



4

Acquisition & Management Concerns
for Agile Use in Government Series

Agile Acquisition and Milestone Reviews



Software Engineering Institute
Carnegie Mellon University

Acquisition & Management Concerns for Agile Use in Government

This booklet is part of a series based on material originally published in a 2011 report titled *Agile Methods: Selected DoD Management and Acquisition Concerns* (CMU/SEI-2011-TN-002).

The material has been slightly updated and modified for stand-alone publication.

Booklet 1: Agile Development and DoD Acquisitions

Booklet 2: Agile Culture in the DoD

Booklet 3: Management and Contracting Practices for Agile Programs

Booklet 4: Agile Acquisition and Milestone Reviews

Booklet 5: Estimating in Agile Acquisition

Booklet 6: Adopting Agile in DoD IT Acquisitions

Agile Acquisition and Milestone Reviews

The DoD 5000 series provides the framework for acquiring systems. This framework includes a series of technical reviews including but not limited to System Requirements Review (SRR), System Design Review (SDR), Software Specification Review (SSR), Preliminary Design Review (PDR), and Critical Design Review (CDR) among others. Historically, documents such as MIL-STD-1521, Technical Reviews and Audits for Systems, Equipment, and Computer Software, or the United States Air Force Weapon Systems Software Management Guidebook are used as guidance for conducting these reviews.¹ Some smaller programs do not require this level of review; however, programs that do require them expect a certain level of documentation and rigor regardless of the development methodology employed.

These milestone reviews are system reviews. From a systems point of view, some would say these reviews are not part of software development methodology and that this is an incorrect use of the terms. For purposes of this report and in the view of the authors, software intensive systems typically are subjected to PDRs, CDRs, and other reviews. While the overall system may be a plane, tank, ship, or satellite, the software still must pass the PDR, CDR, and other milestones. If the system in question is an IT system, then the PDR, CDR, and other reviews apply directly, as software is the main component of the system (along with hardware to run it on). This booklet is aimed at readers who are contemplating using Agile methods and who are subject to the typical review activities prevalent in DoD acquisitions. The authors hope that this booklet will provide some useful guidance on how to approach Agile methods when following traditional technical milestones.

Milestone Review Issue

In the SEI report *Considerations for Using Agile in DoD Acquisition*, the authors stated: *“A very specific acquisition issue and sticking point is that Agile methodology does not accommodate large capstone events such as Critical Design Review (CDR), which is usually a major, multi-day event with many smaller technical meetings leading up to it. This approach requires a great deal of documentation and many technical reviews by the contractor”* [Lapham 2010]. The types of documentation expected at these milestone events are considered high ritual and are not typically produced when using Agile.² If the PMO intends to embrace Agile methods then it will need to determine how to meet the standard milestone criteria for the major milestones reviews, particularly SRR, SDR, PDR, and CDR. However, Agile can be adapted within many different types of systems. One of our reviewers said that “the Atlas V heavy launch guidance system is produced using eXtreme Programming and

¹ Note that this booklet does not address the major milestone decision points, i.e., Milestone A, B, or C. Only technical milestones are addressed here.

² “High ritual” is a term used within the Agile community often interchangeably with “high ceremony.” Alistair Cockburn defined ceremony as the amount of precision and the tightness of tolerance in the methodology [Cockburn 2007]. For instance, plan-driven methods such as waterfall are considered high ritual or ceremony as they require an extensive amount of documentation and control. Agile methods are generally considered low ritual as the amount of documentation and control should be “just enough” for the situation.

has all the DoD procurement program events.” The key here is knowing how long or how much effort is required to review the program in question (an hour or a week).

In addition, according to one of our reviewers who has monitored Agile-based projects from a government perspective:

[I] found that in projects using an Agile methodology, technical reviews are around delivery of major capabilities and/or leadership is invited to certain sprint reviews. [I] also found that technical reviews are not as needed as much because stakeholders buy-in to stories/development/schedule at the end of every sprint, which we found is more productive than large reviews every so often.

In order to determine the type of criteria needed for any review, the intent of the review or purpose must be known. This raises a question: What is the purpose of any technical milestone review?

