Final Report of the NASA Office of Safety and Mission Assurance Agile Benchmarking Team

July 29, 2016

Report Approval and Revision History

This document was submitted to the Chief, Safety and Mission Assurance on June 28, 2016.

Version	Comments
July 29, 2016	Original Release

Martha Wetherholt
NASA Software Assurance Manager and Technical Fellow

Table of Contents

Note: The term “Agile” in this document is used as a shortened form of “Agile development framework” and does not infer any particular mode or type of Agile process.

Executive Summary

To ensure that the NASA Safety and Mission Assurance (SMA) community remains in a position to perform reliable Software Assurance (SA) on NASA’s critical software (SW) systems with the software industry rapidly transitioning from waterfall to Agile processes, Terry Wilcutt, Chief, Safety and Mission Assurance, Office of Safety and Mission Assurance (OSMA) established the Agile Benchmarking Team (ABT). The Team's tasks were:

1. Research background literature on current Agile processes
2. Perform benchmark activities with other organizations that are involved in software Agile processes to determine best practices,
3. Collect information on Agile-developed systems to enable improvements to the current NASA standards and processes to enhance their ability to perform reliable software assurance on NASA Agile-developed systems
4. Suggest additional guidance and recommendations for updates to those standards and processes, as needed.

(See Appendix A for the Agile Benchmarking Team Charter.)

The benchmarking team began organizing in August 2015, based on a preliminary charter. Initial team activities focused on researching available resources on Agile Methodologies and appropriate NASA Software Assurance (SA) standards and processes to support the generation of an effective questionnaire for use in conducting the benchmarking interviews. The final interviewee list consisted of 21 organizations that represented a wide variety of organizational roles and perspectives related to their involvement with and/or their utilization of the Agile methodologies. Six of the interviewees were from internal NASA organizations, twelve were external industry organizations involved in a wide range of commerce activities, one was a long time consultant involved in a wide range of projects, one represented a university, and one represented a government agency outside of NASA. The Team believes the 21 organizations interviewed represented a good cross-section of perspectives on the practical implementation of Agile on software systems.

While other Agile studies have focused on its use as a software development process, this benchmarking effort was focused on the software assurance role in the Agile development process. However, for a complete picture, findings and recommendations for software management, engineering and software assurance are addressed herein. Specific, high level findings are summarized as are the Team’s recommendations. In short, there are several roles for Software Assurance to contribute to the Agile development process and support the Agile teams in creating safe, reliable, high quality systems.

The team's key findings and recommendations are summarized below.

Key Findings: (Note: These findings are described in detail in Section 5.0)

1. Agile approaches in use: Organizations with multiple smaller project teams and short term projects (1-3 teams with 4-7 team members each and completion within 6 months to a year) tended to be more engaged with a purer Agile approach while larger, more complex projects that extend beyond a year tended to have waterfall elements such as an overarching design architecture, rolled up intermediate releases, and reviews.
2. Documentation is an important part of the Agile process, but it is kept to a minimum and most documentation is kept within the suite of tools used to manage development.
3. Most organizations interviewed have very limited actual Software Assurance. For those projects with Software Assurance, it was integral to the teams.
4. A strong teamwork approach is essential. Teamwork is what makes Agile successful.
5. Projects with multiple, simultaneous sprint teams need both strong internal team cohesion and a means to be related to the larger project and other teams.
6. Tools, the right tools, are essential, especially for larger projects.
7. Functional requirements need to exist for Agile projects and are important for their success. They just aren't always referred to as "requirements".
8. Continuous self-improvement of the team is vital to the Agile process.
9. Train the team as a group so everyone has a common understanding.
10. Strong integral verification and validation are key for project success.
11. Creating and maintaining the system view of the overarching architecture is necessary for large projects.
12. Management and acquisition approaches need to change to allow for alternative deliverables.

Key Recommendations: (Full recommendations are found in Section 6.)

Recommendations are divided into three categories, based on the organizational area best suited to address them.

A. Key Recommendations for Managing Agile SW Developments

1. NASA needs to establish a guideline to determine how and when to perform an Agile software development for a project, based on the size, criticality, and complexity of the project.
2. An organization's maturity and processes need to be in place before the organization tries to develop software using Agile. The organization's capabilities need to be defined in processes that address how a development will be managed, performed, and structured using an Agile work environment

including proper training, coaching, management, and selection of personnel who can adapt to the rapid and responsive environment of Agile.

3. Ensure that the entire team, their management, and any SME and SA personnel are trained together on Agile benefits, processes, and chosen methodologies.
4. For a large project or program, Agile team representatives, SA and management need to agree on and document an Agile methodology, schedule, and any needed adaptations to that methodology.
5. Contract requirements need to be written to provide the desired information in forms and time frames applicable for any software development life cycle model, including Agile processes. Include in the contract expected contractor and government involvement as well as product requirements for the government SA audits, analyses, and review. Agile type deliverables and timing need to be accommodated but Agency documentation needs should also be stated.
6. Ensure teams follow the chosen Agile approach in order to get the benefits of Agile, e.g. they should hold regular Sprint Retrospectives or Scrum of Scrums to evaluate team processes, tools, and other potential improvement areas.

B. Key Recommendations for Software/Systems Engineering for Agile SW Developments

1. Ensure that the Agile approach to be used is chosen and tailored to meet the customer's needs such as delivery of certifications or hazard reports, whether the customer is a company or NASA.
2. Use Agile mechanisms such as Sprint Retrospectives and Reviews to address issues and improve the Agile processes, team interactions, and product quality. This takes flexibility and a willingness to makes changes where needed.
3. Establish consistent definitions of "Done" across the project for each sprint, for any releases, and for the final product.
4. Early in the Agile project, have the team(s) choose a tool suite and train the entire team (including SA) on the tool usage.
5. For larger, multi-sprint team projects, it is important to establish several things up front. All teams need to be part of the coordinated decisions for establishing how overarching system components or activities will function. At a minimum address:
    a. Fault Management strategies
    b. Continual integrated testing
    c. Overarching design and requirements
    d. Safety and reliability analyses
    e. Configuration management
    f. Sprint lengths and coordination
    g. Establishment of a Scrum of Scrums for coordination

h. Verification and validation approaches
i. Coordination and completion steps and criteria

Each sprint team needs to address these issues, but for success on the larger project, the multiple sprint teams need to coordinate to integrate the whole project.

6. For large projects, the structure and management of verification and validation (V&V) needs to have an overarching systems approach. The integration of multiple sprint teams needs special attention to manage and address issues when any sprint teams are falling behind on testing and other forms of V&V. This is necessary to maintain the quality of the software and the overall V&V process, ensuring a systematic approach to integration of the Agile teams' continual outputs.

C. Key Recommendations for SW Assurance for Agile Software Developments

1. SA Staffing:
   Staff the SA on a project at the appropriate levels to allow SA to be embedded in the development teams.
   a. When SA cannot be embeded in all the daily sprint team activities, leaving SA personnel to cover several teams, each SA should cover no more than 2 to 4 sprint teams, depending on the teams.
   NOTE: However, with multiple simultaneous sprints, often the Sprint Review meetings and backlog review meetings will all occur at about the same time so a single SA member could only cover one team per sprint.
2. When independent SA is set up as SMEs that are part of the greater project, they are viewed as part of the teams and a shared resource, rather than outsiders slowing the progress of teams.
3. Regardless of who is providing Software Assurance, the SA personnel still need to report independently to project management, and SMA and are responsible for bringing up safety or risk issues. SA support can be provided by SA personnel embedded in each of the teams, by SA personnel serving as a general SME resource working with 2 to 4 teams or by team personnel performing SA activities.
4. Software Assurance processes used for Agile projects need to be flexible to react to the changing Agile environments on projects.
5. Recommended activities for Software Assurance personnel include:
   a. Attendance at Sprint Retrospectives, Sprint Reviews, and Scrum of Scrums
   b. Involvement in sprint planning, definition of "Done", and prioritization of backlogs

c. Help determine quality, reliability, and safety requirements
d. Assuring process compliance
e. Verification that completed sprint products meet the definition of "Done" determined by the team and project.
f. Monitoring of other important activities such as backlog metrics, velocity metrics, other metrics, and general team health and progress.
g. Audits of the tool usage to understand if and how the tools are being utilized and when and why they may need to change in the project.

6. Tailor software assurance processes to include how to assure requirements, design, and verification and validation (V&V) in an Agile environment. Requirements, design, and V&V may come in different forms (backlogs, stories, models, tasks, test scripts, etc.) and are iterative, building up to the completed Agile project. This may require a different approach or tailored processes, but basically the same types of analyses are needed.
7. Software assurance personnel need to be trained to understand the different methods of requirements expression (e.g. stories, tasks, models) being used and how they are transformed into design, code, and testing requirements.
8. Have all the SA, systems safety, and system reliability practitioners working on software and the system meet regularly to assure a coordinated safety, quality, and reliabiity approach for the software. (Periodically brief management on this.)
9. Use this report's recommendations and conduct any further research needed to create guidelines for performing SA on various size, criticality and mission type Agile projects. A new Agile tailoring section for the SA standard may also be needed.

Acknowledgements: First, the Team thanks the Office of Safety and Mission Assurance for supporting this effort. The Benchmark Team extends its thanks and appreciation to all organizations and participants. The openness and willingness to communicate exhibited by the interviewed organizations was tremendous and is of great value to NASA and to the software engineering community. In exchange for the honest and accurate information provided by the interviewed organizations, NASA has agreed not to disclose the names of the organizations or the personnel that participated.

1.0 Introduction

The Agile Benchmarking Team (ABT) was chartered by the Chief, Safety and Mission Assurance (SMA), in December, 2015, to better understand the best practices for

ensuring the quality, safety, and reliability of systems being developed with Agile processes. With the rapid increase in the use of Agile development processes, it is critical that NASA develop the best strategies for using Agile and for incorporating Software Assurance (SA) into the processes. A wide range of benefits is expected to result from this effort.

To assist in accomplishing these goals, NASA needed to take the best of NASA and industry practices and make them available to those who will be doing Agile software development work and Software Assurance on NASA projects.

To this end, the SMA Chief established a benchmarking team to perform a study to evaluate the best practices for providing software assurance currently being applied across the industry and develop a set of recommendations to ensure that NASA maintains the safety of its facilities, assets, personnel, and the public. The effort should also support advances in software development technology. The team has developed findings and recommendations that provide needed guidance in assuring the safety and reliability of critical software while using Agile software development methods. In addition, this report provides proposed additions and/or modifications to contractual, procedural, and product requirements.

This activity is very important to NASA's mission and aims to provide valuable insight into the best practices of industry and government. Hopefully, this study will lead to better use of Agile and Safety and Mission Assurance support for our changing software technology and development practices. Further work needs to be done, but this benchmarking effort provides a guide for focusing future work.

For a quick look at a basic Agile framework many of our interviewees used, please refer to Appendix I.

2.0 Agile Benchmarking Team

A letter from the Chief, NASA SMA, announcing the creation of the ABT and a charter containing the responsibilities of the team are attached as Appendix A in this document. The details of the team organization, membership, responsibilities, and the procedures used to satisfy the charter objectives are discussed in the following sections.

The ABT consisted of the following members:

Team Lead:

Martha Wetherholt
NASA Headquarters/NASA Software Assurance Manager & Technical Fellow

Team Members:

Glenn S. Kinney, NASA IV&V Facility, S&MA Office
Laura A. Maynard-Nelson, NASA GRC, Engineering Directorate, LSS0
Rhonda S. Fitz, NASA IV&V Facility, S&MA Office, TASC
Melvin R. Rother, NASA IV&V Facility, S&MA Office, TASC
Trey M. Duckworth, NASA IV&V Facility, S&MA Office
Sally Godfrey, GSFC Emeritus, TASC/Engility Corporation

2.1 Team Tasks and Objectives

A summary of the team's main tasks and objectives include the following:

- Research current literature on the Agile methodology in preparation for the benchmarking and gather and distill knowledge from recognized experts.
- Benchmark approaches and best practices to assuring the quality, safety, and reliability of software developed by Agile processes including the advantages and possible risks.
- Benchmark the methods of Safety, Reliability, and Quality Assurance (SR&QA) used by NASA, other government agencies, and industry and develop strategies for their incorporation into the Agile Software development processes. These strategies may include some or all of the following:

    - Assessment and evaluation of risks associated with various Agile approaches
    - Level of SR&QA oversight provided
    - Software Engineering and Software Assurance participation in major milestone reviews or other methods to gage progress on software development
    - Process for verification and validation of requirements
    - Risk process for Agile development including how risks are recorded, tracked, mitigated, and accepted
    - Hazard analyses processes and how the software and system hazard processes work together
    - Software reliability methods used to assure that the software will provide the needed fault and failure tolerance

- Consider different Agile approaches. Provide input to guidance and, if warranted, recommend changes to standards and policies that help build safety, quality, and reliability into new technology development without impeding progress.
- Identify possible opportunity areas for future software assurance research including tools, training, development, and process improvement.

2.2 Team Products

The Benchmarking Team was tasked to generate this report that contains the following types of information based on the results obtained during the benchmarking activities:

- Findings of the benchmarking studies and highlights of the current process,
- Propose further studies into areas such as: anticipated future process, and gaps in current NASA requirements, resources, and capabilities.
- Recommendations for SR&QA approaches to meet anticipated needs including:
  - Minimum SMA policy and requirements
  - Various insight and oversight stratagems
  - Possible variations based on type of Agile methods
  - Possible NASA SMA products, processes, and services needed to support safety reliability, and quality and testing, verification, and validation of Agile SW processes
  - How the technical authority waiver process will work

- Recommendations for OSMA actions to be taken to ensure adequate SR&QA acquisition and implementation when contractors are used.
- Suggested ways to work with and support various types of efforts, providing Software Assurance and software development management with methods for, or on the services to reach risk-informed design decisions in a quick and timely manner.
- Suggested approaches to build sufficient safety, quality, and reliability into Software Agile development efforts and products as part of the requirements, design, and general approach.
- Recommendations for future OSMA research and technology development.

3.0 Benchmarking Approach

To satisfy the intent of the ABT charter, the team researched available resources related to Agile methodologies to obtain a background for the benchmarking activities. This background allowed the team to prepare for and conduct interviews that would result in obtaining meaningful information and allow the team to perform significant analysis of that information. After doing the research, the ABT developed a questionnaire for use in guiding the discussions during the interviews. As suggested by the team charter, the ABT developed a list of candidate organizations that had experience with various Agile methodologies on a wide range of projects. The following sections present further details of team activities performed in obtaining appropriate data for analysis by the team.

3.1 Research Agile-Related Information

Agile-related information was researched to develop the background on Agile development methodologies so the team could develop a meaningful questionnaire for use during interviews and the background to do a meaningful analysis of the information obtained during the interviews. A bibliography of the material that the team used to obtain the background is contained in Appendix D.

In addition to the information contained in the literature mentioned in Appendix D, and the team members' own Agile and development experience, the team participated in

training, seminars, and discussions with personnel at the NASA Independent Verification and Validation (IV&V) Facility. IV&V provided their perspective on issues they have faced when performing IV&V on Agile developed systems. A summary of the takeaways from these discussions is contained in Appendix E. All this research and preliminary interviews, and discussions gave the benchmarking team insight on the Agile methodologies. Especially helpful were the insights and difficulties provided by the IV&V practicioners currently involved with NASA Agile projects.

3.2 Develop Benchmarking Questionnaire

The ABT generated a questionnaire for use during the benchmarking interviews that was designed to gather information that could be analyzed to derive the information the team needed to meet the objectives outlined in their charter. A copy of the detailed questionnaire is contained in Appendix F. The focus of the questionnaire was to obtain information that gave insight on the variations in Agile methodologies and Software Assurance processes that were used by organizations supporting the development of software systems for projects of various sizes and complexities. This focus allowed the ABT to gather data that would support the team's main objective of assessing the adequacy of the current NASA Standards and Processes for performing software assurance on software being developed by Agile methodologies.

The basis for the questionnaire was an integration of the results of the research done on Agile methodologies and the current NASA requirements, standards and policies related to performing software assurance outlined in the following documents:

NPR 7150.2A - NASA Software Engineering Requirements

NASA-STD-8719.13B – NASA Safety Standard

NASA-STD-8739.8 – Software Assurance Standard

While a copy of the detailed questionnaire is contained in Appendix F, the following is a list of the main topics discussed during the interviews:

1. Organization's software development background/perspective for interview
2. Organization's amount and types of software produced
3. Details of typical software system discussed in interview
   a. Agile approach used
   b. Factors influencing approach used
   c. Development team structure
   d. Development team processes
   e. Approach used compared to Agile Manifesto tenets
4. Software Quality Assurance involvement
5. Software Safety involvement

6. Software Reliability involvement
7. Verification and validation involvement
8. Independent verification and validation involvement
9. Customer involvement
10. Cyber Security concerns addressed
11. Results/Conclusions
    a. Value added due to use of Agile approach
    b. Challenges due to use of Agile approach
    c. Lessons learned related to performing SQA functions on Agile system

3.3 Benchmarking Interviews

Once the Agile methodology research and benchmarking questionnaire development were completed, the ABT developed a potential list of organizations to be interviewed that would constitute the expected range of Agile methodologies and project complexities desired by the team. The organizations were then contacted and a list of organizations that agreed to be interviewed was assembled. Interviews were conducted and data collected by the ABT. The details of these activities are outlined in the following sections. The interviewees were promised anonymity for themselves and their organization as well as a copy of the final report.

3.3.1 Interviewee Backgrounds

After developing a list of potential organization candidates that could supply the variations in organizations desired by the ABT, the ABT determined which organizations were interested in participating in the benchmarking exercise. The ABT then issued formal invitations along with introductory letters, team charter, and a copy of the questionnaire to the participating organizations.

The final interviewee list consisted of 21 organizations that represented a wide variety of organizational roles and perspectives related to their involvement with and/or utilization of the Agile methodologies. Six of the interviewees were from internal NASA organizations, twelve were external industry organizations involved in a wide range of commerce activities, one was a long time consultant involved in a wide range of projects, one represented a university, and one represented a government agency outside of NASA. The team felt that the backgrounds of the 21 organizations interviewed represented a good cross-section of perspectives on and roles in implementing software systems by the Agile methodologies.

The wide range of organizational backgrounds and perspectives are illustrated by the following items:

- Functional roles represented by the organizations included project management with varying degrees of oversight and insight, systems engineering, complete range of software system development activities, software assurance including quality, safety, and reliability, independent verification and validation, and Agile methodologies expert consultants.

- The sizes of the systems represented ranged from a few hundred lines of code in a web-based application to millions of lines of code in a complex, safety critical system.
- Efforts in developing the systems ranged from one co-located sprint team completing the system in one week to dozens of widely distributed sprint teams working in parallel for several years.
- Requirements complexity ranged from software only systems with no external interfaces to systems involving several hardware systems and subsystems interacting with each other and the systems requirements and associated interfaces all evolving at the same time.

3.3.2 Interviews Conducted

Of the twenty-one interviews that were conducted, sixteen were accomplished by face-to-face interviews at the organization's location by one to three of the ABT members. The remaining five were conducted by telecoms. All of the ABT members could participate in all of the interviews since all were available via telecom. The interviews generally lasted approximately two hours and were informal in nature. The questionnaire was used to structure the interviews and guide the discussions. However, at any time during the discussions, the interviewees were encouraged to provide additional information that was pertinent to the desired information the team was seeking.

3.3.3 Data Collection and Processing

Available ABT members recorded the organization's pertinent responses during the interviews. The recorded information was then integrated and transcribed into a composite response on the questionnaire form and reviewed by the team. As part of the transcription process, the information was intentionally sanitized such that the identity of the organizations providing the responses was not available. The composite responses were then entered into a specially created and secure database at the NASA IV&V facility. The database has a filter capability that allows the team to then extract data in ways that support multiple analyses and explorations of the data. Several view points from the database are in the appendices.

4.0 Benchmarking Data Analysis

The ABT members analyzed the data that was collected during the interviews to gain a better understanding of current Agile methods being utilized by software practitioners and to determine the best practices for ensuring the quality, safety, and reliability of systems being developed with those processes. This section introduces a topic area, discusses the findings from research and interviews on that topic, then discusses the ABT's thoughts and take-a-ways, and finally provides a list of recommendations on that topic.

4.1 Overview of Interviewed Organizations and their Agile Software Processes

Background information of the organizations interviewed and their Agile software development approaches and processes are summarized in the Organizations' Backgrounds and Agile Software Processes Table in Appendix G. The background information illustrates the broad range of perspectives on software systems developed by Agile processes.

The focus of the various organizations interviewed ranged from developing a web-based application with a small team in less than a week to a group of teams totaling several hundred people working on huge, complex systems over several years. Results from the benchmarking effort show that the smaller projects/organizations usually followed the Agile Manifesto more closely than the larger more complex projects/organizations.

Most of the larger project/organizations maintained some of the "rigor" associated with the waterfall process. Primarily, this was the use of the traditional program management/system engineering approaches, due to the inherent size, complexity, and integration challenges typically experienced for large scale projects. Sometimes called an Agile-Fall approach, this combined methodology has been recommended for larger, critical or complex projects by Barry Boehm and Richard Turner in their book, *Balancing Agility and Discipline: A Guide for the Perplexed* [1], as well as by the NESC study, *TI-14-00943 Alternative Software Programming For Human Space Flight*[2], led by Michael Aguilar, as well as several others.

In a few instances, the entire organization, particularly one organization, follows an Agile or continuous improvement approach from its inception. Another large organization completely restructured to start all their new software development projects with fully Agile processes from the top down. Thus, there were two larger organizations, truly following and utilizing a complete Agile approach for development and management. An important thing to note is that one organization is relatively new and the other is a larger long-time industry leader. The second industry still has some upper management struggles with the Agile approach, while the first one is Agile all the way through the entire organization. There was one federal Agency that also completely switiched to Agile after some trial projects, however we were not able to interview them in the time alloted. Most of their projects were web-based tools or applictions.

Thus, the organizations that were part of this benchmarking effort were both diverse and in differing stages of adaption to Agile. Some organizations were still trying Agile

[1] Boehm, Barry and Turner, Richard, Balancing Agility and Discipline: A Guide for the Perplexed, Addison and Wesley, Boston, August, 2003.
[2] Aguilar, Michael, TI-14-00943, Alternative Software Programming for Human Space Flight. October21,2014 (Available as a NASA Engineering and Safety Center Technical Report)

on only some of their projects while others had embraced it fully. Some new organizations and some older, some small, and some large organizations, each in their own way were trying to get the benefits of using Agile development.

