Software Engineering Institute
Carnegie Mellon University

RFP Patterns and Techniques for Successful Agile Contracting

Authored by members of the NDIA System Engineering Agile Working Group:

Mary Ann Lapham
Larri Ann Rosser
Steven Martin
Thomas E. Friend
Peter Capell
Keith Korzec
Greg Howard
Michael Ryan
John H. Norton III

November 2016

SPECIAL REPORT
CMU/SEI-2016-SR-025

Software Solutions Division

http://www.sei.cmu.edu

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

This report was prepared for the
SEI Administrative Agent
AFLCMC/PZM
20 Schilling Circle, Bldg 1305, 3rd floor
Hanscom AFB, MA 01731-2125

DM-0004062

Table of Contents

Acknowledgments **v**

Executive Summary **vi**

Abstract **viii**

1 Introduction **1**
- 1.1 Purpose 1
  - 1.1.1 Rationale 1
  - 1.1.2 Topics Not Included 3

2 Background **4**
- 2.1 Agile Definition 4
- 2.2 Lean Thinking and Kanban 5
- 2.3 INCOSE Definition of Agility 5

3 Acquisition Context **6**

4 RFP Changes **7**
- 4.1 Section C 7
  - 4.1.1 The Statement of Objectives (SOO) 8
  - 4.1.2 Statement of Work (SOW) 9
  - 4.1.3 Reviews in an Agile Environment 9
  - 4.1.4 Key Technical Reviews 12
- 4.2 Section L 16
  - 4.2.1 Subfactor 1 – Agile Development Process 16
  - 4.2.2 Subfactor 2 – Systems Engineering Practices 17
  - 4.2.3 Subfactor 3 – System Test and Delivery 17
- 4.3 Section M 17
- 4.4 CDRLs 18
- 4.5 Cost Proposal and Schedule Instructions 20
- 4.6 Negotiation 22

5 Changes in Program Execution **24**
- 5.1 Start-Up (Kickoff) 24
- 5.2 Measures/Metrics 24
- 5.3 WBS Discussion 26
- 5.4 Change Management 27
- 5.5 Risk Assessment 27
- 5.6 Development Methodologies 27
  - 5.6.1 The Waterfall Model 28
  - 5.6.2 The Agile Model 29
- 5.7 Mixing Waterfall and Agile Methodologies 30
  - 5.7.1 The Changing Landscape of Software Acquisitions—Employing Hybrid Agile Models 31
  - 5.7.2 DoDI 5000.02 Facilitates Hybrid Agile Software and Waterfall Hardware Development 31
- 5.8 Bidder's Library 33

6 Conclusion **34**

Appendix A - Agile Fundamentals **35**

Appendix B - Marbach Updated Principles to the Agile Manifesto **37**

Appendix C - Acronyms **38**

Appendix D - Glossary **41**

References **43**

List of Figures

Figure 1: Requirements Moving En Masse Through the Process [Palmquist 2013] 10

Figure 2: Agile Building Blocks [Palmquist 2013] 10

Figure 3: Agile Development Pattern (Example 1) 11

Figure 4: Another View of Reviews in Agile Environment 12

Figure 5: CDRL Delivery 19

Figure 6: SW Development MIL-STD 881C Appendix K WBS Breakout (Traditional Waterfall) [PARCA 2016] 26

Figure 7: Possible Agile SW Development MIL-STD-881C WBS Breakout (Agile Capability-Based) [PARCA 2016] 26

Figure 8: Engineering V Model 28

Figure 9: DoD 5000.02 Hybrid Model 32

Figure 10: The Agile Software Development Lifecycle 36

List of Tables

Table 1: SOO versus SOW 7

Table 2: Example Definition for Adjectival Ratings 18

Acknowledgments

To Dan Ward for answering our call for how F.I.R.E. can support or shape technical reviews.

To the author team for sticking with the project and enduring the many meetings to produce this product:

- Mary Ann Lapham, SEI
- Keith Korzec, SEI
- Larri Ann Rosser, Raytheon Intelligence Information and Services
- Greg Howard, The MITRE Corporation
- Steven Martin, Space and Missile Systems Center
- Michael Ryan, BTAS
- Thomas E. Friend, Agile On Target
- John H. Norton III, Raytheon Integrated Defense Systems
- Peter Capell, SEI

To our independent reviewers for their insight:

- Dr. Elizabeth J. Wilson, Raytheon Integrated Defense Systems (Retired)
- Gari Palmer, Raytheon Integrated Defense Systems
- William E. Novak, SEI

And always to Gerald Miller, our unfailingly patient editor.

Executive Summary

Agile methods have proven effective in rapidly and responsively delivering functionality in commercial environments in which products are developed based on internally identified needs and offered for sale to multiple uncommitted customers. Agile approaches are becoming more and more attractive to the Department of Defense (DoD) and other federal agencies in light of recent budget pressures and a widespread need to bring capabilities to users more quickly than in the past.

To set the stage for this report, the authors turn to the *Performance Assessments and Root Cause Analyses (PARCA) Agile and Earned Value (EV) Desk Guide,* which describes the situation:

> *Constantly evolving threats have presented a demand for an acquisition process that is able to respond quickly to emerging requirements and rapidly changing environments. To address this, the DoD 5000.01 has encouraged the following characteristics in acquisitions:*
>
> *1. Flexibility: tailoring program strategies and oversight*
>
> *2. Responsiveness: rapid integration of advanced technologies*
>
> *3. Innovation: adapt practices that reduce cost and cycle time*
>
> *4. Discipline: use of program baseline parameters as control objectives*
>
> *5. Effective Management: decentralization to the extent practicable*
>
> *These characteristics have led to an increased focus on flexible development approaches that include Agile philosophies and integrated program management tools such as Earned Value Management* [PARCA 2016].

The *TechFAR Handbook* further observes

> *In the Government, digital services projects too often fail to meet user expectations or contain unused or unusable features. Several factors contribute to these outcomes, including the use of outdated development practices and, in some cases, overly narrow interpretations of what is allowed by acquisition regulations… The TechFAR consists of a handbook, which discusses relevant FAR authorities and includes practice tips, sample language, and a compilation of FAR provisions that are relevant to Agile software development [OUSD 2014].*

Agile methods by design comprise technical and programmatic practices employed to reduce wasted effort, decrease bureaucratic overburden, and enhance the productivity and engagement of developer teams. This document is intended to support the writers of RFPs as they incorporate Agile concepts into programs early in the lifecycle, by providing examples of language that will lay the groundwork for contracts on which programs rely.

Specific topics addressed include

- RFP Section C

- RFP Section L
- RFP Section M
- RFP contract data requirements lists (CDRLs)
- RFP Cost Proposal and Schedule Instructions
- Negotiation
- Start-Up
- Changes in Program Execution
- Hybrid Approaches
- Bidder’s Library

Abstract

Increasing budget constraints and emphasis on fielding capability faster have led the U.S. Department of Defense (DoD) and other federal entities to pursue the benefits of Agile software development—reduced cycle times, flexibility to adapt to changing conditions and user needs— that software development practitioners have achieved in the commercial market.

This report is written by the National Defense Industrial Association's System Engineering Agile Working Group to provide information on request-for-proposal (RFP) patterns and techniques for successful Agile contracting that can and have been used for contracts seeking to employ Agile methods. This report is intended to support the writers of RFPs in bringing Agile concepts into programs at the earliest possible time, providing examples of the kinds of language that will affect the foundations of contracts on which programs rely.

1 Introduction

This report is authored by the National Defense Industrial Association's (NDIA) System Engineering Agile Working Group in response to multiple inquiries about what language should be used in a request for proposal (RFP) to enable the use of an Agile approach. The report addresses the following topics:

Section 1 provides the introduction, purpose, and topics *not* included in this discussion.

Section 2 provides the background including an Agile definition, Lean thinking and Kanban, and the International Council on Systems Engineering (INCOSE) definition of Agility.

Section 3 discusses the overall acquisition context for employing Agile methods.

Section 4 discusses overall RFP changes including Section C (statement of objectives [SOO] or statement of work [SOW], reviews in an Agile environment, key technical reviews); Section L; Section M; contract data requirements lists (CDRLs); cost proposal and schedule instructions; negotiation; and start-up.

Section 5 discusses potential changes in program execution that will be encountered when employing Agile concepts including measures and metrics, work breakdown schedules (WBS), change management, risk assessment, and hybrid approaches.

1.1 Purpose

This report provides guidance on the language of government-contract RFPs to assist RFP authors in supporting agility in government systems acquisitions. "Agile methods" comprise those technical and programmatic practices that are employed to conscientiously be responsive to change, improve quality and fit for purpose, reduce wasted effort, decrease bureaucratic overburden on programs, and enhance the productivity and engagement of developer teams. This report is intended to support the writers of RFPs in incorporating Agile concepts into programs early in the lifecycle through various elements of the RFP process.

1.1.1 Rationale

The government is becoming increasingly aware that the traditional methods of RFP and contract development that are typically employed are not well suited to provide a quickly fielded capability based on an iterative or Agile development approach. Routinely, an RFP is formulated around a known solution with a majority of the requirements already identified. As in the past, the developers and acquirers follow the processes laid out in directed "best practices" such as the DoD 5000 series. Also like so many programs before, the results are often less than expected, cost much more, and are fielded years later than needed [GAO 2008].

Thus, a modified approach for the acquisition process is needed. Instead of contracting for an "end-to-end" system, segments of the government recently began following the lead of the private sector by acquiring capabilities that build on one another to eventually create the complete system

incrementally. The challenges with this approach are in defining the scope (SOO and SOW requirements), assessing the offer (evaluation criteria), evaluating progress (milestones and earned value [EV]), and confirming completion (sell-off criteria).

Confusion often occurs when the program office tries to interpret iterative activities relative to the traditional DoD acquisition lifecycle framework. There seems to be a common misunderstanding that the development methodology must mirror the acquisition lifecycle [Lapham 2014]. Problems occur when trying to overlay "traditional" acquisition milestone events directly atop development methodologies that use iterative work units, without understanding the relationship among the work units and the milestone events. This can be even more complex if Agile software methodologies are being used to develop a system that includes significant specialized hardware (e.g., acquisition of a satellite, ship, plane, or the like.). Many of these are issues that can be addressed by carefully writing an RFP that enables the government to take full advantage of the benefits that an Agile acquisition offers [Lapham 2014].

Agile or iterative system developments are based on collaboration between the contractor and customer. The acquirers are a part of the team rather than just a compliance organization. Without direct involvement of the system's intended users or knowledgeable surrogates (program management office [PMO] members or appointed contractor resources), early validation of the direction of the implementation is more difficult than necessary to achieve. This early validation is essential to ensure that iterative development proceeds in a stable manner, based on layer after layer of evolving, useful functionality. Some programs may refer to these evolving layers as "threads," "releases," "builds," "iterations," "spirals," "slices," and other similar terms [Lapham 2014].

> *There isn't only one approach to specifying contractual terms to incentivize proposal of productive Agile approaches. Because there isn't a single definition of "Agile," saying "we want a contractor to use Agile methods" isn't useful. Being clear about the relationship expected without inappropriately constraining the "how" things get done is the balance to seek.*
>
> *We have seen every DoD contract type used at one time or another for programs using Agile. Conditions under which one contract type or another works well are still anecdotal. Firm Fixed Price seems to be the most difficult to use to build the collaborative relationships that are the hallmark of effective Agile programs* [Foreman 2014].

Like the system under development, the RFP must address technical evaluations that are incremental. Similarly, plans, documentation, and other CDRLs are also developed incrementally over the entire life of the program. While different from the traditional acquisition world, this method has the advantage of providing documentation of the system "as built" rather than "as designed." This approach allows the government to take advantage of technical advances, changing operational environments, and/or requirements, as detailed design decisions are typically made closer to their implementation versus all up front. Furthermore, using as-built documentation removes any additional burden to the maintenance and logistics community that may come from using as-designed documentation.

Since the system is intended to evolve over the duration of the contract, the RFP must include the definition of success, especially if it used for incentive fees or progress payments to the developer.

