How Agile Development Can Transform Defense IT Acquisition

Su J. Chang, Angelo Messina and Peter Modigliani

Abstract Traditional defense acquisition frameworks are often too large, complex, and slow to acquire information technology (IT) capabilities effectively. Defense acquisition organizations for years have been concerned about the lengthy IT development timelines and given the pace of change in operations and technology, it is critical to look for new strategies to acquire IT for defense systems. Over the last decade, agile software development emerged as a leading model across industry with growing adoption and success. Agile is centered on small development Teams delivering small, frequent releases of capabilities, with active user involvement. From a planning and execution viewpoint, agile emphasizes an iterative approach with each iteration informing the next. The focus is less on extensive upfront planning for entire programs and more on responsiveness to internal and external changes, such as operations, technology, and budgets. Based on US and Italian experiences, this paper discusses some of the common challenges in implementing agile practices and recommended solutions to overcome these barriers.

Angelo Messina: Deputy Chief Italian Army General Staff Logistic Department and DSSEA Secretary.

S.J. Chang (✉) · A. Messina · P. Modigliani
The MITRE Corporation, Virginia, USA
e-mail: sjchang@mitre.org

A. Messina
e-mail: angelo.messina@esercito.difesa.it

P. Modigliani
e-mail: pmodigliani@mitre.org

P. Ciancarini et al. (eds.), *Proceedings of 4th International Conference in Software Engineering for Defence Applications*, Advances in Intelligent Systems and Computing 422, DOI 10.1007/978-3-319-27896-4_2

1 Introduction

Agile software development practices integrate planning, design, development, and testing into an iterative lifecycle to deliver software at frequent intervals. Structuring programs and processes around small, frequent agile releases enable responsiveness to changes in operations, technologies, and budgets. These frequent iterations effectively measure progress, reduce technical and programmatic risk, and respond to feedback and changes more quickly than traditional waterfall methods.

While the commercial sector has broadly adopted agile development to rapidly and dynamically deliver software capability, agile has just begun to take root across many defense organizations.[1,2]

Agile development can be distilled into four core elements:

- Focusing on small, frequent capability releases;
- Valuing working software over comprehensive documentation;
- Responding rapidly to changes in operations, technology, and budgets; and
- Active users involved throughout the development process to ensure high operational value.

The foundation of agile is a culture of small, dynamic, empowered Teams actively collaborating with stakeholders throughout product development. Agile development requires Team members to follow disciplined processes that require training, guidance, and openness to change.[3] While agile does impose some rigor, the method does not consist of simply following a set of prescribed processes, but instead allows dynamic, tailored, and rapidly evolving approaches that suit each organization's IT environment.

The first section of this paper will discuss the Italian Army (ITA) experience with agile implementation and how they have managed the hurdles related to process and culture. The second section of the paper will discuss agile in the US DoD acquisition environment, the prerequisites for agile adoption in the USA, and the barriers as it relates to program structure, requirements, and contracting. Lastly, the paper will summarize with a comparison of both the ITA and DoD experiences with agile adoption.

2 Agile in the Italian Army Environment

The lessons identified and learned for the last ten years of military operations in the area of the Command and Control software have clearly shown the limits of the software engineering traditional approach. The rapidly changing scenario of every

[1]Lapham et al. [10].

[2]Northern et al. [15].

[3]US Government Accountability Office [8].

operation has pointed out how difficult it is to define a stable and consolidated user requirement for the C4I software applications. The ITA General Staff call it as "volatility of the user requirement." On the other end, the software products to be used in the military applications are traditionally developed in strict adherence to software quality standards to limit the risk produced by errors, faults, and malfunctions.

At the ITA General Staff, it was decided to try to overcome the problem of the requirement volatility by substituting the physical document with a "Functional-technological Demonstrator" a software prototype to be used by the military user to describe his needs in practical terms by interacting with the prototype. The "agile" methods seemed to be best candidates to this particular kind of production. The SCRUM declination of agile was the selected approach because of the short and fixed length of the "Sprint" production cycles and the clear definition of the roles.

When the ITA Team was at the point of starting this new approach for the first time, there was a wide spread awareness in the military community of the risks connected to the agile methods: Scrum, XP, Kanban, etc. Nonetheless, the need for a paramount change was such to be willing to take the risk. This is the first step every organization must face: taking the risk but managing it deliberately and aggressively.

