USING HYBRID SCRUM TO MEET WATERFALL PROCESS DELIVERABLES

by

Emil Moster

May, 2013

Director of Thesis/Dissertation: Dr. Nasseh Tabrizi

Major Department: Department of Computer Science

System Development Life Cycles (SDLCs) for organizations are often based upon traditional software development models such as the waterfall model. These processes are complex, heavy in documentation deliverables, and are rigid and less flexible than other methods being used in modern software development.

Consider by contrast, agile methods for software development. In essence, agile methods recommend lightweight documentation and simplified process. The focus shifts to completed software as the “measure of success” for delivery of product in software projects, versus accurate and comprehensive documentation, and the accomplishment of static milestones in a work breakdown structure.

This thesis implements, explores, and recommends a hybrid agile approach to Scrum in order to satisfy the rigid, document-laden deliverables of a waterfall-based SDLC process. This hybrid Scrum is a balance of having enough documentation and process - but not too much - to meet SDLC deliverables, while at the same time focusing on timely product delivery and customer interactions that come from an agile approach to software development.

USING HYBRID SCRUM TO MEET WATERFALL PROCESS DELIVERABLES

A Thesis

Presented to the Faculty of the Department of Computer Science

East Carolina University

In Partial Fulfillment of the Requirements for the Degree

Master of Science in Software Engineering

by

Emil Moster

May, 2013

USING HYBRID SCRUM TO MEET WATERFALL PROCESS DELIVERABLES

by

Emil Moster

APPROVED BY:

DIRECTOR OF THESIS: _____

M.H. Nassehzadeh Tabrizi, PhD

COMMITTEE MEMBER:_____

Junhua Ding, PhD

COMMITTEE MEMBER:_____

Sergiy Vilkomir, PhD

COMMITTEE MEMBER:_____

Karl Abrahamson, PhD

CHAIR OF THE DEPARTMENT

OF COMPUTER SCIENCE: _____

Karl Abrahamson, PhD

DEAN OF THE GRADUATE SCHOOL: _____

Paul J. Gemperline, PhD

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

The United States Coast Guard (USCG) has a system development process defined called the System Development Life Cycle (SDLC) [1]. The USCG SDLC is based upon a waterfall process model which is defined and maintained by CG-6, by authority of USCG Headquarters. CG-6 is the authoritative entity within the USCG for Enterprise Information Systems. All system development projects which CG-6 undertakes must follow the SDLC. This process is complex, heavy in documentation deliverables, and it is rigid and less flexible than other methods being used in modern software development.

Consider by contrast, agile methods for software development. In essence, agile methods recommend lightweight documentation and simplified process. The focus shifts to completed software as the “measure of success” for delivery of product in software projects, versus accurate and comprehensive documentation and the accomplishment of static milestones in a work breakdown structure (WBS).

The Aviation Logistics Center Information Systems Division (ALC ISD), is one of the three data centers for the USCG. It is investing in using agile methods - specifically Scrum - to produce and deliver systems with projects, and it is incorporating agile methods within its existing processes. Where the conflict between “becoming agile” and “adhering to rigid process” occurs is where ALC ISD serves as the System Development Agent (SDA) for the Coast Guard Logistics Information Management System (CG-LIMS) project, and the Project Management Office (PMO) of this same project must adhere to the SDLC. Scrum doesn’t fit neatly within the framework of the SDLC and it doesn’t satisfy the expected deliverables of this waterfall process.

These conflicts are evident in numerous ways throughout the project. Trying to align SDLC milestones and deliverables with Scrum deliverables, trying to report metrics from Scrum which satisfy SDLC requirements, and trying to align requirements defined with the IEEE-830 standard [2] with user stories created to represent the work completed in agile sprints are but a few of the challenges. The conflicts manifest themselves in numerous ways, ranging from confusion within the PMO about how success is measured, to frustration within the Development Team due to lack of familiarity with new methods.

This thesis implements, explores, and recommends a hybrid agile approach to Scrum in order to satisfy the rigid, document-laden deliverables of the waterfall-based SDLC process. This hybrid Scrum is a balance of having enough documentation and process - but not too much - to meet SDLC deliverables, while at the same time focusing on timely product delivery and customer interactions that come from an agile approach to software development. The two need not be at odds and conflict with each other; there can be a happy middle ground where quality, timely software products and customer satisfaction are delivered, while at the same time satisfying the SDLC process to ensure the proper safety, maintainability, and accountability of an enterprise system.

CHAPTER 2: RELATED WORK

The USCG is not alone in it's desire to become agile with software systems development. In 2009, 76% of organizations reported using agile methods to accomplish software development [3]. The software industry recognizes that there is value to be gleaned by setting aside traditional, sequential development models such as waterfall, and adopting one of the many agile process models such as Scrum, Extreme Programming (XP), Crystal, and Agile Unified Process (AUP) to name a few. Waterfall, with it's heavy documentations requirements, rigidly structured sequential approach, and phase exit reviews (PER) which gate the phase exits and entries has been the cause of many a failed system development project. The Department of Homeland Security (DHS) - the parent organization of the USCG - realized this and formed the DHS Agile Working Group, who worked together to draft a whitepaper on using agile methods in DHS [4]. Interestingly, this effort by DHS was happening simultaneously as the USCG was beginning the CG-LIMS project. While DHS was formulating a proposal for a solution to replace their waterfall Systems Engineering Life Cycle (SELC) with an agile process based on Scrum, the USCG was also embarking on, empirically proving how two processes - waterfall-based SDLC and agile Scrum (see Figures 1 and 2) - could coexist.

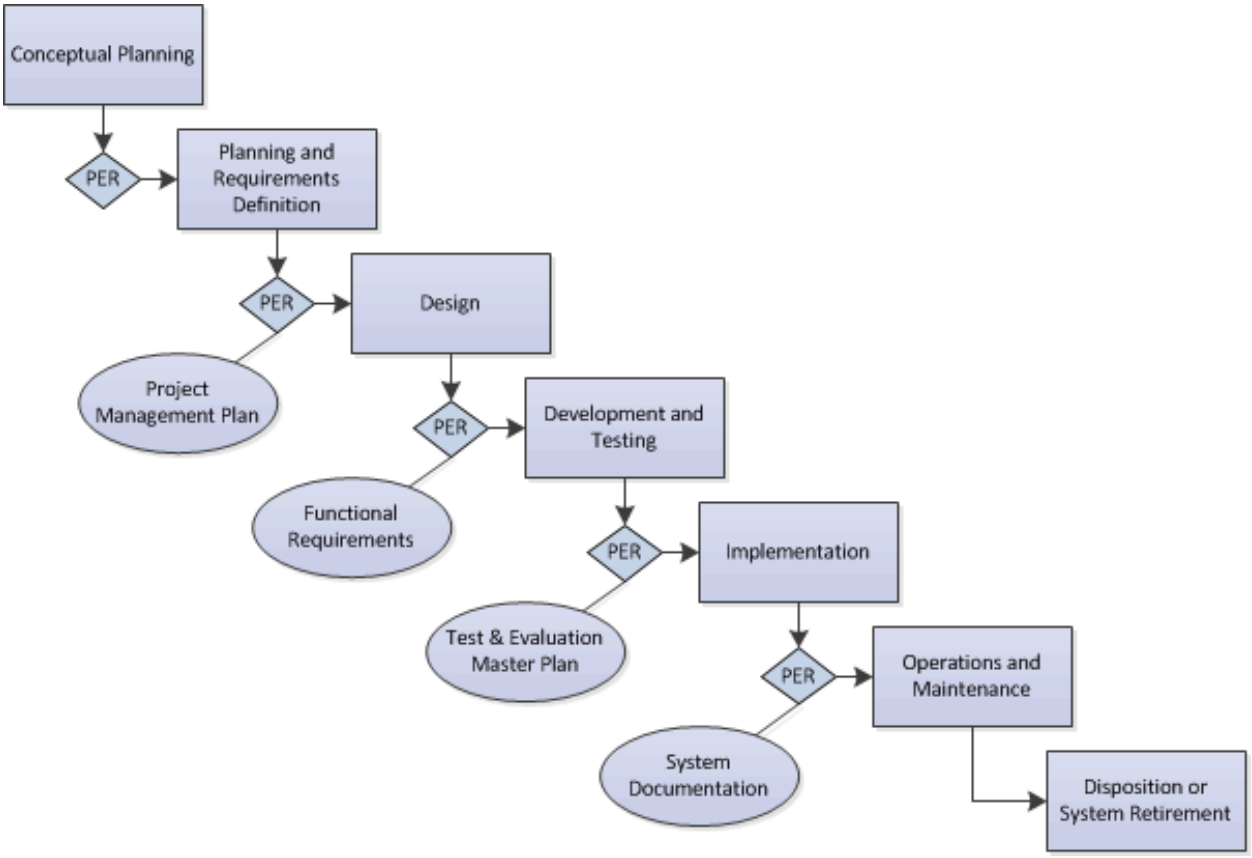


Figure 1: The USCG SDLC

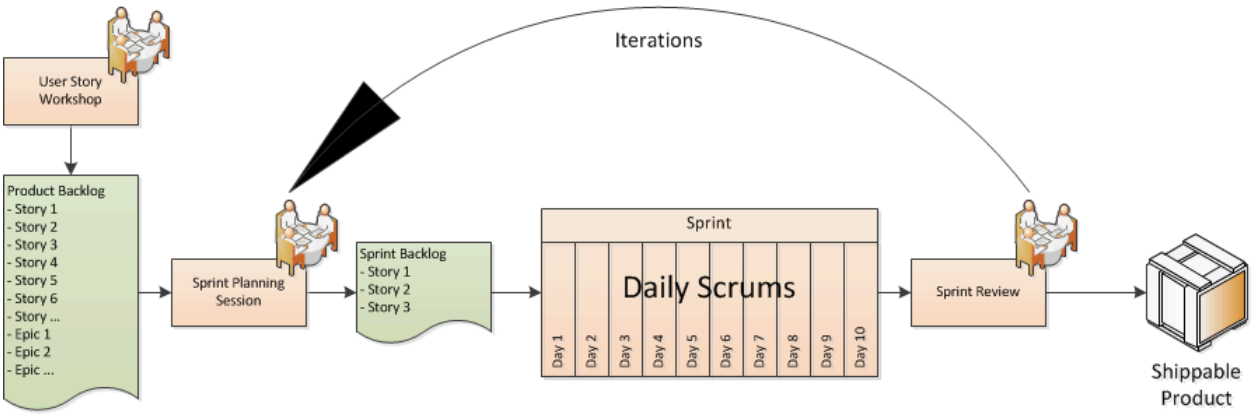


Figure 2: Scrum Process Model

Since the USCG's ALC ISD had already recognized the need to implement agile methodologies in working projects to modernize its legacy systems, their teams had

more freedom to implement Scrum in a pure manner; unconstrained by having to adhere to the rigid SDLC waterfall process. ALC ISDs support process (effectively their own SDLC) is not based on a waterfall process (see Figure 3). It is an incremental approach to software development, delivering incremental portions of software in three month phases. While ISD was not using the waterfall process for its projects, it found