Intent of Technical Milestone Reviews

The Defense Acquisition Guidebook (DAG) says,

Technical Assessment activities measure technical progress and assess both program plans and requirements. Activities within Technical Assessment include ...the conduct of Technical Reviews (including Preliminary Design Review/Critical Design Review Reports)...A structured technical review process should demonstrate and confirm completion of required accomplishments and exit criteria as defined in program and system planning.... Technical reviews are an important oversight tool that the program manager can use to review and evaluate the state of the system and the program, redirecting activity if necessary. [Defense Acquisition University 2011a]

From a software perspective, each of these reviews is used to evaluate progress on and/or review specific aspects of the proposed technical software solution. Thus, expectations and criteria need to be created that reflect the level and type of documentation that would be acceptable for those milestones and yet work within an Agile environment.

Challenges

The intent of any technical milestone review is for evaluation of progress and/or technical solution. For PMOs trained and experienced in the traditional acquisition methods, evaluating program progress and technical solutions follows well-established guidelines and regulations. Very specific documentation is produced to provide the data required to meet the intent of the technical review as called out in the program specific Contract Data Requirements List (CDRL). The content of these documents and the entry and exit criteria for each review is well documented. However, even in traditional acquisitions (using traditional methods), these documents, exit and entry criteria can be and usually are tailored for the specific program. Since the documentation output from Agile methods appears to be “light”

in comparison to traditional programs, the tailoring aspects take on additional aspects. Some of the specific challenges for Agile adoption that we observed during our interviews that must be addressed are

- incentives to collaborate
- shared understanding of definitions/key concepts
- document content—the look and feel may be different but the intent is the same
- regulatory language

Many of these challenges are not specific to adopting Agile in DoD acquisitions but are also common to other incremental development approaches such as Rational Unified Process (RUP). In fact, RUP does address this level of milestone review. The reviews are called Lifecycle Architecture (LCA) Review and Lifecycle Objectives (LCO) Review. Some potential solutions to these challenges for Agile adoption are discussed in the following sections.

Agile Success Depends on Tackling Challenges

The PMO that is adopting or thinking of adopting Agile for an acquisition must set the stage to allow for the necessary collaboration required between the PMO and other government stakeholders and between the PMO and the contractor(s). The introduction of Agile into DoD acquisition can be considered yet another type of acquisition reform. The Defense Science Board has provided some cautionary words on any novel approaches for acquiring IT:

With so many prior acquisition reform efforts to leverage, any novel approach for acquiring IT is unlikely to have meaningful impact unless it addresses the barriers that prevented prior reform efforts from taking root. Perhaps the two most important barriers to address are experienced proven leadership and incentives (or lack thereof) to alter the behavior of individuals and organizations. According to the Defense Acquisition Performance Assessment Panel, ‘... current governance structure does not promote program success—actually, programs advance in spite of the oversight process rather than because of it.’ This sentiment was echoed by a defense agency director in characterizing IT acquisition as hampered by the oversight organizations with little “skin in the Game.” [Defense Science Board 2009]

Incentives for Acquirers and Contractors to Collaborate

While this topic is relevant to all contracting issues, it is particularly important if the “world of traditional milestone review” collides with the “world of agile development.” The authors have observed programs where collaboration was not yet optimal, which resulted in major avoidable issues when milestone review occurred. Thus, we emphasize them here.

We take the comment “skin in the game” to mean that the parties involved must collaborate. Thus, some form of incentive to do so must be created. (As the Defense

Science Board noted, “the two most important barriers are leadership and incentives or lack thereof.”). It is key that leadership convey clear goals, objectives, and vision for people to create internal group collaboration and move toward success. One of our interviewees likened an Agile program to a group of technical climbers going up a mountain: you are all roped together and if one of you falls, you all fall. If the Agile project is thought of in these terms, it is quite easy to have an incentive to collaborate. The key for DoD programs is to determine some incentive for both contractor and government personnel that is as imperative as the life and death one is for technical climbers.

