Incorporating Agile Methodologies into DoD Software Sustainment Operations

Lacey Schley
University of Oklahoma
Norman, USA
laceyd@ou.edu

Randa L. Shehab
University of Oklahoma
Norman, USA
rlshehab@ou.edu

John K. Antonio
University of Oklahoma
Norman, USA
antonio@ou.edu

***Abstract*—The DoD is increasingly employing agile methods in efforts to rapidly produce software. However, the specific domain of agile methodologies as applied to the sustainment of embedded systems remains in need of further research. One initial area examined here is the relationship between agile principles and the policies and practices used by DoD sustainment organizations. It is shown that current sustainment policy for the Air Force Sustainment Center is compatible with agile principles. Challenges specific to sustainment operations can be resolved through careful consideration of the scope and organizational levels at which agile principles and practices are implemented.**

Keywords—agile software development, sustainment, aerospace, software, department of defense

I. Introduction

The need to deploy software at a faster pace is acutely felt within the military aerospace community. At the 2019 Air Force Association's Air Warfare Symposium, assistant secretary of the Air Force for acquisition, technology and logistics Dr. William Roper noted that the Air Force must be capable of updating the software rapidly if it hopes to "win" [1]. A year prior at the Air Force Information Technology and Cyberpower Conference, Secretary of the Air Force Heather Wilson spoke of the need to modernize processes to enable the Air Force to deploy new technologies at a rate that can keep up with what our adversaries are achieving [2]. For the past decade, the United States Department of Defense (DoD) has looked toward agile methodologies as a means of rapidly developing and fielding new software capabilities. The term "agile" has come to refer to a software development culture centered around collaborative, cross-functional teams that work closely with customers and end users to define requirements as the product evolves.

Several reports have examined not only the applicability of agile practices within the defense sector, but also the types of programs that are suited for agile practices. In the past, certain programs have demonstrated successful adoption of agile practices within the defense sector. These programs largely focused on new development of standalone, support products rather than existing, operational aircraft products.

One area within the DoD that still struggles with adopting agile methodologies is the software sustainment segment. Unlike the development of new products, software sustainment deals with maintaining, updating, and modernizing existing aircraft software. In the context of sustainment, major design and architectural decisions for the software have already been made, often decades prior, and cannot easily be recreated to align with modern software practices. Moreover, modern software development tools and environments cannot easily interface with these legacy software systems. Further complications are due to the mismatch between the traditional overarching DoD sustainment processes and versus the philosophy behind agile software practices.

Even putting aside agile practices, overlaying the traditional DoD sustainment process onto software maintenance has resulted in unattainable expectations on the management and performance of DoD software teams. To understand the difficulty in applying the traditional DoD sustainment process to maintaining software, consider first a scenario in which the traditional process actually works well. Specifically, suppose the landing gear on an aircraft breaks; in this situation, the maintainers can generally plan and accurately estimate how long that aircraft will be down to replace the necessary parts of the faulty landing gear assembly. In contrast, if a cockpit display unit incorrectly displays an error message when the landing gear extends, the root cause could be any number of software problems, e.g., a bug in the graphics, improper condition handling, or incorrect interpretation of the requirements in code. Without knowing the root cause up front, however, it is nearly impossible for the software maintenance teams to accurately plan and estimate when the issue will be fixed. For these reasons, the software sustainment community faces extra challenges to achieving the speed and quality espoused by DoD programs that have successfully employed agile approaches in other contexts, e.g., for new and/or standalone software platforms.

This paper will explore how agile methodologies could be implemented within the DoD sustainment process. In the following section, basic definitions of DoD Acquisition and Sustainment are provided, as well as an overview of the Agile software development process. A survey is then provided to examine the ways in which agile methods have been utilized for the development of DoD software, including some specific case studies. Key similarities and differences between the DoD programs that have demonstrated success with developing software using agile methods are explored. Also examined are the challenges other DoD programs face with the implementation of agile methods towards understanding key characteristics of successful application. An analysis of the Air Force Sustainment Center's management system, Art of the Possible (AoP) [3], is reviewed to demonstrate the challenges of integrating the DoD's agile efforts with its software sustainment operations. Through a synthesis of existing ideas in the literature, new insights are provided for how agile might be effectively applied to software sustainment programs. Finally, areas for further research and investigation are identified.

