods.

Q10 E11. Use the text block below to describe how the system is in operation and handles development at the same time, this is sometimes called “devops.” If there is an alternative process when the system deploys an Emergency Patch for vulnerabilities include that description.

Q72 E12. How is collaboration between sustainment staff and end users enabled?

Q73 E13. How are practices like interim demonstration and delivery accommodated within the contract or memorandum of understanding (MOU) for the sustainment activities?

Q75 E14. How do you align the software-related goals with the program level goals?

Q76 E15. Whatever mechanism you are using to authorize sustainment work, does that mechanism (i.e., contract, memorandum of agreement (MOA), MOU, etc.) account for Agile methods? If so, how? If not, what kinds of problems do you think this causes for your sustainment teams who are using Agile methods?

Q31 E16. Explain how Agile processes used impacts or supports the certification & accreditation process for example gaining the Authority to Operate (ATO).

Q32 E17. Explain how information assurance is handled in the sustainment development, test environment, staging/pre-production environments; and the tools used within these environments. For example, are there hardening sprints, or some other method applied to support information assurance processes.

Q33 E18. Explain how your organization handles regulatory or policy changes (i.e., DoD, command, local). The response might want to focus on the following:

- Is it easier to employ these regulatory or policy changes due to your new techniques?
- Does the process remove roadblocks, if yes which ones?

- What barriers did you encounter with external groups understanding your Agile processes?

Q34 E19. Explain how the team handles updates to COTS software, for example are changes completed in parallel with functionality changes or are COTS software updates separate releases.

Q80 E20. Explain the deployment process for your team. For example, answer questions such as:

- Does your team use any scripting or automation to support deployment?
- If yes, can you provide a percentage of automation to manual operations? (i.e., 70-30, 80-20, 90-10)
- How long does it take to deploy your system into operations on average?

Q35 E21. Explain how end user support (i.e., tier 1 help desk, tier 2 help desk, etc.) occurs along with the parallel sustainment of the system.

Q35 E22. Characterize the recovery from failures during traditional operations and sustainment, For example, answer questions such as:

- Did the recovery from failures change when the team used Agile processes?
- Did the number of recovery instances increase or decrease?
- What, if any process improvements, or lessons learned were realized by the team?
- Which development method (i.e., traditional or Agile) deliver more reliable changes?

Q36 E23. Characterize how much it costs your team if the system fails. For example:

- Is there an hourly cost when a hard outage occurs?
- Does your team implement service level agreements (SLAs) to maintain 24/7 operations?
- What happens if there are multiple application failures within a month?

Appendix C SEI Published Agile Papers

- *Considerations for Using Agile in DoD Acquisition* [CMU/SEI 2010-TN-002]
  http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=9273
- *Agile Methods: Selected DoD Management and Acquisition Concerns* [CMU/SEI-2011-TN-002]
  http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=9769
- DoD Information Assurance and Agile: Challenges and Recommendations Gathered Through Interviews with Agile Program Managers and DoD Accreditation Reviewers [CMU/SEI 2012-TN-24]
  http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=34083
- *A Closer Look at 804: A Summary of Considerations for DoD Program Managers* [CMU/SEI-2011-SR-015]
  http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=9751
- *Agile Methods and Request for Change (RFC): Observations from DoD Acquisition Programs* [CMU/SEI-2013-TN-031]
  http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=77732
- *Potential Use of Agile Methods in Selected DoD Acquisitions: Requirements Development and Management* [CMU/SEI-2013-TN-006]
  http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=89158
- *Parallel Worlds: Agile and Waterfall Differences and Similarities* [CMU/SEI-2013-TN-021]
  http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=62901
- *Agile Metrics: Progress Monitoring of Agile Contractors* [CMU/SEI-2013-TN-029]
  http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=77747

Appendix D Reading List

The reading list in tabular form below is comprised of an aggregation of lists from the following sources. We provide this list for our reader's convenience; the SEI has not endorsed the materials. The information in the brackets identifies the table heading label, and the "X" in the row indicates the reading list(s) in which the book appears:

