**Agile Documentation**

**Documentation Development for Mission-Critical Department of Defense Systems using Scrum Agile Software Development Methodologies**

**George Mason University**

**Systems Engineering and Operations Research Department**

**May 2013**

**Qingyun Gu**

**Seung Jae Lee**

**Stephen Smith**

Table of Contents

[1. Background 4](#_Toc355618828)

[1.1 Introduction: 4](#_Toc355618829)

[1.2 Methodology: 5](#_Toc355618830)

[2. Research 5](#_Toc355618831)

[2.1 What Defines a Mission-Critical System? 5](#_Toc355618832)

[2.2 Applying Agile in Mission-Critical and Department of Defense Environments 5](#_Toc355618833)

[2.3 Focus of Documentation in Mission-Critical Systems 6](#_Toc355618834)

[2.4 Why Does the Government Want to Adopt Agile Software Development? 8](#_Toc355618835)

[2.5 What is Agile Software Development? 9](#_Toc355618836)

[2.6 Misconceptions of the Agile Methodology 11](#_Toc355618837)

[2.7 Scrum Agile 12](#_Toc355618838)

[2.8 Origins of Scrum 13](#_Toc355618839)

[2.9 Ideology Behind Documentation in Agile 14](#_Toc355618840)

[2.10 Two Approaches to Agile Documentation 16](#_Toc355618841)

[2.10.1 Adapted Agile 16](#_Toc355618842)

[2.10.2 Aligned Agile 18](#_Toc355618843)

[3. Framework: 21](#_Toc355618844)

[3.1 Introduction 21](#_Toc355618845)

[3.2 A Framework for Developing Documentation for Mission-Critical DoD Systems 22](#_Toc355618846)

[3.2.1 The Scrum Document Development Framework 22](#_Toc355618847)

[4. Defense IT Systems Acquisition Process change 24](#_Toc355618848)

[5. Conclusion 25](#_Toc355618849)

[6. Acknowledgements 27](#_Toc355618850)

[7. References: 28](#_Toc355618851)

**List of Figures**

[Figure 2‑1: Technology or Initiative Transition to Program of Records (MITRE) [2] 7](#_Toc355618852)

[Figure 2‑2: Global Project Failures [3] 8](#_Toc355618853)

[Figure 2‑3: Defective and Wasteful Requirements in Traditional Development Process [4] 9](#_Toc355618854)

[Figure 2‑4: Definition of Agility 11](#_Toc355618855)

[Figure 2‑5: Defense System Acquisition Framework (DAG) [14] 15](#_Toc355618856)

[Figure 2‑6: Major Milestone Review and the Documents Associated With Them According to the Defense Acquisition Guide 20](#_Toc355618857)

[Figure 2‑7: Document Generation in Agile Development Process [15] 20](#_Toc355618858)

[Figure 3‑1: Scrum Agile Document Development Framework 23](#_Toc355618859)

[Figure 4‑1: MITRE’s suggested DoD IT Acquisition Process [2] 25](#_Toc355618860)

**List of Tables**

Table 2‑1: DoDI 5000 Systems Engineering Documents 7

Table 2‑2: Traditional Systems Engineering Document Generation vs. Agile Systems Engineering Document Generation (DoDI 5000 series Documents) 20

**Abstract**

Agile development methodologies are increasing in prevalence within the Department of Defense Acquisition environment. While comprehensive documentation is not an emphasis of agile processes, robust documentation is required for Department of Defense mission-critical software systems. Through examination of existing case studies, industry standards, publications, and academic dissertations, multiple adaptations of Agile methods were encountered for complying with documentation requirements. With modifications to the Agile process, the documentation requirements for mission-critical software systems can be achieved. Through research, a process framework was developed for ensuring documentation generated using Agile development methodologies would be suitable for Department of Defense mission-critical products.

# Background

## Introduction:

In the current economic environment, survival of companies becomes an art of balancing work load and cost. The previous statement cannot be truer today than for government contractors. In an effort to better align with their customers’ needs, many contractors have looked at different ways to cut down costs of completing projects. The main method being pursued is to switch the development method from a traditional waterfall model to the more lean Agile method.

In the spring semester of 2013, Boeing presented George Mason University with questions regarding some of the unknown challenges of switching to the new development model. In specific, Boeing has approached the team with the question of whether the Agile methodology can adequately address the rigorous Department of Defense (DoD) standards and needs for documentation on mission-critical software systems. In addition, Boeing would like to know if documentation processes are impacted by adopting the Agile methodology.

This project effort addresses the problem in the several ways. The first is research to address the definitions of terms present in the project. This includes the overview of what the Agile methodology is and defining what is meant by mission-critical systems. Next is a comparative explanation of documentation philosophies and focus for mission-critical systems using a waterfall model versus that on an Agile project. The third is an explanation of two common approaches for generating documentation when using the Agile methodology with supporting research and case studies for each from existing literature and from company interviews respectively. Lastly, a documentation process framework using Scrum Agile developed by the team is presented.

## Methodology:

The team took a research approach to dive into the given problem. There are multiple resources to look at in terms of Agile development and the DoD standards for projects. However, there are few that took an integrated look at how the Agile process impacts documentation specifically. The team has proceeded with research in the following topics. The first topic is a look at what “mission-critical systems” means and how to define a mission-critical system under industry standards. The second area researched looks at academic dissertations on how to approach projects using Agile. This portion also looks at how using an Agile methodology impacts the type and quality of documentation generated. Specifically, two documentation approaches that were the most widely encountered in research will be discussed in a later section. Lastly, the team interviewed various companies for input on any existing mission-critical projects they are able to share regarding that used Agile methods.