As a principle, Agile welcomes changing requirements, even late in development. Thus, a clear plan on how to deal with the inevitable changes should be included in responses to an RFP to avoid the need to renegotiate the contract when issues arise. Likewise, it is advisable to address and document the government process to de-scope lower priority features and re-prioritize remaining features in response to new or changed requirements.

Finally, the RFP must include mechanisms to give acquirers real-time insight into the capabilities to be delivered, metrics, plans, specifications, schedules, and the like.

1.1.2 Topics Not Included

The authors have deliberately avoided including contractual language that could be "cut and pasted" into an RFP and convey a preference for one Agile methodology over any other.

In addition, discussion of specific program work plans or tasks, Rapid Acquisition, and hardware-only buys are not included.

2 Background

Traditional and Agile development both have a place in modern systems acquisition. Some general characteristics of programs more suited to each development approach are provided below. Keep in mind that these are generalities and there may be instances where a particular approach can be used successfully on a program that does not strictly conform to the characteristics listed for that approach. (For example, an Agile development approach could provide significant benefits to a program with stable requirements.) More detail on Agile development concepts is provided in Appendix A.

Traditional Approach

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

Agile Approach

- programs with evolving requirements and environment
- programs where solutions are sufficiently unknown that significant experimentation is likely to be needed
- programs for which the technology base is evolving rapidly
- programs with stakeholders who can engage with developers in ongoing, close collaboration [Nidiffer 2014]

So what exactly do we mean when we say "Agile"? For purposes of this report, the authors are not limiting the discussion to just software, although that's where the Agile Manifesto originated. Many of these Agile practices started with Agile and Lean manufacturing [Dove 1993]. INCOSE has also developed a definition of what it means by agility. One last piece that needs to be defined is the use of Lean in this Agile methodology and implementation. All three of these ideas are defined in the following paragraphs.

These ideas are explored throughout the paper and can be applied across the system lifecycle. In addition, many of the concepts discussed in this report can also be employed where the government understands the end state but the contractor wishes to use an Agile development methodology.

2.1 Agile Definition

"Agile is an iterative and incremental (evolutionary) approach to ... development which is performed in a highly collaborative manner by self-organizing teams within an effective governance framework with 'just enough' ceremony that produces high quality software in a cost effective and timely manner which meets the changing needs of its stakeholders" [Ambler 2013].

The concept was formalized in 2001 with the publication of the *Manifesto for Agile Software Development* as shown in Appendix A.

2.2 Lean Thinking and Kanban

Lean thinking is a way of thinking about any activity and seeing the waste inadvertently generated by the way the process is organized by focusing on the concepts of

1. value
2. value streams
3. flow
4. pull
5. perfection[1]

Kanban is a method for managing knowledge work with an emphasis on just-in-time delivery while not overloading the team members. This approach presents all participants with a full view of the process from task definition to delivery to a customer. Team members pull work from a queue.[2]

Lean thinking and Kanban techniques are often incorporated into programs using iterative and incremental approaches.

2.3 INCOSE Definition of Agility

Today we operate in unpredictable, uncertain, risky, variable, and evolving (UURVE) environments. Responding to this type of environment requires agility. This agility can be achieved by implementing Agile practices (e.g., Agile contracting, which buys services and/or capabilities instead of an overly specified product). With this environment in mind, INCOSE defines agility in the following manner:

> *This working group views agility as a sustainable system and process capability under UURVE conditions, enabled and constrained fundamentally by system architecture. This architecture delivers agile capability as reconfiguration, augmentation, and evolution of system and process functionality, during operation throughout the lifecycle. The architecture enables the system or process to respond to new and immediate situational requirements effectively. Effectiveness of response is measured in response time, response cost, response quality, and response scope sufficient to sustain functional intent* [INCOSE 2016].

Achieving agility for systems in today's environment requires the use of philosophies and methods such as those seen in Agile, Lean, and Kanban.

---

[1] See https://en.wikipedia.org/wiki/Lean_thinking

[2] See https://en.wikipedia.org/wiki/Kanban_(development)

3 Acquisition Context

Agile methods have proven effective in rapidly and responsively delivering functionality in commercial environments where products are developed based on internally identified needs and offered for sale to multiple uncommitted customers. While the federal acquisition model is different, recent budget pressures and environmental changes make the benefits of such methods attractive to the DoD and other federal entities. The *Performance Assessments and Root Cause Analyses (PARCA) Agile and Earned Value (EV) Desk Guide* describes the situation:

> *Per the DoD Defense Acquisition System Directive 5000.01, an acquisition system is "a directed, funded effort that provides a new, improved, or continuing materiel, weapon or information system, or service capability in response to an approved need… [t]he primary objective…is to acquire quality products that satisfy user needs with measurable improvements to mission capability and operational support, in a timely manner, and at a fair and reasonable price." Constantly evolving threats have presented a demand for an acquisition process that is able to respond quickly to emerging requirements and rapidly changing environments. To address this, the DoD has encouraged the following characteristics in acquisitions:*
>
> *1. Flexibility: tailoring program strategies and oversight*
>
> *2. Responsiveness: rapid integration of advanced technologies*
>
> *3. Innovation: adapt practices that reduce cost and cycle time*
>
> *4. Discipline: use of program baseline parameters as control objectives*
>
> *5. Effective Management: decentralization to the extent practicable* [PARCA 2016]
>
> *These characteristics have led to an increased focus on flexible development approaches that include Agile philosophies and integrated program management tools such as Earned Value Management* [PARCA 2016].

Inevitably, the attempt to merge the practices of the federal acquisition community and the commercial Agile community has resulted in questions and challenges, beginning with the acquisition process. The *TechFAR Handbook* observes

> *In the Government, digital services projects too often fail to meet user expectations or contain unused or unusable features. Several factors contribute to these outcomes, including the use of outdated development practices and, in some cases, overly narrow interpretations of what is allowed by acquisition regulations… The TechFAR consists of a handbook, which discusses relevant FAR authorities and includes practice tips, sample language, and a compilation of FAR provisions that are relevant to Agile software development* [OUSD 2014].

The rest of this report continues the exploration of applications of Agile methods and processes to federal acquisitions specifically addressing some of the issues that may arise when creating an RFP that enables Agile methods.

4 RFP Changes

Traditional RFPs are generally based on the government expecting a traditional waterfall-structured[3] program approach for creating the system in question. The traditional government procurement process and associated guidance does not necessarily facilitate employing Agile methods as part of the solution. As discussed in Section 3 of DoDD 5000.01 [DoD 2007], the PARCA *Agile and Earned Value Management* guide [PARCA 2016] and the *TechFAR Handbook* [OUSD 2014] address this issue by providing some alternative guidance. In addition, DoDI 5000.02 provides some alternative models to address iterative and incremental development. This section explores options that would either directly request Agile responses as part of the solution or at the very least allow for Agile responses even if the government is still expecting a traditional program structure.

4.1 Section C

Section C provides the government's requirements and expectations of the contractor's performance in the form of a SOO or SOW as part of the RFP. The SOO reflects a performance-based acquisition (PBA) and is best suited for an Agile acquisition.[4] If a SOO is provided, the government will normally expect the contractor to provide a SOW or a performance work statement (PWS) as part of its proposal. The government-provided SOW is best suited for a traditional acquisition in which the government has a high degree of confidence in the ability to specify (both qualitatively and quantitatively) the expected approach and product end state. Table 1 highlights the differences.

Table 1: SOO versus SOW

SOO	Factor	SOW
The government understands the objectives but expects the end state to evolve.	**Government Understanding**	The government has a high level of confidence in the end state.
Change is expected to be a significant factor in achieving the end state.	**Change**	Change is not anticipated, or if encountered will not be disruptive.
This approach provides the offerors trade space and flexibility in developing their proposals. It should probably be used unless the totality of the work effort required is very well understood by the government.	**Constraint**	Constrains offerors to the specific tasks identified, so it must be unambiguous and comprehensive. The government needs to apply specific constraints on the tradeoff space of lifecycle cost, performance, interoperability, logistics/training, etc.

[3] The authors assume some familiarity with sequential, waterfall-based software development models. A brief refresher is provided in Section 5.6.1.

[4] See *AcqNotes*, Acquisition Process, Performance-Based Acquisition, at http://acqnotes.com/acqnote/acquisitions/performance-based-acquisitions

This approach conforms to PBA practices.	**Performance-Based Acquisitions (PBA)**	NA

4.1.1 The Statement of Objectives (SOO)[5]

The SOO is used in PBA "which structures the acquisition around the purpose and outcome desired as opposed to the process by which the work is to be performed."[6] With performance-based acquisition, the government no longer develops a prescriptive SOW directing how the contractor will achieve project milestones. Instead, the government develops a SOO or PWS that describes the overall outputs and objectives but does not specify how to achieve those outputs.

The SOO is "incorporated into the solicitation that states the overall performance objectives. It is used in solicitations when the Government intends to provide the maximum flexibility to each offeror to propose an innovative approach."[7] It focuses on the government's objectives and identifies the broad, basic, top-level objectives of an acquisition/procurement. The SOO is developed after performing a risk assessment that highlights the high and moderate risks for business, programmatic, and technical areas against the requirement document. It is best applied when the government can describe the objectives but is not ready to specify the end state and expects change as a factor in achieving the end state. It provides offerors the flexibility to develop cost-effective solutions and propose innovative alternatives to meet the objectives.

A SOO supplements the performance-based requirements document. In a DoD acquisition, a SOO aligns with an initial capabilities document (ICD); programmatic direction from the acquisition decision memorandum (ADM), acquisition strategy, and acquisition plan; technical requirements from system specifications; and the draft WBS and dictionary.

The SOO provides enough information and detail for the offeror to structure a sound program, designed to be executable and satisfy government objectives. The offeror uses the SOO to develop the contractor statement of work (CSOW), the contract(or) work breakdown structure (CWBS), the integrated master schedule (IMS), and other documents supporting and defining the proposed effort.

The contractor will sell off the government objectives by meeting the predefined and negotiated definition of done[8] when using Agile methods. Once the definition of done is met, the government objectives can be said to have been met.

---

5 Derived from *AcqNotes*, Proposal Development, Statement of Objectives, at http://acqnotes.com/acqnote/tasks/statement-of-objectives

6 See *AcqNotes*, Acquisition Process, Performance-Based Acquisition, at http://acqnotes.com/acqnote/acquisitions/performance-based-acquisitions

7 See http://www.acq.osd.mil/dpap/ccap/cc/jcchb/HTML/Topical/sow.html

8 Done is also referred to as "Done" or "Done Done." This term is used to describe a product increment that is considered releasable—it means that all design, coding, testing, and documentation have been completed and the increment is fully integrated into the system.

4.1.2 Statement of Work (SOW)[9]

The SOW describes the government's needs for the work to be done in developing or producing the goods or services. The SOW is "a portion of a contract which establishes and defines all non-specification requirements for contractor's efforts either directly or with the use of specific cited documents."[10] It also facilitates the preparation of a proposal and aids the government in conduct of the source selection and contract administration after award.

The preparation of an effective SOW requires both an understanding of the goods or services that are needed to satisfy a particular requirement and an ability to define what is required in specific, performance-based, quantitative terms. This requires the government to have a high level of confidence that it understands the end state and doesn't expect change to be disruptive. The SOW should reference qualitative and quantitative design and performance requirements contained in specifications such as those developed according to MIL-STD-961.

4.1.3 Reviews in an Agile Environment

In a traditional waterfall model, technical reviews are used as control gates to move from one sequential phase to the next (e.g., concept > requirements > design > develop > test and integration > deployment). Figure 1 shows the traditional progression of requirements through the waterfall model [Palmquist 2013]. A traditional program tends to be risk averse and uses reviews for risk mitigation.

In an Agile program, the focus is on completing each work unit from analysis to deployment instead of completing each phase in sequential order. For example, as shown in Figure 2, the Agile world uses the same building blocks but looks at them differently than the traditional world [Palmquist 2013]. An Agile program tends to use technical reviews as opportunities for information sharing, face-to-face coordination, and confidence building.