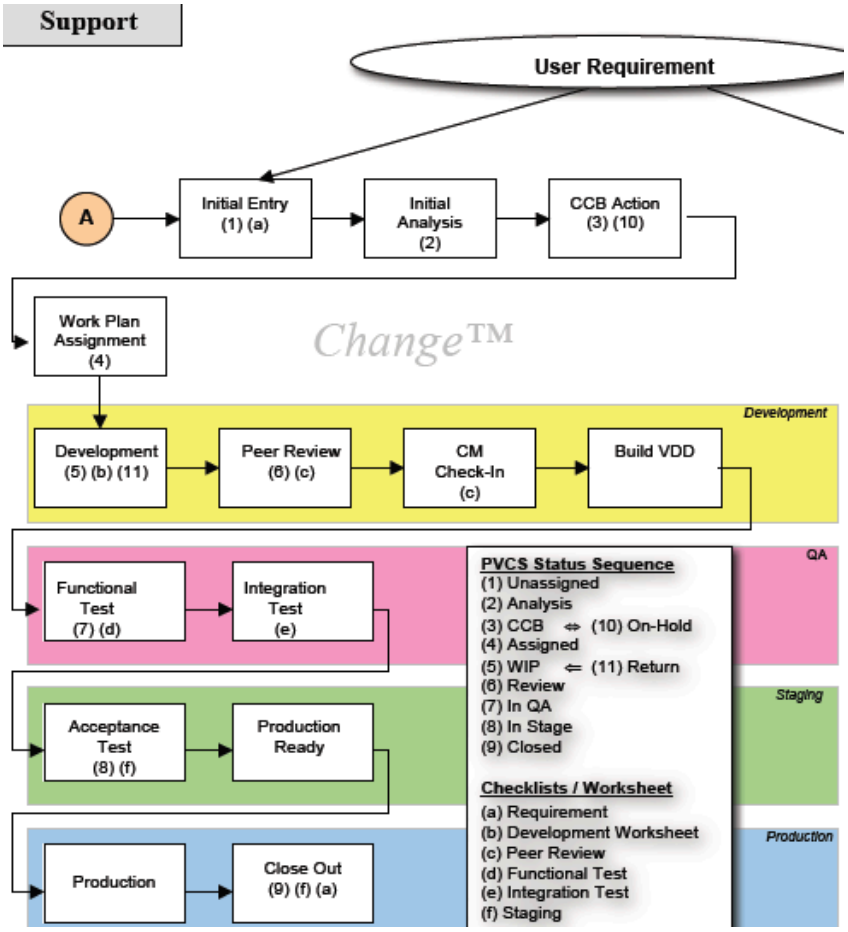


Figure 3: ISD Production Promotion Process

that building software increments in three month phases was still too long of a cycle without delivering potentially shippable product to the customer. Customer reviews and feedback were infrequent, and led to “mini-waterfall” pitfalls such as scope creep, engineering delights, large amounts of rework due to developer/customer misunderstandings, and delays in software deliveries.

In the same respect that DHS could not completely drop its SELC and adopt agile methodologies overnight [5], the USCG could not drop its SDLC and adopt Scrum overnight. The USCG ALC ISD could, however, adopt agile Scrum into the Development phase of its support process (see Figure 3) essentially overnight, as long as the remainder of the process remained intact as required for its current ISO 9001 certification.

As the DHS Agile Working Group continued to formulate its position in its draft white paper, the working group benefited from one of its contributors being the Project Manager for the CG-LIMS project. And so, while the working group was not embarking on it's own empirical study, they were getting empirical input from one of it's contributors who was "living the dream" day-in and day-out with the CG-LIMS project. DHS also was not trying to come up with a solution to make waterfall SDLC and Scrum coexist; rather they were preparing a solution to replace the waterfall SDLC with Scrum. This is the primary difference between the effort at DHS and the CG-LIMS project.

It was a given by June 2011 that ALC ISD would be developing the solution for the CG-LIMS project, and it was expected that it would be using agile Scrum to accomplish the project. It was not specifically determined, however, how Scrum would be used to satisfy waterfall SDLC requirements. Lessons were taken from Cohn's [6] experience with making two process models coexist. He notes that most organizations which have a sequential process implemented and choose to migrate to agile methodologies will not be able to do so overnight, so they must coexist together if for even a short while. Sliger [7] suggests three different ways that coexisting processes interact: waterfall-up-front, waterfall-at-end, and waterfall-in-tandem. The CG-LIMS

project team chose to follow the waterfall-in-tandem scenario as it best satisfied the need for continual benefits from agile methods, while continually keeping the SDLC deliverables satisfied. It was not an option to give a “handshake” to the waterfall SDLC at the beginning of the project and then wave goodbye to it, nor to put it off until the end of the project. Cohn notes that this is the most difficult of the three approaches to take, as one team works from the perspective of the sequential approach (in the case of CG-LIMS, the PMO did this) and they prefer to communicate through meetings and documents. The other team (the CG-LIMS Scrum Team) chose to communicate informally but frequently to progressively define work and functionality - agility.

While this was not optimal, it did satisfy this particular projects needs. Consider what may have been an example of an optimal situation, by contrast, albeit a story of a dark cloud with a silver lining. The Federal Bureau of Investigations (FBI) had a long-delayed waterfall project - the Sentinel project [8] - which was plagued with many of the challenges of an IT project gone awry: missed deadlines, budget overruns, and shortfalls on promised features. The decision was made in September 2010 to turn the project to agile methods, and the FBI credits this decision to do so as the ultimate reason for completing the project. In the FBIs case, they did not have to make process models coexist, and they turned their failing project around with overwhelming success using agile methods only. Sentinel is now functioning bureau-wide as the FBIs digital case management system. Their project not only delivered faster, but it was also within budget for the project.

One of the major complaints about large, failed waterfall projects is that too much time is spent upfront on documentation which may not ever define an actual working

system. If and when it does, the documentation will be obsolete as the requirements most likely have changed. The Agile Manifesto [9] states that "[we] value working software over comprehensive documentation." McMichael and Lombardi [10], using agile methods while working on Primavera Systems quality management system noted there were significant concerns with violating this principle, but they provided just enough documentation to be a useful reference; to help with enforcing the existing process, but no more than that. This was the approach the CG-LIMS Scrum Team took as well; just enough documentation to satisfy the SDLC process needs. Again, not an optimal solution as Scrum would forego documentation to this extent, but one that would prevent the inevitable roadblock that the team would meet at the first PER requiring satisfactory documentation. While some might think that developers implementing agile methods would be completely averse to accomplishing documentation for the system, a field survey of software professionals actually showed the opposite [11]. Respondents in the survey actually noted that, had they had the time, and resources were assigned to tasks appropriately, they would have rather spent more time in planning and documentation versus coding and debugging. This gives credence to the fact that documentation is recognized as being good and necessary; but only the right documentation lends value to the project. If it does not bring specific value, then it is not necessary.

Members of the CG-LIMS Scrum Team and the PMO had previous experience with delivering a developed enterprise system and the customer interactions necessary to be successful with this. For this reason, the team knew that Scrum would provide that type of interaction again, which would be necessary for success. Saving customer

interaction for only a few touch points that waterfall SDLC would provide - System Requirements Generation, Preliminary Software Review, Critical Software Review, and Final Software Acceptance Review - [12] would be disastrous to the project. As the CG-LIMS project has proven and continues to prove, constant customer interaction throughout all the steps of the process does not inhibit or encumber the Scrum Team, but rather liberates them to develop a solution that quickly and ultimately satisfies the customer. Essentially, the customer cannot find issues with that which they've defined all along the way, and to which they've contributed in the decision-making. Mann and Maurer [13] showed in a 2 year study of the industry that Scrum provided an increase in customer satisfaction, primarily due to being involved throughout the process of developing functionality. The customers appreciated more involvement throughout, in comparison to their previous roles, which were only limited to acceptance testing.