Incentives need to be team based, not individual based. Agile incentives need to cross government and contractor boundaries as much as is practical, given regulations that must be followed. Each individual program has a specific environment and corresponding constraints. Even with the typical constraints, the successful programs we interviewed had a program identity, which crossed the contractor–government boundary. The incentives to collaborate were ingrained in the team culture with rewards being simple things like lunches or group gatherings for peer recognition.

It is incumbent upon the PMO and the contractor to negotiate the appropriate incentives to encourage their teams to collaborate. This negotiation needs to consider all the stakeholders that must participate during the product’s lifecycle, including external stakeholders. The external stakeholders (i.e., special review panels, other programs that interface with the program, senior leadership in the Command and OSD, etc.) need to be trained so they understand the methods being used, including, but not limited to, types of artifacts that are produced and how they relate to typical program artifacts. Without specific training, serious roadblocks to program progress can occur. Buy-in occurs with understanding and knowledge of Agile and the value it can bring. Defense Acquisition University (DAU) is beginning to provide some training on Agile methods and the associated acquisition processes.

Incentives should not be limited to just the contractor team but also encouraged for the government team as well. Government teams are often incented to grow their programs and follow the regulations. However, these actions should not be incented in an Agile environment unless the users’ needs are met first.

The type of contract could also play into the incentive issue. One of the interviewees using a T&M contract stated that they had a limited structure for rewards mainly due to the structure of the contract, which provided limited motivation for the contracting company to provide rewards as the contract personnel were relegated to the role of body shop. With this said, we observed a very collaborative environment on this program.

In order to collaborate, all the parties must have a common understanding of terms and key concepts. According to Alan Shalloway, Agile “has to be cooperative [in] nature and [people] are not going to be convinced if they don’t want to [be]” [Shalloway 2009]. We addressed a set of Agile and DoD terms for which common understanding is required in Section 2.3.1 of the original Tech Note from which this booklet is drawn. Some additional key concepts specific to supporting technical reviews in an Agile setting are discussed in the following sections.

Shared Understanding of Definitions/Key Concepts

While there are numerous key concepts involved with Agile development methods, the two we heard most about during our interviews were type and amount of documentation and technical reviews.

One of the key concepts related to Agile is the type and amount of documentation that is produced. Typically, Agile teams produce just enough documentation to allow the team to move forward. Regulatory documentation is still completed and may be done in a variety of ways. In our interviews, we saw several different approaches to producing regulatory documentation that was not considered to contribute directly to the development activities, but will be needed in the future (user manuals, maintenance data) or to meet programmatic requirements:

- A SETA contractor was hired by the program office to review the repository of development information (embedded in a tool that supported Agile methods) and produce required documentation from it.
- Technical writers embedded with the Agile team produced documentation in parallel with the development activities.
- Contractor personnel doing program controls activities produced required documentation toward the end of each release. The contractor program control personnel took the outputs from the Agile process and formatted them to meet the 5000 required documents.

Note that if the documentation is not of any value to the overall product or program, then waivers should be sought to eliminate the need.

Another key concept is that the whole point of using Agile is to reduce risk by defining the highest value functions to be demonstrated as early as possible. This means that technical reviews like PDR and CDR will be used to validate early versions of working software and to review whatever documentation had been agreed to in the acquisition strategy. Thus, additional documentation would not be created just to meet the traditional set of documents since that set would be tailored a priori.

Addressing Expected Document Content Mismatch

As previously cited Defense Acquisition Guidebook explains, *“A structured technical review process should demonstrate and confirm completion of required accomplishments and exit criteria as defined in program and system planning.”*

Thus, the documents produced for any of these reviews should support the required accomplishments and exit criteria that were defined in the program and system planning, and reflected in the acquisition strategy chosen by the program.

Agile planning is done at multiple levels, similar to a rolling-wave style of planning, though usually in smaller increments than is typical of rolling waves. The top-level overview of the system and architecture is defined and all releases and associated iterations (or sprints) are generally identified up front. The releases and associated iterations to be undertaken soon have more detail than those to be undertaken later—the further in the future, the less detail. Given this approach to planning,

how does document production fit in? Typically using traditional methods, the same level of detail is provided for all sections of a document and then refined as time progresses. This approach does not work for programs employing Agile methods. Thus, the way documents are created will need to evolve just as the methods for development are evolving. Future methods could employ tools that examine the code to extract information to answer predefined and ad hoc questions.