# Research

## What Defines a Mission-Critical System?

The problem statement is explicitly concerned with documentation for mission-critical DoD systems. The project team has given a specific criterion for defining what a mission-critical system means. In this project’s context, the definition of mission critical system has been narrowed down to be defined as systems critical to DoD’s ability to meet its responsibilities and include command and control systems, satellite systems, inventory management systems, transportation management systems, medical systems and equipment, and pay and personnel systems [1]. Other examples of types of mission-critical systems include: navigation systems and avionics display systems.

## Applying Agile in Mission-Critical and Department of Defense Environments

Traditional development methodologies for software, which are rational, highly documented, plan-driven processes such as the V-Curve, use highly regulated, controlled and predictable frameworks to guide projects towards completion. Risk is a driving factor, along with the desire to remove uncertainty for the selection of such rigorous methodologies. The V-Curve defines a desired end state, and develops detailed plans early in the project to achieve the end state. Such rigidity prevents the projects from quickly responding to technical and functional requirements which may emerge during the development process [2].

The rigor present in V-Curve engineering covers all aspects of the project including documentation development. Traditional software development methodologies are driven by up-front planning, formal documentation, and teams sorted by function to develop the product in a linear method. Following such a traditional method, the product is handed from one functional team to the next at the end of each phase of development. While a traditional process may be suitable, and sometime preferred for product improvement or enhancement, traditional processes can hinder creativity [8]. The hindrance in creativity can lead teams away from traditional development and towards Agile development methodologies such as Scrum.

Various advantages and disadvantages exist when looking to develop software via Scrum, as opposed to a traditional method like the V-curve. While the creativity factor may be significantly higher with Scrum, certain trades are made to achieve the higher creativity. Using an unmodified version of Scrum could lead to a lack of proper emphasis on documentation for the system. For a DoD mission-critical system, having poor documentation is rather undesirable, and in fact would most likely be unacceptable to the stakeholders.

## Focus of Documentation in Mission-Critical Systems

Due to the strict safety regulations for mission-critical systems, the documentation needs are more heavily focused on following, verifying, and qualifying the system to industry safety standards. The heavy focus on ensuring safety lends mission-critical systems to the use of traditional software development methods, such as the waterfall model, to accommodate for the need to verify at each step whether the system satisfies safety requirements. Aside from a concentration on the safety aspect of the system, a DoD mission-critical system, or any Defense Acquisition Projects developed with a traditional development process (waterfall model), would still require and consist of the same type of documentation. The project team has narrowed down a list of documents descended from DAG (Defense Acquisition Guide), to a list of documents still applicable on DoD projects today. These documents are highlighted after Figure 2‑1, listed by review schedule. The research effort scope is only dealing with the DoD 5000 Milestone Documents. Documents associated with DoD 5000 Defense Acquisition Guide can be broken down according to review schedule. A few of the DoDI 5000 documents are listed in Table 2‑1.



Figure 2‑1: Technology or Initiative Transition to Program of Records (MITRE) [2]

Table 2‑1: DoDI 5000 Systems Engineering Documents

|  |  |
| --- | --- |
| Milestone Review (Per DoD 5000) | Documentation of Relevance |
| SRR | * SSS – System/Subsystem Specification * SRS – Software Requirements Specification * IRS – Interface Requirements Specification |
| PDR | * SSDD–System/Subsystem Design Description * SDD – Software Design Description * IDD – Interface Design Description |
| CDR | * TEMP- Test and Evaluation Management Plan * SEP- System Engineering Plan |

## Why Does the Government Want to Adopt Agile Software Development?

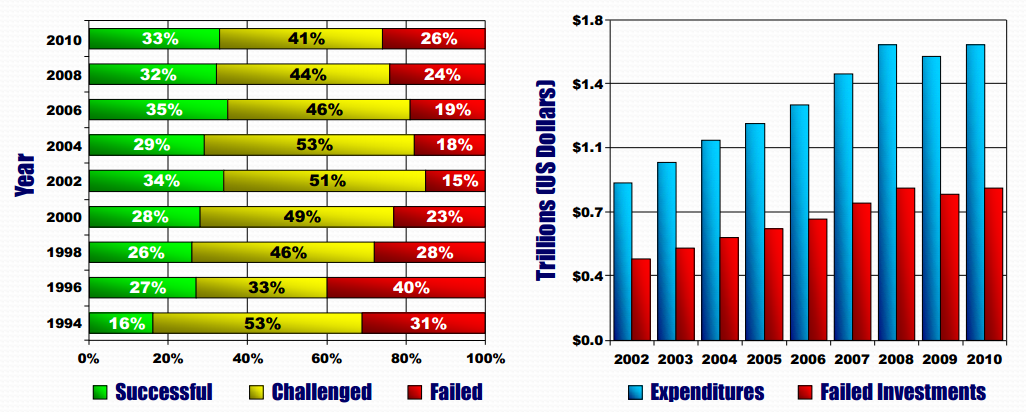


Figure 2‑2: Global Project Failures [3]

Globally, 67% of the IT Projects have been challenged or failed [1]. In 2010 alone, of $1.7T spent on IT projects, $858B was lost. Amongst the biggest reasons for the high failure rate was initial Requirement defects. Traditional project methodologies (i.e. waterfall development process) specify too many requirements [1] where 65% of the functionality from early defined requirements are never used at all (see Figure 2‑3 below) [1]. When following a traditional development process, requirement identification is done early on in the system development stage.