Without a specific implementation of a hybrid Scrum model which exercised team agility, yet also met waterfall SDLC documentation requirements, the CG-LIMS Team deemed it necessary to take lessons learned from current industry studies, standards, and best practices regarding agile software development, and others' experience with navigating waterfall SDLC processes, and to create a hybrid Scrum model which satisfied both camps. While DHS had not actually implemented their agile framework which was being proposed, it was convenient for the CG-LIMS Team to leverage concepts that DHS had proposed in formulating and implementing its hybrid Scrum model. Some elements brought specific, realized value to the the hybrid Scrum model, while others were yet to be realized. So the hybrid Scrum model specifically benefited from the research done by DHS, and in turn the CG-LIMS Team empirically gained data

and vetted some of the concepts for DHS. What also came to light during this research and as a result of the actual implementation of the hybrid Scrum model is that not only does it facilitate the coexistence of two process models - Scrum and waterfall - but it also serves the purpose of method organizations can use for migration from waterfall to Scrum methodology. Cohn [6] notes that many organizations cannot completely drop the existing processes that they've had in place for years; they need an incremental transition which eases the burden and cost of taking their organization from waterfall to Scrum (or to other agile methods). And so hybrid Scrum is offered up as a solution to facilitate this transition, where it may be a period of many years that the organization must take in proving out the viability of Scrum in eventually replacing their waterfall process.

CHAPTER 3: CURRENT PROJECT BACKGROUND

3.1 ALMIS - The Existing Legacy System

The Asset Logistics Management Information System (ALMIS) is an organically developed conglomerate of various systems, both old and new, which have been loosely integrated over years to meet the specific needs of the USCG fleet and personnel. The fleet includes airplanes, helicopters, boats, cutters, unmanned aerial vehicles (UAVs), and Digital Global Positioning System (DGPS) towers; currently well over 1,000 complex assets are maintained in ALMIS. The existing user base is over 16,000 customers performing various tasks and duties related to logistics management within the system.

Asset configuration management and scheduled maintenance management are supported by the Asset Configuration Management System (ACMS) subsystem. ACMS is a two-part system: a “green screen” character-based application, and a thick-client graphical user interface (GUI) application. The system uses a combination of Ingres Applications By Forms (ABF), Ingres Open Rapid Object Application Development (OpenROAD), and the Ingres Relational Database Management System (RDBMS). Supply chain management, including inventory control, purchasing and requisition, financial management, and transportation are supported by the Asset Maintenance Management Information System (AMMIS) subsystem. AMMIS is primarily based upon Ingres ABF and Ingres RDBMS. Both of these subsystems are over 20 years old, and have reached the end of their sustainable life cycles. They are in need of replacement with a modern system.

Operations, mission tracking, training and qualifications, and unscheduled maintenance are supported by the Electronic Asset Logbook (EAL) subsystem. EAL is a web-based system which has been developed on a combination of Microsoft Active Server Page (ASP) pages and Personal Home Page (PHP) pages. Decision support needs including reporting, dimensional analysis, and business intelligence (BI) in all business areas are supported by the Decision Support System (DSS) subsystem. DSS is programmed and maintained using the IBM COGNOS BI Suite. Since both of these subsystems are based upon current technology, there is not an immediate need to replace either of them with a modern system.

Technical Information Management is currently accomplished with multiple subsystems for various needs. Arbortext is the system used to Author the content for technical information used in ALMIS. Asset Technical Information Management System (ATIMS) is currently the system which is used to accomplish content management of technical information. Technical Manual Application System (TMAPS) is a Department of Defense (DoD)-owned system which is used by the USCG to store and deliver technical documentation used in logistics management for USCG assets. The USCG pays an annual fee to DoD in order to use the TMAPS system.

Each of these subsystems, while related by business processes implemented around the subsystems, are not directly integrated with each other from a technical aspect. A single, main database is used for the ACMS, AMMIS, and EAL subsystems, but data structure overlap and reuse is minimal where business processes actually overlap. The DSS subsystem uses a separate, synchronized reporting database to offload report processing from the transactional database. ATIMS and TMAPS use

separate databases entirely. So from the perspective of system integration, efficiency, and reduction of waste, the legacy system does not employ the optimal model. It is quite disparate when compared with Commercial-Off-The-Shelf (COTS) logistics systems which are offered on the market today.

Giving consideration to this architecture and it's lack of efficiency, the cost and effort to maintain this aging system warrant the decision made by the USCG to acquire and implement a new logistics system - a COTS solution - which would replace the legacy system.

3.2: CG-LIMS - The New Replacement System

The Coast Guard Logistics Information Management System (CG-LIMS) project began in 2008 as a major acquisition for the USCG [14]. CG-LIMS is a technology refresh of ALMIS, and would be a Commercial-Off-The-Shelf (COTS)-based enterprise logistics system using the Oracle E-Business Suite, which will replace the existing logistics system. The new system will provide the same support as the existing system for asset lifecycle configuration management and maintenance management, supply chain management, technical information management, and decision support, without all the inefficiencies or shortcomings identified with the legacy system.

While CG-LIMS is a COTS-based system solution, it is still an enterprise logistics system, and requires significant configuration in order to implement it with the specific business of the U.S. Coast Guard. Additionally, there is significant software development which must be done in order to interface the system with legacy data, and to migrate/convert this data into usable data - with integrity - into CG-LIMS. So while the configuration and development effort on the part of the Development Team is

minimal compared to a completely organic solution which is built from the ground up, there is still significant software development and configuration which must be accomplished within the project, in a controlled and phased implementation.

In June 2011, the CG-LIMS project was presented to ALC ISD as a project in which ISD would be the System Support Agent (SSA) as well as the System Development Agent (SDA). This was done for several reasons: ISD is physically collocated in proximity to a nucleus of the logistics customers, ISD has first-hand experience in implementing new system solutions for logistics management in the USCG, and ISD has experience in agile development. As ISD accepted the project, it officially kicked off on January 17, 2012 with a new Scrum Team - the CG-LIMS Development Team - comprised of existing Developers, Analysts, and a ScrumMaster. Additionally, Subject Matter Expert (SME) Consultants were hired with specific expertise in the COTS product to be implemented. Their role would be to begin delivering the first products of the COTS implementation using agile methodologies to gain customer and stakeholder confidence early, and to continually deliver in a high but sustainable pace for the life of the project. Given these expectations, and the known encumbrance of the governance from the USCG SDLC, the viable solution was to define and implement a hybrid solution which leveraged agile methods, and satisfied waterfall deliverables as well.

CHAPTER 4: HYBRID SCRUM METHODOLOGY

Based on the desired results for the CG-LIMS project, and considering the given constraints of satisfying the USCGs existing waterfall SDLC, a hybrid Scrum methodology was implemented in order to leverage the full benefits of an agile approach to software development while providing acceptable conformance to the governance of the SDLC. The hybrid Scrum model is depicted in Figure 4. In this model, the modified or additional elements of the hybrid Scrum model which are not

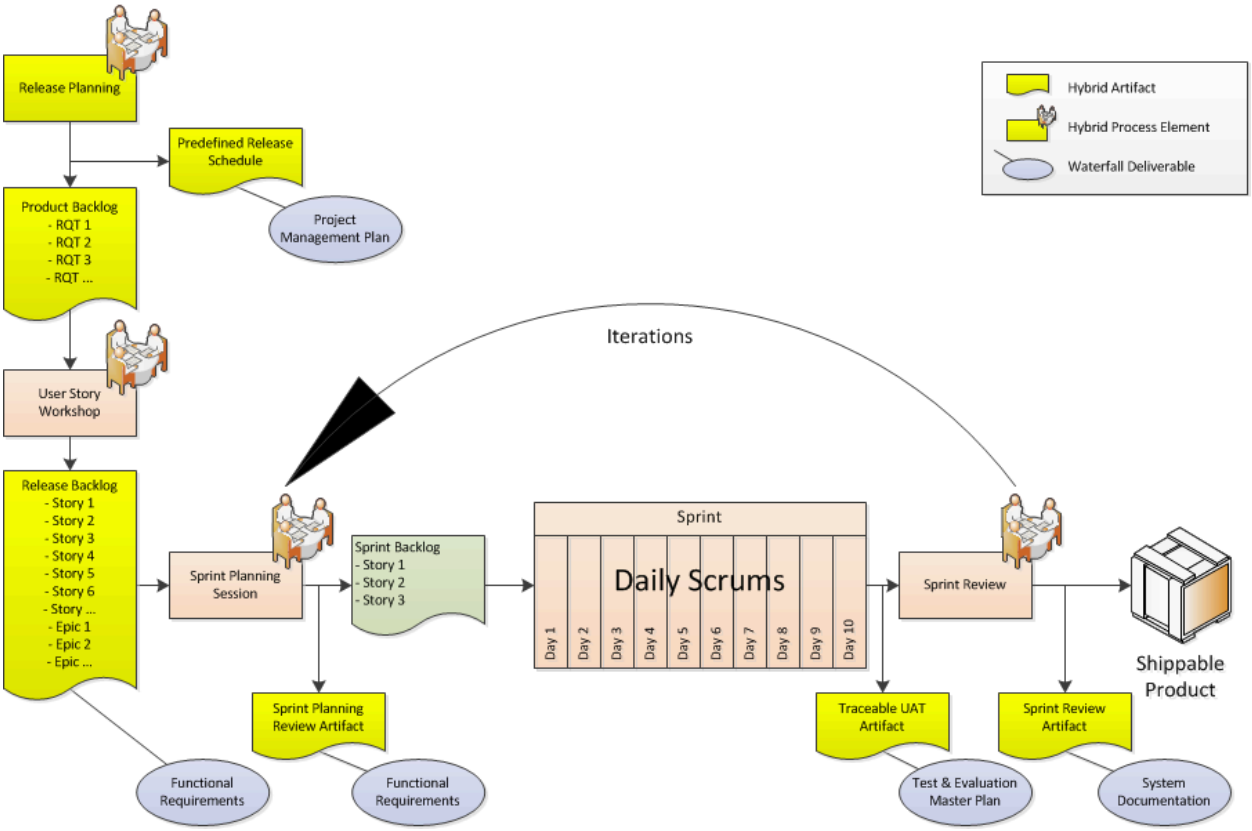


Figure 4: Hybrid Scrum Model

part of Scrum-proper are highlighted in yellow. The deliverables of the waterfall SDLC which they satisfy are referenced in the blue oval callouts. Aydin, et. al [15] note that

many organizations tailor what is necessary with agile methods in order to meet the needs of their organization, and to still implement agile methods. Optimally, the CG-LIMS Scrum Team would be tailoring Scrum to meet the needs of the project purely for efficiency and quality sake. This hybrid approach, however, is for coexistence sake; to be able to satisfy both processes for the foreseeable life of the project.