At the time the first Team was started, agile scrum had been around as long as almost 10 years and though there were many success stories, there are as many that can also be characterized as a failure. As a matter of fact, when the first Team finished the first Sprint and there was a "real" delivery, it was quite a surprise. The people in charge of the effort where quite ready to watch a couple of "warm up pseudo-sprints" but to their surprise, all worked pretty well since the very beginning.

In the initial period, a technical investigation or a comparative analysis with other similar efforts was out of the scope of the initiative. Later on, the need to come to a thorough understanding of why this Scrum-derived method worked so well in the Army Logistic environment became a real need. After the first release of the software realized with this new approach in a seven-month period, the need to understand the reasons of such a good performance was no longer a methodology issue or a software engineering curiosity but a real necessary step in the process of consolidating a production procedure.

After a year in the process, the ITA has seven agile Teams in place and working. The full "Scrum of Scrum" complex production line is active and synchronized. The Teams turned on progressively taking charge of the full architecture of an overarching strategic Command and Control tool that implements full spectrum of the decision support activity, stretching from the facilities management to the operational Field Application Services (FAS).

The project was named LC2EVO and is now working on a 5-week "Sprints" delivery cycle.

3 Italian Army Culture and Processes

Effective agile adoption requires a new culture and set of processes that may be radically different from those in many defense organizations today. Agile practices, processes, and culture often run counter to those in the long-established defense organizations. The agile model represents a change in the way the government conducts business, and programs must rethink how they are staffed, organized, and managed, as well as whether the business processes, governance reviews, funding models, that support an acquisition are structured to support agile. The following eight reasons are why early adopters in ITA Army defense organizations are succeeding.

1. **Trust in people**. The empowerment of the Team should be deeply enforced from the beginning. In the case of the ITA, it was not easy nor simple, but the first step in shaping the agile effort was to value individual skills and commitments. A scrum tool was used for this purpose: the use of a scrum dashboard with "post-its" and all the relevant connected "ritual" processes was used by the ITA. The scrum "Champion" and his co-workers took full responsibility for the result of the effort but implemented a zero-tolerance acceptance of all the scrum elements the Team accepted to implement. In particular, some psychological side effects were actively pursued. In the case of the scrum dashboard for example, the electronic equivalent of the "post-it" user story was implemented in parallel to ensure traceability and documentation, but it never substituted the physical act of drawing the "post-it" with the programmer name and sticking it on the "to do" area of the dashboard. The value of this simple (public) ritual has deep implications to gain personal commitment.
2. **People do their best if given enough freedom**. An integrated development Team with subject-matter experts (SMEs) working together with both contractor's and government analysts and programmers is the optimal Team mix. The pressure generated by the direct relationship with the product owner (PO) is useful to inspire strong work commitments and create positive tension. The SMEs benefit from actually witnessing "their" functionality take shape and become working software; and the programmers take pride in their ability to translate user requirements into software functionality. When the right level of performance or quality is not being achieved, there is shared accountability for the success or failure of the results that force the Team to work together on common solutions.
3. **No project management on top of Scrum Teams**. According to the ITA agile methodology, putting program management on top of a "Scrum Team" is a conceptual mistake. Sprints are timeboxed and the only date available for public release is the end of the Sprint. Any question such as "are you able to finish by xx xx?" has to be rephrased as "How many stories are you able to finish within the time box of the next Sprint?" This is probably the toughest issue to solve with any legacy organization. Flowcharts, pert charts, and time-marked programs are incompatible with this vision of agile scrum.