II. Basic Definitions

A. *DoD Acquisitions and Sustainment*

The term "acquisition" has multiple meanings within the DoD. Big "A" Acquisition refers to the overall lifecycle management strategy for a capability or platform [4]. This is the process DoD professionals use to evaluate the feasibility of a system that encompasses all planning, budgeting, and design activities from initial conception through system disposal [4] [5]. Little "a" acquisition more specifically refers to the phases of technical development and deployment, corresponding to the Systems Acquisitions phase shown below in Fig. 1 [5]. Furthermore, the Sustainment phase covers all activities that occur after deployment to maintain the readiness and operational capability of a system, subsystem, or component until that system reaches end-of-life [5].

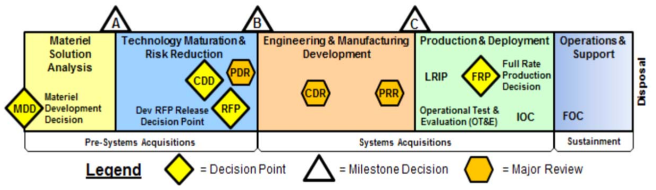
A
B
C
Materiel Solution Analysis
Technology Maturation & Risk Reduction
Engineering & Manufacturing Development
Production & Deployment
Operations & Support
Disposal
MDD
Materiel Development Decision
CDD
PDR
Dev RFP Release Decision Point
RFP
CDR
PRR
LRIP
FRP
Full Rate Production Decision
Operational Test & Evaluation (OT&E)
IOC
FOC
Pre-Systems Acquisitions
Systems Acquisitions
Sustainment
Legend
= Decision Point
= Milestone Decision
= Major Review

Fig. 1. The big "A" Acquisition process [5]

As shown in Fig. 1 [5], Sustainment is synonymous with the Operations & Support phase of the Acquisition process. This is the specific phase of software work that is referred to throughout this paper. Software sustainment is primarily concerned with addressing safety and critical operational-degrading issues such as interoperability concerns, computing resource utilization improvements, cyber security vulnerabilities, knowledge loss, version support, and obsolescence [4]. The rapid pace of software technology advances makes these issues increasingly pressing concerns for the software sustainment community.

B. *What is Agile?*

The term "agile" has become an umbrella for iterative software development practices and cultures that focus on providing value to the customer through close collaboration and feedback. Agile encompasses a variety of software development practices that, in some cases, pre-date the formal definition. Agile was first codified as a singular ideology in 2001 with the Agile Manifesto [6]. It is built around four core values - individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan [6]. These values are further supported by twelve principles [6]. The original goal of agile was to create an alternative to the traditional "waterfall" processes of software development which were, and still are, plagued by an inability to accommodate changing requirements and an over-reliance on strict adherence to a schedule and budget to prove project success [7]. Rather than relying on strict adherence to a plan that often gets developed before any of the analysis or design work has been accomplished, agile software development emphasizes rapidly responding to changing requirements and customer feedback to measure the overall success of a project. Agile, as defined by the Agile Manifesto, is first and foremost a mindset and culture. It is a system of ideas and ideals which form the basis of a software development environment [8]. To delineate itself from traditional thinking, the agile manifesto's values and supporting principles guide followers toward creating a truly people-centric environment, both for the customers and for the developers [6].

Because it is open to interpretation, several "methodologies" have since been formalized and marketed to help teams turn the agile philosophy into practice. The most widely used of these methodologies for individual teams are Scrum and various Scrum Hybrid models [9]. Scrum was created to develop and manage products using timeboxed iterations called sprints. Each sprint contains a series of formal events – sprint planning, daily scrum, sprint review, and sprint retrospective [10]. Scrum also defines specific roles; the product owner focuses on producing value for the customer, the scrum master focuses on supporting the prescribed Scrum events and helping the team understand the framework, and the development team focuses on getting the work done [10].

For scaling agile to a team-of-teams or enterprise, the Scaled Agile Framework (SAFe) is the leading methodology [9]. SAFe applies agile to large teams and businesses by drawing upon Lean tenets and systems thinking [11]. There are four different implementations of SAFe: Essential, Large Solution, Portfolio, and Full [11]. It is expected that an organization determines its implementation based on the complexity of the products or solutions they are delivering [11]. Similar to Scrum, each level of SAFe also contains dedicated roles and specific events. Rather than sprints, SAFe relies on release trains to be the anchor [11]. There are several more defined roles and artifacts that can be found within the SAFe methodology [11].