Most of the organizations interviewed transitioned to Agile at various times or ways but the common reason was centered around the promise of added benefits and savings relative to cost, schedule and performance. The promises of added benefits and savings relative to cost, schedule and performance with Agile use are very tempting, but they are often more difficult to achieve than expected at first glance. Smaller projects closely following the Agile principles seem to do reasonably well with achieving the promised benefits. Larger projects often need to do more tailoring to improve communication across mulitple teams of developers and they may also be required to conform to requirements for additional documentation to meet outside requirements, thus reducing some of the potential for cost savings. Larger projects, particularly those with longer product lifetimes and/or safety critical concerns, may find that the minimal documentation of a pure Agile implementation hinders the analyses of safety critical factors and makes long-term mantenance difficult and more expensive.

Generally,for larger projects, expected cost savings and reduced time to develop were not met, especially early in the Agile learning process. However, when customer requirements were not well established upfront, or prototyping was needed to help discover what was really needed or posible, Agile has shown itself to be cost effective over time as long as the customer and users are heavily involved and the projects are not too large or complex. It must be stated that most experienced organizations did claim that eventually they did have savings in cost and schedule, and felt they had heightened quality and customer relations. In addition, most of the experienced organizations felt that the benefits coming from enthusiastic, self-determined teams led to more inventive, higher quality work products and a better work environment.

The transition to Agile for several organizations did not occur without challenges and lessons learned. In most cases, the organizations implemented the Agile development process on smaller projects and/or subsystems with minimal or no system integration requirements to gain expereince and knowledge. Additonally, training, team interaction, and team make-up were key elements in overcoming several process execution and proficiency challenges which will be discussed in detail in the following paragraphs.

The implementations of the Agile processes by the organizations interviewed covered a broad range. The size of the projects ranged from one sprint team to over forty sprint teams and the length of the sprints ranged from one week to six weeks. In some cases, the interaction between sprint teams was negligible and in some cases there were several levels of Scrum of Scrum meetings that helped coordinate multiple teams. Some of the projects had sprint teams that were sychronized and some worked asychronously. There were numerous variations of the Agile methodologies being used. In some cases, the sprint teams varied their processes as the team desired in order to accomplish their tasks. With the need for more integration between teams, sprint times

were usually set to one length for all sprint teams by either the teams jointly or by a manager for the project

		Organizations									
		N1	N2	N3	N4	N5	I1	I3	I5	I9	I12
Size of Projects	Large	X	X	X	X	X	X	X	X	X	X
Tailoring of Agile	Scrum tailored with Waterfall			X	X	X	X	X	X	X	
	Tailored Agile	X	X								
	Pure Agile (Agile, Scrum, Safe)										X
	Minimal Agile										

Size of Projects and Tailoring of Agile

Table 4.1-1 Large Projects & Tailoring of Agile

		Organizations											
		I2	Fed1	C1	N6	I4	I5	I6	I7	I8	I10	I11	U1
Size of Projects	Medium	X	X	X									
	Small				X	X	X	X	X	X	X	X	X
Tailoring of Agile	Scrum tailored with Waterfall				X						X		
	Tailored Agile	X		X				X					
	Pure Agile (Agile, Scrum, Safe)					X	X		X	X		X	X
	Minimal Agile		X										

Size of Projects and Tailoring of Agile

Table 4.1-2 Medium to Small Projects & Tailoring of Agile

In addition to the software development perspective, some of the interviewed organizations were focused on providing software assurance functions on a wide range of systems. Other organizations interviewed consisted of coaches, consultants, or customers of the developed systems. Some of the organizations functioned as both developers and coaches to other teams within another organization. The organizations ranged from those involved with man-rated safety critical systems to other organizations which just tried to keep the software defects to a reasonable level during use were interviewed. The complexity of the systems ranged from simple software applications for the web; single processor systems, with few interfaces with other hardware or software systems; to a system involving many subsystems with complex interfaces. Many of the organizations had a great deal of experience with the use of Agile processes and others were just starting involvement with Agile. One of the organizations with a very large complex project was just getting started, about 1 year into it, and struggling to figure out how best to make Agile work for them.

The many specific Agile development methods began to emerge in early 2000 as a change in thinking, as many programmers searched for more efficient, less burdensome

software development methods. In February, 2001 a group of software developers met in Snobird, Utah, to discuss lightweight methods. This meeting resulted in the documentation of a set of Agile principles and "The Agile Manifesto". These are now widely used to support the Agile values including teamwork, process adaptability, continuous early deliveries of working software, customer collaboration, and continuous improvement. See Table 4.1-3, column 1 for the Agile Manifesto and Table 4.1-4 fColumn 2 for the Agile principles.

The Agile methodologies and processes, encountered or practiced by the interviewees, adherence to the tenets of the Agile Manifesto purpose and principles were also mixed as noted in the following:

Agile Manifesto Purpose	General Findings
Individuals and Interactions over Processes and Tools	Even though most of the organizations emphasized the collaboration and interactions of individuals in the processes, the use of disciplined processes and complete tool suites were equally emphasized.
Working Software over Comprehensive Documentation	Working software was emphasized for small, non-safety critical projects; whereas, comprehensive documentation that had strict configuration management was emphasized for more complex and safety critical systems.
Customer Collaboration over Contract Negotiation	Again, the ABT received a mixed response. Even though close customer/developer collaboration was noted by most of the Agile practitioners, the need for details in contracts to clearly define processes and products required of the developer was emphasized. Many SW developers did state that the old contract language made it hard to meet the expectations of the customers and new contract approaches were needed for Agile work and deliverables. Different deliverables, milestones and metrics are needed for true Agile development.
Responding to Change over Following a Plan	It was acknowledged that Agile methodologies were extremely well suited to reacting to changes in requirements as the system was being developed. However, the need to have a detailed plan upfront was more strongly noted as being extremely beneficial to project success and, in fact, necessary in most cases. In fact, most organizations stressed the need for planning (everything from the overall project to the sprint level) especially for more complex systems.
Secondary Tenant Expressed by Some Organizations	

Agile Manifesto Purpose	General Findings
Production Sprints over Building Block Sprints	Some companies ran their sprints in a manner that had production ready software at the end of each sprint. Others just used the sprints as building blocks of the larger system and only some of the sprints completed with production ready software. For mid-sized to larger projects, it was more common to have one to several building block sprints and then transition to production sprints. For very large projects, many more sets of parallel building block sprints would combine into a series of incremental "releases" which would eventually lead to a production software product.

TABLE 4.1-3 Agile Manifesto Purpose Adherence

	Agile Principles	ABT Discussion
1	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	For some large, critical projects, early, partial releases can not be used. Parts or single functions might be able to be delivered for initial assessment. This approach works well for web-applications, prototypes, and interface software or when a project's multiple functions can be broken down and worked independently
2	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	With safety critical systems, late changes are not welcome as the software cannot go through the rigor needed to assure they will be complete, thouroughly tested, and have safety and reliability analyses to back them up.
3	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale.	(see 1. above)
4	Business people and developers work together daily throughout the project.	The idea is that everyone involved in a project all work together, the SW engineers, marketing, management, acquisition, IT, SA, and whomever is needed are all part of the team in one way or another. The Scrum Master, or PO, if they exist, make sure the customer is represented if the customer is not part of the team at least at the sprint review level.
5	Build projects around motivated individuals, give them the environment and support they need and trust them to get the job done.	Trust in the teams to do the work and do it well is a key element and a hard one for some management to adjust to. Again, with safety critical projects, it is required to "trust but varify". It was also noted that

	Agile Principles	ABT Discussion
		the teams need to have some flexibility in their membership to assure all work well together. In large part a lot of the motivation comes from being self empowered and taking pride in what the team develops, not the individual. Many interviewees mentioned the need for teams to determine their own tools, pace, and deliverables, again, for safety critical or even some large multi-team projects, this may lead to interface issues and other problems. Still, to the extent possible, teams should be given the guidance and tools needed and trusted to do the work.
6	The most efficient and effective method of conveying information with and within a development team is face-to-face conversation.	This is what the daily team meetings (often called Scrums) are for as well as all the team meetings. Also, the preference for co-location, though some teams did very well working across multiple locations and time zones by using Skype and other telecommunication tools.
7	Working software is the primary measure of progress.	This gets down to what is delivered. Basically, each Sprint delivers some portion, some feature or features of the software in a working state. It will not be the finished state, and each subsequent delivery builds on the old delivery with added functionality. The question is, when and how are things like safety features and reliability features delivered? As part of each delivery in some cases. As it's own project with it's own teams in other cases and needs to then be integrated iteratively as well.
8	Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.	This depends in large part on the teams' ability to break down the functions for each sprint into stories and tasks that are achieveable within the sprint. And for testing to be integral, timely and complete. In truth, team burn out can be a problem when getting started until a good team pace can be determined and met.
9	Continuous attention to technical excellence and good design enhances agility.	Freed up from the more administrative details, the developers and testers can consentrate on helping each other come up with the best design and best

	Agile Principles	ABT Discussion
		processes. And have the freedom to improve as needed.
10	Simplicity—the art of maximizing the amount of work not done—is essential.	While some interpret this as freedon to not create any plans or documentation, as you will see below, that is not the intention of those that created the Manifesto.
11	The best architectures, requirements and designs emerge from self-organizing teams.	This is held to be true for work at the team level. It does not however, always work when a project is very large and an overarching architecture must be created and maintained. Here, it was felt this can be met by giving teams portions of the larger design or functionality to meet but maintain a high level control on the bigger project.
12	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Most of the organizations interviewed agreed that the retrospecitive meetings were very important and some warned that it should never be missed.

TABLE 4.1-4 Agile Manifesto Principles

In the words of one of the Agile Manifesto[3] authors, Jim Highsmith, of the Agile Alliance: "The Agile movement is not anti-methodology, in fact, many of us want to restore credibility to the word methodology. We want to restore a balance. We embrace modeling, but not in order to file some diagram in a dusty corporate repository. We embrace documentation, but not hundreds of pages of never-maintained and rarely used tomes. We plan, but recognize the limits of planning in a turbulent environment. Those who would brand proponents of XP or SCRUM or any of the other Agile Methodologies as "hackers" are ignorant of both the methodologies and the original definition of the term hacker." [4] Reference Merriam – Webster on line dictionary: Definition 2: Hacker - an expert at programming and solving problems with a computer.

4.1.1 Summary of interview information

a. Team work and constant improvement as a team is essential to Agile.
b. Agile can work for many kinds of organizations and project sizes and complexity.

[3] Beck, Kent, Grenning, James, Martin, Robert, and others, 2001, A Manifesto for Agile Software Development, http://www.agilemanifesto.org, downloaded May 2016.
[4] Fowler, Martin and Highsmith, Jim, The Agile Manifesto, in Software Development magazine, October, 4, 2010

c. Adjustments usually need to be made to the basic Agile processes when there are multiple teams working on the same project.
d. Cost savings were more readily achieved in small to mid-sized projects with only one to three teams. Larger projects with multiple teams and complex, even safety critical, software products had yet to realize any substantial cost or schedule benefits.
e. The vast majority of those interviewed felt there were improvements in quality and-innovation due to team spirit and the ability to self-organize within each team.
f. While many organizations began their Agile experience with one or two projects and a ground up approach, top down support was eventually needed for full adoption of Agile.
g. Many of the organizations with larger projects had separate, more waterfall-like teams doing special functions such as architecture, fault management, integration testing, etc. Others maintained an Agile approach, but with more structures and processes for these functions.
h. The Agile Manifesto, or the type of Agile process that is chosen as a base process (Kanban, Scrum, Crystal, etc.), is meant as a guideline or goals. The important point is to do what is needed to accomplish the task at hand and get the best product possible to the customer. That is, within the guidelines of Agile, if certain documentation is useful or required by the customer, do it – just keep it to a minimum and do only what is needed. If 4 week sprints work better than 2-week sprints, do it. If testing once per week is better for the integrated team, then do it. If the team finds a development tool helps them capture stories and activities better than sticky notes, then get and use the tools that are needed. However, the basics need to be observed, such as: 1) daily meetings to discuss what everyone has accomplished, any obstacles, and work to be accomplished that day; 2) maintaining the backlog(s); 3) consistent sprints; 4) review meetings; 5) retrospective meetings; 6) team credit, not individual credit, for work accomplished.
i. Scrum was the most common type of Agile development process followed among those interviewed, primarily due to its straightforward, easy to learn and use processes. It was often modified to some extent.

4.1.2 ABT's Observations

a. Improvements to the team process and products are the responsibility of the team. Being considered a part of an Agile team is important to working with that team, especially if making suggestions for improvement
b. Modifying a chosen Agile methodology, even adding in waterfall type reviews, integration, and deliverables can be part of the Agile process. The Agile

Manifesto suggests purposes and principles, but in practice it is up to the team or organization to determine the best ways of meeting those principles.

c. Definitions of "Done" for a sprint, for a release and for the final product is highly recommended, and should describe the expected task completion status, quality and testing level of all products. Not all organizations had the same level of definition specifics. Some organizations had very strict definitions of "Done" at each level equivalent to formal organizational work instructions, while others were less formal and kept the definitions as part of the team norms.
d. Not all organizations approached metrics and quality the same way. Some had very strict definitions of "done" which they followed to assure quality and some had personnel on each team dedicated to measuring and reporting on quality while others relied on team members to check each other or to test until they considered it done, or even use a "fail quick and fix quick" method until it is right and the customer accepts it.
e. Most teams had some measure of backlog tracking and team velocity although most of the interview information was antidotal. The research material did recommend the need for some metrics especially to help a team understand and improve its processes and quality. Backlog measurements and tracking were the most common metrics recommended, followed by with team velocity. Velocity is the speed of going from requirements at the start of a sprint to the end of sprint with a finished, working software product meeting the definition of "Done."

4.1.3 ABT's Recommendations

a. Since NASA has many projects of various sizes and criticality, the "one size fits all" approach will not work for Agile any more than it works for a Waterfall type approach.
b. NASA needs a policy or at least guidance for determining when Agile is appropriate and how Agile developments should be performed at varying levels of complexity, criticality, and size.
c. Provide guidance on the NASA expectation for the definition of "Done" for a sprint, Interim Release, and for a final product. This can be used for internal projects directly and as a comparison when work is done outside of NASA.
d. Since the value of Agile comes from healthy, creative, self determined teams, promote an atmosphere where teams can work together and be as self regulating and autonamous as possible while still meeting the guidelines.

4.2 Software Assurance Discipline

Many of the organizations we interviewed did not have SW Assurance, per se', but had others who filled that role to some degree and most were embedded in the team or were responsible for 2 to 4 teams at most. In one case, each of their many teams, had 2 dedicated embedded team members who did all the quality control, engineering, safety, and assurance work, but they did not call them SA. In other cases, team coaches, consultants or the Scrum Master performed the work. Many of the organizations with smaller development efforts (6-12 months and only one or two teams per project) just counted on their development approach (e.g. pair programming, walkthroughs, definition of "done", entrance and exit criteria, etc.), testing, and on customer feedback. In NASA organizations, where SA has long been a partner in development, SA as such has been kept, but is spread even thinner in some cases than on traditional waterfall projects. The idea that an outside person or group, software assurance, would need to occasionally interfere with the team's work and judge them is drastically opposed to the way Agile teams work and thus some negativity can be associated with even the idea of software assurance/software quality. The team is supposed to determine, monitor and correct its own quality issues. Thus, while quality, safety and reliability may be on-going, and members of the team may be assigned to track the team's adherence to process, configuration management, and the quality of their deliverables and testing, outside of the Aerospace industry, a person designated as "software quality" is seldom observed. In keeping with NASA policies, only the larger projects, typically with safety critical software, that are being developed by NASA or a NASA contractor had any IV&V.

Some assurance responsibilities may call for modified techniques or tools. For example, assessments of progress and completeness in Agile projects may be accomplished by tracking and analyzing the backlogs. Similarly, assurance of the Agile processes in place will probably call for modifications in the current process checklists so the appropriate Agile process steps and products can be checked. Activities like tracing requirements to tests will require an understanding of user stories and features as the requirements.

There are opportunities for increased assurance with Agile, due to Agile's iterative nature with a focus on small sections of the requirements at a time and continuous integration and testing. However, these same Agile features can also cause issues for assurance. Certainly, more assurance can be done in the requirements and V&V areas, but with the larger projects, that means many more resources are needed to cover all the various teams and activities going on. It also means that the entire integration is much more complex and requires more attention and coordination. Assurance activities like safety and reliability can also be much more labor intensive on

larger projects if the overarching system architecture is not done and the early design is not detailed enough to do the early analyses. Analyses may need to be repeated late in development if changes affecting critical features are accepted late in development. Any testing on changes in safety critical areas would need to be retested and could potentially cause a delay in schedules.

Many of the tools currently used for software assurance will still be useful for assurance on Agile projects. Examples of those tools would be static code analyzers, software requirement verifiers, and test coverage tools. Some of the tools may have to be applied more frequently for assurance since the products of an Agile development are built and tested incrementally through the sprint process. Other tools like the software requirement verifiers may need modification to work with Agile user stories and product features.

The majority of the reliability and safety analyses, such as FMEA/CIL, FTAs, etc., would still be conducted much the same in an Agile project as in a traditional Waterfall type of project. For these analyses, an overarching architecture and a good design would need to be produced early. While these were often found in the larger, safety-critical projects, that was not always the case, making analysis difficult. In addition, with Agile's ability to handle changes much later in development, some of the analysis work may need to be revisited if and when the design changes.

4.2.1 Summary of Interview and Research Information

a. SA work or areas of concern are much the same for Agile as it is for waterfall.
b. Generally, within Agile, all the same quality and assurance processes are still being done, despite not being labeled as performed by SA.
c. Agile difference: the team is more collaborative (between developers and SA) than auditor-based. "We're on the same team" needs to be the feeling.
d. Sprint Retrospective meetings are where the team discusses what is working and what is not, and what could be improved in the Agile process. Suggested changes are determined, and help is assigned when necessary. This is an embedded team meeting with the common goal of constant improvement. The team's honesty at the scrums and retrospective in bringing forth problems is key to addressing some of the SA type roles.
e. Quality and checks and balances are to be built into the Agile process and in each sprint, they are to be examined if there is a perceived problem.
f. From the small sample we had, SA personnel liked working within the Agile (or hybrid) framework especially when they were integral members of the Agile team.
g. Agile development does not eliminate any tasks or necessary artifacts to conduct SA or IV&V, but may impact what is deemed acceptable. The artifacts may not be as formal and may be kept within the tools such as JIRA until such time as the information is needed, so a document, per se, is not needed.
h. For large systems, system architect meetings with a safety perspective can be held weekly or bi-weekly at some organizations. This information then can be transmitted

to the effected sprint teams in a couple of ways. It can be treated as a Scum of Scrum meeting in which some member of each team is present, or it can be the Project Owner who ensures the information gets on the appropriate team's backlogs. When SA or system safety were on the Scrum of Scrums, that is, safety analyses could be assessed for possible design changes to control, reduce or mitigate hazards. Also, some safety and reliability features will be the requirements that are on the backlogs of the sprint team and the project. Verification of the implementation of safety, quality and reliability features must be verified and tested.

i. Often Agile SA personnel are over-booked and their time is stretched thin over several different projects, sprints teams, or activities, just as in waterfall type development efforts.
j. Safety (reliability, SQA) has same tie in points as it did prior to the Agile transition.
k. The PM also remains outside of Scrum teams, which are made up of, ideally, a Product Owner, Scrum Master, embedded testers, developers, build managers, and SMEs. Some teams have these roles overlapping in one or more of the team members. Agile has a plan, design, code, test, integrate, test integration model that is accomplished on a few features within the limits of sprint.
l. Following the Agile framework is thought to build reliability in early and continuously. However, many organizations, when asked about reliability, associated it with rigorous testing if they addressed it at all. The larger, safety critical NASA projects still performed Failure Mode and Effects Analyses and Fault Tree Analyses to assess weaknesses and determine where best to build in fault management.
m. One project is trying to address fault management by having it addressed by a cross project sprint team,
n. Another aspect of reliability is worked in all along the way, since from the first sprint in Agile, the goal is to always have something working. Problems are found, worked immediately until resolved, and the team doesn't move on to next thing until the current sprint requirements are met.