For this reason, Agile programs may treat some reviews (most frequently design and test readiness) incrementally as part of the design-develop-integrate-test (DDIT) cadence, which allows for a faster start (from contract start to first incremental design review) and more opportunities to address changes in technology and the mission environment. In general, the larger part of the program lifecycle that is Agile, the greater the benefit, but Agile methods can provide value even when applied only to software development. The decision about which reviews are conducted sequentially and which incrementally should be based on program needs and challenges. Keep in mind there is no one correct way to implement Agile concepts within a program. Having provided this caution, we are providing in the following paragraphs two examples of how a program may implement Agile concepts within the milestones typically seen in a government program.

---

[9] Derived from *AcqNotes* Proposal Development, Statement of Work (http://acqnotes.com/acqnote/tasks/statement-of-work)

[10] See http://www.acq.osd.mil/dpap/ccap/cc/jcchb/HTML/Topical/sow.html

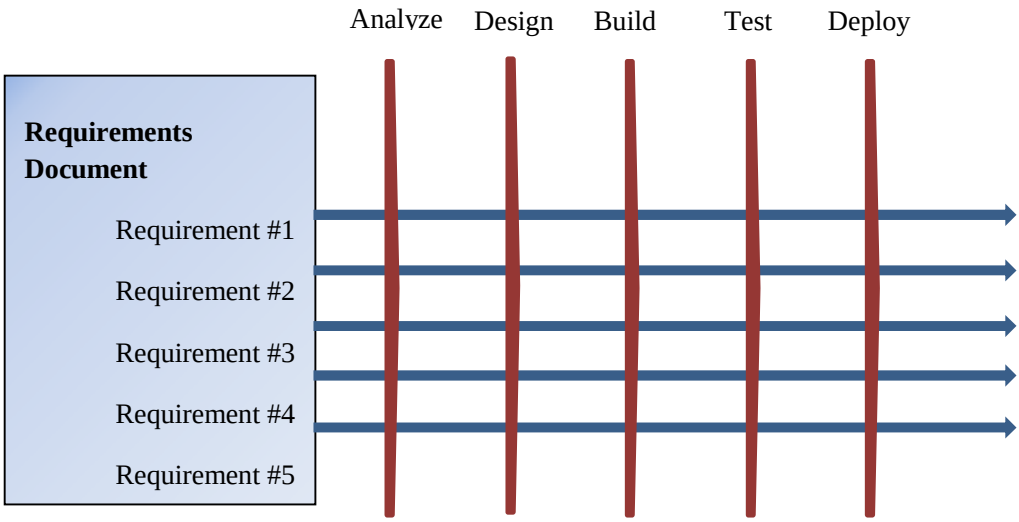

Figure 1: Requirements Moving En Masse Through the Process [Palmquist 2013]

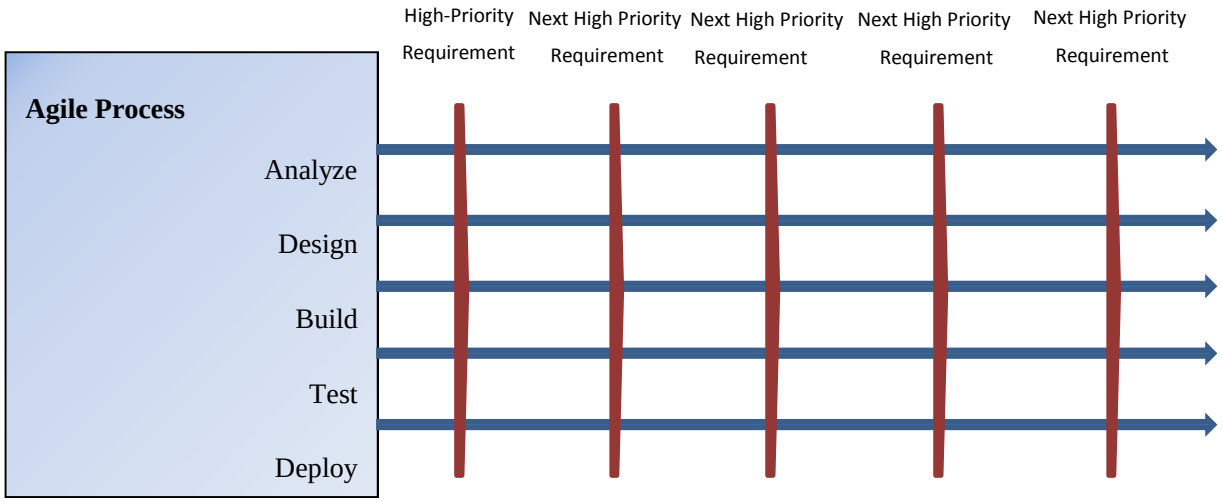

Figure 2: Agile Building Blocks [Palmquist 2013]

A generic Agile program development pattern is shown in Figure 3. In this example, the pattern is based on a structure where the period of performance contains a series of time-boxed releases (1-n). Each release (as shown in the expanded portion) begins with release planning, followed by a series of DDIT iterations (e.g., sprints 1-n) and concludes with a system demo followed by a deployment decision. Notice that a design review is shown at release planning and a deployment decision is shown at the end of the release. These reviews can be used as tailored alternatives to satisfy the preliminary design review (PDR) and critical design review (CDR).

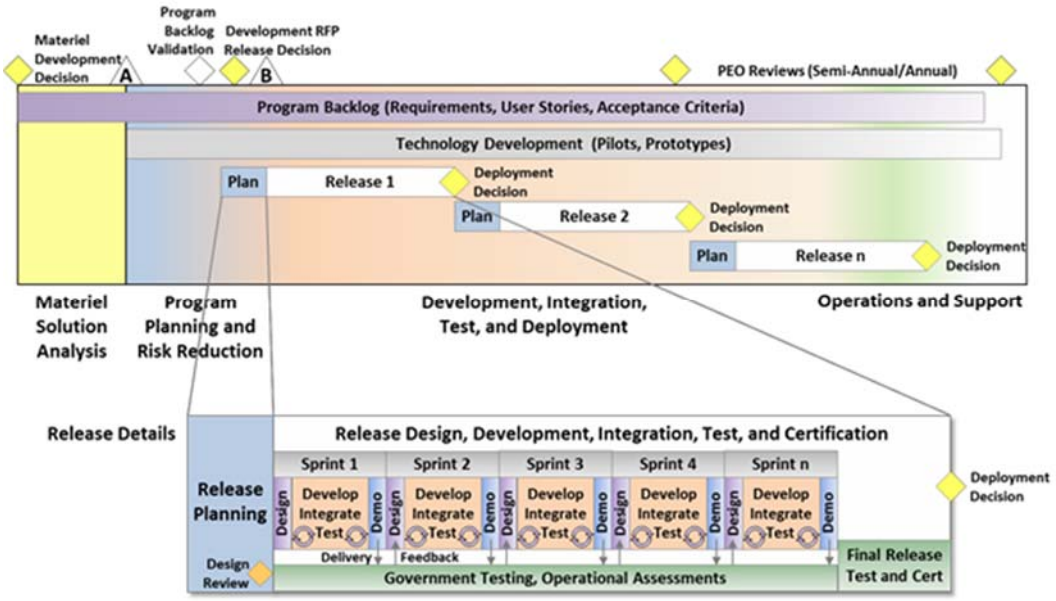

Figure 3: Agile Development Pattern (Example 1)[11]

Note that the graphic implies deployment at the end of a release. A more sophisticated Agile model separates deployment from the release time box. In this model, software is still developed on cadence (the release time box) but released on demand. Rather than push new software to the system, new software is always available for the system to pull at any time.

Our second example, Figure 4, depicts the contrast between typical milestones and reviews in a traditional waterfall environment versus incremental reviews in an Agile model. In this model, the traditional System Requirements Review (SRR), PDR, and CDR events are replaced by incremental design reviews and, if needed, system-level reviews. The system-level reviews are smaller in scope than the corresponding events in the traditional waterfall programs, focusing on the system architecture, hardware integration, system-level risks, and other system-level concerns.

[11] Graphic provided by the MITRE Corporation (https://www.mitre.org/publications/technical-papers/defense-agile-acquisition-guide-tailoring-dod-it-acquisition-program).

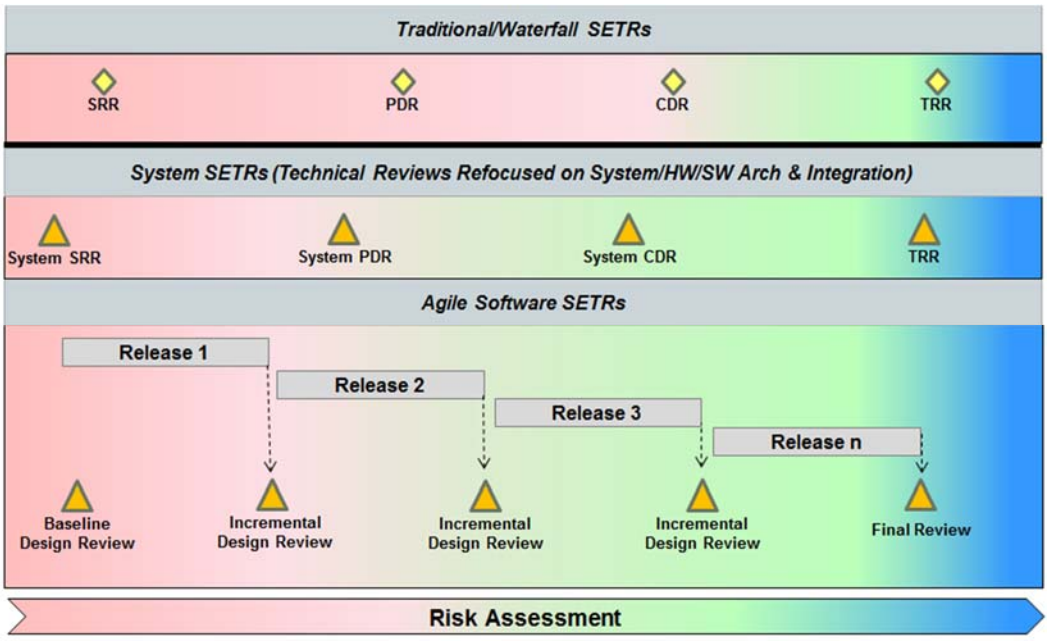

Figure 4: Another View of Reviews in Agile Environment[12]

4.1.4 Key Technical Reviews

Technical reviews should reflect the Agile principles, the Agile principles as defined in the Marbach paper presented at INCOSE 2015 [Marbach 2015], guidance from F.I.R.E. by Dan Ward [Ward 2014], or some combination thereof, depending on the environment and needs of the individual program.

Two Agile principles—"Business personnel, customers or their advocates, and implementers must work together daily" and "Continuous attention to technical excellence and good design enhances agility"—are the most applicable in this area. Marbach also lists nearly identical counterparts: "Business personnel, customers or their advocates, and implementer must work together daily throughout the project" and "Continuous attention to technical excellence and good design enhances agility" [Marbach 2015]. (See Appendix B for a comparison of the Agile principles and the Marbach principles.) To summarize F.I.R.E.: reviews must add value to the program—not just be held because status quo demands them; keep them to a minimum while keeping them small, quick and agile to minimize potential changes in the technical and operational environments. In addition, apply the heuristic of "Minimize the distance between decision and action" [Ward 2014].

Key technical reviews are listed below.

[12] Reproduced and modified with permission from the Raytheon Company.

System Requirements Review (SRR)

When Conducted. At the beginning of the period of performance and any time at which there is a major change in the capabilities baseline.

Purpose. The System Requirements Review (SRR) is a multi-disciplined technical review to ensure that the developer is ready to proceed with the initial system design.[13] The review will include the contractor's understanding of the program's objectives, capabilities and non-functional requirements baseline, constraints such as architecture, time critical deliveries, and milestones.

Input. Capabilities baseline, program vision, and roadmap.

Output. Initial program backlog.

Preliminary Design Review (PDR)

When Conducted. Post SRR/system functional review (SFR) (in traditional/waterfall); scheduled when appropriate to verify architectural consistency between the hardware and software, technical performance measures (TPMs), and the like (in an Agile development program, only if needed).