Although many refer to Scrum and SAFe as methodologies, they are classified as frameworks by their creators [10, 11]. In context, frameworks are designed to provide just enough guidance toward a preferred method of doing things without becoming so rigid they become prescriptive [12]. By contrast, methodologies are intended to address specific problems using defined methods and processes [13]. It is tempting to think that the inherent flexibility of frameworks makes them easy to follow; however, the reality is that this flexibility requires more discipline than teams or their leadership might initially assume [7].

III. Agile Within the DoD: A Survey

Discipline and strict adherence to processes are common within the defense community and culture. As an extension, standardization has been widely adopted to reduce costs and improve efficiency [14]. What this means in practice is that once a particular component, criteria, or process has been selected, it is then applied to as many situations as possible. This is true for software as well. Until very recently, the instructions and regulations governing software efforts were predicated on processes developed for the hardware domain. It was assumed, at least implicitly, that software, like hardware, was developed from set and stable requirements, procured only after these requirements were met, maintained via block updates, and upgraded by repeating this process linearly, i.e., according to the waterfall method [7]. However, unlike physical hardware artifacts, software is a constantly evolving product and what has traditionally worked well for hardware production has proved to be costly and slow when producing software for military aerospace use [7].

As operational platforms become increasingly defined by and dependent upon software, and as our operational competitive advantage has diminished, the DoD has recognized that the processes for acquiring, developing, and maintaining software-intensive systems need to change and has sponsored several reports on the subject of agile software development. In fact, the Defense Innovation Board's Software Acquisition and Practices (SWAP) study points out that the DoD has been investigating its software capabilities and requesting recommendations for improvement since 1982 [15]. Within the last decade, several reports have come out to address specific concerns and recommendations for supporting and adopting agile practices. The surveyed reports fall into two major categories: agile acquisitions and the general applicability of agile practices within the DoD environment. Seventeen of the twenty-nine reports surveyed here focus on agile acquisitions [15 - 31]. The remaining twelve more broadly address adapting agile practices to the DoD environment [7, 32 - 42]. Collectively, these twenty-nine reports generally find that: i) agile can and should be applied to DoD software programs after careful consideration of the work and its environment [18, 29, 36, 37]; ii) perceived barriers such as the formal acquisition policy, guidance, and CMMI®[1] requirements can be overcome [17, 19, 22, 25, 26]; iii) communication and culture remain barriers to successful adoption of agile [20, 32, 34, 35]; and iv) having an understanding of what needs to be done is not a barrier, rather, a fundamental challenge is a matter of putting known previous recommendations and lessons learned into action [7, 15, 24, 28. A few reports surveyed include descriptions of benefits, challenges, and lessons learned from applying an agile framework to specific case studies [16, 39, 40]. Of the case studies covered, three were for IT systems [16, 40], and one focused on the objective of meeting contractual milestone requirements outlined in acquisition policy and guidance rather than the technical details of the project [39].

The DoD has made headlines within recent years for successful agile programs. Perhaps the most notable of these is the Air Force's Kessel Run program. Kessel Run was initially focused on addressing inefficiencies within the Air Operations Center (AOC) weapons system that oversees most combat airpower in the Middle East [43]. This system is comprised of a series of applications and other computer functions that allow Airmen to coordinate the deployment of aircraft for combat operations [43]. Prior to the Kessel Run project, this coordination had taken place on whiteboards and excel spreadsheets [44, 45]. In just four months, the Kessel Run team developed and fielded a new app, JIGSAW, to replace the tanker planning piece of the operation [44, 45, 46]. The team was able to accomplish the development of a new app so quickly by rethinking how software acquisition takes place and employing an iterative, agile approach to requirements and software development [43, 44]. The Kessel Run team has since expanded and developed additional applications for the AOC [44]. A separate team called Mad Hatter is following the example set by Kessel Run, its parent unit, to address maintainer complaints with the F-35 Autonomous Logistics Information System [47].

Kessel Run may be one of the most visible agile success stories within the Air Force, but a recent report from the Government Accountability Office (GAO) found that, across the DoD, 22 major defense acquisition programs (MDAPs) and 7 major IT programs reported using agile software development methods [30]. The MDAPs surveyed cover a wide range of systems including training systems, mission systems for air, land, and sea, weapons systems, unmanned aircraft systems, battle management systems, and software for entirely new platforms [30]. The surveyed IT programs cover both business and nonbusiness services such as logistics, human resources, and information security [30].