		Organizations									
		N1	N2	N3	N4	N5	I1	I3	I5	I9	I12
Size of Projects	Large	X	X	X	X	X	X	X	X	X	X
	Safety Critical	X		X	X	X	X		X	X	X
Type of SA, Training of SA, Adequacy of SA	Independent SA			X	X	X	X			X	X
	Independent SA by Customer	X									
	SA done by Agile team								X*		
	No SA										
	SA embedded in team							X			
	SA trained with team										
	SA trained on Agile									X	
	team member performing SA trained in SA										
	Mentioned inadequate resources			X		X	X			X	

Table 4.2.1-1 Large Agile Team SW Assurance

Table notes: * Only QA performed

		Organizations										
		I2	Fed1	C1	N6	I4	I5	I6	I7	I8	I10	I11	U1
Size of Projects	Medium	X	X	X									
	Small				X	X	X	X	X	X	X	X	X
	Safety Critical		X*	X*		X*	X						X*
Type of SA, Training of SA, Adequacy of SA	Independent SA		X							X	X	X***	
	Independent SA by Customer												
	SA done by Agile team	X				X*	X*		X			X	
	No SA				X								
	SA embedded in team	X						X**		X			
	SA trained with team												
	SA trained on Agile										X		
	team member performing SA trained in SA												
	Mentioned inadequate resources		X										

Table 4.2.1-2 Small & Medium Agile Team SW Assurance

Table notes: * Few projects safety critical; * (bottom section of chart) QA Only; ** No independent reporting chain, but not done by developers; * Independent SA only if customer pays for it; Fed1, C1, I7 didn't develop their own software**

4.2.2 ABT's Observations

a. As noted in the responses above, the SA tasks are much the same for Agile systems as for waterfall systems.
b. Agile begs a "New Model" for SA and IV&V. However, there is much that remains the same and an independent look with a different perspective and reporting still has value.
    1. Generally, all of the same development processes are being done even though they may be labeled as Agile and done repetitively and rapidly on small increments that build to a final whole.
    2. The timing of the looks and the particular items reviewed changes with Agile, however, for critical high value projects, a second or even a third look at the quality of the products and the processes is still necessary and valuable. Much of SA observation on general sprint team health and progress can take place in conversations or at the daily scrums where there are three things every member of the team is required to report: 1) "What got accomplished yesterday?" 2) "What barriers to your work are you having?" and 3)"What are you going to be working on today?"
    3. When there are "barriers to progress", these can be opportunities to provide quality support and input, or, at the very least, to track issues and see if they continue or get resolved in a timely fashion.
    4. The Agile difference is that the team is more collaborative (between developers and SA) than auditor-based. In truth, SA should be conducted this way in the traditional waterfall projects, as well.
    5. Often the SA personnel will need to adjust their processes during a project based on changes to the processes being used by the development organization.
    6. When Agile sprint teams are under schedule pressure, they often cut back on the documentation of sprint results and issues that affect the ability of SA personnel to accomplish their assurance tasks. When SA is incorporated as part of the team, they have better access to the information they need for assurance.
c. The importance of embedding the SA personnel within the sprint teams, to the extent possible, should be emphasized. SA should be incorporated as part of the sprint team, often acting as proxy for the users. Process adherence as well as testing is emphasized
d. SA resource considerations:
    1. The need to be in a more collaborative environment usually results in resource issues for SA, as there are insufficient resources to be a part of every sprint team.
    2. Often, when there is insufficient SA to be embedded in each sprint, then just the larger, more integrated meetings are attended, with some "audit" of the various sprint team meetings. This is not ideal as the tight knit team structure of Agile

can be disrupted by outside interruptions and slowed by the need to provide explanations of current activities.

e. SA roles within the Agile process:
    1. SA can perform many roles within the team. SA participation as a team member dedicated to SW quality and safety or performing another team role has been demonstrated to be useful.
    2. As a Scrum Master, the SA would:
        a. Ensure the smooth workings of the team, and
        b. Help the team follow and update their processes, acting as a guide to changes and an advocate for the value of certain processes.
        c. Work to resolve impediments internal to the team or ensure any team impediments are raised to the appropriate chain of command outside of the team.
    3. As a designated member of the team dedicated to safety and quality, the SA sprint team member would:
        a. Assure the workings of the team's process,
        b. Collect metrics (e.g., velocity – amount of "done" work over time, backlog progress, errors added to backlogs),
        c. Monitor and raise issues to team on progress,
        d. Assure testing completeness,
        e. Assure safety analyses results are addressed, and
        f. Work with the Product Owner on establishing risk levels.
    4. As a consultant and or coach for one or more teams, the SA would:
        a. Cover only 2 to 4 teams at a maximum to allow effective coverage when covering multiple teams,
        b. Ensure the sprint processes agreed to by the team were followed and any overarching processes were also observed,
        c. Collect assets and metrics received from each team,
        d. Participate in any Scrum of Scrums, sharing metrics, ideas for overall performance improvements, and identifying concerns with all the teams and helping resolve them.,
        e. Provide improvement measures from team to team, and helping them all improve,
        f. Concentrate on the safety aspects and assure that system and software safety issues placed on the team backlogs are getting worked as high priority items.

f. Safety support of Agile developed systems generally works almost the same as in a waterfall environment. It is usually done by the project's systems engineering organization and performs the usual analysis tasks such as hazard analysis. Hazards related to the software systems are brought into the sprint team planning

and placed on the project and sprint backlogs and tracked to resolution. Additional defects or hazards discovered may be brought up at the Scrums and then raised or addressed at the team level via planning and backlogs. Any effects on the software are addressed through the backlog process. Safety and Reliability methods are really not changed within the Agile framework, other than progressing in smaller, more incremental bites. The analysts are actively engaged in all elements of development alongside the project teams. Yet, the system view is still needed to reduce unneeded repetition and missing coverage.

g. Reliability support of Agile developed systems seems to be based on thorough safety analysis and software system testing. Some of the organizations involved with larger, more complex systems had engineering or assurance perform support analysis such as FMEA/CIL, FTA, etc. Also, there was a lot of emphasis on ensuring that all sprint team members felt responsible for the reliability of the system. Some organizations relied on metrics to provide a measure of expected reliability. Monitoring volatility of a project may be an insightful metric for SA. Providing assurance piecemeal, where the system appears stable, may give an illusion of confidence that is not applicable to the system as a whole.
h. Verification and validation:
    1. SA needs to consider its roles in the validation and verification process including testing and reviews. Since resources are limited and V&V and testing are such an integral part of the Agile daily process, SA needs to decide where it can add value and monitor progress to provide management with its independent assessment.
    2. Just a few of the organizations interviewed noted that independent verification and validation (IV&V) was performed on any of their systems. Those who did have some IV&V did not provide a lot of insight into how they were affected by the Agile methodology.
i. The team needs to keep in mind the shared goal of a quality product, with safety addressed at a system level, but integrated down to lower level subsystems and detailed requirements.
    1. Safety measures depend on a robust hazard analysis and clear traceability of documentation in support of a well-developed assurance case independent of process approach.
    2. For safety and reliability, conduct a hazard analysis, but “just enough”; be professional, but you can’t always predict the future, so don’t spend excessive time trying to anticipate every situation (in a non-safety critical environment).
    3. Software Safety has a role in the review of fault management algorithms, ensuring the loop is closed between them and the Hazard Analysis
    j. When SA is represented on teams and not independent, the quality of the product may be affected.
    k. Tying characteristics of Agile projects to potential SA and IV&V strategies may help accomplish a more thorough analysis, keeping open to adaptability in order to optimize the value that can be provided. Different tactics could be taken for

different projects and their development environments to bring forth recommended SA approaches.

l. Agile development places new stresses on SA and IV&V activities, since the responsibility to assure Class A missions is not altered, nor has tolerance for a system failure or mishap changed.
m. Providing assurance within an Agile framework is difficult because results aren't always captured formally or if captured, done after the fact. Time pressures often mean less testing, and reliability is often overlooked as a success criterion.
n. One major area of contention is the need for early analyses results to help form the design and the lack of documentation early in the Agile process that allows those analyses to take place. It can become a vicious circle for which some measures, e.g. contract specifications, need to be put in place to have needed artifacts up front delivered to safety, quality and reliability in order for them to do their analyses and get the results back to the sprint teams and any overarching design for impacts. If the project has adequate resources, they may have an overarching team handling this as well as safety, quality and reliability analysts on each sprint team that work with the team and the overarching SMA team.

4.2.3 ABT Recommendations:

a. Additions are needed to the current software assurance and safety standards to better address software assurance for Agile. Language changes and examples are needed. A section in the electronic SA guidance document needs to be written as well.
b. For larger projects, the need for SA and Engineering plans at the contractor and NASA level still exists, but they need to be written for the Agile-type development. SA involvement in the development activities at both the acquirer and provider level needs to be planned and addressed.
c. Reliability, safety, and security should all be included in User Stories. Performance specificity is important along with broad capabilities and constraints.
d. Test methods for User Story implementation must be included with the User Stories. For comprehensive testing, high-level requirements and inadequate success criteria need to be refined.
e. SA participation within the Agile project:
    1. SA should be integrated into the Agile teams and as such be part of the decisions made within sprints. This allows the SA personnel to have the necessary insight to understand the development process of that team.
    2. To the extent possible, have sufficient resources to embed SA personnel in each of the sprint teams. This is unlikely, since there are just not that many SA personnel and projects would prefer to hire a developer. In this case, the project may need to train developers to perform some of the SA roles. SA then works with those on the team to assure processes and products are working.

3. Having software assurance personnel seen as SME's on a project and providing the expertise on obtaining and maintaining quality for the sprint teams could help when SA personnel is either part of a sprint team or works with 2 to 4 teams. Then they are seen as part of the teams and a shared resource rather than outsiders slowing down the progress of the teams.

f. SA's basic role's in Agile:
   1. Keeping work to small batches; but still maintaining a systems perspective
   2. Keeping an eye on how the rank order of the backlog is being made and kept,
   3. Attending the demonstrations at the end of the iterations to see if the completed product is at the proper level of quality, the criteria for "done" is met, there are no known outstanding defects, the testing automation has been incorporated, and to receive the customer feedback,
   4. Maintaining velocity metrics and insight into task story point for capacity metrics. Task story point assignment metrics associated with relative people effort is preferable to measurements of time.
   5. Assure that the sprint and backlogs are prioritized with the highest risk and safety critical functions at the top of the backlog. This should be one of the things SA monitors for software system and sprint level backlogs, providing a systems view across the project.
   6. Assure any safety, reliability and quality analyses are performed on an ongoing basis in order to support the inputs to the design while also maintaining inputs from the systems overarching safety and reliability assessments. This also means working with the teams to assure needed artifacts are produced in a timely fashion so that ongoing analyses can take place and try to keep pace with each team.
   7. Even though Agile purports the advantage of being able to make late changes, for critical and high reliability systems, this cannot be allowed. NASA's caution for thorough testing and analyses of changes and their impacts to safety and reliability still need to be followed, thus changes late in the development cycle should be disallowed.

g. Agile activities benefiting from SA participation:
   1. SA needs to participate in sprint planning, examining new or changed requirements (as part of a Change Review process) and examining where specific requirements and features fit within the sprint planning.
   2. Have SA participate in the project definition of "Done" and help determine the team(s) quality, reliability, and safety requirements, which should be drawn from the current standards.
   3. For large systems, where there is an overarching system architecture, and system safety is discussed, SA should attend meetings to keep apprised of any changes, and their impact, either at the system level or at the team(s) level.

4. The Sprint Reviews and Retrospectives also require SA participation as changes for the next sprint and sprint processes can be decided at those meetings as well. For larger projects, SA should also be represented on things like integration teams, Scrum of Scrums, release preparation/planning meetings, verification planning, etc.

h. If SA can only be outside the sprint teams or is covering several teams:
   1. Participation at the Sprint Review meetings of each Team is important. Other important activities to audit are testing fidelity and completeness, backlog metrics, other metrics, and general team health and progress.
   2. However, with multiple simultaneous sprints, often the Sprint Review meetings and Backlog Review Meetings will all occur at about the same time so a single SA member could only cover one team per sprint.
   3. Sprint Reviews should not be scheduled all at the same time (at the end of the sprint) but should instead be held in a more incremental fashion, staggered in order to allow for better SA support.
   4. Participation in any Scrum of Scrums and release efforts should also be considered as minimum participation points. Review of teams' initial planning and procedures as well as auditing to observe progress or changes is recommended.

i. SA (and IV&V) could take several approaches or a combination:
   1. Assessment of product and process (in concert with development personnel on a team) of things like: technical practices, leadership, quality, and organizational culture in order to identify areas where the team needs coaching to get on track,
   2. Analysis of the metrics such as velocity, adherence to "done", metrics like backlog creep, test coverage, test completion, etc.,
   3. Help to establish a good definition of "done" that includes the quality factors,
   4. Coaching the team in the development/process with SQA embedded
   5. Reviewing baseline documents and plans, including the backlogs, stories, tasks and activities with the team or product and sprint backlog.
   6. Reviewing status for keeping on track following every sprint, and checking for completion or need for re-planning.

j. SA participation in V&V:
   1. SA personnel, in planning the best use of their limited resources, need to rank test set up, completeness and integration as high priorities.
   2. Taking the Agile perspective into account, SA needs to assess whether the verification testing performed by the project meets NASA SA standards. A review of the testing area and processes should be conducted either by SA or by an SME internally within the team or project.

k. Reporting channels for SA:
   1. Reporting channels for SA should begin at the lowest level, working initially directly with the development teams. As with traditional development, have SA report up to the SMA technical authority and

management chain, as needed, to resolve issues that cannot be addressed working with the sprint team.

2. Safety and mission assurance needs to be under separate management than software development so that they are a “protected” spokesperson for how well performance is progressing. Independent evaluators are needed for safety and quality.
3. Have all the SA, systems safety, and system reliability practitioners working on software and the system meet regularly to assure a coordinated safety, quality and reliabiity approach for the software.
4. Also, SA needs to meet with SMA mangement at least monthly. It helps the SA embedded in the teams remember that they have a separate reporting chain and while they are responsible to the teams and project, SA is still part of the SMA chain of responsibility and needs to maintain a certain independence.
5. The SA on different teams should meet and discuss progress and potential problems, sort of an SA Scrum of Scrums.

l. Fault Management (FM) should be built in on a continuous basis or at least addressed as critical backlog items. The problem is integrating a comprehensive FM across many teams and appropriately breaking down each sub-project’s sprint contribution.

m. An overarching design needs to be created and maintained as well as the tests needed for each release of integrated sprints. Metrics should be kept on number of open/closed problem incident reports, how long issues remain open, and other data used to project reliability. For Agile, these will be added to the critical backlog items.

n. Training SA for Agile:
   1. Formally train all SA personnel in the general Agile methodologies with the Agile development teams before the start of the project. Where possible, SA needs to participate in, and be trained in any changes the teams choose to make to the processes,
   2. The SA personnel should be trained to understand how the different forms of requirements in use by the Agile teams can be used in performing the SA functions and how they can be translated into design, code, and testing requirements.
   3. SA needs to be trained on the project tools for management, testing, configuration management, tracking, metrics , etc. This includes systems that will be required to access the project information.

4.3 Agile Teams

There was a consistent theme that team interaction and make-up played a critical role in the success of developing timely and quality software products. An overwhelming majority (95% plus) of the benchmarked organizations,referenced in the Organizations' Backgrounds and Agile Software Processes Table in Appendix G, implemented and used the SCRUM Agile development approach in some form. SCRUM is an iterative

planned development approach where each iteration is referred to as a sprint. A sprint is a predetermined period of time based in the scope of the development iteration. The time period can drive the scope of work or the scope of work can drive the time period. A majority of the organizations that had single independent teams each working on separate projects usually had the teams establish their own sprint length, changing it as needed to establish and maintain the team's best efforts. Larger projects with interacting teams usually wanted to establish a consistent rhythm among the entire SW development team. Based on the benchmarking results reflected in Appendix G, two to four week time periods are typically established and used for the entire project. Appendix I has the basic Scrum Sprint framework.

When a project is too large for one team, multiple team sprints can be in progress at the same time. Some organizations followed the Agile recommendation for self organizing team planning and allowed each team to determine their own sprint intervals. However, most organizations with multiple teams working on the same final product found that they needed to coordinate the teams and have uniform sprint intervals. One organization with multiple interacting teams was trying to maintain the Agile precept of allowing each team to determine their own sprint time, however, they admitted they were struggling to bring the software together and get a handle on the project status as a whole.

The size of the sprint teams varied from organization to organization, and even within organizations, depending on the planned products. The size range was approximately from 6-10 team members. Gathered from the interviewees and some of the research material, the factors listed below are important to consider when creating sprint teams:

a. Talented people who can function in many roles, such as coder, designer, tester, system engineer, etc. are desirable.
b. Team members should have good person-to-person skills and work well in a team environment. That is, they are good collaborators.
c. Team members who are invested in the overall good and success of the project and products as a whole and speak up when they need help or see problems.
d. Team members who are willing to keep "pushing the ball down the field." In other words, team members are committed to keeping the work progressing at a good pace.

Depending upon the complexity of the development project, and whether there are multiple sprint teams working towards an integrated product, a technique referred to as the Scrum of Scrums is used. It is a daily, weekly or monthly meeting with representatives from each sprint Team reporting progress status, next steps and/or

obstacles on behalf of the team. In addition, a key outcome is coordination among the teams relative to overcoming obstacles, interface issues, understanding the larger picture, and responsibilities. Similar to sprints, a backlog is used to track open issues, assignments, and status. The Scrum of Scrums meetings were highlighted as a key factor in ensuring coordination, cooperation, and understanding. The Scrum of Scrums also ensured that items that get moved to the backlog are re-scheduled back into the work in a future sprint(s) even if the item was moved to another team. The goal is to avoid reaching the end of a given schedule (sprint or release) without having addressed the items on the backlog list.

The following tables summarize the information gathered from the interviews on team size, number of sprints, and length of sprints. It also includes the number of sprints per release and the number of releases, where that information was collected.

TEAM and SPRINT INFORMATION

		Organizations									
		N1	N2	N3	N4	N5	I1	I3	I5	I9	I12
Size of Projects	Large	X	X	X	X	X	X	X	X	X	X
	Safety Critical	X		X	X	X	X		X	X	X
Team, Sprint Information	Number of Teams		1		3 dev 3 test					40	9
	Team size		10		6 to 10	6 to 10	5 to10		8 to 10	4 to 10	7 to 8
	SA Co-Located (Yes/No)				Y	Y	Y	Y	Y		
	Sprint lengths (Weeks)	2	2	2	4 dev 2 integ	4 to 6		4 to 6	2 to 4	2	
	Sprints/Release	6	3 month rel.		4			4	2 to 3		
	Number of releases	25	14								

Blank cell indicates no information

Table 4.3-1 Agile Team & Sprint Information

		Organizations											
		I2	Fed1	C1	N6	I4	I5	I6	I7	I8	I10	I11	U1
Size of Projects	Medium	X	X	X									
	Small				X	X	X	X	X	X	X	X	X
	Safety Critical		X*	X*		X*	X						X*
Team, Sprint Information	Number of Teams				1	7 to 8			42			2	
	Team size	8 to 9			3	8 to 10	8 to 10	1 to 26	8 to 10	4 to 8		3 to 4	
	SA Co-Located (Yes/No)	N		Y		Y	Y						
	Sprint lengths	2		2 to 8	4	2	2 to 4	1 to 2	2	3 to 4	2	1 to 3	
	Sprints/Release						2 to 3				4		

Table 4.3-2 Agile Team & Sprint Information

Table notes: * Had a few projects safety critical; Information not available for FED1; Fed1, C1, I7 didn't develop their own software; blank cell indicates no information available

4.3.1 Summary of interview information

a. The team or project, if multiple teams, needs to establish and apply a definition of "Done" that includes the quality, documentation, test coverage, usability, and other factors for a work product. In some organizations, this was more formal than in others.
b. Team make up is important and only those willing to participate fully in the Agile process should be team members. For those that don't work well with others, find other work, perhaps as floating consultants or SMEs.
c. The work is considered fast-paced and there is considerable pressure. However, the individuals usually love the work and are fully committed to products as a whole and not just their piece of it since everyone shares in the work for each product. There is no individual credit for work accomplished, just team credit.
d. An Agile coach can be very beneficial to get teams started, trained and keeping them moving. This can be an outside consultant or a member of the team(s).
e. Monitoring metrics including backlog progress, velocity, and the team's improvements is important to ensure schedules are maintained and all the work is completed and of sufficient quality. Customer satisfaction is also used as a metric.
f. In general, the more a sprint team can determine its own work processes, roles, schedule, and tools, the happier and more productive the team will be.
g. When there are multiple interacting teams, some conformity is needed to ensure progress and ease of integration. Factors that should be consistent include the same sprint length with start and stop dates, same tools, some conformity in documentation, and the same definition of "Done".

4.3.2 ABT's Observations

a. Work on an Agile team is very fast paced and can be exillerating or exhausting or both. The idea of the Agile Manifesto is that a team finds a pace they can maintain without burnout and still produce a high quality product without skipping

any needed steps. The use of a coach or consultant can help teams find this pace. Once established, the teams can monitor and adjust for themselves.

b. One of the hardest skills to master is to take requirements, or items, off the backlog for the work to be done in a sprint, translate it to stories or models and then to tasks to be assigned to developers and testers. This planning and strategizing is essential to doing just the right amount of work within a sprint timeframe. If there is too much work, the team may be tempted to take short cuts like reduced testing. To achive quality, proper planning and translation is needed.
c. Many organizations started using Agile on small projects and may have been initiated from the bottom up. Successes on the smaller projects then encouraged into more projects until some organizations ended up doing mostly Agile work. A few had a corporate approach to Agile and started from the top down. Either way, program management and above were often still having some difficulty adjusting.
d. Adjustments and some compromises were made to the chosen Agile approaches when large, complex and critical projects needing a multi team approach occurred. While some are trying to use a Scaled Agile Framework® (SAFe®) and stay in the Agile environment, it too has limitations and adds some levels of complication. SAFe® provides a recipe for adopting Agile at the enterprise scale.
e. Agile teams work best when trusted to make the decisions they need to for their work products. One or two of the corporations still valued an integrated quality approach, and one organization made sure there were 2 team members on each team dedicated to collecting and evaluating team metrics.
f. Starting small, that is, gaining experience on small projects and as experience in the workforce grows, tackling larger projects and multi-team projects was the usual approach. Jumping into a large complex project with little Agile experience and many teams and team members inexperienced in Agile is not a recipe for success. If a project can make small, independent teams for parts of the overall project with a good plan to bring them together, it is possible to gain experience in Agile while working at the enterprise level using a more waterfall method.

4.3.3 ABT Recommendations

a. To the extent possible, allow teams to self-form and determine their own operating processes, tools and metrics. Team empowerment is key to taking ownership and pride in what they develop.
b. A coach or consultant is useful to help new teams determine how it can best work together.
c. If possible, staff the teams with individuals who are self-motivated and good communicators. Training and experience can only go so far, but it helps. If an individual is not working out, find another assignment for them, Agile may not be their thing.
d. Start small, that is, gain experience on small projects and as experience in the workforce grows, tackle larger projects and multi-team projects. Jumping into a

large complex project with little Agile experience and many teams and team members inexperienced in Agile is not a recipe for success. If a project can be broken up into small, independent pieces that each team works on for the overall project with a good plan to bring them together, it might be possible to gain experience in Agile while working at the enterprise level using a more waterfall method.

e. When resources are low and the number of teams needing a particular resource is high, consider setting up a group of recognized SMEs that the teams can call on. However, resist the temptation to have the SME's do full time work and be on call. The team needs to manage their workload.

4.4 Tools

The dependency on tools to execute the Agile processes effectively varied widely from the use of almost no tools to the use of a large set of specialized tools to perform various functions. Tools dicussed in this section include all the types of tools mentioned in the interviews by both the developers and the assurance personnel. Most of the discussion in this section focuses on the tools used by the Agile teams,

Most of the interviewees used tools to support their Agile processes and recommended that tools be set up early in the process and all team members be trained on their use. Many of the organizations depended on tools for performing continuous, automated testing of releases.

While some believed each team should choose their own tools and be willing to change as the team saw fit, other larger organizations with large projects and multiple teams recommended a more standard set of tools across the teams. These larger organizations with large projects and multiple teams also felt it was necessary to listen to the developers' and testers' suggestions and to be willing to upgrade or change tools as needed. This builds team support and is important in maintaining their commitment to the overall project. Where feasible, teams should be encouraged to discover and try new tools and processes, elevating the more promising ones. Where SA was part of a team, they helped determine the tools. When outside the team(s), SA was only occasionally given a chance to have input into the tools.