4. **Scrum does not improve software quality, capable people do**. Having low-skilled personnel on the Team is not acceptable. For some professionals (security, data architecture, software architecture, and few others) top-level expertise is required. Sometimes, teamwork and a good Scrum Master can work out situations where there is a lack of specific expertise. In such cases, "technical gap filling training" stories have to be added to the product backlog and their priority level has to be acknowledged by the product owner/stake holder.
5. **An agile Team continuously improves**. That is why Scrum has retrospectives to see what went well, what can be improved, and to define actions. People have a natural tendency to seek improvement. In the ITA agile "scrum" environment, improvement is almost automatic: It comes out by the continuous confrontation process in the Team. Each Team member learns from the others especially from the non-programmers.
   Resistance to change is one of the major issues to overcome when implementing any agile production Team. Resistance to the new way is almost as natural as the tendency to seek continuous improvement. In the LC2EVO implementation, not much of this problem was found among the analysts and programmers, but it could have been an issue among the management. As stated before, no traditional program management of any kind was allowed in the ITA Army agile effort.
6. **The product owner role**. The PO is never alone. A product ownership board is recommended that it includes stake holder representatives with decision capability. A PO has to continuously work with the Team to ensure the "nominal" scrum functions are met and give the Team a precise point of reference. On the other end, in the ITA vision, a PO has to keep close contact with the stake holder to refine the user vision of the evolving "requirements" and make sure the stake holder's expectations are met. The adoption of a PO board ensures a continuous link with both sides of the community: the stake holders and the Team. Being part of the Team (at least with one member), the PO board is fully aware of the level of quality of the developed software. In the domain of "mission-critical" applications, the quality and then the security and reliability of the product are specified by the definition of high priority user stories (written by the relevant experts on the Team) and the implementation of such stories cannot be different. It is true that the stake holder side of the PO board will try to go back to the old way "Just deliver those features as fast as possible" and will try to transform the Sprint delivery date in a deadline but this can be avoided with the continuous watch by the POs and Scrum Master on the correctness of the approach.
7. **Product quality**. A controlled environment where the use of coding standards was the rule was implemented by the ITA Army. To avoid "cowboy coding," the ITA Army agile environment includes a layer of software tools which give in real time the quality level of the produced code. This environment is, first of all, a support to the programmers to understand how well they are performing. In the event the level of quality decreases, they have the possibility to insert "knowledge acquisition" stories in the product backlog to fill "technical or cultural" gaps.

8. **Tailored organization**. As stated before, no project management on the stakeholder side is accepted as the "fee" to pay in exchange of the possibility of modifying the product backlog after any of the Sprints. Any negotiation is carried on in terms of features (user stories to be deleted, inserted or changed in priority) to be delivered in the next Sprint(s).

The matter of the contractual implementation of the agile framework is far from being solved and many groups are working on this issue. In the ITA Army agile effort [3] to produce LC2EVO, a relevant decrease in the total cost of the produced software was experienced. Detailed investigation on the technical details of this decrease is still ongoing, but at first glance, it is reasonable to say that the main cost reduction drivers were the following:

- Only the required documents were elaborated, no one was tasked to write useless pieces of paper that nobody was going to read just because of a standard.
- The focus by the Team on the most desired features ends up in a "natural" code density in the areas most valued by the customer producing some kind of multiplying factor (still under investigation).
- User stories with no business value are quickly dropped out and canceled from the product backlog. In this area, the quick delivery feature of the agile methods helps to stimulate the customer focus on what he really wants which may not be so clear at the beginning of the project.

4 Agile in the DoD Environment

Despite the success that agile development has achieved in the private sector, commercial implementation of agile does not directly translate to agile adoption in the DoD environment. The barriers to program structure, requirements, contracting, and culture and processes often stem from these key differences. First, the government must adhere to a set of rigorous policies, statutes, and regulations that do not apply to the same degree to the commercial sector.[4] Following the rules that govern federal acquisition often involves a bureaucratic, laborious, and slow process that greatly influences how effectively DoD can implement agile. Second, the commercial sector has a different stakeholder management process than the government. Private firms are accountable to an internal and layered management structure that usually goes no higher than a corporate board of directors; the few possible external stakeholders (e.g., labor unions) rarely cause frequent and major disruptions.

The government bureaucracy has layers upon layers of stakeholders with a high degree of influence that can create frequent and significant disruptions. Everything from a change in the political administration to budget sequestration can exert

[4]Lapham et al. [11].

significant external influence on a DoD program. Lastly, the bureaucratic layers of government make it difficult to empower agile Teams to the same extent as in the private sector. The commercial sector has considerable latitude to make adjustments throughout the course of the development because companies closely link accountability, authority, and responsibilities to push decision making to the lowest levels. The government's tiered management chain of command makes it difficult for the agile Team to make decisions quickly and unilaterally.