Although there appears to be a diverse range of programs applying agile principles, there are two important factors to note here. First, almost none of the projects discussed in the case studies deal with embedded systems, defined as software components that function within, interface with, and often control larger hardware systems [15]. This would include software for weapon systems, mission systems, avionics, and battle management. Secondly, it is noted that broad representation of embedded systems exists across the agile MDAPs reported in the GAO assessment; however, these programs are in the Systems Acquisitions phase; not the Sustainment phase (refer to Fig. 1). Only two of the surveyed reports discuss how agile principles could be applied to programs in the Sustainment phase of their lifecycle. Recall that when a program enters sustainment, the product has been initially fielded – meaning requirements are fixed, architecture has been defined, the computing environment is set, and any major structural or functional changes to the baseline software will require a new acquisitions effort. Furthermore, a contractor is typically in control of the baseline software and receives support from government developers. The intersection of agile embedded systems and agile Sustainment is sparsely represented in DoD sponsored reports and is an area that desperately needs more attention.

[1] Capability Maturity Model Integration (CMMI)® is an appraisal system intended to find strengths and weaknesses within an organization's processes. https://cmmiinstitute.com/cmmi

IV. IS THE ART OF THE POSSIBLE AGILE?

The Air Force Sustainment Center (AFSC), just one of many DoD sustainment organizations, has very clear direction regarding how its workload is to be managed. This guidance is provided through a handbook known as the Art of the Possible (AoP) [3]. When we consider applying agile principles and practices to DoD sustainment programs, these circumstances present challenges that can be solved with careful consideration and application of the agile philosophy. To explore possible avenues for incorporating agile principles into sustainment operations, the AoP Handbook [3] is examined in this section. The AoP describes, generally, how the AFSC has been doing business since 2012 [48]. Historically within the Air Force, fielded platforms and systems were overwhelmingly hardware-defined. Furthermore, and as stated previously, process standardization is a highly valued and common practice within the DoD. Increasingly, as fielded platforms evolve and are modernized to include software-enabled functionality, the AFSC maintenance groups are tasked with not only maintaining mechanical subsystems such as engines and landing gear, but also new and evolving software-centric subsystems. Sustainment processes that were originally developed in response to maintenance requirements associated with hardware-centric subsystems, e.g., engines, have become the de facto standard for software-centric subsystems as well. The AoP is a management strategy predicated on the ideas that AFSC processes are: “machines” with predictable results; that these machines apply to all sustainment center activities; and that speed, cost reduction, and quality result from tight and disciplined control of these process machines [3].

Like the Agile Manifesto, the AoP is centered around a set of core tenets: leadership model; radiator chart; establish flow; identify work in progress (WIP); identify constraints; and resolve constraints [3]. The leadership model focuses on developing strong leaders and skilled, motivated employees to perform the work. The radiator chart is a visual tool used to convey the intricacies of how all the tenets of AoP come together. Establishing flow and identifying WIP speak to how the AFSC efficiently manages workload. Identifying and resolving constraints describe the importance of finding process inefficiencies and working together to remove them. The main ideas behind the AoP tenets can be directly linked to agile values and principles. The leadership model aligns with the Agile Manifesto’s value of individuals and interactions and principles of building projects around motivated individuals and self-organizing teams [6]. The agile value of producing working software and the principle of delivering that software frequently require that work move through the process as efficiently as possible so that functional software can be rapidly released [6]. Addressing constraints is part of the agile principle of regularly reflecting on how to become more efficient and adjusting team behavior accordingly. While the radiator chart does not inherently map to any one Agile value or principle, it does provide a succinct all-encompassing representation of AoP; similar to how the Agile Manifesto defines agile in a concise manner.

Even though the main themes of the AoP accommodate the Agile Manifesto, there are a few points that present challenges to employing agile principles and methods. First, agile strictly targets software development while the AoP aims to cover all manner of AFSC processes. This is not insurmountable, but rather indicates a difference in scope and intended applicability. Additionally, agile teams are driven by customer needs and product requirements whereas AoP rests heavily on organizational leadership to move the enterprise forward. When taken literally, the idea of process “machines” that produce predictable results is in direct conflict with the agile principle of embracing change. Again, this can be resolved by careful consideration of scope. Agile can accommodate change within the software development process whereas AoP can be used to stabilize the surrounding processes.