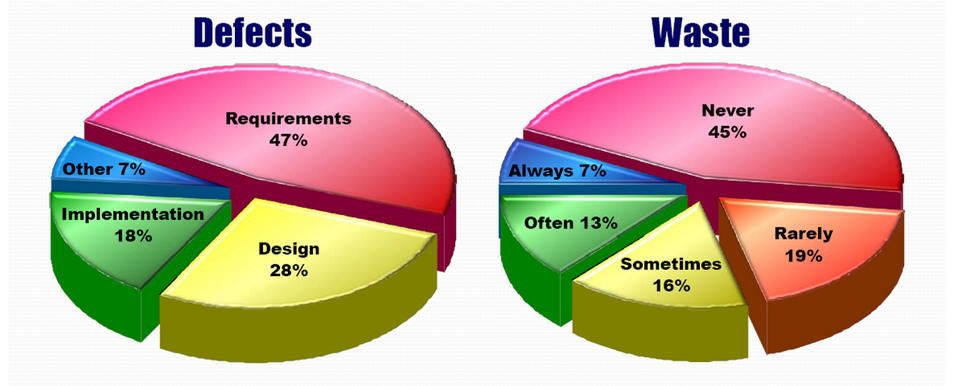


Figure 2‑3: Defective and Wasteful Requirements in Traditional Development Process [4]

## What is Agile Software Development?

The Agile software development method is not new by any means, yet its principles and methods of execution are very different from the way software projects have historically been run for the DoD. The Agile Manifesto is a set of principles for software development derived from a set of best practices that were already in use in the software industry [1]. The Agile Manifesto was developed by a group of industry experts in 2001 to formally consolidate and agree to a set of principles for lean software development. The Agile Manifesto revolves around four main principles [5]:

* + Individual and interactions over processes and tools
  + Working software over comprehensive documentation
  + Customer collaboration over contract negotiation
  + Respond to change over following a plan

The principles of Agile development aim to break down the software development cycle into multiple smaller cycles with each cycle ending in a complete working product [6]. The Agile method allows for changes in requirements and direction of the project and argues that lining up to the customer’s most current needs in each cycle actually helps save cost while gaining better customer satisfaction [7].

There are multiple styles of Agile methodology in use in industry today, but the team is limiting the discussion to Scrum Agile. Scrum Agile is an incremental development method where the total project completion is planned in “sprints” or small increments where a piece of the overall software is completed [6]. In the Scrum Agile context, principle number two would affect how much documentation is generated in each of the sprints, and principle number four would affect the needs to update existing documentation for up to date project state.

Agile is one of many software development methodologies which encourage development in a systematic and disciplined manner[[1]](#footnote-1). Within the classification of Agile software development are such methodologies as Extreme Programming, Lean software development, Crystal Clear, Scrum and more. Despite the many different instantiations of Agile software development methodologies, commonalities do exist between the methodologies. The previously mentioned Agile methodologies are a group of software developments processes which are iterative, incremental, self-organizing, and emergent [8]. The meaning of each of term, in the context of Agile software development methodologies, is explained below:

**Iterative** – having multiple iterations. In the case of Agile methodologies, being iterative is not simply being repetitive, but working to solve a software problem by finding multiple approximations to the solution beginning from the initial minimal set of requirements.

**Incremental** – Each element of an Agile system is developed to allow more requirements to be gathered and used during development, based upon previous requirements. As the development continues, the usable functionalities increase until a full operable system is created.

**Self-Organizing** – In an Agile environment, the team itself is permitted to organize autonomously according to skills and corresponding tasks. Doing so allows the team to determine how best the team can complete work tasks.

**Emergent** – Three items are implied by the concept of emergent in Agile methodologies.

1. Based upon the incremental development approach, the system emerges from a series of increments
2. Based on self-organization, an effective method of completing work emerges.
3. As the system emerges, a framework also emerges, supporting the method of working implemented.

Figure 2‑4 provides a visual depiction of the commonalities between all Agile software development methodologies.

Figure 2‑4: Definition of Agility [8]

## Misconceptions of the Agile Methodology

Agile has the misfortune of being subject to many misconceptions. Prior to delving deeper into Agile software development methodologies and the penultimate development of a document development framework, the misconceptions must be addressed. Multiple significant misconceptions are addressed below.

1. **Agile is a “Pizza Box” Methodology**

A common perception is that Agile is nothing more than an informal meeting between stakeholders and developers over a “pizza box” and the writing of code with no structure, process, tools, or defined methodology [9]. In actuality, Agile does require a planned and disciplined approach to the project, especially given the fast pacing of work on an Agile project. Being able to consistently develop software that is successful is the result of having skilled professionals engaged in a collaborative process of continuous planning and disciplined execution [10]. When properly implemented, Agile provides definitive structure, process, tools, and methodology through which software is developed.

1. **Agile is an All-or-Nothing Methodology**

Throughout industry, Agile has the tendency to be viewed in a binary manner, meaning Agile development must both be fully adopted and followed to the letter in practice and principal, or the development effort cannot be considered as Agile [9]. Such a sweeping generalization is not true. An Agile methodology such as Scrum can be modified to best suit the individual needs of any given project. Such capability for customization will be essential for the implementation of a documentation development process framework within Scrum Agile.