The above comparisons demonstrate the need for DoD to tailor agile processes to its unique set of policies and laws. Herein lies the fundamental issue with agile adoption in DoD. The practices, processes, and culture that have made agile development successful in the commercial sector often run counter to the current practices, processes, and culture in the long-established defense acquisition enterprise.[5] In many ways, the acquisition environment needed to execute agile development is the opposite of the acquisition environment in place today.

- The small, frequent capability releases that characterize the agile development approach directly contrast with the traditional DoD acquisition model designed for a single big-bang waterfall approach.[6] Currently, every step in the acquisition system must be extensively documented and approved prior to execution. For example, according to DoD policies, an IT acquisition program must meet 34 statutory and regulatory documentation requirements prior to beginning development,[7] whereas agile emphasizes working software over comprehensive documentation.[8]
- Agile also enables rapid response to changes in operations, technology, and budgets. By contrast, DoD requires budgets, requirements, and acquisitions to be planned up front, often several years in advance of execution, and changing requirements, budgets, and strategies during the execution process is disruptive, time-consuming, and costly.[9]
- Lastly, agile values active involvement of users throughout the development process to ensure high operational value and continuously re-prioritizes the ongoing requirement process on the basis of feedback from the user community on deployed capabilities. Today's DoD requirement process is static, and rigid and limits active user involvement and feedback during the development process.[10]

[5]Broadus (January–Feburary [2]).

[6]Ibid.

[7]Defense Acquisition University (DAU) [4].

[8]Lapham, DoD agile Adoption, [9].

[9]Modigliani and Chang (March [14]).

[10]Lapham et al. [10].

Given the above key differences, DoD has been ill prepared to adopt agile development practices and in fact agile implementations so far have not always succeeded. Some early DoD adopters attempted what they thought or promoted as "agile," yet they did not incorporate some of the foundational agile elements into their structures or strategies. This resulted partly from the lack of definition and standardized processes for agile in the federal sector. In some cases, programs implemented a few agile principles, such as breaking large requirements into smaller increments, but did not integrate users during the development process to provide insight or feedback. Other programs structured capability releases in a timeboxed manner,[11] yet did not understand what to do when releases could not be completed in time.

Adopting only a handful of agile practices without a broader agile strategy often fails to achieve desired results.[12] For example, one DoD early adopter initially attempted to implement agile practices by breaking large requirements into several 4-week Sprint cycles. However, the program lacked high-level agreement on what to develop in each cycle and did not have a robust requirements identification and planning process in place. Furthermore, the program lacked an organized user community and active user-participation throughout the development process—a fundamental agile tenet. As a result, the agile processes quickly degenerated and the program only delivered 10 % of its objective capability after two years of failed agile development attempts. The program finally retreated to a waterfall-based process. It simply could not execute the agile strategy without the proper environment, foundation, and processes in place. On the other hand, DoD has recorded some significant successes with agile, such as the Global Combat Support System–Joint (GCSS-J) program, which has routinely developed, tested, and fielded new functionality and enhanced capabilities in six-month deployments.[13]

Another successful early agile adopter had their testing leaders report "after a decade of waterfall developed releases, the first agile iteration is the most reliable, with fewest defects, than any other release or iteration." They attributed it to the frequent feedback, number of releases, continuous builds, and engineering releases with stable functionality. DoD programs are beginning to turn the corner to successfully adopt agile practices. The programs that have had the most success were ones with major capabilities already deployed and used agile practices for subsequent upgrades. The goal is to position IT programs to adopt agile from the start of the acquisition life cycle.

[11]A timebox is a fixed time period allocated to each planned activity. For example, within agile, a Sprint is often timeboxed to a 4- to 6-week-time period or a release is timeboxed for a 4- to 6-month time frame.

[12]US Government Accountability Office [8].

[13]Defense Information Systems Agency [5].