For today's Agile-developed work, because the work is being accomplished using an incremental approach, one should expect the documentation will also be accomplished in that manner. However, as discussed in Section 1.4.2, several of the programs interviewed have used varying approaches to creating the documentation. None of the approaches are better than the others; they are just tailored for the specific situation found within the program.

The intent and content of each artifact should be considered and agreed upon by both the PMO and the developer. The current Data Item Descriptions (DIDs) applied to the contract can be used as a starting point for these discussions. Once the intent and content are determined, appropriate entry and exit criteria can be created for each document. Keep in mind the timing of the releases and iterations when setting the entry and exit criteria so that the content aligns with the work being performed. Each program we interviewed took slightly different approaches to creating documentation. In all cases, the content and acceptance criteria were determined with the PMO.

Addressing Regulatory Language

While we do not know of any regulations that expressly preclude or limit the use of Agile, many in the acquisition community seem to fear the use of Agile because of their prior interpretations of the regulations [Lapham 2010]. Thus, care should be taken to ensure that all regulations are met and corresponding documentation is produced. At the very least, there should be traceability from the Agile-produced documentation to the required documentation. Traditional documentation as defined by CDRLs follow well established DIDs. Agile methods do produce documentation; however, the form is not the same as that prescribed in the DIDs, although the content may be the same. This difference tends to spark issues about documentation unless the contractor and the government PMO agree on some type of mapping. If needed, waivers can be requested to reduce or eliminate some of the regulatory implications.

Many of the current regulatory practices are at the discretion of the Milestone Decision Authority (MDA). In order for Agile to succeed, the MDA will need to be supportive of Agile processes. As one of our interviewees pointed out, ideally more responsibility would be delegated to PMs as they are more closely associated with Agile teams. In fact, this issue was one that a particular program continually ran into.

There are several endeavors underway to help define a more iterative or incremental acquisition approach for the DoD. These include the IT Acquisition Reform Task

Force, which is responding to Section 804.³ In addition, a smaller white paper effort is underway in response to Secretary Gates's solicitation for transformative ideas to improve DoD processes and performance, titled Innovation for New Value, Efficiency & Savings Tomorrow (INVEST).⁴ The white paper submitted to INVEST originated within the Patriot Excalibur program at Elgin Air Force Base and suggests specific language changes to some regulations to make it easier for those choosing an acquisition strategy to understand how they could effectively use Agile methods.

Potential Solutions for Technical Reviews

Many DoD programs require technical reviews; however, there is very little written about accomplishing these types of reviews when employing Agile methods. Written guidance on performing such Agile-related reviews is beginning to emerge at a high level (e.g., the 804 response paper), and several programs have done or are doing these reviews. The information presented here is based on many interviews performed during the research for this technical note and on the ideas of the author team, which arose during spirited debates.

First, the underpinning for any solution has to be developed. This is the agreement the PMO and developer have made about how to perform their reviews. The reviews are planned based on the program and system planning. Given the incremental nature of Agile work, the most productive approach to technical reviews will be incremental in nature also. We recognize that an incremental approach to reviews may not always be possible.

Our interviews showed that there tend to be two ways to accomplish the technical reviews:

1. Traditional technical reviews, like those that have existed on “waterfall” programs, have been added to an Agile process.
2. Technical reviews are an integral part of the Agile process and as such are a series of progressive technical reviews.

Traditional Technical Reviews

Where Agile methods are being used in a program to develop software, we saw two common approaches to dealing with technical milestone reviews:

- The program proceeded as though the software was being developed via waterfall as traditionally practiced, and the developer was responsible for ensuring that they produced the expected documentation and satisfied the entry and exit criteria.
- The program office or developer performed necessary reviews based on the Agile methods in use and then translated the Agile documentation into the expected artifacts for the review.

³ <http://www.afei.org/WorkingGroups/section804tf/Pages/default.aspx>

⁴ http://www.defense.gov/home/features/2010/0710_invest/. At least one of the programs we interviewed submitted a paper to this contest sponsored by Secretary of Defense Gates.