The hybrid Scrum methodology proceeds as follows:

- *Release Planning* - a hybrid process element - is a series of planning sessions conducted prior to each release cycle. Planning is initially accomplished with IEEE 830 requirements only.
- *Predefined Release Schedule* - a hybrid artifact - is a regularly refined product of each Release Planning. At a high level, this shows the entire release schedule for the project.
- *Product Backlog* - a hybrid artifact - is a change to the normal Scrum Product Backlog. It is populated with IEEE 830 Requirements, and it is groomed each release cycle through the Release Planning.
- User Story Workshop - a normal Scrum process element.
- *Release Backlog* - a hybrid artifact - is an additional backlog which is used to define the scope of the current release. It is populated by the Scrum Team and PMO in User Story Workshops, and it serves the purpose that a Product Backlog serves in normal Scrum.
- Sprint Planning Session - a normal Scrum process element.
- *Sprint Planning Review Artifact* - a hybrid artifact - is a product of the Sprint Planning Session which captures the scope of the current sprint which was just captured.

- Sprint Backlog - a normal Scrum process element.
- Sprint and Daily Scrums - are normal Scrum process elements.
- *Traceable UAT Artifact* - a hybrid artifact - is a clearly defined UAT for each user story which is used to test features and trace their verification from the signed UAT back to the IEEE 830 requirements which they satisfy.
- Sprint Review - a normal Scrum process element.
- *Sprint Review Artifact* - a hybrid artifact - is a product of the Sprint Review which captures the satisfactory accomplishment and acceptance of each of the stories in the sprint, with traceability back to original IEEE 830 requirements.
- Shippable Product - a normal Scrum process element.

The overarching concern of the PMO is that, as the Agile Manifesto states [9], "[valuing] individuals and interactions over processes and tools", and "[valuing] working software over comprehensive documentation." Doing so will produce a product which customers feel that they "own" because they took part in the development, rather than being brought in at certain touch points in the project to provide blind reviews and feedback. Doing so will also generate product faster, putting smaller increments in the customers hands sooner than later, rather than waiting until the end of a multi-year project to "turn on the switch" of a complete system. This way customer confidence is increased, as well as continued support by customers and sponsors alike as feedback comes in throughout the duration of the project, rather than at the end. Thus, working with the constraints of the waterfall SDLC and taking this hybrid Scrum approach was imperative to the PMO (Product Owner) in being successful.

4.1 *Predefined Release Schedule*

Normally, an agile approach to development does not plan the entire project out from start to finish as in traditional software development models. The traditional WBS which lays out milestones, tasks, dependencies, resource loading, and predecessors is abandoned for the creation of a Product Backlog of features. The challenge which the PMO faced, and specifically the Project Manager (an O-6 Captain) who answers to an Executive Steering Committee (ESC) made of flag-level officers, was to present a forward looking plan, and report progress to-date in a manner which instilled confidence in the success and continued funding of the project. This meant presenting an end-to-end sequential plan of releases scheduled for the entire project. While this is not optimal, there is still a way to accomplish the scheduling of releases and still remain agile.

Reference Figures 4 and 5 for this discussion. The high level release schedule

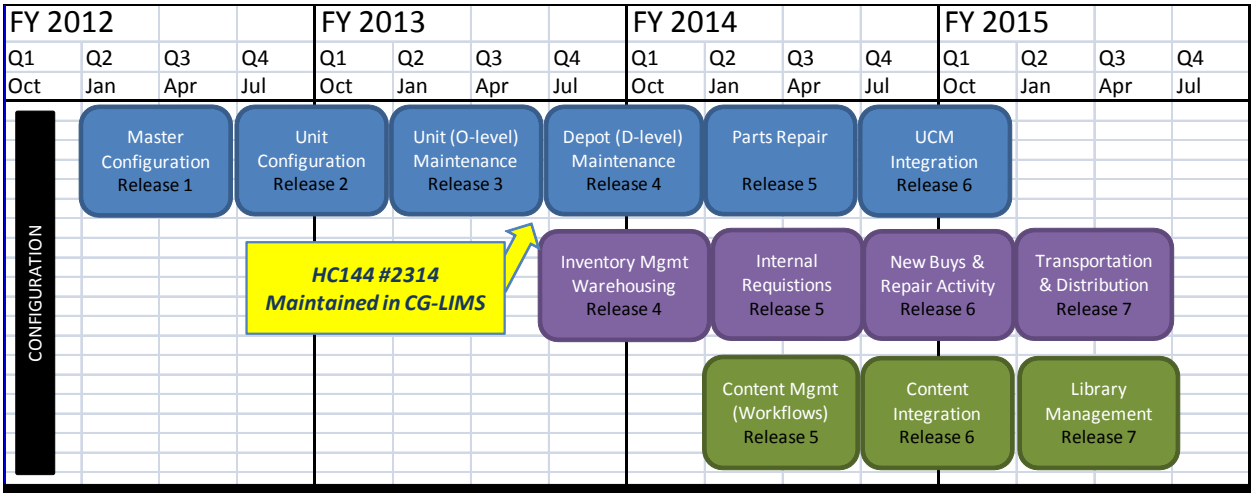


Figure 5: Predefined Release Schedule

depicted satisfies the need for this type of planning for oversight and governance purposes. The Scrum Team and PMO participate in Release Planning sessions prior to

each release cycle begins in order to accomplish high level planning based upon the IEEE 830 requirements already defined for the project. In these sessions, which normally encompass a weeks worth of meetings, the IEEE 830 requirements are reviewed for relevancy to anticipated scope for the oncoming release, and initially prioritized into the release if they are relevant. Focus is concentrated on the oncoming release, with less attention paid to requirements being specifically “bucketed” into future releases. Future releases are discussed, but in increasing generality as releases lay out into the future. From this effort, a high level release schedule like the one that displayed in Figure 5 is produced. IEEE 830 requirements primarily make up the Product Backlog, along with the occasional epic, but not the normal user stories represented in normal Scrum implementations.

This divergence from the Scrum approach does not violate any agile principles; in combination with the Release Backlog, this solution still satisfies agility by keeping the focus on the closest work to current day - the oncoming release. Granted, a picture based on the team’s best guesstimates is painted of the entire project end-to-end, and this picture can change as clarity of the scope of each release comes into view through the project, but it satisfies a specific deliverable for the waterfall SDLC - the Project Management Plan. The primary difference between an end-to-end WBS for a traditional project and the predefined release schedule for the hybrid Scrum approach is that no firm expectations are specifically promised for later releases beyond the next one which is upcoming. Team and Stakeholders alike are educated on the concept of having the “furious four” variables of a project - Time, Budget, Quality, and Scope [16]. The high level release schedule being defined as it is suggests that time and budget are

generally going to be fixed, and quality is always expected, so that can be considered fixed as well. This leaves scope to be leveraged each release to the Product Owner's satisfaction. The scope of future releases isn't known or considered relevant beyond the oncoming release. The defined scope for a release can be determined when it is relevant to the team; just before the future release is about to become the oncoming release.

4.2 The Release Backlog

The Release Backlog is a new element of the hybrid Scrum methodology. Scrum normally employs two backlogs - the Product Backlog and the Sprint Backlog. The need to be able to transition IEEE 830 requirements into epics and eventually user stories is facilitated by having this new backlog. See Figure 6 for the placement of the Release Backlog in the process defined for the CG-LIMS project.