- Scrum Alliance
  - [SA-S] Scrum Reading List (source: http://www.scrumalliance.org/community/articles/2007/august/scrum-reading-list)
  - [SA-L] Leadership Reading List (source: http://www.scrumalliance.org/community/articles/2007/january/leadership-reading-list)
- Program Management Institute (PMI) Agile Certified Practitioner (ACP) reading lists point to the same information:
  - [PMI-ACP RL] Reference List (source: http://www.pmi.org/Certification/~/media/PDF/Certifications/ACP_Reference_list_v2.ashx)
  - [PMI-ACP RL] PMI-ACP Examination Reference List (source: http://www.pmi.org/Certification/~/media/Files/PDF/Agile/PMI000-GainInsightsAIGLE418.ashx
- [LSS] Lean System Society (LSS) Reading List

Title	Author	Publishing Information	SA-S	SA-L	PMI-ACP-RL	LSS
Agile CMMI: Why Isn't This Conversation Dead Yet?		Cutter Journal (issue available for purchase) http://www.cutter.com/itjournal/fulltext/2012/11/index.html				X
Situational Leadership for Agile Software Development	Mike Cohn	Cutter IT Journal, June 2004		X		
Agile and Iterative Development: A Manager's Guide	Craig Larman	ISBN-10: 0131111558	X			
Agile Estimating and Planning	Mike Cohn	ISBN-10: 0-13-147941-5	X		X	
Agile Project Management with Scrum	Ken Schwaber	ISBN10: 073561993X	X		X	
Agile Project Management: Creating Innovative Products – 2nd Edition	Jim Highsmith	ISBN 0321658396			X	
Agile Retrospectives	Esther Derby and Diana Larsen	ISBN: 978-0-9776-1664-0	X			

Title	Author	Publishing Information	SA-S	SA-L	PMI-ACP-RL	LSS
Agile Retrospectives: Making Good Teams Great	Esther Derby, Diana Larsen, and Ken Schwaber	ISBN 0977616649			X	
Agile Software Development: The Cooperative Game – 2nd Edition	Alistair Cockburn	ISBN 0321482751	X		X	
Agile Software Development with Scrum	Ken Schwaber and Mike Beedle	ISBN10: 0130676349	X			
Becoming Agile: ...in an Imperfect World	Greg Smith and Ahmed Sidky	ISBN 1933988258			X	
CMMI – Lean First	Hillel Glazer	http://www.agileCMMI.com/index.php/2012/03/short-cut-to-CMMI-lean-first				X
CMMI or Agile? Why Not Embrace Both	Hillel Glazer, Jeff Dalton, David J. Anderson, Mike Konrad, and Sandy Shrum	http://www.sei.cmu.edu/reports/08tn003.pdf				X
Coaching Agile Teams	Lyssa Adkins	ISBN 0321637704			X	
Collaboration Explained: Facilitation Skills for Software Project Leaders	Jean Tabaka	ISBN 0321268776	X			
Confessions of an UnManager: Ten Steps to Jump Start Company Performance by Getting Others to Accept Accountability	Debra Boggan and Anna Versteeg	ISBN 1892538148		X		
High Performance Leadership: Creating, Leading and Living in a High Performance World	Graham Winter	ISBN 0470820810		X		
Leadership Passages: The Personal and Professional Transitions that Make or Break a Leader	David L. Dotlich, James L. Noel, and Norman Walker	ISBN 0787974277		X		
Leading Change	John Kotter	ISBN 0875847471		X		
Leading Geeks: How to Manage and Lead the People Who Deliver Technology	Paul Glen	ISBN 0787961485		X		
Leading Minds: An Anatomy of Leadership	Howard Gardner with Emma Laskin	ISBN 0006381235		X		
Lean-Agile Software Development: Achieving Enterprise Agility	Alan Shalloway, Guy Beaver, and James R. Trott	ISBN 0321532899			X	
Managing Transitions: Making the Most of Change	William Bridges	ISBN 0738208248		X		
On Becoming a Leader	Warren Bennis	ISBN 0738208175.		X		
Primal Leadership: Realizing the Power of Emotional Intelligence	Daniel Goleman, Annie McKee, and Richard E. Boyatzis	ISBN 157851486X		X		
Resolving Conflicts at Work: Eight Strategies for Everyone on the Job	Kenneth Cloke and Joan Goldsmith	ISBN 0787980242		X		
Servant Leadership: A Journey into the Nature of Legitimate Power and Greatness	Robert Greenleaf	ISBN 0809105543		X		
The Art of Agile Development	James Shore	ISBN 0596527675			X	
The Enterprise and Scrum	Ken Schwaber	ISBN-13:	X			