In either case, a translation step exists. In some instances, the developers are performing the translation internal to their program and presenting the standard artifacts one normally sees at the technical reviews. This approach has been called *covert Agile*. The developer and presumably the acquirer are still getting some of the risk reduction benefits from employing Agile but may be losing some of the potential cost savings due to creating the *translation layer*.

The other form of traditional technical reviews is one where the PMO is fully aware of the Agile method being employed by the development contractor. In this instance, the PMO does not want its developers dealing with the translation but rather hires a Systems Engineering and Technical Assistance (SETA) contractor to take the “raw” data produced from the Agile method and uses it to produce the required artifacts.

Progressive Technical Reviews

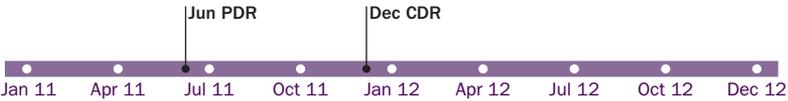
Progressive technical reviews use each successive wave of reviews to build on its predecessors. What does this mean? The program has only one PDR, CDR, etc., of record. However, when Agile is employed, the complete set of information for all requirements that is normally required for a PDR will not be available at that time. Thus, the available information will be used for the PDR with the understanding that multiple “mini” or progressive PDRs will occur as more information becomes available. One program we worked with is using this approach. Another program uses mini-PDRs and CDRs where they obtain signoff by the customer representatives. This is used to prove that the customer signed off, saying “they liked it.” This type of sign-off can be used to document performance throughout the program. A third program calls these Incremental Design Reviews (IDRs). Currently, SDR is the only milestone besides IDRs that they use. At SDR, they establish the vision, roadmap, and release plan for the remainder of the project (which spans multiple contracts). At the end of each release, they have an IDR. Each IDR will deliver threads of functionality defined at SDR.

Some of the information available for the PDR will be far more detailed, as it will represent completed functionality from the iterations that are already finished. For example, in Figure 1 the relationship of a PDR to a release and its associated sprints is shown. This notional diagram shows that the PDR is programmatically scheduled for June and the CDR is scheduled for December (milestone timeline). The iteration timeline shows all iterations. Iterations 1–5 are completed by June. Thus, there are five iterations completed by the PDR date. The PDR would include data from iterations 1–5.

The PDR⁵ also needs to demo a working, high-level architecture for the entire program implementing a critical set of functionality from the iterations completed by the PDR. This provides a demonstrable risk-reduction activity. In addition, PDR should also show how the requirements have been tentatively allocated to each release. For releases nearer in time (i.e., those closer to execution) the requirements will have

⁵ It should be noted that this discussion is redefining the terms PDR and CDR for use with Agile. This could lead to confusion and contribute to people “acting Agile” as opposed to “being Agile.” Perhaps the milestones should be renamed. This renaming is left to future work.

Milestone Timeline



Iteration Timeline

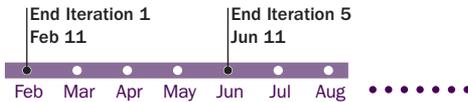


Figure 1: Relationship of Sprint to Release to PDR of Record

been allocated to individual sprints. The PDR should also address at a high level the external interfaces and the infrastructure for the program. This information is used to provide context and enable the overall review.

The PMO should consider holding CDRs for each release, which would ensure the next level of detail is available and that the highest value capabilities are done or broken down to a level that is ready for implementation.

An Alternate Perspective on Milestones

Another way to look at milestones such as PDR is to see how they align to releases and the associated iterations. This discussion provides an alternate look at how the milestones could be achieved.

Each iteration within a release follows a standard process. However, before the iteration starts the development team and stakeholders determine which of the capabilities (functional or non-functional) will be accomplished within the upcoming iteration. These capabilities have already been defined earlier in the program and listed in a backlog by priority and/or risk-reduction value. Once a capability is assigned to a release, it is decomposed into features as shown in Figure 2.