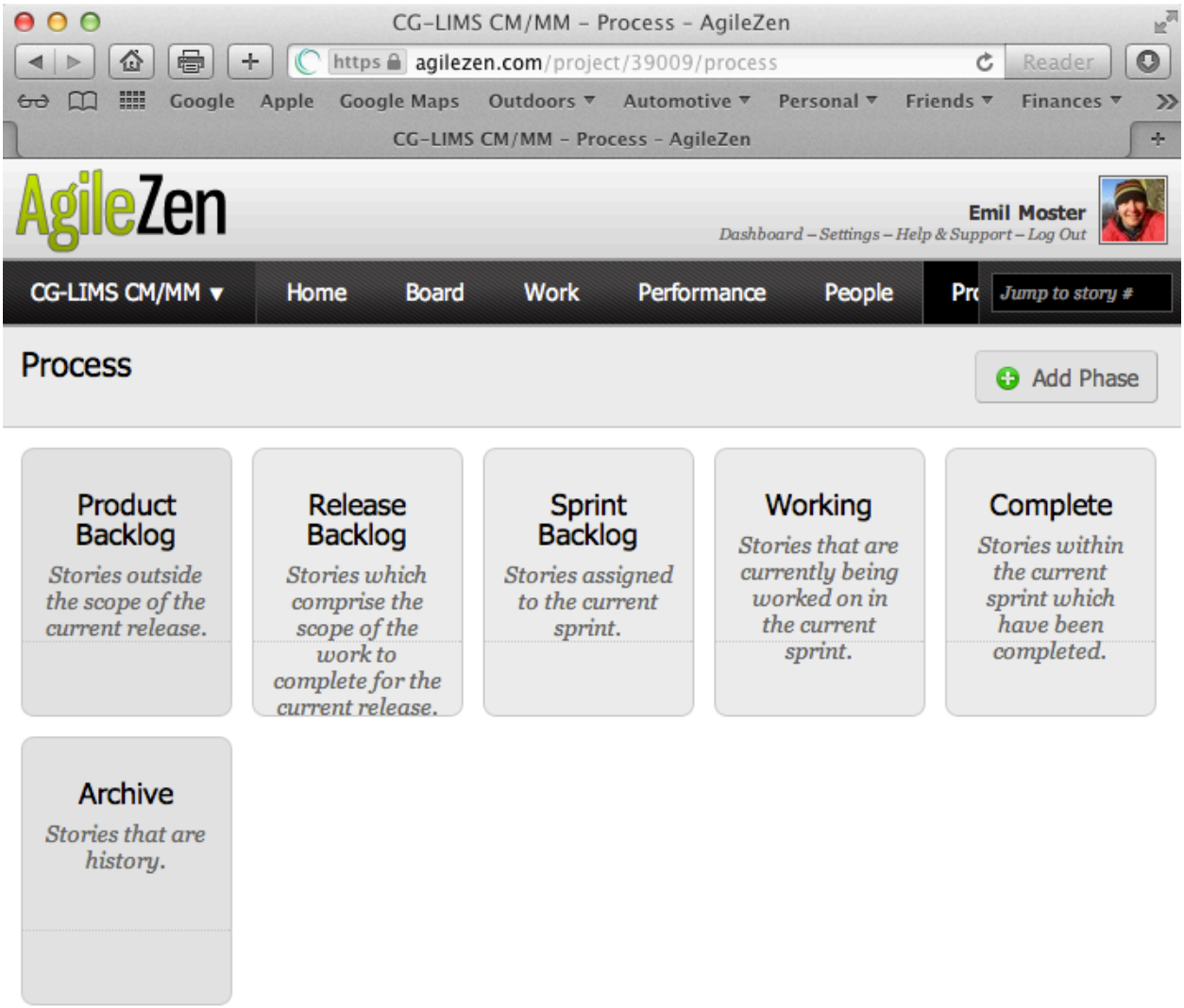


Figure 6: The Release Backlog Inserted into the Process

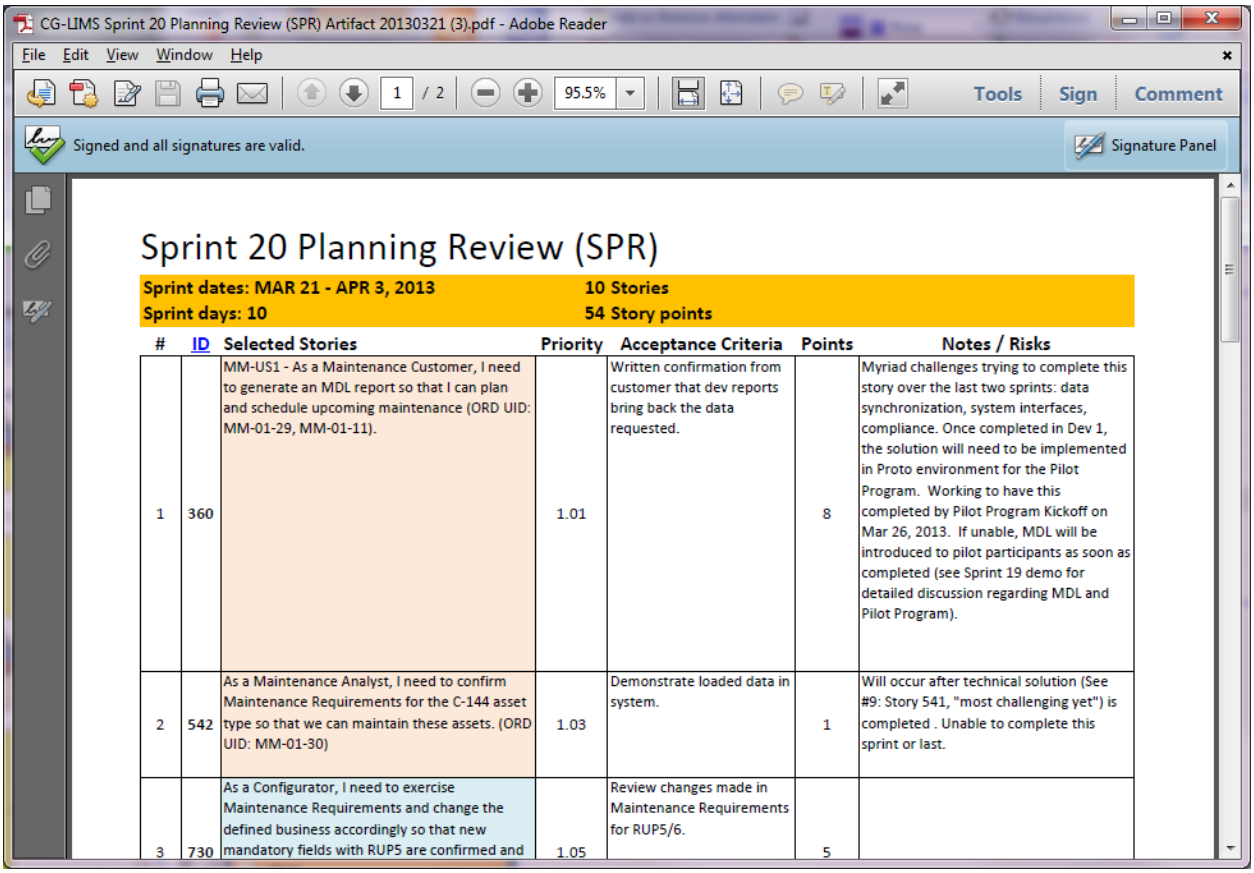
Only the current release is considered close enough and relevant enough to give detail to, so the Scrum Team and PMO conduct User Story Workshops for the current release to create, prioritize, and estimate epics and stories from the IEEE 830 requirements in the Product Backlog. These go into the Release Backlog, and define the more detailed scope of the oncoming or current release. The Release Backlog is populated each release cycle; as it runs empty for the current release, it is filled from the Product Backlog in User Story Workshops prior to the next release cycle. The Release Backlog is prioritized and groomed in the same manner as the Product Backlog in

normal Scrum is. When it is time to plan a sprint, the Scrum Team selects from the Release Backlog versus the Product Backlog.

The Release Backlog, with its detailed information captured for each user story, represents functional requirements defined for the system. They are more detailed than the IEEE 830 requirements for the project, referencing specific features and business functionality which will be delivered in the given release. This satisfies the Functional Requirements deliverable in the waterfall SDLC process, for the Design PER.

4.3 Sprint Planning Review Artifact

The Sprint Planning Review Artifact is a signed document that is created after each Sprint Planning Session to represent the stories and work scoped into the current sprint. It captures notes relevant to the execution of the stories, and also defines the acceptance criteria which the development team will satisfy for each story in the sprint (see Figure 7). In hybrid Scrum, this artifact in addition to the Release Backlog satisfies the Functional Requirements documentation for the waterfall SDLC, where there is no element of normal Scrum which satisfies this deliverable otherwise.



CG-LIMS Sprint 20 Planning Review (SPR) Artifact 20130321 (3).pdf - Adobe Reader

File Edit View Window Help

1 / 2 95.5% Tools Sign Comment

Signed and all signatures are valid. Signature Panel

Sprint 20 Planning Review (SPR)

Sprint dates: MAR 21 - APR 3, 2013 **10 Stories**
Sprint days: 10 **54 Story points**

#	ID	Selected Stories	Priority	Acceptance Criteria	Points	Notes / Risks
1	360	MM-US1 - As a Maintenance Customer, I need to generate an MDL report so that I can plan and schedule upcoming maintenance (ORD UID: MM-01-29, MM-01-11).	1.01	Written confirmation from customer that dev reports bring back the data requested.	8	Myriad challenges trying to complete this story over the last two sprints: data synchronization, system interfaces, compliance. Once completed in Dev 1, the solution will need to be implemented in Proto environment for the Pilot Program. Working to have this completed by Pilot Program Kickoff on Mar 26, 2013. If unable, MDL will be introduced to pilot participants as soon as completed (see Sprint 19 demo for detailed discussion regarding MDL and Pilot Program).
2	542	As a Maintenance Analyst, I need to confirm Maintenance Requirements for the C-144 asset type so that we can maintain these assets. (ORD UID: MM-01-30)	1.03	Demonstrate loaded data in system.	1	Will occur after technical solution (See #9: Story 541, "most challenging yet") is completed . Unable to complete this sprint or last.
3	730	As a Configurator, I need to exercise Maintenance Requirements and change the defined business accordingly so that new mandatory fields with RUP5 are confirmed and	1.05	Review changes made in Maintenance Requirements for RUP5/6.	5	

Figure 7: Sprint Planning Review Artifact

4.4 Traceable UAT Artifact

In the hybrid Scrum approach, IEEE 830 requirements which are kept in the Product Backlog must trace forward to the UATs which are performed to verify that requirements are met, and likewise, the UATs must trace backwards to the requirements which they satisfy. This traceability is accomplished through multiple steps in the hybrid Scrum model. First, IEEE 830 requirements in the Product Backlog are decomposed in User Story Workshops into epics, and then eventually into the multiple user stories which develop and implement the features for the given requirement (see Figure 8). The number of epics and eventually user stories which represent the features of the requirement are dependent on each requirement. In the CG-LIMS project, requirements

have decomposed to as few as one story, and as many as twelve stories. Note in Figure 8 that as epics and stories are created, they are assigned the UID for the