Purpose. In a traditional waterfall program, PDR demonstrates that the preliminary design meets all system requirements with acceptable risk and within the cost and schedule constraints. It establishes the basis for proceeding with detailed design.[14] In Agile, the large scale and formality of a PDR is replaced with more frequent, less formal interactions between the customer and the contractor. Examples include release planning and release demos. As such, a traditional PDR is not recommended on purely Agile programs. In some hybrid programs, a PDR may be desirable. For example, in a program using Agile software development and a waterfall approach to hardware development, a system-level PDR might be used to assess overall program progress. In that case, the focus of the PDR event should be verifying architectural consistency between the hardware and software, TPMs, and hardware components. See Figure 4 for an example of how PDR and CDR can be adapted to an Agile environment.

Input. Artifacts demonstrating that the preliminary design meets all system requirements with acceptable risk within cost and schedule constraints.[15]

Output. Decision to proceed with detailed development (in traditional/waterfall); insight into current program status regarding architectural consistency between the hardware and software, TPM maturity, etc. (in an Agile development program).

---

13 See https://acc.dau.mil/CommunityBrowser.aspx?id=638317

14 See https://en.wikipedia.org/wiki/Design_review_%28U.S._government%29

15 See https://en.wikipedia.org/wiki/Design_review_%28U.S._government%29#cite_note-11

Critical Design Review (CDR)

When Conducted. Post PDR (in traditional/waterfall); scheduled when appropriate to verify architectural consistency between the hardware and software, TPMs, etc. (in an Agile development program, only if needed).

Purpose. In a traditional waterfall program, CDR demonstrates that the maturity of the design is appropriate to support proceeding with full-scale fabrication, assembly, integration, and test. CDR determines that the technical effort is on track to complete the flight and ground system development and mission operations, meeting mission performance requirements within the identified cost and schedule constraints.[16] As is the case with PDR, a traditional CDR is not recommended on purely Agile programs for the same reason; however, again, on some hybrid programs, a CDR may be desirable and would be focused on the same type of topics as an "Agile PDR," but the expected level of maturity (of TPMs for example) would be greater.

Input. Artifacts demonstrating that the maturity of the design is appropriate to support proceeding with full-scale fabrication, assembly, integration, and test, within cost and schedule constraints.[17]

Output. Decision to proceed with fabrication, assembly, integration, and test (in traditional/waterfall); insight into current program status regarding architectural consistency between the hardware and software, TPM maturity, etc. (in an Agile development program).

Release Planning and Review

When Conducted. At the beginning of the release (nominally a week in duration).

Purpose. Demonstrates confidence (sufficient understanding of the requirements and design approach) for moving into the DDIT activities shown in Figure 3 that produce the release code baseline. The release planning combines elements of a traditional PDR and CDR, with enough confidence in maturity to move into the next increment; the remaining level of maturity is achieved during the DDIT phase. Release planning meeting and design reviews are iterative—the first release is typically different from subsequent ones in that there is a greater focus (sometimes an exclusive focus) on architectural concepts and enabling elements. Typically, the first release also does not involve adjudicating high-priority defects against features, although it may require effort to refactor inherited system artifacts from legacy programs. The design items presented at release planning pertain to the work (capabilities and architecture) allocated to that release. When an item (e.g., architecture) from a previous release changes in the current release, the changes would be presented as an update (i.e., "updates as needed"). The program backlog drives release planning and determines the priorities of the system component requirements allocated to the release. Program and business stakeholders establish release priorities and objectives. Development teams derive stories from the requirements and commit to their team objectives, which are rolled up to program objectives.

---

[16] See https://en.wikipedia.org/wiki/Design_review_%28U.S._government%29

[17] See https://en.wikipedia.org/wiki/Design_review_%28U.S._government%29#cite_note-11

Input. Release backlog (allocated from the program backlog).

Output. Finalized release backlog and initial team backlogs. Confidence-based decision to move into DDIT iterations jointly made between government and contractor.

Test Readiness Review (TRR)

When Conducted. Prior to system demo.

Purpose. The TRR is conducted to determine if the system under review is ready to proceed into formal testing by deciding whether the test procedures are complete and verify their compliance with test plans and descriptions.[18] In an Agile program, the preference is for the test organization to be continuously involved beginning at release planning, where the test strategy is discussed, the backlog is prioritized to align with test events, and the like. The TRR may be needed on a program where the test organization has not been closely involved with the iteration demos or if significant hardware is involved and/or the system is being tested at a government site. The TRR will review the results of the tests conducted during the DDIT iterations and identify workarounds, shortfalls, constraints and level of confidence for conducting a system demo.

Input. Iteration test results, system demo test plan.

Output. Decision to conduct system demo.

4.1.4.1 Release Demo and Deployment Review

When Conducted. At the conclusion of the DDIT iterations.

Purpose. The release demo and deployment review is a technical review of the release requirements and the as-built system design. It determines, based on the results of the release demo, if the release objectives and requirements have been met and the system performs as expected. If necessary, this may include a deployment review to determine the user's and the system's readiness to accept the new baseline.

Input. Results of the system demo.

Output. Decision on the adequacy of the as-built baseline and readiness for deployment.

4.1.4.2 Other Regulatory Review Information

IEEE Standard 15288.2-2014, *IEEE Standard for Technical Reviews and Audits on Defense Programs*, lists the following reviews: alternative systems review (ASR), system requirements review (SRR), SFR, preliminary design review (PDR), CDR, test readiness review (TRR) (contained within the program's test and evaluation master plan), functional configuration audit (FCA), system verification review (SVR), production readiness review (PRR), and physical configuration

[18] See http://acqnotes.com/acqnote/acquisitions/test-readiness-review

audit (PCA) [IEEE 2016]. Note that IEEE Standard 15288-2 has not been updated for Agile methods, and lists traditional plan-driven reviews such as PDR and CDR, which are tailored into the Agile SDLC.

Other technical reviews (e.g., site acceptance review, operational readiness review, transition readiness review, etc.) may be employed according to the needs of the program. However, if additional reviews are employed, they should be tailored to be as streamlined as possible. Remember the Agile principle that "simplicity—the art of maximizing the amount of work not done"—is essential, along with the clarification recommended by Marbach that "a truly Agile development project does not force artificial reporting and process requirements on the Implementation Team" [Marbach 2015].

4.2 Section L

In an RFP, Section L contains the instructions for providing the proposal to the government. These instructions can be long and complicated. But the key elements of the proposal are answered in the cost and technical factors. Technical factors will be discussed in this section but cost considerations will be deferred to Section 4.5. An RFP can have one or more technical factors depending on the scope and complexity of the system to be developed. For this discussion, only one technical factor is used for system development, but the factor has three subfactors. During the evaluation of the proposals, the rating for each subfactor is rolled up into an overall rating for the technical factor. The method in which the subfactors are evaluated is explained in Section M of the RFP. The technical subfactors are Agile development process, systems engineering practices, and system test and delivery.

1. The first subfactor is essential in describing how the entire system lifecycle will be managed and implemented using Agile concepts and methods.
2. The second subfactor requests the detailed description of the Agile systems engineering and other discipline-specific practices to be used during the development. It would also include the schedule of work and key events that will enable the integration of the total solution. These engineering processes should reflect and be compatible with Agile concepts discussed in the other subfactors since the intent is to enable a holistic Agile development approach.
3. The final subfactor is used to evaluate how the completed system will be delivered to the user, which would include several steps such as acceptance, certification, installation, training, and transition from a previous system. Again, the engineering processes in this subfactor should reflect and be compatible with Agile concepts discussed in the other subfactors.

In all cases, how the subfactors work together (since this is a request for an Agile proposal) should be included in the evaluation criteria. The following sections contain example wording of the subfactors to be used in the RFP.

4.2.1 Subfactor 1 – Agile Development Process

Describe the Agile development lifecycle process to design and build "system name" based on the Agile development methodology. When describing the process, explain how the process implements the Agile principles listed in Appendix B from *Principles for Agile Development* [Marbach 2015], which are adapted for DoD and federal systems from the principles listed on the Agile

Manifesto website (http://www.agilemanifesto.org). The description should include specific details on the management of requirements decomposition and the product backlog (prioritization of work with the government), the orchestration of the demonstrations (collaboration with the users on product evaluations during development), and the definition of "done" (meeting the government's product delivery requirements defined in the CDRLs). Include shorter but more frequent integrated client/developer meetings, user involvement, and other items specific to the Agile approach.

4.2.2 Subfactor 2 – Systems Engineering Practices

Describe systems engineering and any other specific engineering processes required to develop the architecture for "system name" not included in Subfactor 1. In addition, a system capabilities and features roadmap should be provided that maps capabilities and features to the development iterations and system build deliveries. This subfactor should also include further details about meeting the entrance, exit, and review criteria for the technical reviews as stated in IEEE 15288 Volumes 1 and 2 and IEEE 12207. (More details on the Technical Reviews are in Section 4.1.4.)

4.2.3 Subfactor 3 – System Test and Delivery

Describe the testing planned prior to delivering the software to the government. While some of this information should be included in the Agile development process subfactor, specific details for automated test, functional, integration, vulnerability, and defect tracking should be added here. Also, describe how support will be provided during government development test and evaluation (DT&E) and operational testing and evaluation (OT&E). This should include test environment, test data/connections to external systems/stub to simulate outside systems, test scripts, support, training, and response to issues (e.g., deficiency reports [DRs]).

Describe the process for delivering the software and its associated CDRLs to the government for fielding the system including documentation, distribution, tracking, transition of ops, training, certifications, build process, archiving, and recovery.

4.3 Section M

Section M of the RFP explains how the decision for the award will be made. As with traditional proposals, the evaluation of Agile proposals is based on the following factors:

- technical
- past performance
- cost/price
- cost/price risk

The relative importance of the above factors must be addressed in accordance with FAR section 15.304(e). Within the technical factor, the subfactors are the key discriminators to support a meaningful evaluation of competing proposals. Since these subfactors were all objective, they would receive an adjectival rating such as outstanding, satisfactory, marginal, and unsatisfactory. If a subfactor is found to be unsatisfactory, it must be rated unsatisfactory [Navy 1998]. Table 2 provides an example of definitions for the adjective ratings.

Table 2: Example Definition for Adjectival Ratings

Adjectival Rating	Definition
Outstanding	Proposal significantly exceeds requirements in a way that benefits the government or meets requirements and contains at least one exceptional enhancing feature that benefits the government. Any weakness is minor.
Satisfactory	Proposal meets requirements. Any weaknesses are acceptable to the government.
Marginal	Proposal contains weaknesses or minor deficiencies that could have some impact if accepted.
Unsatisfactory	Proposal does not comply substantially with requirements.

Based on the example subfactors used in the previous section, the following paragraphs are examples of how the subfactors could be evaluated:

Subfactor 1: Agile development process subfactor will primarily be evaluated by how the proposed system development process will adhere to each of the principles of Agile development and the government's Agile development approach. Additionally the proposal will be evaluated on how the contractor will partner with the customer to manage the product backlog with the program office; provide software demonstrations to the program office and the user; and if the artifacts to be provided at the end of each sprint meet the program office's definition of done.

Subfactor 2: Systems engineering practices subfactor will be evaluated by how the proposed systems engineering process will be performed as part of a holistic Agile development approach while meeting the government expectations for technical reviews. Also, this evaluation will include the review of the capabilities and features roadmap and its mapping to the iteration schedule, the planned releases for integration and test, and planned schedule of activities to complete the delivery of the system.

Subfactor 3: System test and delivery subfactor will be evaluated by activities, test events, and delivery of artifacts required to perform government developmental and operational testing and fielding of the system to the user. In addition, details about the test environment and supporting test structure such as test scripts, test data, and simulated connections to external systems will be evaluated for meeting government test and security requirements. Also, this subfactor will evaluate the use of automated tools used for testing throughout the development process and automated tools for installing and sustaining the software.

4.4 CDRLs

The number and type of CDRLs will depend on the program environment. The same approach of mindful tailoring as cited in DoDI 5000.02 should be applied to the CDRLs the program needs. Due to the anticipated close and continuous coordination between the government and the contractor, the number of formal deliveries of the CDRLs may be reduced (e.g., real-time information sharing between the contractor and the government via an integrated data environment). However,

all items that are needed for the operation of the system, including the software, user manuals, and technical data for sustainment activities, must be on the CDRL.