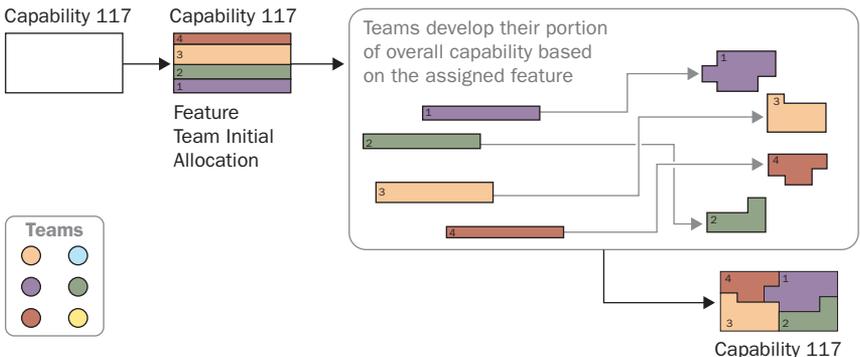


Figure 2: Capabilities Decomposed [Schenker 2007]

In this illustration, each feature of the capability is assigned to a different team for development during the iteration. The feature would be described using stories. The teams would be considered feature teams with one team singled out to be the capability guardian. Each feature team would then develop its portion of the overall capability using the process shown in Figure 3.

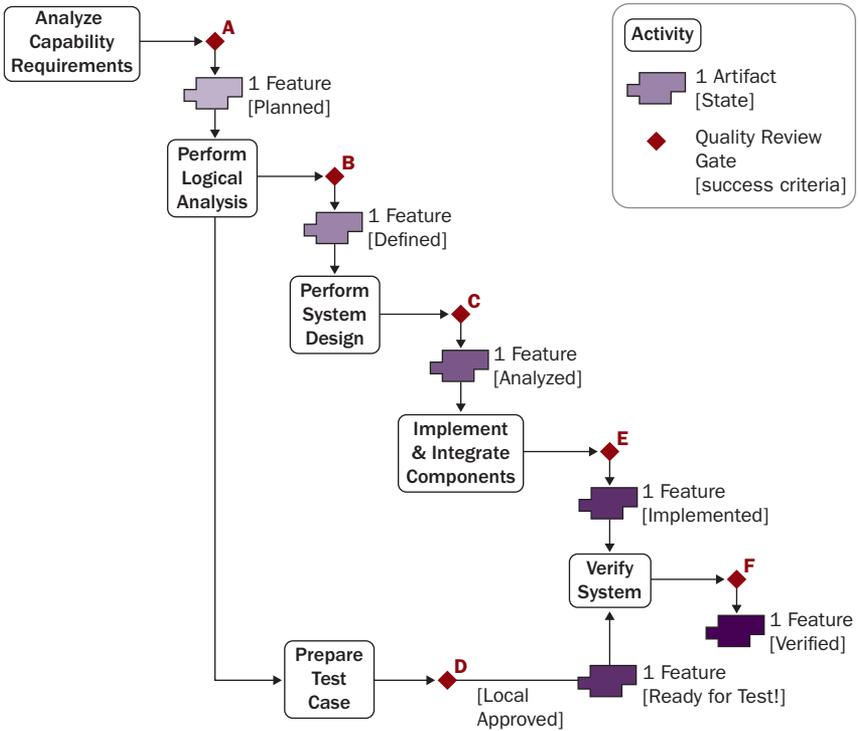


Figure 3: Development Process Within Each Team

Figure 3 depicts the development process each team uses on individual features during their iteration. For simplicity of comparison, the diagram looks very much like a waterfall process. However, all disciplines are represented on the team and they all work on the process together, with each person contributing in their discipline (analysis, coding, testing, etc.). Typically, the PMO is invited to participate at each Quality Review (QR) gate. This insight provides more touch points for the acquisition agent, typically the government, to be involved in the progress of the program. The information gained by attending these QR gates can be used to augment the formal PDRs, CDRs, etc. The diamond shapes lettered A-F are QR gates which must be passed before the artifacts from the iteration can be considered done. Each QR can be considered “mini” PDRs, CDRs, etc. As you can see, analysis, design, implementation and integration, and test are all accomplished by the team within

the time box allowed for the iteration. This type of PDR iteration would be expected to go beyond two weeks. Two weeks seemed to be a norm that is common in the interviewed programs that are not using formal technical reviews.