Title	Author	Publishing Information	SA-S	SA-L	PMI-ACP-RL	LSS
		9780735623378				
The Leadership Challenge	James M. Kouzes and Barry Z. Posner	ISBN 0787968331		X		
The Next Level: Essential Strategies for Achieving Breakthrough Growth	James B. Wood	ISBN 0738201596		X		
The One Thing You Need to Know: ... About Great Managing, Great Leading, and Sustained Individual Success	Marcus Buckingham	ISBN 0743261658		X		
The Ropes to Skip and the Ropes to Know: Studies in Organizational Behavior	R. Richard Ritti and Steven Levy	ISBN 0471736465		X		
The Software Project Manager's Bridge to Agility	Michele Sliger and Stacia Broderick	ISBN 0321502752			X	
User Stories Applied for Agile Software Development	Mike Cohn	ISBN-10: 0321205685	X		X	
Viewing a Lean System Through a CMMI Lens	Richard Hensley [at Lean Systems and Software Conference 2010]	http://www.leanssc.org/videos/lssc11/Hensley/Hensley.html				X

Additional reading related to DevOps:

Title	Author	Publishing Information
Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation	Jez Humble and David Farley	Addison-Wesley, August 2010 ISBN: 0321601912
Continuous Delivery and DevOps: A QuickStart Guide	Paul Swartout	Packt Publishing October 2012 ISBN-13: 9781849693684

References

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

[Agile 2001]
Manifesto for Agile Software Development. 2001.
http://agilemanifesto.org/

[Agile Alliance 2013]
Agile Alliance. *Backlog*. 2013.
http://guide.agilealliance.org/guide/backlog.html#sthash.tcYs4W14.dpuf

[Aiello 2011]
Aiello, B. & Sachs, Leslie. *Configuration Management Best Practices: Practical Methods that Work in the Real World*. Addison-Wesley Pearson Education, Inc., 2011 (ISBN 0-321-68586-5).

[Air Force 2013]
Department of the Air Force. *Presentation to the House Armed Services Committee Subcommittee on Tactical Air and Land Forces*, U.S. House of Representatives; Combined Statement of Lt. Gen. Burton M. Field and Lt. Gen. Charles R. Davis. 2013.
http://docs.house.gov/meetings/AS/AS25/20130417/100671/HHRG-113-AS25-Wstate-FieldL-20130417.pdf

[Anderson 2005]
Anderson, David; Augustine, Sanjiv; Avery, Christopher; Cockburn, Avery; Cockburn, Alistair; Cohn, Mike; DeCarlo, Doug; Fitzgerald, Donna; Highsmith, Jim; Jepsen, Ole; Lindstrom, Lowell; Little, Todd; McDonald, Kent; Pixton, Pollyanna; Smith, Preston; & Wysocki, Robert. *The "Declaration of Interdependence" for Modern (Agile/Adaptive) (Product/Project) Management.* 2005.
http://alistair.cockburn.us/The+declaration+of+interdependence+for+modern+management+or+DOI/v/slim

[Boxer 2009]
Boxer, P. & Garcia, S. *Limits to the Use of the Zachman Framework in Developing and Evolving Architectures for Complex Systems of Systems*. SATURN Conference, May 2009. Software Engineering Institute, 2009.
http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=20518

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

[Crispin 2009]
Crispin, L. & Gregory, J. *Agile Testing: A Practical Guide for Testers and Agile Teams*. Addison-Wesley Pearson Education, Inc., 2009 (ISBN 0-321-53446-8).