4.4.1 Summary of interview information

a. Since Agile is really a tool-driven process, a comprehensive set of appropriate tools is needed to be successful and gain the expected benefits from the use of Agile. In particular, the use of an integration management tool from the start of a project is recommended, since integration is extremely important to any project.
b. Automated testing and the tools that allow for this are also primary recommended tools. Test suites need to be complete, able to be updated, and should be run at the end of each day with inputs from all the developers. Full regression testing as well as individual unit testing benefits from test automation. This includes tools that assure test coverage. There were several organizations that have dedicated testing experts at the team and unit level. For projects with multiple teams, some had one

or more teams dedicated to testing design, automation, and integration. These teams assist with team level testing and oversee integration testing.

c. Many tools are useful in supporting the Agile process. The various interviewees recommended different tools but most felt the most important were JIRA, used for general management, and some form of code testing management tools. The following partial list of tools, grouped by their functional use, are some that were either used or recommended by the interviewees to support their Agile projects:
    1. General Project Management Tools: JIRA, Confluence, SharePoint, Trello App, Daptiv, wikis
    2. Document Generator: Doxygen
    3. Communication: SKYPE
    4. Prototyping: AXURE
    5. Code Testing Analysis: Code Scanner, Application Scanner, Code Collaborator, Klocwork, SonarQube
    6. Test Management: Rational Quality Manager (RQM)
    7. Defects/Change Management: Rational Team Concept (RTC), Jenkins open source tool, Subversion (SVN), Trac, Git/Stash
    8. Requirements Management: DOORS

    Appendix H contains a more complete list of the tools that were mentioned during the interviews.

4.4.2 ABT's Observations

a. All interviewees felt tools were important and needed to be set up at the start. All members of the team should be part of the tool selection decisions since they will need to use them to assess a team's processes and work products. Some learned this the hard way and definitely felt it was a lesson learned. "No one was successful without a good set of tools!"
b. Tools can also mean different things to different people and can run the gamut from a spreadsheet or process to expensive tool suites.
c. Configuration management is needed to support the daily updates to the code and constant test updates, results, and improvements. Tools to support this are so much a part of the Agile process, at least 11 out of the 21 interviewees used traditional means (which included JIRA in many cases) to maintain their documentation.
d. While some teams had very few tools and others used entire suites of tools, all felt tools were valuable, especially for managing the process and the team environment. For this, JIRA or Rational tools were mentioned often. These are usually multi-use, suites of tools and are often used several purposes on a project.
e. Tools to support testing, such as test coverage, test procedures, test suites maintenance, regression testing support, and any needed models, were mentioned though few real specifics were gathered in this area.

f. To perform successfully in the Agile environment, all the team personnel, SME's, SA , Safety, and sometimes even customers will need to understand the function of the various tools being used in the Agile process and how to interact with the tools to have access to the information needed to perform the team and SA functions. To understand how the tools are being used in the project, the SME's and the SA personnel should be trained on the use of all the tools the sprint team(s) uses. The use of the tools will be integral in performing many of the SA functions.

4.4.3 ABT Recommendations

a. Research and set up an Agile tool suite before or as part of the first sprint, the team developers, testers and SA need to be involved in this process. When management determines what tools everyone will use without consulting the teams, it is not conducive to team building and team spirit.

   Consider Tools for:

   1. Continuous test integration and management
   2. Configuration management
   3. Capturing & reporting of backlogs, stories, tasks, activities, sprint meeting decisions, general management of processes, etc.

b. Determine any needed changes to tools at regular Sprint Retrospectives or Scrum of Scrums when tools are used for an entire project with many teams. Again, SA needs to be part of this discussion and determination.
c. Formally train the entire Agile team, including SA, in the use of the tools being employed on their projects at the start of the project and retrain as needed.
d. A tool usage assessment by the team is needed to understand how effectively the tools are being utilized and whether a change in tools is needed.

4.5 Training

It was apparent from several of the survey participants that training was a key factor and the first step towards successfully integrating Agile into their SW development process. It is important to note that several different training approaches were used. Some interviewed organizations chose in-house team training from a consultant and others sent a select team member or members to be trained who in turn would return and train the team. Some just did research and taught themselves. Ultimately, a common understanding of the Agile process and execution among the team is the goal, particularly applied to following and implementing standard Agile practices.

Expansion of Agile training to the various levels of the organization helped facilitate awareness and mindfulness since Agile is usually a significant culture change. An Agile-

knowledgeable management can provide better support. Ongoing coaching, training and refinement of the Agile processes from either an internal Agile coach or from a consulting company was a common occurrence and interviewees that had coaches felt they were necessary and had made a vast difference in team productivity.

One of the hardest things to learn in the Agile process was how to determine the amount and portions of the work that should be part of any single sprint. It was also difficult to learn how to take the functional requirements and turn them into stories and tasks, which are then divided into the activities the team members work on. A class or two or even a sprint or two does not make someone an expert at these critical steps to successful Agile development.

4.5.1 Summary of interview information

a. A culture change is needed to fully embrace Agile. Agile training should start top down, with the management team learning about the Agile process and championing Agile training of the organization. All the varying levels who are stakeholders in the Agile process, from management, to procurement, to the customers and users, to the development team, and to SA, need to be aware of the Agile process. Training should establish a common understanding for the entire organization and should include the changes needed by each part of the organization.. Instances were reported or referenced where this didn't occur and it created problems in the organization.
b. Training was a common theme. In-house training as a team prior to starting an Agile development was highly recommended by several of the interviewees. One interviewee stated, "It's difficult to do Agile right, biggest hurdle to doing Agile is thinking you know what it is… without really learning what it is… need to actually learn Agile Scrum" or whatever method of Agile is chosen.
c. Team members must be willing to be trained and accept change. It is more challenging for some to change than others. New hires and recent graduates seem to adapt to it more easily. A little extra time and patience may be needed for some team members.
d. Scrum was the most common Agile methodology used by the interviewees and probably has the most easily accessible information and straightforward approach.
e. At least one person on the team needs to be trained in the Scrum Agile Process, usually as a Scrum Master (or trained in whatever the selected Agile process is, e.g. Kanban, Crystal) and then he or she can train the team(s).
f. Other team members, once introduced to the principles, can do well with on the job training (OJT) training. A lot of OJT seemed to occur among the interviewees. However, initial training as a group helps build norms and expectations and is usually the best training approach.
g. Generally, the time it takes for a team to get into a good sprint cadence than is overestimated at the start of a project. There is a learning curve for effective use of Agile.

h. Also, it is very important that team members have training in the required standard practices required by the organization (independent of Agile process). This needs to include any regulatory, security, safety, and customer requirements.

i. Agile process problems should be discussed as a team in the retrospective meeting. This helps minimize the temptation to break the rules and improves process efficiency (continuous improvement). This is an expected Agile norm that is often neglected by the Agile teams not thoroughly imbued in the Agile methodology through training, coaching, or long time experience.

		Organizations									
		N1	N2	N3	N4	N5	I1	I3	I5	I9	I12
Size of Projects	Large	X	X	X	X	X	X	X	X	X	X
	Safety Critical	X		X	X	X	X		X	X	X
Use of Coach and Type of Training	Had Coach									X	
	Scrum Master-trained							X	X	X	
	Team Lead was trained	X		X							
	Team trained in class									X	
	Team trained by team member or team lead							X			
	OJT		X	X	X	X	X		X	X**	X
	No training										

Table 4.5.1-1 Large Agile Team Training

Table notes: **Most of team previously trained

		Organizations											
		I2	Fed1	C1	N6	I4	I5	I6	I7	I8	I10	I11	U1
Size of Projects	Medium	X	X	X									
	Small				X	X	X	X	X	X	X	X	X
	Safety Critical		X*	X*		X*	X						X*
Use of Coach and Type of Training	Had Coach			X		X		X	X	X	X	X	X
	Scrum Master-trained	X		X		X	X				X	X	
	Team Lead trained team												
	Team trained in class				X					X			
	Team trained by team member or Lead					X						X	X
	OJT	X				X	X	X	X**		X	X	
	No training												

Table 4.5.1-2 Small to Medium Agile Team Training

Table notes: * Few projects were safety critical; ** Most of team previously trained; Fed1, C1, I7 didn't develop their own software

4.5.2 ABT's Observations

a. The data in the tables above shows that smaller, medium sized teams had more direct training via coaches, Scrum Master training, or whole team training. The big projects seemed to jump in without formal training or a coach to support them. Five organizations only had OJT, which could mean either these teams were mostly made up of Agile-experienced personnel, or the whole team learned as they went along, or a combination. We cannot actually tell from the data above which is the case, but we know that some of the larger companies that have been doing Agile for a while, felt their personnel were all pretty much at the expert level and mentoring was used for any new hires.

b. Initially, it's critical that the team is trained in the general Agile methodologies and this needs to include SA. It is important that they train with the development team(s) so that they are aware of the process variations the team may choose to use.

c. Also, since the team's processes often change during the project, it is imperative that the SA personnel participate periodically in team Scrums, Reviews, and Retrospectives to assure that any changes to the methods are understood and the impact on safety, reliability and quality be considered.

d. Teams sometimes take shortcuts and eliminate some of the vital Agile processes. This is generally due to a lack of understanding of why Agile, which is already a lean development method, put those process steps in to the process to begin with. Several interviewees stated that while Agile was adaptive, the basic steps still needed to be followed. One interviewee stated, "Be humble and follow the Agile steps, the experts put them in there for a reason."

4.5.3 ABT Recommendations

a. NASA Management should, at a minimum, take the SATERN on-line introduction course on Agile development. Project and Center management with several Agile projects should have training and coaching from a consultant in the Agile form(s) being used on their projects. Involving members of the teams allows for questions to be asked by all sides and can result in a mutual understanding.
b. Support from management begins with understanding the real benefits of Agile and the changes that may be needed in the safety critical environment of most NASA projects.
c. Training of the NASA and contractor SA personnel in the use of any tools or systems that will be required to access the project information is necessary.

4.6 Documentation and Configuration Management

Documentation and Configuration Management (CM) are key enablers in maintaining and establishing system confidence, assurance, auditability, efficiency, and predictability. The amount of documentation done on projects varied considerably from comments inside the code to a full, well-structured, set of documents usually associated with a formal Waterfall process. The survey participants shared several different perspectives relative to documentation and CM. While documentation should not be the focus, it is necessary to have the information to support audits, certifications, reviews, maintenance, logistics, and training. Interviewees highlighted that there are various forms and approaches to documenting needed information. Several examples of automated tools and models were recommended as alternatives to traditional paper documents. Since Agile is very dynamic, an integrated and readily accessible means to capture updates, changes, and backlogs is needed to support the very fast-paced culture of the Agile development process. Those who are writing contract requirements need to be aware of various means and methods available for Agile documentation, and they should list alternatives in the contract to provide the flexibility needed to fulfill the documentation and CM requirements with methods chosen for use with the Agile developments.

4.6.1 Summary of interview information

		Organizations									
		N1	N2	N3	N4	N5	I1	I3	I5	I9	I12
Size of Projects	Large	X	X	X	X	X	X	X	X	X	X
	Safety Critical	X		X	X	X	X		X	X	X
Configuration Management/ Amount of Documentation	All artifacts with CM rigor	X	X	X	X	X	X			X	
	CM in documentation tools							X			X
	Teams decide how to do CM										
	Limited CM								X		
	Tool mentioned		X					X			
	Waterfall type documentation	X		X	X	X	X			X	
	Limited doc. for reviews		X								
	Doc. in code, tools							X			
	Limited documentation								X		X

Table 4.6.1-1 Large Agile Team Documentation

		Organizations											
		I2	Fed1	C1	N6	I4	I5	I6	I7	I8	I10	I11	U1
Size of Projects	Medium	X	X	X									
	Small				X	X	X	X	X	X	X	X	X
	Safety Critical		X*	X*		X*	X						X*
Configurationo Management/ Amount of Documentation	All artifacts with CM rigor	X									X		
	CM in documentation tools							X	X	X		X	X
	Teams decide how to do CM			X		X							
	Limited CM						X						
	Tool mentioned	X							X				
	Waterfall type documentation										X		
	Minimum for regulations, audits		X					X	X	X			
	Doc. in code, tools	X											X
	Limited documentation			X		X	X					X	

Table 4.6.1-2 Small & Medium Agile Team Documentation

Table notes: * Few project safety critical; Fed1, C1, I7 didn't develop their own software

4.6.2 ABT's Observations

a. Since the use of a wide range of information and methods is expected to make the information available to project participants, there should be increased importance on making sure the contract states the expected artifacts and the means and schedule to measure them.

b. The contract and the NASA SW Development and Assurance Plans should specifically include the methods and tools/systems to be used for accessing or accepting delivery of the information, metrics and artifacts. This may mean

NASA access to contractor Agile tools such as JIRA. At the very least, contracted Agile projects need to give NASA Engineering and SA limited access to these tools, upon request, or they need to provide outputs from the tools with the requested information.

c. The Configuration Management of the project documentation, artifacts, and information is expected to take on added importance and complexity with the expected large increase in variations of the software systems that are being developed using Agile processes around the Agency. A NASA SW Development Plan should address the specifics of how the project will perform Configuration Management and what tools will be used. Based on the documentation, information, and configuration management plans and/or processes provided, the NASA and contractor teams should be trained in the use of those systems. SW Engineering, SW Management, and SA all need this information.

4.6.3 ABT Recommendations:

a. NASA needs to put additional emphasis on the Contract, the NASA SW Development Plan, and contractor development and team practices and processes to ensure they address appropriate Agile artifacts and processes.
b. What, when, and how information will be available to NASA personnel needs to be documented. The level of NASA integration into each team or project of teams needs to be understood, agreed upon, monitored, and measured.

4.7 Requirements/Functional Stories and Agile Tasks

As with any development approach, requirements need to be high quality and clearly defined to support engineering and development and ultimately lead a successful project. Specifically for large complex projects the survey participants stated that it is extremely beneficial to have well defined, high quality requirements at the start of a project. At least the high to mid-level, architectural requirements need to be well defined, leaving room for invention at the sprint levels. Contrary to this, smaller projects reiterated that the lack of requirements, other than high-level feature and interface based requirements, was not a hindrance. This gave them the freedom to discover what would be the best for the customer. Alternative functional/user stories are used to define product features that map to software requirements and include success criteria, from which test cases are written for verification. It is important to recognize the various forms of requirements and how they can be used/applied. Sprint and Scrum meetings, and for Scrum, the Sprint and Project Backlog Review meetings are where requirements are managed. This is where new or changed requirements, and requirements that did not complete in the sprint are analyzed and prioritized for incorporation into the software.

4.7.1 Summary of interview information

a. System level requirements are a must for the larger projects. Large, complex or critical projects found it extremely beneficial to have a good set of requirements at start of project – especially, an overarching design with defined interfaces. A team, systems engineer, or program owner maintains this higher level architecture and helps break it out for the sprint teams, defining intermediate releases and interfaces.
b. On smaller projects such as web applications or mobile tools, when the customer was not sure what he wanted or when the team needed to do prototyping to determine the best method, the lack of project requirements was not a hindrance. They still needed requirements for the platform usage and interfaces, but the applications started with very general features and interfaces.
c. Usually, a set of solid system requirements or functions are established, which are then broken into functions and features for Sprint teams. These lead to stories for that team and from there a backlog is created for the project, each team and each sprint. The stories themselves can be shared and created with the customer. The stories need to include nominal and off nominal scenarios and be broken into 1 to 2 day doable tasks that each coder or pair of programmers can design, code, and test.
d. Establish a solid requirements, functions, and features base with good stories, use cases and tasks before coding for each sprint begins.
e. An infrastructure for an iterative process is not that unusual. Requirements partitioning already is common on projects, and quality testing remains nearly the same.
f. In situations where there is no clear set of requirements, an Agile approach may help get the requirements determined by pulling together the stakeholders in a dynamic process of discovery. Early builds will solicit feedback from the customer and users. Building incrementally on the baseline will ensure the product meets the needs and desires of the customer.
g. Generate a minimum set of system level architectural requirements up front based on the desired functional requirements for the system that will allow the functional details to be developed. The architecture design is done at the project/program level before the sprint teams start development.
h. Some projects struggle with the gap that exists between traditional approaches that focus on requirements and those that focus on user stories and features. When the project tries to maintain both perspectives, a translation is necessary to bridge the gap from one to the other.
i. Include user stories or use cases as part of documentation for requirements.
j. Be sure user stories include acceptance criteria in order to form better requirements.
k. User stories define product features, which are incorporated into release objectives, integrated across multiple teams. Product features map to software

requirements, which have success criteria, from which test cases are written for verification.

l. Requirements (could be task, story, model, or function descriptions) direct the testing.
m. Sprint coding should be done after well-defined tasks are established and selected.
n. Advisory groups should be set up for detailed subsystem requirements. These groups are made up of specialists or unique user groups, like pilots, or SMEs.
o. Scrum of Scrums keep others informed. They are a problem-solving, communication and coordination vehicle, to ensure the ripple-effect of issues is well-known. Addressing the right requirements is more high level. - It is key for the project to consider every iteration.
p. At a lower level, establish requirements prior to the sprints. The predefined, “Done”, is written so that no one gets credit, and then only the team gets credit, once the development iteration/unit meets the definition of “Done”
q. Agile promotes information sharing rather than code hoarding, and territorialism. This is especially important as all team members need to take ownership and may work on all parts of a product. A cultural shift is needed to get used to this.
r. Testers need to be involved throughout the sprint, learning, and writing new tests. When the coders are also testers, they need to write the test along with the code.
s. It was recommended from an Agile perspective that the philosophy is to develop/tackle the more challenging backlog items (features/requirements) first. Highest priority items to be worked from the backlogs are those that are most critical or of highest risk to the project.

		Organizations									
		N1	N2	N3	N4	N5	I1	I3	I5	I9	I12
Size of Projects	Large	X	X	X	X	X	X	X	X	X	X
	Safety Critical	X		X	X	X	X		X	X	X
Requirements	User Stories	X		X				X	X	X	X
	Requirements generated by other groups			X	X						
	Captured in models, DOORs				X	X					
	Developed by interacting with customers using interim products						X				
	Archiectural requirements done up front/requirements done before development sprints	X		X	X**		X**			X	**
	Use of tickets to manage requirements								X		X
	Use cases										

Table 4.7.1-1 Large Agile Team Requirements

Table notes: ** Had problems when requirements were not base lined early

		Organizations											
		I2	Fed1	C1	N6	I4	I5	I6	I7	I8	I10	I11	U1
Size of Projects	Medium	X	X	X									
	Small				X	X	X	X	X	X	X	X	X
	Safety Critical		X*	X*		X*	X						X*
Requirements	User Stories	X		X		X	X	X	X	X	X	X	X
	Requirements generated by other groups		X						X				
	Captured in models, DOORs												
	Developed by interacting with customers using interim products				X	X							
	Use of tickets to manage requirements						X						
	Archiectural requirements done up front/requirements done before development sprints					X	**		X	X	X	X	X
	Use cases												X

Table 4.7.1-2 Small & Medium Agile Team Documentation

Table notes: * Few projects safety critical; ** Problems with architecture decisions – architects not always consulted; Fed1, C1, I7 didn't develop their own software

4.7.2 ABT's Observations

a. Even though the Agile manifesto does not emphasize generation of requirements at the start of a project, the organizations interviewed almost always generated a good set of requirements before going into the software development phase, where the project is divided into one or more sprint teams which then take the features or requirements and make the sprint stories and tasks from them. This was especially true of large, complex projects that typically are supported by NASA SA activities.
b. Even those that use some of the Agile methodologies, some levels and forms of requirements were in place early on in the project. However, the requirements can take various forms that may be different than the traditional Software Requirements Specifications (SRSs). These include models, features, use cases, UML, stories, and others. Since project requirements are a critical input to many of the SA activities, it is important that SA understand how the various forms of requirements translate into traditional requirements and how they can be used in performing the SA functions on the project. Also, since Agile based projects generally are amenable to changing requirements during the development of the project, SA should be closely tied into the change process to be in a better position to efficiently and accurately perform their functions. A good way of keeping up to date on requirements would be to embed SA into the sprint teams and their Scrums where requirement backlogs, which include new or changed requirements, are analyzed for incorporation into the software. Also the backlogs are discussed and evaluated at Sprint Review meetings and the Sprint Planning Meetings. Some organizations have a special backlog-grooming meeting between the Sprint Retrospective and the Sprint Planning Meeting. Others make it part of one of the other meetings. During this meeting, the backlog, which holds the functions and requirements to be done, is carefully reviewed and re-prioritized. Anything that was found missing or any new customer issues added, or items that should have been done in the last sprint, but didn't, are put back on the backlog. Safety and reliability features that may come from another team are examined and all this work is prioritized, with the highest priority items going to the front of the line.
c. Safety Requirements: all the system level safety functions and hazards (hazard causes, effects, controls and mitigations) were placed on the project backlogs and then those related to the features for a sprint team were rolled in their backlogs and identified as safety requirements/items. Once at the Sprint level, safety critical items need to be evaluated and based on criticality. Any system hazards related to software must have been eliminated, mitigated, controlled, warning provided for, and/or accepted.
d. While the quintessential Agile process shows requirements/functions being worked and maintained as a series of sticky notes that evolve into user stories and tasks with matching tests, and in deed they may well use this method to

brainstorm ways to combine and work the requirements into user stories and appropriate tasks, eventually all this does get captured. Capture methods may include pictures of the sticky boards or a written record from someone on the team. The frequent use of JIRA or other tools to maintain and monitor team progress shows that many of the organizations do more than just sticky notes. The interviews did support the claim that sticky notes, except for the smallest, quickest projects, were only part of the documentation and development process.

e. Many of the readings, classes and interviewees admitted that learning to write good use cases is hard and that not everyone does it well, or at least not right away. Many learned the hard way that it took many sprints to get right-sized and complete use cases that could be turned into appropriate sized tasks that could actually be accomplished in a sprint. Often those same organizations that learned through OJT, recommended a coach or consultant to help get started faster.

4.7.3 ABT Recommendations:

a. Project management, SA, acquisition, and procurement all need a level of training to understand the different forms that requirements take and how they are documented within an Agile process. Specific training for the documentation approach taken is needed on a project by project basis as well. How well a team can manage the requirements and the process to transition the requirements to design, use cases, models, tasks, and tests should be understood. The safety critical and high risk functions of the system should be monitored. It is especially important for SA to understand and even be part of the transition from requirements to task and test and then to monitor the configuration management of those elements.
b. Obtain a coach to help teams get started, especially to help the team learn how to write good use cases and user stories. Use cases and user stories are fairly universally used to describe how requirements go from high level to a sprint's set of requirements, and then those get turned into tasks for each developer and tester. This is an area that it takes practice and support to gain some proficiency and it is key for working off requirements. SA should be part of this as well.
c. For medium to large projects with multiple teams, create and maintain an overarching architectural set of requirements that include interfaces.
d. Assure that sprint and project backlogs are prioritized with the highest risk and safety critical functions at the top of the backlog. This could be one of the things SA monitors for software system and sprint level backlogs, providing a systems view across the project.
e. Ensure the existence of a valid trace of the system hazards to the software hazards to the project and sprint backlogs. Ensure that the specific tasks/functions are identified as safety-critical, and that the verification method is test. If the verification method is other than test, ensure that the rationale is justified.

f. Configuration control is a key component and it is important to ensure that there are good configuration management plans and tools. The configuration control execution needs to be established and monitored.