1. **Traditional Development is no Longer Feasible as Compared to Agile**

Another misconception is that traditional software development and the associated best practices associated with traditional development are either obsolete or irrelevant, and Agile fully replaces those concepts [9]. Good reasons still exist and permit argument to the contrary. Depending on the risk and complexity involved within a project, a traditional Waterfall or V-Model development process may even be preferred. Agile is flexible in nature, and is capable of supporting more traditional practices and tools to achieve higher levels of control and predictability over a project if needed.

1. **Becoming Agile Only Affects the Developers**

Many believe becoming Agile will only affect the developers [9]. Agile, by its very nature, necessitates greater stakeholder involvement. Additionally, becoming an Agile organization affects more than just the developers of the developing organization. Agile requires an organization-wide commitment from the business side of the organization to work in tandem with the development team.

## Scrum Agile

Scrum is a specific Agile approach for developing products and services [11]. In particular, Scrum is well suited to be applicable for the agile development of software. With an Agile approach, development begins with the creation of the product backlog. A product backlog is an ordered list of capabilities and features needed for the successful development of a product. The intent of the product backlog is to keep the development team focused upon the most important or highest priority tasks first.

Typically, work during a Scrum Agile development cycle is performed in timeboxed iterations. A Sprint is the Scrum terminology for a timeboxed iteration of development. The timeboxed iterations ranged in length from approximately one week, to a full month, depending on the particular project, instantiating organization, or any of a multitude of additional factors. During each of the iterations, a cross-functional, self-organizing team performs the various work tasks [11]. The cross-functional team needs to be able to perform such tasks as designing, building, and testing of the software. Such tasks are an essential part of producing completed, working, feature rich software, which is ready to enter production.

Due to the iterative nature of Scrum, multiple iterations are needed to perform the entirety of work outlined within the product backlog. Thus, at the start of each iteration, the team will plan which of the high-priority tasks will be the focus of the team for that particular iteration. The goal is to choose the subset of high-priority tasks the team will be capable of completing [11]. Once an iteration has been completed by the team, a review is held of the completed features. Stakeholders are required participants in the review. Based on the feedback garnered from stakeholders, the product backlog is updated. When updating the product backlog, the subset of high-priority tasks is free to change, as well as the priority of the various tasks. Again, the decisions of priority and the tasks for the next iteration are based upon stakeholder interaction.

Primarily, the intent of the methodology is for the team to have a potentially shippable product, or increment of the product, at the completion of each iteration [11]. If releasing after every iteration of development is not practical or effective, a team may release a set of features, compiled across multiple iterations of the development. The cycle repeats developing or grooming the backlog, choosing the tasks for the iteration, performing the development, reviewing with stakeholders and releasing a shippable product throughout the development cycle.

## Origins of Scrum

Scrum is a process that arose from the convergence of three innovative threads during the late twentieth century. The three threads were: Japanese advancements in Operations Research, growing incremental, iterative development research in the U.S., and the rise of Object Oriented technologies in software [12]. At the lowest sub-layer, Scrum relies upon the principles of the lean manufacturing movement and quality circles derived from Japanese manufacturing during the late 1980s.

During a similar time frame, the U.S. software industry was exploring project management using iterative and incremental methods [12]. One highly influential effort in the vein was IBM’s NASA Space Shuttle software project involving 17 distinct iterations over a 31 month time period, and the widely read paper “A Spiral Model of Software Development and Enhancement” by a researcher at TRW [12].

The third thread came about during the mid-1990s, as software managers came to the realization that object-oriented developers had yet to experience the breakthrough productivity seemingly promised by object-oriented coding technologies [12]. Operating under the belief that inheritance and polymorphism made object-oriented components more flexible than modules built with traditional procedural code, software managers began stripping the procedural aspects from current development techniques in an effort to make the development as nimble as the style of coding implemented with object-oriented coding techniques.

Eventually, it was the work of Dr. Jeff Sutherland and Ken Schwaber, who brought the three threads together. Sutherland and Schwaber combined the concepts of the threads together with their own research and experience on working in iterative methods for object-oriented development and further studies of process theory performed at DuPont’s Advanced Research Facility [12]. The first documented instance of Scrum development occurred at Easel Corporation in 1994. Easel Corporation was then a maker of a fourth generation language for mainframe data integration. Sutherland and Schwaber formalized their project management approach in a presentation given in 1995 before the Object Management Group’s “Object-Oriented Programming, Systems, Languages, and Applications” conference. Following the presentation, Sutherland and Schwaber began to offer Scrum-specific training and innovation conferences [12].

For the first few years of Scrum’s existence, many Scrum implementations were bottom up, but as Scrum achieved increasingly more successful projects, a top-down implementation was adapted. Today Scrum is practiced by global, name-brand companies including Microsoft, Yahoo, Ariba, Cadence, Adobe, GE Healthcare, Borland, Google, Primavera, Sun, Siemens, State Farm, Phillips, IBM, U.S. Federal Reserve Bank, HP, Motorola, SAP, Bose, CapitalOne and others [12].

## Ideology Behind Documentation in Agile

It is apparent from a quick glance at the four principles of Agile that documentation in the traditional sense would be impacted. Principle number two specifies that rather than spending a project’s time and budget on writing full documentation, the resources are preferred to be spent on churning out a useable piece of software at any point in time during development. Principle number four specifies that the project should respond to changes instead of following a strict plan for the project from the beginning to the end. Here, changes are defined as any new customer direction that arises during the progress of the project. As the project direction evolves over time, so would the documentation for the project to reflect the true state of the software product [13].