5 Prerequisites for Agile Adoption

As a starting point, defense organizations should adopt a common understanding of agile and identify the underlying set of values that describe the purpose and meaning of their agile practices. The authors propose the following guiding principles for agile adoption:

1. **Focus on small, frequent capability releases to users**—Smaller releases are easier to plan, present lower risks, and are more responsive to changes. Projects should focus on delivering working software as the primary objective.
2. **Embrace change**—Projects must allow for changes to scope and requirements based on operational priorities, user feedback, early developments, budgets, technologies, etc. This requires flexible contracts, strong collaboration, and rigorous processes. Projects should plan early and then adapt based on current conditions.
3. **Establish a partnership between the requirements, acquisition, and contractor communities**—Projects should foster active collaboration on operations, technologies, costs, designs, and solutions. This requires committed users who contribute to development, tradeoff discussions, and regular demonstrations of capabilities. One DoD program had the acquirers, contractors, and users all participating in agile training. This provided a foundational education and a forum to discuss the specific agile processes for their program.
4. **Rely on small, empowered, high-performing Teams to achieve great results**—Organizing around each release with streamlined processes and decisions enables faster deliveries that are more successful. A DoD program benefited greatly with an integrated, high-performing Team by co-locating users, acquirers, testers, requirements leads, and many developers.
5. **Leverage a portfolio structure**—Individual programs and releases can deliver capabilities faster by using portfolio or enterprise strategies, processes, architectures, resources, and contracts.

These tenets align with the recommended set of principles in the Government Accountability Office (GAO) report on "Effective Practices and Federal Challenges in Applying agile Methods."

6 Agile Requirements Process

The agile requirements process values flexibility and the ability to reprioritize requirements as a continuous activity based on user inputs and lessons learned during the development process. In contrast to current acquisition practices, the agile methodology does not force programs to establish their full scope, requirements, and design at the start, but assumes that these will change over time.

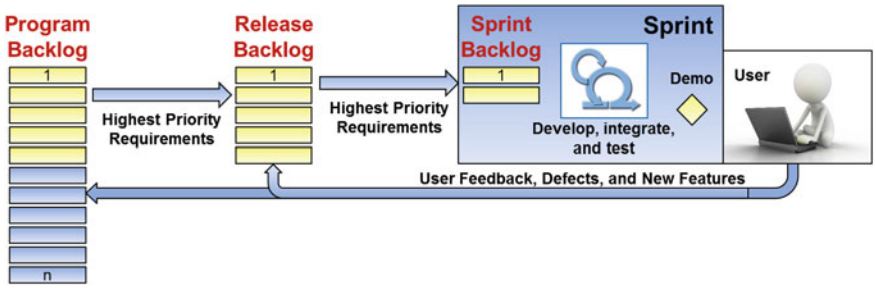

Fig. 1 Program, release, and Sprint backlogs

The program backlog contains all desired functionality and requirements. A release backlog typically comprises the highest priority requirements from a program backlog that a Team can complete within the established time frame. A Sprint then addresses the highest priority requirements from the release backlog. Once the development Team commits to the scope of work for a Sprint, that scope is locked. Sprint demonstrations conducted by the contractor at the end of a Sprint may identify new features or defects that the Team would add to the release or program backlogs. Figure 1 shows the relationships among the program, release, and Sprint backlogs.

One DoD program developed half-page work packages for the program backlog, each with a rough government estimate, some design context, and technical dependencies/interfaces. Users prioritized the work packages from an operational perspective and aligned epics or themes to mission threads. Each spiral had roughly 60–70 work packages.

The product owner, actively collaborating with users and stakeholders, is responsible for grooming the backlog to ensure the content and priorities remain current as Teams receive feedback and learn more from developments and external factors. DoD programs found having a single or multiple product owners was program dependent and found success with both models. Programs should consider the size and diversity of the user base and the operational stakeholders' empowered representatives. Users and development Teams may add requirements to the program or release backlog or shift requirements between them. The release and development Teams advise the PO on the development impacts of these decisions, while users advise the release Team about the operational priorities and impacts. To address a specific user story, the program must understand dependencies on existing or planned capabilities. Some programs may turn to a change control board to make some of the larger backlog-grooming decisions. The use of this requirements process can help set a DoD agile acquisition program on the right path for implementation.