Finally, a major component of completing any work within the DoD is periodically reporting on the status of that work in terms of schedule and cost performance. Both the Agile Manifesto and AoP advocate for transparent, visual representations of project status but they rely on different metrics to convey this [3, 6]. The stated goals of AoP are to complete work faster and cheaper, i.e., according to schedule and cost metrics, without sacrificing quality [3]. In contrast, both the Scrum and SAFe frameworks emphasize that the primary metrics of success are customer satisfaction and quality [10]. Perhaps one of the more entrenched difficulties in reconciling the Agile Manifesto and AoP are the nuanced realities of culture and environment within which they are utilized. Reciting succinct core values, principles, and tenets can result in an oversimplification of how both ideologies are actually realized in practice.

V. LINKING AGILE AND SUSTAINMENT

Particularly within software sustainment, a creative interpretation of management strategies such as the AoP must be utilized to successfully implement its values as much of the language refers to hardware and business processes. A prime sticking point in many hardware-turned-software processes is the idea that stability leads to predictability. Predictability to the DoD sustainment community traditionally means knowing the expected cost and schedule of a process prior to initiating any work through that process. An agile proponent would argue that each software change or additional function is unique and therefore cannot be predictably scheduled or budgeted. Resolving this discrepancy comes down to a matter of perception and the scale to which it is being applied. Each software change has unique requirements and specific design considerations. It is not feasible to predict the time and cost of each specific change. However, the process utilized in completing each change is indeed the same. Thus, over time, accurate average schedule and cost estimates (for a number of changes) could be produced while also having high confidence in the quality of the end product. After all, a process ensures consistency in the work performed, not uniformity of the products produced.

When incorporating agile into DoD software sustainment, careful consideration of the scope and scale at which it is applied is important. AoP principles can be used at a macro scale to predict the cost, schedule, and quality of a software release containing several individual changes. This would encompass mostly workflow at the enterprise level. These principles should not be used to make the same predictions at the micro scale of individual software modifications or to predict the exact set of changes to be included into the overall software release. Workflow processes at the micro level are predominantly owned by the development team. Agile principles and metrics should be utilized for predictions at the micro level and define the change contents of the release.

Over time and with close monitoring of the agile software development process, constraints at each level will become visible and can be addressed while maintaining alignment with both AoP and agile principles. These constraints will appear differently when utilizing agile frameworks and tools compared to using AoP practices. Teams operating under AoP may simply calculate the amount of work in progress and associated flow days to show a process constraint; a traditional numerical representation of the constraint. Teams using agile frameworks and tools may present a cumulative flow diagram to show constraints; a graphical representation of the constraint. It is important to utilize appropriate methods for workflow predictions and the proper set of principles at both the micro and macro scale. This will be unique for each organization and dependent upon the type of software work being performed. It is equally vital that clear communication of how the micro scale agile workflow processes relate to traditional processes used at the macro level. Enterprise leadership needs to understand and embrace how the development team's agile measures influence the overall status of the release. In turn, development teams need to understand and appreciate how traditional metrics can be impacted by agile workflow measures. Not only will this communication facilitate identifying proper, relevant data to use for cost and schedule estimates, it will also provide a full picture of the health of the software effort at all levels. As mentioned previously, changing the culture is a frequently encountered barrier to utilizing agile principles within the DoD. When the entire software sustainment organization collectively understands how to relate and translate agile principles and metrics to traditional principles and metrics, a common language and understanding can then be established and used to initiate a shift in culture.

To bolster a culture shift, lessons from past studies and from the acquisition community should be used, where applicable, within software sustainment organizations. Sustaining software is a perpetual development process. Each software upgrade or release must go through many of the same processes as new acquisitions programs: technology readiness reviews, developmental and operational testing, and field certification to name a few. Therefore, much of the information gathered for the acquisition program offices on how to incorporate agile into these requirements should be shared with, and taught to, the sustainment program offices. Software developers working for a sustainment organization must convey the importance of modernizing legacy software for use with modern tools and infrastructure to the program office and to the end user. Without upgrading the development environment, much of the limitations in speed reported by the teams will be due to the inefficiencies inherent in using legacy software development tools and environments. This is even more challenging given that not all sustainment organizations are co-located with their program offices or end users. In these cases, it is vital that sustainment program offices and program end users dedicate personnel to interface regularly with the development teams. Organizations that support the software sustainment effort need to buy-in to agile for it to succeed as the new way of producing software.