4.8 Verification and Validation and Testing

Verification and validation (V&V) typically includes major and minor reviews, code walkthroughs, formal inspections, and testing. While the typical milestone reviews are not usually performed on Agile projects, they can be done on larger projects. The typical Waterfall type milestone reviews serve two major purposes: 1) to inform the management and customer of progress and 2) to allow internal and external scrutiny into the viability, completion, quality, reliability and safety of the requirements, design, testing, and conformance to customer needs.

Agile mostly relies on continuous testing, Sprint Reviews with customer feedback, and backlog refinement and reviews. That is not to say that many of those interviewed did not also have and use some progress metrics, usually the backlog of features (requirements) to be done and velocity at which the team is able to complete the set features in a given sprint. There are project backlogs and team backlogs, and any errors found and not addressed in a sprint are added to the sprint team's backlog. At the start of each sprint, the backlog is reviewed and work to be done in the next sprint is pulled off and worked (the features turned into stories, tasks and details). It is important that the team define "done". Some organizations had very elaborate definitions of what done consisted of and "done" can be defined at multiple levels. For the typical sprint, the software for that sprint was not considered "Done" unless it completely covered the set of tasks the team undertook, was operational, was fully tested and documented, and any quality or safety checks were performed.

Part of the Agile methodology includes a very vigorous testing approach. Testing is done almost daily, if not daily, and is expected to have a well-maintained and integrated regression test suite as the project evolves. In fact, a strong regression testing suite that is maintained, i.e., kept up to date with the daily completion of tasks, along with the tools and configuration management required is considered to be essential to Agile success. For large systems being produced, it was not uncommon to find that one team was dedicated to creating, maintaining and running the master regression tests. They would take input from each teams' test suites and incorporate them into the larger system test suite. They also usually acted as advisors to the sprint teams to help them with their testing. All of the teams would then use this master suite of tests.

4.8.1 Summary of interview information

a. Everyone should be jointly responsible for quality assurance and for verification and validation of safety critical requirements. There should not be an "us versus them" attitude between separate roles - this is a red flag, indicating a dysfunctional team.
b. Verification and testing is continuous, if possible. (Tools are needed to make this feasible.) Don't let the developers get too far ahead of verification, or a lot

of rework may have to be done as changes arise. Some development efforts may need to be redirected to help with verification.

c. For large projects, where an Agile framework like SAFe is used to manage multiple sprint teams, called trains, the teams need to match the rates of the trains to sync entry into the Integration train every three months or whatever the determine time period is. It is important that each sprint team be on target for meeting their portion of an integration period and have their portions of the code tested and ready. Spread the verification more uniformly across the development trains being fed in.
d. Increase automation wherever possible along the way.
e. Several modifications to standard Agile process were noted:
    1. A separate automation team could be assigned to ensure the automation of verification and validation keeps up with project development. This team works together with a separate testing team, who is responsible for integrating hardware and software team outputs on a continuous basis.
    2. One modification to pure Agile is that instead of "requirements – design – test" being done every sprint, the requirements may be "done", or base lined, before the sprint, and just the design done within the sprint, with the next sprint run for testing. This was not observed very often.
    3. Testing may have to follow a modified schedule outside of the regular Sprint schedule, depending on the level of automated testing and emulation that is in place.
f. Process adherence as well as testing is emphasized.
g. While some organizations interviewed believed that quality came from "failing fast" and "fixing it fast" or from continuous testing, others felt that "Testing does not equate to quality." Both were needed.
h. Whenever possible, try to work towards continuous integration to streamline testing and discover defects early. Maintain unit testing at the developer level.
i. Several interviewees stated, "Automated testing is necessary."
j. The secret to faster development is to build in automation as you go. Continually add testing with regression with a goal of no less than 90% of testing automated (from unit to functional to system integration). Make everyone responsible for testing, and have everyone follow a tight discipline of testing daily.
k. It may be a cultural change, but testing should be "baked in" (part of the daily routine) from the start. Everyone should take ownership of work products with no competing attitudes between developers and testers. All team members are equally part of the team even if their roles are separated.
l. Developers are usually happy with team self-organization and the flexibility that Agile brings. Testers often are separate from the developers, but some teams have everyone do everything, while others have their testers do limited coding, and coders do some testing for overall collaborative teamwork.
m. Providing assurance within an Agile framework is difficult because testing results aren't always captured formally. Time pressures often mean less testing. Reliability is often overlooked as a success criteria or a user story. Hazards and risks always need to be identified with their likelihood and impact.

n. Have clear marching orders, clear acceptance criteria, and definition of “Done”, i.e. every user story or feature should be production ready and automated testing should be in place for recursive or regression testing.
o. With multiple teams working simultaneously, such as in a SAFe® (Scaled Agile Framework) environment, keep all iteration lengths the same, with same start/stop times. Localize system testing and at end of each iteration use dedicated system-wide testing to integrate everything together.
p. Independent testing is important at every release, if not at every sprint.
q. Incorporating SA within Scrum teams would be ideal, but it is important to guard against the loss of impartiality and fail to recognize issues or problems or defend poor practices. Risks need to be pointed out honestly. More testing does not equal SA.
r. Invest in the infrastructure early. Tools to enable continuous testing are essential. Stand down development until the setup is complete, if needed. If output slows to a trickle, stop and reassess.
s. Bring requirements in line with product-based features, instead of the current SRS-based listing to get to feature testing sooner. Sprints are usually geared to produce features such as command acceptance, initialization, create report, etc.
t. Design for testability. Use automated static testing tools for test coverage metrics. Simulate mock errors to do off-nominal or “evil” testing (trying to break code). Test daily.
u. Everyone is responsible for a good product.
v. Before final deployment after acceptance testing, security penetration testing should be performed.
w. It is imperative that the V&V/testing members of the sprint and development members of the teams develop the test and acceptance criteria together so that there is no uncertainty about what is expected and what constitutes working software.
x. When there are separate teams dedicated to testing and fuller V&V activities, they usually do their own sprints. They helped the product sprint teams with their lower level testing which in turn got rolled up into a complete regression test for the entire project or parts of the project which the testing sprint team then maintained. Any changes at the lower levels needed to be brought up and assured to work at the higher level. Product level teams then could use the higher integration level test scripts when testing the greater workability of their part of the project and when working on integrating with other product sprint teams.
y. Determine if the end of each sprint is a releasable product or not. This should put the appropriate level of verification and SA support in place early on.

4.8.2 ABT’s Observations

a. Based on the information gathered, it is apparent that automated and continuous testing is a mainstay of successful Agile based development projects. In concert, understanding the quality aspects of testing is a critical component. It is also noted that it is very important that the testing plans, success criteria, and the use

of testing tools should be addressed/developed by the integrated sprint teams early in the process. To this end, it is important that the sprint teams' planning meetings assess the quality aspects of the software and system testing in order to provide insight into needed software assurance and quality aspects of the entire verification and validation processes.

b. Also, many of the projects underestimated the amount of verification testing that would be required during the planning phases due to a poor understanding of the number of releases that would be made and the amount of regression testing that would be required and performed on the releases. The method of handling inadequate verification planning for a sprint or project needs to be decided and the risks assessed.

c. The larger projects may assess their verification activities at the sprint level and at the intermediate and final releases level.

4.8.3 ABT Recommendations:

a. For large projects, the structure and management of verification and validation (V&V) should still have an overarching systems approach. This will help manage and address any teams falling behind on testing and other forms of V&V, maintain quality of the overall V&V process, and assure a systematic approach to integration.

b. Assure that testing goes beyond, "Does it work?" but also considers, "How does it fail?" Stress testing needs to be performed as part of the daily testing schedule.

c. Especially for larger projects, a precedent may need to be established for all sprint teams to determine what happens when the end of a sprint does not produce usable code that meets the "done" criteria. Each sprint team needs to address this issue, but failure to address this inevitable position has a greater impact on the project when multiple sprint teams are working toward one goal.

d. Invest in the verification, validation, and testing infrastructure early, especially for the larger projects. Tools enabling continuous testing are essential. Ideally, before development gets underway, a set of tools for testing needs to be agreed upon and set up and processes for the testing need to be in place. Changes and problems need to be discussed and addressed as the project unfolds. Be willing to make changes if necessary as the software progresses through the levels of testing.

e. Ensure that the SA personnel can be embedded into the project sprint teams during the planning of the testing processes to the extent necessary to both understand the fidelity and limitations of the testing tools, test plans, and procedures. Then they can provide insight into verification, validation and testing options to improve a sprint team's risk posture.

f. Since testing is an important part of the Agile process, emphasis needs to be put on test procedures, set-up completeness and integration as well as the management of the inputs and outputs of the daily testing. Taking the Agile perspective into account, the project and the sprint teams need to assess whether the verification testing performed by the project meets NASA Engineering and SA standards. A review of the testing processes, procedures, configuration management of tests, test

set up, and the results should be conducted by either SA or by an SME internally within the team. This review should include the extent and depth of the testing suites, the configuration management, and the procedures including checking for off nominal and stress testing as well as testing for safety controls and mitigations, when present.

4.9 Acquisition/Contracts

A consistent theme the Benchmark Team heard was that the procurement agreements did not refect the Agile schedule, metrics, or product delivery artifacts. Many procurement agreements still ask for waterfall type milestones and products. This caused problems for normal Agile development, since the deliveries are actual versions of working, tested code with various features or changes added until the project is determined complete by both developer and customer.

If a customer wanted to know about completion times and how the agile team was doing, they would need access to their velocity metrics and backlog records. If the acquirer is working with the agile team on their sprint schedule, they will have that ccess. Customers also need to schedule more of their time to participate in the delivery assessment for each sprint or at least work more closely with the supplier Product Owner since each actual sprint cycle code delivery should have customer interaction. Agile development is intended to have much more customer interaction both much sooner and more often than in traditional waterfall style development. This can work to the advantage of customers who are not exactly sure of what they want and tend to change the requirements as they go along. Changing requirements and rapid prototyping are two of the areas where Agile excels. Contracts need to be written with the Agile process in mind, when that sort of project is being proposed.

Acquisition and/procurement organizations unfamiliar with Agile development often write in requests for the traditional planning documents, requirements documents, design documents, then testing documents and finally the acceptance test and delivery of the software product as code. For larger more complex projects where a hybrid approach is best, many of the waterfall type reviews and products might still need to exist, and the contract or agreement needs to be written with this in mind. The bottom line is we could address the requirements better in contracts that involve software and allow for Agile development as part or all of the supplier process.

4.9.1 Summary of interview information

a. The RFP (Request for Proposal) needs to allow for Agile-based programming as a possible approach and if so, have the proposer provide details of the Agile approach and list their experience and personnel with specified skill levels.
b. The RFP needs to specify the customer interface and feedback mechanisms in some detail since customer interaction is a major part of the Agile process. This should include the metrics the company will use to measure progress.

c. Provisions to allow for change should be built into the contract such that changes should not cost anything extra if they are within scope. The Agile process is geared to react to changes quickly and to provide a way of making the final product what the customer requires and wants.

4.9.2 ABT's Observations

a. Even though an Agile process is proposed, it is advisable to ask for reasonable documentation particularly of hazard and reliability analyses, as well as items like overarching design, SW assurance plans and development plans. It is complicated when contracts call for progress status at software milestone reviews when instead they need to ask for progress at set intervals and observe that the project backlog is going down and the most critical items are being addressed first. In Agile, the close interaction with the customer should keep the customer informed on progress and any impediments on a monthly or quarterly basis.

4.9.3 ABT's Recommendations

a. Training for PM and acquisition and procurement officials in Agile is recommended. Perhaps provide coaching on how to write contracts and RFP's when Agile may be a possible development approach.
b. NASA needs to write the contracts to allow for Agile-type deliverables, metrics and interactions. It is important that the adaptive nature of Agile development be exploited and the contract should specify that final product acceptance will occur only upon NASA's approval that the final software is working and that all needed/required documentation is up to date and finalized. This means that changes are part of the development process and a certain level of change should be accommodated in the contract so additional cost is unnecessary unless there is a request for significant change in overall functionality.
c. Contracts need to include the type and frequency of access to the sprint teams and a description of how that important interaction with the customer, NASA, will take place. Closer customer/acquirer interaction is necessary. NASA designates need to be aware of the time involved and the needed means of communication. Will NASA be integrated in to the teams in some fashion, or will a company Product Owner represent NASA for most of the sprints? All these decisions need to be made and documented in the contract.
d. NASA SA and SW developers may need to have some access to sprint tools which hold the artifacts and determine what needs to be delivered besides "working code" For instance, are the tests suites delivered, any operational manuals? Once we take ownership what will be needed to maintain and change the code? All this needs to be determined and spelled out in a contract.
e. Ask for what is needed to assure the quality, reliability and safety of the product. This certainly should include the hazard analyses and reliability analyses as they are developed as well as the needed deliverables of the complete safety hazard process. The quality metrics for most Agile projects include backlog burn down

rates, ability of the teams to produce “done” code at determined rates of sprints. The percent complete of the functionality or code statistics metrics are also valuable.

5.0 Key Findings

1. **Agile approaches in use: Organizations with multiple smaller project teams and short term projects (1-3 teams with 4-7 team members each and completion within 6 months to a year) tended to be more engaged with a purer Agile approach while larger, more complex projects that extend beyond a year tended to have waterfall elements such as an overarching design architecture, rolled up intermediate releases, and reviews.** All projects are tailored to some extent either within the structure of an Agile method (e.g. Scrum, Kanban, Crystal) or blending Agile with other development methods such as Waterfall. “Pure Agile” of any kind would be hard to determine as each team, project, and organization puts their own characteristics on how the teams operate – that is part of an Agile approach. The form of interaction with the customer also has an impact on the Agile team processes.
2. **Documentation is an important part of the Agile process, but it is kept to a minimum and most documentation is kept within the suite of tools used to manage development.** It was very important to have the tools to capture the functions, stories, tasks, backlogs, code, and testing scripts and processes used for a project. Those who claim Agile does not have or need documentation are not really using the Agile processes. Documentation is mostly electronic in form and kept to a minimum, defined as “just as much as is needed”. In addition, Deliverables that are customer needs, such as hazard analyses reports, regulatory or certification documentation, testing coverage proofs, etc., are considered necessary and must be completed to a predetermined level for each sprint and for the final product.
3. **Most organizations interviewed have very limited actual Software Assurance. For those projects with software assurance, it was integral to the teams.** Organizations using Agile have differing views on how to achieve safe, reliable, quality software, but, in general, they certainly do not perform Software Assurance to the level of NASA in most cases. It should be noted, that only 7-8 of the interviewees stated that more than 10% of their work could be considered safety critical, and 4 of those were NASA projects. However, several high-producing, high quality-oriented organizations who have fully embraced Agile have a form of software quality embedded in their teams and consider it an essential part of the daily team effort. Several organizations have software quality and assurance performed by Agile consultants, who work with 1-3 teams within an organization to help them better manage the Agile process and stay on track with quality and performance of the process, products, and teams. Safety, reliability and security, when considered, were often provided by SMEs that

joined the teams when and where necessary. Usually, the SMEs supported a few teams at a time unless a team needed them on a daily basis, then they became part of the team.

4. **A strong teamwork approach is essential. Teamwork is what makes Agile successful.** Once accustomed to the team approach, most team members excel in the team environment. The Agile team needs to have some level of autonomy to jointly agree on how to best work together, tailoring the processes to meet their needs, and altering it as needed. Agile teams working on smaller projects usually determine most of their own processes, tools, schedule, roles and responsibilities, and the definition of "done" within the given Agile framework. Collaboration is key component and only the team gets credit for success, or failure, not individuals. Communication and honest interaction is essential to team health, progress, and a successfully completed project. Teams can be configured in many ways. Some teams are heavily cross-functional, where everyone develops and tests and plans to some degree and team consists mostly of generalists and systems engineers. Other teams are formed to have more specialists within the team: those dedicated to testing, those who only develop, those who code primarily, and those who plan and organize. Still, all are responsible for creating working code and supporting products. SMEs often were available for either generalist or specialist role types of teams.
5. **Projects with multiple, simultaneous sprint teams need both a strong internal team cohesion and a means to be related to the larger project and other teams**. Some independence of the single team needed to be sacrificed for a more uniform project level approach in most cases. A regular Scrum of Scrums brings members of each team together to help decide best tools, schedules, roles and responsibilities and helps determine progress and changes for all sprint teams.
6. **Tools, the right tools, are essential, especially for larger projects**. Set up tools early and be willing to advance or change those tools as needed. Smaller, more autonomous teams often picked their own tools. Some organizations provided more centralized tools for all the teams especially if they needed to coordinate, but were most successful when the teams helped determine the tools to be used. Buy in from the team(s) members on the tools was critical to their usage success. Automated test tools, allowing for continuous integration and regression testing set up at the start of the project were cited as the most important. The other tools most commonly mentioned were tools for capturing and maintaining stories, models, tasks, use cases, and backlogs as well as tools for managing and reporting on deliverables, intermediate products, and development progress. The whole team and any SMEs, assurance, and consultants need to be trained on the tools together. JIRA was cited most often for overall management of the process and products. When writing contracts, it is important to include tool usage expectations. Contract Data Requirements

Lists (CDRLs) or documents may not be the best method for sharing or showing work progress. The customer needs to state visibility and access requirements and may need to be trained on the tools as well.

7. **Functional requirements need to exist for Agile projects and are important for their success. They just aren't always referred to as "requirements".** Information from research and interviewees both indicated that large projects need to perform the development of the overarching architecture requirements and design, upfront, and then maintain it at a systems level throughout development. Functional requirements are pulled off the backlog and further defined are using many forms, including models, user stories, and product features, which are then broken down into activities or tasks for each sprint. Packaging functions and requirements into proper sized activities and stories is a difficult skill to learn. Often, use of a coach or consultant helps reduce the learning curve for this team skill. Priority is determined and recorded in the backlogs and high priority items are to be worked first. The teams agree that safety critical features and high-risk items are high priority.
8. **Continuous improvement of the team is vital to the Agile process.** Built into the Agile process are the daily scrums and sprint retrospectives where honest discussions of what is working and what does not work takes place. At the end of each sprint, after the product review meeting, the team works together at the sprint retrospective to look for potential improvements in processes, tools, and ways of working. This is an important step that some not truly imbued in the Agile process may skip. Team norms, changes to Agile processes and meetings, refinement of critical definitions such as "done" and "quality", tool needs and uses, communication needs, best practices, and even team member makeup and performance are all explored, discussed, as well as plans for improvement changes for the products and the team as a whole. The team determines the steps needed to make improvements, documents them and tries them in the next sprint or two to see if improvements are working.
9. **Train the team as a group so everyone has a common understanding.** Training for all members of the team, including any management, assurance, SMEs, and even customers is important. Often either a consultant would train the team(s) or one or two members would get formally trained, then they would train the rest of the team. Especially when getting started at an organization, the use of a Champion and coach was found to be very helpful to the teams.
10. **Strong integral verification and validation processes are key for project success.** Both research and interview results stress the Agile reliance on testing and other forms of verification and validation on a daily, weekly and sprint basis. In order to perform adequate and continuous testing, automation is almost a necessity. Each day as sections of a sprint's worth of code are completed, it needs to be tested, typically with all the other pieces of code produced by other members of the team, for its compatibility with previous sprint work, or with the

previous day's work. Without automated test suites and good configuration management, this rapidly becomes very difficult. Underestimation of resources often occurs for testing and for all verification and validation. That leads to poor testing or a lack of comprehensive testing which results in a lower quality product. Continuous integration facilitates full and complete verification and validation. Testing tools and their integration with the configuration management systems is best established early and team members should be thoroughly trained on their use.

11. **Creating and maintaining the system view of the overarching architecture outside the individual sprints is necessary for large projects.** Maintaining a system view of projects with multiple sprint teams, working toward a combined software system helps to ensure the integration is planned, and tracked and that risks and issues are properly addressed. It also helps to maintain awareness of maturity of SW development.
12. **Management and acquisition approaches need to change to allow for alternative deliverables**. This was heard at all levels from both the research and development side. A general frustration occurs on both sides when customer or acquirer management expectations and participation are not well understood in the Agile environment. For example, customers may review and comment on partial deliverables that come early in the form of prototypes rather than receiving information in standard milestone reviews. Rather than written reports, a customer may be granted access to team tools and technical information in order to obtain information directly. But first and foremost, as the customer, NASA needs to be savvy in knowing when an Agile process will work best and when a more traditional development approach may be needed as well as how to specify the best of both worlds.

6.0 Key Recommendations

6.1 Key Recommendations for Managing Agile SW Developments

Recommendations are divided into three categories, based on the organizational areas best suited to address them.

1. NASA needs to establish a guideline to determine how and when to perform an Agile software development for a project, based on the size, criticality, and complexity of the project.
   When preparing guidelines for the use of Agile, the following questions should be considered:
   a. Are there ideal types of software projects to recommend for the usage of the Agile methodology? (Consider size, complexity, criticality, type of application, maturity of organization with Agile use, etc.)

   b. Are there types of projects where is not recommended or recommended only if used with certain tailoring?
   c. Is Agile suitable for NASA safety critical projects? If so, are there additional guidelines necessary for those projects?
   d. Should there be any particular training requirements for Agile team members? For SA personnel?
   e. What types of Agile projects need SA?
   f. What changes should be made to SA techniques for Agile assurance?
   g. Are there any types of tools that should be required or recommended for use with Agile?
2. An organization's maturity and processes need to be in place before the organization tries to develop software using an Agile process. The organization's capabilities need to be defined in processes that address how a development process will be managed, performed and structured using an Agile work environment including proper training, coaching, management, and selection of personnel who can adapt to the rapid and responsive environment of Agile.
3. Ensure that entire team, their management and any SME and SA personnel are trained together on Agile benefits, processes, and chosen methodologies.
4. For a large project or program, Agile team representatives, SA and management need to agree on and document an Agile methodology, schedule, and any needed adaptations to that methodology.
5. Contract requirements for Agile developments need to be written to provide the desired information in forms and in the time frames applicable for any software development life cycle model, including Agile processes. Include in the contract expected contractor and government involvement as well as product requirements for the government SA audits, analyses, and review. Agile type deliverables and timing need to be accommodated but Agency documentation needs should also be stated.
6. Ensure teams follow the chosen Agile approach in order to get the benefits of Agile, e.g., they should hold regular Sprint Retrospectives or Scrum of Scrums to evaluate team processes, tools, and other potential improvement areas. These important meetings are needed for the health of the Agile process and management needs to make sure that these meetings occur and that SA is included in discussions.