[DAU 2011]
Defense Acquisition University. *Integrated Product Element Guidebook.* DAU, November 2011. https://acc.dau.mil/CommunityBrowser.aspx?id=496319

[DAU 2013]
Defense Acquisition University. *Sustainment Archived References,* 2013. https://acc.dau.mil/CommunityBrowser.aspx?id=677043

[Haight 2010]
Haight, C. *DevOps: Born in the Cloud and Coming to the Enterprise*. Gartner Research, 2010. ID Number: G00208163

[Humble 2010]
Humble, Jez & Farley, David. *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation.* Addison-Wesley, August 2010 (ISBN: 0321601912).

[Johnson 2012]
Johnson, Phil. *You May Kiss the Bride and Update the Scrum Board*. 2012. http://www.itworld.com/it-managementstrategy/289060/you-may-kiss-bride-and-update-scrum-board

[Jones 2008]
Jones, C. *Applied Software Measurements: Global Analysis of Productivity and Quality, Third Edition*. Addison-Wesley Pearson Education, Inc., 2008 (ISBN 978-0-07-150244-3).

[Lapham 2006]
Lapham, Mary Ann & Woody, Carol. *Sustaining Software-Intensive Systems* (CMU/SEI-2006-TN-007). Software Engineering Institute, Carnegie Mellon University, 2006. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7865

[Lapham 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://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9273

[Lapham 2011]
Lapham, Mary Ann; Garcia-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://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9769

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

[Leffingwell 2013]
Leffingwell, Dean. *Hardening | Innovation | Planning Abstract.* 2013
http://scaledagileframework.com/hardening-innovation-planning/

[Mountain Goat Software 2005]
Mountain Goat Software. *Key Elements of Scrum.* 2005.
http://www.mountaingoatsoftware.com/scrum/figures

[NRC 2011]
National Research Council. *Examination of the U.S. Air Force's Aircraft Sustainment Needs in the Future and Its Strategy to Meet Those Needs.* Committee on Examination of the U.S. Air Force's Aircraft Sustainment Needs in the Future and Its Strategy to Meet Those Needs, National Academy of Sciences, 2011.
http://www.nap.edu/openbook.php?record_id=13177&page=R1

[NRC 2013]
National Research Council. *Zero-Sustainment Aircraft for the US Air Force: A Workshop Summary*. Committee on Zero-Sustainment Aircraft for the U.S. Air Force: A Workshop; Air Force Studies Board; Division on Engineering and Physical Sciences, 2013.
http://www.nap.edu/catalog.php?record_id=18295

[Palmquist 2013]
Palmquist, Steven; Lapham, Mary Ann; Garcia-Miller, Suzanne; Chick, Timothy; & Ozkaya, Ipek. *Parallel Worlds: Agile and Waterfall Differences and Similarities* (CMU/SEI-2013-TN-021). Software Engineering Institute, Carnegie Mellon University, 2013.
http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=62901.

[Swartout 2012]
Swartout, Paul. *Continuous Delivery and DevOps: A QuickStart Guide*. Packt Publishing, 2012 (ISBN-13 9781849693684).

[Taylor 2012]
Taylor, Mike & Murphy, Joseph "Colt." "OK, We Bought This Thing, but Can We Afford to Operate and Sustain It?" *Defense AT&L: Product Support Issue* (March–April 2012): 17-21.
http://www.dau.mil/pubscats/ATL%20Docs/Mar_Apr_2012/Taylor_Murphy.pdf

[VersionOne 2014]
VersionOne. *8th Annual State of Agile Survey*. VersionOne, Inc. 2014.
http://stateofagile.versionone.com/

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

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

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Agile Methods in Air Force Sustainment: Status and Outlook	FA8721-05-C-0003

6. AUTHOR(S)
Colleen Regan
Mary Ann Lapham
Eileen Wrubel
Stephen Beck
Michael Bandor

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2014-TN-009

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

11. SUPPLEMENTARY NOTES

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

13. ABSTRACT (MAXIMUM 200 WORDS)

This paper examines using Agile techniques in the software sustainment arena—specifically Air Force programs. The Software Engineering Institute has researched the viability of Agile software development methods within Department of Defense programs and barriers to the adoption of those methods for several years. How software sustainers leverage Agile methods and avoid barriers to using Agile methods are addressed in this paper. In addition, the potential use of a construct called DevOps, a blending of development and operations, is discussed.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Agile, sustainment, DevOps	66

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