Furthermore, in organizations as large as the Air Force or DoD, it is imperative that those involved in making the transition to agile software development recognize that it is the philosophy behind agile that must spread to all levels of the organization, whereas agile practices should be utilized only for the teams and levels which require them. The Agile Manifesto is clear about the driving ideas behind agile frameworks like Scrum and SAFe. These frameworks can be tailored so long as the motivations behind them remain aligned with the values and principles of the Agile Manifesto. As many of the regulations and policies for sustainment organizations like the AFSC often stem from wider Air Force or DoD guidance, flexibility with specific agile frameworks and practices is key.

VI. Summary and Future Work

This paper is an initial, and by no means exhaustive, effort to consolidate information published on the subject of incorporating agile methodologies into the DoD software sustainment environment. For this reason, the reports surveyed are either sponsored by the DoD or relate specifically to DoD programs reportedly using agile methodologies. Additional academic sources and studies on the subject need to be explored further. Additionally, this paper draws heavily upon Air Force programs and processes as an insight into DoD sustainment operations. An examination of sustainment organizations within other DoD branches should be pursued. Software sustainment as it relates to sectors such as the automotive industry or health care systems may yet provide additional insights and recommendations that can be adopted for applying agile to DoD software sustainment and should be researched further. Lastly, this paper has demonstrated a lack of published information on the application of agile processes to DoD *software sustainment of embedded systems*. Surveys, interviews, and case studies of specific DoD software sustainment organizations and programs that are transitioning to agile software development processes should be carried out in the future. An extension of this should be to study potential challenges facing an agile software program as it shifts from the Systems Acquisition phase to the Sustainment phase.

References

[1] A. Christopherson, "Faster, smarter: Speed is key in acquisition reform," Air Force News Service, 28 February 2019. [Online]. Available: https://www.af.mil/News/Article-Display/Article/1771387/faster-smarter-speed-is-key-in-acquisition-reform/. [Accessed 6 May 2020].

[2] P. Berube, "SecAF Wilson urges innovation in software acquisition model," Air University Public Affairs, 6 September 2018. [Online]. Available: https://www.af.mil/News/Article-Display/Article/1622568/secaf-wilson-urges-innovation-in-software-acquisition-model/. [Accessed 6 May 2020].

[3] "Art of the Possible Handbook," Air Force Sustainment Center: Tinker Air Force Base, OK, 2018.

[4] *Defense Acquisition Guidebook*, Defense Acquisition University, [Online]. Available: https://www.dau.edu/tools/dag. [Accessed 12 May 2020].

[5] *Acquisition Process*, AcqNotes. [Online]. Available: http://acqnotes.com/acqnote/acquisitions/acquisition-process-overview#:~:text=The%20Department%20of%20Defense%20(DoD,of%20the%20Adaptive%20Acquisition%20Framework%E2%80%9D.. [Accessed 8 July 2020].

[6] *Manifesto for Agile Software Development*, Agile Manifesto, 2001. [Online]. Available: https://agilemanifesto.org/. [Accessed 5 May 2020].

[7] M. S. Palmquist, M. A. Lapham, S. Miller, T. Chick and I. Ozkaya, "Parallel Worlds: Agile and Waterfall Differences and Similarities," Software Engineering Institute: Pittsburgh, PA, Rep. CMU/SEI-2013-TN-021, 2013.

[8] "Ideology," Merriam-Webster, [Online]. Available: https://www.merriam-webster.com/dictionary/ideology. [Accessed 18 May 2020].

[9] "13th Annual State of Agile Survey," CollabNet VersionOne, [Online]. Available: https://www.stateofagile.com/#ufh-c-473508-state-of-agile-report. [Accessed 5 May 2020].

[10] K. Schwaber, "Scrum.org: The Home of Scrum," Scrum.org, [Online]. Available: https://www.scrum.org/. [Accessed 20 May 2020].

[11] D. Leffingwell, D. Jemilo, I. Oren, L. Hohmann, R. Knaster, H. Koehnemann, S. Mayner, A. Sales and E. Willeke, "SAFe," Scaled Agile, Inc, [Online]. Available: https://www.scaledagileframework.com/. [Accessed 20 May 2020].

[12] "Framework," Merriam-Webster, [Online]. Available: https://www.merriam-webster.com/dictionary/framework. [Accessed 22 May 2020].

[13] "Methodology," Merriam-Webster, [Online]. Available: https://www.merriam-webster.com/dictionary/methodology. [Accessed 22 May 2020].

[14] *Defense Standardization Program Mission and Vision Statement*, Defense Statndardization Program, [Online]. Available: https://www.dsp.dla.mil/Home/Mission/. [Accessed 8 June 2020].