For development and design documentation such as the system requirements specification and software design description, additional care needs to be taken to ensure the due dates reflect the incremental development and delivery of the products. The design documentation will reflect the "as-built" product not the "as-designed" product. Keep in mind that the contractors will most likely employ rapid documentation techniques such as auto-generated documents and wikis. Access to any joint integrated data environment should be provided to the government so that it has visibility into the technical information to meet the delivery requirement. The use of these constructs should be explained in their overall Agile development management plan. Formal deliveries should be aligned with the program cadence.

As shown in Figure 5, deliveries on an Agile program occur more frequently at smaller events than in a traditional waterfall program. As such, much of the documentation will be delivered and reviewed while it is in-process with the maturity of the CDRL increasing over time. This can reduce risk earlier than in a traditional waterfall approach as the government can see early versions of the system and its documentation, then watch as risks are burned down during each subsequent iteration.

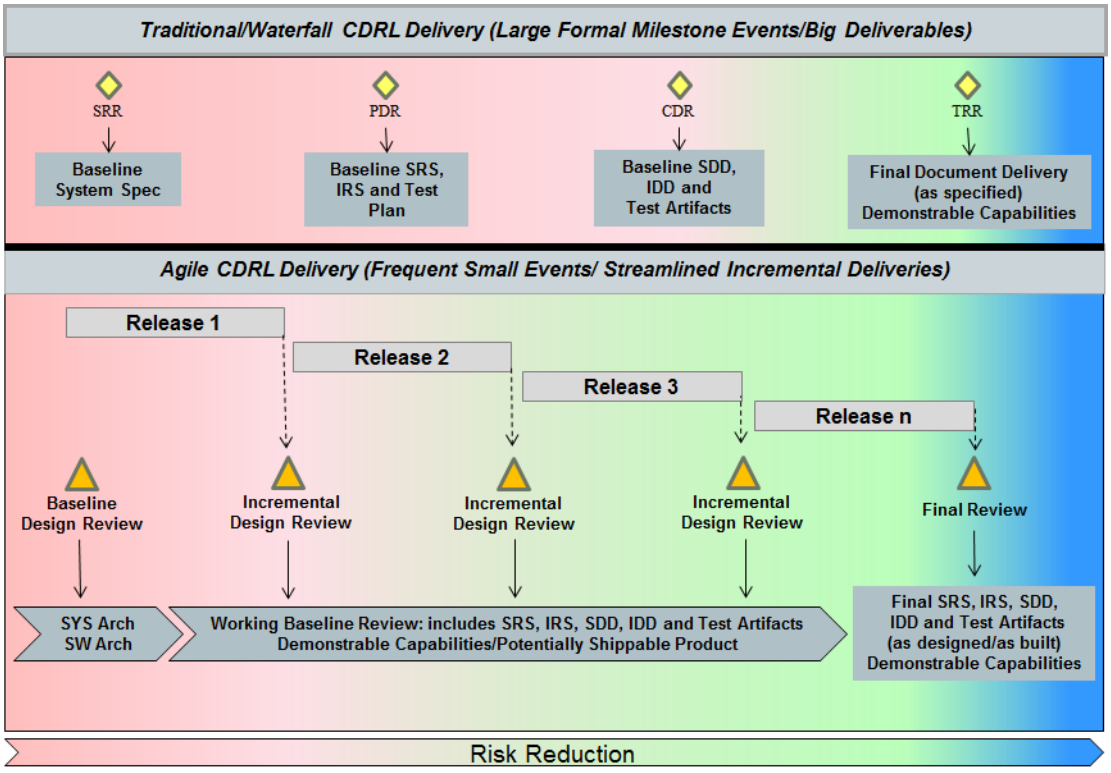

Figure 5: CDRL Delivery

4.5 Cost Proposal and Schedule Instructions

Cost

In a traditional waterfall model, the scope is fixed with cost with schedule and/or quality being variable. In a "customary" Agile program, cost, schedule, and quality are fixed with the scope (capabilities and features) being prioritized. Agile is flexible, however, and allows for several approaches to be used in the RFP and the cost proposal in particular. Specifically, the RFP must define the deliverable scope and, additionally, what is fixed and what is variable. The structure and approach of the contract should be based on the driving needs of the particular government entity responsible for the contract.

Defining Deliverable Scope—What's Fixed, What's Variable?

The first option, referred to as "customary" above, is to give a contractor a fixed schedule and budget while enabling the content to be variable. This is achieved through prioritization of the backlog to ensure that if there are capabilities or features that are not completed within the allocated time and cost, they are of lower priority. In this case the government is essentially buying "development capacity" for some specified period or multiple periods. This approach is similar to a "time and materials" contractual approach. A variant of this is discussed in the *TechFAR Handbook*, which recommends an initial period (on the order of six months) using the time and materials approach to establish a velocity for the Agile team and then moving the contract to firm-fixed-price (FFP) periods [OUSD 2014]. This approach lends itself to software-intensive systems and may not be appropriate for systems where hardware development is involved.

Another approach, sometimes used by the National Geospatial-Intelligence Agency, is to use a multi-year indefinite delivery, indefinite quantity (IDIQ) contract with task awards set up on a short fixed schedule (on the order of six months). A SOO describing the desired functionality (in terms of a prioritized backlog) is provided to the contractor. The contractor then proposes a PWS to address the prioritized backlog for the upcoming task order period. It is somewhat expected that not all low-priority backlog items will be completed before the end of the task order. Near the end of the existing task order, the next task order is developed with a revised and reprioritized backlog. This model enables the National Geospatial Agency (NGA) to be very responsive to new mission needs.

In highly complex systems or systems requiring hardware development, such as is often the case with defense systems, a hybrid model may be more appropriate. In this case, the high-level capabilities are fixed by the government, the features and their associated priority (prioritized backlog) are developed jointly between the government and the contractor, and the stories (the "how") are totally within the contractor's control. Changes to capabilities or features usually require a contract modification (if total scope is going to increase) or trading (reducing) other scope so as not to impact the overall cost and schedule of the program. In this case, a FFP-type development contract is not normally appropriate. Cost plus fixed-fee (CPFF) or a similar contract vehicle would be applicable.

Parametric Bidding: Waterfall Actuals Don't Necessarily Apply

Historical actuals and other metrics are key to the cost estimate. The government must understand the metrics/measures supporting the contractor's estimate to accurately assess a contractor's cost bid. Waterfall actuals (and to a large extent waterfall metrics) don't necessarily carry over to an Agile development program. Actuals reflect the particular skill mix within specific disciplines and efficiencies that often correlate to different development stages in the waterfall model. In an Agile program, the focus is on optimizing the skill mix and defining metrics that provide overall value to the customer. As such, the staffing mix and staffing curves will likely look quite different when comparing Agile and waterfall programs.

When looking at historical actuals from previous Agile programs, the higher the percentage of the program that was performed using the Agile methodology, the more accurately they can be assessed and the less likely they will be broken out by discipline. If the breakdown by discipline can be determined, it will likely be inconsistent with similar programs executed using the waterfall model.

Using Story Points for Bidding

First the authors must warn that using story points for bidding is *not* a good practice unless there is a standard and normalized process to determine story point metrics. This is because story points are typically a sizing measure unique to each team and are not used to compare productivity across teams but rather for the team to estimate work and determine if they are improving.

However, if monetized story points are the expected work unit in the RFP, the historical actuals must be examined from the perspective of what the contractor is using as a basis for the expected productivity. The most reliable estimates will come from existing Agile teams with historical data on their velocity for the equivalent type of development. The next most reliable estimate would come from a prior Agile program. If no existing or prior Agile programs are cited, the government could use the approach cited previously, which is to start up the program using a time and materials approach to "set" the expected velocity for the Agile teams per the TechFAR. In these cases, remember to account for slower velocities during the start-up of the program, as the teams require some initial time to ramp up.

Schedule

Unlike a traditional schedule, which is often organized in a serial fashion following the waterfall process (i.e., design then code then test) by discipline, an Agile integrated master schedule should be focused on the development of system capabilities and features. Known dependencies between features should be clearly identified. Completion criteria for each feature should be clearly understood and the associated progress towards working capabilities (which is the overall goal) should also be easily discernible. An Agile schedule will employ a rolling baseline where details are developed two events forward. This just-in-time planning approach defers commitment-level decisions until they are ready to execute, preserving trade space and agility to respond to change. The temptation to include stories in the IMS should be *avoided*. Capturing that low level of detail overly constrains the contractor and likely ensures that the IMS will be in a constant state of change. Stories should be managed in the Agile toolset at the working-team level. As such, these

tools will provide the quantifiable backup data (sometimes known as inchstones) used to update the IMS.

4.6 Negotiation

While the Agile Manifesto [Agile Alliance 2001] values "customer collaboration over contract negotiation," this does not suggest that contract negotiation is not required, but rather that the contract as negotiated should enable Agile behavior, including making allowance for refinement and prioritization of requirements by the government during program execution, including the possibility that some low-priority requirements will not be completed [OUSD 2014].

To apply Agile management processes effectively, the government and contractor must discuss and clearly define a few key elements of the contract.

The scope of the project must be defined clearly, so that it's clear when an engineering change proposal (ECP), request for change (RFC) or new task order (for IDIQ type contracts) is required, but not in such detail that swapping features or changing focus within the scope of the project is impeded. This requires a change in mindset by both parties. First, there must be greater focus on defining the boundaries of the project scope, so that it's obvious when a proposed change is out of scope. Second, the parties must recognize that not every possible feature, function, or idea within the scope of the project will be done, and develop a plan to ensuring that government needs are met without the contractor being required to do more work than was bid and planned.

The *TechFAR Handbook* recommends developing a high-level product or system vision to describe the anticipated scope of the overall acquisition along with tactical documents like a SOO and PWS for each task order, increment, or delivery to share short-term expectations and define sign-off criteria. Since these tactical documents cover short time periods (8 to 26 weeks are common), the focus is on streamlined formats that deliver high-value information. In a typical example, the government uses the SOO to specify the features (part of a capability) to be developed in this period, the non-functional requirements to be met, and a (minimal) set of CDRLs. The contractor responds with a PWS that focuses on the extent of the expected implementation and how the customer expectations will be verified.

When negotiating a contract for Agile development, focus on defining the following:

- identification and management of change
- in-scope tradeoffs versus ECPs/RFCs
- in-scope trade considerations
  - What level do you trade? Features, capabilities, or both?
  - Who can approve trades?
  - How are trades documented?
  - What are the sell-off expectations and requirements for traded items?
- setting the stage for the first program increment
- lessons learned and pitfalls to avoid

See Agile Government Leadership's *Agile Government Handbook* for additional references and recommendations [Agile 2016].

5 Changes in Program Execution

While this section is not specific to RFP development, it provides additional information that the government needs to understand as it frames its RFP.

5.1 Start-Up (Kickoff)

The start-up for an Agile program can be quite different from that of a traditional program. The individual government program offices may also want to include an explanation of this activity within Section L under the Agile development management plan.

Because Agile practices focus on small increments of work executed in tight time boxes and/or employ a Kanban pull process, start-up on programs using such methods tends to be austere, intense and focused on getting the Agile machine up and running. A clear set of expectations and plan for start-up is essential, as the output of start-up provides initial input for Agile (usually scrum) operations, and any start-up tasks that are not completed will detract from the capacity of the teams. The following topics should be included in the Agile development management plan:

- reviews
- development and infrastructure runway (What has to be in place before the scrum teams can work?)
- initial backlog grooming (population of the backlog and subsequent prioritization)
- cadence (battle rhythm on which the program operates)
- subcontractors and suppliers and their interaction with the Agile cadence
- managing the technical baseline

5.2 Measures/Metrics

Agile methods are seen by some as an effective means to shorten delivery cycles and manage costs for the development and maintenance of major software-reliant systems in the DoD. If these benefits are to be realized, the personnel who oversee major acquisitions must be conversant in the measures/metrics used to monitor these programs.