As noted in the earlier example, the ITA stated that "volatility of the user requirement" was a considerable challenge in implementing agile. They substituted the physical document with a "functional-technological demonstrator." Much like

the Italians, several DoD programs have also adopted similar approaches, using databases, Excel spreadsheets, or agile-based software tools to track changes to user requirements. The agile manifesto emphasizes "working software over comprehensive documentation." Plans should be developed by the Team, for the Team, to provide some level of consistency and rigor, but the level of documentation should not impede the Team's ability to focus on capability delivery.[14] The key is to do "just enough" documentation that describes high-level requirements to satisfy the institutionalized requirements processes, but using another type of requirements tool and process to track the detailed user-level requirements that are subject to frequent changes and reprioritizations.

7 Structuring and Tailoring for Agile

To effectively integrate agile principles into a rigid, bureaucratic defense environment one must proactively tailor program structures and acquisition processes. Agile experts working closely with executives and process owners across each functional area can collaboratively design an agile, new framework for acquiring IT capabilities. The foundational element is the structure of the program's releases and redesigning the key acquisition processes to support the release time frames. While many of DoD's large IT systems are structured in five year increments, they have historically taken over eight years to deliver IT capabilities. Programs should structure releases to deliver capabilities in less than 18 months with a goal of 6–12 months. The smaller scoped releases reduce risk and improve chances of successful deliveries. Small, frequent releases enable the program to be responsive to changes in operations, technologies, and budgets.

Many of DoD's early agile adopters have designed releases 6–18 months long further subdivided by Sprints. Those programs that had 18-month releases believe an 8- to 12-month release structure would have been better suited to their environment. During release planning prior to development, they finalize their requirements and designs for the scope of the release. Monthly Sprints provided working software that address the highest priority release requirements first and deliver to the government. Some programs required interactive demonstrations with users every two Sprints, while others required integration into a government testing and demonstration environment at the release midpoint. Many reserved the final Sprint to not have planned features, but instead finalize any incomplete features and rework based on testing and feedback of the previous Sprints.

[14]Modigliani and Chang (March [14]).

8 Contracting for Agile Development

Contracting for agile development has proven tremendously difficult for US and Italian governments. While commercial firms often rely on in-house staff to execute the agile practices, the government must obtain agile development through contracted support. Long contracting timelines and costly change requests have become major hurdles in executing agile developments and enabling small, frequent releases. Contracting strategies for agile programs must be designed to support the short development and delivery timelines that IT requires.[15]

Complex laws and regulations often drive long contracting timelines, defined requirements upfront, rigid government contractor interaction, and contractor selection based on the strength of technical proposal. These runs counter to agile's tenets of short delivery cycles, dynamic requirements, close Team coordination, and using an expert Team with demonstrated success.

To overcome many of these challenges, programs should use a service contract instead of a product contract. A service contract provides the program with greater flexibility to modify requirements along the development process, because it describes the people and time required to execute the development process rather than locking-down the technical details of the end-product deliverable. However, this strategy assumes the government is the lead systems integrator and is responsible for overall product rollout and delivery. If the government expects the contractor to act as the systems integrator, determine the release schedule, and be accountable for overall product delivery, then a product-based contract in which the government describes overall delivery outcomes and objectives is more practical. However, this scenario would make it difficult for the government to execute a true agile process, because changes to requirements in the course of development, or a change to the delivery schedule, will require a contract negotiation that could affect the agile process.

Lastly, the program should focus on the competition strategy to be used for the initial award as well as for follow-on task orders and awards. This will help determine how to scope the contract or task order for each contract action. In some cases, the program would benefit from bundling a set of releases into a single contract action to minimize the number of contract activities during the development process. However, the program should balance this against the need to maintain continuous competition throughout the program life cycle to keep rates low and receive the best value for products and services.

One DoD program used many contractors, each having their own agile processes, epics, stories, story point system, etc. This required the DoD to learn and operate via multiple approaches and proved to be a challenge for requirements traceability. This is a trade-off between competition and developer autonomy versus government integration and management complexity.

[15]See Footnote 14.

9 Summary

ITA and DoD have experienced many similar challenges with agile adoption; however, both governments have also witnessed the potential for great success using this development methodology. Both encountered similar challenges with introducing agile methods [16] because they are so different from long-held traditional development methods. The culture and processes are the biggest obstacles to overcome, especially as it relates to a requirements community that is characterized as dynamic and volatile. However, despite these challenges, both countries were able to break from tradition and in many cases, fully embrace the agile methodology to deliver working software in short capability drops.