[15] J. M. McQuade, R. M. Murray, G. Louie, M. Medin, J. Pahika and T. Stephens, "Software Is Never Done: Refactoring the Acquisition Code for Competitive Advantage," Defense Innovation Board, 2019.

[16] C. Northern, K. Mayfield, R. Benito and M. Casagni, "Handbook for Implementing Agile in Department of Defense Information Technology Acquisition," The MITRE Corporation: McLean, VA, Rep. 11-0401, 2010.

[17] M. A. Lapham, S. Miller, L. Adams, N. Brown, B. Hackemack, C. Hammons, L. Levine and A. Schenker, "Agile Methods: Selected DoD Management and Acquisition Concerns," Software Engineering Institute: Pittsburgh, PA, Rep. CMU/SEI-2011-TN-002, 2011.

[18] P. Modigliani and S. Change, "Defense Agile Acquisition Guide: Tailoring DoD IT Acquisition Program Structures and Processes to Rapidly Deliver Capabilities," The MITRE Corporation: Bedford, MA, Rep. 14-0391, 2014.

[19] M. A. Lapham, M. Bandor and E. Wrubel, "Agile Methods and Request for Change (RFC): Observations from DoD Acquisition Programs," Software Engineering Institute: Pittsburgh, PA, Rep. CMU/SEI-2013-TN-031, 2014.

[20] K. E. Nidiffer, S. M. Muller and D. Carney, "Potential Use of Agile Methods in Selected DoD Acquisitions: Requirements Development and Management," Software Engineering Institute: Pittsburgh, PA, Rep. CMU/SEI-2013-TN-006, 2014.

[21] R. Cagel, M. Kristan and T. Rice, "DevOps for Federal Acquisition," The MITRE Corporation: Bedford, MA, Rep. 15-2842, 2015.

[22] E. Wrubel and J. Gross, "Contracting for Agile Software Development in the Department of Defense: An Introduction," Software Engineering Institute: Pittsburgh, PA, Rep. CMU/SEI-2015-TN-006, 2015.

[23] A. Pinto, M. E. Liggan, N. K. Subowo, H. G. Goodwin, S. Sekhavat-Tafti and A. M. Staley, "Federal Aviation Administration Agile Acquisition Principles and Practices," The MITRE Corporation: Bedford, MA, Rep. 16-1231, 2016.

[24] S. Miller, D. Ward, M. A. Lapham, R. Williams, C. (. Hammons, D. Burton and A. Schenker, "Update 2016: Considerations for Using Agile in DoD Acquisition," Software Engineering Institute: Pittsburgh, PA, Rep. CMU/SEI-2016-TN-001, 2016.

[25] M. A. Lapham, L. A. Rosser, S. Martin, T. E. Friend, P. Capell, K. Korzec, G. Howard, M. Ryan and J. H. Norton III, "RFP Patterns and Techniques for Successful Agile Contracting," Software Engineering Institute: Pittsburgh, PA, Rep. CMU/SEI-2016-SR-025, 2016.

[26] L. Adams, "Agile Software Development in the Department of Defense Environment," The Defense Acquisition University: Aberdeen Proving Ground, MD, 2017.

[27] M. McCollum, "Creating Software Systems That Keep Pace With Aviation Change," Mar. 2017, [Online]. Available: https://www.mitre.org/publications/project-stories/creating-software-systems-that-keep-pace-with-aviation-change. [Accessed 28 June 2020].

[28] "Design and Acquisition of Software for Defense Systems," Defense Science Board: 2018.

[29] S. Roe, R. Novak, P. R. J. Staresina, A. Bouffard, J. Collens, K. Gantt, K. Horinek, M. Reigler and K. Stauffer, "Driving Impact Through Acquisition Flexibility," The MITRE Corporation: Bedford, MA, Rep. 19-3551, 2020.

[30] "Defense Acquisitions Annual Assessment," Government Accountability Office: Washington, DC, Rep. 20-439, 2020.

[31] S. W. Roe, R. Novak, J. Raines, P. Staresina, M. Zaharee, J. Kattman and M. Lee, "Challenge-Based Acquisition, 5th Ed.," The MITRE Corporation: McLean, VA, Rep. 20-0745, 2020.

[32] M. Casagni, M. Heeren, D. Hanf, M. Kristan and S. Kuwana, "2016 Federal DevOps Computing Summit Report," The MITRE Corporation: Bedford, MA, Rep. 16-4460, 2016.