There is generally a reliance on a time-box approach in place of the traditional phase-gate approach to managing progress. This means that the development organization is working to maximize valuable work performed within strictly defined time boundaries. The schedule remains fixed, and the work to be performed is the variable. If Kanban methods are used in conjunction with Agile methods, then the program needs to be aware of a synchronization activity that must occur on release boundaries at a minimum. This is in contrast to many traditional approaches where the period of performance may be subject to negotiation, while attempting to fix the scope of the work product(s) to be delivered.

A focus on delivering usable capabilities replaces the scheduled delivery of interim work products and the ceremonies that typically focus on milestones. This means that the customer (or customer representative) should expect to see the working product frequently. The planned interactions and demonstrations permit a disciplined process for changing the requirements and shaping the final form of the delivered product. Ideally, acceptance testing happens iteratively throughout development, rather than at the end.

Quality and customer satisfaction is an area where Agile methods provide greater opportunity for insight than traditional development approaches tend to allow. Delivery and progress monitoring is the area in which perhaps the greatest difference is seen in Agile development, compared to traditional approaches. The frequent delivery of working (potentially shippable) products renders a more transparent view of progress than is typically apparent through examination of intermediate work products. Demonstrations of system capabilities allow early opportunities to refine the final product, and ensure that the development team is moving toward the desired technical performance—as opposed to just asking whether they will complete on schedule and within budget [Hayes 2013].

In many instances, earned value management (EVM) metrics are required. These are typically based on a plan that is completed up front and rarely changes [Agile Alliance 2001]. If learning occurs that changes the plan, both revision of the plan and the EVM milestones require an arduous change process. Agile implementations, by definition, allow and even embrace change. An Agile response to EVM requirements could provide considerable amounts of data taken from a typical Agile implementation, but that data would need to be interpreted and translated to compliance metrics. The RFP should request information on how that translation would be accomplished and, if missing, the government should request that information. In addition, the government personnel need to have an understanding of the Agile metric data and what it does or doesn't mean [Hayes 2013].

Agile and Earned Value Management: A Program Manager's Desk Guide states "Agile, as a product development methodology, can exist within the disciplines required for EVMS compliance." There are currently no DoD standards for Agile methodology metrics. Metrics such as velocity and burn-down/burn-up must be agreed on by the contractor and the government PMO unless they are only used internally by the contractor teams. Agile metrics should provide status that supports and aligns with similar EVM metrics [PARCA 2016].

Be prepared to mine and effectively use the metrics data that naturally occur in typical Agile teams. Keep in mind that the metrics from one Agile team cannot be compared to those of another team as the type and nature of both the work performed and skills of the teams are typically quite different [Hayes 2013].

The increased number of iterations that result in both a product to evaluate and the exercise of defect detection and testing activities actually makes some data collection for measurement easier in Agile settings—*but* the typical traditional measures used for reporting progress need adaptation to provide productive views of progress in Agile projects [Foreman 2014].

5.3 WBS Discussion

The WBS is one of the key artifacts in an RFP as it is one of the primary vehicles for communication between the government and the contractor. An appropriately constructed WBS is critical for successfully structuring and tracking the progress of the program. For an Agile program, providing a standard waterfall WBS can be the cause of significant unnecessary work on the part of both the government and the contractor. Although MIL-STD-881C is required to facilitate cost and earned value reporting, it does also allow for a structure that is both compliant and geared towards an Agile program [PARCA 2016]. In lieu of the "standard" waterfall-based workflow hierarchy most commonly associated with 881C, a capabilities- and features-based structure should be used. See *Agile and Earned Value Management: A Program Manager's Desk Guide* for a more thorough discussion including a generic example.

Figure 6 and Figure 7 provide two different views of WBS, one from the waterfall perspective and one from an Agile perspective. Notice that the constructs of the WBS reflect the constructs of the method as was depicted in Figure 1 and Figure 2.

1.1	Prime Mission Subsystem					
1.1.1		Computer Software Configuration Item A				
1.1.1.1			CSCI Requirements Analysis			
1.1.1.2			CSCI Design			
1.1.1.3			CSCI Code and Unit Test			
1.1.1.4			CSCI Integration and Test			
1.1.2		Computer Software Configuration Item B				
1.1.3	High level Integration, Assembly, Test, and Checkout					
1.1.4	...					

Figure 6: Software (SW) Development MIL-STD 881C Appendix K WBS Breakout (Traditional Waterfall) [PARCA 2016]

1.1	Prime Mission Subsystem					
1.1.1		Capability A				
1.1.1.1			Feature A1			
1.1.1.2			Feature A2			
1.1.1.3			Feature A3			
1.1.1.4			Feature A4			
1.1.2		Capability B				
1.1.3	High level Integration, Assembly, Test, and Checkout					
1.1.4	...					

Figure 7: Possible Agile SW Development MIL-STD-881C WBS Breakout (Agile Capability-Based) [PARCA 2016]

5.4 Change Management

Effective change management should be as simple as adding to the product backlog. It seldom is. Change management becomes an issue when the developer contends the change is out of scope of his or her original estimate and negotiated price. One possible solution is a written understanding between the developers and the acquirers that every change introduced may displace an existing backlog item thus keeping the initial planned effort consistent. This understanding must take into account what level of change is being considered. By level of change, the authors are referring to level of abstraction of the requirements—capability/feature level versus story level. The highest level of requirements abstraction that establishes the program's scope should be under formal change control and assessed for scope impact. The lower levels of requirements abstraction (e.g., stories) should be able to be changed by the product owner without needing a contract modification.

It is important to define how changes will be handled in the RFP and how the resulting agreement will be documented in the contract.

5.5 Risk Assessment

The underlying risk management process will not change substantially under an Agile development/acquisition program. However, the iterative nature of an Agile program will address risks on a more frequent basis as the iterative, incremental work reduces the risk throughout the program.

The projects will benefit from seeing some risks mitigated more quickly. Mitigation activities that come out of the risk management process are fed into the backlog and prioritized in the same manner as other requirements, with the stakeholders deciding the level of risk with which they are comfortable.

The RFP should request that the developer identify the existing risk management processes to ensure they meet the project's needs.

5.6 Development Methodologies

Many organizations still employ waterfall development methods for situations that have well-known, stable requirements. Hardware development is also more often implemented using waterfall. Alternatively, teams developing applications that must be highly flexible and quickly updated to changing environments can find Agile methods more conducive to success. Other programs are not completely black or white in their software and hardware development methodologies: they benefit by employing a hybrid approach that takes advantage of the strengths of both Agile and waterfall development.

Before creating an RFP, the government must decide within its acquisition strategy what development method is acceptable for its program. This section compares and highlights the advantages and disadvantages of these three development models: waterfall, Agile, and the Agile/waterfall hybrid model. Keep in mind that even within each of these models, there is a great deal of variation with no black and white answers.

5.6.1 The Waterfall Model

The waterfall model is an approach for developing software that breaks a project into sequential phases. A program moves to the next phase only when the preceding phase completes and is verified. (See Figure 1 in Section 4.1.3.) Figure 8 presents the waterfall phases and highlights some potential waterfall advantages and disadvantages.

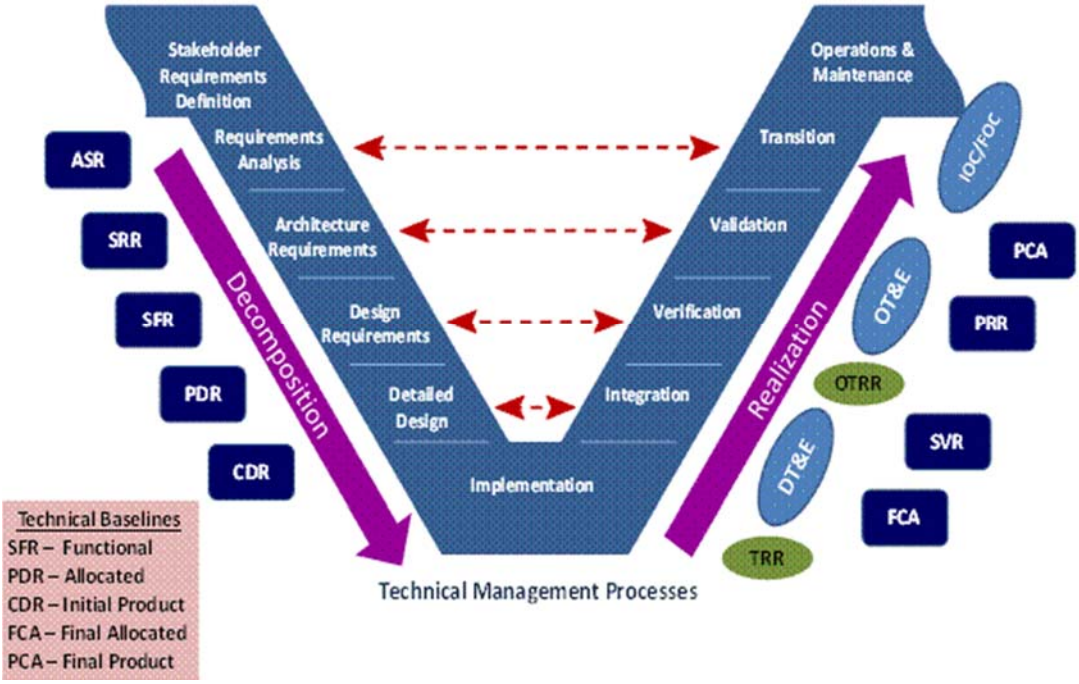

Figure 8: Engineering V Model[19]

Potential Advantages of the Waterfall Software Development Model

- Waterfall development is typically suitable for hardware projects and where software requirements are well understood and stable. Note that hardware Agile development may be less mature than Agile software development, but the principles still hold true across hardware, software, and systems.
- The waterfall model is well known within the government, thoroughly defined, and focuses on schedule-driven activities by project phase.
- It is a very regulated model in that each phase has specific deliverables and checkpoint review processes.
- Documentation and developmental artifacts are meticulously reviewed.

[19] See *AFLCMC Systems Engineering Technical Review (SETR) Guide* Version 1.4.

Potential Disadvantages of the Waterfall Software Development Model

- The waterfall model is not typically suitable for projects where requirements are at risk of emerging or changing. There are many examples of programs where the requirements were thought to be well understood but still suffered at the hands of the waterfall model.
- There is infrequent communication between the stakeholders, end users, and the development teams.
- Development projects typically take longer to produce a useable product than similar Agile or hybrid programs.
- Because system capabilities are not normally demonstrable until late in the lifecycle, isolating and fixing defects is more difficult and costly.

Keep in mind the above advantages and disadvantages are what is typically thought of for the waterfall model. There is no black and white to these two sets of ideas; these advantages and disadvantages must be considered based on the specific program environment.

Note: The founder of the waterfall development methodology, Winston Royce, warned of employing this methodology for large software systems back in 1970: "I believe in this concept, but the implementation is risky and invites failure. The testing phase, which occurs at the end of the development cycle, is the first event for which timing, storage, input/output transfers, etc. are experienced, as distinguished from analyzed. If these phenomena fail to satisfy the various external constraints, then invariably a major redesign is required. Either the requirements must be modified, or a substantial change in the design is required. In effect, the development process has returned to the origin and one can expect up to 100-percent overruns in schedule and/or cost" [Royce 1970].

5.6.2 The Agile Model

Wikipedia defines the pure Agile model as "a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams." The scrum model is presented in Appendix A as it is the most commonly used Agile method.

The following presents generalizations of the potential advantages and disadvantages of the Agile methods. As with waterfall, there is no black and white answer. The implementation of Agile in a specific environment is dependent on the issues that are being solved within an UURVE environment.

Potential Advantages of the Agile Software Development Model

- It is typically good for programs with high requirements volatility and frequent changes to the user interface.

- It facilitates high stakeholder, product owner, scrum master, and development team collaboration.[20]
- Generally, there is increased software quality as working software is continuously developed, re-tested, and demonstrated.
- Emergent requirements and developmental priority fluctuations are more easily accommodated than in the waterfall process.
- Continuous improvement to the product and process is facilitated through refactoring and sprint retrospectives.
- Release times are shortened for deliverable working code; the overall product development lifecycle can shrink.