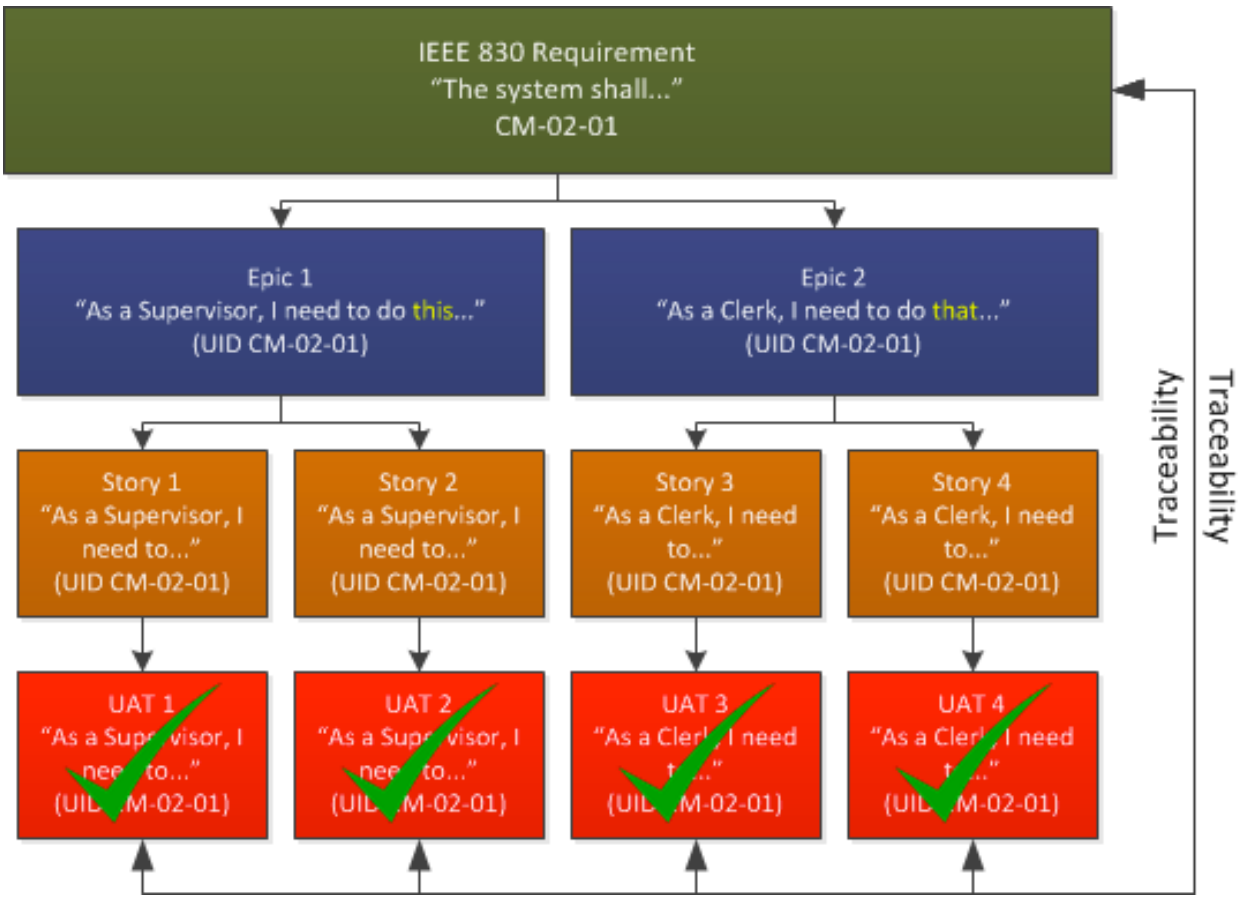


Figure 8: Requirements Traceability to UATs

requirement in the Product Backlog. As stories are worked in respective sprints, the UATs for those stories are written as well. UATs, likewise, are given the UID of the requirement to which they trace so as to have end-to-end traceability for satisfaction of requirements. Considering there may be two or more stories and their UATs which trace to a given requirement, status of the completion of stories and their UATs must be tracked in order to determine when a requirement is completely satisfied by the related UATs. Based on priorities and logical grouping, some requirements may carry over multiple sprints before they are completed fully. The CG-LIMS Team utilizes IBM

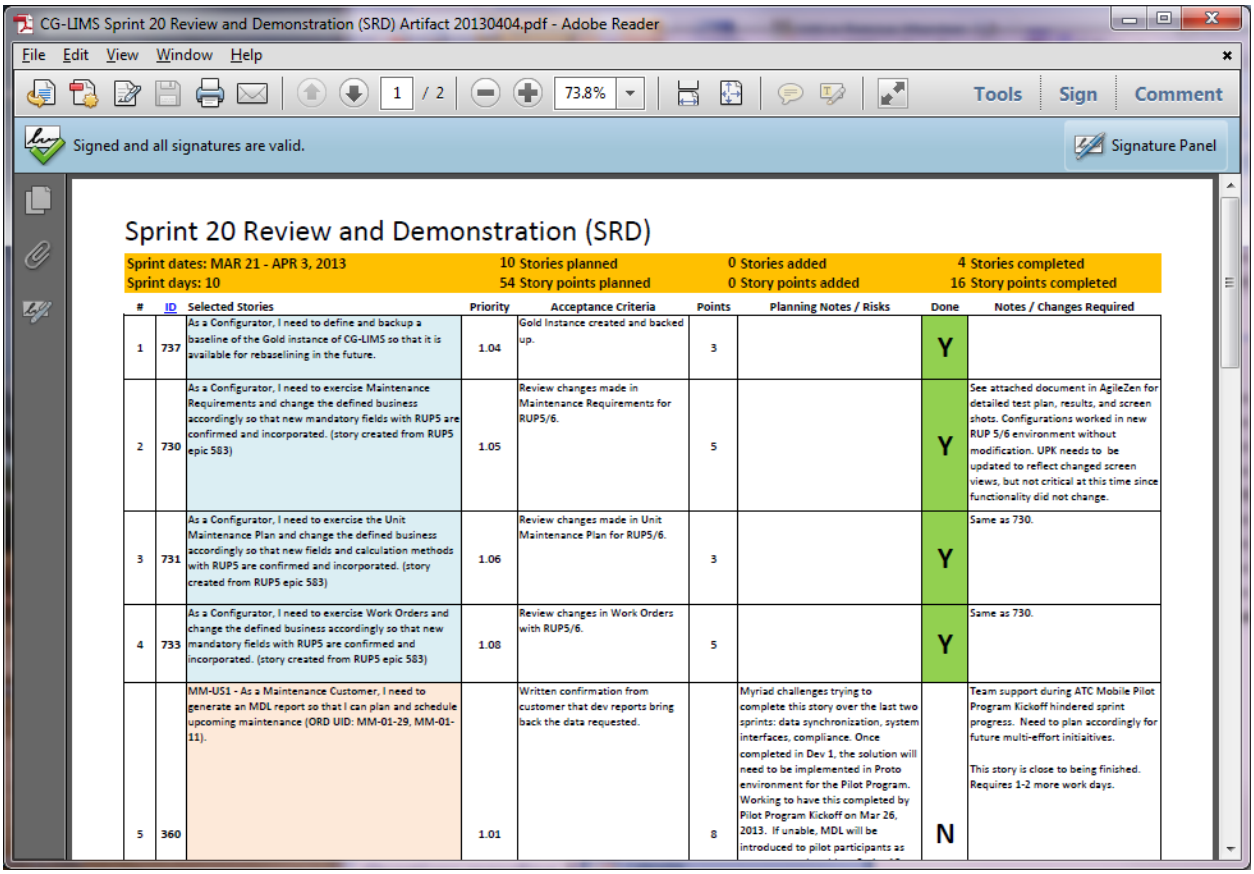
DOORS as their Requirements Management tool/repository, so a status flag denoting that a requirement is complete, based upon the stories being defined for a given requirement is set on the requirement when it is decomposed to all of its related stories. This way, it can be traced in the tool that a requirement is complete when the UATs are captured for each of the stories.

Normally, the Scrum approach to creating user stories is to conduct a User Story Workshop where brainstorming and other methods are used to draw features out of Product Owners and customers during the session. Our method uses the IEEE 830 requirements as a basis for defining features, and Product Owners and customers lend input into the creation of epics and user stories which come from the requirements. Furthermore, testing in a purely Scrum approach would not develop significant testing artifacts. Testing would be satisfied by simply scribing constraints to be met and validations to be confirmed on the back of index cards. This would not satisfy the waterfall SDLC requirement for a Test and Evaluation Master Plan (TEMP), but UATs which trace through user stories to requirements do. And so the TEMP deliverable which is required for the PER of the Development and Testing phase is satisfied.