[33] H. Reed, J. Nankervis, L. J. Cochran, R. Parekh, F. Stein, R. (. Benito, C. Magrin, D. Hanf and M. Casagni, "ICCRTS 2014 Agile and Adaptive IT Ecosystem, Results, Outlook, and Recommendations," The MITRE Corporation, Bedford, MA, Rep. 14-1857, 2014.

[34] T. J. Mueller, "Critical Decision Factors for Agile on Federal IT Projects," The MITRE Corporation: McLean, VA, Rep. 15-3449, 2015.

[35] T. Mueller, D. Harvey, A. Sheikh and S. Johnson, "Making Agile Work in Government," The MITRE Corporation: McLean,VA, Rep. 15-0045, 2015.

[36] W. Hayes, S. Miller, M. A. Lapham, E. Wrubel and T. Chick, "Agile Metrics: Progress Monitoring for Agile Contractors," Software Engineering Institute: Pittsburgh, PA, Rep. CMU/SEI-2013-TN-029, 2014.

[37] W. Hayes, M. A. Lapham, S. W. E. Miller and P. Capell, "Scaling Agile Methods for Department of Defense Programs," Software Engineering Institute: Pittsburgh, PA, Rep. CMU/SEI-2016-TN-005, 2016.

[38] C. Regan, M. A. Lapham, E. Wrubel, S. Beck and M. Bandor, "Agile Methods in Air Force Sustainment: Status and Outlook," Software Engineering Institute, Pittsburgh, PA, Rep. CMU/SEI-2014-TN-009, 2014.

[39] E. Wrubel, S. Miller, M. A. Lapham and T. Chick, "Agile Software Teams: How They Engage with Systems Engineering on DoD Acquisition Programs," Software Engineering Institute: Pittsburgh, PA, Rep. CMU/SEI-2014-TN-013, 2014.

[40] N. Brown, J. Davenport, L. P. Gates and T. Marshall-Keim, "FedCLASS: A Case Study of Agile and Lean Practices in the Federal Government," Software Engineering Institute: Pittsburgh, PA, Rep. CMU/SEI-2018-SR-016, 2018.

[41] "Software Development: Effective Practices and Federal Challenges in Applying Agile Methods," Government Accountability Office: Washington, DC, Rep. 12-681, 2012.

[42] S. Bellomo and C. Woody, "DoD Information Assurance and Agile: Challenges and Recommendations Gathered Through Interviews with Agile Program Managers and DoD Accreditation Reviewers," Software Engineering Institute: Pittsburgh, PA, Rep. CMU/SEI-2012-TN-024, 2012.

[43] B. Newell, "Kessel Run hits hyperdrive," 10 May 2018. [Online]. Available: https://www.hanscom.af.mil/News/Article-

Display/Article/1517830/kessel-run-hits-hyperdrive/. [Accessed 22 June 2020].

[44] M. Wallace, "The U.S. Air Force learned to code—and saved the Pentagon millions," 5 July 2018. [Online]. Available: https://www.fastcompany.com/40588729/the-air-force-learned-to-code-and-saved-the-pentagon-millions. [Accessed 22 June 2020].

[45] H. Bray, "Air Force lab takes aim at Han Solo's record," 11 May 2018. [Online]. Available: https://www3.bostonglobe.com/business/2018/05/11/air-force-lab-takes-aim-han-solo-record/MLDf23GSy1ZgvmSVEXF09I/story.html?arc404=true. [Accessed 22 June 2020].

[46] *AOC Pathfinder is Saving USAF Big Money, And It Wants More of It*, Air Force Magazine, 20 February 2018. [Online]. Available: https://www.airforcemag.com/AOC-Pathfinder-is-Saving-USAF-Big-Money-And-It-Wants-More-of-It/. [Accessed 22 June 2020].

[47] B. Newell, "Hanscom AFB software teams decode F-35 maintenance," 14 May 2019. [Online]. Available: https://www.af.mil/News/Article-Display/Article/1847580/hanscom-afb-software-teams-decode-f-35-maintenance/. [Accessed 23 June 2020].

[48] H. Logan-Arrington, "Art of the Possible involves everyone in the Air Force," 25 Septermber 2018. [Online]. Available: https://www.afmc.af.mil/News/Article-Display/Article/1644592/art-of-the-possible-involves-everyone-in-the-air-force/. [Accessed 23 May 2020].