6.2 Key Recommendations for Software/Systems Engineering for Agile SW Developments

1. Ensure that the Agile approach to be used is chosen and tailored to meet the customer's needs, such as certification requirements or hazard reports, whether the customer is a company or NASA. In other words, be sure the right Agile process for the project is determined and followed.

2. Use Agile mechanisms such as Sprint Retrospectives and Reviews to address issues and improve the Agile processes, team interactions, and product quality. This takes flexibility and a willingness to makes changes where needed.
3. Establish consistent definitions of "Done" across the project for each sprint, for any releases, and for the final product. This includes working code state, testing completeness, quality factors, documentation, etc. (see definitions, Appendix B)
4. Early in the Agile project, have the team(s) choose a tool suite and train the entire team (including SA) on the tool usage. Tools particularly important for Agile projects are:
    a. Continuous test integration and management tools
    b. Configuration management tools
    c. Tools to support capture and reporting of backlogs, stories, tasks, activities, meeting decisions, general management of processes, etc.
5. For larger, multi-sprint team projects, it is important to establish several things up front. All teams need to be part of the coordinated decisions for establishing how overarching system components or activities will function. At a minimum address:
    a. Fault Management strategies
    b. Continual integrated testing
    c. Overarching design and requirements
    d. Safety and reliability analyses
    e. Configuration management
    f. Sprint lengths and coordination
    g. Establish a Scrum of Scrums for coordination
    h. Verification and Validation approaches
    i. Coordination and completion steps and criteria
    j. Each sprint team needs to address these issues, but for success on the larger project, the multiple sprint teams need to coordinate to integrate the whole project.
6. For large projects, the structure and management of verification and validation (V&V) needs to have an overarching systems approach. The integration of multiple sprint teams needs special attention to manage and address issues when any sprint teams are falling behind on testing and other forms of V&V. This is necessary to maintain the quality of the software and the overall V&V process, ensuring a systematic approach to integration of the Agile teams' continual outputs.

6.3 Key Recommendations for SW Assurance for Agile SW Developments

1. SA Staffing:
    a. Staff the SA on a project at the appropriate levels to allow SA to be embedded in the development teams. Either independent SA or members of the Agile team need to perform the SA activities. When

SW developers, or other team members perform the SA activities, they need to establish periodic checks (audits) from the independent SA.

b. When SA cannot be embeded in all the daily sprint team activities, leaving SA personnel to cover several teams, each SA should cover no more than 2 to 4 sprint teams and even then they need to work with the sprint teams to stagger their important meetings so SA can attend.

   NOTE: However, with multiple simultaneous sprints, often the Sprint Review Meetings and Backlog Review Meetings will all occur at about the same time so a single SA member could only cover one team per sprint.

2. When independent SA are set up as SMEs that are part of the greater project, they are viewed as part of the teams and a shared resource, rather than outsiders slowing the progress of teams.
3. Regardless of who is providing Software Assurance, the SA personnel still need to report independently to project management and SMA management and is responsible for bringing up safety or risk issues. SA support can be provided by SA personnel embedded in each of the teams, by SA personnel serving as a general SME resource working with 2 to teams, or by team personnel performing SA activities.
4. Software Assurance processes used for Agile projects need to be flexible to react to the changing Agile environments on projects.
5. Recommended activities for Software Assurance personnel include:
   a. Attendance at Sprint Retrospectives, Sprint Reviews, and Scrum of Scrums
   b. Involvement in sprint planning, definition of “Done”, and prioritization of backlogs
   c. Help determine quality, reliability, and safety requirements
   d. Assuring process compliance
   e. Verification that completed sprint products meet the definition of “Done” determined by the team and project.
   f. Monitoring of other important activities such as backlog metrics, velocity metrics, other metrics, and general team health and progress.
   g. Audits of the tool usage to understand if and how the tools are being utilized and when and why they may need to change in the project.
6. Tailor software assurance processes to include how to assure requirements, design, and V&V in an Agile environment. Requirements, design, and V&V may come in different forms (backlogs, stories, models, tasks, test scripts, etc.) and are iterative, building to a completed Agile

project. This may require a different approach or tailored processes but basically, the same types of analyses are needed.

7. Software assurance personnel need to be trained to understand the different methods of requirements expression (e.g. stories, tasks, models) being used and how they are transformed into design, code, and testing requirements.
8. Have all the SA, systems safety, and system reliability practitioners working on software and the system meet regularly to assure a coordinated safety, quality, and reliabiity approach for the software. (Include management in this.)
9. Use this report's recommendations and conduct any further research needed to create guidelines for performing SA on various size, criticality and mission type Agile projects. A new Agile tailoring section for the SA standard may also be needed.

6.4 General Observations:

The following are general observations related to the use of Agile processes in performing software development and software assurance:

1. Agile software development has many variations and no clear definition. The process is inherently dynamic, evolving to meet the unique needs of the project and the players.
2. An integration problem on complex systems using Agile is underestimating the amount of integration required. This results from the Agile practices of "Do continuous integration" and "Integrate early and often."
    a) Organizations using SAFe^{R} or some similar method generally achieved better results. But even SAFe has limitations and can be ill-used.
3. Teams had problems breaking tasks into small enough functions to implement in short sprints.
    a) Initially, they were too optimistic as to how much could be put into each sprint.
    b) It takes time and practice to get good at stories and task break-downs.
4. Verification resources
    a) Many organizations tend to underestimate the resources required to do verification. Generally, the amount of regression testing needed on the large number of releases is underestimated.
5. Ensure that cadence of Agile is not detracting from assurance tasks (fast pace often is overwhelming to start with)
6. Conducting IV&V and SA is not a static practice, but is adaptable to the context and scoping of the project and challenges faced

7. Many of recommendations were to embed SA into the development process teams to more effectively execute their SA functions. However, the allocated SA resources generally do not support this level of involvement.
8. Generally, projects had limited quality metrics. This could be an important role for SA to perform on each team.
9. Mixing Agile and Waterfall management
    a) The mixture makes it hard to measure progress against project milestones. To assess progress, you need to measure the progress on team and project and sub-project backlogs and note the number of issues and task put back on the backlog that are added with each sprint or release.
10. Expect challenges if management isn't on board with Agile methodology, if customer asks for more than is in the SOW or if scope creep occurs, and with communication in general.
11. Challenges faced with Agile processes are often the same as traditional problems: introduction of new requirements, scope adjustments or refinement of understanding during development, documentation expectations, architectural decisions, dependencies, integration issues, coordination between teams, quality of testing, coverage of off-nominal scenarios.
12. It takes at least at least a year to figure out cadence. Continuously meeting the challenges will bring pride of team success and increasing confidence. That supports improved team spirit and results. Increasing automation will improve coordination and time pressures. Trickle down ways to make the processes easier. The biggest challenge is to deliver what was promised, and stay on cadence as they receive products late, experience time crunches, the sting gets felt downstream.
13. Human factors issues may be real deal-breakers and can sabotage a project if team members don't get along. If there is an underlying personality problem, you can't "put lipstick on a pig". Adjust the team roles and even personnel if needed.
14. Do not be surprised to find productivity decreasing with time. This is often due to issues that rise and need to be fixed, taking away from "new" development time. Time to fix issues and bugs and take care of change requests should be calculated in. Integration also takes a significant amount of time and should be included in the planning.

7.0 Conclusions

This Benchmarking effort, including all the research that went into it as well as the rich source of experience from the interviewees, has provided a wealth of information. Not all the great suggestions and ideas can possibly be brought forth in this one report and forward work needs to done. It is the conclusion of this team that a NASA guidebook and eventual standard or policy be put in place addressing acceptance of Agile on projects and the minimal conditions needed to meet a NASA approach to conducting Agile on our many kinds of projects.

Due the safety aspects of many of our projects and the size and complexity of many projects, criteria for when Agile should and should not be used and how it should be used, including an integrated waterfall- Agile process needs to be laid out. While Safety, size and or complexity is not an actual barrier to Agile use, it does pose some possible modifications, additional processes, and limitations.

Based on our observations during this study, it appears that Agile works well with the smaller projects, particularly in areas where considerable customer interaction is needed to help determine or finalize early requirements. Based on our sample of large projects with safety critical software, it is likely future such projects that use Agile would probably use many of the Waterfall-like features, such as more documentation, development of an overall design, reviews of overall design, stc. These additional features would assist with tasks like hazard analyses and maintenance.

We also propose further studies into areas such as: anticipated future process, as well as gaps in the current NASA requirements, resources, and capabilities.

Appendix A – Charter and Announcement Letter

National Aeronautics and Space Administration
Headquarters
Washington, DC 20546-0001

December 16, 2015

Reply to Attn of: Office of Safety and Mission Assurance

TO: Distribution

FROM: Chief, Safety and Mission Assurance

SUBJECT: Office of Safety and Mission Assurance (OSMA) Agile Benchmarking Team

With changes in the way software is developed and delivered, the Agency's Safety and Mission Assurance (SMA) community needs to better understand the best practices for the safety, reliability and quality of Agile iterative development approaches. With the increasing use of Agile programming, the strategies for how to best incorporate Software (SW) Assurance into the process needs to be better understood. We need to find better ways to provide quick and accurate risk evaluations to developers and managers. This will support risk informed designs that incorporate needed hazard controls and mitigations, as well as reliable fault management. An analysis of the relationship safety, reliability, and the Agile/iterative SW development process needs to be conducted to find ways to assure the safety of testing approaches. Also, it has become apparent that reporting on Agile SW development progress at major system milestones is not well understood or implemented. We need to take the best of our Center and industry practices and make them available to those who will be doing Agile/Iterative SW development work on NASA projects.

To this end, I am establishing a team to evaluate our current capabilities. This team will perform a benchmarking study and develop a set of recommendations to ensure NASA maintains the safety of its facilities, assets, personnel, and the public while supporting advances in software development technology. They will provide needed guidance in assuring the safety and reliability of critical software while using Agile/iterative SW development methods and may propose additions and/or modifications to contractual, procedural and product requirements. Any suggested guidance will be incorporated into the NASA Software Assurance Handbook, now being developed.

The NASA Independent Verification and Validation Facility's SMA Support Office will support this effort; however, any interested Center SMA or SW engineering directorates are welcome to provide candidates to serve on this team or as interviewees. The team may ask for additional expertise from the Centers as well as from the Office of the Chief Engineer, the

NASA Engineering and Safety Center, the NASA Safety Center, or others to support specific inquiries or studies. The team's duties and products are outlined in the enclosed charter.

I believe this effort is very important to NASA's mission and will provide valuable insight into the best practices of industry and government leading to better SMA support of our changing software technology and development practices. Please provide your assistance and any candidates to work with the Agile Benchmarking team to my lead in this effort, Ms. Martha Wetherholt who may be reached at (202) 358-1245 or by e-mail at martha.wetherholt@nasa.gov.

Thank you for your support.

Terrence W. Wilcutt

Enclosure

Distribution:
Chief Health and Medical Officer/Dr. Williams
ARC/Mr. Liu
AFRC/Mr. Smolka
GRC/Ms. Liang
GSFC/Ms. Bruner
IV&V/Mr. Blaney
JPL/Ms. Chodas
JSC/Mr. McArthur
KSC/Mr. DeLoach
LaRC/Mr. Watson
MSFC/Mr. Cash
SSC/Mr. Douglas
NSC/Mr. Phillips

cc: Office of Safety and Mission Assurance/Dr.Groen
/Dr. Dezfuli
/Mr. Hughitt
/Ms. Wetherholt

Chief Engineer/Mr. Roe
General Counsel/Ms. Thompson-King
Associate Administrator for Aeronautics Research Mission Directorate/Dr. Shin
Associate Administrator for Human Exploration and Operations Mission Directorate/ Mr. Gerstenmaier
Associate Administrator for Science Mission Directorate/Dr. Grunsfeld
Associate Administrator for Space Technology Mission Directorate/Mr. Jurczyk

MSFC/Dr. Safie
NESC/Mr. Wilson
NESC/Mr. Aguilar
/Mr. Crumbley
GRC/Ms. Maynard-Nelson

National Aeronautics and Space Administration
Charter of the
Office of Safety and Mission Assurance
Agile Software Assurance Benchmarking Team

1.0 Purpose

This charter sets forth the authority for, and the duties, procedures, organization and support of, the Office of Safety and Mission Assurance (OSMA) Agile Software (SW) Assurance Benchmarking Team.

2.0 Authority

The OSMA Agile SW Assurance Benchmarking Team is established in accordance with the requirements set forth in NPD 1000.3, The NASA Organization dated April 15, 2015.

3.0 Duties

The OSMA Agile SW Assurance Benchmarking Team shall:

a. Research current literature on the topic both in preparation for the benchmarking and to gather and distill knowledge from recognized experts.

b. Benchmark approaches and best practices to assuring the quality, safety and reliability of critical software developed under an Agile or iterative process. NASA's Safety and Mission Assurance (SMA) community must understand the advantages as well as the possible risks of using the Agile/iterative process.

c. Benchmark NASA methods in this area as well as other government agencies and industry (e.g., Department of Defense, Federal Aviation Administration, auto industry, aviation industry, and Defense Advanced Research Projects Agency) and seek out strategies for incorporation of Safety, Reliability, and Quality Assurance (SR&QA) into the Agile/iterative SW development processes. This may include:

- assessment and evaluation of risks associated with various Agile/iterative approaches
- level of SR&QA oversight provided (or not)
- SW and Software Assurance participation in major Milestone reviews and acceptance reviews
- process for verification and validation of requirements
- risk process for Agile/Iterative development including how risks are recorded, tracked, and accepted
- Hazard analyses processes and how the SW and System Hazard processes work together
- SW reliability methods used to assure the software provides the needed fault and failure tolerance

d. Consider any differences in approaches that may be needed for specific areas; e.g., aeronautics, ground operations, robotics, modeling and simulation, space flight, and human space flight.

e. Provide input to guidance and, if warranted, recommend changes to standards and policy which helps to build safety, quality, and reliability into new technology development without impeding progress. Including insight into verification, acceptance and status reporting at major Milestone reviews and Key Decision Points (KDPs).

f. Identify possible opportunity areas for future SA research including tools, training, development and process improvement.

g. Compare relevant NASA and industry workforce capabilities.

4.0 Procedures

The OSMA Agile SW Assurance Benchmarking Team shall report to and function in an advisory capacity to the Chief, Safety and Mission Assurance.

To gather additional information and discuss strategies, approaches, ideas, the OSMA Agile SW Assurance Benchmarking Team shall hold face-to-face meetings, telecon/Web-exs, and reviews. They will also form sub-teams of no less than 2 and no more than 5 to interview benchmark organizations. After each interview, the team will examine the evidence, record it in the database, and discuss any needed improvements in the interview approach. Sometime will be spent reviewing existing NASA requirements, processes, and techniques in order to understand requirements and lessons learned and to explore new methods and techniques. One or more technical interchange meetings may be held to garner community inputs and provide insight to the team's activities.

5.0 Products

OSMA Agile SW Assurance Benchmarking Team Report and Recommendations. The team will develop a report that:

- Documents the findings of the benchmarking studies and highlights the current process, anticipated future process, and gaps in current NASA requirements, resources, and capability.
- Includes recommendations for SR&QA approaches to meet anticipated needs including content on:
    i. Minimum SMA policy and requirements
    ii. Various insight and oversight stratagems
    iii. Possible variations based on type of Agile/iterative method

iv. Possible NASA SMA products, processes, and services to needed to support safety, reliability, quality and testing, verification, and validation of Agile/iterative SW processes.
v. How the technical authority waiver process will work.

- Includes recommendations for OSMA actions to be taken to ensure adequate SR&QA acquisition and implementation when contractors are used.
- Provides suggested ways to work with and support various types of efforts, providing Software Assurance and SW development management with methods for, or the services to reach risk informed design decisions in a quick and timely manner.
- Provides suggested approaches to build in sufficient safety, quality, and reliability to SW Agile/iterative development efforts and products as part of the requirements, design, and general approach.
- Includes recommendations for future OSMA research and technology development.

6.0 Organization

The OSMA Agile SW Assurance Benchmarking Team will consist of a Chairperson, the SMA Software Assurance Technical Fellow, and representatives from the following organizations:

Office of Safety and Mission Assurance
Office of the Chief Engineer
NASA IV&V SMA Support Office

Support for review and advice is requested on an as-needed basis from:
Office of the Chief Engineer
NASA Engineering and Safety Center
NASA Safety Center
Office of General Counsel
Aeronautics Research Mission Directorate
Space Operations Mission Directorate
Science Mission Directorate
HEO Mission Directorate

7.0 Support

The OSMA Agile SW Assurance Benchmarking Team shall call upon organizational support, additional expertise, or establish working groups as needed or as instructed by the Chairperson.

8.0 Charter Renewal

The OSMA Agile SW Assurance Benchmarking Team shall terminate within 6 months from the date of this charter, unless renewed by the Chief, Safety and Mission Assurance.

Terrence W. Wilcutt
Chief, Safety and Mission Assurance

Appendix B – Definitions of terms

Agile software development - is a set of principles for software development in which requirements and solutions evolve through collaboration between self-organizing, [1] cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change. [2] Source: Wikipedia [1] Collier, Ken W. (2011). *Agile Analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing*. Pearson Education. pp. 121, "What is a self-organizing team?" [2] "What is Agile Software Development?" Agile Alliance. 8 June 2013. Retrieved 4 April 2015.] (See Appendix I of this document for a diagram of Agile software development)

Backlog - a prioritized features list, containing short descriptions of all functionality desired in the product. [Source: www.mountaingoat.software.com]

Configuration Management (of software) – tasks needed to establish and maintain the integrity of software products throughout their lifecycle. Key tasks include managing changes to the products and maintaining baselines. [Source: adapted from Wikipedia, downloaded May 16, 2016]

Done – point at which the requirement is complete. (Adapted from IEEE) The definition of done establishes what must be true of each product backlog item for that item to be done. Example of definition might include the following activities: Code is well written; Code the checked in; Code was either pair programmed or peer reviewed; The code comes with testing at all appropriate levels; The feature the code implements has been documented in any end-user documentation such as user manuals or help systems; [Source: Multiple Levels of Done by Mike Cohn, 2/17, 2015, mountaingoatsoftware.com]

LEAN software development - is a translation of lean manufacturing principles to the software development domain. It can be summarized by seven principles: Eliminate waste (everything not value-added is waste!), Amplify learning, Decide as late as possible, Deliver as fast as possible, Empower the team, Build integrity in, and See the whole. [Source: Wikipedia, downloaded May 18, 2016]

KANBAN – is a technique for managing a software development process in a highly efficient way. By matching the amount of work in progress to the team's capacity, Kanban gives teams more flexible planning options and faster outputs. A Kanban team is only focused on the work in progress. Once the team completes a work item, they pick the next item off the top of the backlog list. The product owner can re-prioritize the work so the most important work is always on top of the backlog. [Source: Adapted from: atlassian.com, downloaded May 18, 1 2016]

Quality checks – include a variety of checks to assure that the resulting product is high quality and meets the requirements and that the product has been produced using the appropriate processes. It may also include checks to verify that the project is progressing in a satisfactory manner.

Retrospective - a meeting that's held at the end of an iteration in **Agile** software development (ASD). During the **retrospective**, the team reflects on what happened in the iteration and identifies actions for improvement going forward. Source: Wikipedia (downloaded May 16, 2016)

Scrum of Scrums – a regular scrum meeting with representatives from all the other scrum teams on a project. This technique is used to coordinate scrum teams on large projects.

Safety Checks – Safety checks are a particular type of quality check that concentrate on whether the safety requirements of a system have been corrected identified and included in the requirements, flowed correctly into the system and software design, properly implemented and tested completely.

Software Assurance -

1. The planned and systematic set of activities that ensure that software life-cycle processes and products conform to requirements, standards, and procedures. (IEEE 610.12, IEEE Standard Glossary of Software Engineering Terminology) For NASA this includes the disciplines of software quality (functions of software quality engineering, software quality assurance, and software quality control), software safety, software reliability, software security, software V&V, and IV&V.
2. "The level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its lifecycle, and that the software functions in the intended manner." [1] The main objective of software assurance is to ensure that the processes, procedures, and products used to produce and sustain the software conform to all requirements and standards specified to govern those processes, procedures, and products.[2] Source: Wikipedia

Scrum - an iterative and incremental Agile software development framework for managing product development. Source: Whatis.com (downloaded May16, 2016). Comes from a rugby term for a team huddle.

Sprint - in **Agile** software development, is a set period of time during which specific work has to be completed and made ready for review. Source: Whatis.com, (downloaded May 16, 2016)

Appendix C: Acronyms

ABT	Agile Benchmarking Team
CDR	Critical Design Review
CDRLs	Contract Deliverable Requirements Lists
CIL	Common Intermediate Language
CM	Configuration Management
CMMI	Capability Maturity Model Integration
COCOMO	Constructive Cost Model
CSCI	Computer Software Configuration Item
DevOps	Agile approach that emphasizes working with computer and HW developers as well as operators and customers
DOORS	Dynamic Object-Oriented Requirements System
Doxygen	Documentation Generator
DSDM	Dynamic Systems Development Method
FDIR	Fault Detection, Isolation and Recovery
FM	Fault Management
FTA	Fault Tree Analysis
FMEA	Failure Modes Effects Analysis
GRC	Glenn Research Center
HEO	Human Exploration and Operations
HTML	Hyper Text Markup Language
IEEE	Institute of Electrical and Electronic Engineers
IV&V	Independent Verification and Validation
KDP	Key Decision Point
KSLOC	1,000 Source Lines of Code

LSSO	
NA	Not Applicable
NASA	National Aeronautics and Space Administration
NPR	NASA Procedural Requirement
OSMA	Office of Safety and Mission Assurance
OJT	On the Job Training
PM	Project Manager
POC	Point of Contact
RFP	Request for Proposal
RTC	Rational Team Concept
RQM	Rational Quality Manager
SA	Software Assurance
SAFe	Scaled Agile Framework environment
SBM	Serena Business Management
SMA	Safety and Mission Assurance
SME	Subject Matter Expert
SQA	Software Quality Assurance
SR&QA	Software Reliability and Quality Assurance
SRS	Software Requirements Specification
STD	Standard
SW	Software
SVC	Subversion
TASC	Total Administrative Services Corporation
TFS	Team Foundation Server

UML	Unified Modeling Language
V&V	Verification and Validation
XP	Extreme Programming

Appendix D -- Bibliography for Agile Background

Sources for Agile Background	Key Takeaways	What this means for SA
Boehm, B.W., & Turner, R. (2004). *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston: Addison-Welsey.	Neither Agile nor traditional methods provide a silver bullet; each has advantages within particular applications; the trend is toward development using both agility and discipline; methods to balance the two are emerging; building up a method is preferred to tailoring it down; greatest benefits are more likely to be found in areas dealing with people, values, communication, and expectations management.	Take a risk-based approach and utilize analytic techniques to determine a way to balance discipline and agility for each particular project, whether for engineering or for assurance.
Sidky, A., & Gaafar, A. (2014). The Mindset Behind Estimating and Planning in Agile Methods. Retrieved January, 2016, from http://www.pmi.org/	Agility is the key to managing knowledge work as opposed to task work. The difference is in applying an empirical process rather than a defined process control model. During development, you cannot predict or control circumstances like human creative thought processes; knowledge work should be dealt with differently than task work; the output of knowledge work is not knowable in advance, and therefore planning and estimating should be approached in a different, adaptive manner.	"Inspect and adapt" to allow for the creativity of the human thought process to address natural unpredictability. Consider looking at size of work items versus time required, and relative versus absolute estimated time to completion for planning and estimation activities. Allow for evolution of estimates rather than commitments.
Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M....Thomas, D. (2001). Manifesto for Agile Software Development. Retrieved October, 2015, from http://agilemanifesto.org/	Value individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; responding to change over following a plan.	Take context into consideration for Agile application. The manifesto offers guidelines, or a philosophical framework, not strict rules. Every project will be unique, and necessitate critical thought in the application of Agile methodologies.