It is obvious then that the ideals for documentation for Agile development are almost in contrast with that of the needs for documentation on mission-critical systems. Typically, DoD contracts require very strict tracking of the progress of the project at predetermined intervals. The DoD also requires a specific set of documents to be delivered as part of project completion. These documents’ (DoD 5000 series documents) main purpose is to serve as a means of communicating the usage to the end user as well as a means of transferring knowledge of the system design to any future teams working on the software, whether making fixes or adding new functionality [13]. The research suggests that it is important to define what it means for any documentation to be “completed” for each part of the project using Agile [13].

The DoD’s needs for documentation was originally aligned with using the waterfall model. The waterfall model would require documents and reviews to be done at specific phases of the project and finally delivered in its most complete and up to date state at the end as deliverables to the customer. Figure 2‑5 illustrates the Defense System Acquisition Framework. Each triangle in the diagram represents a milestone review, and associated with each of those reviews are DoD 5000 defined documents.

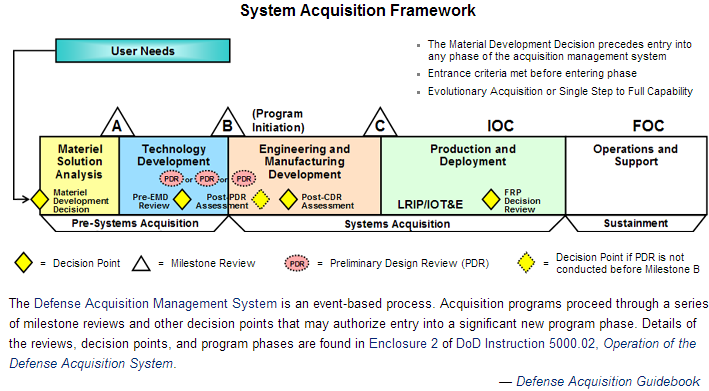


Figure 2‑5: Defense System Acquisition Framework (DAG) [14]

Agile documentation principles do not naturally accommodate for phased delivery of documentation nor are the principles of Agile accustomed for making very thorough documentation. However, there are multiple ways for a company who is constrained by documentation requirements to satisfy the documentation needs while still using Agile methods for development. The malleability of an Agile project’s direction is of major concern when applied to mission-critical systems. The rigid safety needs may not allow certain features to be added or removed without review and approval against safety standards. The Scrum Agile method cannot guarantee that changes made will not impact safety standards without some oversight on the entire project at every sprint. Although there are caveats with using Agile on a project in terms of the documentation it generates, there are methods that can be used to adapt the native Agile method to accommodate for more strict documentation needs. Two methods found through research are discussed in the following sections.

## Two Approaches to Agile Documentation

Despite the gap in the ideologies, it is possible to achieve necessary levels of documentation using the Agile approach of development. A look at existing literature on the topic revealed two main methods for accomplishing this. The first is to adapt Agile with a constraining framework to accommodate for documentation needs and safety requirements and the second is to factor in and negotiate a more Agile-oriented style of documentation with the customer and follow a very strict plan of communicating with and coming to agreement with the customer. These two methods will be discussed in more detail.

### Adapted Agile

A few pieces of research suggest that the use of pure Agile principles would not allow for the development of proper documentation, especially for mission-critical systems. The differences actually come from a mismatch between the program management side and the development side of the program. One of the biggest assumptions behind this idea is that the contract for the project is still using a standard DoD model for project management [13]. In this scenario, the DoD is expecting the company to meet specific in-project due dates like the critical design review meeting, where a portion of design documentation are reviewed by the customer. Other similar in-progress review dates follow the same paradigm and complete documentation is due in phases. A pure Agile approach may generate only a small portion of the expected documentation or not at all. The principles followed on Agile would only allow generating documents when they are needed and for a specific use. Thus, using pure Agile would not help accomplish the goal of documentation per the DoD’s expectations.

Research suggests that in order to meet documentation standards of Defense Acquisition Guidelines, while still using Agile for software development, the project needs to be framed to use the documentation needs and safety requirements as binding constraints within which the Agile project is operating [13]. Using the SCRUM Agile example, this would mean that the iterations must be planned such that they include documentation work to be align with customer’s key review dates [13]. The last iteration in a SCRUM Agile framework would need to accommodate for a much bigger documentation effort to get all the documents for delivery completed and up to the DoD’s standards [13]. It may even be suggested that the project plan one iteration just for documentation as the last iteration. The objective of this approach is to produce complete documentation at DoD specified time intervals that are typical of a waterfall approach to program management. The adapted Agile approach is suitable for when the DoD program management has not switched to using the Agile ideals and is expecting the traditional structures in place to support documentation. This approach would also suit the phased approach in the waterfall model that mission-critical systems are more adapted for by requiring the strict safety standards to still be in place and tested thoroughly.

Through interviews with companies in the industry, the team has identified two existing projects that have used the adapted Agile method for their documentation needs. They are discussed in more detail below. Please note that due to the nature of some of these projects, more detail cannot be obtained from the companies.

**Case 1: Raytheon Program of Record**

*Method:* Scrum Agile

*Documentation method:* Adapted Agile

*Size of Project:* 6 months

The project at Raytheon was trying to satisfy documentation needs by using the first method discussed. The documentation needs for each iteration was planned at the beginning of the iteration and is aligned with the customer’s directions for the program for the iteration. The last iteration has a heavy emphasis on generating deliverable documentation. In-progress reviews are followed as normal as part of the constraining factors for the Agile project. Any supporting documentation for those reviews is planned into the logical iteration where it should be generated to meet the customer’s dates.