Potential Disadvantages of the Agile Software Development Model

- Many DoD acquisition program offices do not fully understand or embrace Agile development methodologies.
- Contracting often locks in scope and budget up front, making pure Agile somewhat difficult to implement.
- Technical reviews and audits often require iterative execution, as opposed to one-time waterfall events.[21]
- It can be difficult to manage partners employing waterfall development methodologies.[22]
- It requires training and experience in Agile vocabulary, roles, ceremonies, tools, and metrics collection and analysis. It requires adjustments to program-level activities.
- There is a challenge of learning and implementing the new product owner role (or equivalent in non-scrum Agile implementations). This is only an initial disadvantage and should be resolved once the personnel are trained, gain experience, and use this as the new way of doing business.

5.7 Mixing Waterfall and Agile Methodologies

There are several factors that may lead to the use of mixed development methodologies on a program. They could include the maturity level of Agile as applied to systems engineering and hardware engineering, the Agile maturity level of an organization, or the size/complexity of a program. Another possibility is that a company may desire to take advantage of Agile software development methodologies (to reduce cost) while adhering to a traditional (waterfall) SOW. Regardless, there is not a blanket recommendation for or against the use of such mixed methodologies. However, it should be noted that mixing methodologies is not without pitfalls and may be disruptive to both the government's program office and the companies with established processes.

---

[20] Product owner and scrum master are terms usually associated with the scrum Agile method. These terms may not be associated with other forms of Agile even though the roles performed by those positions may be represented.

[21] Some may consider this an advantage as there is early insight into technical data.

[22] Managing Agile and waterfall teams working together requires coordinated planning and synchronization points.

As Agile software development methodologies are generally more mature than Agile systems engineering (SE) or Agile hardware processes, one notable model combines traditional waterfall phases, process reviews, and artifact collection with iterative Agile development and functional releases. Requirements analysis and design are performed in a waterfall fashion with code development of evolving functionality being iteratively demonstrated to the product owner and other stakeholders.

In additional to potential cost savings, leveraging a combined waterfall and Agile approach may facilitate incremental buy-in by a program manager cautious of converting directly from waterfall to Agile acquisition. While potentially reaping some of the efficiencies often seen in Agile software development, this model still suffers from the problems inherent with completely defining the solution (captured in requirements) before coding even begins. This is only one example of many possible hybrid approaches.

5.7.1 The Changing Landscape of Software Acquisitions—Employing Hybrid Agile Models

Agile author and software expert Chuck Cobb proposes a hybrid approach consisting of a plan-driven "envelope" at the macro level, coupled with a micro-level Agile approach for use case (UC) development and test from the product backlog. This envelope makes it less difficult to provide cost and schedule estimates for a hybrid Agile acquisition program. His advice is that software development using Agile or waterfall is not a binary decision, noting: "Seeing Agile and plan–driven approaches as mutually exclusive is an example of Binary Thinking. There are lots of ways to blend Agile and plan-driven approaches together in the right proportions to fit a particular software and hardware situation; it just requires more skill to do that" [Cobb 2013].

5.7.2 DoDI 5000.02 Facilitates Hybrid Agile Software and Waterfall Hardware Development

Acquisition instruction DoDI 5000.02 proposes six acquisition program models as a starting point for program-specific planning [DoD 2015]:

- Model 1: Hardware Intensive Programs
- Model 2: Defense Unique Software Intensive Programs
- Model 3: Incrementally Fielded Software Intensive Programs
- Model 4: Accelerated Acquisition Programs
- Model 5: Hybrid Program A (Hardware Dominant Programs)
- Model 6: Hybrid Program B (Software Dominant Programs)

The hybrid acquisition model shown below from 5000.02 is a model depicting how a major weapons system combines hardware development as the basic structure with a software intensive development that is occurring simultaneously with the hardware development program. In a hardware intensive development, the design, fabrication, and testing of physical prototypes may determine overall schedule, decision points, and milestones, but iterative software development and releases often dictate the pace of program execution and must be tightly integrated and coordinated with hardware development decision points.

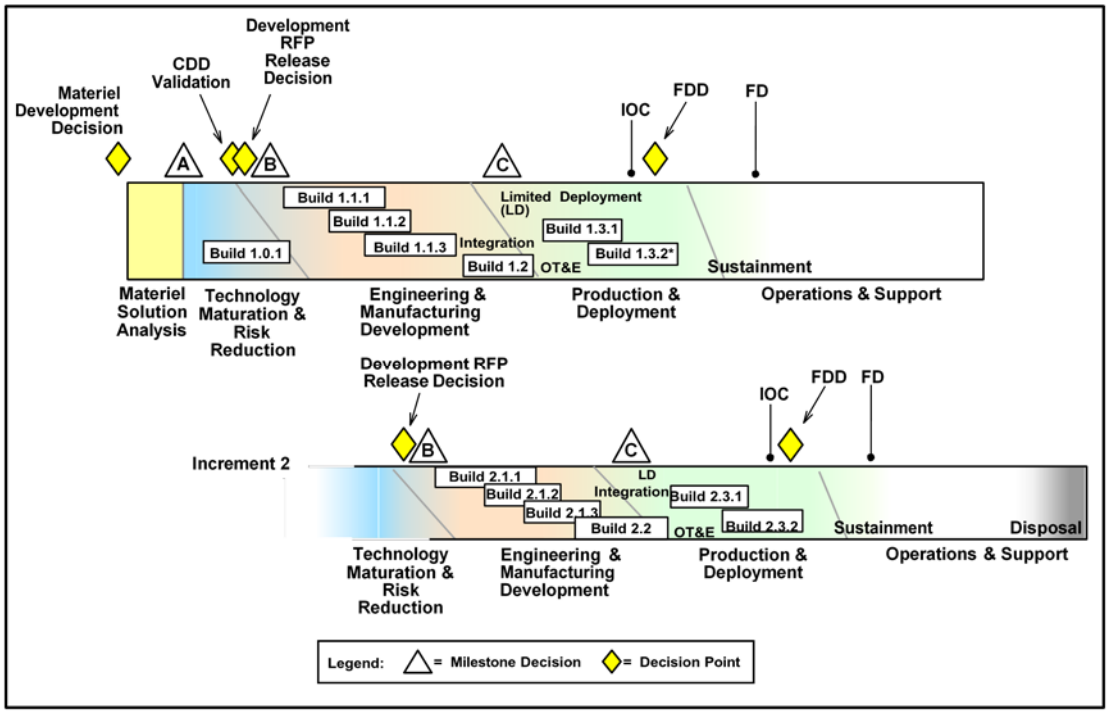

Figure 9: DoD 5000.02 Hybrid Model [DoD 2015]

Potential Advantages of the Hybrid Model

- It is good for integrated hardware/software product development teams: hardware (waterfall) and software (Agile or Agile hybrid).
- It is good for software-intensive product development with IEEE 15288.2[23] checkpoints such as PDR and CDR, making it easier to meet audits, standards, and regulatory requirements than with a purely Agile development.
- The model facilitates presentation layer changes to components implementing the user interface and managing user interaction.
- The model facilitates waterfall requirements/design with iterative development, test, and demonstration (Agile).[24] However, other hybrid models may include SE in the Agile methodology side.
- It facilitates frequent presentation layer changes such as thin (mobile) and rich (stand-alone) client applications.
- A hybrid approach facilitates increased emphasis on documentation as a deliverable.
- A hybrid approach facilitates increased emphasis on automated test cases and test scripts as iterative deliverables.

---

[23] IEEE-Std-15288.2 is the 2014 IEEE *Standard for Technical Reviews and Audits on Defense Programs*

[24] See https://www.scrumalliance.org/community/articles/2010/october/negotiating-scrum-through-a-waterfall

- Integration test commences earlier in a hybrid Agile/waterfall software development lifecycle (SDLC), and Agile tool integration is gradual.
- It's easier to gain buy-in from program offices with an internal culture of and propensity for waterfall development.
- Hybrid models merge strategic planning and cost management with iterative product owner value, balancing the need for control with adaptability to changing requirements.

Disadvantages of the Hybrid Model

- It requires extensive knowledge of waterfall and Agile models to implement.
- Project planning and resultant scheduling are more difficult—such as differing EV structures between the Agile/waterfall portions, completion of system capabilities in software (Agile) versus incremental progress in waterfall, and others.

5.8 Bidder's Library

The contents of the bidder's library should not vary from the types of data typically included on a traditional program RFP.

6 Conclusion

The authors have provided information that can be employed to write an RFP that allows the use of Agile methods. Many of the items in the RFP will remain unchanged. However, the implementation of certain practices, such as metrics and cadence, will change.

This report is not meant to be prescriptive but rather to be used as a resource to help answer some of the questions RFP writers will encounter when they endeavor to write an RFP allowing the implementation of Agile methods. Undoubtedly, once a larger majority of government program office personnel have experience in this arena, this report will require a much needed update.

Appendix A - Agile Fundamentals

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

The guiding principles behind the Agile Manifesto are as follows:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity—the art of maximizing the amount of work not done—is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly [Agile Alliance 2001].

Scrum is the most commonly used Agile method. Its lifecycle is depicted in Figure 10.

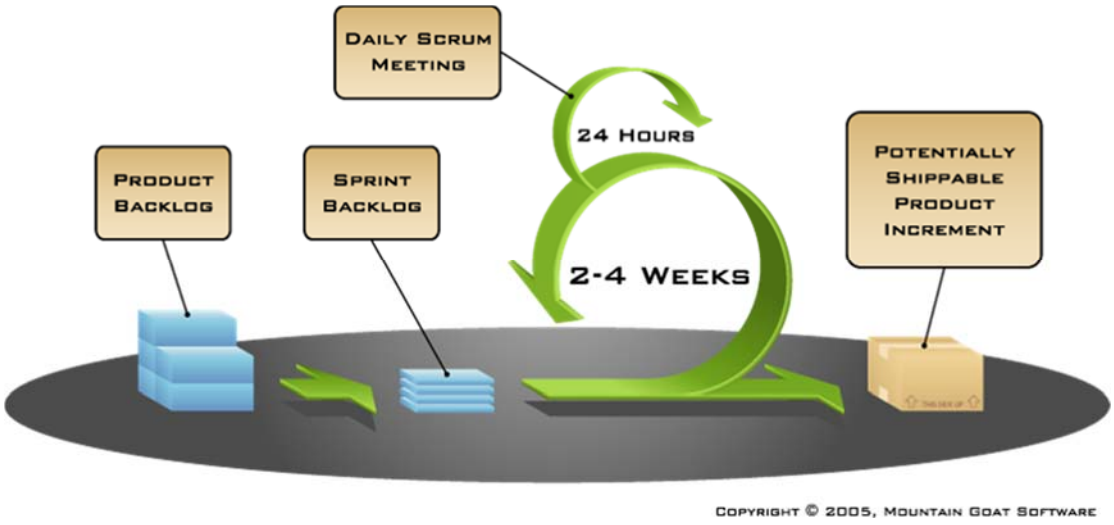

Figure 10: The Agile Software Development Lifecycle (reproduced with permission from Mountain Goat Software)

Appendix B - Marbach Updated Principles to the Agile Manifesto

Original Agile Manifesto Principles [Agile Alliance 2001]	Marbach Updated Principles [Marbach 2015]
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	First satisfy the customer through early and continuous delivery of valuable capabilities.
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Plan for evolving requirements, and retain as much flexibility as is valuable throughout development, especially when change leads to a competitive advantage.
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	Deliver working capabilities frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
Business people and developers must work together daily throughout the project.	Business personnel, customers, or their advocates, and implementers must work together daily throughout the project.
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	No change.
`primary measure of progress agile processes promote sustainable development the sponsors`	The most efficient and effective method of conveying information to and within a delivery team is personal conversation.
Working software is the primary measure of progress.	Working capabilities are the primary measure of progress.
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Agile processes promote sustainable delivery. The sponsors, developers, and users should be engaged and able to maintain a constant pace to the project completion.
Continuous attention to technical excellence and good design enhances agility.	No change.
Simplicity—the art of maximizing the amount of work not done—is essential.	Simplicity (“the art of maximizing the amount of work not done”) is essential, especially within the implementation team. A truly Agile development project does not force artificial reporting and process requirements on the implementation team.
The best architectures, requirements, and designs emerge from self-organizing teams.	The best architectures, requirements, and designs emerge from self-organizing teams, based on a minimal set of guiding principles.
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	No change.