Sources for Agile Background	Key Takeaways	What this means for SA
Reference Materials for PMI Agile Certified Practitioner (PMI-ACP) Examination. Retrieved November, 2015, from http://www.pmi.org/~/media/PDF/Certifications/reference/agile-certified-reference-materials.ashx/	A resource of publications that provide accurate information regarding the Agile framework.	Learn the fundamentals of the Agile framework and how a project might benefit from its application.
Carilli, J. (2013). Transitioning to Agile. Retrieved November, 2015, from http://www.pmi.org/learning/transitioning-agile-ten-success-strategies-5841/	To transition to Agile software development method: secure management commitment; empower your team; understand the collaborative culture; embrace agile methods; develop a roadmap and individual plans; acquire an Agile coach and train the team; start small and gain early successes; establish Agile performance measures; create Agile contracts; adopt application lifecycle management tools.	Introduce or consider applying these strategies to improve likelihood for a successful transition from traditional to Agile development. Investigate new ways to assure software that aligns closely with the development efforts.
Alaverdyan, N. (2011). The Definition of Done (DoD). Retrieved November, 2015, from http://alaverdyan.com/readme/2011/01/the-definition-of-done-dod/	Having and maintaining a common understanding of the definition of done for sprints, user stories, releases, or even bug fixes ensures completeness and integrity.	Be sure the entire team has explicit understanding of what done means and is held accountable. A visible posting is helpful.
Cohn, M. (2010). Succeeding with Agile – Mike Cohn's Blog. Retrieved November, 2015, from http://www.mountaingoatsoftware.com/articles/	Topics include: patterns of Agile adoption; writing the product backlog just in time and just enough; the chivalrous team member; how to fail with Agile; rolling look ahead planning; the art of compromise; comparative agility assessment - determining how Agile you are comparatively; Agile teamwork; the roles of the project management office in Scrum; choosing to start small or go all in when adopting Agile.	Embed SA within the team, if possible, to minimize handoffs between specialists. Communicate frequently and in person, when possible. Embrace the concept of whole-team responsibility and commitment to deliver quality, working software at the end of each sprint.

Sources for Agile Background	Key Takeaways	What this means for SA
Sims, C., & Johnson, H. L. (2011). *The Elements of Scrum*. Foster City, CA: Dymaxicon.	Sections include: the development of Agile methodologies as a reaction to the traditional waterfall method for software development; a primer on scrum; and additional enabling practices for an Agile work environment.	Get certified in Scrum or the methodology that the project is utilizing so that the process is familiar. Inform the team of practical advice, such as test as you go, not at the end; deliver product early and often, to demonstrate working software to the customer and garner feedback; document as you go, as needed or required; build cross-functional teams to break down silos.
Burba, D., (2013). Policing the Agile Expressway. Retrieved November, 2015, from http://www.pmi.org/learning/policing-agile-project-team-4035/	A skepticism of Agile exists, that Agile means anarchy or sloppiness when it comes to software development. Concerns include the following perceptions: lack of up-front planning; loss of management control; lack of predictability; and lack of engineering discipline.	Alleviate concerns with a disciplined approach to listening to the stakeholders at every delivered iteration and ensuring the team is applying feedback; identifying root causes of problems and troubleshooting; and ensuring user stories are not ambiguous and well-understood by the team.
Archer, S., & Kaufman, C. (2013). Accelerating Outcomes With A Hybrid Approach Within A Waterfall Environment. Retrieved November, 2015, from http://www.pmi.org/learning/outcomes-hybrid-approach-waterfall-environment-5839	There are benefits to waterfall, benefits to Agile, and benefits to hybrid approaches to software development. Using a hybrid approach may allow the project team and stakeholders to see the realization of the big picture throughout the project instead of at the end.	If the organizational implications of adopting Agile are not realizable, then consider adopting a hybrid approach to gain some of the benefits of Agile techniques: pre-defining requirements sets the stage for upcoming sprints; frequent interaction with the customer results in direct feedback and acceptance of implemented functionality; testing is conducted alongside development during sprints and not delayed until the end; risk is reduced by enabling visibility into the quality of the product during the development.

Sources for Agile Background	Key Takeaways	What this means for SA
The 9th Annual State of Agile™ Survey (2015). Retrieved February, 2016, from http://info.versionone.com/state-of-Agile-development-survey-ninth.html/	With the continued momentum of Agile, the top three benefits include: the ability to manage changing priorities; team productivity; and project visibility. The majority of projects are deemed successful, tracking velocity, iteration burndown, and release burndown and measuring on-time delivery, product quality, and customer/user satisfaction metrics.	Collect metrics that are meaningful to provide evidence of assurance activities that reflect project progress and measurements of quality, reliability, safety, security, and success.
Zaleski, P., Bostian, C., & McDyer, C. (2011). Balancing Agility with Conformance On Complex Government Programs. Retrieved January, 2016, from http://www.pmi.org/learning/balancing-agility-conformance-government-programs-6123/	There is no "one-size-fits-all" way to implement Agile methods, and applying Agile methods should be complimentary to core principles, not a replacement. Within the context of conformance requirements and contract deliverables, iterative and lightweight concepts intended on reducing exposure to risk and increasing success probability may be implemented. A cultural change, beginning with education and small successes may be leveraged and lead to the ability to adapt to inherently uncertain and complex environments.	Investigate ways that a combination of Agile methods and traditional methods within the organizational structure may bring forth the benefits of both within the given constraints.
ASPE SDLC Training, *Certified Scrum Master Workshop* (2015).	Officially recognized resource for the Scrum Alliance Scrum Master certification program.	Achieve certification status for improved understanding and credibility within Agile teams.
Farley, J. (2005). Leadership In Agile Projects – What Makes for Success? Retrieved October, 2015, from http://www.pmi.org/learning/leadership-agile-projects-makes-success-7592/	Environments favorable to Agile projects are organizations that are receptive to change, have a learning culture, are forgiving to mistakes, foster collaborative partnering across skill sets, and above all are flexible and adaptive. Leadership, in this context, provides the vision, enables risk mitigation, removes roadblocks, and promotes communication with an adaptable style that fits the situation.	Attend checkpoints regularly. Assure that risks are well-defined and escalation paths are clear and easily accessible for timely mitigation.
The 2015 State of Scrum Report (2015). Retrieved February, 2016, from https://www.scrumalliance.org/landing-pages/2015-state-of-scrum-report-download/	Scrum popularity is increasing, fulfilling customer needs, meeting business goals, and improving the quality of work life.	Assure that proper training has been provided for the team to follow the Scrum framework.

Sources for Agile Background	Key Takeaways	What this means for SA
Bell, S. (2013). Agile is great but don't bet lives on it, says founder. Retrieved November, 2015, from http://www.computerworld.co.nz/article/457007/agile_great_don_t_bet_lives_it_says_founder/	When reliability is more important than flexibility, in highly-critical, human-rated missions, it's best to utilize traditional methods.	Plan on providing more rigorous assurance for safety and quality throughout the mission.
Lapham, M. (2015). Agile in Government: Validating Success Enablers and Inhibitors. Retrieved January, 2016, from https://resources.sei.cmu.edu/asset_files/Presentation/2015_017_001_446272.pdf/	Adopting Agile requires a change in culture, inclusive of new strategy, structure, procedures, and skills. The Agile Continent illustrates challenges and pitfalls that are commonly faced.	Learn as much background about Agile methodology strengths and weaknesses and be aware of challenges that may arise within the context of the project.
Lapham, M. (2012). Agile Methods: Tools, Techniques, and Practices for the DoD Community. Retrieved January, 2016, from https://resources.sei.cmu.edu/asset_files/Webinar/2012_018_101_24363.pdf/	A comparison between traditional incremental delivery and agile methods points out transitional changes that most often need to occur in organizational structure, leadership style, staffing and rewards systems, and communications and decision making models. Overcoming these barriers may bring the benefits of being responsive to inevitable changes faster than with traditional methods.	Recognize ways in which traditional cultural elements may be barriers to successful implementation of Agile methodology, and be prepared to contribute suggestions to adapt current processes.
Smith, J. & Menzies, T. (2002). Should NASA Embrace Agile Processes? Retrieved November, 2015, from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.473.6590&rep=rep1&type=pdf/	Pair programming, a practice within the Agile framework called extreme programming, has limited applicability for success, and specifically is not recommended for high risk projects.	Assure processes that claim increased or more cost-effective productivity or lower error rates are validated and not simply assumed.

Sources for Agile Background	Key Takeaways	What this means for SA
Ballard, M. (2013). DWP Drops Agile from Flagship Government Software Project. Retrieved February, 2016, from http://www.computerweekly.com/blogs/public-sector/2013/05/dwp-drops-agile-from-flagship.html/	The Department for Work and Pensions (UK) reversed its decision to use Agile software development methodology for Universal Credit, a large, complex financial project, reporting that Agile and waterfall may be appropriate at different times. It is unclear whether the project was ever fully agile, but may have used the Agile claim as a buffer, or a political fireguard.	Be cautious about project claims of being Agile, and don't assume true Agile processes are in place without evidence.
DeWitt, D. (2015). Estimate Agile Projects and Improve Success. Retrieved November, 2016, from http://galorath.com/	An estimation model removes many ambiguities and uncertainties by providing greater insight into the likeliness of achieving project cost and schedule.	Apply proven estimation methods to Agile projects to help with release planning, deliver backlog indicators, and help understand impact of changes. Employ meaningful metrics and tools, whenever possible.
Stacey, R.D. (1996). *Complexity and Creativity in Organizations*. Berrett-Koehler.	Discusses connections between complexity science and the world of organizations and innovative change. Source of project complexity diagram.	Recognize the distinction between environments in which a defined process may be used, and those that are better suited for empirical processes available within the Agile framework.

Sources for Agile Background	Key Takeaways	What this means for SA
Measey, P. (2015). *Agile Foundation: Principles, Practices and Frameworks*. BCS Learning and Development Ltd.	Comprehensive introduction to Agile principles and methodologies, including discussion of myths: The agile framework is inherently about enabling inspection and adaptation in dynamic environments, and is not new. It requires a cultural change to implement, however, which may be difficult within NASA. The transition will probably entail a learning curve before benefits are seen, and is best done with experienced leaders. Agile doesn't mean "no" planning or documentation, just focused, value-driven artifacts that enable understanding to support development and maintain the system, often with frequent and evolutionary planning. Agile software development isn't hacking code together with little thought or design, but actually provides techniques for the team to produce high quality code and continuous assurance with properly applied practices.	Become aware of myths that may be circulating on the project, and focus assurance on the leading concept of Agile, which is Kaizen (continuous improvement).

Appendix E – Discussions of Performing IV&V on Agile Systems

From IV&V related tech discussions at the NASA IV&V Facility:

1. IEEE Standard for Software Verification and Validation Working Group is engaged in surveys and discussions with representatives familiar with Agile applications in order to revise the 1012 Standard for System, Software and Hardware Verification and Validation
2. The Agile Manifesto is not deterministic, but showcases values or principles, rather than “how-to”
3. The working group asked three questions: What is Agile, How should we apply IV&V, and What systems are good candidates for the Agile framework
4. Working within an Agile framework enables projects to keep up with evolving hardware changes
5. From a high level, Agile projects may look very similar to waterfall, and some insist that there are no issues with maintaining the same processes that are currently in place; “no problem exists”
6. Some projects struggle with the gap that exists between traditional approaches that focus on requirements and those that focus on user stories and features; when the project tries to maintain both perspectives, a translation is necessary to bridge the gap from one to the other
7. Proponents of Agile cite a sense of “trust” in the highly capable, cross-functional development teams to do the “right” thing because of the continuous iterations and pride in ownership, which is foreign to the IV&V culture of requiring rigorous evidence for software assurance, especially in safety-critical environments
8. A challenge exists in the understanding of the complete, integrated system
9. Keeping up with the fast pace of Agile development is difficult and leaves less time for analysis, bringing forth risk in a less rigorous approach
10. Agile approach in the pure sense is different than an Agile (lowercase a) approach that is incremental/iterative or hybridized in some manner
11. Monitoring volatility of a project may be an insightful metric for SA or IV&V; providing assurance piecemeal, where the system appears stable, may give an illusion of confidence that is not applicable to the system as a whole
12. When there is a great deal of churn, there is a lot of wasted effort on analysis that gets thrown out
13. Getting test environment/tools up and running is challenging, but crucial to have in place early
14. Agile is not a “silver bullet” that solves the cost/schedule/performance problem
15. Tying characteristics of Agile projects to potential SA or IV&V strategies may help accomplish a more thorough analysis, keeping open to adaptability in order to optimize the value that can be provided; different tactics to take for different pressures may bring forth recommended SA approaches
16. Providing analysis in phase with the software development team is less important than adding value

17. Teams in Agile environments need the benefit of IV&V or SA in the traditional sense
18. Agile is not utilized in nuclear industry software development to date due to five major factors:
    a. Maintenance – extensive documentation is typically required; the risk of system defects causing a shut down for repairs, or worse, is highly undesirable
    b. Governance – regulations are stringent for reliability, and the industry is very risk averse
    c. Safety – failure could be catastrophic, including loss of life or birth defects due to radiation poisoning or long term environmental impacts
    d. Development Process – conservative models favored over more flexible open-ended schedule models that do not have historical data to support expectations
    e. Operations and Cost – a shutdown to remedy any system errors would incur huge costs
19. The greater the complexity of the project, the more difficulties will be faced with the Agile approach
20. Developers should plan on and incorporate separate Builds just for bug fixes
21. IV&V culture, tech framework, processes, tools, and workforce must be adaptive; flexible enough to make best use of available information when and where it is available
22. Continue to develop and leverage thorough understanding of acquirer and system needs and integrating system; Be more system-oriented
23. Continue IV&V emphasis on assuring the software will work vs. process conformance
24. Continue IV&V emphasis on risk assessment and risk reduction
25. Increase IV&V emphasis on independent testing
26. Increase IV&V analysts' expertise – e.g. application-specific SMEs
27. Develop efficiency improvements to shorten turn-around time on analysis, assurance, and reporting of defects and risks
28. Promote development of NASA requirements, guidelines, and best practices for development
29. From August 21, 2014 blog from Enterprise Knowledge website: http://www.enterprise-knowledge.com/ivv-for-agile-projects/ the following best practices for IV&V of Agile development practices:
    a. Identify the control points
    b. Review the backlog
    c. Manage the roadmap
    d. Question business value
    e. Validate consistent practices
30. Get involved early in development, build working relationship with developer, and disseminate best practices and information regarding the value of quality and cost and schedule savings of best practices
31. Understand what is being produced and how system needs are being allocated to release backlogs/sprints

   a. Ensure key and driving requirements and defect and risk resolution/mitigation are given appropriate priority
   b. Consider on-site participation to leverage development collaboration, including collaboration with stakeholders
32. Determine when to begin critical assessments, e.g., IV&V in every sprint and of every release may not be warranted
   a. Consider “real-time” IV&V during key sprints
   b. Use non-critical development periods for IV&V planning, evidence consolidation, and documentation/reporting
33. Develop interface models and apply code checking against the models
34. Implement efficient regression testing and change impact analysis capabilities for efficient and effective support to sprints

From NASA IV&V Facility Personnel Performing IV&V on Agile Systems:

1. Agile software development has many variations and no clear definition; the process is inherently dynamic, evolving to meet the unique needs of the project and the players
2. Conducting IV&V or SA is not a static practice, but is adaptable to the context and scoping of the project and challenges faced
3. Many Agile principles are recognizable as generally good development practices that are undeniable
4. Agile development does not eliminate any tasks or necessary artifacts to conduct SA or IV&V, but may impact what is deemed acceptable
5. Agile development places new stresses on SA or IV&V activities, since the responsibility to assure Class A missions is not altered, nor has tolerance for a system failure or mishap changed
6. Existing project planning templates based on parametric models cannot be applied “as is” to Agile development; existing models presume model factors are available and matured while Agile is not delivering similar model factors; top down and bottom up labor estimation models are not successful
7. Software development conducted in a series of mini waterfalls may undergo assurance that is analogous to traditional SA or IV&V
8. Risks are associated with tasking cost (to support higher frequency of deliverables), with skills cost (providing highly skilled, diversified analysts), and with maintaining independence over integration
9. Information/artifacts to perform SA or IV&V is not always forthcoming from a cooperative team
10. Lack of synchronized design and implementation repositories make independent evaluation of deliveries less achievable
11. Rapid turnaround review processes may actually demand more dependence on processes and tools

12. Assuming a sustainable pace for timely analysis remains a challenge, and "completeness" criteria from one development phase to the next no longer has meaning in an Agile environment where completeness is not accomplished until the final software release; multiple intermediate stages of "Done" keep moving the target out from under any rigorous analysis that might otherwise be performed

Appendix F– Agile Benchmarking Questionnaire

1. Organization's Software Development Background
    a. Describe your organization/scope for this discussion
    b. Highlight your organization's software development products
    c. What development processes are used by your organization
    d. When and why did you transition to Agile
    e. How would you rate your level of experience using Agile
    f. Overall, have you been satisfied with Agile projects
2. Software Produced by your Organization
    a. Quantify amount
    b. What amount/percentage is safety critical
    c. What amount/percentage is developed using Agile
    d. What amount/percentage developed using Agile is safety critical
    e. What amount/percentage of safety critical is developed using Agile
3. Specific (Typical) System Produced using Agile Approach - Case Study
    a. Provide brief project overview
    b. What is the size of the project
    c. What is the planned longevity for system operation
    d. Will operation/maintenance be managed by the developing organization
    e. What is the number and length of sprints
    f. Is Agile used for the entire software development effort
    g. Is Agile used beyond software development
    h. Specify/describe Agile approach used
        1. SCRUM
        2. Crystal
        3. Lean
        4. Kanban -process-management system that tells what to produce, when to produce it, and how much to produce - inspired by the Toyota Production System[1] and by Lean manufacturing.[2]
        5. Extreme Programming (XP)
        6. Dynamic Systems Development Method (DSDM)
        7. Feature Driven Development
        8. Integrated with traditional waterfall
        9. Hybrid of one of the above
        10. Not named
        11. Type previously not listed
    i. Is a pure approach taken or an adaptation to your organizational setting

1. Modifications to your approach
2. Rationale for decisions
3. Any workflow/team-reorganization during the process

j. What difficulties have been faced/overcome

4. Agile Software Team Performance
   a. What is the make-up of the team(s)
      1. Number of teams, number of team members
      2. Functional roles of team members
      3. Co-located or distributed
   b. How does the team communicate with the product owner or customer/user
   c. How does the team collaborate with other teams/functions within the project/system
   d. How does the team integrate with interactive systems
   e. How does the team manage access to hardware or test labs
   f. Is the Agile Manifesto followed strictly or loosely
      1. Individuals and Interactions over Processes and Tools
         1. How are complexities with interactions dealt with
         2. Any specific tools useful in supporting the Agile process
         3. Are processes flexible
      2. Working Software over Comprehensive Documentation
         1. How are software complexities dealt with (large scale, reuse, multiple/distributed developers, asynchronous delivery schedules, integration of separate development efforts, integration of features/capabilities across sprints)
         2. What is considered “working” software
         3. How are continuous software deliveries received
         4. Is documentation reflected in artifacts other than word/excel files
         5. How is CM of documentation done
         6. Are all decisions formally captured
         7. How is support of project milestones (e.g., CDR) impacted by limited documentation?
      3. Customer Collaboration over Contract Negotiation
         1. How available is the customer to the Agile team
         2. Are “proxy” customers utilized
         3. Have issues risen in contract alignment with deliverables
      4. Responding to Change over Following a Plan
         1. How are baseline requirements defined
         2. How smooth is the response to change
         3. How much planning is involved
5. Provision of Software Quality Assurance
   a. Is SQA integrated as part of the team
   b. Is SQA performed by an independent reviewer
      1. Participation at Sprint Reviews

2. Participation at other project reviews/milestones
3. Frequency

c. Is SQA not explicitly involved
    1. Replaced by something else
d. Rationale for approach taken
e. How is assurance conducted and results captured
    1. What methods of software assurance are used
    2. How are software assurance concerns identified and tracked
    3. When during the development process are concerns identified
    4. How are software assurance concerns controlled/mitigated or accepted
    5. How are software assurance concerns verified and validated
    6. How are software assurance strategies, objectives, and assumptions documented
    7. How are adverse conditions identified and addressed