**Case 2: MITRE ACAT-IA Program of Record**

*Method:* Hybrid Scrum Agile

*Documentation method:* Adapted Agile

*Size of project:* 2 years

ACAT-IA: a cost in excess of $360 million in FY 1996 constant dollars

MITRE’s project formed a Requirements Governance Council to oversee and manage any requirement changes per direction from the customer. The project was actually split in a way that’s closer to a spiral development effort than pure Scrum Agile. Every spiral consisted of 6 months of work broken down into a requirement analysis phase, a critical design phase, a development phase, a contractor testing phase, and finally a government testing and information assurance phase. Partially completed products are sent back to the customer for review every three months to ensure that the effort is going as the customers are expecting. Complete documentation for new capabilities is developed per spiral.

### Aligned Agile

A second approach to generating documentation on Agile projects is much more proactive in aligning the documentation needs with the principles of Agile development.

In this approach, the contractor must negotiate for an entirely different management style for in-progress reviews and along with that the scope and expectations of the associated documentation [7]. As stated before, the purpose of documentation is to be used as a tool for communicating information whether to the team or for future development efforts [13]. The contractor must define to the DoD what the objectives are for necessary documentation and how they would satisfy the objectives using a different approach to documentation [13]. In practice this could mean that the contractor generates just enough documentation necessary to communicate between its team members while generating more for final documentation of the product to be delivered to the customer. With this approach, it may not be till near the end of the project till deliverable documentation is finally put together for the customer. This second approach is suitable when a company is trying to move to an Agile development framework company-wide. In this case, customer cooperation in management style is desired. Unfortunately, research suggests that the DoD may not be as receptive to a new program management style [7]. However, this approach is the suggested route to go supported by research from the Software Engineering Institute [7] because it better aligns the project entirely to the Agile methodology. There is inherently less risk for the project if the customer’s expectations for the deliverables are changed.

From a solicitation on existing projects using Agile, the team has found two other projects that are currently using the aligned Agile method for their documentation needs. Please keep in mind that details may be sparse due to the nature of the projects discussed.

**Case 3: ACAT-III MITRE Program of Record**

*Method:* Hybrid Scrum Agile

*Documentation method :* Aligned Agile

*Size of project:* 3 years

ACAT-III: a cost in excess of $360 million in FY 1996 constant dollars

The information given suggested that as the project design was changed so did the documentation. The documentation is done as needed during each sprint to update to the latest design but is kept minimal. Most often the documentation was not done till the completion of a piece if not the entire part of the project. The team noted that emphasis was placed on defining what is “complete” for documentation. The definition used by this project was completion is not achieved until “the documentation can be given to a group that is not associated with the design, and that group can duplicate it completely and successfully.”

**Case 4: ACAT-IA MITRE Program of Record**

Method: Hybrid Scrum Agile

Documentation method: Aligned Agile

Size of project: 2 years

ACAT-IA: a cost in excess of $360 million in FY 1996 constant dollars

The MITRE project used 4-week long sprints and established a requirements committee to help develop and prioritize the stories that are used for development guidance on each sprint. The team held multiple sessions with the customer for clarifying requirements and use cases to the operational users. Every two to four sprints, the project progress is reviewed with the customer for comments and feedback. Documentation is limited to stories and resulting technical documents. Requirements and other technical documents are changed as directions changed from customer. A high-level comparison of Systems Engineering Documentation generation process for Defense Systems Acquisition Projects between Traditional Systems Development to Agile Systems Development can be found in Table 2‑2.

Table 2‑2: Traditional Systems Engineering Document Generation vs. Agile Systems Engineering Document Generation (DoDI 5000 series Documents)

|  |  |
| --- | --- |
| **Traditional Development and**  **Adapted Agile Development Documentation** | Figure 2‑6: Major Milestone Review and the Documents Associated With Them According to the Defense Acquisition Guide |
| -Systems Engineering Documents for DAG are created in chronological order  -Requirement Generation to Test and Evaluation Plan  -Traditional Development Process  -Comprehensive detailed systems engineering Documents |
| **Aligned Agile Documentation Process** | Figure 2‑7: Document Generation in Agile Development Process [15] |
| -Systems Engineering Documents are generated during each sprint.  -Requirements are drawn from capabilities documents and user stories  -Requirements after sprint  -Continual integration and testing |

The two methods found in research both have pros and cons of their own. The Adapted Agile method takes away the flexibility of the project management that Agile was supposed to allow. The second method is more aligned with Agile ideals but the lack of rigorousness of the process in terms of documentation may be unacceptable by the DoD. Based on the two methods found and what is used in industry, the team believes that a more hybrid solution may be the better approach to generating documentation while still using Scrum Agile for software development. The team has generated a process framework to detail the hybrid approach.

# Framework:

## Introduction

Software development is not a one-size-fits-all field. Every project is subject to unique constraints, be it the constraints for the system itself, the way the system is developed, or even how the documentation is generated in support of the system. Despite the uniqueness of each system and development effort, certain frameworks can be developed to guide the development efforts. The frameworks need not be limited to strictly high level software development guidelines. A framework, by design, may be created to guide more detailed aspects of the software development process. An example would be using a framework to guide the development of documentation in support of a Department of Defense mission-critical software development initiative.

Following the discussion of software development frameworks, a documentation development process framework will be created for software development projects using the Scrum Agile methodology. The framework is intended to support the development of documentation for DoD mission-critical systems, as such DoD systems that are subject to stringent documentation requirements not necessarily addressed by a simple implementation of the Scrum Agile development methodology.