Appendix C - Acronyms

ADM	Acquisition Decision Memorandum
ASR	Alternative Systems Review
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CPFF	Cost Plus Fixed-Fee
CSOW	Contractor Statement of Work
CWBS	Contract(or) Work Breakdown Structure
DDIT	Design-Develop-Integrate-Test
DoD	Department of Defense
DoDD	Department Of Defense Directive
DoDI	Department Of Defense Instruction
DR	Deficiency Report or Discrepancy Report
DT&E	Developmental Test & Evaluation
ECP	Engineering Change Proposal
EV	Earned Value
EVM	Earned Value Management
FAR	Federal Acquisition Regulations
FFP	Firm-Fixed-Price
F.I.R.E.	Fast, Inexpensive, Restrained, And Elegant
GAO	Government Accounting Office
ICD	Interface Control Document

IDIQ	Indefinite Delivery, Indefinite Quantity
IMS	Integrated Master Schedule
INCOSE	International Council on Systems Engineering
MIL STD	Military Standard
NDIA	National Defense Industrial Association
NGA	National Geospatial Agency
OT&E	Operational Test and Evaluation
OUSD	Office of the Under Secretary of Defense
PARCA	Performance Assessments and Root Cause Analyses (PARCA)
PBA	Performance-Based Acquisition
PDR	Preliminary Design Review
PMO	Program Management Office
PWS	Performance Work Statement
RFC	Request for Change
RFP	Request for Proposal
SDLC	Software Development Lifecycle
SE	Systems Engineering
SEI	Software Engineering Institute
SFR	System Functional Review
SOO	Statement of Objectives
SOW	Statement of Work
SRR	System Requirements Review
SVR	System Verification Review
SW	Software

TPM	Technical Performance Measurement
TRR	Test Readiness Review
UC	Use Case
UURVE	Unpredictability, Uncertainty, Risk, Variability, and Evolution
WBS	Work Breakdown Structure

Appendix D - Glossary

Glossary terms as defined in a variety of industry sources [CC Pace 2016, Scrum Alliance 2016, Scrum Guides 2016, SolutionsIQ 2016].

Agile	The name coined for the wider set of ideas that scrum falls within; the Agile values and principles are captured in the Agile Manifesto.
Backlog Item	See Product Backlog Item.
Burndown	See Sprint Burndown, Product Burndown.
Capability	A customer-required ability of the system that provides value to the customer.
Daily Scrum	A short daily team meeting to share progress, report impediments and make commitments.
Done	Also referred to as “Done” or “Done Done” this term is used to describe a product increment that is considered releasable; it means that all design, coding, testing, and documentation have been completed and the increment is fully integrated into the system. Done also refers to holistic completion of smaller components of an Agile system including user stories, features, and capabilities that may not be releasable.
Empiricism	Empiricism is the principle of “inspect and adapt,” which allows teams or individuals to try something out and learn from the experience by conscious reflection and change.
Epic	A very large user story that is eventually broken down into smaller stories; epics are often used as placeholders for new ideas that have not been thought out fully. (This definition is used by the SAFe scaling framework.) See also, Capability.
Estimation	The process of agreeing on a size measurement for the stories in a product backlog.
Feature	A part of a capability that can be completed within a program increment; it also has business value, is estimable and testable.
Iteration	See Sprint.
Non-Functional Requirements (NFR)	Constraints on functional items in the backlog. NFRs include standards, reliability, maintainability, availability, and performance. NFRs are qualities of the delivered items and cannot be delivered by themselves.
Planning	See Sprint Planning
Process	Simply the way someone works. Everyone has a process. It can be pre-defined, empiric, or merely chaotic.
Product Backlog	A prioritized list of stories that are waiting to be worked on.
Product Backlog Item	Any item that is on the backlog list, which will include user stories, epics, and possibly technical stories to deal with technical debt, etc.
Product Owner	The person who holds the vision for the product and is responsible for maintaining, prioritizing, and updating the product backlog.
Release	The term used to describe a concrete time box or cadence used to complete features. Release duration can vary, but is typically three to six months. Many practitioners use the release cadence as their rolling wave planning period.

Retrospective	A session where the team and scrum master reflect on the process and make commitments to improve.
Roadmap	Capabilities and their associated features mapped to program increments and software builds/deliveries; the dependencies between the features are also captured. A roadmap is a living artifact, which is less accurate the farther in the future it projects.
Scrum Master	A servant leader to the team, responsible for removing impediments and making sure the process runs smoothly so the team can be as productive as possible.
Self Organization	The principle that those closest to the work best know how to do the work, so set clear goals and boundaries and let them make all tactical and implementation decisions.
Sprint	A term frequently used interchangeably with iteration to describe a concrete time box or cadence used to complete stories. Sprint duration can vary, but is typically two to four weeks.
Sprint Burndown	A visible chart that indicates on a daily basis the amount of work remaining in the sprint.
Sprint Planning	A meeting between the team and the product owner to plan the sprint and arrive at an agreement on the commitment.
Stakeholder	Anyone external to the team with an interest in the product being developed.
Story	A backlog item usually using the template form: as a [user], I want [function], so that [business value]; also known as a user story. Stories contribute to the completion of a feature and can be completed within a single sprint.
Team	The development team, responsible for committing to work and delivering and driving the product forward from a tactical perspective.
Team Member	Any member of the team, including developers, testers, designers, writers, graphic artists, and database admins.
Time Boxing	Setting a duration for every activity and having it last exactly that (i.e., neither meetings nor sprint are ever lengthened—ever).
Velocity	The rate at which a team completes work, usually measured in story points.
Vision Statement	A high-level description of a product that includes who it is for, why it is necessary, and what differentiates it from similar products.
Work in Progress (WIP)	Limiting the work in progress is a principle used in Kanban to encourage quality and completion of work, among other things.

References

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

[Agile 2016]
Agile Government Leadership. *Agile Government Handbook*. Agile Government Leadership. 2016. http://www.agilegovleaders.org/wp-content/uploads/2016/02/AgileGovernmentHandbook-1.pdf

[Agile Alliance 2001]
History: The Agile Manifesto. *Agile Alliance*. 2001. http://agilemanifesto.org/history.html

[Ambler 2013]
Ambler, Scott. *Agile Testing and Quality Strategies: Discipline Over Rhetoric*. 2013. http://www.ambysoft.com/essays/agileTesting.html

[CCPace 2016]
CCPace. *Glossary of Scrum Terms*. 2016. http://www.ccpace.com/asset_files/ScrumGlossary-TermsNEW.pdf

[Cobb 2013]
Cobb, Charles G. *Managing Agile Development–Making Agile Work for Your Business*. Outskirts Press. 2013.

[DoD 2007]
U.S. Department of Defense. *DoDD 5000.01 The Defense Acquisition System*. Nov. 20, 2007. http://www.dtic.mil/whs/directives/corres/pdf/500001p.pdf

[DoD 2015]
U.S. Department of Defense. *DoDI 5000.02 Operation of the Defense Acquisition System*. Jan. 7, 2015. https://acc.dau.mil/CommunityBrowser.aspx?id=716926

[Dove 1993]
Dove, R. Lean and Agile: Synergy, Contrast, and Emerging Structure. *Defense Manufacturing Conference '93*. Nov. 29-Dec. 2, 1993.

[Foreman 2014]
Foreman, John. *Report from the Trenches: Agile Adoption in the Federal Government*. Association for Enterprise Information Agile in Government Conference. June 2014. http://www.afei.org/PE/4A02/Pages/default.aspx

[GAO 2008]
U.S. Government Accountability Office. *Defense Acquisitions: Assessments of Selected Weapon Programs* (GAO-08-467SP). 2008. http://www.gao.gov/new.items/d08467sp.pdf

[Hayes 2013]
Hayes, William; Miller, Suzanne; Lapham, Mary Ann; Wrubel, Eileen; & Chick, Timothy. *Agile Metrics: Progress Monitoring of Agile Contractors*. CMU/SEI-2013-TN-029. Software Engineering Institute, Carnegie Mellon University. 2014.
http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=77747

[IEEE 2016]
IEEE. *IEEE Standard for Technical Reviews and Audits on Defense Programs* (IEEE Standard 15288.2-2014). 2016. https://standards.ieee.org/findstds/standard/15288.2-2014.html

[INCOSE 2016]
Agile Systems & S.E.: Mission Objectives. *INCOSE*. 2016. http://www.incose.org/ChaptersGroups/WorkingGroups/Transformational/agile-systems-se

[Lapham 2014]
Lapham, Mary Ann; Bandor, Michael; & Wrubel, Eileen. *Agile Methods and Request for Change (RFC): Observations from DoD Acquisition Programs*. CMU/SEI-2013-TN-031. Software Engineering Institute, Carnegie Mellon University. 2014.
http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=77732

[Marbach 2015]
Marbach, Phyllis; Rosser. Larri; Gundars, Osvalds; & Lempia, David. Principles for Agile Development. Volume 25. Issue 1. Version of Record online: 28. *INCOSE International Symposium*. October 2015.
http://onlinelibrary.wiley.com/doi/10.1002/j.2334-5837.2015.00079.x/pdf

[Navy 1998]
U.S. Navy. Acquisition Packages: Proposal Evaluation Plan Template. *Acquisition Guide*. 1998.
http://www.navair.navy.mil/nawctsd/Resources/Library/Acqguide/acqpack.htm

[Nidiffer 2014]
Nidiffer, Kenneth; Miller, Suzanne; & Carney, David. *Potential Use of Agile Methods in Selected DoD Acquisitions: Requirements Development and Management*. CMU/SEI-2013-TN-006. Software Engineering Institute, Carnegie Mellon University. 2014.
http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=89158

[OUSD 2014]
Office of the Undersecretary of Defense. *TechFAR Handbook for Procuring Digital Services Using Agile Processes*. 2014. https://playbook.cio.gov/techfar/

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

[PARCA 2016]
Office of the Undersecretary of Defense, Performance Assessments and Root Cause Analyses. *Agile and Earned Value Management: A Program Manager's Desk Guide.* 2016. http://www.acq.osd.mil/evm/docs/PARCA%20Agile%20and%20EVM%20PM%20Desk%20Guide.pdf

[Royce 1970]
Royce, Winston. *Managing the Development of Large Software Systems,* IEEE WESCON Proceedings. TRW Corp. August 1970.

[Scrum Alliance 2016]
Scrum Alliance. *Scrum Alliance*. 2016. https://www.scrumalliance.org/

[Scrum Guides 2016]
The Scrum Guide. *Scrum Guides*. 2016. http://www.scrumguides.org/scrum-guide.html

[SolutionsIQ 2016]
Agile Glossary. *SolutionsIQ*. 2016. http://www.solutionsiq.com/agile-glossary/

[Ward 2014]
Ward, Dan. *F.I.R.E., How Fast, Inexpensive, Restrained, and Elegant Methods Ignite Innovation*. HarperCollins Books. 2014.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	November 2016	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
RFP Patterns and Techniques for Successful Agile Contracting	FA8721-05-C-0003

6. AUTHOR(S)

Mary Ann Lapham, Keith Korzec, Larri Ann Rosser, Greg Howard, Steven Martin, Michael Ryan, Thomas E. Friend, John H. Norton III, Peter Capell

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2016-SR-025

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

11. SUPPLEMENTARY NOTES

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

13. ABSTRACT (MAXIMUM 200 WORDS)

Increasing budget constraints and emphasis on fielding capability faster have led the U.S. Department of Defense (DoD) and other federal entities to pursue the benefits of Agile software development – reduced cycle times, flexibility to adapt to changing conditions and user needs - that software development practitioners have achieved in the commercial market.

This report is written by the National Defense Industrial Association's System Engineering Agile Working Group to provide information on request-for-proposal (RFP) patterns and techniques for successful Agile contracting that can and have been used for contracts seeking to employ Agile methods. This report is intended to support the writers of RFPs in bringing Agile concepts into programs at the earliest possible time, providing examples of the kinds of language that will affect the foundations of contracts on which programs rely.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Agile, government, RFP, DoD	56

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18
298-102