The focus on iterative development and frequent capability deployments makes agile an attractive option for many defense acquisition programs, especially time-sensitive and mission-critical systems. However, agile differs so profoundly from traditional development practices that defense organizations must overcome significant challenges to foster greater agile adoption. Leaders cannot expect individual programs to tailor current acquisition processes on their own, because the complexities of the laws and policies do not lend themselves to obvious solutions, let alone accommodate processes so fundamentally different from current defense practices.

This paper has offered potential solutions to these key challenges in order to aid programs in laying a foundation for successful agile implementation. As agile adoption continues to take root and expand across programs, defense organizations would benefit from additional guidance and training to ensure consistent and pervasive success in agile IT acquisition.

References

1. Balter BJ (Fall 2011). Towards a more agile government: the case for rebooting federal IT procurement. Publ Contract Law J
2. Broadus W (January–Feburary 2013) The challenges of being agile in DoD. Defense AT&L Mag, 5–9
3. Cotugno F, Messina A (2014) Implementing SCRUM in the army general staff environment. The 3rd international conference in software engineering for defence applications—SEDA, Roma, Italy, 22–23 September 2014
4. Defense Acquisition University (DAU) (2015, March 20) Milestone document identification (MDID). Retrieved from DAU: https://dap.dau.mil/mdid/Pages/Default.aspx?ms=2&acat=2&acatsub=1&source=All&type=chart0
5. Defense Information Systems Agency (2015, March 20) GCSS-J. Retrieved from Defense Information Systems Agency: http://www.disa.mil/Mission-Support/Command-and-Control/GCSS-J/About
6. Defense Science Board (2009) Report of the defense science board task force on: department of defense policies and procedures for the acquisition of information technology. Retrieved from Office of the Under Secretary of Defense for Acquisition and Technology: www.acq.osd.mil/dsb/reports/ADA498375.pdf

7. Duquette J, Bloom M (2008) Transitioning agile/rapid acquisition initiatives to the warfighter. The MITRE Corporation, Virginia
8. U.S. Government Accountability Office (2012) Software development: effective practices and federal challenges in applying agile methods (GAO-12-681), U.S. Government Accountability Office, Washington, DC
9. Lapham MA (2012, January/February) DoD agile adoption. Retrieved from Cross Talk: http://www.crosstalkonline.org/storage/issue-archives/2012/201201/201201-Lapham.pdf
10. Lapham MA., Williams R, Hammons C, Burton D, Schenker A (2010) Considerations for using agile in DoD Acquisition. Software Engineering Institute, USA
11. Lapham M, Miller S, Adams L, Brown N, Hackemarck B, Hammons C, Schenker A et al. (2011) Agile methods: selected DoD management and acquisition concerns. Software Engineering Institute, USA
12. Messina A (2014) Adopting Agile methodology in mission critical software production. Consultation on Cloud Computing and Software closing workshop, Bruxelles, 4 November 2014
13. Messina A, Cotugno F (2014) Adapting SCRUM to the Italian army: methods and (open) tools. The 10th international conference on open source systems, San Jose, Costa Rica, 6–9 May 2014. ISBN 978-3-642-55127-7
14. Modigliani P, Chang S (March 2014) Defense agile acquisition guide. The MITRE Corporation, Virginia
15. Northern C, Mayfield K, Benito R, Casagni M (2010) Handbook for implementing agile in department of defense information technology acquisition. The MITRE Corporation, McLean, VA
16. Office of the Secretary of Defense (2010, November) A new approach for delivering information technology capabilities to the department of defense. Retrieved from Office of the Deputy Chief Management Officer: http://dcmo.defense.gov/documents/OSD%2013744-10%20-%20804%20Report%20to%20Congress%20.pdf

http://www.springer.com/978-3-319-27894-0

Proceedings of 4th International Conference in Software Engineering for Defence Applications
SEDA 2015
Ciancarini, P.; Sillitti, A.; Succi, G.; Messina, A. (Eds.)
2016, XI, 330 p. 120 illus. in color., Softcover
ISBN: 978-3-319-27894-0