Software development process frameworks are used to guide the software development process. A software development process framework is an instance of a particular software development life-cycle. The framework by which a software product is developed is defined through the software development life cycle model chosen for a particular project [16].

While there are many advantages for using a life cycle model for software development, the primary advantage is the encouragement of software development in a systematic and disciplined manner [17]. Software being developed for Department of Defense mission-critical systems is complex and cannot be created by one individual in the time allotted. As a result, software is commonly prepared by a team of individuals. When developing software with a team, a precise understanding is essential, and must exist among the team members as to when to perform the various tasks of the development [17]. Without a unifying framework, a development team can quickly become disjointed and ineffective.

To prevent having unacceptable documentation developed for a DoD mission-critical system, a simple process framework has been developed. The framework is designed to be implemented within a Scrum Agile software development lifecycle. As such the framework is designed to intertwine with Scrum and become a distinct yet seamless part of such core Scrum concepts as the product backlog, sprints, sprint planning, and end of cycle or iteration.

## A Framework for Developing Documentation for Mission-Critical DoD Systems

### The Scrum Agile Document Development Framework

Central to the framework is one core concept, the Definition of Done. The definition of done is orthogonal to user acceptance (or functional acceptance) criteria for a feature. Definition of done is a comprehensive checklist of necessary, value-added activities asserting the quality of a feature. Additionally the definition of done can be described as [18]:

* A checklist of valuable activities required to produce software
* The primary reporting mechanism for team member
* A non-static auditable checklist informed by reality

Simply stated, for DoD mission-critical software development initiatives, the definition of done must comprehensively address required documentation. Involving documentation within the definition of done not only definitively places documentation within the product backlog, it also is valuable in reporting the documentation as completed. Such a function is in some ways not too dissimilar from a milestone when working towards definition of done.

The Scrum Agile Document Development Framework is highlighted in Figure 3‑1 below.

Figure 3‑1: Scrum Agile Document Development Framework

The primary goal of the Scrum Agile Document Development Framework is to fit document generation within the Scrum Agile software development life cycle. Agile methodologies should be just that, agile. While the framework provides needed structure to develop documentation for DoD mission-critical system being developed with Scrum Agile, the framework should not become cumbersome or be a hindrance in any way to the development cycle. The six phases of the framework are addressed below.

#### **Examine Required Documents:**

Before beginning, both the developing organization and the stakeholders need to see eye to eye on the documentation requirements. The Examine Required Documents phase begins with a detailed review of the document needs of the stakeholders. Each required document and the associated content for the document are to be identified.

#### **Finalize Contract Agreement**

In keeping with the Agile tradition of stakeholder involvement, the Finalize Contract Agreement phase would consist of a meeting between the developing organization and the stakeholders. The purpose of the meeting is to ensure all parties are on the same page, in terms of the document requirements. Once an agreement is reached, the agreement is verified as part of the current contract for the project. Should changes be required, the Finalize Contract Agreement phase is where the modifications are made. The finalized agreement is used in the next phase.

#### **Establish Definition of Done**

The Definition of Done must be established for the documentation. The definition of done will not only be used when prioritizing tasks for the product backlog, the definition of done is also used to validate whether the major tasks are account for [18]. The definition is used as a checklist to verify whether all necessary documentation requirements were met.

#### **Develop Product Backlog**

Development of the product backlog features not only the capabilities of the software visible to the customer, but also the technical requirements of the project. The product backlog is to contain the relevant tasks associated with completing the documentation requirements the developing organization is contractually obligated to deliver [19].

#### **Sprint**

During the Sprint phase, the documentation is developed alongside the software as appropriate. When planning for the Sprint, the development team identifies and plans how documentation will be approached within the Sprint. A relatively unstructured part of the framework, the Sprint phase is dependent on the development team to properly handle the development of the documentation.

#### **End of Cycle Review**

Once the Sprint phase is completed, the last step is to evaluate the work completed for documentation. The evaluation is made against the definition of done. Stakeholders are also consulted and the development team must achieve buy-in from the stakeholders on the documentation, or progress of the documentation, should additional iterations be required. Doing an evaluation of the documentation allows the development team to analyze the progress made and determine if more effort needs to be expended towards meeting the contractual document agreements.

# Defense IT Systems Acquisition Process change

IT systems acquisitions in the DoD are required to follow the DoD Instructions 5000 detailed by the JCIDS (Joint Capability Integration Development System) [2]. As discussed in Section 2.3, the current DoD Systems Acquisition Process places a heavy emphasis on extensive Systems Engineering Documentations, as part of each milestone reviews. These extensive documentation requirements are very appropriate for large weapons systems development. Such programs have been recognized for their propensity for excessive cost and schedule overruns as software projects (see Figure 2‑2). As a result, the DoD has initiated a change in the current DoD Instruction 5000 to better serve the Agile Software Development Process. (National Defense Authorization Act for fiscal year 2010, Public Law 111-84, in section 804). The changes to the DoD acquisition process for IT systems, specifically states to include the following to the acquisition process, which can be recognized for their similarity to Agile best practices [2]:

* Early and continual involvement of the user
* Multiple, rapidly executed increments or releases of capability
* Early, successive prototyping to support an evolutionary approach
* A modular, open-systems approach

A sample DoD IT systems acquisition guidelines by MITRE can be seen in Figure 4‑1[2]. These changes to the DoD IT systems acquisition guidelines will accommodate the Agile Development Process. Through the paper it was emphasized that the traditional systems engineering documents (SRS, TEMP, SEMP, etc.), which are currently requirements of Major Milestone Review (SRR, PDR, CDR). Yet, these documents can be indeed, and have been, retroactively generated when required.