4.5 Sprint Review Artifact

Scrum has a sprint review “ceremony” where the Scrum Team reviews the work that has been completed with the customers, Product Owner, and stakeholders to demonstrate developed features. This is a normal meeting, but the results of the review and demonstration are not normally tracked in an artifact. The hybrid Scrum approach creates the Sprint Review Artifact (see Figure 9) which itemizes the stories completed, notes regarding the review of the stories, and positive confirmation that acceptance

criteria was met for each story of the sprint. This artifact supports the traceability of the completion of stories/UATs, as it is signed as well and held as a project artifact. Additionally, it supports System Documentation deliverable for the PER of the Implementation waterfall SDLC phase. It does not stand alone in this purpose,



Sprint 20 Review and Demonstration (SRD)

Sprint dates: MAR 21 - APR 3, 2013 | 10 Stories planned | 0 Stories added | 4 Stories completed
Sprint days: 10 | 54 Story points planned | 0 Story points added | 16 Story points completed

#	ID	Selected Stories	Priority	Acceptance Criteria	Points	Planning Notes / Risks	Done	Notes / Changes Required
1	737	As a Configurator, I need to define and backup a baseline of the Gold instance of CG-LIMS so that it is available for rebaselining in the future.	1.04	Gold Instance created and backed up.	3		Y	
2	730	As a Configurator, I need to exercise Maintenance Requirements and change the defined business accordingly so that new mandatory fields with RUP5 are confirmed and incorporated. (story created from RUP5 epic 583)	1.05	Review changes made in Maintenance Requirements for RUP5/6.	5		Y	See attached document in AgileZen for detailed test plan, results, and screen shots. Configurations worked in new RUP 5/6 environment without modification. UPK needs to be updated to reflect changed screen views, but not critical at this time since functionality did not change.
3	731	As a Configurator, I need to exercise the Unit Maintenance Plan and change the defined business accordingly so that new fields and calculation methods with RUP5 are confirmed and incorporated. (story created from RUP5 epic 583)	1.06	Review changes made in Unit Maintenance Plan for RUP5/6.	3		Y	Same as 730.
4	733	As a Configurator, I need to exercise Work Orders and change the defined business accordingly so that new mandatory fields with RUP5 are confirmed and incorporated. (story created from RUP5 epic 583)	1.08	Review changes in Work Orders with RUP5/6.	5		Y	Same as 730.
5	360	MM-US1 - As a Maintenance Customer, I need to generate an MDL report so that I can plan and schedule upcoming maintenance (ORD UID: MM-01-29, MM-01-11).	1.01	Written confirmation from customer that dev reports bring back the data requested.	8	Myriad challenges trying to complete this story over the last two sprints: data synchronization, system interfaces, compliance. Once completed in Dev 1, the solution will need to be implemented in Proto environment for the Pilot Program. Working to have this completed by Pilot Program Kickoff on Mar 26, 2013. If unable, MDL will be introduced to pilot participants as	N	Team support during ATC Mobile Pilot Program Kickoff hindered sprint progress. Need to plan accordingly for future multi-effort initiaitives. This story is close to being finished. Requires 1-2 more work days.

Figure 9: Sprint Review Artifact

however. Significant documentation captured from COTS documentation, story details, and specifications written in developing interface solutions all provide the necessary detail to satisfy system documentation needs.

4.6 Requirements metrics tracked

Metrics which are innate to Scrum are the Sprint Burndown, Team Velocity, and the Release Burndown. Consider the metrics which are not normally tracked in Scrum -

those metrics which are innate to a waterfall process model. Requirements defined with the IEEE 830 standard are not normally a part of Scrum. IEEE 830 requirements are tracked in the hybrid Scrum approach, however. It is necessary to track requirement completion to satisfy the waterfall SDLC and its customers; the normal way to track progress in Scrum - story point completion - only provides value to the Scrum Team.

While it is necessary to track requirements completed per sprint and per release, it is not necessary to track requirements completed in a daily Sprint Burndown; this level of granularity is too low to provide value. Likewise, it is not necessary to track Team Velocity based upon requirements completion. Team Velocity is used for planning purposes by the Scrum Team and the PMO, and there is no value for tracking velocity with a second unit of measure.

Requirements traceability through user stories to UATs and back facilitates the ability to make the jump from points to system requirements counts. UATs are completed for stories which are based upon points, and these trace to requirements which are the target unit. Another adaptation made to facilitate requirements completion based upon story completion was giving partial credit for requirements completed. As demonstrated in Figure 8, stories, and thus UATs can be completed successfully, but a single story or even multiple stories may not constitute satisfying a requirement completely. If a given system requirement was decomposed to four user stories and four UATs, the decision could be made to give partial credit for a requirement completion if some, but not all of the stories were complete. This allowed the PMO to report progress being made from sprint to sprint even if only in 1/2 credit units (the

consensus was that giving less than 1/2 credit was not valuable in improving the tracking of requirements completion progress).

The first metric tracked in hybrid Scrum is Requirements per Release (see Figure 10). As project success is ultimately measured in requirements from the ORD being

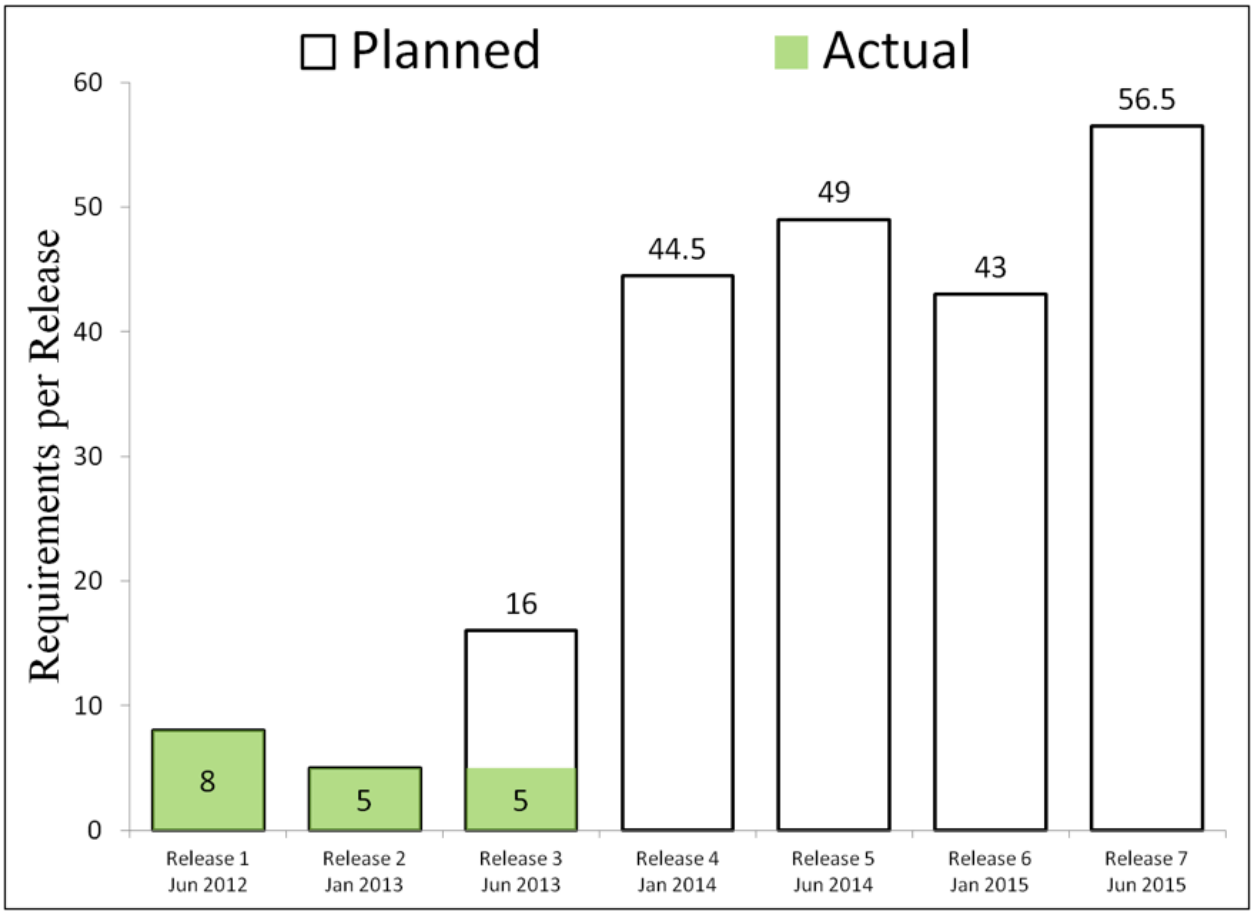


Figure 10: Requirements per Release

completed, this metric uses requirements as the unit of measure. As requirements trace directly to stories and UATs, and stories are planned for releases, requirements planned can be tracked. Likewise, as stories are completed for a release, and their UATs are signed off, the requirements which trace to these stories and UATs are also tracked for completion. In this manner, the traceability which is a hybrid Scrum artifact provides the

mechanism for accomplishing this waterfall deliverable, simply by doing the normal activities of Scrum.

The second metric which tracks requirements is Cumulative Requirements accomplished. This chart, shows planned and actual completed requirements per