6. Implementation and Assessment of Software Safety
    a. Is system/software safety integrated as part of the team
    b. Is there an independent system/software safety team or effort
        1. Participation at Sprint Reviews
        2. Participation at other project reviews/milestones
        3. Frequency
    c. Is system/software safety not explicitly involved
        1. Replaced by something else
    d. Rationale for approach taken
    e. How is system/software safety (esp. for safety critical software) assessed and results captured
        1. Delegation of responsibility for software safety
        2. Litmus test
        3. Standards compliance
    f. How is safety assessment conducted and results captured
        1. What methods of safety analysis are used
            1. Hazards Analysis
            2. Fault Tree Analysis
            3. Failure Modes and Effects Analysis
            4. Other
        2. How are software safety concerns identified and tracked
        3. When during the development process are concerns identified
        4. How are software safety concerns controlled/mitigated or accepted
        5. How are software safety concerns verified and validated
        6. How are safety “must work” and “must not work” functions tested and assured
        7. How are off-nominal, boundary conditions, race conditions, etc. tested and documented
7. Assessment of Software Reliability

   a. Is software reliability incorporated into the Agile approach
      1. How/when is software reliability addressed
      2. How/when is software reliability assessed
   b. Is software reliability not explicitly assessed
      1. Replaced by something else
   c. Rationale for approach taken
   d. What methods are used to ensure software reliability
      1. Are functional analyses performed to determine weak areas
      2. Are analyses used (FMEA, FTA, other) to inform the design
      3. How are analyses fed into the Agile process
   e. What assessment methods are in place
   f. Are metrics kept to measure the overall health and weaknesses of the software
   g. Are there statistical testing methods used to determine reliability
8. Verification and validation of the integrated system developed by Agile processes
   a. What methods are used for V&V
   b. What challenges have been faced due to the Agile approach
9. Independent verification and validation of Software
   a. Is IV&V being performed by a financially, managerially, and technically separate group
   b. What methods are used for IV&V
   c. What challenges have been faced due to the Agile approach
      1. How are frequent changes kept up with
      2. How is limited documentation handled
      3. How is completeness assessed
      4. How is communication enabled
      5. How is analysis kept in-phase
      6. How are issues raised and addressed
      7. How are risks identified and mitigated
10. Contractor-Provided Critical Software Developed with an Agile Approach
   a. How does an outside developer participate as a stakeholder
   b. Are there lessons learned for contracts
   c. Are there lessons learned for oversight and insight
   d. As the acquirer or holder of the contract, what steps are taken for assuring the final product is safe and reliable
11. Cyber Security Concerns
   a. How is Cyber Security addressed
   b. Are there any unique requirements due to the Agile approach
12. Results/Conclusions from use of Agile on the Case Study Project
   a. What value was provided due to the Agile approach
   b. What challenges did you find due to the Agile approach
   c. What lessons learned related to the quality assurance, safety and reliability efforts resulted due to the Agile approach

Appendix G: Organizations' Backgrounds and Agile Software Processes

Organizations	Type of Organization	Size of SW Systems	SW Safety Critical?	SW Complexity	Agile Experience	Agile Processes	Sprint Team Information
NASA 1	Management/ Oversight of SW Development (See NASA 3)	Large	Yes - 100%?	High	High	SCRUM SAFe	2 week sprints Varied # members
NASA 2	SW Development	Large	No	High	Medium	Scrum	2 week sprints 10 members
NASA 3	SW SQA (See NASA 1)	Large	Yes -100%	High	Medium	SCRUM & Waterfall	2 week sprints Varied # members
NASA 4	SW Engineering and Development in support of agency	Large	Yes – Large %	High	High	SCRUM, integrated with waterfall	Vary based on scope 4-6 weeks 6-10 members
NASA 5	Independent SQA of NASA 4 SW	Large	Yes – Large %	High	Moderate	SCRUM, integrated with waterfall	Vary based on scope 4-6 weeks 6-10 members
NASA 6	SW Development for small agency systems	Small % (Agile)	No	Medium	Low	SCRUM Hybrid with Waterfall	Vary based on scope, Usually Months instead of weeks 3 on team
Industry 1	Aviation Industry, SW systems Development	Very Large	Yes – Large % based on NASA Standards	High	Medium	Tailored with Waterfall SCRUM, LEAN Kanban Agile for planning	Work forces the sprint schedule/length 5-10 members
Industry 2	SW Development for government and non-government	Medium	No	Medium	very High	Hybrids of: SCRUM & XP Scrum and Lean Also use Agile processes in planning and management	2 week sprints, standard sprints help with metrics 8-9 team members some use of asynchronous methodology

Organizations	Type of Organization	Size of SW Systems	SW Safety Critical?	SW Complexity	Agile Experience	Agile Processes	Sprint Team Information
Industry 3	SW Development and Software Assurance for government and non-government	Medium & Large 10 to 500 KSLOCS	Yes Small % based on NASA Standard	Medium & High	High	SCRUM & LEAN, large projects use "water scrum fall" hybrid Agile used in planning	4-6 week sprints
Industry 4	Software Department organization within a large computer company	range from small to large up to 500 KSLOCS	No info available	High	High	100% Pure Agile SCRUM & SAFe	Generally 2 week sprints, but flexible length 8-10 members 7 to 8 teams for large SAFe projects
Industry 5	Software Development in Medical Industry	Generally small, some medium to large	Yes small % based on NASA Standards	Medium	Medium	flexible SCRUM for small projects , Waterfall for large,	2-4 week sprints 8-10 members
Industry 6	Software Development and Software Consultant in Medical Industry	Small Projects 10% use Agile	Yes- 5-10%	Low	Medium	Hybrid SCRUM & Lean	1-2 week sprints 1-26 members project durations 6 weeks to 1 year
Industry 7	Agile coach and consultant within a large software development organization and also external consultant	Small & Very Large	No	High	Very High	100% Pure Agile SCRUM SAFe Lean Scrum of Scrums	2 week sprints length of sprint common across project 8-10 members 1 to 42 teams on project, experience with widely distributed teams
Industry 8	SW Development and consulting company to government and non-government customers	Small and Large	small % based on NASA Standards	High	Very high	Pure and Tailored Agile SAFe Lean Rational and Ruff processes	3-4 week sprints, but flexible 4-8 members, diminishing returns with more on teams

Organizations	Type of Organization	Size of SW Systems	SW Safety Critical	SW Complexity	Agile Experience	Agile Processes	Sprint Team Information
Industry 9	Aviation Industry, SW systems Development (see Industry 10)	Large	Yes – 95%	High	High	SCRUM & SAFe generally waterfall type management	2 week synchronized sprints 6-7 Avg. members About 40 scrum teams, four trains of scrum teams
Industry 10	Software QA Organization in Large Aviation Industry company (see Industry 9)	Small Subset of CSCI's, minimal external interfaces	No	Medium	Medium	SCRUM integrated with Waterfall	2 week sprints
Industry 11	Organization develops SW tools to support research organizations	small 5-25 KSLOCS	Yes – 10%	Medium	very High	SCRUM	1 week sprints 3-4 members
Industry 12	Large Aerospace Company Software Development and Software Assurance	Large	Yes – 100%	High	Very High	100% Agile Organization, do not prescribe to a specific Agile methodology	7-8 people/team 9 teams SW Developers are System Engineers
Consultant 1	Consulting & Coaching	Medium & Large Systems	Yes – Small Percentage	Medium	Very High	Tailored, SCRUM SAFe	2-8 week sprints Varies depending on size of project
Federal Agency 1	Medical Device Systems, assess process & products, does not build software	Medium Systems	Yes, but don't develop them	Medium	Low	Minimal, None specified	NA
University 1	Member of consortium that assesses Agile processes usage	small to mid-size is focus of assessments	No	medium	high	focus on use of Scrum	NA

Organizations	Documentation Approach	Tools Used	SQA Involvement	Safety Involvement	Reliability Involvement
NASA 1	Traditional Documentation	DOORS, CRADLE, Rhapsody, INSCOPE, and JIRA	Integrated part of the organization not part of the sprints	Yes, priority focus relative sprints, and 100% of project milestones	No info provided
NASA 2	SharePoint	SharePoint	Not integrated as part of the team	Separate Safety group from a systems perspective	No info provided
NASA 3	Traditional Documentation	SPORT (Surveillance Problem Observation Reporting Tool)	Integrated as part of the team	Yes, priority focus relative sprints, and 100% of project milestones	Project FMEA and FDIR
NASA 4	Traditional Documentation HTML, DOORS	SBM, DOORS, Code Collaborator, Maturation Metric	Co-located, participate in sprint reviews not sprints, independent reporting chain	Integrated approach and IV&V	No info provided
NASA 5	Traditional Documentation HTML, DOORS	SBM, DOORS, Code Collaborator, Maturation Metric	Co-located, participate in sprint reviews not sprints, independent reporting chain, Same SA processes for waterfall and Agile	Independent, Integrated approach and IV&V, uses hazard analysis	Integrated continuous approach with metrics
NASA 6	No specific information	Rational Tool	Early stages of doing SQA	No Info provided	No Info Provided
Industry 1	Traditional, Independent of method, CMMI Methodology	Simulations, Paper Demos, Table Top Demos, DOORS	Independent, Yes, process focused not truly integrated, checklists, corrective actions	Independent, Separate, Yes, Safety is handle at the system level	NA
Industry 2	SharePoint, MS Products, CMMI Methodology, Visio or Power Point for Design	Microsoft Team Foundation Server (TFS) SCRUM templates, & Automated tools, quality defects, faults	Yes, integrated as part of the sprint team's focus both on testing and process. Industrial engineers assigned to teams to ensure process compliance.	Do not have independent team for safety. Goal to provide system that meets customer needs and be error free.	Metrics, Error Logs, Usage parameters, no need to build in fault tolerance for the types of systems being built. Use Microsoft Code Analysis to determine reliability concerns.

Organizations	Documentation Approach	Tools Used	SQA Involvement	Safety Involvement	Reliability Involvement
Industry 3	Traditional, JIRA	JIRA, Cocomo II, SVN, AXURE, SKYPE, Sonar Cube, Confluence, Automated Testing Tools	2 people integrated as part of all the teams	No Info provided	No info provided
Industry 4	Rational tool set, Doors, RTC, and RQM, document architecture, not as well with design and code	Rational tool set, Jenkins open source tool. Urban tool Code scanner Code coverage Dev Ops model	No independent QA organization, QA is owned by everyone on the team	No independent safety, Safety part of the team	No independent reliability, part of the team
Industry 5	Yes, mostly traditional since many of projects require audits	JIRA, Trello, Daptiv, CM Tool	Not independent reporting, but trained SQA personnel are part of the team	No, Agile works best with non-safety critical systems	Qualitatively tied to the test effort
Industry 6	Most requirements documentation captured towards end of project	Microsoft Project, Windchill	Generally independent, but integral part of the Team	Yes, integral part of the team	Yes, covered via the validation testing No fault tree or FMEA.
Industry 7	Rational tool set, Doors, RTC, and RQM Level of Documentation of requirements driven by need of customer, audits, etc.	Rational tool set, Jenkins open source tool. Urban tool Code scanner Code coverage Dev Ops model Team decision	Yes, integrated part of the team POC said SQA was part of the concept of "all for one, one for all" Independent SQA organization diminishes this concept. If project needs independent QA, figure out how to integrate into team process.	No independent safety organization, Safety part of the team	Part of the team Reliability required by the product owner is built into the system by the team's process.
Industry 8	DocRocket MapRocket GeoWeb Solution JAVA Docs and JIRA	Code Sonar Virtualization tools Automated code testing Code Coverage	Independent, but integral part of the Team activities and processes	No, Safety part of the team	No, part of the team

Organizations	Documentation Approach	Tools Used	SQA Involvement	Safety Involvement	Reliability Involvement
Industry 9	Similar to traditional waterfall	JIRA, DOORS, DevOps, Crucible, SQE link tool, Code Collaborator, Frame Work, Trace	Independent, but not integral part of team due to lack of SQA resources, do work with team, but cannot cover many of the processes and activities	Independent, but resource level prohibits extensive involvement in Agile processes, mostly depends on building in safety.	Independent, but resource level prohibits extensive involvement in Agile processes, mostly depends on building in reliability.
Industry 10	Similar to traditional waterfall	Prototyping, rapid and parallel prototyping, backlog tool used Excel, Unit test, code review	Independent, Yes, but not integral part of the development teams	No safety critical SW was part of the Agile Development	SW reliability is covered in the Agile project at the unit test level.
Industry 11	GoogleDocs, Confluence, JIRA, JAVA Docs	DevOps, Jenkins, SonarQube	No dedicated staff but it is part of the process independent QA performed if customer pays for	Yes, it is accounted for via the Agile process and built into stories.	Yes, as part of the Agile process and best practices No formal hazard analyses.
Industry 12	Ticket Process, & SW Records, Doxygen & Wiki Systems	Trac SVN JIRA Git/Stash Code collaborator Doxygen Klocwork	Continuously thru ticketing process	Yes, it's part of the process	Yes, Testing and data reviews
Consultant 1	Many are sane as waterfall	none referenced	Varied by project, collaboration environment is key	Safety is part of the process Hazard Analysis	NA
FDA 1	Traditional	none referenced	Don't look at code unless recall problems	Assess proof of safety	Asses proof of reliability
University 1	Varies with projects	JIRA SharePoint Wikis Models	No info provided.	No Info provided	Relies on process building in reliability.

Appendix H: Tools Mentioned During Interviews

		N1	N2	N3	N4	N5	I1	I3	I5	I9	I12
Size of Projects	Large	X	X	X	X	X	X	X	X	X	X
	Safety Critical	X		X	X	X	X		X	X	X
Tools Mentioned in Interviews	Doors	X		X	X	X	X			X	
	CRADLE	X									
	Windchill										
	Rapsody	X									
	Trace									X	
	Crucible									X	
	Frame Work									X	
	INSCOPE	X									
	JIRA	X		X				X	X	X	
	Sharepoint		X								
	Google Docs										
	Java Docs										
	Doxygen										X
	Vision Powerpoint										
	Virtualization tools										
	Code Coverage										
	Dev Ops model										
	DocRocket, MapRocket										
	Sport Database			X							
	Serena Business Mgmt.				X	X					
	Maturation Metric				X	X					
	Code Collaborator				X	X				X	X
	Automted test tools					X		X		X	
	Team Foundation Server										
	Trac										X
	Git/Stash										X
	Web Impact										
	GeoWeb Solution										
	Microsoft Code Analysis										
	Sonar Qube							X			

		N1	N2	N3	N4	N5	I1	I3	I5	I9	I12
Size of Projects	Large	X	X	X	X	X	X	X	X	X	X
	Safety Critical	X		X	X	X	X		X	X	X
	Source Version Number							X			X
	Configuration Mgmt Tool								X		
	Axure							X			
	Confluence							X			X
	Jenkins Open Source Tool										
	Rational tools(RTC,RQM)										
	SQE Link tool									X	
	Klocwork										X
	Code Sonar										
	Code Scanner										
	Skype							X			

[No information from Fed1, C1]

		I2	Fed1	C1	N6	I4	I5	I6	I7	I8	I10	I11	U1
Size of Projects	Medium	X	X	X									
	Small				X	X	X	X	X	X	X	X	X
	Safety Critical		X*	X*		X*	X						X*
Tools Mentioned in Interviews	Doors					X							
	CRADLE												
	Windchill							X					
	Rapsody												
	Trace												
	Crucible												
	Frame Work												
	INSCOPE												
	JIRA						X			X		X	X

		I2	Fed1	C1	N6	I4	I5	I6	I7	I8	I10	I11	U1
Size of Projects	Medium	X	X	X									
	Small				X	X	X	X	X	X	X	X	X
	Safety Critical		X*	X*		X*	X						X*
	Sharepoint	X											X
	Google Docs											X	
	Java Docs									X		X	
	Doxygen												
	Vision Powerpoint	X											
	Virtualization Tools									X			
	Code Coverage									X			
	Dev Ops Model												
	DocRocket, MapRocket									X		X	
	Sport Database												
	Serena Business Mgmt.	X											
	Maturation Metric	X											
	Code Collaborator												
	Automted test tools	X							X	X			
	Team Foundation Server												
	Trac												
	Git/Stash												
	Web Impact												
	GeoWeb Solution									X			
	Microsoft Code Analysis												
	Sonar Qube											X	
	Source Version Number												
	Configuration Mgmt Tool						X						
	Trello						X						
	Daptiv						X						
	Axure												
	Confluence											X	X

		I2	Fed1	C1	N6	I4	I5	I6	I7	I8	I10	I11	U1
Size of Projects	Medium	X	X	X									
	Small				X	X	X	X	X	X	X	X	X
	Safety Critical		X*	X*		X*	X						X*
	Rational tools(RTC,RQM)				X	X			X				
	Jenkins Open Source Tool					X			X			X	
	Urban Tool								X				
	SQE Link tool												
	Klocwork												X
	Code Sonar									X			
	Code Scanner					X			X		X		
	Skype												

* Few projects safety critical
** Most of team previously trained
Fed1, C1, I7 didn't own develop software

APPENDIX I: BASICS for AGILE SCRUM

The main take away for most Agile processes lies in the sprint, a single pass through a portion of the final product, but a working portion.

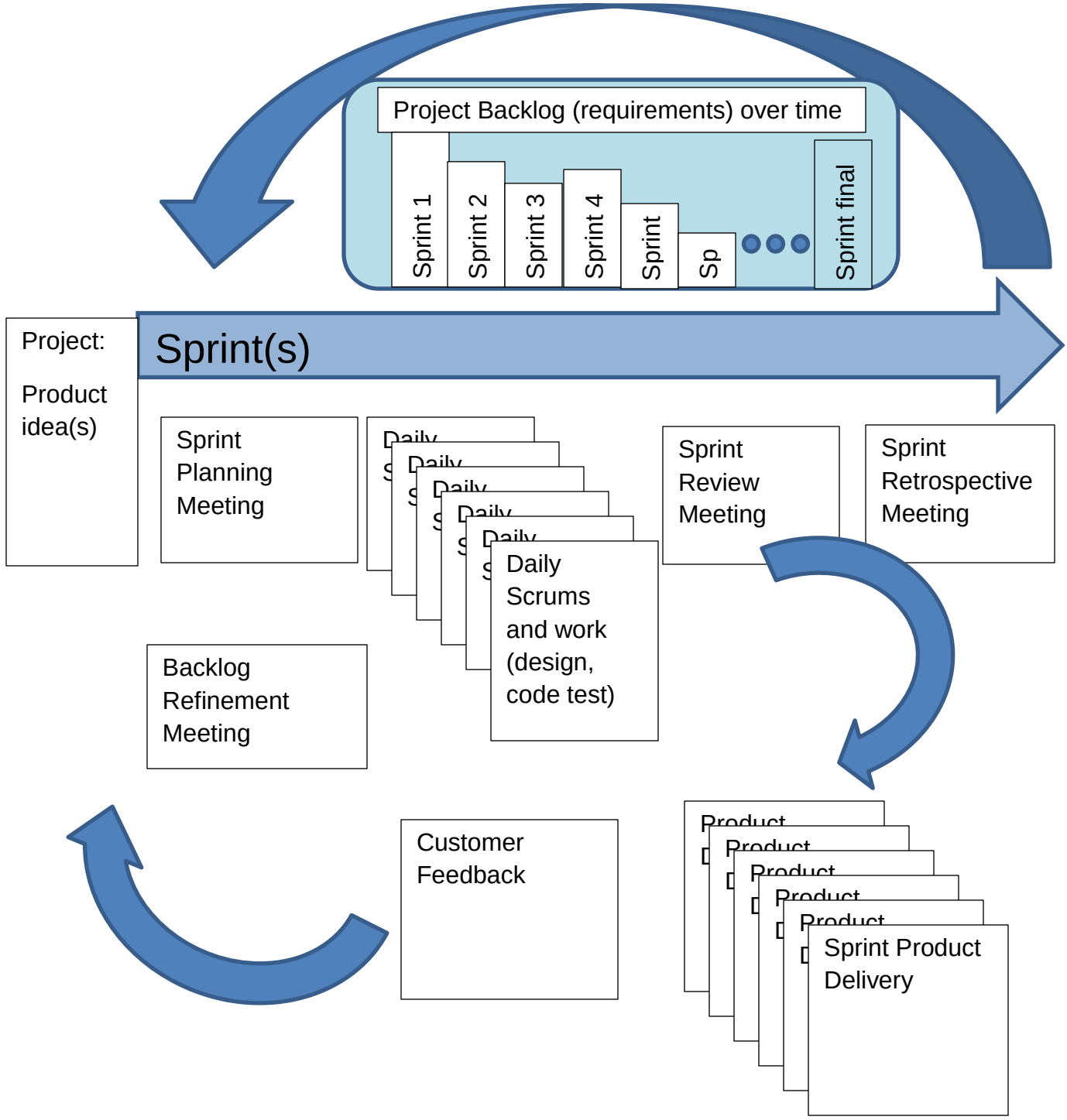
Both the Product and each sprint has a backlog, a list of items that are to be completed. During the Sprint Planning Meeting, each sprint takes one or more items from the product backlog, refines them and details them for the sprint, usually making stories and tasks to be accomplished. Tasks are tracked as: tasks not started, tasks in progress and tasks completed. During a sprint, all committed tasks are to be completed and tested individually and as a whole. The first Sprint Planning Meeting is the time the team determines how they will operate, the length of the sprint, any special roles or interface needs, the definition of "DONE", and general operating rules, metrics, and norms.

A definition of "DONE" is usually set by the sprint team at the very first Sprint Planning Meeting. "Done" is determined at both the sprint and product level and most of the time means the sprint product is complete, tested, and working. It can mean more, such as additional analyses is completed, integration with other sprints is completed, etc. Often this includes any write up is complete and posted.

Daily Scrums are short stand up meetings where all participants (even the customers for some) listen to the developers and testers answer 3 questions: What have you done since yesterday? What are you planning to do today? And what problems/issues are you encountering? If multiple teams are running simultaneously for a single product or interacting products, a team member from each sprint team may attend a daily or weekly "Scrum of Scrums" to keep each team informed of progress and any issues.

The Sprint Review Meetings are where the team shows its work to the product owner and sometimes the customer, end users and other sprint team members. A live demonstration is preferred. It is determined by the Project Owner what is done, the backlogs are updated according to the most critical or needed items. This is where the product is adapted and refined.

Each sprint should end with a Retrospective Meeting. The team determines and discusses what is working and not working and makes adjustments. It can be things like: recommending help for a junior developer, discussing organizational impediments, need for changing the sprint length, to changing how to do story boards. It needs to be a safe place where the team can openly discuss what bothers them as well as what is working. The team resolves or proposes solutions to try for the next sprint, any outside issues are taken up by the Scrum Master to the Product Owner and where needed to try and resolve. This is where the team improves itself and its abilities to produce high quality products.


Project Backlog (requirements) over time
Sprint 1
Sprint 2
Sprint 3
Sprint 4
Sprint
Sp
Sprint final
Project:
Product idea(s)
Sprint(s)
Sprint Planning Meeting
Daily Scrums and work (design, code test)
Sprint Review Meeting
Sprint Retrospective Meeting
Backlog Refinement Meeting
Customer Feedback
Sprint Product Delivery