Figure 4‑1: MITRE’s suggested DoD IT Acquisition Process [2]

# Conclusion

Given the current fiscal environment surrounding Department of Defense system acquisitions, Agile development methodologies are increasing in popularity. The associated costs of traditional systems development and the significant failure rates of existing development efforts also drive the need for alternative development options. With proper planning and modification to Defense Acquisition Guidelines, Agile development methodologies are suitable for development of mission-critical software systems in the DoD environment. Currently, the Defense System Acquisition Framework is tailored for systems developed using traditional systems engineering methods such as the Waterfall method and V-Curve. Based upon the success of agile projects, the DoD is currently overhauling DoD Instruction 5000 to better support the use of Agile development methodologies.

Scrum Agile is found to be a commonly implemented Agile solution for managing software development. However, Scrum Agile alone may not meet the stringent documentation requirements for DoD mission-critical software systems development efforts. Largely due to the fundamental ideas behind Agile development methodologies, documentation development on Agile projects receives less emphasis. Documentation is an essential part of systems engineering efforts and plays a particularly important role for mission-critical software systems in the DoD environment. The adapted Agile and the aligned Agile methods present two unique approaches which may be used in combination to inform a framework for document development.

The Scrum Agile Document Development Framework, as developed, is one method for addressing the modifications needed to promote a more robust way of developing documentation for Scrum Agile projects. Designed to seamlessly integrate with Scrum Agile, the core concept of the SADD is to implement the document requirements as a part of the Definition of Done. With the document requirements defined within the Definition of Done, documentation becomes an integral part of the development efforts.

# Acknowledgements

The team would like to give recognition for all the help and cooperation received from various stakeholders and associated staff and industry experts.

A special thanks to Dr. Andrew Loerch and Dr. Philip Barry at George Mason University, Colette DeLabarre at Boeing, Patrick Benito at MITRE, Casey Brenner at Raytheon, and Samuel Opoku at ITT Exelis.

# References:

1. Larson, E., Peters, J. Preparing the U.S. Army for Homeland Security: Concepts, Issues, and Options, Issue 1251. RAND Corporation, 2001.
2. D. K. M. R. B. M. C. Carlton Northern, Handbook for Implementing Agile in Department of Defense Information Technology Acquisition, The MITRE Corporation, 2010.
3. Standish Group. (2010). Chaos summary 2010. Boston, MA: Author.
4. Session R. (2009). The IT complexity crisis: Danger and opportunity, Houston, TX: Object Watch.
5. Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., et al. (2001). *Manifesto for Agile Software Development.* <http://agilemanifesto.org/> Accessed April 24, 2013.
6. Scrum Is an Innovative Approach to Getting Work Done. The SCRUM Alliance (2010)  
   <http://www.scrumalliance.org/pages/what_is_scrum> Accessed April 27, 2013.
7. Burton, D., Hammons, C., Lapham, M., Schenker, A., Williams, R. (2010) “Considerations for Using Agile in DoD Acquisition.” <http://www.sei.cmu.edu/reports/10tn002.pdf> Accessed March 19, 2013.
8. I. G. Stamelos and P. Sfetsos, Agile Software Development Quality Assurance, Idea Group Inc (IGI), 2007. [also steve figure 1]
9. C. G. Cobb, Making Sense of Agile Project Management: Balancing Control and Agility, John Wiley & Sons, 2011.
10. P. Schuh, Integrating Agile Development in the Real World, Charles River Media, INC, 2005.
11. K. S. Rubin, Essential Scrum: A Practical Guide to the Most Popular Agile Process, Addison-Wesley, 2012.
12. Ralph Hughes and the Ceregenics Data Management Team, Agile Data Warehousing, Delivering World-Class Business Intelligence Systems using Scrum and XP, iUniverse, 2008.
13. Forward, A. (2002) Software Documentation – “Building and Maintaining Artifacts of Communication,” [www.site.uottawa.ca/~tcl/gradtheses/aforward/aforward\_thesis.doc](http://www.site.uottawa.ca/~tcl/gradtheses/aforward/aforward_thesis.doc)

Accessed March 30, 2013.

1. DAGS (Defense Acquisition Management System), [Enclosure 2](https://acc.dau.mil/dag_5000.02p14) of [DoD Instruction 5000.02, Operation of the Defense Acquisition System](https://acc.dau.mil/dag_5000.02p1), Defense Acquisition Guidebook
2. Dr Steven J. Hutchison, Achieving Enterprise Agility in DoD IT Acquisition, 2010
3. K. A. Saleh, Software Engineering, J. Ross Publishing, 2009.
4. R. Mall, Fundamentals of Software Engineering, PHI Learning Pvt. Ltd., 2009.
5. D. Panchal, "Scrum Alliance," 03 September 2008. [Online]. Available: http://www.scrumalliance.org/articles/105-what-is-definition-of-done-dod. [Accessed 27 April 2013].
6. Scrum Alliance, "Scrum Alliance, Transforming the World of Work," [Online]. Available: http://www.scrumalliance.org/pages/scrum\_artifacts. [Accessed 4 April 2013].

1. While many other software development methodologies do exist (Waterfall, Prototype model, Incremental, Iterative, V-Model, Spiral, Lean, etc.), the purpose of this paper is to define a process framework for documentation development when using Scrum Agile development. As such, the primary software development methodology examined will be Agile and its various instantiations. [↑](#footnote-ref-1)