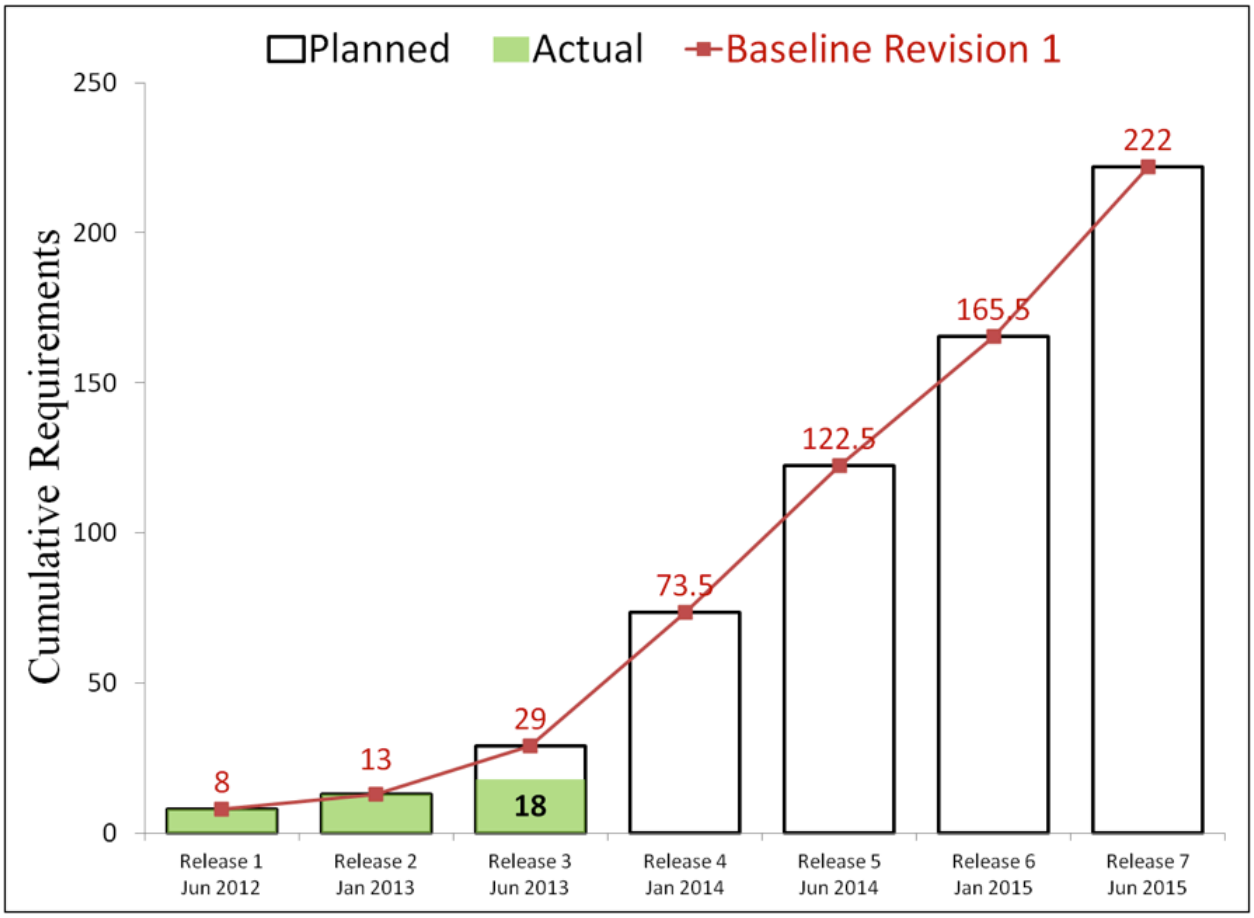


Figure 11: Cumulative Requirements per Release, Baseline Revision 1

release, and additionally, it shows the desired cumulative total of requirements through the extent of the project. This particular chart shows an actual change to the baseline requirements for the project - Revision 1. Re-baselining requirements is a waterfall process activity, yet the metrics for such activities can still be shown in relation to the progress of the Scrum Team in a hybrid Scrum approach.

As simple as the two new metrics may seem in this hybrid Scrum approach, there is measurable value in communicating to project Sponsors and Stakeholders the progress made in the project in terms to which they can relate - completed requirements. This is especially the case when governance directed in the SDLC requires that progress be measured a certain way. These metrics are not arbitrary either; they are directly correlated and traceable to the work being accomplished with agile methods.

And so it is clearly demonstrated that the hybrid Scrum methodology has significant, tangible elements which provide specific value in a situation where the disparity between two coexisting processes could be considered otherwise insurmountable. Changes in the normal Scrum process afford the team the activities which accommodate waterfall activities and deliverables; just enough to be able to satisfy requirements and PERs, but not so much as to make hybrid Scrum unrecognizable as an agile methodology. Additional document artifacts are implemented in a manner which minimizes the disruption to the agility of the team, while satisfying waterfall SDLC process requirements, and specifically satisfying project Sponsors and stakeholders.

CHAPTER 5: CONCLUSION

Defining and implementing a hybrid Scrum methodology was quintessential to the success of the project in the case study. While the existing SDLC was in place and in force in organization, it was also evident at the beginning of the project that if the project team did not deliver successes quickly, the project was in jeopardy of being cancelled. In this respect, the organization's policies made it it's own worst enemy. Taking a waterfall approach to development would not have generated the successes early enough to keep the project off the chopping block. Out of necessity was born the hybrid Scrum methodology. To date, the project has not completed, but it continues to track successfully from release to release. Features continue to be delivered, and customers on the ground level who use the CG-LIMS in day to day activities appreciate the attention spent in customer interaction; a benefit of agile methods. In the same vein, Product Owner, Sponsor, and Stakeholder appreciate the visibility and participatory role they play in Scrum and hybrid Scrum activities, as well as the confidence that the standard to which they know the project is ultimately held - the SDLC process - is being satisfied. The benefits of the hybrid Scrum approach are being realized: regular and frequent customer interaction, potentially shippable product being delivered every two weeks, and substantive releases being delivered every six months. With the transparency that the Scrum Team provides to all teams and Stakeholders, surprises are minimized, deficiencies are detected early and addressed, and Scrum Team morale is high, contributing to a high operations tempo at a sustainable pace.

Key to the success of the hybrid Scrum approach is to clearly define what deliverables from the waterfall process are required, and then clearly defining how the

team will accomplish meeting and delivering them. A highly disciplined Scrum Team is also a key to success; adopting agile methodologies, and leaving behind inefficiencies of “the way it has always been done” takes dedication from team members and management alike. In the hybrid Scrum implementation used for CG-LIMS, six specific methods or artifacts were modified or added in order to meet the specific needs of the USCG SDLC, but as hybrid Scrum is applied in organizations in need of the same results, other methods may be modified as well in order to meet other specific organizational needs.

Another specific benefit of the experience of this implementation is the realization that not only does hybrid Scrum provide a means for two disparate processes - Scrum and waterfall - to coexist for an organization, but it also provides a transitory mechanism for organizations to migrate from a waterfall process to Scrum in an incremental approach. Organizations can’t always afford to “flip the switch” and turn off a waterfall process overnight and to turn on an agile Scrum approach the next day. Inhibitors to doing so could include budgetary reasons, organization logistics, training, and political reasons. As with most successful process change implementations in IT organizations, success does not volunteer organically; it usually is carried to the forefront by a champion who will be ever-vigilant in overcoming minor setbacks for the bigger wins which come with patience and perseverance. Based on the experience of implementing hybrid Scrum with the CG-LIMS project, it is easy to envision other entities having champions who use hybrid Scrum to transition their own organizations from waterfall to Scrum.

While the study is conclusive in its limited scope, one major shortcoming is just that - its limited scope. A further, more thorough study may reveal even more value to be gleaned in satisfying more SDLC/waterfall deliverable requirements with additional modifications or improvements to the hybrid Scrum approach. This thesis does conclude that combining the two approaches - Scrum and waterfall - can be accomplished, and even show success with both processes.

REFERENCES

[1] “Command, Control, Communications, Computers and Information Technology (C4&IT) System Development Life Cycle (SDLC) Policy” - Commandant Instruction 5230.66A, *United States Coast Guard*, 11 December 2009, http://www.uscg.mil/directives/ci/5000-5999/CI_5230_66A.pdf

[2] “IEEE Recommended Practice for Software Requirements Specifications” - IEEE Standard 830-1998, *IEEE*, 1998, http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5841.

[3] “Scaling Agile: An Executive Guide”, Ambler, *IBM agility@scale Whitepaper,* 2010, https://www.ibm.com/developerworks/mydeveloperworks/blogs/ambler/entry/scaling_agile_an_executive_guide10.

[4] “Agile Methodologies in DHS (Draft)”, Schwartz, et al. (DHS Agile Working Group), 2011.

[5] “Homeland Security Tackles Agile Development”, Hoover, *Information Week Government,* February 28, 2012, http://www.informationweek.com/government/enterprise-architecture/homeland-security-tackles-agile-developm/232601660.

[6] “Chapter 19: Coexisting with Other Approaches”, Cohn, *Succeeding with Agile: Software Development Using Scrum,* Addison-Wesley, 2010.

[7] “Bridging the Gap: Agile Projects in the Waterfall Enterprise”, Sliger, *Better Software*, July/August 2006, pp. 26-31.

[8] “FBI's Sentinel Project: 5 Lessons Learned”, Foley, *Information Week Government,* August 3, 2012, http://www.informationweek.com/government/enterprise-applications/fbis-sentinel-project-5-lessons-learned/240004888.

[9] “The Agile Manifesto”, The Agile Alliance, 2001, http://www.agilemanifesto.org.

[10] “ISO 9001 and Agile Development”, McMichael and Lombardi, *Proceedings of the Agile 2007 Conference*, IEEE Computer Society, 2007, pp. 262-265.

[11] “Empirical Studies in Software Development Projects: Field Survey and OS/400 Study”, Phan, Vogel, and Nunamaker, *Information & Management, Volume 28, Issue 4,* April 1995, pp. 271-280, http://www.sciencedirect.com/science/article/pii/037872069400046L.

[12] “Managing the Development of Large Software Systems”, Royce, *IEEE Computer Society Press*, 26 August 1970, http://dl.acm.org/citation.cfm?id=41765.41801.

[13] “A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction”, Mann and Maurer, *Proceedings of the Agile Development Conference, IEEE Computer Society,* pp. 70-79, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1609806&isnumber=33795.

[14] “Coast Guard Logistics Management Information System”, USCG, http://www.uscg.mil/acquisition/cglims/.

[15] “An Agile Information Systems Development Method in Use”, Aydin, Harmsen, Slooten, and Stegwee, *Turkish Journal of Electrical Engineering and Computer Science, Volume 12, Issue 2,* 2004, pp. 127-138, http://journals.tubitak.gov.tr/elektrik/issues/elk-04-12-2/elk-12-2-5-0404-6.pdf.

[16] “The Furious Four”, Rasmusson, *The Agile Samurai*, The Pragmatic Bookshelf, 2011.