Now that we understand how the team functions, we can return to creating the capability shown in Figure 2. Capability 117 is decomposed into clear features that can be completed by a team within one iteration. Each feature is assigned to a team (1-4). This work is described in the work plan or scoping statement for the individual iteration. The teams get their individual features and proceed to develop their portion of the capability. Once all features are done, the capability is also completed.

This discussion of capabilities and features showing quality gates is another way to think about doing progressive milestones. It shows one possible detailed approach to solving the issue of technical milestones while using Agile methods.

It should be noted that this discussion does not address the difference between systems engineering/design and iterative development of the capabilities. These could be two very different items depending on the type of system. The systems engineering and design effort essentially feeds the product backlog. Iterative development addresses the creation of the actual products. The rhythm of each does not need to be the same. Further analysis and discussion of this area of concern are left to future work.

Conclusion

If a PMO intends to embrace Agile methods on a DoD program, it will need to determine how to meet the criteria for the major milestones reviews, particularly SRR, SDR, PDR, and CDR. These reviews tend to be significant events, requiring extensive documentation and presentations. This “high ritual” approach is not well aligned with Agile’s preference for “just enough” documentation and oversight, but it is possible for Agile programs to satisfy the intent of milestone reviews.

Even in traditional acquisitions (using traditional methods), the review artifacts as well as the exit and entry criteria are generally tailored for the specific program. While the documentation output from Agile methods appears to be “light” in comparison to traditional programs, the tailoring of traditional program reviews is a useful precedent. An Agile program simply tailors more than a traditional one. Similarly, an Agile program could adopt a progressive technical review scheme, where each successive wave of reviews build on its predecessors.

The point is that there are several viable alternatives which allow Agile programs to satisfy the objectives of technical milestone reviews. Program offices who wish to use Agile should therefore become familiar with the different ways to define and execute milestone reviews, and select the tailored approach best suited to their effort.

References

[Cockburn 2007]

Cockburn, A. *Agile Software Development: The Cooperative Game*, 2nd ed. Addison-Wesley, 2007.

[Defense Acquisition University 2011a]

Defense Acquisition University. *Defense Acquisition Guidebook*, July 2011. Defense Acquisition University, 2011.

[Defense Science Board 2009]

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

[Lapham 2010]

Lapham, M.A.; Williams, R.; Hammons, C.; Burton, D.; & Schenker, A. Considerations for Using Agile in DoD Acquisition (CMU/SEI-2010-TN-002). Software Engineering Institute, Carnegie Mellon University, 2010. <http://www.sei.cmu.edu/library/abstracts/reports/10tn002.cfm>

[Schenker 2007]

Schenker, F. & Jacobs, R. Project Management by Functional Capability. Presented at the 7th Annual CMMI Technology Conference and User Group, Denver, CO, November 2007. www.dtic.mil/ndia/2007cmmi/Thursday/3amSchenker.pdf

[Shalloway 2009]

Shalloway, A.; Beaver, G.; & Trott, J. R. *Lean-Agile Software Development: Achieving Enterprise Agility*. Addison-Wesley, 2009.

Copyright 2017 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, pursuant to the copyright license under the clause at 252.227-7013.

Internal use:* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use:* This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

*These restrictions do not apply to U.S. government entities.

DM17-0009



About the SEI

For more than three decades, the Software Engineering Institute (SEI) has been helping government and industry organizations acquire, develop, operate, and sustain software systems that are innovative, affordable, enduring, and trustworthy. We serve the nation as a federally funded research and development center (FFRDC) sponsored by the U.S. Department of Defense (DoD) and based at Carnegie Mellon University, a global research university annually rated among the best for its programs in computer science and engineering.

Contact Us

Software Engineering Institute
4500 Fifth Avenue, Pittsburgh, PA 15213-2612

Phone: 412.268.5800 | 888.201.4479

Web: www.sei.cmu.edu | www.cert.org

Email: info@sei.cmu.edu